MODUL DIGITAL

MACHINE LEARNING

Panduan Praktis dan Sistematis
Belajar Machine Learning

- Cocok untuk pemula sampai dengan mahir
- Disertai berbagai studi kasus menarik
- Dibahas secara rinci dan komprehensif
- Menggunakan bahasa yang mudah dipahami
- Menghadirkan teknik spesifik dan terperinci
- Menggunakan Google Colab atau Jupyter
- Menyeimbangkan teori dan praktik
- Langung ke inti, skip yang tak perlu
- Cocok untuk referensi pembelajaran
- Disertai cara instalasi secara jelas

HARTONO

MODUL DIGITAL
MACHINE LEARNING

Panduan Praktis dan Sistematis
Belajar Machine Learning

Hartono

Daftar Isi

Pendahuluan

Tentang Buku Ini
Siapa yang Sebaiknya Membaca Buku Ini?
Pengetahuan Mendasar (*Knowledge Base*)

No	Pengetahuan Mendasar	Deskripsi Singkat dan Beberapa Sumber Referensi Tambahan
1		▪
2		▪
3		▪
4		▪
5		▪
6		▪
7		▪
8		▪
9		▪
10		▪
11		▪

Penyamaan Persepsi Terkait Istilah dan Gaya Selingkung Buku

No	Daftar Istilah	Keterangan

Ikon yang Digunakan pada Buku Ini

	**DID YOU KNOW?** Ikon ini digunakan untuk memberikan informasi berupa fakta atau informasi tambahan terkait bahasan.
	**WHAT EXPERT SAID** Ikon ini digunakan untuk menyampaikan pendapat dari para ahli, baik kutipan langsung maupun tidak langsung.
	**WEB SECURITY TRICK** Ikon ini digunakan untuk menyampaikan trik, menjawab, dan mengatasi masalah keamanan yang menjadi bahasan.
	**IMPORTANT ALERT!** Ikon ini digunakan untuk menyampaikan informasi atau bacaan penting dan vital terkait pembahasan.
	**ADDITIONAL READING** Ikon ini digunakan untuk menyisipkan tautan (dapat berupa QR Code) informasi tambahan yang terkait dengan bahasan.

Aplikasi Pendukung yang Diperlukan

Untuk mengikuti tutorial dan eksperimen di dalam buku ini secara optimal, pembaca harus menyiapkan perangkat lunak (aplikasi) pendukung. Ketika mengawali suatu bahasan atau memulai tutorial, penulis sebenarnya juga akan menjelaskan tahapan instalasi perangkat lunak pendukung, namun akan sangat efektif jika menyiapkannya lebih awal. Dengan begitu, Anda akan lebih siap mengikuti tutorial pada pembahasan selanjutnya. Adapun perangkat lunak yang diperlukan adalah sebagai berikut:

No	Pengetahuan Mendasar	Deskripsi Singkat dan Beberapa Sumber Referensi Tambahan
1	Browser *	Modzila Firefox (Rekomendasi Penulis) https://www.mozilla.org/id/firefox/new/ Google Chrome https://www.google.com/chrome/ Opera https://www.opera.com/
2	Python3 untuk Windows *	Active Python (Paket Python untuk Pemula) https://www.activestate.com/products/python/downloads/ Anaconda Python (Paket Python untuk Pemula++) https://www.anaconda.com/products/individual Python Installer Official (Direkomendasikan untuk ahli) https://www.python.org/downloads/
3	Virtual Box	https://www.virtualbox.org/wiki/Downloads
4	Apache Web Server	Apache Friends XAMPP https://www.apachefriends.org/index.html

No	Pengetahuan Mendasar	Deskripsi Singkat dan Beberapa Sumber Referensi Tambahan
	(opsional)	Bitnami LAMP https://bitnami.com/stack/lamp/installer
5	Windows Subsystem for Linux (opsional)	▪ https://docs.microsoft.com/en-us/windows/wsl/install-win10 ▪ https://ubuntu.com/wsl

Keterangan: * pilih salah satu

Kode Sumber yang Digunakan Lingkungan Uji Coba (Sandbox)

Bab 1. Konsep, Teori, dan Istilah Umum di Machine Learning

DAFTAR PEMBAHASAN PADA BAB INI

1. Pengantar Machine Learning: definisi, sejarah, dan perkembangan.
2. Supervised Learning: prinsip, metode, dan contoh penerapan.
3. Unsupervised Learning: prinsip, metode, dan contoh penerapan.
4. Reinforcement Learning: prinsip, metode, dan contoh penerapan
5. Perbandingan Metode Machine Learning: perbedaan dan kelebihan masing-masing metode.
6. Istilah-istilah umum pada machine learning seperti dataset, feature, isntance, class, label, dan lain-lain

1.1. Pengantar Machine Learning

A. Definisi

Machine Learning adalah cabang dari kecerdasan buatan (AI) yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data dan membuat prediksi atau keputusan berdasarkan data tersebut tanpa diprogram secara eksplisit. Tujuan utama machine learning adalah untuk mengembangkan sistem yang dapat meningkatkan kinerja mereka dari waktu ke waktu melalui pengalaman (data).

B. Sejarah

Pada tahun 1950-an, Alan Turing memperkenalkan konsep mesin belajar dalam makalahnya yang terkenal, "Computing Machinery and Intelligence." Ide dasarnya adalah bahwa komputer dapat belajar dari pengalaman dan membuat keputusan tanpa diprogram secara eksplisit. Pada periode yang sama, Donald Hebb mengusulkan teori Hebbian Learning, yang menjadi dasar bagi pengembangan jaringan saraf buatan. Selanjutnya, di tahun 1960-an hingga 1980-an, Frank Rosenblatt mengembangkan perceptron, model awal dari jaringan saraf buatan, sementara John McCarthy dan Marvin Minsky mengembangkan teori dasar kecerdasan buatan.

Pada tahun 1990-an, terjadi kemajuan signifikan dengan pengembangan algoritma seperti Support Vector Machines (SVM) oleh Vladimir Vapnik dan Alexey Chervonenkis, serta pengenalan algoritma ensemble seperti Random Forest dan Boosting. Dengan ledakan data besar dari internet dan teknologi digital lainnya pada awal 2000-an, bersama dengan revolusi deep learning yang dipopulerkan oleh Geoffrey Hinton, Yann LeCun, dan Yoshua Bengio, machine learning menjadi salah satu bidang paling dinamis dan berpengaruh dalam teknologi saat ini. Dalam

beberapa dekade terakhir, machine learning telah berkembang pesat dan diterapkan luas dalam berbagai bidang seperti kesehatan, keuangan, transportasi, dan teknologi.

Periode	Peristiwa Kunci
1950	Alan Turing memperkenalkan konsep mesin belajar dalam makalah "Computing Machinery and Intelligence".
1950	Donald Hebb mengusulkan teori Hebbian Learning, dasar untuk jaringan saraf buatan.
1960 - 1980	Frank Rosenblatt mengembangkan perceptron, model awal jaringan saraf buatan.
1960 - 1980	John McCarthy dan Marvin Minsky mengembangkan teori dasar kecerdasan buatan.
1990	Vladimir Vapnik dan Alexey Chervonenkis mengembangkan Support Vector Machines (SVM).
1990	Pengenalan algoritma ensemble seperti Random Forest dan Boosting.
2000	Ledakan data besar dari internet dan teknologi digital lainnya.
2000	Revolusi deep learning dengan jaringan saraf dalam, dipopulerkan oleh Geoffrey Hinton, Yann LeCun, dan Yoshua Bengio.

C. Perkembangan

Perkembangan machine learning telah menempuh perjalanan yang luar biasa sejak awal konsepnya pada tahun 1950-an. Dari konsep awal oleh Alan Turing tentang kemungkinan komputer untuk belajar dari pengalaman, hingga pengembangan algoritma dan teknik baru seperti Support Vector Machines (SVM) pada tahun 1990-an, revolusi deep learning pada awal 2000-an, dan munculnya teknologi komputasi seperti GPU dan TPU.

Perkembangan ini didorong oleh beberapa faktor, termasuk ledakan data besar dari internet dan teknologi digital lainnya, serta kemajuan dalam teknologi komputasi yang memungkinkan pemrosesan dan penyimpanan data dalam skala yang lebih besar. Dengan adanya teknik baru seperti reinforcement learning dan transfer learning, serta penerapan praktis dalam berbagai bidang seperti kesehatan, keuangan, transportasi, dan teknologi, machine learning menjadi salah satu bidang yang paling berpengaruh dalam era digital ini.

Aspek	Perkembangan
Teknologi Komputasi	Penggunaan GPU dan TPU untuk mempercepat komputasi. Cloud computing untuk memproses dan menyimpan data dalam skala besar.
Algoritma dan Teknik Baru	Pengembangan reinforcement learning yang memungkinkan agen belajar dari lingkungan. Transfer learning memungkinkan model yang telah dilatih pada satu tugas digunakan pada tugas lain.
Aplikasi Praktis	Kesehatan: Diagnostik medis dan personalisasi pengobatan. Keuangan: Analisis risiko dan prediksi pasar saham. Transportasi: Mobil otonom dan optimasi rute. Teknologi: Pengenalan wajah, asisten virtual, dan rekomendasi konten

1.2. Metode Machine Learning

A. Supervised Learning (Belajar Terbimbing)

Supervised Learning adalah salah satu paradigma dalam machine learning di mana algoritma belajar dari data yang sudah "diberi label". Artinya, setiap contoh data dalam kumpulan data pelatihan memiliki label yang sesuai dengan output yang diharapkan. Algoritma belajar untuk memetakan input ke output berdasarkan contoh yang diberikan. Contoh penerapan supervised learning termasuk klasifikasi dan regresi. Dalam klasifikasi, tujuannya adalah untuk memprediksi label kelas dari data input yang baru. Contoh aplikasi klasifikasi termasuk pengenalan gambar, klasifikasi email spam, dan diagnosis medis. Sedangkan dalam regresi, tujuannya adalah untuk memprediksi nilai numerik dari data input yang baru.

Contoh aplikasi regresi meliputi prediksi harga saham, perkiraan cuaca, dan penilaian properti. Dalam supervised learning, model dilatih menggunakan kumpulan data pelatihan yang berisi pasangan input-output. Algoritma belajar kemudian menggunakan kumpulan data ini untuk menyesuaikan parameter model sehingga dapat membuat prediksi yang akurat untuk data yang belum pernah dilihat sebelumnya.

B. Unsupervised Learning (Belajar Tak Terbimbing)

Unsupervised Learning adalah paradigma dalam machine learning di mana algoritma belajar dari data yang tidak memiliki label atau informasi yang terstruktur sebelumnya. Tujuannya adalah untuk menemukan pola alami atau struktur tersembunyi dalam data. Metode yang umum digunakan dalam unsupervised learning termasuk clustering, dimensionality reduction, dan association rule learning. Dalam clustering, algoritma mengelompokkan data ke dalam kelompok-kelompok yang serupa berdasarkan karakteristik yang teramati. Contoh penerapan clustering termasuk segmentasi pasar dan pengelompokkan dokumen. Dimensionality

reduction bertujuan untuk mengurangi jumlah fitur dalam data sementara mempertahankan informasi yang relevan.

Contoh penerapan dimensionality reduction adalah analisis komponen utama (PCA) dalam pengolahan citra. Association rule learning digunakan untuk menemukan keterkaitan antara variabel dalam kumpulan data. Contoh penerapan association rule learning adalah analisis keranjang belanja. Dalam unsupervised learning, model diberikan data tanpa label dan diharapkan dapat menemukan pola yang bermanfaat atau struktur tersembunyi dalam data tersebut tanpa bimbingan eksternal. Algoritma unsupervised learning dapat digunakan untuk eksplorasi data, pengelompokkan, pengurangan dimensi, dan pemodelan asosiasi.

C. Reinforcement Learning

Reinforcement Learning (RL) adalah paradigma pembelajaran mesin di mana agen belajar berinteraksi dengan lingkungannya untuk mencapai tujuan tertentu. Prinsip utamanya adalah bahwa agen diberi umpan balik positif atau negatif dalam bentuk hadiah atau hukuman berdasarkan tindakan yang diambilnya, yang memungkinkannya memperbaiki strateginya seiring waktu. Metode utama dalam reinforcement learning termasuk Q-learning, metode Monte Carlo, dan algoritma actor-critic. Dalam Q-learning, agen mempelajari nilai dari setiap tindakan dalam setiap keadaan, sementara metode Monte Carlo menggunakan sampel episod untuk memperkirakan nilai aksi. Algoritma actor-critic menggabungkan elemen pembelajaran kebijakan (actor) dan pembelajaran nilai (critic) untuk meningkatkan kestabilan dan kecepatan pembelajaran.

Contoh penerapan reinforcement learning meliputi pengendalian robot, permainan video, dan optimisasi strategi perdagangan saham. Dalam pengendalian robot, agen belajar untuk mengoptimalkan gerakan berdasarkan umpan balik dari sensor. Dalam permainan video, agen belajar untuk mengambil tindakan yang optimal untuk mencapai tujuan permainan. Dalam strategi perdagangan saham, agen belajar untuk membuat keputusan investasi berdasarkan informasi pasar dan hasil historis.

1.3. Perbedaan Metode Machine Learning

Perbandingan metode machine learning adalah proses membandingkan berbagai pendekatan atau algoritma dalam memecahkan suatu masalah tertentu. Berikut ini adalah contoh perbandingan metode machine learning, mencakup perbedaan dan kelebihan masing-masing:

A. Supervised Learning vs Unsupervised Learning

Supervised learning membutuhkan data yang diberi label, sementara unsupervised learning tidak memerlukan label pada dataa. Supervised learning cocok untuk masalah klasifikasi dan regresi dengan data yang terstruktur, sementara

unsupervised learning cocok untuk pengelompokan dan analisis data yang tidak terstruktur.

B. Reinforcement Learning vs Supervised Learning:

Reinforcement learning adalah tentang belajar dari interaksi dengan lingkungan, sedangkan supervised learning adalah tentang belajar dari data yang sudah diberi label. Reinforcement learning cocok untuk masalah dengan lingkungan yang dinamis dan tidak terstruktur, sementara supervised learning cocok untuk masalah klasifikasi dan regresi dengan data yang diberi label.

Dalam memilih metode machine learning yang tepat, penting untuk mempertimbangkan jenis masalah yang akan dipecahkan, sumber daya yang tersedia, dan karakteristik data yang digunakan. Setiap metode memiliki kelebihan dan kelemahan tertentu yang harus dipertimbangkan dengan cermat.

1.4. Istilah-istilah Umum di Machine Learning

A. Dataset

Dataset adalah kumpulan data yang digunakan dalam berbagai tahap pembuatan model machine learning, seperti pelatihan, pengujian, dan validasi. Dataset umumnya terdiri dari sejumlah besar contoh, atau instance, yang masing-masing berisi fitur, yaitu atribut atau karakteristik data, dan label jika data tersebut bersifat terawasi. Misalnya, dataset bunga iris berisi pengukuran panjang dan lebar dari bagian sepal dan petal tiga jenis bunga iris yang berbeda.

B. Feature

Feature adalah karakteristik atau atribut khusus dari data yang berperan sebagai input untuk model machine learning. Setiap contoh data atau instance dalam dataset biasanya memiliki satu atau lebih fitur yang menjadi variabel penentu dalam proses pembelajaran. Pada dataset bunga iris, fitur dapat berupa panjang sepal, lebar sepal, panjang petal, dan lebar petal yang menjadi penentu dalam membedakan jenis bunga.

C. Instance

Instance mengacu pada setiap contoh individual dalam dataset, yaitu satu baris data yang berisi berbagai nilai fitur dan label jika data bersifat terawasi. Setiap instance menyajikan satu observasi atau entri yang nantinya akan dipelajari oleh model. Sebagai contoh, dalam dataset bunga iris, sebuah instance dapat berisi panjang sepal 5.1, lebar sepal 3.5, panjang petal 1.4, lebar petal 0.2, dan label "Setosa."

D. Label

Label adalah nilai output atau target yang menjadi tujuan prediksi dalam model machine learning yang bersifat terawasi. Dalam supervised learning, label

merupakan data yang dicari atau diestimasikan model berdasarkan input fitur. Contohnya, dalam prediksi jenis bunga iris, label dapat berupa "Setosa," "Versicolor," atau "Virginica," yang masing-masing mewakili kategori atau kelas yang coba diprediksi oleh model.

E. Data Training

Data Training adalah proses di mana model machine learning dilatih menggunakan dataset. Selama proses ini, model akan belajar dari contoh-contoh data dan berupaya meminimalkan kesalahan prediksi melalui algoritma optimasi. Sebagai ilustrasi, data training digunakan untuk melatih model regresi linear dalam memprediksi harga rumah berdasarkan fitur seperti luas rumah atau jumlah kamar tidur.

F. Data Testing

Data Testing adalah langkah evaluasi yang dilakukan pada dataset yang belum pernah dilihat oleh model selama proses pelatihan. Tujuannya adalah untuk menilai kemampuan generalisasi model pada data baru. Misalnya, setelah model prediksi harga rumah dilatih, model ini diuji menggunakan sekumpulan data testing baru untuk mengukur akurasi prediksi yang dihasilkan.

G. Model

Model dalam machine learning adalah representasi matematis dari hubungan antara fitur dan label yang telah dipelajari dari data melalui proses pelatihan. Model ini kemudian digunakan untuk melakukan prediksi atau pengambilan keputusan berdasarkan data input baru. Contoh model mencakup model regresi linear yang memprediksi harga rumah, model klasifikasi yang mengidentifikasi jenis bunga, atau jaringan saraf tiruan yang mengenali gambar.

1.5. Cara Kerja Machine Learning

Machine learning adalah salah satu cabang dari kecerdasan buatan (AI) yang berfokus pada pengembangan sistem yang dapat belajar dari data, mengenali pola, dan membuat keputusan secara otomatis dengan minimal campur tangan manusia. Secara umum, proses kerja machine learning meliputi beberapa tahapan utama.

A. Pengumpulan Data

Tahap pertama adalah pengumpulan data yang relevan dengan permasalahan yang akan diselesaikan. Data ini dapat diperoleh dari berbagai sumber seperti basis data, sensor, atau internet. Misalnya, untuk membangun model prediksi penjualan, data transaksi penjualan akan dikumpulkan.

B. Persiapan Data

Pada tahap ini, data yang telah dikumpulkan diolah agar siap digunakan dalam proses machine learning. Persiapan ini mencakup pembersihan data dari nilai yang hilang, penghapusan duplikasi, serta transformasi data mentah menjadi fitur yang lebih mudah dianalisis. Sebagai contoh, dalam data penjualan, entri yang duplikat dapat dihapus, dan nilai yang hilang dapat diisi sesuai kebutuhan.

C. Pemilihan Model

Langkah ini melibatkan pemilihan algoritma atau model machine learning yang paling sesuai dengan karakteristik data dan tujuan dari permasalahan. Pemilihan model yang tepat sangat penting karena setiap algoritma memiliki keunggulan dan kelemahannya masing-masing. Misalnya, model regresi linear dapat dipilih untuk prediksi penjualan jika hubungan antara fitur dan target bersifat linier.

D. Pelatihan Model

Proses pelatihan adalah tahap di mana model dilatih menggunakan data pelatihan untuk meminimalkan kesalahan prediksi. Selama pelatihan, algoritma akan menyetel bobot atau parameter model berdasarkan data yang diberikan. Contohnya, model regresi linear dapat dilatih menggunakan data penjualan historis untuk memprediksi penjualan di masa mendatang.

E. Evaluasi Model

Tahap evaluasi dilakukan untuk menilai kinerja model dengan menggunakan data yang belum pernah dilihat sebelumnya oleh model (testing data). Evaluasi ini penting untuk mengukur seberapa baik model dapat melakukan prediksi pada data baru. Beberapa metrik kinerja yang digunakan di antaranya adalah akurasi, presisi, recall, F1-score, dan ROC-AUC. Misalnya, model prediksi penjualan diuji pada data terbaru untuk melihat ketepatan hasil prediksinya.

F. Penyetelan dan Optimasi Model

Tahap ini melibatkan pengoptimalan hyperparameter model untuk meningkatkan performa. Teknik yang sering digunakan adalah cross-validation, grid search, atau random search untuk menemukan kombinasi parameter terbaik. Misalnya, dalam model regresi, penyetelan parameter seperti learning rate atau regularization dapat membantu meningkatkan hasil prediksi.

G. Prediksi atau Inferensi

Setelah model dilatih dan dievaluasi, model ini kemudian dapat digunakan untuk membuat prediksi atau inferensi pada data baru. Sebagai contoh, model prediksi penjualan yang sudah terlatih digunakan untuk memperkirakan penjualan di bulan mendatang berdasarkan data terbaru.

H. Deployment

Tahap terakhir adalah deployment, di mana model yang sudah siap diterapkan ke dalam lingkungan produksi untuk digunakan oleh pengguna akhir. Pada tahap ini, model diintegrasikan ke dalam sistem atau aplikasi yang akan memanfaatkan model tersebut. Misalnya, model prediksi penjualan dapat diintegrasikan ke dalam sistem manajemen inventaris perusahaan untuk mendukung pengambilan keputusan terkait stok barang.

1.6. Contoh Ilustarsi Cara Kerja Machine Learning

1. Kumpulan Data: Mengumpulkan data tentang penjualan produk dari toko online selama setahun.
2. Persiapan Data
    2.1. Membersihkan data dengan menghapus entri yang duplikat dan mengisi nilai yang hilang.
    2.2. Mengubah data tanggal ke dalam format yang dapat dianalisis seperti hari dalam minggu, bulan, dan musim.
3. Pemilihan Model: Memilih model regresi linear untuk memprediksi jumlah penjualan berdasarkan fitur seperti harga, jumlah ulasan, dan kategori produk
4. Pelatihan Model: Melatih model regresi dengan data penjualan dari Januari hingga Oktober.
5. Evaluasi Model: Menguji model dengan data penjualan dari November dan Desember untuk melihat seberapa baik model tersebut memprediksi penjualan.
6. Penyetelan dan Optimasi Model: Menggunakan teknik cross-validation untuk menemukan kombinasi parameter yang memberikan kinerja terbaik.
7. Prediksi atau Inferensi: Menggunakan model yang sudah dioptimalkan untuk memprediksi penjualan untuk bulan Januari berikutnya.
8. Deployment: Mengintegrasikan model ke dalam aplikasi manajemen stok untuk membantu pengambilan keputusan oleh manajer toko.

Silakan baca notebook tentang Konsep, Teori, dan Istilah Umum di Machine Learning berikut ini.

Scan disi atau klik gambar di samping untuk mengakses Google Colab pembelajaran

Untuk mendapatkan pemahaman yang lebih baik tentang definisi machine learning, silakan saksikan video berikut ini. Video ini akan memberikan gambaran lengkap tentang konsep dasar machine learning, bagaimana sistem ini bekerja, serta penerapannya dalam berbagai bidang. Dengan menyimak video ini, diharapkan

kalian dapat memahami peran penting machine learning dalam analisis data dan pengambilan keputusan secara otomatis.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Dalam video ini, kita akan membahas perbedaan dan persamaan antara Artificial Intelligence (AI) dan Machine Learning (ML). AI adalah konsep yang mencakup seluruh upaya untuk menciptakan mesin yang mampu meniru proses berpikir dan bertindak manusia, seperti pengambilan keputusan, pemahaman bahasa, dan pemecahan masalah. Di sisi lain, Machine Learning adalah salah satu cabang dari AI yang berfokus pada metode agar mesin dapat "belajar" dari data yang tersedia, meningkatkan kinerja dan prediksinya tanpa pemrograman tambahan di setiap tahap. Dengan memahami hubungan ini, Anda akan memperoleh gambaran yang lebih jelas tentang bagaimana AI dan ML bekerja bersama, serta peran pentingnya dalam teknologi modern.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Video ini menjelaskan bagaimana machine learning bekerja sebagai teknologi yang memungkinkan komputer untuk belajar dari data tanpa membutuhkan pemrograman eksplisit. Proses machine learning dimulai dengan pengumpulan dan persiapan data, yang menjadi dasar untuk melatih model. Model kemudian mempelajari pola-pola dalam data tersebut untuk dapat membuat prediksi atau keputusan yang sesuai. Setelah model dilatih, kinerjanya dievaluasi menggunakan data baru guna menilai performanya. Jika diperlukan, model dapat disempurnakan

dan dilatih ulang untuk mencapai hasil yang lebih akurat. Dengan pemahaman ini, Anda akan dapat melihat bagaimana machine learning mampu mengotomatisasi berbagai tugas dan meningkatkan efisiensi dalam proses pengambilan keputusan berbasis data.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 2. Machine Learning Menggunakan Python

Hal-hal yang diungkapkan pada Bab ini:

1. Conditional statement.
2. Input dan output.
3. Looping.
4. Deklarasi dan operasi tipe data.
5. Struktur data.
6. Pembacaan dataset.
7. Visualisasi data sederhana.

2.1. Pengantar Python

Python adalah bahasa pemrograman yang mudah dipelajari dan banyak digunakan di berbagai bidang, termasuk pengembangan web, analisis data, dan pembelajaran mesin. Python memiliki sintaks yang bersih dan sederhana, serta komunitas yang besar dan aktif. Python adalah salah satu bahasa pemrograman paling populer dan serbaguna yang digunakan di berbagai bidang. Berikut adalah beberapa alasan mengapa Python sangat disukai oleh banyak orang, terutama di komunitas pembelajaran mesin (machine learning):

a. Sintaks yang Sederhana dan Mudah Dipelajari.
b. Komunitas yang Besar dan Aktif.
c. Ekosistem Pustaka yang Luas.
d. Interoperabilitas yang Baik.
e. Dukungan untuk Paradigma Pemrograman Berorientasi Objek dan Fungsional.
f. Produktivitas dan Pengembangan yang Cepat.
g. Dukungan Multiplatform.
h. Popularitas dan Penggunaan di Industri.

Kelebihan paling signifikan pada Bahasa Pemrograman Python adalah pada sintaknya yang sederhana. Kesederhanannya ini membuat Python lebih mudah dipahami, terutama oleh pemula. Dengan kemudahan tersebut, komunitas Python dapat berkembang pesat.Selain itu, Python juga dapat dimanfaatkan untuk berbagai kebutuhan antara lain:

a. Analisis data dan statistik seperti analisis data keuangan, tren penjualan, dan pemrosesan data sensor untuk IoT.
b. Machine Learning dan Artificial Intelegence seperti pengenalan gambar, analisis sentimen, dan prediksi penyakit.
c. Pengembangan Website seperti Django dan Flask yang memungkinkan kita membangun aplikasi web hingga skala besar.

d. Automasi dan Skrip yang dapat digunakan untuk mengotomasi tugas berulang dan membuat skrip untuk berbagai kebutuhan.
e. Pengembangan Game sederhana menggunakan Pygame.
f. Visualisasi data yang dapat bekerja dengan data besar menggunakan pustaka Dask, PySpark, dan Hadoop.
g. Pemrosesan Teks dan Bahasa Alami (NLP) dengan pustakan NLTK, SpaCy, TextBlob.
h. Robotik dan Otomasi Industri yang dapat digunakan dalam robotika dan otomasi industri.
i. Internet of Things (IoT)
j. Keamanan Siber yang dapat digunakan untuk pengujian penetrasi seperti analisis log, alat pengujian keamanan dan lain-lain

A. Instalasi Python

Untuk menggunakan Python, kita harus menginstalasi Python di perangkat, baik laptop atau komputer. Sebenarnya, terdapat beberapa alternatif dalam menginstalasi Python. Namun, secara garis besar, terdapat 3 cara yang direkomendasikan untuk menginstalasi Python yaitu (1) instalasi dari situs resmi Python; (2) instalasi menggunakan paket distribusi Anaconda; dan (3) instalasi menggunakan paket distribusi Active Python. Tiap alternatif tentunya memiliki kelebihan dan kelemahan masing-masing. Oleh karena itu, kita bebas memilih alternatif yang paling sesuai dengan kebutuhan kita

```
from IPython.display import IFrame
IFrame("https://www.youtube.com/embed/micqBtVH4hQ", width=650, height=350)
```

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

a. Sistem Operasi Windows

1. Kunjungi situs resmi Python dan unduh installer Python terbaru (misalnya, python-3.x.x.exe).
2. Jalankan Installer: Buka file installer yang telah diunduh dan pastikan Anda mencentang opsi "Add Python to PATH". Ini akan memudahkan Anda menjalankan Python dari command prompt.

3. Install Python: Klik "Install Now" untuk memulai proses instalasi. Tunggu hingga proses instalasi selesai.
4. Verifikasi Instalasi: Buka command prompt dan jalankan perintah berikut untuk memverifikasi instalasi (python --version)

b. Sistem Operasi Mac

1. Unduh Installer: Kunjungi situs resmi Python dan unduh installer Python untuk macOS (misalnya, python-3.x.x-macosx10.x.pkg).
2. Jalankan Installer: Buka file installer yang telah diunduh dan ikuti petunjuk instalasi.
3. Verifikasi Instalasi: Buka terminal dan jalankan perintah berikut untuk memverifikasi instalasi (python --version).

c. Sistem Operasi Linux

Python umumnya sudah terinstal secara default di banyak distribusi Linux. Namun, jika Anda perlu menginstal atau memperbaruinya, berikut adalah langkah-langkah untuk beberapa distribusi Linux yang paling umum digunakan.

Ubuntu/Debian: ```bash sudo apt update sudo apt install python3

Fedora ```bash sudo dnf install python3

Arch Linux

```
sudo pacman -S python
```

B. Menjalankan Script Python

Untuk mulai menggunakan Python, kita harus telah berhasil menginstalasi Python. Setelah diinstal, kita dapat mencoba Python dengan cara berikut ini:

Membuat Script Python

Buat skrip Python menggunakan editor teks favorit Anda dan simpan dengan ekstensi .py. Misalnya, buat skrip sederhana bernama hello.py:

```
# hello.py
print("Hello, World!")
```

Output:

```
Hello, World!
```

b. Gunakan Terminal atau Command Prompt

Buka terminal atau command prompt dan navigasi ke direktori yang berisi script hello.py. Gunakan perintah cd untuk berpindah ke direktori di mana skrip Python Anda disimpan. Misalnya, jika skrip disimpan di direktori ~/projects. Untuk menjalankan script Python, ketik perintah berikut ini di terminal. ```bash python hello.py

c. Menjalankan Script dengan Virtual Environment

Jika Anda menggunakan virtual environment, pastikan Anda sudah mengaktifkan virtual environment sebelum menjalankan skrip. Misalnya:

1. Membuat Virtual Environment

```
python3 -m venv myenv
```

2. Mengaktifkan Virtual Environment

```
source myenv/bin/activate
```

3. Menonaktifkan Virtual Environment Setelah Selesai

```
```bash deactivate
```

C. Dasar Penggunaan Python

Setelah menginstal Python, mari kita pelajari beberapa konsep dasar yang akan membantu Anda memahami cara kerja Python. Contoh sederhana dari program Python:

```
print("Hello, world!")
```

a. Variabel dan Tipe Data

Python adalah bahasa yang dynamically typed, artinya Anda tidak perlu mendeklarasikan tipe variabel secara eksplisit.

```
# Deklarasi variabel
x = 10          # Integer
y = 3.14        # Float
nama = "Alice"  # String

print(x, y, nama)
```

Output:

```
10 3.14 Alice
```
```

b. Operasi Aritmatika

Python mendukung operasi aritmatika dasar seperti penjumlahan, pengurangan, perkalian, dan pembagian.

```
a = 5
b = 2

print(a + b)  # Penjumlahan
print(a - b)  # Pengurangan
print(a * b)  # Perkalian
print(a / b)  # Pembagian
print(a ** b) # Perpangkatan
```

Output:

```
7
3
10
2.5
25
```

c. Conditional Statement

Conditional statement digunakan untuk membuat keputusan berdasarkan kondisi tertentu.

```
x = 10

if x > 0:
    print("x adalah bilangan positif")
elif x == 0:
    print("x adalah nol")
else:
    print("x adalah bilangan negatif")
```

d. Looping (For dan While)

Looping digunakan untuk menjalankan satu atau beberapa baris kode berulang kali.

```
# Loooping For
for i in range(5):
    print(i)

my_list = [1, 2, 3, 4, 5]
for elemen in my_list:
    print(elemen)
```

Output:

```
0
1
2
3
4
1
2
3
4
5
```

```
i = 0
while i < 5:
    print(i)
    i += 1
```

Output:

```
0
1
2
3
4
```

e. Fungsi

Fungsi adalah blok kode yang dapat digunakan kembali untuk menjalankan tugas tertentu. ```python def salam(nama): print("Hello, " + nama) salam("Alice") salam("Bob")

f. Struktur Data

Python mendukung berbagai struktur data seperti list, tuple, dictionary, dan set.

```
# List: List adalah kumpulan elemen yang terurut dan dapat diubah.
my_list = [1, 2, 3, 4, 5]
print(my_list)
print(my_list[0])  # Akses elemen pertama
my_list.append(6)  # Menambahkan elemen ke akhir list
print(my_list)
```

Output:

```
[1, 2, 3, 4, 5]
1
[1, 2, 3, 4, 5, 6]
```

```
# Tuple: Tuple adalah kumpulan elemen yang terurut tetapi tidak dapat diubah.
my_tuple = (1, 2, 3, 4, 5)
```

```
print(my_tuple)
print(my_tuple[0])  # Akses elemen pertama
```

Output:

```
(1, 2, 3, 4, 5)
1
```

```
# Dictionary: Dictionary adalah kumpulan pasangan kunci-nilai yang tidak terurut.
my_dict = {"name": "Alice", "age": 25}
print(my_dict)
print(my_dict["name"])  # Akses nilai dengan kunci
my_dict["age"] = 26     # Mengubah nilai
print(my_dict)
```

Output:

```
{'name': 'Alice', 'age': 25}
Alice
{'name': 'Alice', 'age': 26}
```

```
# Set: Set adalah kumpulan elemen unik yang tidak terurut.
my_set = {1, 2, 3, 4, 5}
print(my_set)
my_set.add(6)  # Menambahkan elemen
print(my_set)
```

Output:

```
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5, 6}
```

2.2. Conditional Statement

Pernyataan kondisional dalam Python memungkinkan Anda untuk mengeksekusi potongan kode tertentu berdasarkan apakah suatu kondisi benar atau salah. Ini sangat penting dalam mengontrol alur program. Jenis utama pernyataan kondisional dalam Python adalah if, elif, dan else.

A. Pernyataan if

Pernyataan if digunakan untuk menguji sebuah kondisi. Jika kondisi tersebut bernilai True (benar), maka blok kode di dalam pernyataan if akan dijalankan. Dalam contoh berikut ini, karena x bernilai 10, yang lebih besar dari 5, maka pesan "x lebih besar dari 5" akan dicetak. Contoh sederhana:

```
x = 10
if x > 5:
```

```
    print("x lebih besar dari 5")
```

Output:

```
x lebih besar dari 5
```

B. Pernyataan elif

Pernyataan elif (singkatan dari "else if") digunakan untuk menguji beberapa kondisi tambahan jika kondisi sebelumnya dalam pernyataan if adalah False. Anda dapat memiliki beberapa pernyataan elif. Dalam contoh berikut ini, karena x bernilai 10, yang lebih besar dari 5, maka pesan "x lebih besar dari 5" akan dicetak. Contoh:

```
x = 10
if x > 15:
    print("x lebih besar dari 15")
elif x > 5:
    print("x lebih besar dari 5 tetapi kurang dari atau sama dengan 15")
else:
    print("x kurang dari atau sama dengan 5")
```

Output:

```
x lebih besar dari 5 tetapi kurang dari atau sama dengan 15
```

C. Pernyataan else

Pernyataan else digunakan untuk menangani semua kasus yang tidak ditangani oleh pernyataan if dan elif. Blok kode di dalam else akan dijalankan jika semua kondisi sebelumnya adalah False. Pada contoh berikut ini, karena x bernilai 3, yang tidak memenuhi kondisi if maupun elif, maka blok else akan dijalankan dan pesan "x kurang dari atau sama dengan 5" akan dicetak. Contoh:

```
x = 3
if x > 15:
    print("x lebih besar dari 15")
elif x > 5:
    print("x lebih besar dari 5 tetapi kurang dari atau sama dengan 15")
else:
    print("x kurang dari atau sama dengan 5")
```

Output:

```
x kurang dari atau sama dengan 5
```

2.3. Input Output (masukan luaran)

Dalam Python, input dan output adalah cara untuk menerima data dari pengguna dan menampilkan data kepada pengguna. Mari kita bahas masing-masing dengan contoh sederhana.

A. Input (Masukan)

Untuk menerima input dari pengguna, Python menggunakan fungsi input(). Fungsi ini membaca sebuah string yang dimasukkan oleh pengguna dari keyboard. Contohnya

```
nama = input("Masukkan nama Anda: ")
print("Halo, " + nama + "!")
```

Output:

```
Masukkan nama Anda: Rama Apriando
Halo, Rama Apriando!
```

Dalam contoh ini:

- input("Masukkan nama Anda: ") akan menampilkan pesan "Masukkan nama Anda: " dan menunggu pengguna untuk memasukkan sebuah string.
- Nilai yang dimasukkan oleh pengguna disimpan dalam variabel nama.
- print("Halo, " + nama + "!") akan mencetak pesan yang menggabungkan string "Halo, " dengan nilai dari variabel nama dan tanda seru.

B. Output (Luaran)

Untuk menampilkan output kepada pengguna, Python menggunakan fungsi print(). Fungsi ini dapat mencetak teks, angka, dan hasil ekspresi lainnya ke layar. Contohnya:

```
nama = "Budi"
umur = 20
print("Nama saya " + nama + " dan saya berumur " + str(umur) + " tahun.")
```

Output:

```
Nama saya Budi dan saya berumur 20 tahun.
```

Dalam contoh ini:

- Variabel nama dan umur masing-masing menyimpan nilai "Budi" dan 20.

- Fungsi print() mencetak teks yang menggabungkan string literal dengan nilai variabel.
- Fungsi str(umur) mengkonversi nilai umur dari integer ke string agar dapat digabungkan dengan string lainnya.

C. Penanganan Tipe Data

Secara default, fungsi input() mengembalikan nilai sebagai string. Jika Anda perlu bekerja dengan tipe data lain seperti integer atau float, Anda harus mengonversi input tersebut menggunakan fungsi int() atau float(). Contohnya:

```
umur = int(input("Masukkan umur Anda: "))
tinggi = float(input("Masukkan tinggi Anda dalam meter: "))
print("Anda berumur " + str(umur) + " tahun dan tinggi Anda " + str(tinggi) + "
meter.")
```

Output:

```
Masukkan umur Anda: 19
Masukkan tinggi Anda dalam meter: 167
Anda berumur 19 tahun dan tinggi Anda 167.0 meter.
```

Penjelasan:

- int(input("Masukkan umur Anda: ")) mengonversi input menjadi integer.
- float(input("Masukkan tinggi Anda dalam meter: ")) mengonversi input menjadi float.
- Fungsi str() digunakan untuk mengonversi nilai integer dan float kembali ke string untuk dicetak.

2.4. Deklarasi dan Operasi Tipe Data dan Struktur Data

Dalam Python, terdapat berbagai tipe data dan struktur data yang dapat digunakan untuk menyimpan dan memanipulasi data. Berikut adalah penjelasan mengenai deklarasi dan operasi yang dapat dilakukan dengan tipe data dan struktur data tersebut. Contoh deklarasi:

```
x = 5
y = 7

nama_lengkap = 'Hartono'
data = [1, 2, 3, 4, 5]
```

A. Tipe Data Integer

Integer adalah tipe data bilangan bulat, misalnya sebagai berikut:

```python x = 10

Pada tipe data integer, kita dapat melakukan operasi aritmatematika sebagai berikut:

```
x = 10
y = 3

# Operasi aritmatika
penjumlahan = x + y   # 13
pengurangan = x - y   # 7
perkalian = x * y   # 30
pembagian = x / y   # 3.333...
pembagian_bulat = x // y # 3
sisa_bagi = x % y   # 1
pangkat = x ** y   # 1000
```

B. Tipe Data Float

Berbeda dengan integer, Float adalah tipe data untuk bilangan desimal. Contohnya:

```
pi = 3.14
```

```
x = 10.5
y = 2.5

penjumlahan = x + y    # 13.0
pengurangan = x - y    # 8.0
perkalian = x * y      # 26.25
pembagian = x / y      # 4.2
```

C. Tipe Data String

String adalah tipe data untuk teks atau karakter. Contohnya:

```
nama_depan = "Alice"
nama_belakang = "Smith"

# Penggabungan string
nama_lengkap = nama_depan + " " + nama_belakang  # "Alice Smith"

# Pengulangan string
ulang = nama_depan * 3  # "AliceAliceAlice"

# Mengambil karakter
karakter_pertama = nama_depan[0]  # "A"
karakter_terakhir = nama_depan[-1] # "e"

# Panjang string
panjang = len(nama_depan)  # 5
```

D. Tipe Data Boolean
```

Boolean adalah tipe data yang hanya memiliki dua nilai: True dan False. Contohnya:

```
a = True
b = False

# Operasi logika
and_operation = a and b  # False
or_operation = a or b    # True
not_operation = not a    # False
```

E. Struktur Data List

List adalah struktur data yang dapat menyimpan berbagai tipe data dan bersifat mutable (dapat diubah). Contohnya:

```
angka = [1, 2, 3, 4, 5]
# Menambah elemen
angka.append(6)  # [1, 2, 3, 4, 5, 6]

# Mengakses elemen
elemen_pertama = angka[0]  # 1
elemen_terakhir = angka[-1]  # 6

# Mengubah elemen
angka[0] = 10  # [10, 2, 3, 4, 5, 6]

# Menghapus elemen
angka.remove(10)  # [2, 3, 4, 5, 6]
elemen_terakhir = angka.pop()  # 6, angka menjadi [2, 3, 4, 5]

# Panjang list
panjang = len(angka)  # 4
```

F. Struktur Data Tupple

Tuple adalah struktur data yang mirip dengan list tetapi bersifat immutable (tidak dapat diubah). Contohnya:

```
angka = (1, 2, 3, 4, 5)
# Mengakses elemen
elemen_pertama = angka[0]  # 1
elemen_terakhir = angka[-1]  # 5

# Panjang tuple
panjang = len(angka)  # 5
```

G. Struktur Data Dictionary

Dictionary adalah struktur data yang menyimpan pasangan kunci-nilai (key-value pairs). Contohnya

```
data_mahasiswa = {
    "nama": "Alice",
    "umur": 20,
```

```
    "jurusan": "Informatika"
}
```

```
# Mengakses nilai
nama = data_mahasiswa["nama"]  # "Alice"

# Mengubah nilai
data_mahasiswa["umur"] = 21

# Menambah pasangan kunci-nilai
data_mahasiswa["alamat"] = "Bandung"

# Menghapus pasangan kunci-nilai
del data_mahasiswa["jurusan"]

# Panjang dictionary
panjang = len(data_mahasiswa)  # 3
```

H. Struktur Data Set

Set adalah struktur data yang menyimpan elemen unik dan tidak berurutan.

```
angka = {1, 2, 3, 4, 5}

# Menambah elemen
angka.add(6)

# Menghapus elemen
angka.remove(3)

# Operasi himpunan
set_a = {1, 2, 3}
set_b = {3, 4, 5}

union = set_a | set_b        # {1, 2, 3, 4, 5}
intersection = set_a & set_b # {3}
difference = set_a - set_b   # {1, 2}
```

2.5. Looping for and while

Looping adalah teknik yang memungkinkan Anda untuk menjalankan satu atau lebih pernyataan berulang kali. Python menyediakan dua jenis loop utama: for dan while. Mari kita bahas kedua jenis loop ini secara komprehensif.

A. Looping Menggunakan for

Loop for digunakan untuk mengulangi sebuah blok kode untuk setiap elemen dalam sebuah urutan (seperti list, tuple, string, atau range).

```
for elemen in urutan:
    # blok kode yang akan diulang
```

Pada contoh berikut ini, loop for akan mengulang blok kode print(num) untuk setiap elemen dalam list angka. Artinya jumlah pengulangna bergantung pada jumlah elemen.

```
angka = [1, 2, 3, 4, 5]

for num in angka:
    print(num)
```

Output:

```
1
2
3
4
5
```

Tidak hanya di dalam list, looping for juga dapat diterapkan pada dictionary, set, dan bahkan string. Berikut ini adalah contoh looping for pada data string.

```
kata = "Python"

for huruf in kata:
    print(huruf)
```

Output:

```
P
y
t
h
o
n
```

Pada beberapa situasi, kita memerlukan index atau urutan untuk mempermudah operasi coding. Untuk memenuhi kebutuhan tersebut, kita dapat menggunakan enumerate. Contohnya:

```
angka = [10, 20, 30, 40]

for indeks, nilai in enumerate(angka):
    print(f"Indeks: {indeks}, Nilai: {nilai}")
```

Output:

```
Indeks: 0, Nilai: 10
Indeks: 1, Nilai: 20
Indeks: 2, Nilai: 30
```

```
Indeks: 3, Nilai: 40
```

B. Looping dengan while

Loop while digunakan untuk mengulangi sebuah blok kode selama suatu kondisi bernilai True. Contoh penggunaan while:

```
while kondisi:
    # blok kode yang akan diulang
```

```
count = 0

while count < 5:
    print(count)
    count += 1
```

Output:

```
0
1
2
3
4
```

Dalam contoh ini, loop while akan terus mengulang blok kode print(count) dan count += 1 selama nilai count kurang dari 5.

Menggunakan break dan continue

Pernyataan break digunakan untuk menghentikan loop sepenuhnya. Contoh break dalam Loop for:

```
for num in range(10):
    if num == 5:
        break
    print(num)
```

Output:

```
0
1
2
3
4
```

Loop ini akan berhenti ketika num bernilai 5, sehingga hanya mencetak angka 0 hingga 4. Contoh break dalam Loop while.

```
count = 0

while count < 10:
    if count == 5:
        break
    print(count)
    count += 1
```

Output:

```
0
1
2
3
4
```

Pernyataan continue digunakan untuk melewati iterasi saat ini dan melanjutkan dengan iterasi berikutnya dalam loop. Contoh continue dalam Loop for:

```
for num in range(10):
    if num % 2 == 0:
        continue
    print(num)
```

Output:

```
1
3
5
7
9
```

Loop ini akan melewati angka genap dan hanya mencetak angka ganjil dari 0 hingga 9. Contoh continue dalam Loop while:

```
count = 0

while count < 10:
    count += 1
    if count % 2 == 0:
        continue
    print(count)
```

Output:

```
1
3
5
7
9
```

2.6. Manipulasi string

Manipulasi string adalah teknik untuk memanipulasi dan mengubah teks dalam program Python. Berikut adalah beberapa operasi umum untuk manipulasi string di Python.

A. Penggabungan String

Anda dapat menggabungkan dua atau lebih string menggunakan operator +.

```
nama_depan = "John"
nama_belakang = "Doe"
nama_lengkap = nama_depan + " " + nama_belakang
print(nama_lengkap)  # Output: John Doe
```

Output:

```
John Doe
```

B. Pengulangan String

Anda dapat mengulangi string menggunakan operator *.

```
salam = "Hai! "
ulang = salam * 3
print(ulang)  # Output: Hai! Hai! Hai!
```

Output:

```
Hai! Hai! Hai!
```

C. Mengakses Karakter dalam String

Anda dapat mengakses karakter individu dalam string menggunakan indeks. Indeks dimulai dari 0.

```
kata = "Python"
huruf_pertama = kata[0]
huruf_terakhir = kata[-1]
print(huruf_pertama)  # Output: P
print(huruf_terakhir)  # Output: n
```

Output:

```
P
n
```

D. Mengiris String (Slicing)

Anda dapat mengambil subset dari sebuah string menggunakan slicing.

```
kata = "Python"
sub_kata = kata[0:3]  # Mengambil karakter dari indeks 0 hingga 2
print(sub_kata)  # Output: Pyt
```

Output:

```
Pyt
```

E. Mengubah Kasus String

Python menyediakan metode untuk mengubah kasus string.

```
kata = "Python Programming"

# Mengubah semua karakter menjadi huruf kecil
lowercase = kata.lower()
print(lowercase)  # Output: python programming

# Mengubah semua karakter menjadi huruf besar
uppercase = kata.upper()
print(uppercase)  # Output: PYTHON PROGRAMMING

# Mengubah huruf pertama dari setiap kata menjadi huruf besar
titlecase = kata.title()
print(titlecase)  # Output: Python Programming

# Mengubah huruf pertama dari string menjadi huruf besar
capitalize = kata.capitalize()
print(capitalize)  # Output: Python programming

# Menukar kasus huruf, huruf besar menjadi kecil dan sebaliknya
swapcase = kata.swapcase()
print(swapcase)  # Output: pYTHON pROGRAMMING
```

Output:

```
python programming
PYTHON PROGRAMMING
Python Programming
Python programming
pYTHON pROGRAMMING
```

F. Menghapus Spasi

Python menyediakan metode untuk menghapus spasi di awal dan akhir string.

```
kata = "   Hello, World!   "
```

```
# Menghapus spasi di awal dan akhir string
stripped = kata.strip()
print(stripped)  # Output: Hello, World!

# Menghapus spasi di awal string
left_stripped = kata.lstrip()
print(left_stripped)  # Output: Hello, World!

# Menghapus spasi di akhir string
right_stripped = kata.rstrip()
print(right_stripped)  # Output:    Hello, World!
```

Output:

```
Hello, World!
Hello, World!
   Hello, World!
```

2.7. Penggunaan modul dan package

Modul adalah file Python yang berisi definisi dan pernyataan Python, seperti fungsi, kelas, dan variabel. Modul memungkinkan Anda untuk mengelompokkan kode terkait ke dalam satu file, sehingga kode tersebut dapat digunakan kembali dalam berbagai bagian dari proyek atau bahkan dalam proyek lain. dapat menggunakan modul yang telah dibuat dengan mengimpor modul tersebut ke dalam file Python lain. Contohnya

```python import nama_modul

Package adalah cara untuk mengelompokkan beberapa modul terkait ke dalam satu direktori. Package memungkinkan Anda untuk mengatur kode dalam hierarki yang lebih kompleks dan terstruktur. Untuk membuat package, buat direktori yang akan berfungsi sebagai package, dan letakkan modul-modul di dalamnya. Selain itu, buat file init.py dalam direktori tersebut (file ini dapat kosong, tetapi harus ada agar Python mengenali direktori sebagai package). Misalnya, buat direktori my_package dengan struktur berikut:

my_package/

- init.py
- module1.py
- module2.py

```
module1.py

def fungsi1():
    return "Ini adalah fungsi1 dari module1"
```
```

```
# module2.py

def fungsi2():
    return "Ini adalah fungsi2 dari module2"
```

Anda dapat mengimpor modul dari package dengan cara berikut:

```
# main.py

from my_package import module1, module2

print(module1.fungsi1())
print(module2.fungsi2())
```

Anda juga dapat menggunakan alias atau mengimpor fungsi tertentu:

```
from my_package.module1 import fungsi1
from my_package.module2 import fungsi2

print(fungsi1())
print(fungsi2())
```

Manfaat Penggunaan Modul dan Package

1. Organisasi Kode

   Modul dan package membantu mengatur kode ke dalam file dan direktori yang lebih terstruktur, membuatnya lebih mudah dipahami dan dikelola.

2. Penggunaan Kembali Kode

   Kode yang telah diorganisir dalam modul dan package dapat dengan mudah digunakan kembali di berbagai bagian proyek atau bahkan di proyek lain.

3. Pemeliharaan Kode

   Dengan modul dan package, pemeliharaan kode menjadi lebih mudah karena setiap bagian kode terpisah dalam file-file kecil yang lebih mudah di-debug dan diuji.

4. Kolaborasi

   Modul dan package mempermudah kolaborasi di antara tim pengembang dengan memisahkan kode ke dalam bagian-bagian yang dapat dikerjakan secara terpisah.

2.8. Excetion handling

Exception handling adalah mekanisme dalam pemrograman yang memungkinkan Anda untuk menangani kesalahan atau kondisi tak terduga yang terjadi selama eksekusi program. Python menyediakan struktur khusus untuk

menangani exceptions sehingga program dapat terus berjalan atau berhenti dengan cara yang terkontrol. Exception adalah kondisi kesalahan yang terjadi selama eksekusi program. Ketika sebuah kesalahan terjadi, Python akan mengangkat (raise) exception yang sesuai. Contoh exception yang umum termasuk ZeroDivisionError, TypeError, ValueError, dan FileNotFoundError.

Blok Try-Except

Blok try-except digunakan untuk menangkap dan menangani exception. Struktur dasar dari blok try-except adalah sebagai berikut:

```
try:
    # kode yang mungkin menyebabkan exception
except ExceptionType:
    # kode untuk menangani exception
```

Misalnya, jika terjadi ZeroDivisionError, pesan "Tidak bisa membagi dengan nol!" akan dicetak.

```
try:
    hasil = 10 / 0
except ZeroDivisionError:
    print("Tidak bisa membagi dengan nol!")
```

Output:

```
Tidak bisa membagi dengan nol!
```

Blok Try-Except-Else

Anda dapat menambahkan blok else untuk menentukan kode yang harus dijalankan jika tidak ada exception yang terjadi.

```
try:
    hasil = 10 / 2
except ZeroDivisionError:
    print("Tidak bisa membagi dengan nol!")
else:
    print("Pembagian berhasil, hasil:", hasil)
```

Output:

```
Pembagian berhasil, hasil: 5.0
```

Blok Try-Finally

Blok finally digunakan untuk menentukan kode yang harus dijalankan terlepas dari apakah exception terjadi atau tidak. Blok finally sering digunakan untuk membersihkan sumber daya seperti file atau koneksi jaringan.

```
try:
    file = open("data.txt", "r")
    data = file.read()
except FileNotFoundError:
    print("File tidak ditemukan.")
finally:
    file.close()
    print("File ditutup.")
```

Menangkap Beberapa Exception

Anda dapat menangkap beberapa jenis exception dengan menggunakan beberapa blok except.

```
try:
    x = int(input("Masukkan angka: "))
    hasil = 10 / x
except ValueError:
    print("Input harus berupa angka.")
except ZeroDivisionError:
    print("Tidak bisa membagi dengan nol.")
```

Output:

```
Masukkan angka: 100
```

Menangkap Semua Exception

Untuk menangkap semua jenis exception, Anda dapat menggunakan except tanpa menyebutkan jenis exception, tetapi ini tidak dianjurkan kecuali dalam kasus tertentu. Contohnya:

```
try:
    hasil = 10 / 0
except:
    print("Terjadi kesalahan.")
```

Output:

```
Terjadi kesalahan.
```

2.9. Fungsi-fungsi dasar

Python memiliki sejumlah fungsi dasar yang sangat berguna dalam pemrograman sehari-hari. Berikut adalah beberapa di antaranya:

```
# len(): Mengembalikan panjang (jumlah elemen) dari suatu objek.

my_list = [1, 2, 3, 4, 5]
print(len(my_list))  # Output: 5
```

Output:

```
<class 'int'>
```

```
# int(), float(), str(): digunakan untuk konversi tipe data.
num_str = "123"
num_int = int(num_str)
print(type(num_int))  # Output: <class 'int'>

num_float = float(num_int)
print(type(num_float))  # Output: <class 'float'>

num_str_again = str(num_float)
print(type(num_str_again))  # Output: <class 'str'>
```

Output:

```
<class 'int'>
<class 'float'>
<class 'str'>
```

```
# range(): Menghasilkan urutan angka yang digunakan dalam loop.
for i in range(5):
    print(i)
```

Output:

```
0
1
2
3
4
```

```
# sum(): Menghitung total dari elemen-elemen dalam sebuah iterable.
my_list = [1, 2, 3, 4, 5]
print(sum(my_list))  # Output: 15
```

Output:

```
15
```

```
# max() dan min(): Mengembalikan nilai maksimum dan minimum dari sebuah iterable.
my_list = [1, 2, 3, 4, 5]
print(max(my_list))  # Output: 5
print(min(my_list))  # Output: 1
```

Output:

```
5
1
```

```
# sorted(): Mengembalikan daftar yang diurutkan dari elemen-elemen dalam iterable.
my_list = [3, 1, 4, 1, 5, 9]
sorted_list = sorted(my_list)
print(sorted_list)  # Output: [1, 1, 3, 4, 5, 9]
```

Output:

```
[1, 1, 3, 4, 5, 9]
```

```
# abs(): Mengembalikan nilai absolut dari suatu bilangan
num = -42
print(abs(num))  # Output: 42
```

Output:

```
42
```

```
# round(): Membulatkan angka ke bilangan bulat terdekat atau ke jumlah desimal
tertentu.
num = 3.14159
print(round(num, 2))  # Output: 3.14
```

Output:

```
3.14
```

```
# open(): Digunakan untuk membuka file.
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

2.10. Pembacaan Dataset

Python menyediakan banyak pustaka dan modul yang memudahkan kita untuk membaca dan memanipulasi data dalam berbagai format. Berikut ini adalah beberapa cara umum untuk membaca dataset menggunakan Python, tergantung pada format dataset tersebut.

A. Modul CSV (Comma Separated Values)

Format CSV adalah salah satu format yang paling umum untuk menyimpan data tabel. Anda dapat menggunakan modul bawaan Python csv atau pustaka populer pandas untuk membaca file CSV. Modul csv adalah modul bawaan Python yang digunakan untuk membaca dan menulis file CSV.

```
import csv

# Membuka file CSV
with open('data.csv', mode='r') as file:
    csv_reader = csv.reader(file)
    # Membaca header
    header = next(csv_reader)
    print(f'Header: {header}')
    # Membaca setiap baris dalam file CSV
    for row in csv_reader:
        print(row)
```

B. Modul Pandas

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.head())
```

Untuk membaca file Excel, Anda bisa menggunakan pustaka pandas bersama openpyxl atau xlrd untuk file Excel lama.

```
import pandas as pd

df = pd.read_excel('data.xlsx')
print(df.head())
```

Membaca JSON dengan pandas. JSON adalah format file yang sering digunakan untuk pertukaran data.

```
import pandas as pd

df = pd.read_json('data.json')
print(df.head())
```

C. Dataset File SQL Database

Anda bisa menggunakan pustaka pandas bersama SQLAlchemy untuk membaca data dari database SQL.

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('sqlite:///database.db')
df = pd.read_sql('SELECT * FROM table_name', engine)
print(df.head())
```

D. Membaca File Parquet

Parquet adalah format file kolumnar yang sering digunakan untuk data besar menggunakan Pustaka pandas:

```
import pandas as pd

df = pd.read_parquet('data.parquet')
print(df.head())
```

E. Membaca File Text dan HTML

Jika Anda memiliki file teks dengan format tertentu, Anda bisa membacanya menggunakan Python menggunakan Pustaka pandas untuk File Teks dengan Delimiter:

```
import pandas as pd

df = pd.read_csv('data.txt', delimiter='\t')
print(df.head())
```

2.11. Pengenalan Visualisasi Data

Matplotlib adalah pustaka visualisasi data di Python yang sangat populer. Pustaka ini menyediakan antarmuka yang sederhana namun sangat kuat untuk membuat berbagai jenis grafik, mulai dari grafik garis sederhana hingga plot 3D yang kompleks. Matplotlib biasanya digunakan bersama dengan pustaka lain seperti NumPy dan pandas untuk mempermudah manipulasi dan analisis data sebelum divisualisasikan. Langkah-langkah Dasar Menggunakan Matplotlib:

1. Impor Pustaka: Impor matplotlib.pyplot sebagai plt.
2. Persiapkan Data: Siapkan data yang ingin divisualisasikan.
3. Buat Plot: Gunakan fungsi-fungsi plot dari matplotlib untuk membuat grafik.
4. Hiasi Plot: Tambahkan judul, label sumbu, legenda, dan elemen lain yang diperlukan.
5. Tampilkan Plot: Gunakan plt.show() untuk menampilkan grafik.

Contoh Kasus: Nilai Mahasiswa

Misalkan kita memiliki dataset nilai ujian akhir dari beberapa mahasiswa dalam beberapa mata pelajaran. Kita ingin membuat grafik untuk memvisualisasikan nilai rata-rata dari setiap mata pelajaran.

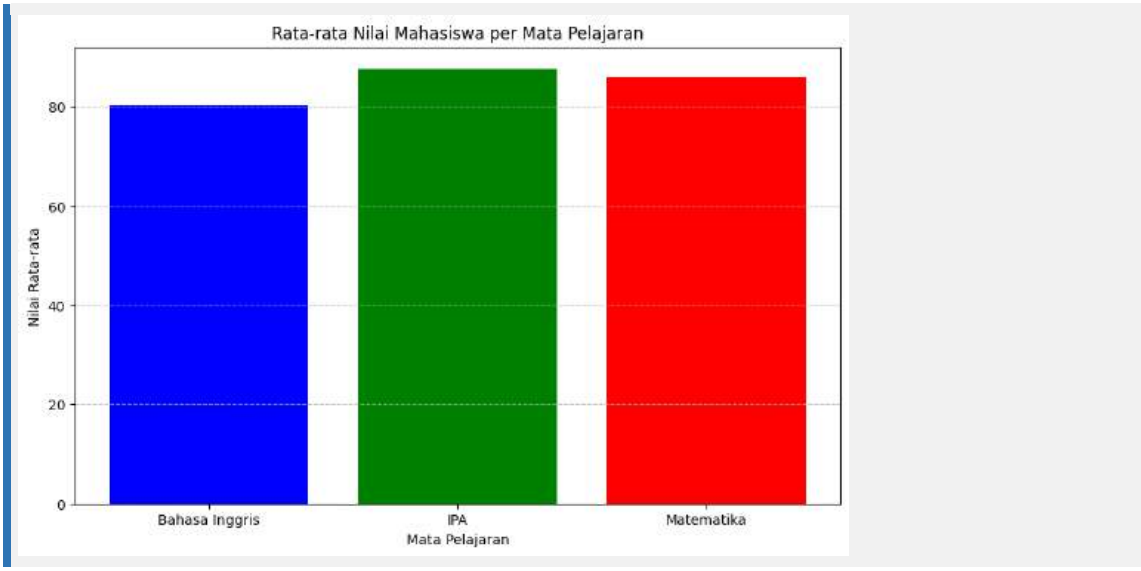
Langkah-langkah Visualisasi

1. Buat file CSV: Simpan dataset di atas ke dalam file bernama nilai_mahasiswa.csv.
2. Baca dataset dan visualisasikan menggunakan matplotlib.

```
import pandas as pd
import matplotlib.pyplot as plt

# Membaca dataset dari file CSV
df = pd.read_csv('nilai_mahasiswa.csv')

# Menghitung nilai rata-rata untuk setiap mata pelajaran
mean_scores = df.groupby('Mata Pelajaran')['Nilai'].mean()

# Menampilkan nilai rata-rata
print(mean_scores)

# Membuat grafik batang
plt.figure(figsize=(10, 6))
plt.bar(mean_scores.index, mean_scores.values, color=['blue', 'green', 'red'])

# Menambahkan judul dan label
plt.title('Rata-rata Nilai Mahasiswa per Mata Pelajaran')
plt.xlabel('Mata Pelajaran')
plt.ylabel('Nilai Rata-rata')

# Menambahkan grid
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Menampilkan grafik
plt.show()
```

Output:

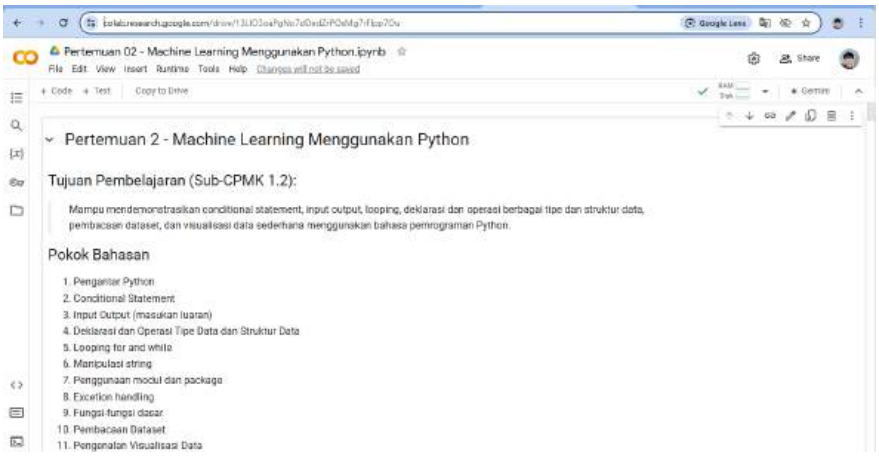
```
Mata Pelajaran
Bahasa Inggris    80.333333
IPA               87.666667
Matematika        86.000000
Name: Nilai, dtype: float64
```



Penjelasan Kode

1. Mengimpor Pustaka: Mengimpor pandas dan matplotlib.pyplot.
2. Membaca Dataset: Menggunakan pd.read_csv() untuk membaca dataset dari file CSV.
3. Menghitung Nilai Rata-rata: Mengelompokkan data berdasarkan mata pelajaran dan menghitung nilai rata-rata menggunakan groupby() dan mean().
4. Menampilkan Nilai Rata-rata: Menampilkan nilai rata-rata untuk setiap mata pelajaran.
5. Membuat Grafik Batang: Menggunakan plt.bar() untuk membuat grafik batang dengan nilai rata-rata untuk setiap mata pelajaran.
6. Menambahkan Judul dan Label: Menambahkan judul, label sumbu, dan grid pada grafik untuk memberikan informasi yang lebih jelas.
7. Menampilkan Grafik: Menggunakan plt.show() untuk menampilkan grafik.

Scan disi atau klik gambar di samping untuk mengakses Google Colab pembelajaran



2.12. Mengapa Python Digunakan untuk Machine Learning

Video ini akan menjelaskan mengapa Python menjadi bahasa pemrograman yang dominan dalam pengembangan machine learning. Python memiliki sintaksis yang sederhana dan mudah dipahami, menjadikannya ideal baik untuk pemula maupun profesional. Selain itu, Python didukung oleh ekosistem pustaka yang kaya, seperti NumPy, Pandas, Scikit-learn, TensorFlow, dan PyTorch, yang menyediakan alat lengkap untuk memproses data, membangun, dan melatih model machine learning. Python juga memiliki komunitas global yang aktif, sehingga banyak sumber daya, tutorial, dan dukungan yang mudah diakses. Kombinasi kelebihan ini menjadikan Python pilihan utama untuk membangun solusi machine learning secara efisien dan efektif.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

2.13. Modul Python yang Paling Sering Digunakan dalam Machine Learning

Video ini akan membahas modul-modul Python yang sering digunakan dalam pengembangan machine learning. Modul NumPy menjadi dasar untuk berbagai operasi matematika dan manipulasi array, mendukung pengolahan data dalam format numerik. Pandas membantu dalam pengolahan dan analisis data tabular secara efisien. Modul Scikit-learn menyediakan beragam algoritma machine learning yang mudah diimplementasikan untuk tugas-tugas seperti klasifikasi, regresi, dan klastering. Untuk deep learning, TensorFlow dan PyTorch merupakan framework andalan yang memungkinkan pembuatan dan pelatihan model neural network yang kompleks. Visualisasi data juga penting dalam machine learning; oleh karena itu, Matplotlib dan Seaborn digunakan untuk memvisualisasikan pola dan tren data dengan jelas. Dengan menguasai modul-modul ini, Anda akan memiliki dasar yang kuat dalam membangun dan mengembangkan aplikasi machine learning.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

2.14. Kenapa Menggunakan Numpy dan Pandas?

Video ini akan membahas pentingnya modul NumPy dan Pandas dalam analisis data dan pengembangan machine learning. NumPy digunakan untuk melakukan operasi matematika secara cepat dan efisien pada array multidimensi, serta menyediakan fungsi tambahan seperti linear algebra dan transformasi Fourier, yang sangat mendukung perhitungan numerik tingkat lanjut. Di sisi lain, Pandas menawarkan struktur data yang mudah diakses, seperti DataFrame, yang memfasilitasi manipulasi, pembersihan, dan analisis data dalam format tabular. Dengan Pandas, data dapat dikelola secara intuitif dan menyerupai pengalaman kerja

pada spreadsheet, sehingga analisis data lebih sederhana dan cepat. Menggunakan keduanya memberikan fleksibilitas dan efisiensi yang dibutuhkan untuk menangani data besar dan kompleks, menjadikan NumPy dan Pandas sebagai komponen penting dalam pipeline machine learning.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 3. Eksplorasi, Visualisasi, Pelatihan, dan Pengujian Dataset Berdasarkan Kasus Sederhana

Hal-hal yang dibahas pada Bab ini:

1. Persiapan dataset untuk pelatihan.
2. Proses pelatihan model.
3. Pengujian dan evaluasi model.
4. Implementasi tahapan inti machine learning.
5. Studi kasus sederhana.

3.1. Explorasi dan Visualisasi Dataset

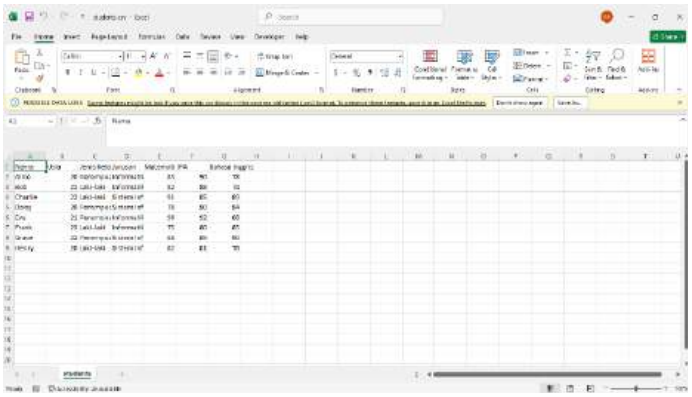
Eksplorasi dan visualisasi dataset adalah langkah penting dalam proses analisis data dan pengembangan model machine learning. Ini melibatkan pemahaman tentang struktur data, distribusi variabel, serta hubungan antar variabel. Mari kita jelajahi langkah-langkah ini dengan menggunakan contoh dataset terkenal, yaitu Iris dataset. Langkah-langkah Eksplorasi dan Visualisasi Dataset:

1. Memuat Dataset
2. Eksplorasi Awal Dataset
3. Visualisasi Data
    a. Distribusi Variabel
    b. Hubungan Antar Variabel
    c. Visualisasi Kelas

A. Dataset Mahasiswa

Eksplorasi dan visualisasi dataset mahasiswa membantu kita memahami karakteristik data dan hubungan antara variabel. Langkah-langkah ini sangat penting sebelum melangkah ke tahap pemodelan lebih lanjut dalam machine learning. Dengan visualisasi, kita bisa mengidentifikasi pola dan anomali dalam data yang mungkin tidak terlihat hanya dari analisis statistik sederhana.

Scan disi atau klik gambar di samping untuk mendownload dataset pembelajaran



1. Memuat Dataset dan Eksplorasi Awal

```
import pandas as pd

# Memuat dataset
df = pd.read_csv('students.csv')

# Menampilkan 5 baris pertama dataset
df.head()
```

Output:

index	Nama	Usia	Jenis Kelamin	Jurusan	Matematika	IPA	Bahasa Inggris
0	Alice	20	Perempuan	Informatika	85	90	78
1	Bob	21	Laki-laki	Informatika	82	88	74
2	Charlie	22	Laki-laki	Sistem Informasi	91	85	89
3	Daisy	20	Perempuan	Sistem Informasi	78	80	84
4	Eve	21	Perempuan	Informatika	90	92	88

```
# Memeriksa ukuran dataset
print(f"Ukuran dataset: {df.shape}")

# Memeriksa tipe data setiap kolom
print(df.dtypes)

# Memeriksa nilai yang hilang
print(df.isnull().sum())

# Deskripsi statistik dasar
print(df.describe(include='all'))
```

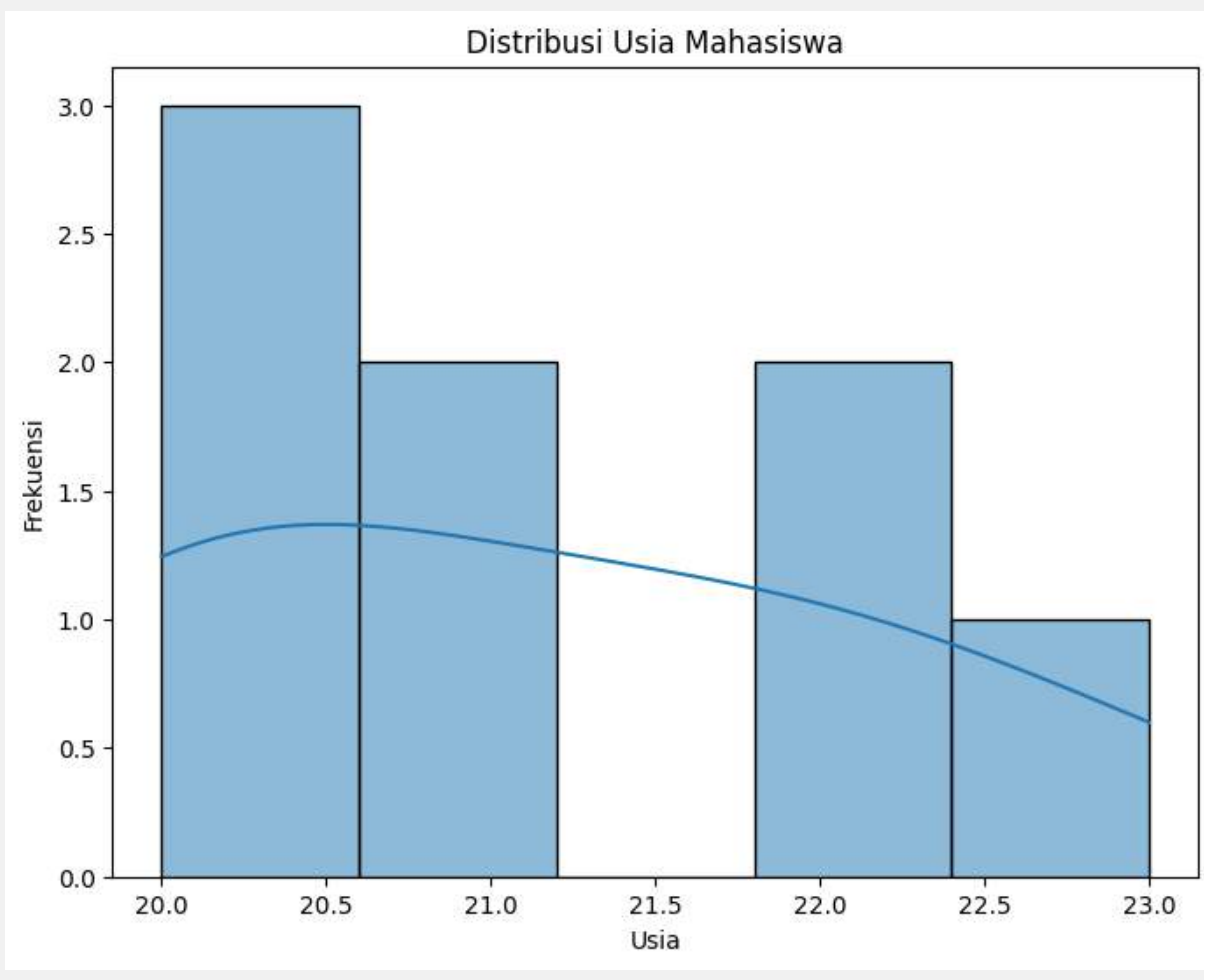
Output:

```
Ukuran dataset: (8, 7)
Nama              object
Usia               int64
Jenis Kelamin     object
Jurusan           object
Matematika         int64
IPA                int64
Bahasa Inggris     int64
dtype: object
Nama              0
Usia              0
Jenis Kelamin     0
Jurusan           0
Matematika        0
IPA               0
Bahasa Inggris    0
dtype: int64
         Nama       Usia Jenis Kelamin      Jurusan  Matematika        IPA  \
count       8   8.000000              8            8    8.000000   8.000000
unique      8        NaN              2            2         NaN        NaN
top     Alice        NaN     Perempuan  Informatika         NaN        NaN
freq        1        NaN              4            4         NaN        NaN
mean      NaN  21.125000            NaN          NaN   83.875000  85.625000
std       NaN   1.125992            NaN          NaN    5.693041   4.808846
min       NaN  20.000000            NaN          NaN   75.000000  80.000000
25%       NaN  20.000000            NaN          NaN   81.000000  80.750000
50%       NaN  21.000000            NaN          NaN   83.500000  86.500000
75%       NaN  22.000000            NaN          NaN   88.500000  89.250000
max       NaN  23.000000            NaN          NaN   91.000000  92.000000

        Bahasa Inggris
count         8.000000
unique             NaN
top                NaN
freq               NaN
mean         83.250000
std           5.922114
min          74.000000
25%          78.000000
50%          84.500000
75%          88.250000
max          90.000000
```

2. Visualisasi Data

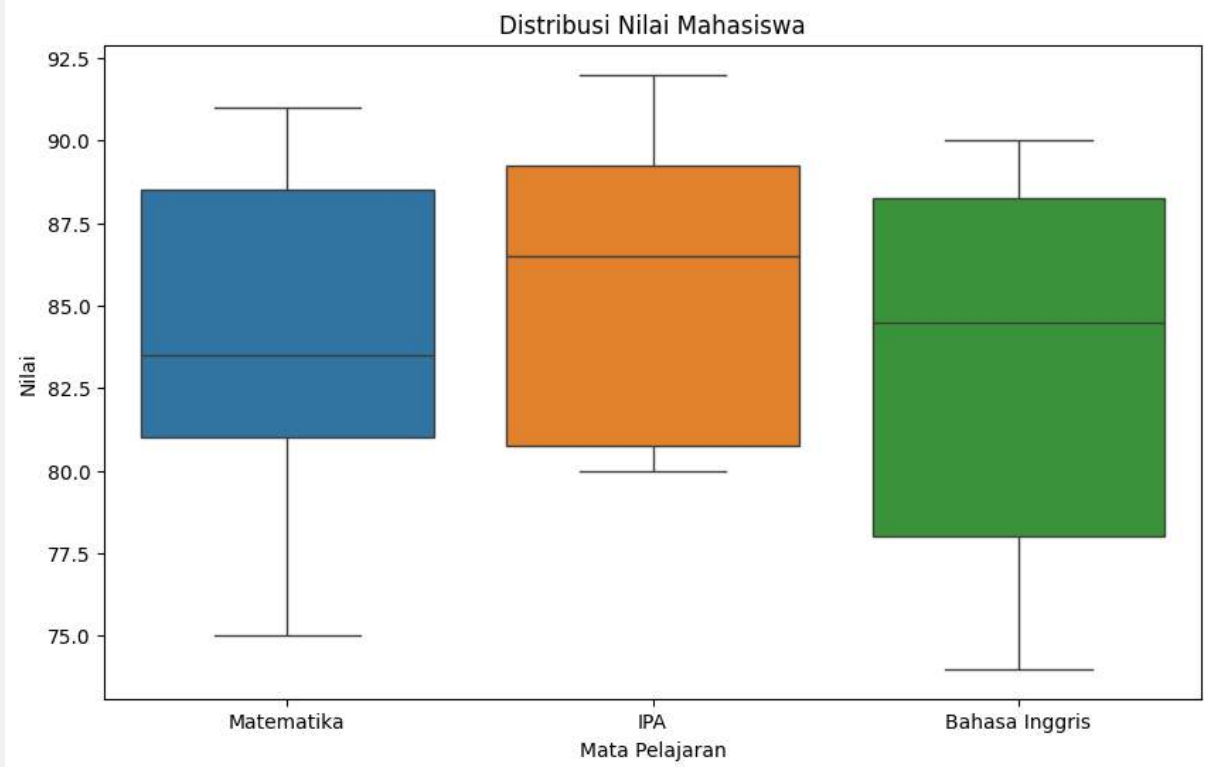
```
# Distribusi Usia: Mari kita visualisasikan distribusi usia mahasiswa.

import matplotlib.pyplot as plt
import seaborn as sns

# Histogram untuk distribusi usia
plt.figure(figsize=(8, 6))
sns.histplot(df['Usia'], bins=5, kde=True)
plt.title('Distribusi Usia Mahasiswa')
plt.xlabel('Usia')
plt.ylabel('Frekuensi')
plt.show()
```

Output:



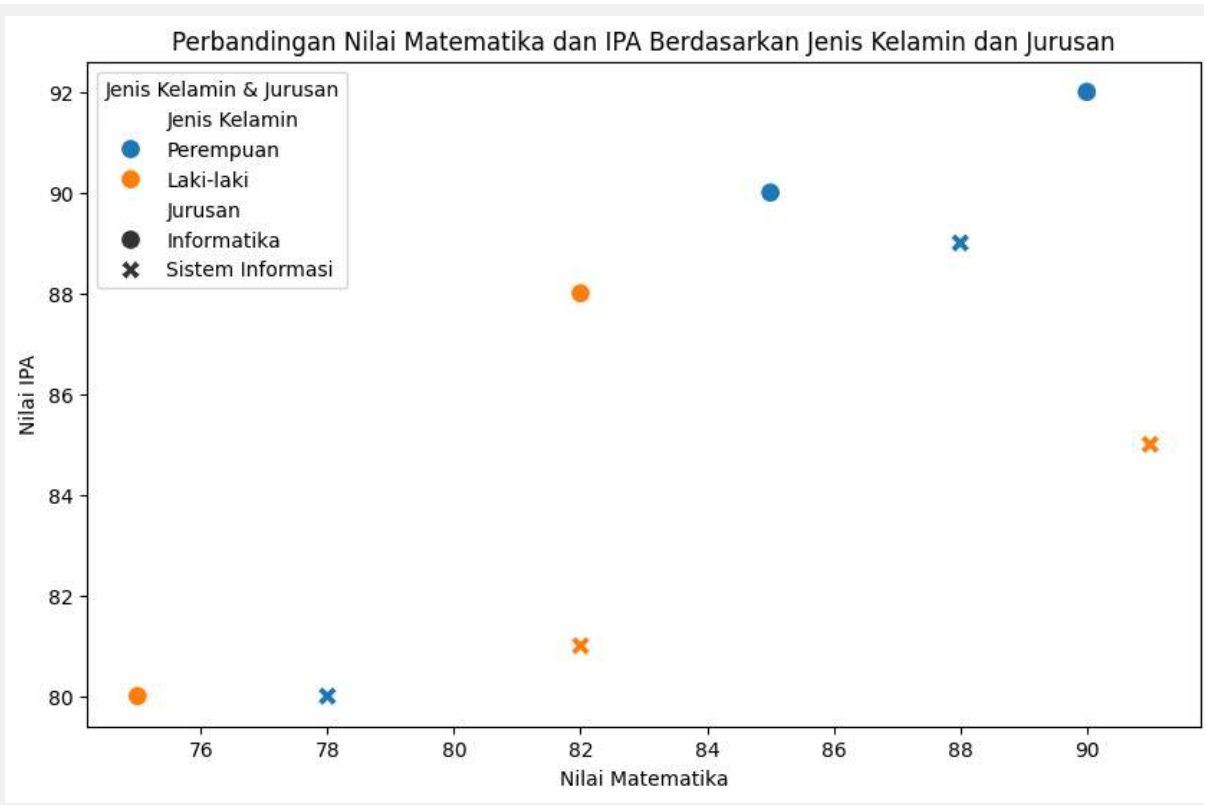

```
# Distribusi Nilai: Visualisasikan distribusi nilai untuk setiap mata pelajaran.

# Boxplot untuk nilai setiap mata pelajaran
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['Matematika', 'IPA', 'Bahasa Inggris']])
plt.title('Distribusi Nilai Mahasiswa')
plt.xlabel('Mata Pelajaran')
plt.ylabel('Nilai')
plt.show()
```

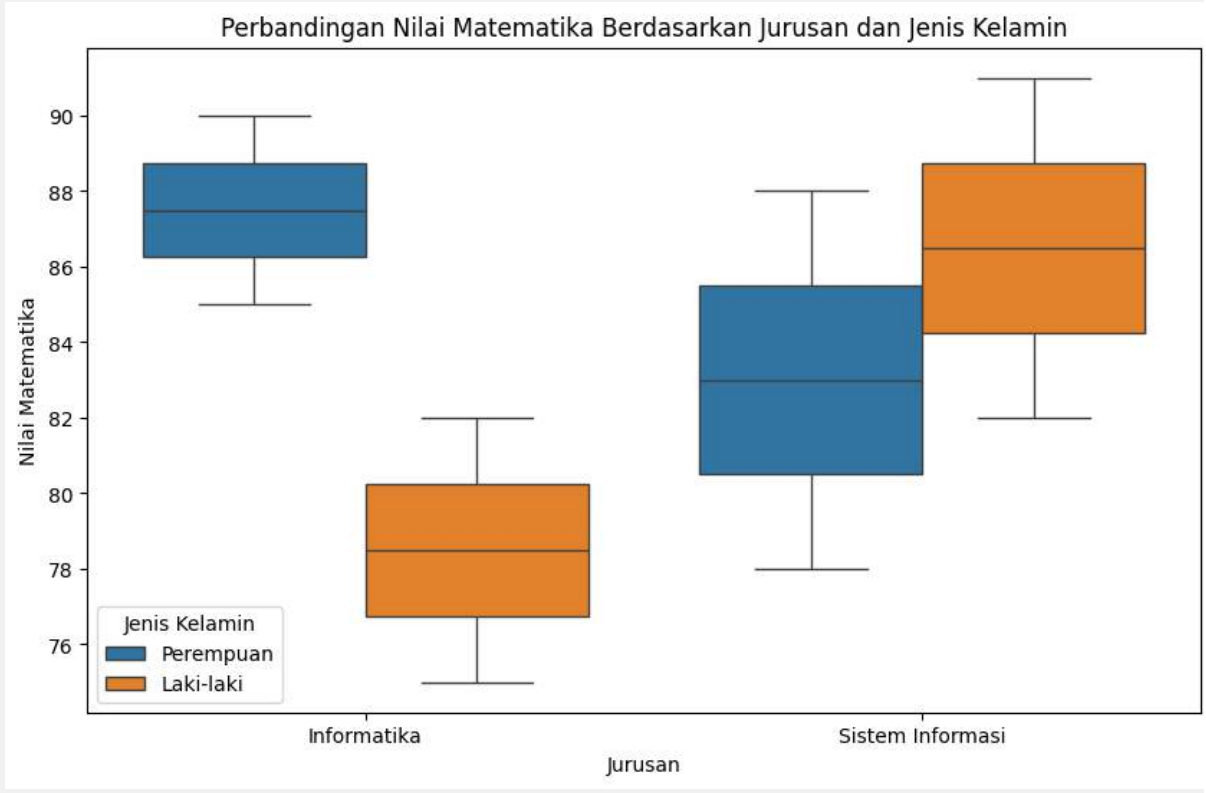
Output:




```
# Perbandingan Nilai Berdasarkan Jenis Kelamin dan Jurusan
# Visualisasikan perbandingan nilai berdasarkan jenis kelamin dan jurusan.

# Scatter plot untuk perbandingan nilai berdasarkan jenis kelamin
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Matematika', y='IPA', hue='Jenis Kelamin', style='Jurusan',
data=df, s=100)
plt.title('Perbandingan Nilai Matematika dan IPA Berdasarkan Jenis Kelamin dan
Jurusan')
plt.xlabel('Nilai Matematika')
plt.ylabel('Nilai IPA')
plt.legend(title='Jenis Kelamin & Jurusan')
plt.show()

# Boxplot untuk perbandingan nilai berdasarkan jurusan
plt.figure(figsize=(10, 6))
sns.boxplot(x='Jurusan', y='Matematika', hue='Jenis Kelamin', data=df)
plt.title('Perbandingan Nilai Matematika Berdasarkan Jurusan dan Jenis Kelamin')
plt.xlabel('Jurusan')
plt.ylabel('Nilai Matematika')
plt.show()
```

Output:





B. Spesies Bunga Iris

Bunga iris adalah salah satu genus tanaman berbunga yang terdiri dari berbagai spesies dengan warna-warna indah dan mencolok. Genus ini sangat populer di dunia

hortikultura dan sering ditanam di taman-taman sebagai tanaman hias. Dalam konteks dataset Iris yang sering digunakan dalam pembelajaran mesin, kita berfokus pada tiga spesies utama: Iris setosa, Iris versicolor, dan Iris virginica.

1. Memuat Dataset

Iris dataset dapat ditemukan di pustaka sklearn. Dataset ini berisi informasi tentang panjang dan lebar sepal dan petal dari tiga spesies bunga Iris. Dataset Iris dimuat menggunakan load_iris dari sklearn dan dikonversi menjadi DataFrame pandas untuk kemudahan manipulasi. Dataset Iris berisi pengukuran morfologi dari ketiga spesies ini, termasuk:

1. Sepal Length (Panjang Kelopak)
2. Sepal Width (Lebar Kelopak)
3. Petal Length (Panjang Petal)
4. Petal Width (Lebar Petal)

Pengukuran ini digunakan untuk membedakan ketiga spesies bunga iris. Berikut adalah ringkasan karakteristik morfologi yang sering digunakan:

a. Sepal Length and Width: Ukuran sepal cenderung lebih besar pada Iris virginica dibandingkan dengan Iris setosa dan Iris versicolor.
b. Petal Length and Width: Petal pada Iris virginica biasanya lebih panjang dan lebar dibandingkan dengan dua spesies lainnya. Iris setosa memiliki petal yang paling kecil di antara ketiganya.

```
# !pip install -U scikit-learn
from sklearn.datasets import load_iris
import pandas as pd

# Memuat dataset Iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target

# Menampilkan 5 baris pertama dataset
df.head()
```

Output:

index	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0

4	5.0	3.6	1.4	0.2	0

2. Eksplorasi Awal Dataset

Eksplorasi awal melibatkan pemeriksaan ukuran dataset, tipe data, nilai yang hilang, dan deskripsi statistik dasar. Selain itu, pada proses ini juga dilakukan pemeriksaan ukuran dataset, tipe data, nilai yang hilang, dan statistik deskriptif dasar.

```
# !pip install -U scikit-learn
from sklearn.datasets import load_iris
import pandas as pd

# Memuat dataset Iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target

# Menampilkan 5 baris pertama dataset
df.head()# Memeriksa ukuran dataset
print(df.shape)

# Memeriksa tipe data setiap kolom
print(df.dtypes)

# Memeriksa nilai yang hilang
print(df.isnull().sum())

# Deskripsi statistik dasar
print(df.describe())
```

Output:

```
(150, 5)
sepal length (cm)    float64
sepal width (cm)     float64
petal length (cm)    float64
petal width (cm)     float64
species                int64
dtype: object
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
```

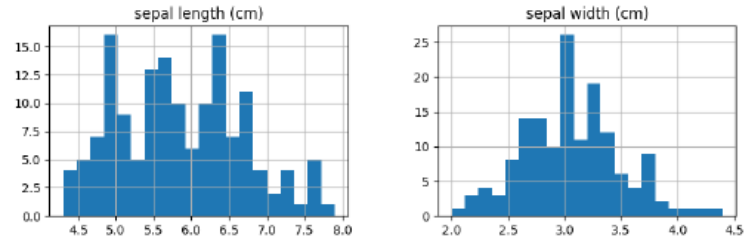
```
25%             5.100000          2.800000          1.600000
50%             5.800000          3.000000          4.350000
75%             6.400000          3.300000          5.100000
max             7.900000          4.400000          6.900000

       petal width (cm)     species
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```

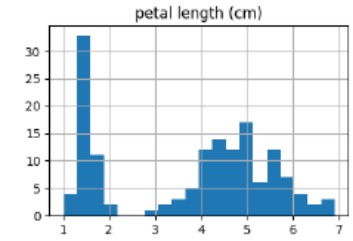
3. Visualisasi Data

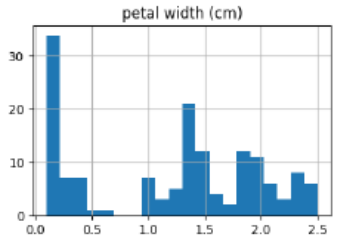
Eksplorasi dan visualisasi dataset adalah langkah krusial dalam analisis data. Dengan memahami distribusi data, hubungan antar variabel, dan distribusi kelas, kita dapat memperoleh wawasan berharga yang membantu dalam proses pengembangan model machine learning. Langkah-langkah ini membantu kita memahami karakteristik dataset sebelum melangkah ke tahap pemodelan lebih lanjut. Visualisasi membantu memahami distribusi data dan hubungan antar variabel.

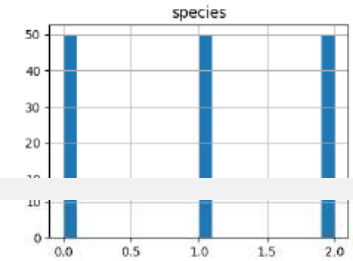
```
#  Distribusi Variabel: Distribusi variabel individual dapat divisualisasikan
menggunakan histogram atau boxplot.

import matplotlib.pyplot as plt
import seaborn as sns

# Histogram untuk setiap fitur
df.hist(bins=20, figsize=(10, 10))
plt.show()

# Boxplot untuk setiap fitur
plt.figure(figsize=(10, 6))
sns.boxplot(data=df.iloc[:, :-1])
plt.title('Boxplot of Iris Features')
plt.show()
```

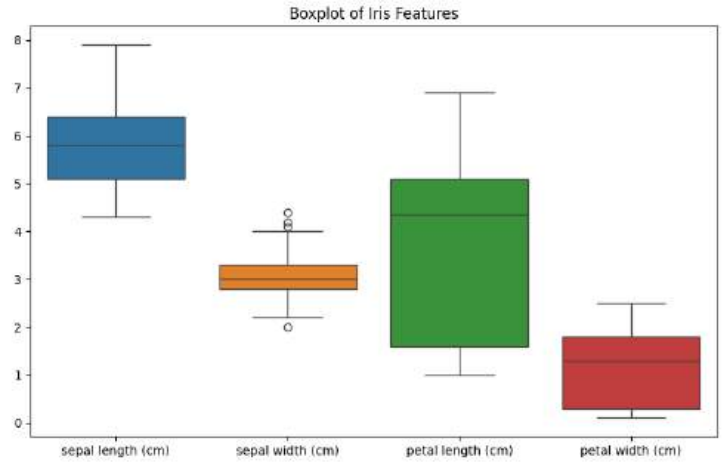
Output:





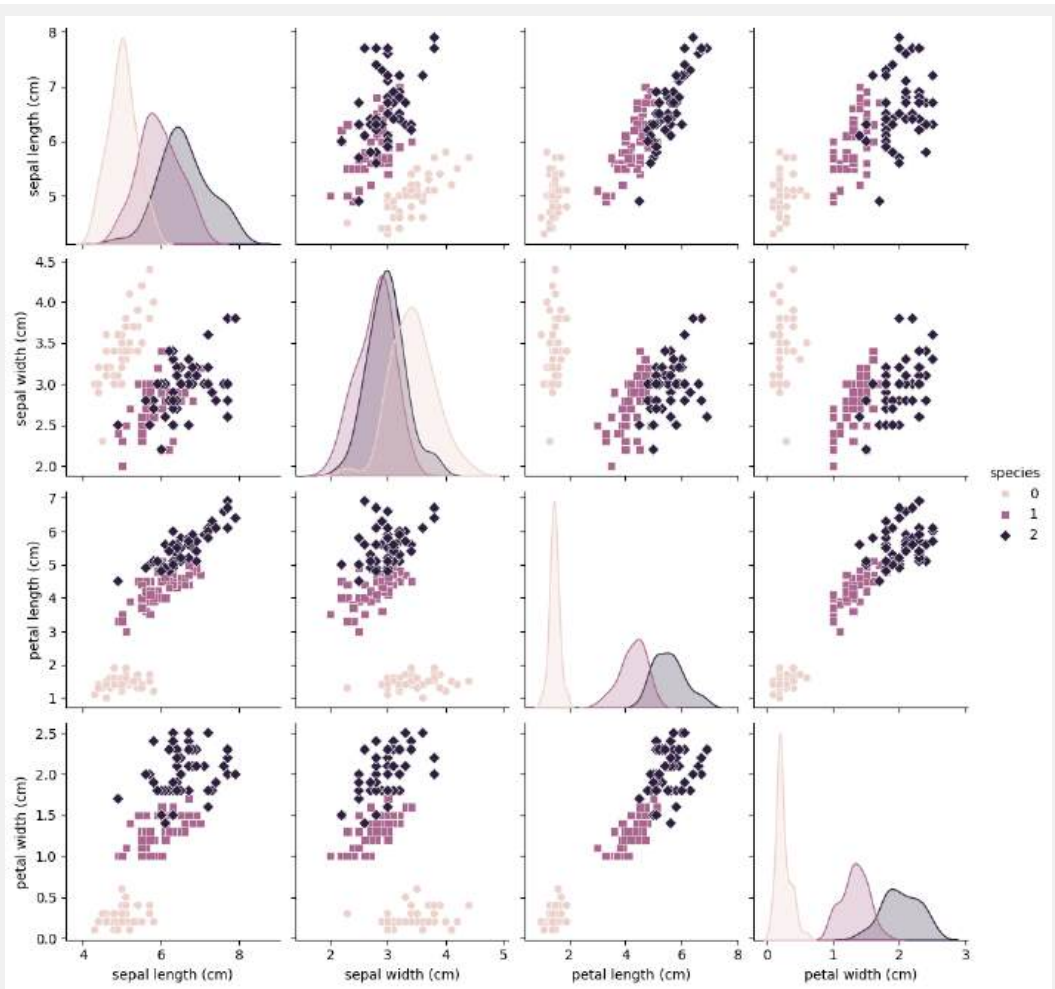


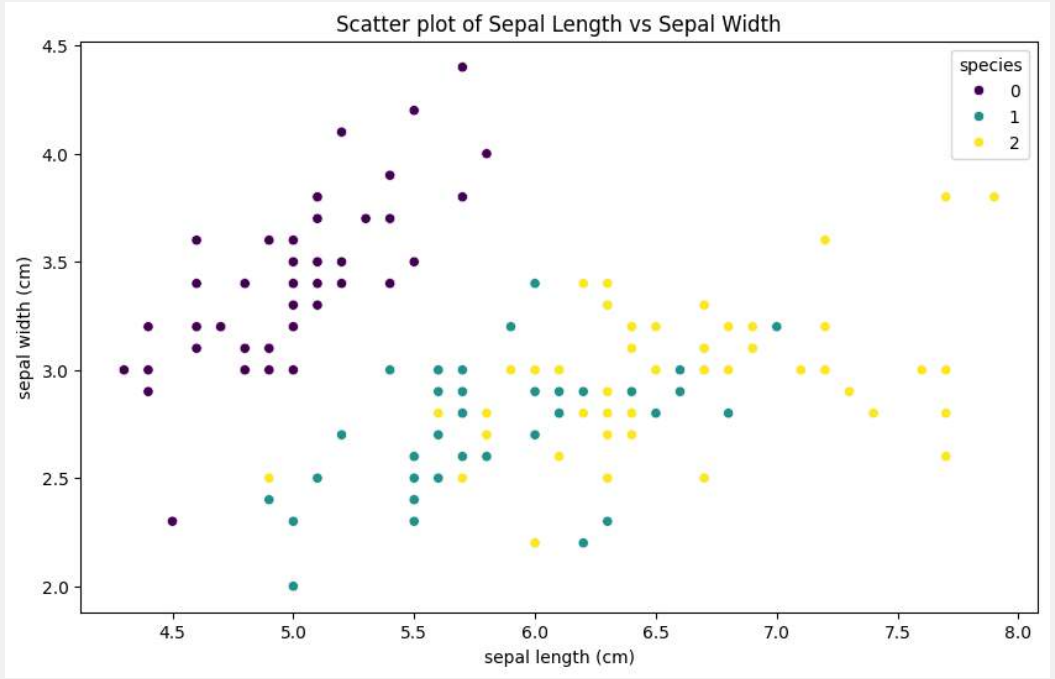





```
# Hubungan Antar Variabel: Hubungan antar variabel dapat divisualisasikan menggunakan pairplot atau scatter plot.
# Pairplot untuk melihat hubungan antar variabel
sns.pairplot(df, hue='species', markers=["o", "s", "D"])
plt.show()

# Scatter plot untuk hubungan spesifik
plt.figure(figsize=(10, 6))
sns.scatterplot(x='sepal length (cm)', y='sepal width (cm)', hue='species', data=df, palette='viridis')
plt.title('Scatter plot of Sepal Length vs Sepal Width')
plt.show()
```

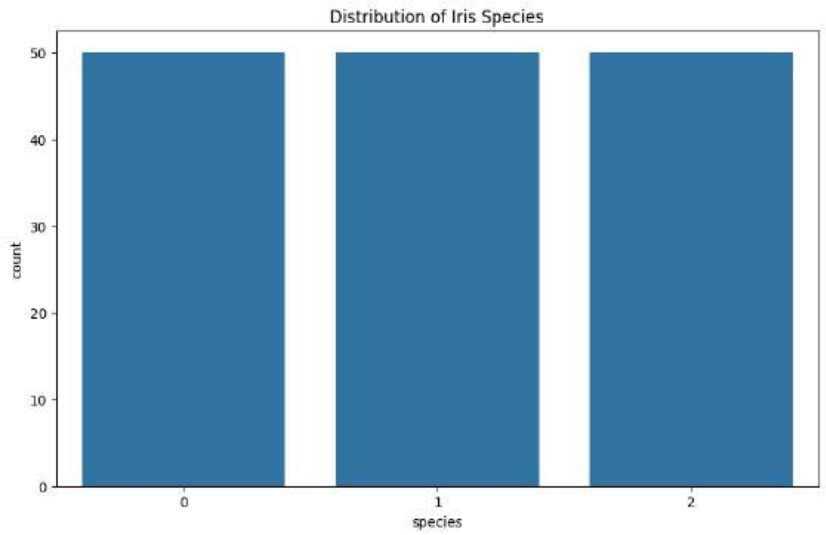
Output:

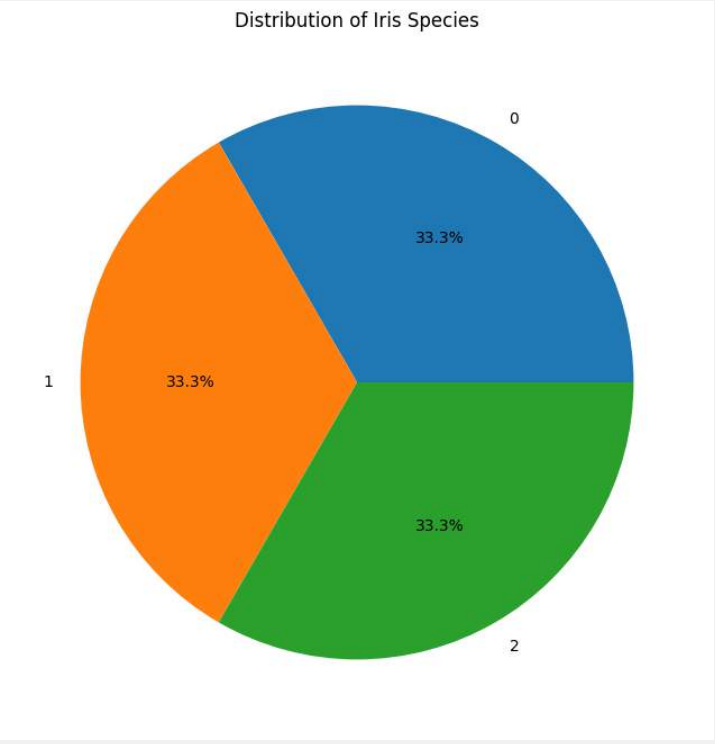

sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
species
0
1
2


Scatter plot of Sepal Length vs Sepal Width
species
0
1
2
sepal width (cm)
sepal length (cm)

```
# Visualisasi Kelas: Distribusi kelas dalam dataset dapat divisualisasikan menggunakan count plot atau pie chart.
# Count plot untuk distribusi kelas
plt.figure(figsize=(10, 6))
sns.countplot(x='species', data=df)
plt.title('Distribution of Iris Species')
plt.show()

# Pie chart untuk distribusi kelas
df['species'].value_counts().plot.pie(autopct='%1.1f%%', figsize=(8, 8))
plt.title('Distribution of Iris Species')
plt.ylabel('')
plt.show()
```

Output:





3.2. Evaluasi Model

Evaluasi model adalah langkah krusial dalam proses pengembangan model machine learning. Metrik evaluasi kinerja seperti accuracy, precision, recall, dan F1 score digunakan untuk menilai seberapa baik model melakukan prediksi pada data yang belum pernah dilihat sebelumnya. Metrik-metrik ini membantu kita memahami performa model secara komprehensif, terutama dalam situasi di mana terdapat ketidakseimbangan kelas atau ketika kesalahan tertentu lebih kritis dibandingkan kesalahan lainnya. Dengan menggunakan metrik evaluasi yang tepat, kita dapat memilih dan mengoptimalkan model machine learning dengan lebih baik. Mari kita bahas setiap metrik tersebut dan bagaimana menghitungnya. Beberapa contoh metrik Evaluasi Kinerja Model:

A. Accuracy (Akurasi)

Akurasi adalah persentase dari prediksi yang benar dari keseluruhan prediksi yang dibuat oleh model. Ini adalah metrik yang paling dasar dan mudah dipahami.

B. Precision (Presisi)

Presisi adalah proporsi dari prediksi positif yang benar-benar positif. Ini adalah metrik yang penting ketika biaya kesalahan positif palsu (false positive) tinggi. image.png

C. Recall

Rekal adalah proporsi dari benar-benar positif yang diidentifikasi oleh model. Ini adalah metrik yang penting ketika biaya kesalahan negatif palsu (false negative) tinggi.

3.3. Cross Validation

Cross-validation adalah teknik yang digunakan untuk mengevaluasi kinerja model machine learning dengan lebih akurat. Tujuan utamanya adalah untuk mengurangi variabilitas hasil evaluasi yang disebabkan oleh pemisahan data menjadi training dan testing set. Cross-validation membantu dalam memastikan bahwa model dapat digeneralisasi dengan baik untuk data yang belum pernah dilihat sebelumnya. Pada intinya, cross-validation membagi dataset menjadi beberapa subset atau "fold". Model dilatih pada beberapa fold dan diuji pada fold yang tersisa. Proses ini diulang beberapa kali dengan fold yang berbeda untuk mendapatkan rata-rata performa model. Cross-validation adalah teknik yang efektif untuk mengevaluasi kinerja model machine learning dengan lebih akurat dan mengurangi risiko overfitting. Dengan menggunakan berbagai tipe cross-validation, kita dapat

memastikan bahwa model memiliki kemampuan generalisasi yang baik. Berikut adalah ringkasan dari masing-masing tipe cross-validation:

A. K-Fold Cross-validation

Dataset dibagi menjadi kk subset atau fold. Model dilatih pada k−1k−1 fold dan diuji pada fold yang tersisa. Proses ini diulang kk kali sehingga setiap fold digunakan sekali sebagai data uji. Hasil akhir adalah rata-rata performa dari setiap iterasi.

B. Stratified K-Fold Cross-validation

Mirip dengan K-Fold Cross-validation, tetapi pembagian fold dilakukan sedemikian rupa sehingga proporsi kelas dalam setiap fold sama dengan proporsi kelas dalam dataset asli. Ini sangat berguna untuk dataset dengan distribusi kelas yang tidak seimbang.

C. Leave-One-Out Cross-validation (LOOCV)

Setiap instance data digunakan sebagai data uji sekali, sementara sisa data digunakan untuk melatih model. Proses ini diulang sebanyak jumlah instance dalam dataset. LOOCV cocok untuk dataset yang sangat kecil.

D. Repeated K-Fold Cross-validation

K-Fold Cross-validation diulang beberapa kali dengan pembagian fold yang berbeda setiap kali. Ini memberikan evaluasi yang lebih stabil karena mengurangi variabilitas yang disebabkan oleh pembagian data yang berbeda.

Langkah-langkah Implementasi Cross-validation

Mari kita lihat contoh implementasi menggunakan K-Fold Cross-validation dengan dataset Iris.

A. Memuat Dataset dan Mengatur Model

Membagi data menjadi kk fold, ideal untuk evaluasi yang cepat dan dataset yang tidak terlalu besar.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LogisticRegression

# Memuat dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Mengatur model
model = LogisticRegression(max_iter=200)
```

B. K-Fold Cross-validation

Menjaga proporsi kelas dalam setiap fold, cocok untuk dataset dengan distribusi kelas yang tidak seimbang.

```
# Mengatur K-Fold Cross-validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Melakukan cross-validation dan menghitung skor
scores = cross_val_score(model, X, y, cv=kfold)

print(f'K-Fold Cross-validation Scores: {scores}')
print(f'Mean Score: {scores.mean()}')
```

Output:

```
K-Fold Cross-validation Scores: [1.          1.          0.93333333 0.96666667
0.96666667]
Mean Score: 0.9733333333333334
```

C. Stratified K-Fold Cross-validation

Menggunakan setiap instance sebagai data uji, cocok untuk dataset kecil.

```
from sklearn.model_selection import StratifiedKFold

# Mengatur Stratified K-Fold Cross-validation
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Melakukan cross-validation dan menghitung skor
stratified_scores = cross_val_score(model, X, y, cv=stratified_kfold)

print(f'Stratified K-Fold Cross-validation Scores: {stratified_scores}')
print(f'Mean Score: {stratified_scores.mean()}')
```

Output:

```
Stratified K-Fold Cross-validation Scores: [1.         0.96666667 0.93333333 1.
0.93333333]
Mean Score: 0.9666666666666668
```

D. Leave-One-Out Cross-validation (LOOCV)

Mengulang K-Fold Cross-validation beberapa kali untuk hasil evaluasi yang lebih stabil

```
from sklearn.model_selection import LeaveOneOut

# Mengatur LOOCV
loocv = LeaveOneOut()

# Melakukan cross-validation dan menghitung skor
loocv_scores = cross_val_score(model, X, y, cv=loocv)

print(f'Leave-One-Out Cross-validation Mean Score: {loocv_scores.mean()}')
```

Output:

```
Leave-One-Out Cross-validation Mean Score: 0.9666666666666667
```

E. Repeated K-Fold Cross-validation

```
from sklearn.model_selection import RepeatedKFold

# Mengatur Repeated K-Fold Cross-validation
repeated_kfold = RepeatedKFold(n_splits=5, n_repeats=10, random_state=42)

# Melakukan cross-validation dan menghitung skor
repeated_scores = cross_val_score(model, X, y, cv=repeated_kfold)

print(f'Repeated K-Fold Cross-validation Mean Score: {repeated_scores.mean()}')
```

Output:

```
Repeated K-Fold Cross-validation Mean Score: 0.9640000000000001
```

3.4. Grid Search

Grid Search adalah teknik yang digunakan dalam machine learning untuk menemukan kombinasi parameter hyperparameter yang optimal untuk model. Proses ini melibatkan pencarian melalui ruang parameter yang telah ditentukan sebelumnya untuk menemukan set parameter yang memberikan kinerja terbaik berdasarkan metrik evaluasi tertentu. Berikut ini adalah prinsip-prinsip Grid Search.

1. Definisi Ruang Parameter: Pertama, tentukan hyperparameter yang akan disesuaikan serta nilai-nilai yang mungkin untuk setiap hyperparameter. Misalnya, untuk model Support Vector Machine (SVM), hyperparameter bisa berupa C dan gamma.
2. Pencarian Grid: Grid Search secara sistematis mencoba semua kombinasi yang mungkin dari nilai hyperparameter yang telah ditentukan. Ini seperti menciptakan sebuah grid dari semua kemungkinan kombinasi dan mengevaluasi setiap titik dalam grid tersebut.
3. Evaluasi Model: Setiap kombinasi hyperparameter diuji dengan melakukan training pada subset data (umumnya menggunakan teknik cross-validation) dan dievaluasi berdasarkan metrik performa tertentu, seperti akurasi, F1-score, atau mean squared error (MSE).
4. Pemilihan Kombinasi Terbaik: Kombinasi hyperparameter yang menghasilkan kinerja terbaik pada data validasi dipilih sebagai kombinasi yang optimal.

Grid Search memiliki manfaat sebagai berikut:

1. Optimasi Hyperparameter: Membantu menemukan set hyperparameter yang paling sesuai untuk model tertentu, meningkatkan performa model secara signifikan.
2. Sistematis dan Komprehensif: Memastikan bahwa semua kombinasi hyperparameter yang memungkinkan telah dieksplorasi.
3. Integrasi dengan Cross-Validation: Menggunakan cross-validation selama proses pencarian, sehingga mengurangi risiko overfitting dan memberikan estimasi kinerja model yang lebih andal.

Contoh Penggunaan dalam Machine Learning

Misalkan kita memiliki dataset dan ingin melatih model Support Vector Machine (SVM). Hyperparameter yang ingin kita optimalkan adalah C (regularization parameter) dan gamma (kernel coefficient). Berikut adalah contoh penggunaan Grid Search dengan Python menggunakan scikit-learn:

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target

# Define the model
model = SVC()

# Define the parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf']
}

# Initialize Grid Search
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy')

# Fit Grid Search
grid_search.fit(X, y)

# Best parameters and best score
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

Output:

```
Best Parameters: {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
Best Score: 0.9800000000000001
```

Setelah menjalankan Grid Search, kita akan mendapatkan kombinasi hyperparameter terbaik berdasarkan kinerja pada data validasi. Misalnya, hasilnya bisa menunjukkan bahwa C=10 dan gamma=0.1 memberikan akurasi terbaik. Model

SVM kemudian dapat dilatih ulang menggunakan kombinasi hyperparameter ini untuk memaksimalkan kinerjanya pada data uji atau dalam aplikasi nyata.

3.5. Random Search

Random Search adalah teknik lain untuk mengoptimalkan hyperparameter model machine learning. Berbeda dengan Grid Search yang secara sistematis mencoba semua kombinasi hyperparameter, Random Search mencoba sejumlah kombinasi hyperparameter yang dipilih secara acak dari ruang parameter yang telah ditentukan. Cara kerja Random Search adalah sebagai berikut:

1. Definisi Ruang Parameter: Tentukan hyperparameter yang akan disesuaikan serta rentang nilai yang mungkin untuk setiap hyperparameter. Misalnya, untuk model Random Forest, hyperparameter bisa berupa n_estimators dan max_depth.
2. Pencarian Acak: Secara acak pilih kombinasi hyperparameter dari ruang parameter yang telah ditentukan. Jumlah kombinasi yang diuji biasanya ditentukan sebelumnya.
3. Evaluasi Model: Setiap kombinasi hyperparameter diuji dengan melakukan training pada subset data (umumnya menggunakan teknik cross-validation) dan dievaluasi berdasarkan metrik performa tertentu.
4. Pemilihan Kombinasi Terbaik: Kombinasi hyperparameter yang menghasilkan kinerja terbaik pada data validasi dipilih sebagai kombinasi yang optimal.

Perbandingan Random Search dengan Grid Search

Kelebihan Random Search

1. Efisiensi: Random Search seringkali lebih efisien karena tidak mencoba setiap kombinasi yang mungkin, tetapi cukup dengan mencoba sejumlah kombinasi acak yang representatif. Ini sangat berguna jika ruang parameter sangat besar.
2. Kemungkinan Menemukan Kombinasi Optimal: Dengan jumlah iterasi yang sama, Random Search dapat mengeksplorasi ruang parameter yang lebih luas dan memiliki peluang lebih besar untuk menemukan kombinasi hyperparameter yang optimal dibandingkan Grid Search, terutama jika beberapa hyperparameter tidak terlalu mempengaruhi performa model.
3. Fleksibilitas: Mudah diterapkan dan dapat berhenti kapan saja jika sumber daya komputasi terbatas.

Kelemahan Random Search

1. Tidak Sistematis: Karena bersifat acak, ada kemungkinan beberapa kombinasi hyperparameter yang penting terlewatkan.
2. Kinerja yang Tidak Konsisten: Kinerja dapat bervariasi antara satu run dan run lainnya karena sifatnya yang acak.

Kelebihan Grid Search

1. Eksplorasi Sistematis: Semua kombinasi hyperparameter yang mungkin dieksplorasi, memastikan bahwa tidak ada kombinasi yang terlewat.
2. Prediktabilitas: Hasil yang konsisten dan dapat diulang, karena semua kombinasi telah diuji.

Kelemahan Grid Search

1. Biaya Komputasi Tinggi: Grid Search bisa sangat memakan waktu dan sumber daya, terutama jika ruang parameter sangat besar.
2. Kurang Efisien: Mencoba semua kombinasi, termasuk kombinasi yang mungkin tidak signifikan untuk performa model, dapat menjadi pemborosan sumber daya.

Contoh Penggunaan Random Search dalam Machine Learning

Misalkan kita ingin mengoptimalkan model Random Forest dengan menggunakan Random Search. Berikut adalah contoh penggunaannya dengan Python menggunakan scikit-learn:

```
from sklearn.model_selection import RandomizedSearchCV from sklearn.ensemble import RandomForestClassifier from sklearn.datasets import load_iris import numpy as np
```

Load dataset

```
iris = load_iris() X, y = iris.data, iris.target
```

Define the model

```
model = RandomForestClassifier()
```

Define the parameter grid

```
param_distributions = { 'n_estimators': np.arange(10, 200, 10), 'max_depth': np.arange(1, 20), 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4] }
```

Initialize Random Search

```
random_search = RandomizedSearchCV(model, param_distributions, n_iter=100, cv=5, scoring='accuracy', random_state=42)
```

Fit Random Search

```
random_search.fit(X, y)
```

Best parameters and best score

```
print("Best Parameters:", random_search.best_params_) print("Best Score:",
random_search.best_score_)
```

Setelah menjalankan Random Search, kita akan mendapatkan kombinasi hyperparameter terbaik berdasarkan kinerja pada data validasi. Misalnya, hasilnya bisa menunjukkan bahwa n_estimators=150 dan max_depth=10 memberikan akurasi terbaik. Model Random Forest kemudian dapat dilatih ulang menggunakan kombinasi hyperparameter ini untuk memaksimalkan kinerjanya pada data uji atau dalam aplikasi nyata.

3.6. Model Selection

Model Selection adalah proses memilih model machine learning terbaik dari beberapa kandidat berdasarkan kinerja mereka pada data validasi. Ini melibatkan evaluasi dan perbandingan berbagai model dan set hyperparameter untuk menentukan model yang memberikan performa terbaik. Berikut ini adalah teknik pemilihan model terbaik:

1. Split Data (Training, Validation, Testing): Pisahkan data menjadi set pelatihan, validasi, dan pengujian. Data pelatihan digunakan untuk melatih model, data validasi digunakan untuk memilih model terbaik, dan data pengujian digunakan untuk mengevaluasi kinerja akhir model.
2. Cross-Validation: Teknik yang membagi data pelatihan menjadi beberapa subset (folds) dan melatih model pada beberapa kombinasi subset ini. Cross-validation membantu dalam mendapatkan estimasi kinerja model yang lebih andal dan mengurangi risiko overfitting.
3. Grid Search dan Random Search: Menggunakan Grid Search atau Random Search untuk menemukan kombinasi hyperparameter terbaik untuk setiap model kandidat.
4. K-Fold Cross-Validation: Teknik cross-validation yang umum digunakan, di mana data dibagi menjadi K subset (folds). Model dilatih K kali, setiap kali menggunakan K-1 subset untuk pelatihan dan 1 subset untuk validasi. Hasil akhirnya adalah rata-rata performa dari K iterasi.
5. Metrics Evaluasi: Pilih metrik evaluasi yang sesuai dengan tugas yang dihadapi, seperti akurasi, precision, recall, F1-score, atau mean squared error. Metrik ini digunakan untuk membandingkan kinerja model.
6. Ensembling: Teknik yang menggabungkan prediksi dari beberapa model untuk meningkatkan kinerja keseluruhan. Contoh ensembling termasuk bagging (seperti Random Forest), boosting (seperti Gradient Boosting), dan stacking.

7. Learning Curve Analysis: Menggunakan learning curve untuk memahami bagaimana kinerja model berubah seiring dengan jumlah data pelatihan. Ini dapat membantu dalam mendeteksi overfitting atau underfitting.\

Contoh Implementasi Model Selection dengan Python

Misalkan kita ingin memilih model terbaik antara Logistic Regression, Random Forest, dan Support Vector Machine (SVM) menggunakan k-fold cross-validation:

```
from sklearn.model_selection import cross_val_score, train_test_split from sklearn.linear_model import LogisticRegression from sklearn.ensemble import RandomForestClassifier from sklearn.svm import SVC from sklearn.datasets import load_iris import numpy as np
```

Load dataset

```
iris = load_iris() X, y = iris.data, iris.target
```

Split data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Define models

```
models = { 'Logistic Regression': LogisticRegression(max_iter=200), 'Random Forest': RandomForestClassifier(), 'SVM': SVC() }
```

Perform 5-fold cross-validation

```
cv_results = {} for model_name, model in models.items(): scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy') cv_results[model_name] = np.mean(scores) print(f"{model_name}: {np.mean(scores)}")
```

Select the best model based on cross-validation results

```
best_model_name = max(cv_results, key=cv_results.get) best_model = models[best_model_name]
```

Train the best model on the entire training set and evaluate on the test set

```
best_model.fit(X_train, y_train) test_score = best_model.score(X_test, y_test)

print(f"Best Model: {best_model_name}") print(f"Test Set Accuracy: {test_score}")
```

Dengan menjalankan kode di atas, kita dapat membandingkan rata-rata akurasi dari setiap model menggunakan 5-fold cross-validation pada data pelatihan. Model dengan rata-rata akurasi tertinggi dipilih sebagai model terbaik. Kemudian, model ini dilatih ulang pada seluruh data pelatihan dan dievaluasi pada data pengujian untuk mendapatkan akurasi akhir.

Teknik Tambahan

1. Grid Search with Cross-Validation: Menggunakan GridSearchCV untuk menggabungkan pencarian hyperparameter dan cross-validation dalam satu langkah.
2. Random Search with Cross-Validation: Menggunakan RandomizedSearchCV untuk menggabungkan pencarian hyperparameter dan cross-validation dengan pemilihan hyperparameter secara acak.
3. Model Ensembling: Kombinasikan beberapa model untuk meningkatkan kinerja dengan teknik seperti voting, bagging, boosting, dan stacking.

3.7. Contoh Kasus

Mari kita lihat contoh kasus eksplorasi, visualisasi, pelatihan, dan pengujian dataset menggunakan Python dengan dataset klasik "Iris". Dataset ini terdiri dari 150 sampel data bunga Iris, dengan empat fitur (panjang dan lebar sepal serta petal) dan tiga kelas (setosa, versicolor, virginica).

Langkah 1: Eksplorasi Data

Pertama, kita akan melakukan eksplorasi data untuk memahami karakteristik dasar dari dataset.

```
!pip install pandas
!pip install seaborn

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['species'] = iris.target

# Map target integers to species names
data['species'] = data['species'].map({0: 'setosa', 1: 'versicolor', 2:
'virginica'})

# Display first few rows
print(data.head())

# Summary statistics
print(data.describe())
```

```
# Check for missing values
print(data.isnull().sum())
```

Output:

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-
packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages
(0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-
packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
```

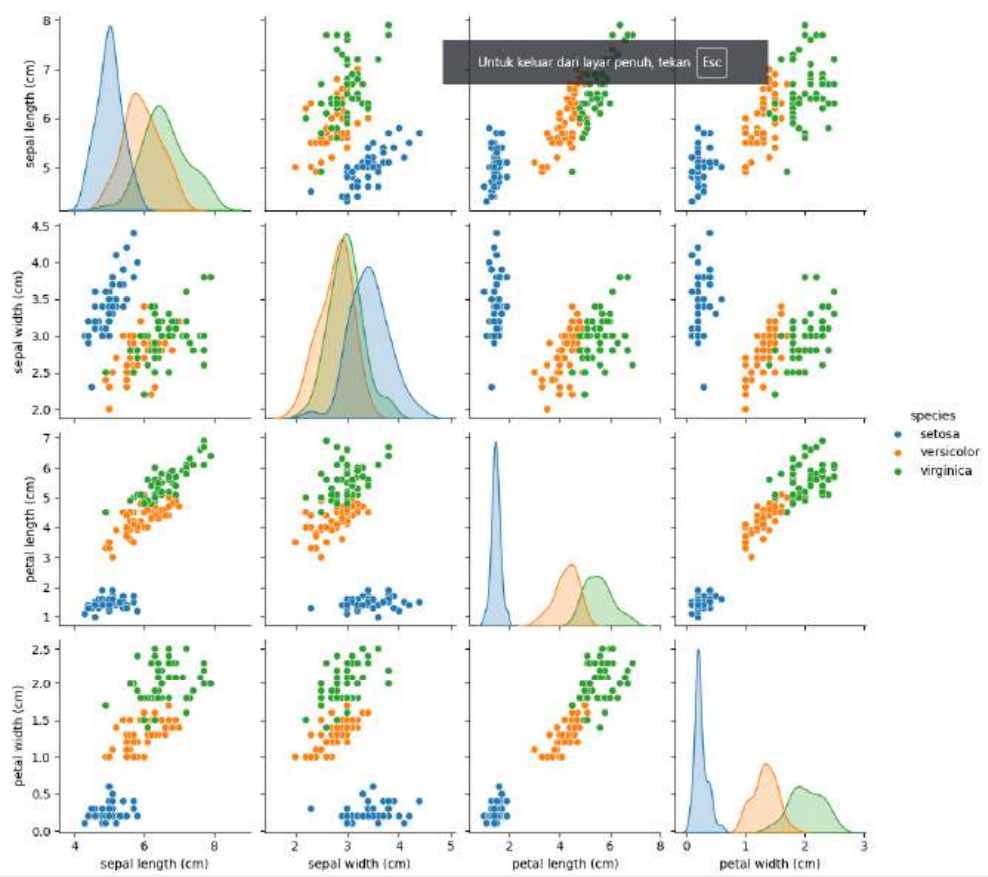
```
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
```

Langkah 2: Visualisasi Data

Kita akan melakukan visualisasi data untuk mendapatkan wawasan lebih lanjut mengenai distribusi dan hubungan antar fitur.

```
# Pairplot
sns.pairplot(data, hue='species')
plt.show()

# Heatmap of correlation
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()

# Boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(x='species', y='sepal length (cm)', data=data)
plt.show()

# Violin plot
plt.figure(figsize=(12, 6))
sns.violinplot(x='species', y='sepal width (cm)', data=data)
plt.show()
```

Output:



Langkah 3: Pelatihan Model

Kita akan membagi dataset menjadi data pelatihan dan pengujian, kemudian melatih model menggunakan Logistic Regression.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Split data into training and testing sets
X = data.drop('species', axis=1)
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Train Logistic Regression model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)
```

Langkah 4: Evaluasi Model

Kita akan mengevaluasi model menggunakan metrik evaluasi seperti confusion matrix dan classification report.

Confusion matrix

```
conf_matrix = confusion_matrix(y_test, y_pred) print("Confusion Matrix:\n", conf_matrix)
```

Classification report

```
class_report = classification_report(y_test, y_pred) print("Classification Report:\n", class_report)
```

Dengan langkah-langkah di atas, kita telah melakukan eksplorasi data, visualisasi, pelatihan, dan pengujian model menggunakan dataset Iris. Berikut adalah ringkasan dari tiap langkah:

1. Eksplorasi Data: Memahami struktur dasar dan statistik deskriptif dari dataset.
2. Visualisasi Data: Mendapatkan wawasan lebih dalam mengenai distribusi fitur dan hubungan antar fitur.
3. Pelatihan Model: Melatih model machine learning (Logistic Regression) pada data pelatihan.
4. Evaluasi Model: Mengevaluasi kinerja model menggunakan metrik evaluasi standar.

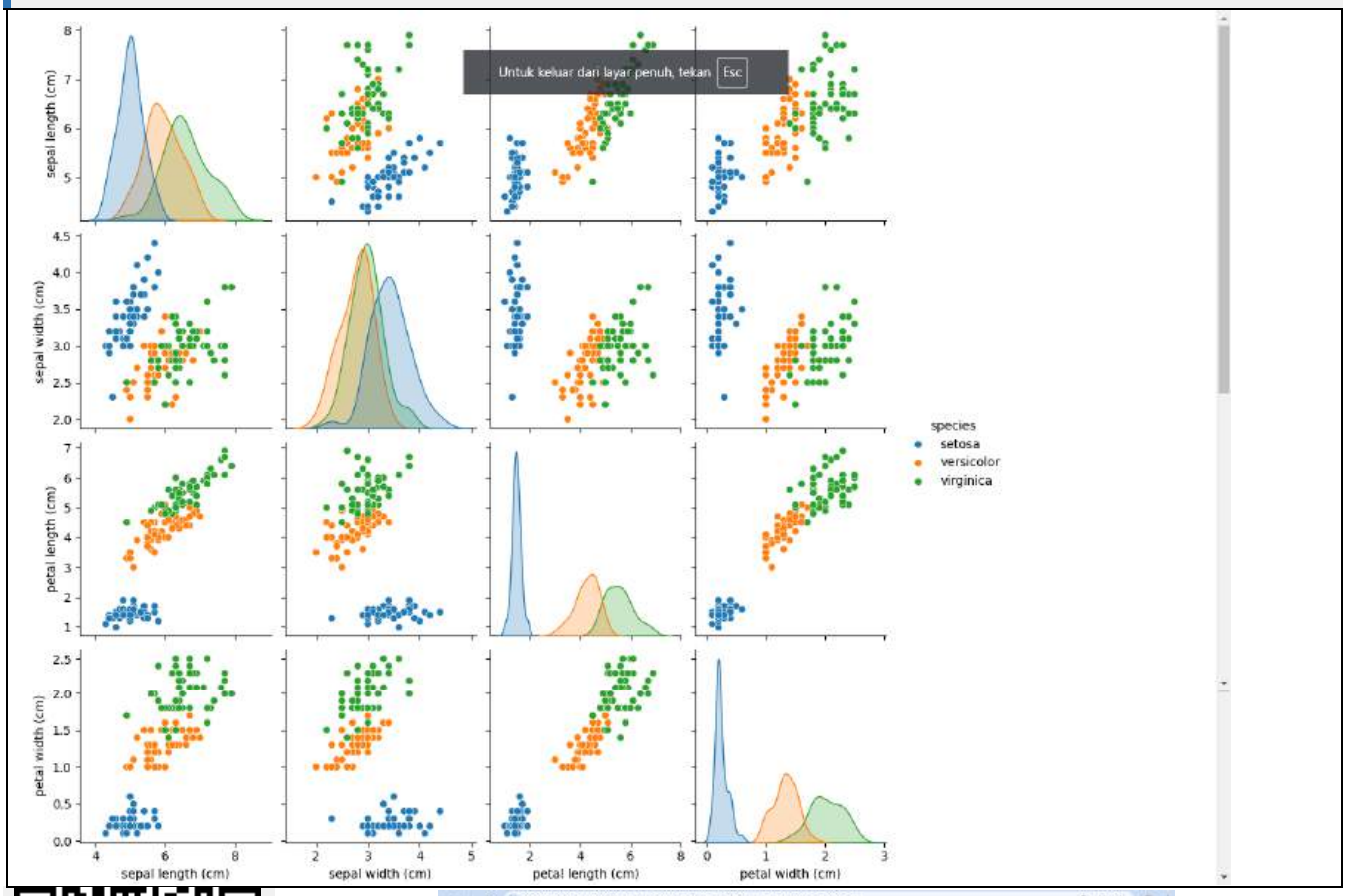
```
# Kode Lengkap
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Load dataset
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['species'] = iris.target
data['species'] = data['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

# Display first few rows
print(data.head())

# Summary statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Pairplot
sns.pairplot(data, hue='species')
plt.show()

# Heatmap of correlation
```

```
    plt.figure(figsize=(10, 8))
    sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
    plt.show()

    # Boxplot
    plt.figure(figsize=(12, 6))
    sns.boxplot(x='species', y='sepal length (cm)', data=data)
    plt.show()

    # Violin plot
    plt.figure(figsize=(12, 6))
    sns.violinplot(x='species', y='sepal width (cm)', data=data)
    plt.show()

    # Split data into training and testing sets
    X = data.drop('species', axis=1)
    y = data['species']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=42)

    # Train Logistic Regression model
    model = LogisticRegression(max_iter=200)
    model.fit(X_train, y_train)

    # Predict on test set
    y_pred = model.predict(X_test)

    # Confusion matrix
    conf_matrix = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:\n", conf_matrix)

    # Classification report
    class_report = classification_report(y_test, y_pred)
    print("Classification Report:\n", class_report)
```

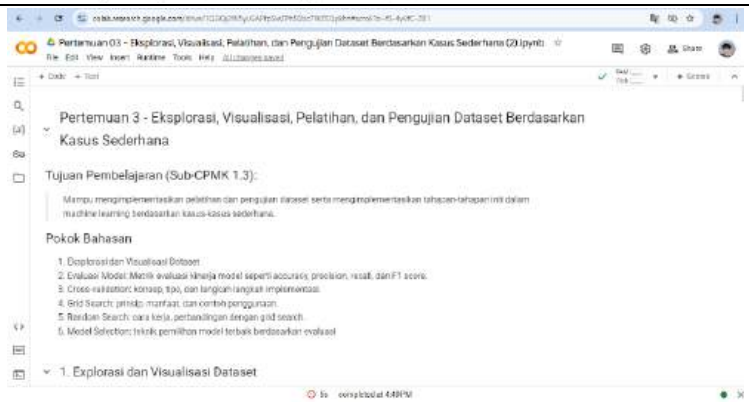
Output:

```
  sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
```

```
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
```



Scan disi atau klik gambar di samping untuk mengakses Google Colab pembelajaran



3.8. Perbedaan Supervised, Unsupervised, dan Reinforcement Learning

Pada kesempatan kali ini, kita akan membahas mengenai perbedaan antara tiga jenis utama metode pembelajaran dalam machine learning: Supervised Learning, Unsupervised Learning, dan Reinforcement Learning. Setiap metode memiliki pendekatan yang berbeda dalam memanfaatkan data dan memberikan hasil. Supervised Learning memerlukan data berlabel untuk mengarahkan model pada pola tertentu. Unsupervised Learning, sebaliknya, bekerja dengan data tanpa label untuk mengidentifikasi pola tersembunyi. Sementara itu, Reinforcement Learning memungkinkan model belajar melalui umpan balik berbasis hadiah atau hukuman dari lingkungan. Dengan memahami perbedaan mendasar ini, kita akan lebih siap dalam memilih metode yang tepat sesuai dengan kebutuhan dan tujuan pembelajaran model.

3.9. Supervised vs Unsupervised

Pada materi ini, kita akan membahas perbedaan antara pembelajaran terawasi (supervised learning) dan pembelajaran tanpa pengawasan (unsupervised learning) dalam machine learning. Supervised learning menggunakan data yang berlabel untuk melatih model agar mampu melakukan prediksi atau klasifikasi. Sebaliknya, unsupervised learning bekerja dengan data yang tidak berlabel dan bertujuan untuk menemukan pola atau struktur yang tersembunyi dalam data tersebut. Pemahaman tentang kedua pendekatan ini akan membantu dalam memilih metode yang sesuai dengan jenis data dan tujuan analisis yang diinginkan.

3.10. Konsep Supervised, Unsupervised, dan Reinforcement Learning

Video ini akan membahas tiga pendekatan utama dalam machine learning: Supervised, Unsupervised, dan Reinforcement Learning. Supervised Learning terjadi ketika model dilatih dengan data berlabel, yang memungkinkan model untuk memprediksi keluaran berdasarkan data baru. Pendekatan ini sering digunakan untuk tugas seperti klasifikasi dan regresi. Sebaliknya, Unsupervised Learning bekerja dengan data tanpa label dan bertujuan untuk mengidentifikasi pola atau kelompok dalam data, seperti pada pengelompokan atau pengurangan dimensi. Reinforcement Learning melibatkan agen yang berinteraksi dengan lingkungan dan belajar melalui proses trial and error untuk mencapai suatu tujuan tertentu, yang biasa diterapkan pada tugas yang membutuhkan pengambilan keputusan berurutan.

Pemahaman mendasar tentang ketiga pendekatan ini sangat penting untuk mendalami machine learning lebih lanjut.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

3.11. Peran Machine Learning pada Masa Mendatang

Video ini membahas bagaimana machine learning akan menjadi elemen kunci dalam perkembangan masa depan dan transformasi industri. Machine learning diharapkan memainkan peran besar dalam otomasi dan pengambilan keputusan yang cepat di berbagai sektor, termasuk kesehatan, keamanan siber, transportasi, dan pendidikan. Teknologi ini akan mendukung peningkatan efisiensi dalam diagnosis penyakit, deteksi serangan siber, dan pengembangan kendaraan otonom, yang memungkinkan sistem bekerja lebih pintar dan efektif. Seiring berjalannya waktu, inovasi dalam machine learning, seperti kecepatan pembelajaran yang lebih tinggi dan peningkatan akurasi model, akan membuatnya semakin penting dalam mendukung perkembangan teknologi dan memenuhi kebutuhan sehari-hari.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 4. Telaah dan Penguraian Dataset

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan Exploratory Data Analysis (EDA).
2. Teknik-teknik dasar EDA.
3. Pengenalan Principal Component Analysis (PCA).
4. Aplikasi PCA dalam analisis data.
5. Identifikasi pola dan anomali.

4.1. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) adalah proses untuk memahami data secara mendalam sebelum melakukan analisis lebih lanjut atau pemodelan. Tujuan EDA adalah untuk meringkas karakteristik utama data dan mengidentifikasi pola, anomali, atau hubungan penting yang mungkin tidak terlihat pada pandangan pertama. Berikut adalah langkah-langkah dan contoh EDA menggunakan dataset Iris. Berikut ini adalah langkah-langkah EDA:

1. Memuat Data
2. Memeriksa Struktur Data
3. Statistik Deskriptif
4. Visualisasi Data
5. Pemeriksaan Outliers dan Missing Values
6. Memeriksa Korelasi

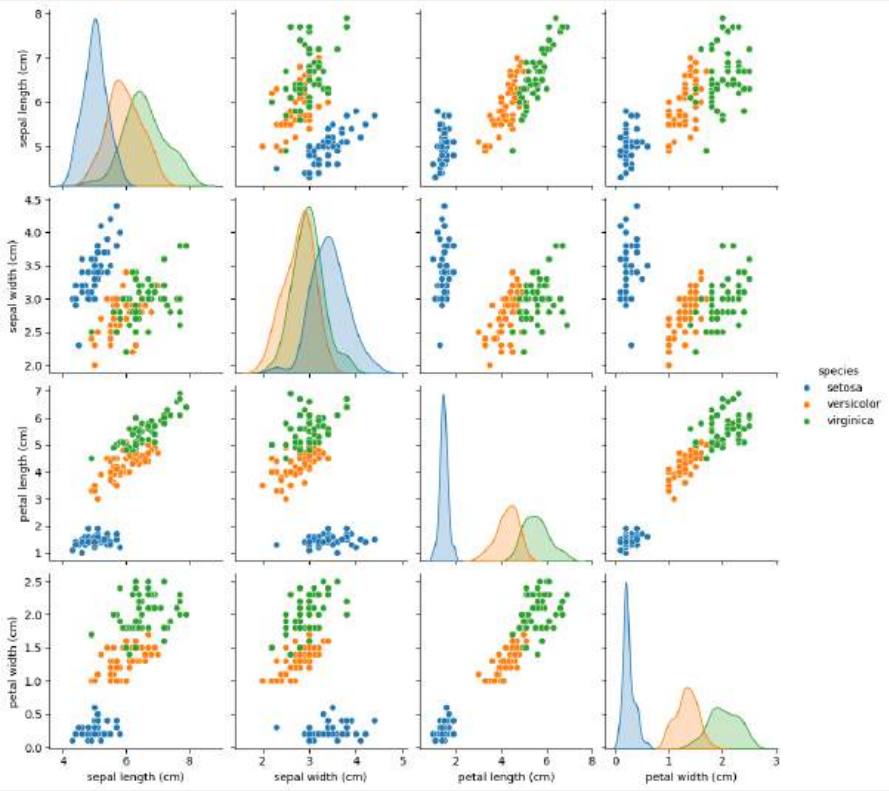
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# 1. Memuat Data
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['species'] = iris.target

# Map target integers to species names
data['species'] = data['species'].map({0: 'setosa', 1: 'versicolor', 2:
'virginica'})

# 2. Memeriksa Struktur Data
print(data.head())
print(data.info())
```

```
print(data['species'].value_counts())

# 3. Statistik Deskriptif
print(data.describe())

# 4. Visualisasi Data

# Pairplot untuk melihat distribusi dan hubungan antar fitur
sns.pairplot(data, hue='species')
plt.show()

# Heatmap untuk melihat korelasi antar fitur
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()

# Boxplot untuk melihat distribusi fitur berdasarkan kelas
plt.figure(figsize=(12, 6))
sns.boxplot(x='species', y='sepal length (cm)', data=data)
plt.show()

# Violin plot untuk melihat distribusi fitur dengan lebih detail
plt.figure(figsize=(12, 6))
sns.violinplot(x='species', y='sepal width (cm)', data=data)
plt.show()

# 5. Pemeriksaan Outliers dan Missing Values

# Memeriksa missing values
print(data.isnull().sum())

# Menggunakan boxplot untuk mendeteksi outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=data)
plt.show()

# 6. Memeriksa Korelasi

# Heatmap korelasi (sudah ditampilkan sebelumnya)
```

Output:

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
```

```
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
```



EDA adalah langkah penting dalam setiap analisis data karena membantu kita memahami data dengan lebih baik dan mengidentifikasi masalah potensial sebelum melakukan pemodelan atau analisis lebih lanjut. Dengan langkah-langkah EDA di atas, kita dapat memperoleh wawasan mendalam tentang dataset Iris dan mempersiapkan data untuk analisis lebih lanjut atau pemodelan machine learning.

4.2. Teknik-Teknik EDA

Eksploratory Data Analysis (EDA) adalah proses analisis data yang dilakukan sebelum pemodelan untuk memahami karakteristik dan struktur data. EDA mencakup langkah-langkah seperti pemahaman data, visualisasi data, pengujian statistik dasar, dan deteksi outlier. Berikut adalah contoh EDA secara komprehensif menggunakan dataset Iris. Berikut ini adalah langkah-langkah EDA:

1. Memuat Data
2. Memeriksa Struktur Data
3. Statistik Deskriptif
4. Visualisasi Data
5. Pemeriksaan Outliers dan Missing Values
6. Memeriksa Korelasi
7. Insight dan Kesimpulan

A. Memuat Data

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['species'] = iris.target

# Map target integers to species names
data['species'] = data['species'].map({0: 'setosa', 1: 'versicolor', 2:
'virginica'})
```

B. Memeriksa Struktur Data

```
# Display first few rows
print(data.head())

# Information about the dataset
print(data.info())
```

```
# Distribution of species
print(data['species'].value_counts())
```

Output:

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

C. Statistik Deskriptif

```
# Summary statistics
print(data.describe())
```

Output:

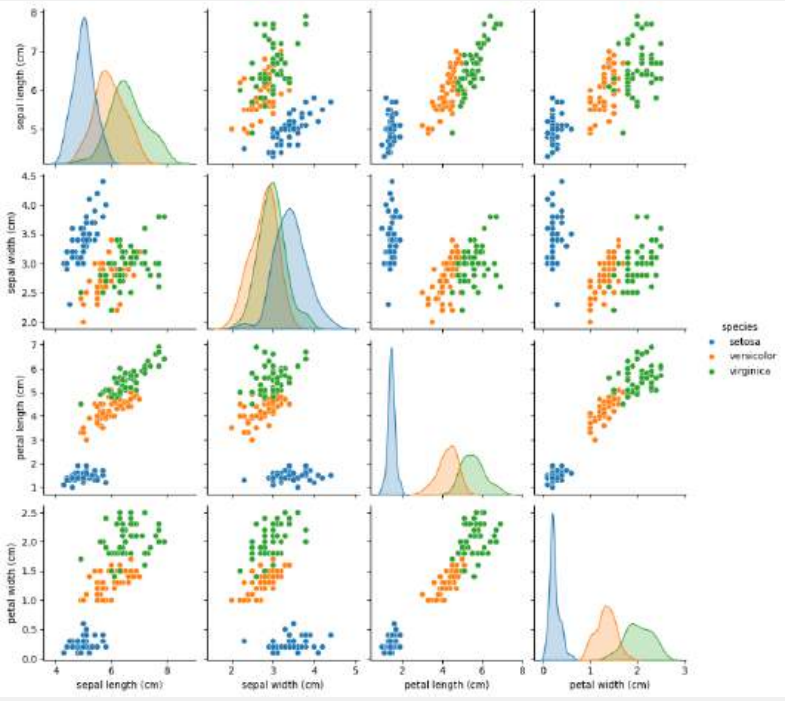
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
```

```
min             0.100000
25%             0.300000
50%             1.300000
75%             1.800000
max             2.500000
addCode
addText
```

D. Visualisasi Data

```
# Pairplot untuk melihat distribusi dan hubungan antar fitur
sns.pairplot(data, hue='species')
plt.show()

# Heatmap untuk melihat korelasi antar fitur
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()

# Boxplot untuk melihat distribusi fitur berdasarkan kelas
plt.figure(figsize=(12, 6))
sns.boxplot(x='species', y='sepal length (cm)', data=data)
plt.show()

# Violin plot untuk melihat distribusi fitur dengan lebih detail
plt.figure(figsize=(12, 6))
sns.violinplot(x='species', y='sepal width (cm)', data=data)
plt.show()
```

Output:

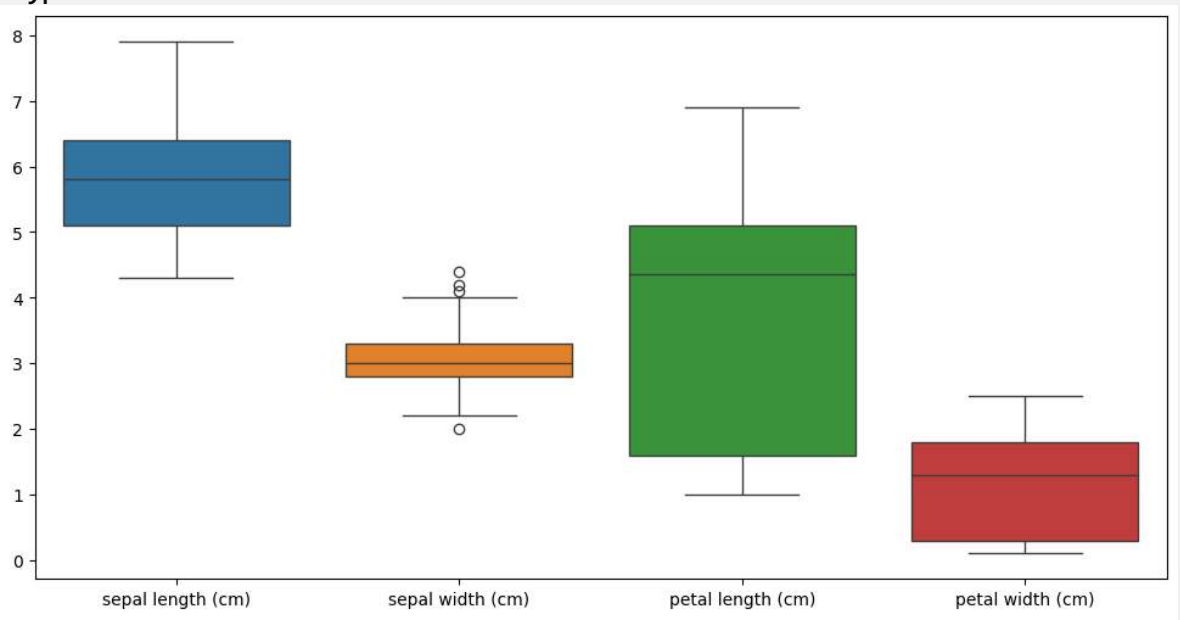


E. Pemeriksaan Outliers dan Missing Values

```
# Memeriksa missing values
print(data.isnull().sum())

# Menggunakan boxplot untuk mendeteksi outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=data)
plt.show()
```

Output:

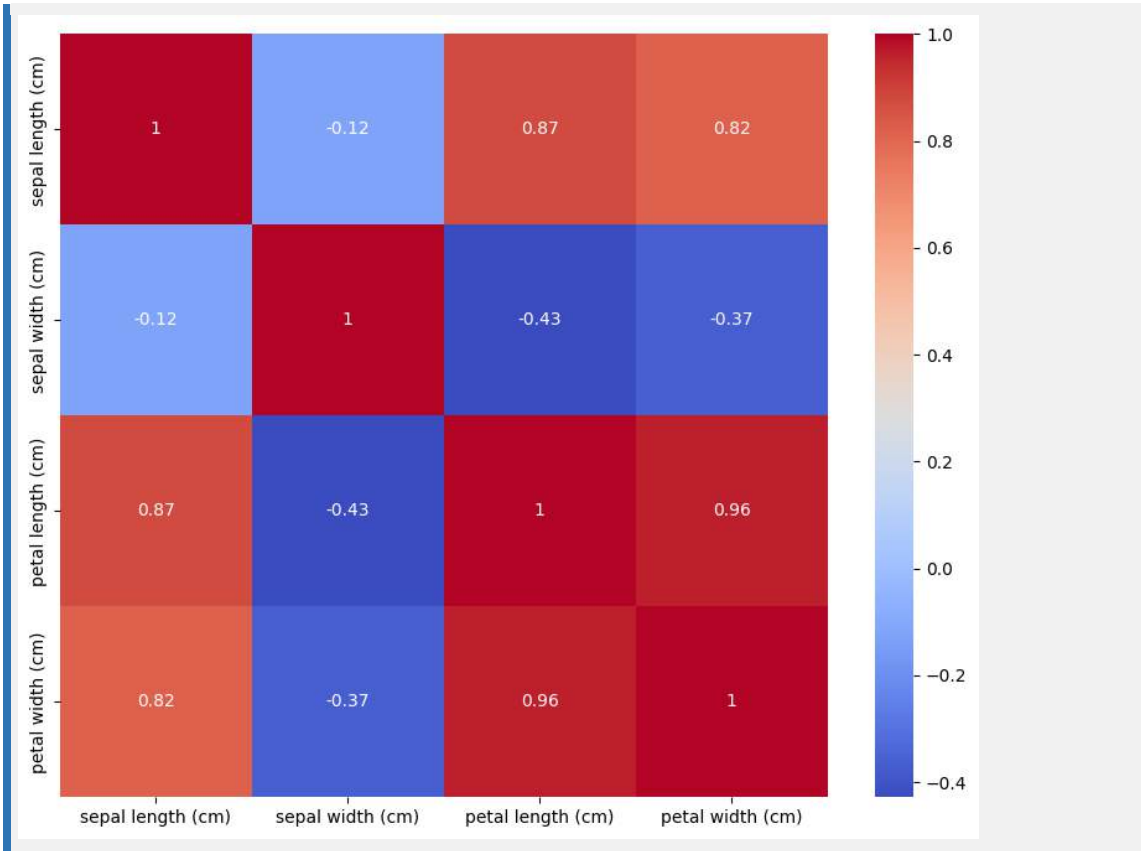
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
```



F. Memeriksa Korelasi

```
# Heatmap korelasi
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()
```

Output:



G. Insight dan Kesimpulan

Setelah melakukan eksplorasi data, kita bisa menarik beberapa insight dan kesimpulan.

1. Distribusi Kelas: Setiap kelas bunga memiliki jumlah data yang seimbang.
2. Korelasi Fitur: Fitur sepal length dan petal length memiliki korelasi positif yang kuat
3. Visualisasi: Pairplot menunjukkan bahwa kelas setosa dapat dipisahkan dengan baik dari kelas lainnya berdasarkan petal length dan petal width.

Dengan melakukan EDA, kita mendapatkan pemahaman yang lebih baik tentang dataset. EDA membantu mengidentifikasi pola, anomali, dan hubungan yang penting dalam data. Hal ini sangat penting sebelum melanjutkan ke tahap pemodelan atau analisis lanjutan.

```
# Contoh Lengkap
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# 1. Memuat Data
iris = load_iris()
```

```
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['species'] = iris.target
data['species'] = data['species'].map({0: 'setosa', 1: 'versicolor', 2:
'virginica'})

# 2. Memeriksa Struktur Data
print(data.head())
print(data.info())
print(data['species'].value_counts())

# 3. Statistik Deskriptif
print(data.describe())

# 4. Visualisasi Data
sns.pairplot(data, hue='species')
plt.show()

plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(x='species', y='sepal length (cm)', data=data)
plt.show()

plt.figure(figsize=(12, 6))
sns.violinplot(x='species', y='sepal width (cm)', data=data)
plt.show()

# 5. Pemeriksaan Outliers dan Missing Values
print(data.isnull().sum())

plt.figure(figsize=(12, 6))
sns.boxplot(data=data)
plt.show()

# 6. Memeriksa Korelasi
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()

# 7. Insight dan Kesimpulan
# Distribusi Kelas
print("Distribusi Kelas:\n", data['species'].value_counts())

# Korelasi Fitur
print("Korelasi Fitur:\n", data.corr())
```

4.3. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) adalah teknik statistik yang digunakan untuk mengurangi dimensi data yang tinggi dengan cara mengidentifikasi komponen utama yang menjelaskan varians terbesar dalam data. PCA membantu dalam menemukan struktur mendasar dalam data dan mengurangi redundansi, membuat visualisasi dan pemrosesan data lebih mudah. Berikut ini adalah langkah-langkah PCA:

1. Standarisasi Data: Menstandarisasi data sehingga setiap fitur memiliki mean 0 dan varians 1.

2. Membangun Matrik Covarian: Menghitung matriks kovarian dari data yang distandarisasi untuk memahami hubungan antar fitur.
3. Menentukan Eigenvalue dan Eigenvector: Menghitung eigenvalue dan eigenvector dari matriks kovarian untuk menemukan arah utama (principal components).
4. Memilih Komponen Utama: Memilih beberapa komponen utama berdasarkan eigenvalue terbesar
5. Membuat Data Proyeksi Baru: Memproyeksikan data asli ke ruang baru yang dibentuk oleh komponen utama.
6. Visualisasi Komponen Utama: Visualisasi data dalam ruang komponen utama untuk analisis lebih lanjut.

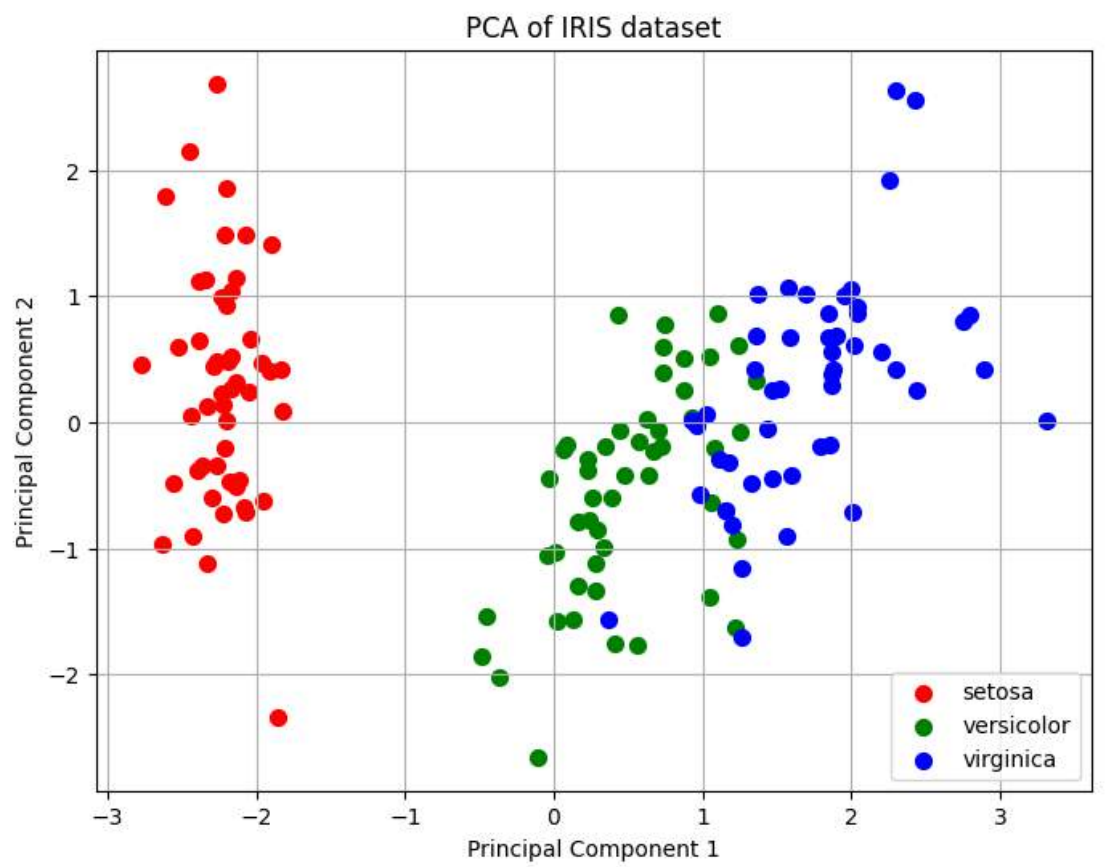
Contoh PCA dengan Dataset Iris

rincipal Component Analysis (PCA) adalah teknik yang digunakan untuk mengurangi dimensi dari dataset. Ini sangat berguna dalam mengidentifikasi pola dalam data dan mengungkapkan hubungan antara variabel-variabel. Berikut adalah contoh implementasi PCA dengan menggunakan dataset Iris:

```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Standardize the data
X = StandardScaler().fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X)
principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Concatenate with the target
final_df = pd.concat([principal_df, pd.DataFrame(y, columns=['target'])], axis=1)

# Plot the results
plt.figure(figsize=(8, 6))
targets = [0, 1, 2]
colors = ['r', 'g', 'b']
for target, color in zip(targets, colors):
    indices_to_keep = final_df['target'] == target
    plt.scatter(final_df.loc[indices_to_keep, 'PC1'],
                final_df.loc[indices_to_keep, 'PC2'],
```

```
                c=color,
                s=50)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of IRIS dataset')
plt.legend(iris.target_names)
plt.grid()
plt.show()
```

Output:



4.4. Implementasi EDA dan PCA menggunakan Python

Implementasi Exploratory Data Analysis (EDA) dan Principal Component Analysis (PCA) menggunakan dataset Iris dengan Python dapat dilakukan dalam beberapa langkah. Berikut adalah contoh lengkapnya:

Exploratory Data Analysis (EDA)

1. Import Libraries: Kita perlu mengimpor pustaka yang diperlukan.
2. Load Dataset: Muat dataset Iris.
3. Descriptive Statistics: Tampilkan statistik deskriptif.
4. Data Visualization: Visualisasikan data untuk memahami pola dan hubungan.

Principal Component Analysis (PCA)

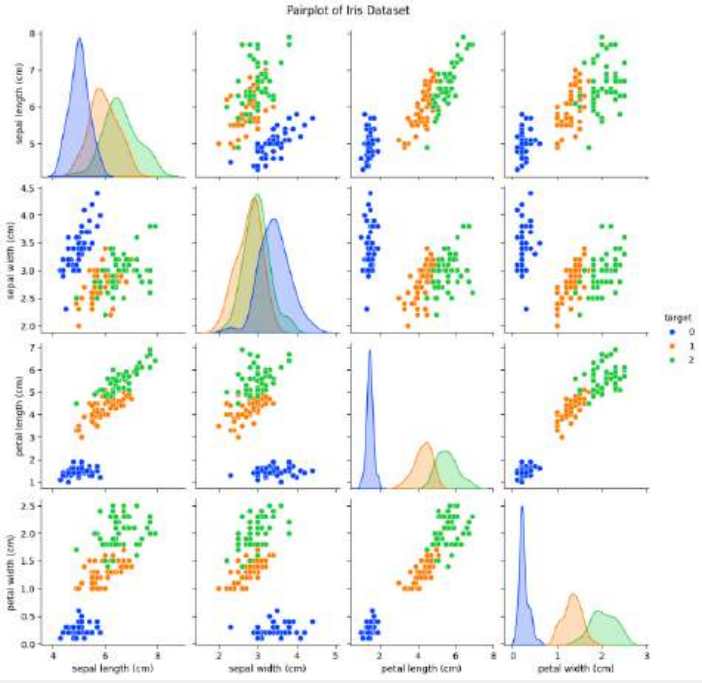
1. Standardize the Data: Standarisasi data.
2. Apply PCA: Terapkan PCA untuk mengurangi dimensi.
3. Visualize PCA Results: Visualisasikan hasil PCA.

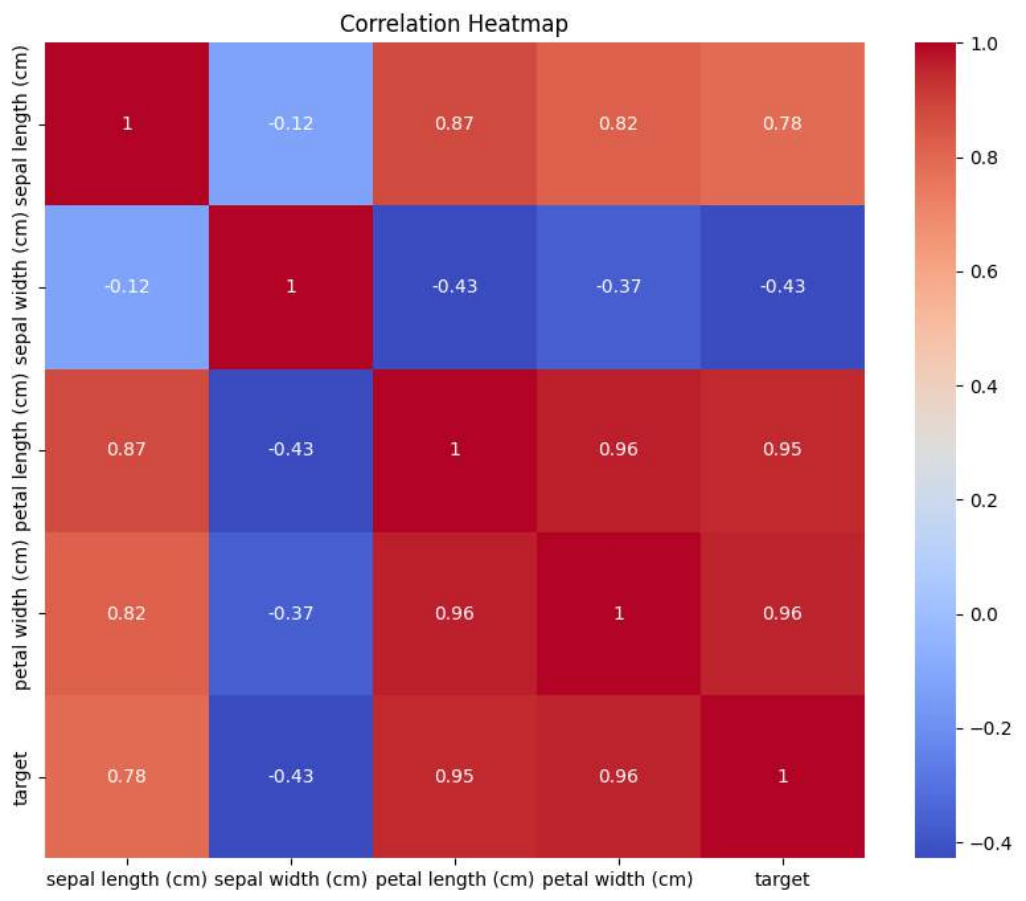
```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the Iris dataset
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['target'] = iris.target

# Exploratory Data Analysis (EDA)
# Descriptive statistics
print(data.describe())

# Pairplot
sns.pairplot(data, hue='target', palette='bright')
plt.suptitle('Pairplot of Iris Dataset', y=1.02)
plt.show()

# Heatmap of correlations
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Boxplot
plt.figure(figsize=(12, 8))
sns.boxplot(data=data, orient='h', palette='Set2')
plt.title('Boxplot of Features')
plt.show()

# PCA Implementation
# Standardize the data
X = data.iloc[:, :-1].values
X = StandardScaler().fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X)
principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Concatenate with the target
final_df = pd.concat([principal_df, data[['target']]], axis=1)

# Plot the PCA results
plt.figure(figsize=(10, 8))
targets = [0, 1, 2]
colors = ['r', 'g', 'b']
for target, color in zip(targets, colors):
```

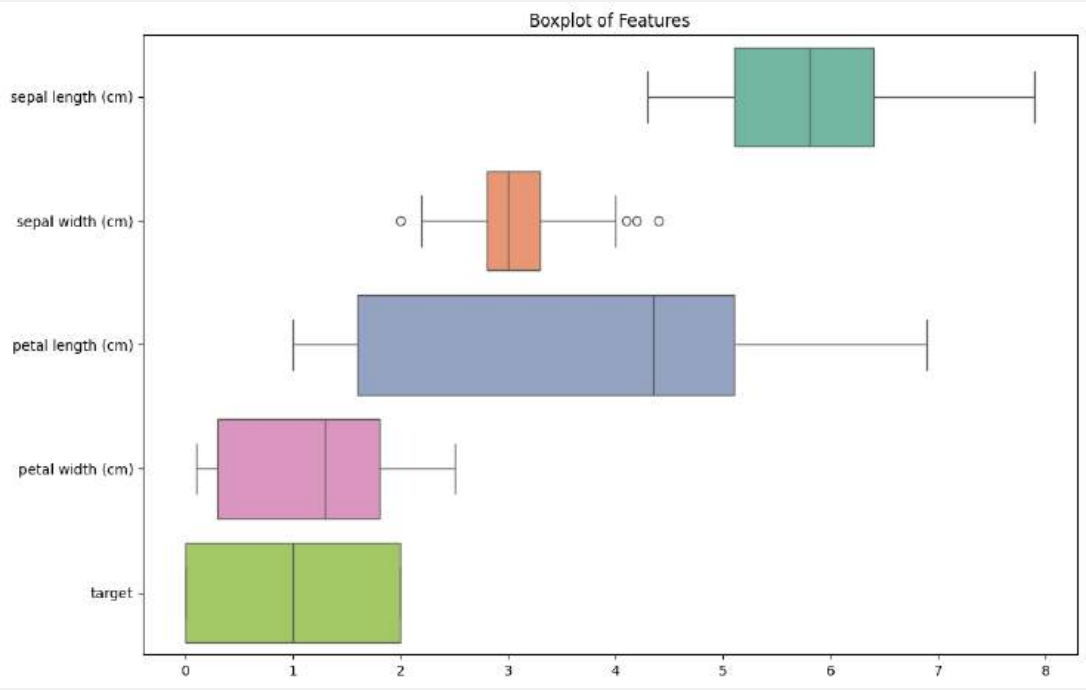
```
    indices_to_keep = final_df['target'] == target
    plt.scatter(final_df.loc[indices_to_keep, 'PC1'],
                final_df.loc[indices_to_keep, 'PC2'],
                c=color,
                s=50)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Iris Dataset')
plt.legend(iris.target_names)
plt.grid()
plt.show()
```

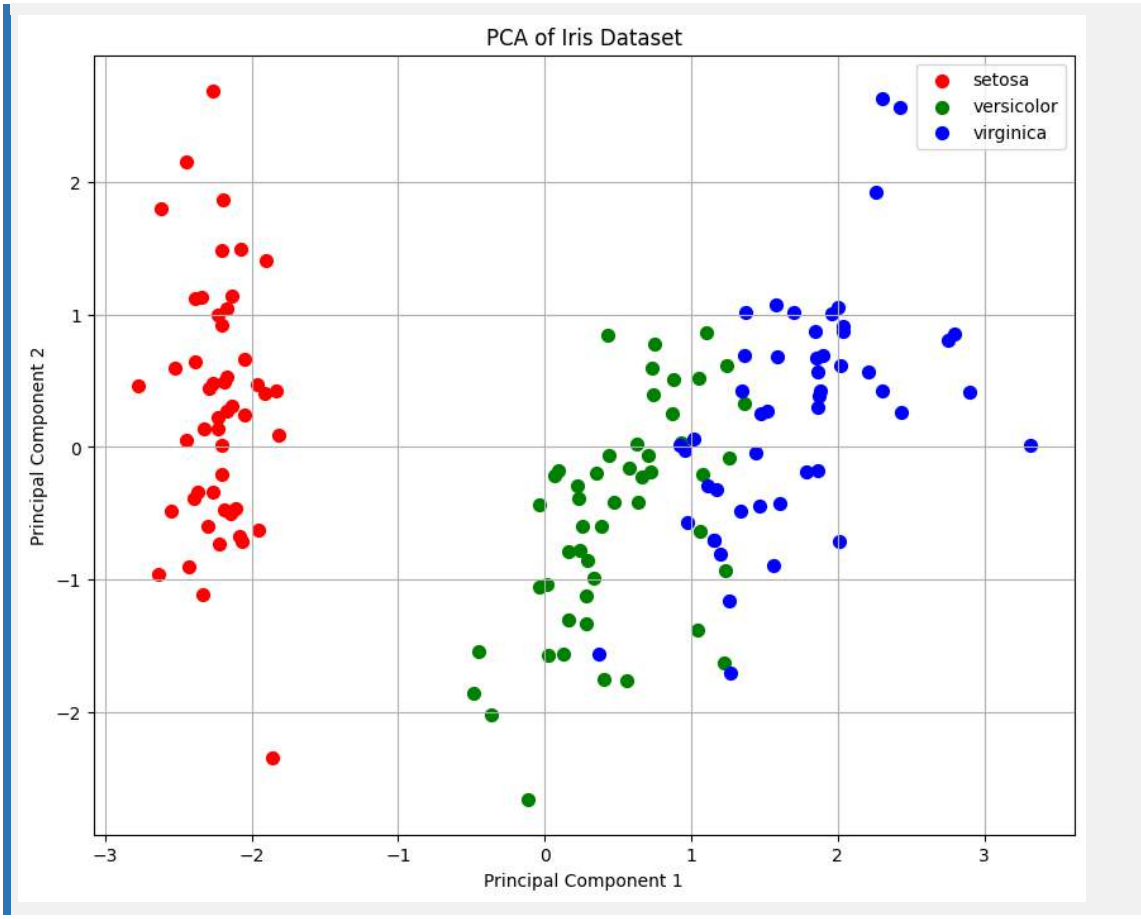
Output:

```
sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)      target
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```




Correlation Heatmap
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
target
1
-0.12
0.87
0.82
0.78
-0.12
1
-0.43
-0.37
-0.43
0.87
-0.43
1
0.96
0.95
0.82
-0.37
0.96
1
0.96
0.78
-0.43
0.95
0.96
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Boxplot of Features
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
target
0
1
2
3
4
5
6
7
8



4.5. Interpretasi Hasil dan Identifikasi Pola serta Anomali yang Relevan

Untuk memberikan interpretasi hasil EDA dan PCA serta mengidentifikasi pola dan anomali yang relevan, kita perlu melihat hasil visualisasi dan statistik yang telah kita buat. Berikut adalah langkah-langkah untuk menginterpretasikan hasil tersebut:

A. Exploratory Data Analysis (EDA)

a. Descriptive Statistics:

- Statistik deskriptif memberikan ringkasan statistik dari dataset. Ini termasuk mean, standar deviasi, min, dan max untuk setiap fitur.
- Misalnya, mean panjang sepal (sepal length) adalah sekitar 5.84 cm, sementara panjang petal (petal length) adalah sekitar 3.76 cm. Standar deviasi memberikan indikasi variasi dalam data.

b. Pairplot:

- Pairplot menunjukkan hubungan antar fitur dan bagaimana data tersebar berdasarkan kelas target.

- Kita dapat melihat bahwa fitur petal length dan petal width memberikan pemisahan yang jelas antara spesies. Spesies setosa dapat dibedakan dengan jelas dari versicolor dan virginica.

c. Heatmap of Correlations:

- Peta panas korelasi menunjukkan seberapa kuat hubungan antar fitur.
- Misalnya, petal length dan petal width memiliki korelasi positif yang sangat tinggi (sekitar 0.96), menunjukkan bahwa ketika satu meningkat, yang lain juga cenderung meningkat.
- Fitur sepal length dan sepal width memiliki korelasi yang rendah (sekitar 0.11), menunjukkan bahwa mereka tidak terlalu berkaitan.

d. Boxplot:

- Boxplot menunjukkan distribusi data dan membantu mengidentifikasi outlier.
- Sebagai contoh, kita dapat melihat bahwa sepal width memiliki beberapa outlier di bagian bawah (nilai rendah) dan di bagian atas (nilai tinggi).

Principal Component Analysis (PCA)

a. Explained Variance

PCA mengubah data ke dalam komponen utama. Dua komponen utama pertama menjelaskan sebagian besar varians dalam data. Misalnya, dalam contoh di atas, dua komponen utama pertama menjelaskan sekitar 95% dari varians total, yang berarti sebagian besar informasi dalam data asli telah dipertahankan dalam dua komponen utama ini.

b.PCA Scatter Plot

- Plot PCA menunjukkan bagaimana data tersebar dalam dua dimensi berdasarkan komponen utama.
- Kita dapat melihat bahwa spesies setosa terpisah dengan jelas dari versicolor dan virginica, yang juga menunjukkan beberapa pemisahan tetapi tidak sejelas setosa.
- Ini menunjukkan bahwa PCA berhasil mengurangi dimensi data dan mempertahankan informasi yang cukup untuk membedakan antara spesies.

Identifikasi Pola dan Anomali

1. Pola

- Spesies setosa dapat dibedakan dengan jelas berdasarkan petal length dan petal width.

- versicolor dan virginica memiliki tumpang tindih, tetapi versicolor cenderung memiliki petal length dan petal width yang lebih rendah dibandingkan virginica.

2. Anomali

- Dalam boxplot, kita melihat beberapa outlier, terutama pada sepal width. Outlier ini mungkin perlu diteliti lebih lanjut untuk memahami apakah mereka adalah data yang valid atau hasil dari kesalahan pengukuran.
- Outlier dapat mempengaruhi analisis dan model prediktif, sehingga penting untuk menangani mereka dengan hati-hati.

Dengan interpretasi ini, kita dapat memahami lebih dalam tentang struktur data dan pola yang ada dalam dataset Iris. Hal ini juga membantu dalam mengidentifikasi anomali yang mungkin memerlukan penanganan lebih lanjut dalam analisis data atau model prediktif.

4.6. Contoh Kasus

Mari kita coba contoh kasus yang komprehensif dengan dataset yang berbeda. Kali ini kita akan menggunakan dataset Diabetes dari scikit-learn. Dataset Diabetes mengandung berbagai pengukuran medis untuk pasien, termasuk usia, jenis kelamin, BMI, tekanan darah, dan beberapa ukuran darah lainnya. Targetnya adalah ukuran progresi penyakit diabetes satu tahun setelah pengukuran.

Studi Kasus 1: Analisis Data Diabetes

Tujuan

1. Melakukan Exploratory Data Analysis (EDA) untuk memahami distribusi, korelasi, dan pola dalam data.
2. Menggunakan Principal Component Analysis (PCA) untuk mengurangi dimensi dan memvisualisasikan data dalam 2D.
3. Mengidentifikasi pola dan anomali dalam data.

Langkah-Langkah

1. Import Libraries: Mengimpor pustaka yang diperlukan
2. Load Dataset: Memuat dataset Diabetes
3. Exploratory Data Analysis (EDA)
    a. Descriptive Statistics
    b. Pairplot
    c. Heatmap of Correlations
    d. Boxplot

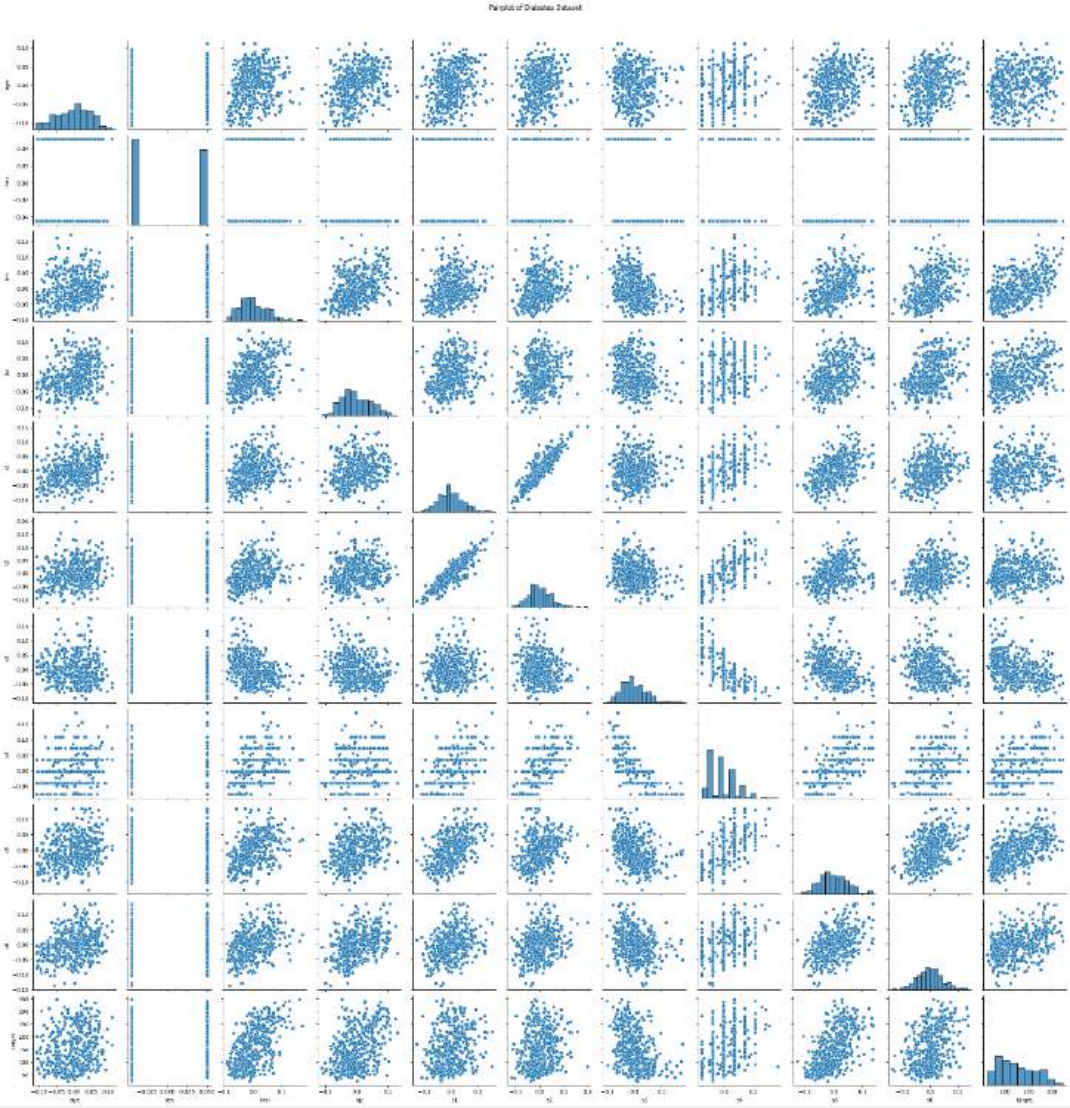
4. Principal Component Analysis (PCA)
    a. Standarisasi Data
    b. Aplikasi PCA
    c. Visualisasi Hasil PCA
5. Interpretasi Hasil dan Identifikasi Pola serta Anomali
    a. Analisis Descriptive Statistics
    b. Interpretasi Pairplot
    c. Analisis Heatmap of Correlations
    d. Analisis Boxplot
    e. Analisis Visualisasi PCA

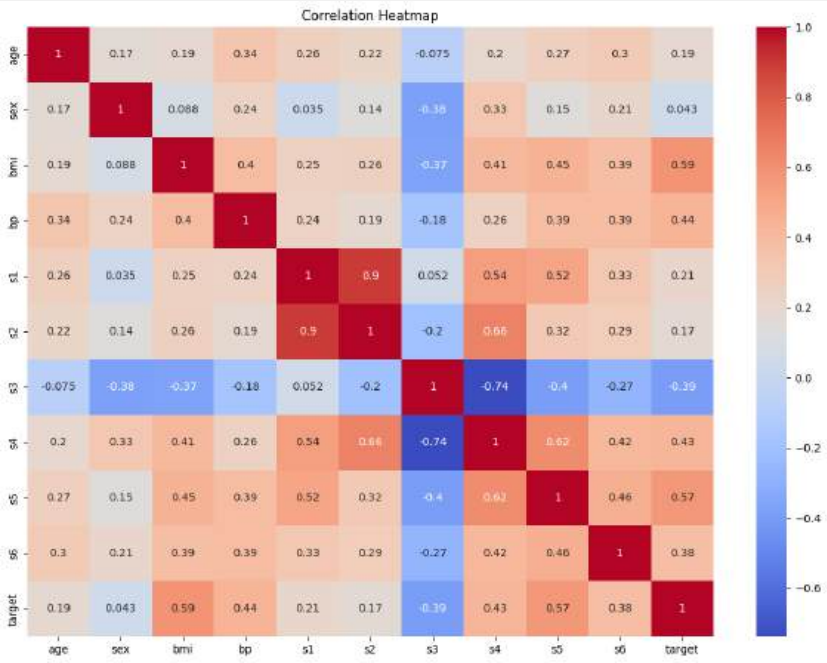
```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_diabetes
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the Diabetes dataset
diabetes = load_diabetes()
data = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)
data['target'] = diabetes.target

# Exploratory Data Analysis (EDA)
# Descriptive statistics
print("Descriptive Statistics:")
print(data.describe())

# Pairplot
sns.pairplot(data)
plt.suptitle('Pairplot of Diabetes Dataset', y=1.02)
plt.show()

# Heatmap of correlations
plt.figure(figsize=(14, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Boxplot
plt.figure(figsize=(14, 10))
sns.boxplot(data=data, orient='h', palette='Set2')
plt.title('Boxplot of Features')
plt.show()

# PCA Implementation
# Standardize the data
X = data.iloc[:, :-1].values
```

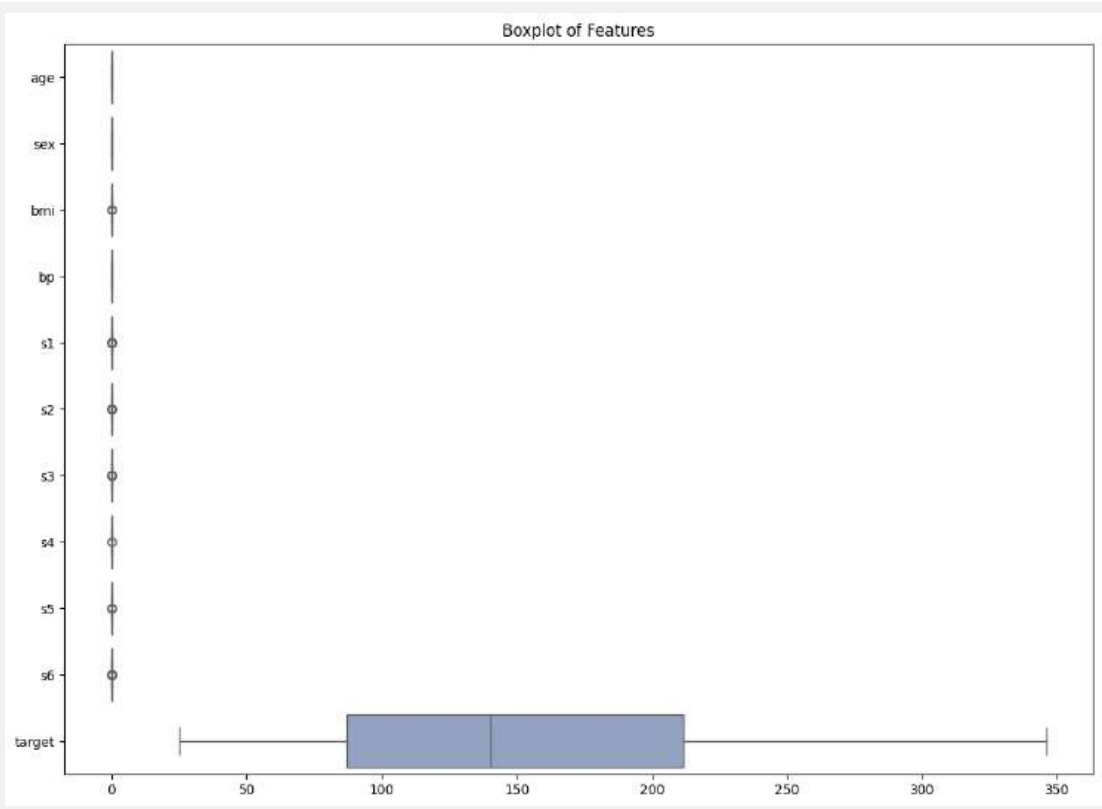
```
X = StandardScaler().fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X)
principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Concatenate with the target
final_df = pd.concat([principal_df, data[['target']]], axis=1)

# Plot the PCA results
plt.figure(figsize=(10, 8))
plt.scatter(final_df['PC1'],       final_df['PC2'],       c=final_df['target'],
cmap='viridis', s=50)
plt.colorbar(label='Progression of Diabetes')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Diabetes Dataset')
plt.grid()
plt.show()
```

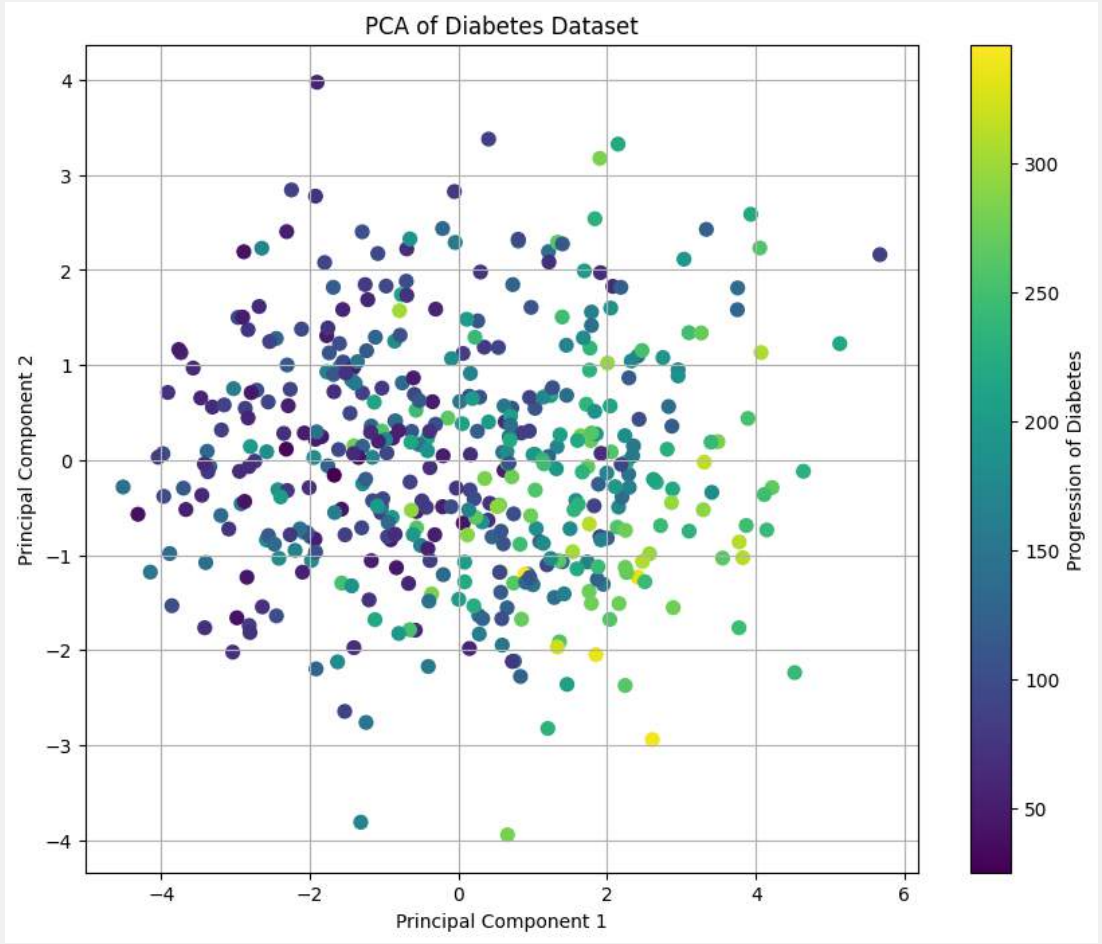
Output:

```
Descriptive Statistics:
                age           sex           bmi            bp            s1  \
count  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02
mean  -2.511817e-19  1.230790e-17 -2.245564e-16 -4.797570e-17 -1.381499e-17
std    4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02
min   -1.072256e-01 -4.464164e-02 -9.027530e-02 -1.123988e-01 -1.267807e-01
25%   -3.729927e-02 -4.464164e-02 -3.422907e-02 -3.665608e-02 -3.424784e-02
50%    5.383060e-03 -4.464164e-02 -7.283766e-03 -5.670422e-03 -4.320866e-03
75%    3.807591e-02  5.068012e-02  3.124802e-02  3.564379e-02  2.835801e-02
max    1.107267e-01  5.068012e-02  1.705552e-01  1.320436e-01  1.539137e-01

                s2            s3            s4            s5            s6  \
count  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02
mean   3.918434e-17 -5.777179e-18 -9.042540e-18  9.293722e-17  1.130318e-17
std    4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02
min   -1.156131e-01 -1.023071e-01 -7.639450e-02 -1.260971e-01 -1.377672e-01
25%   -3.035840e-02 -3.511716e-02 -3.949338e-02 -3.324559e-02 -3.317903e-02
50%   -3.819065e-03 -6.584468e-03 -2.592262e-03 -1.947171e-03 -1.077698e-03
75%    2.984439e-02  2.931150e-02  3.430886e-02  3.243232e-02  2.791705e-02
max    1.987880e-01  1.811791e-01  1.852344e-01  1.335973e-01  1.356118e-01

           target
count  442.000000
mean   152.133484
std     77.093005
min     25.000000
25%     87.000000
50%    140.500000
75%    211.500000
max    346.000000
```




Correlation Heatmap
age 1 0.17 0.19 0.34 0.26 0.22 -0.075 0.2 0.27 0.3 0.19
sex 0.17 1 0.088 0.24 0.035 0.14 -0.38 0.33 0.15 0.21 0.043
bmi 0.19 0.088 1 0.4 0.25 0.26 -0.37 0.41 0.45 0.39 0.59
bp 0.34 0.24 0.4 1 0.24 0.19 -0.18 0.26 0.39 0.39 0.44
s1 0.26 0.035 0.25 0.24 1 0.9 0.052 0.54 0.52 0.33 0.21
s2 0.22 0.14 0.26 0.19 0.9 1 -0.2 0.66 0.32 0.29 0.17
s3 -0.075 -0.38 -0.37 -0.18 0.052 -0.2 1 -0.74 -0.4 -0.27 -0.39
s4 0.2 0.33 0.41 0.26 0.54 0.66 -0.74 1 0.62 0.42 0.43
s5 0.27 0.15 0.45 0.39 0.52 0.32 -0.4 0.62 1 0.46 0.57
s6 0.3 0.21 0.39 0.39 0.33 0.29 -0.27 0.42 0.46 1 0.38
target 0.19 0.043 0.59 0.44 0.21 0.17 -0.39 0.43 0.57 0.38 1
age sex bmi bp s1 s2 s3 s4 s5 s6 target
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6


Boxplot of Features
age
sex
bmi
bp
s1
s2
s3
s4
s5
s6
target
0
50
100
150
200
250
300
350


PCA of Diabetes Dataset
Principal Component 2
Principal Component 1
Progression of Diabetes
4
3
2
1
0
−1
−2
−3
−4
−4
−2
0
2
4
6
300
250
200
150
100
50

Studi Kasus 2: Analisis Data Titanic

Mari kita coba contoh studi kasus yang komprehensif dengan dataset yang berbeda. Kali ini kita akan menggunakan dataset Titanic dari seaborn. Dataset Titanic berisi informasi tentang penumpang Titanic, termasuk apakah mereka selamat atau tidak, kelas tiket mereka, usia, jenis kelamin, dan beberapa informasi lainnya.

Tujuan:

1. Melakukan Exploratory Data Analysis (EDA) untuk memahami distribusi, korelasi, dan pola dalam data.
2. Menggunakan Principal Component Analysis (PCA) untuk mengurangi dimensi dan memvisualisasikan data dalam 2D.
3. Mengidentifikasi pola dan anomali dalam data.

Langkah-Langkah:

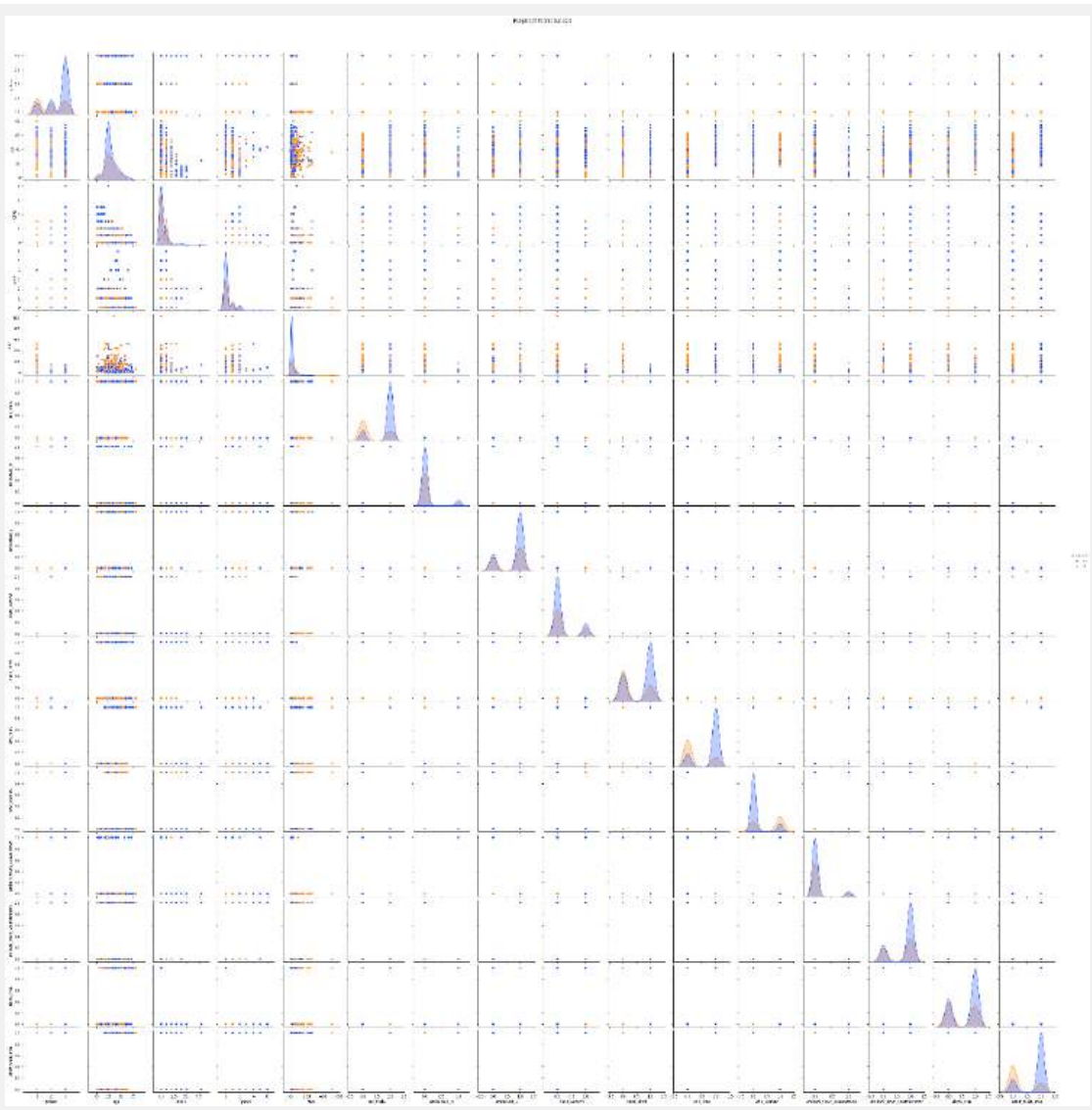
1. Import Libraries: Mengimpor pustaka yang diperlukan.
2. Load Dataset: Memuat dataset Titanic.
3. Data Preprocessing: Menangani missing values dan mengubah data kategorikal menjadi numerik.
4. Exploratory Data Analysis (EDA):
    a. Descriptive Statistics
    b. Pairplot
    c. Heatmap of Correlations
    d. Boxplot
5. Principal Component Analysis (PCA):
    a. Standarisasi Data
    b. Aplikasi PCA
    c. Visualisasi Hasil PCA
6. Interpretasi Hasil dan Identifikasi Pola serta Anomali:
    a. Analisis Descriptive Statistics
    b. Interpretasi Pairplot
    c. Analisis Heatmap of Correlations
    e. Analisis Boxplot
    f. Analisis Visualisasi PCA

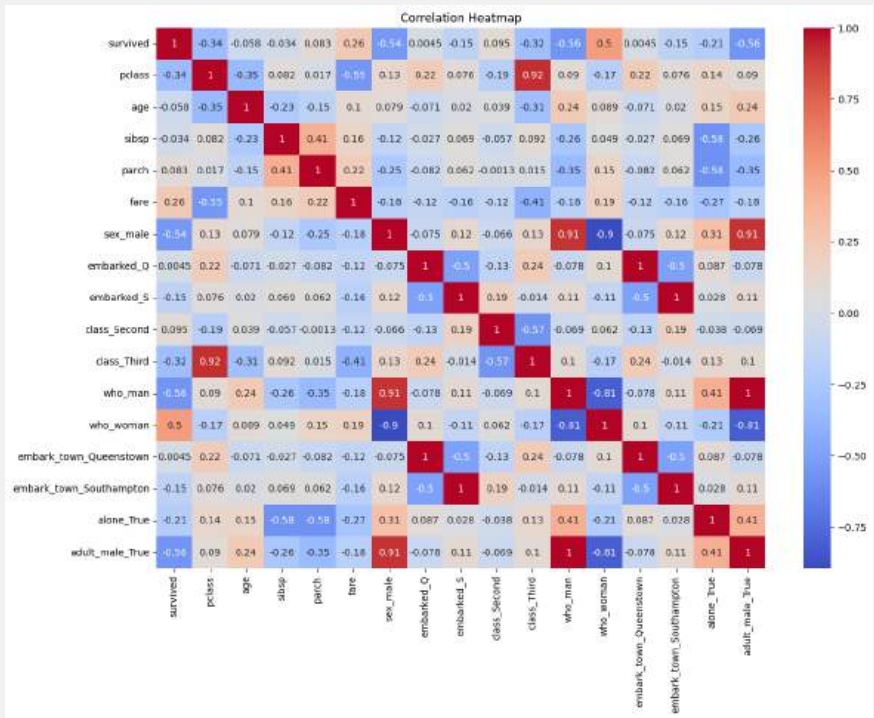
```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decompsition import PCA

# Load the Titanic dataset
data = sns.load_dataset('titanic')

# Data Preprocessing
# Drop rows with missing values in 'embarked' and 'deck' columns
data = data.drop(columns=['deck'])
data = data.dropna(subset=['embarked'])

# Fill missing values in 'age' and 'embark_town' with the mode
data['age'].fillna(data['age'].mode()[0], inplace=True)
data['embark_town'].fillna(data['embark_town'].mode()[0], inplace=True)

# Convert categorical variables to numeric
data = pd.get_dummies(data, columns=['sex', 'embarked', 'class', 'who',
'embark_town', 'alone', 'adult_male'], drop_first=True)

# Drop columns not needed for analysis
data = data.drop(columns=['alive', 'embark_town_Cherbourg',
'embark_town_Queenstown', 'embark_town_Southampton'])

# Exploratory Data Analysis (EDA)
# Descriptive statistics
print("Descriptive Statistics:")
print(data.describe())

# Pairplot
sns.pairplot(data, hue='survived', palette='bright')
plt.suptitle('Pairplot of Titanic Dataset', y=1.02)
plt.show()

# Heatmap of correlations
plt.figure(figsize=(14, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Boxplot
plt.figure(figsize=(14, 10))
sns.boxplot(data=data, orient='h', palette='Set2')
plt.title('Boxplot of Features')
plt.show()

# PCA Implementation
# Separate the features and the target variable
X = data.drop(columns=['survived'])
y = data['survived']

# Standardize the data
X = StandardScaler().fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
```

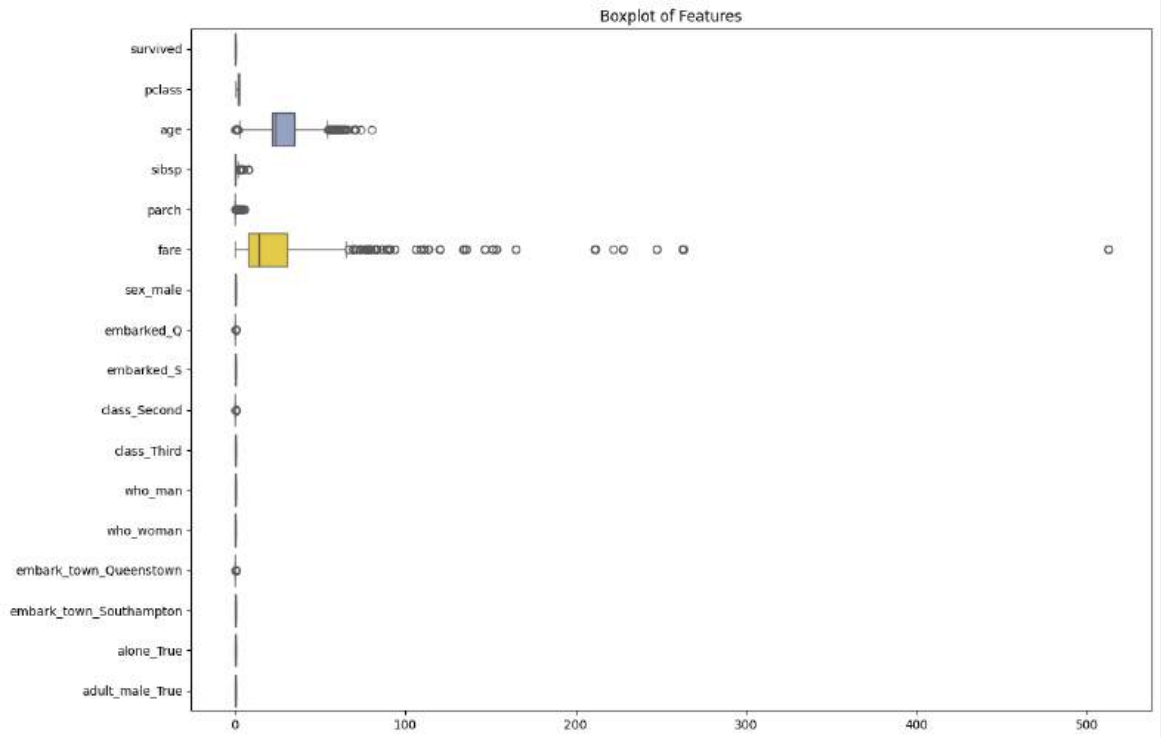
```
principal_components = pca.fit_transform(X)
principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Concatenate with the target
final_df = pd.concat([principal_df, y.reset_index(drop=True)], axis=1)

# Plot the PCA results
plt.figure(figsize=(10, 8))
sns.scatterplot(data=final_df,      x='PC1',      y='PC2',      hue='survived',
palette='coolwarm', s=50)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Titanic Dataset')
plt.legend(title='Survived')
plt.grid()
plt.show()
```

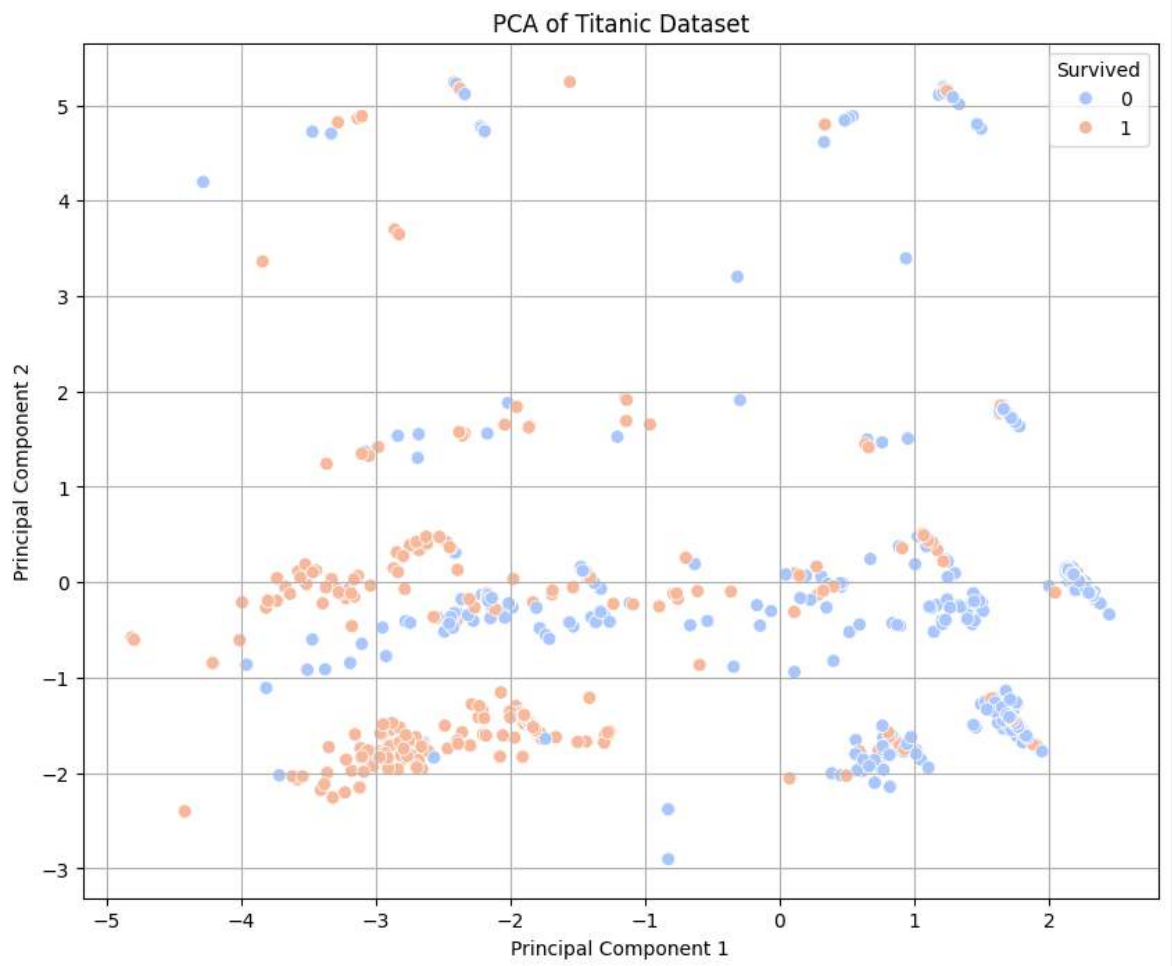
Output:

```
<ipython-input-14-5823df9afe45>:18: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For   example,   when   doing   'df[col].method(value,   inplace=True)',   try   using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead,
to perform the operation inplace on the original object.


  data['age'].fillna(data['age'].mode()[0], inplace=True)
<ipython-input-14-5823df9afe45>:19: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For   example,   when   doing   'df[col].method(value,   inplace=True)',   try   using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead,
to perform the operation inplace on the original object.


  data['embark_town'].fillna(data['embark_town'].mode()[0], inplace=True)
Descriptive Statistics:
         survived      pclass         age       sibsp       parch        fare
count  889.000000  889.000000  889.000000  889.000000  889.000000  889.000000
mean     0.382452    2.311586   28.518751    0.524184    0.382452   32.096681
std      0.486260    0.834700   13.162820    1.103705    0.806761   49.697504
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   22.000000    0.000000    0.000000    7.895800
50%      0.000000    3.000000   24.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   35.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```




Correlation Heatmap
survived
pclass
age
sibsp
parch
fare
sex_male
embarked_Q
embarked_S
class_Second
class_Third
who_man
who_woman
embark_town_Queenstown
embark_town_Southampton
alone_True
adult_male_True
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


Boxplot of Features
survived
pclass
age
sibsp
parch
fare
sex_male
embarked_Q
embarked_S
class_Second
class_Third
who_man
who_woman
embark_town_Queenstown
embark_town_Southampton
alone_True
adult_male_True
0
100
200
300
400
500


PCA of Titanic Dataset
Survived
0
1
Principal Component 2
Principal Component 1
5
4
3
2
1
0
−1
−2
−3
−5
−4
−3
−2
−1
0
1
2

4.7. Peran Feature Engineering dalam Machine Learning

Dalam rangkuman ini, kita akan mempelajari tentang peran penting feature engineering dalam machine learning. Feature engineering adalah proses mengubah data mentah menjadi fitur yang lebih informatif dan relevan, yang dapat meningkatkan performa model secara signifikan. Melalui teknik ini, data disusun dan dipilih sedemikian rupa sehingga memudahkan algoritma machine learning dalam menemukan pola. Dengan pemahaman yang baik tentang feature engineering, Anda dapat mengoptimalkan hasil model dan meningkatkan kualitas prediksi.

4.8. Definisi Corpus

Video ini akan membahas konsep *corpus* dan peran pentingnya dalam machine learning, terutama pada Natural Language Processing (NLP). *Corpus* adalah kumpulan besar teks atau data linguistik yang dirancang untuk melatih model bahasa. Data ini dapat berupa artikel, buku, tweet, atau jenis teks lainnya yang mencerminkan penggunaan bahasa dalam kehidupan sehari-hari. Dengan adanya *corpus*, model machine learning dapat mempelajari pola, makna, dan konteks dalam bahasa secara mendalam, sehingga mampu memahami dan menghasilkan teks yang lebih alami. *Corpus* memberikan landasan bagi model untuk menjalankan berbagai tugas NLP, seperti penerjemahan, analisis sentimen, dan pengenalan entitas.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

4.9. Strategi Memilih Dataset

Video ini akan membahas berbagai strategi dalam memilih dataset yang sesuai untuk proyek machine learning. Pemilihan dataset yang tepat merupakan langkah penting untuk memastikan model memiliki performa optimal. Kita akan mengulas kriteria penting seperti relevansi dataset terhadap tujuan analisis, ukuran dan cakupan dataset, serta kualitas dan keberagaman fitur yang terkandung di dalamnya. Selain itu, video ini akan membahas pentingnya memahami konteks data untuk memilih dataset yang paling sesuai dengan tujuan proyek. Anda juga akan diperkenalkan pada berbagai sumber dataset yang dapat diakses, serta kiat untuk mengurangi potensi bias dalam pemilihan data. Dengan memahami dan menerapkan strategi ini, Anda akan dapat memperbesar peluang keberhasilan proyek machine learning secara keseluruhan.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

4.10. Exploratory Data Analysis

Video ini akan membahas Exploratory Data Analysis (EDA), yang merupakan salah satu tahap awal yang sangat penting dalam proses data science dan machine learning. EDA bertujuan untuk memperoleh pemahaman yang mendalam tentang struktur dan pola yang ada dalam dataset, sehingga kita dapat mengambil keputusan yang lebih tepat sebelum masuk ke tahap pemodelan. Dalam video ini, kita akan mempelajari berbagai teknik utama dalam EDA, termasuk analisis statistik deskriptif, visualisasi data dengan grafik seperti histogram, scatter plot, dan box plot, serta cara mendeteksi outlier dan memahami hubungan antara variabel. EDA juga membantu kita mengenali potensi masalah dalam data, seperti data yang hilang atau distribusi yang tidak normal, yang dapat mempengaruhi kualitas model jika tidak ditangani dengan benar.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

4.11. Principal Component Analysis

Selamat datang di video pembelajaran tentang Principal Component Analysis (PCA). PCA adalah teknik statistik yang dirancang untuk mengurangi dimensi dataset sambil tetap mempertahankan informasi yang paling penting. Dalam video ini, kita akan mengupas konsep dasar PCA, memahami cara kerjanya, serta melihat bagaimana penerapannya dalam analisis data. PCA sangat bermanfaat dalam mempermudah visualisasi data dan meningkatkan efisiensi model machine learning, terutama ketika berhadapan dengan dataset berukuran besar dan kompleks. Mari kita lihat lebih lanjut bagaimana PCA dapat membantu mengungkap pola tersembunyi dalam data!

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

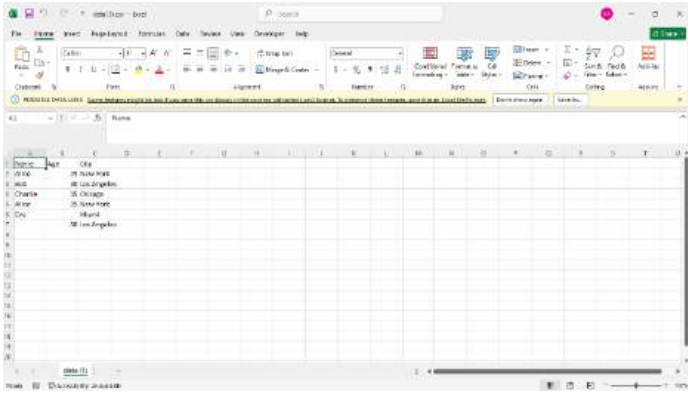
Bab 5. Validasi Kualitas dan Hasil Analisis Dataset

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan validasi model.
2. K-fold cross-validation.
3. Bootstrap sampling.
4. Evaluasi ketepatan model.
5. Peningkatan keandalan model.

5.1. Data Cleaning

Scan disi atau klik gambar di samping untuk mengakses data pembelajaran



Data cleaning adalah proses penting dalam analisis data yang bertujuan untuk memastikan data yang digunakan bersih, akurat, dan siap untuk analisis lebih lanjut. Proses ini mencakup beberapa langkah dan teknik yang dirancang untuk menangani berbagai masalah data, seperti missing values, duplikat, outliers, dan kesalahan format. Berikut adalah langkah-langkah dan teknik umum dalam data cleaning:

A. Mengatasi Missing Values

Missing values atau nilai yang hilang dalam dataset merupakan masalah yang sering dihadapi dalam proses machine learning (ML). Menangani missing values sangat penting karena beberapa alasan yang krusial bagi kualitas dan akurasi model. Pertama, data yang hilang dapat mengurangi kualitas keseluruhan dataset. Model ML bergantung pada data untuk mempelajari pola dan membuat prediksi. Jika data yang digunakan tidak lengkap, model mungkin tidak dapat menangkap pola yang sebenarnya, yang pada gilirannya dapat menghasilkan prediksi yang tidak akurat dan

tidak dapat diandalkan. Dalam banyak kasus, data yang hilang bisa menyebabkan model tidak bekerja dengan baik atau bahkan gagal berfungsi. Banyak algoritma ML, seperti regresi linier atau support vector machine (SVM), tidak dapat menangani missing values secara langsung. Algoritma ini memerlukan data yang lengkap untuk melakukan perhitungan dan analisis. Jika ada nilai yang hilang dalam dataset, algoritma ini bisa memberikan hasil yang salah atau bahkan tidak dapat beroperasi sama sekali.

Nilai yang hilang juga dapat menyebabkan bias dalam model jika tidak ditangani dengan benar. Misalnya, jika missing values tidak terdistribusi secara acak, ini bisa mengakibatkan model belajar dari subset data yang tidak representatif dari populasi sebenarnya. Hal ini dapat menyebabkan kesimpulan yang bias dan menyesatkan, mengurangi keandalan model dalam dunia nyata. Selain itu, menghapus baris atau kolom dengan missing values bisa mengurangi ukuran dataset secara signifikan, terutama jika jumlah nilai yang hilang banyak. Penghapusan ini dapat mengurangi kekayaan informasi dalam dataset, menghambat kemampuan model untuk belajar secara efektif dari data yang tersedia.

Penanganan missing values dapat dilakukan melalui beberapa metode, seperti penghapusan baris atau kolom yang tidak lengkap jika jumlahnya sedikit, atau menggunakan teknik imputasi untuk mengisi nilai yang hilang dengan rata-rata, median, atau metode prediktif lainnya. Ada juga beberapa algoritma seperti Random Forest dan XGBoost yang dapat menangani missing values secara internal. Dengan menangani missing values dengan baik, kita dapat memastikan bahwa data yang digunakan adalah representatif dan berkualitas, yang pada akhirnya akan menghasilkan model ML yang lebih akurat dan andal.

```
# Contoh Data Cleaning dengan Python
import pandas as pd
import numpy as np

# Membuat dataframe contoh
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'Alice', 'Eve', np.nan],
    'Age': [25, 30, 35, 25, np.nan, 50],
     'City': ['New York', 'Los Angeles', 'Chicago', 'New York', 'Miami', 'Los
Angeles']
}
df = pd.DataFrame(data)

# Menampilkan data awal
print("Data Awal:")
print(df)

# Menghapus duplikat
df = df.drop_duplicates()

# Menangani missing values dengan mengisi nilai median untuk kolom numerik
df['Age'] = df['Age'].fillna(df['Age'].median())

# Menghapus baris yang mengandung missing values di kolom 'Name'
df = df.dropna(subset=['Name'])
```

```
# Menampilkan data setelah cleaning
print("\nData Setelah Cleaning:")
df
```

Output:

```
Data Awal:
      Name   Age         City
0    Alice  25.0     New York
1      Bob  30.0  Los Angeles
2  Charlie  35.0      Chicago
3    Alice  25.0     New York
4      Eve   NaN        Miami
5      NaN  50.0  Los Angeles

Data Setelah Cleaning:
     Name      Age    City

 0   Alice     25.0   New York

 1   Bob       30.0   Los Angeles

 2   Charlie   35.0   Chicago

 4   Eve       32.5   Miami
```

```
# Identifikasi Missing Values
import pandas as pd

df = pd.read_csv('data.csv')
print(df.isnull().sum())
```

Output:

```
Name    1
Age     1
City    0
dtype: int64
```

```
# Mengisi Missing Values
df['Age'].fillna(df['Age'].mean())  # Mean
df['Age'].fillna(df['Age'].median(), inplace=True)  # Median
df['Age'].fillna(df['Age'].mode()[0], inplace=True)  # Mode
```

Output:

```
<ipython-input-3-6294aba4b8ca>:3: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].median(), inplace=True)  # Median
<ipython-input-3-6294aba4b8ca>:4: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].mode()[0], inplace=True)  # Mode
```

```
# Forward/Backward Fill
df['Age'].fillna(method='ffill', inplace=True)  # Forward fill
df['Age'].fillna(method='bfill', inplace=True)  # Backward fill
```

Output:

```
<ipython-input-4-3e3b9bda12ba>:2: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(method='ffill', inplace=True)  # Forward fill
<ipython-input-4-3e3b9bda12ba>:2: FutureWarning: Series.fillna with 'method' is
deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
  df['Age'].fillna(method='ffill', inplace=True)  # Forward fill
<ipython-input-4-3e3b9bda12ba>:3: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(method='bfill', inplace=True)  # Backward fill
<ipython-input-4-3e3b9bda12ba>:3: FutureWarning: Series.fillna with 'method' is
deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
```

```
  df['Age'].fillna(method='bfill', inplace=True)  # Backward fill
```

```
# Dropping Missing Values
df.dropna(subset=['Age'], inplace=True)
```

B. Menghapus Duplikat

Menghapus duplikat dalam dataset sangat penting karena duplikat dapat mendistorsi hasil analisis dan model machine learning. Duplikat menyebabkan overrepresentasi beberapa entri, yang dapat mengarah pada bias dalam perhitungan statistik dan model prediktif. Misalnya, jika satu pelanggan muncul beberapa kali dalam data penjualan, total penjualan atau rata-rata pembelian per pelanggan bisa menjadi tidak akurat. Selain itu, data duplikat dapat meningkatkan kompleksitas dan ukuran dataset, yang mengakibatkan penggunaan memori yang lebih tinggi dan waktu pemrosesan yang lebih lama. Dengan menghapus duplikat, kita memastikan bahwa setiap entri unik hanya muncul sekali, yang membantu menjaga integritas data, meningkatkan efisiensi analisis, dan menghasilkan model yang lebih akurat dan dapat diandalkan.

```
# Identifikasi duplikat
print(df.duplicated().sum())

# Menghapus duplikat
df.drop_duplicates(inplace=True)
```

Output:

```
0
```

C. Menangani Outliers

Outliers adalah data yang menyimpang secara signifikan dari sebagian besar data dalam dataset. Mereka dapat berada jauh dari nilai rata-rata atau median dan mungkin menunjukkan adanya variasi atau kesalahan dalam data. Outliers dapat disebabkan oleh berbagai faktor, seperti kesalahan pengukuran, kesalahan pencatatan, atau kejadian yang jarang terjadi tetapi sah. Mengapa Outliers Penting?

1. Pengaruh pada Statistik Deskriptif: Outliers dapat secara signifikan mempengaruhi statistik seperti mean (rata-rata) dan standard deviation (simpangan baku). Mereka dapat menyebabkan hasil statistik yang menyesatkan.
2. Pengaruh pada Model Machine Learning: Dalam machine learning, outliers dapat mempengaruhi kinerja model. Misalnya, model regresi linier dapat condong atau menjadi tidak akurat jika terdapat outliers.

3. Indikasi Masalah atau Informasi Berharga: Outliers kadang-kadang dapat menunjukkan kesalahan dalam data, tetapi mereka juga bisa memberikan wawasan berharga tentang fenomena langka atau penting.

Penanganan Outliers

1. Menghapus Outliers: Jika outliers disebabkan oleh kesalahan pengukuran atau data yang tidak relevan, mereka dapat dihapus dari dataset.
2. Transformasi Data: Menggunakan teknik seperti log transformation untuk mengurangi dampak outliers.
3. Model yang Tahan Outliers: Menggunakan model machine learning yang lebih tahan terhadap outliers, seperti tree-based methods.

```
# Langkah 1: Membuat DataFrame
import pandas as pd
import numpy as np

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'Alice', 'Eve', np.nan],
    'Age': [25, 30, 35, 25, np.nan, 50],
     'City': ['New York', 'Los Angeles', 'Chicago', 'New York', 'Miami', 'Los
Angeles']
}
df = pd.DataFrame(data)

# Langkah 2: Identifikasi Outliers
# Outliers dapat diidentifikasi menggunakan berbagai metode, salah satu yang umum
adalah menggunakan Z-score atau IQR (Interquartile Range).
# Di sini, kita akan menggunakan IQR untuk mendeteksi outliers.
# Menghitung IQR
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1

# Menentukan batas bawah dan atas untuk outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Menandai outliers
outliers = df[(df['Age'] < lower_bound) | (df['Age'] > upper_bound)]
print("Outliers:\n", outliers)

# Langkah 3: Menangani Outliers
# # Menghapus outliers
df_no_outliers = df[(df['Age'] >= lower_bound) & (df['Age'] <= upper_bound)]

# Mengganti outliers dengan nilai median
median_age = df['Age'].median()
df['Age'] = np.where((df['Age'] < lower_bound) | (df['Age'] > upper_bound),
median_age, df['Age'])

# Mengisi missing values dengan median
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Name'].fillna('Unknown', inplace=True)
```

Output:

```
Outliers:
 Empty DataFrame
Columns: [Name, Age, City]
Index: []
<ipython-input-10-c64bb35be9c7>:38: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].median(), inplace=True)
<ipython-input-10-c64bb35be9c7>:39: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Name'].fillna('Unknown', inplace=True)
```

D. Mengubah Data Kategorikal ke Numerik

Mengubah data kategorikal ke numerik dalam analisis data dan pemodelan machine learning sangat penting karena sebagian besar algoritma machine learning membutuhkan input numerik untuk melakukan perhitungan. Ada beberapa alasan utama mengapa transformasi ini diperlukan:

1. Kompatibilitas dengan Algoritma: Banyak algoritma machine learning, seperti regresi linier, regresi logistik, dan algoritma berbasis jarak seperti KNN, tidak dapat beroperasi dengan data kategorikal. Mereka membutuhkan data numerik untuk menghitung jarak, koefisien, dan parameter lainnya.
2. Kinerja dan Akurasi Model: Mengubah data kategorikal menjadi format numerik memungkinkan model untuk lebih efektif mengenali pola dalam data. Representasi numerik dapat membantu dalam memanfaatkan informasi yang terkandung dalam variabel kategorikal, yang dapat meningkatkan kinerja dan akurasi model.
3. Skalabilitas dan Efisiensi: Algoritma yang bekerja dengan data numerik sering kali lebih efisien dan skalabel dibandingkan dengan yang harus bekerja dengan data kategorikal. Data numerik memungkinkan operasi matematika dan statistik yang lebih cepat dan lebih mudah diterapkan.

4. Persyaratan Prapemrosesan: Beberapa langkah prapemrosesan data, seperti normalisasi atau standardisasi, memerlukan data dalam bentuk numerik. Ini penting untuk memastikan bahwa semua fitur dalam dataset berada pada skala yang sama, yang bisa mempengaruhi hasil dari algoritma ML.

```
# Contoh: misalnya, kita memiliki dataset dengan kolom "City" yang berisi data
kategorikal:

import pandas as pd

data = {'City': ['New York', 'Los Angeles', 'Chicago', 'New York', 'Miami']}
df = pd.DataFrame(data)

# One-Hot Encoding
df_one_hot = pd.get_dummies(df, columns=['City'])

print(df_one_hot)
```

Output:

```
   City_Chicago  City_Los Angeles  City_Miami  City_New York
0         False             False       False           True
1         False              True       False          False
2          True             False       False          False
3         False             False       False           True
4         False             False        True          False
```

E. Menangani Data Tidak Valid

Menangani data tidak valid adalah proses penting untuk memastikan integritas dan kualitas dataset. Data tidak valid dapat mengganggu analisis dan menghasilkan model yang tidak akurat. Berikut adalah langkah-langkah yang bisa diambil untuk menangani data tidak valid:

Identifikasi Data Tidak Valid

1. Pemeriksaan Kesalahan: Identifikasi kesalahan ketik atau entri yang tidak sesuai format yang diharapkan.
2. Pemeriksaan Batas Nilai: Pastikan nilai numerik berada dalam rentang yang wajar.
3. Pemeriksaan Konsistensi: Pastikan konsistensi antar kolom, misalnya tanggal mulai tidak boleh setelah tanggal berakhir.
4. Deteksi Anomali: Gunakan metode statistik atau algoritma untuk mendeteksi anomali dalam data.

Penanganan Data Tidak Valid

```
import pandas as pd
```

```
import numpy as np

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'Alice', 'Eve', np.nan],
    'Age': [25, 30, 35, 25, np.nan, 50],
     'City': ['New York', 'Los Angeles', 'Chicago', 'New York', 'Miami', 'Los
Angeles']
}
df = pd.DataFrame(data)

# Menghapus Data Tidak Valid: Jika jumlah data tidak valid kecil dan tidak
signifikan, menghapusnya bisa menjadi solusi yang cepat dan mudah.
df = df.dropna(subset=['Age', 'Name'])  # Menghapus baris dengan nilai 'Age' atau
'Name' yang tidak valid
```

```
# Mengganti dengan Nilai Lain:

# Mengisi dengan Rata-rata/Median: Mengisi nilai yang hilang atau tidak valid
dengan rata-rata atau median.
df['Age'] = df['Age'].fillna(df['Age'].median())

# Mengisi dengan Nilai Default: Mengisi dengan nilai default yang logis atau
sesuai konteks.
df['Name'] = df['Name'].fillna('Unknown')
```

```
# Transformasi Data: Mengubah data tidak valid menjadi format yang sesuai.
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')  # Mengubah nilai 'Age'
yang tidak valid menjadi NaN
```

```
# Pembersihan dengan Logika Bisnis: Menggunakan aturan bisnis untuk memperbaiki
data. Misalnya, jika usia tidak masuk akal (misalnya lebih dari 120 tahun), maka
ubah atau hapus.
df = df[(df['Age'] >= 0) & (df['Age'] <= 120)]
```

Contoh Menangani Data Tidak Valid

```
import pandas as pd
import numpy as np

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'Alice', 'Eve', np.nan],
    'Age': [25, 30, 35, 25, np.nan, 150],  # 150 dianggap sebagai nilai tidak
valid
     'City': ['New York', 'Los Angeles', 'Chicago', 'New York', 'Miami', 'Los
Angeles']
}
df = pd.DataFrame(data)

# Identifikasi nilai tidak valid
```

```
print("Data Awal:")
print(df)

# Mengubah nilai 'Age' yang tidak valid menjadi NaN
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Mengisi nilai 'Age' yang hilang atau tidak valid dengan median
df['Age'] = df['Age'].fillna(df['Age'].median())

# Mengisi nilai 'Name' yang hilang dengan 'Unknown'
df['Name'] = df['Name'].fillna('Unknown')

print("\nData Setelah Menangani Nilai Tidak Valid:")
print(df)
```

Output:

```
Data Awal:
      Name    Age         City
0    Alice   25.0     New York
1      Bob   30.0  Los Angeles
2  Charlie   35.0      Chicago
3    Alice   25.0     New York
4      Eve    NaN        Miami
5      NaN  150.0  Los Angeles

Data Setelah Menangani Nilai Tidak Valid:
      Name    Age         City
0    Alice   25.0     New York
1      Bob   30.0  Los Angeles
2  Charlie   35.0      Chicago
3    Alice   25.0     New York
4      Eve   30.0        Miami
5  Unknown  150.0  Los Angeles
```

Dalam contoh ini, nilai usia yang tidak valid (150) diubah menjadi nilai median, dan nama yang hilang diisi dengan 'Unknown'. Langkah-langkah ini membantu memastikan bahwa data yang digunakan untuk analisis dan pemodelan lebih bersih dan dapat diandalkan.

5.2. Data Normalization

Data normalization adalah proses mengubah skala fitur dalam dataset sehingga mereka berada dalam rentang yang sama atau memiliki distribusi yang serupa. Normalisasi data penting untuk meningkatkan kinerja algoritma machine learning, terutama yang sensitif terhadap skala data seperti k-NN, regresi linier, dan neural networks. Mengapa Data Normalization Penting?

1. Konsistensi Skala: Algoritma berbasis jarak seperti k-NN dan K-Means clustering sangat bergantung pada jarak antar titik data. Normalisasi memastikan bahwa semua fitur berkontribusi secara proporsional terhadap jarak tersebut.

2. Stabilitas Numerik: Model seperti regresi linier dan neural networks dapat mengalami masalah stabilitas numerik jika fitur memiliki skala yang sangat berbeda. Normalisasi membantu menghindari masalah ini dengan menjaga nilai fitur dalam rentang yang seragam.
3. Kecepatan Konvergensi: Normalisasi dapat meningkatkan kecepatan konvergensi algoritma pembelajaran seperti gradient descent dengan menghindari jalan yang curam di sepanjang satu dimensi dan datar di dimensi lainnya.

Metode Normalisasi Data

1. Min-Max Scaling: Mengubah data sehingga berada dalam rentang tertentu, biasanya [0, 1].
2. Z-Score Normalization (Standardization): Mengubah data sehingga memiliki rata-rata 0 dan standar deviasi 1.

   di mana μμ adalah rata-rata dan σσ adalah standar deviasi.
3. Robust Scaler: Menggunakan median dan IQR (Interquartile Range) untuk mengurangi pengaruh outliers.

Contoh Normalisasi dengan Python

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Contoh dataset
data = {
    'Age': [25, 30, 35, 40, 45],
    'Salary': [50000, 60000, 70000, 80000, 90000]
}
df = pd.DataFrame(data)

# Min-Max Scaling
min_max_scaler = MinMaxScaler()
df_min_max_scaled = pd.DataFrame(min_max_scaler.fit_transform(df),
columns=df.columns)

# Z-Score Normalization
standard_scaler = StandardScaler()
df_standard_scaled = pd.DataFrame(standard_scaler.fit_transform(df),
columns=df.columns)

print("Data Asli:")
print(df)

print("\nMin-Max Scaled Data:")
print(df_min_max_scaled)

print("\nStandardized Data:")
print(df_standard_scaled)
```

Output:

```
Data Asli:
   Age  Salary
0   25   50000
1   30   60000
2   35   70000
3   40   80000
4   45   90000

Min-Max Scaled Data:
    Age  Salary
0  0.00    0.00
1  0.25    0.25
2  0.50    0.50
3  0.75    0.75
4  1.00    1.00

Standardized Data:
        Age    Salary
0 -1.414214 -1.414214
1 -0.707107 -0.707107
2  0.000000  0.000000
3  0.707107  0.707107
4  1.414214  1.414214
```

Dalam contoh ini, kita melihat bagaimana fitur Age dan Salary diubah menggunakan Min-Max Scaling dan Z-Score Normalization. Setelah normalisasi, data berada dalam rentang yang lebih seragam, yang membantu meningkatkan kinerja model machine learning.

5.3. Validasi Model

Validasi model adalah proses evaluasi model machine learning untuk memastikan bahwa model tersebut bekerja dengan baik dan dapat diandalkan ketika diaplikasikan pada data baru. Tujuan utama validasi model adalah untuk mengukur kinerja model, mencegah overfitting, dan memastikan generalisasi model. Berikut adalah penjelasan lebih lanjut tentang tujuan dan teknik validasi model. Tujuan Validasi Model

1. Mengukur Kinerja Model: Validasi model membantu mengukur seberapa baik model bekerja dalam hal akurasi, presisi, recall, F1-score, dan metrik lainnya.
2. Mencegah Overfitting: Overfitting terjadi ketika model terlalu menyesuaikan diri dengan data pelatihan dan gagal menggeneralisasi pada data baru. Validasi model membantu mengidentifikasi dan mengurangi overfitting.
3. Memastikan Generalisasi: Validasi memastikan bahwa model dapat bekerja dengan baik pada data yang tidak terlihat sebelumnya, yang sangat penting untuk aplikasi dunia nyata.
4. Pemilihan Model: Membantu dalam pemilihan model terbaik dari beberapa model atau konfigurasi berdasarkan kinerja pada set validasi.

5. Hyperparameter Tuning: Membantu dalam menyetel hyperparameter model untuk mendapatkan kinerja terbaik.

Teknik Validasi Model

1. Holdout Validation: Dataset dibagi menjadi dua subset: satu untuk pelatihan (training set) dan satu lagi untuk pengujian (test set). Contoh: 70% data untuk pelatihan dan 30% data untuk pengujian. Kelemahan: Hasil bisa sangat bergantung pada cara data dibagi.
2. K-Fold Cross-Validation: Dataset dibagi menjadi k subset (folds). Model dilatih k kali, setiap kali menggunakan k-1 folds untuk pelatihan dan 1 fold untuk pengujian. Contoh: 5-Fold Cross-Validation. Keuntungan: Lebih akurat dan stabil karena menggunakan seluruh dataset untuk pelatihan dan pengujian.
3. Stratified K-Fold Cross-Validation: Mirip dengan K-Fold, tetapi memastikan bahwa setiap fold memiliki proporsi yang sama dari kelas target, menjaga distribusi kelas yang konsisten di seluruh fold. Keuntungan: Berguna untuk dataset yang tidak seimbang.
4. Leave-One-Out Cross-Validation (LOOCV): Setiap contoh data digunakan sekali sebagai set pengujian sementara sisanya digunakan sebagai set pelatihan. Keuntungan: Menggunakan semua data yang tersedia untuk pelatihan. Kelemahan: Sangat memakan waktu dan komputasi untuk dataset besar.
5. Time Series Cross-Validation: Digunakan untuk data urutan waktu. Data dibagi berdasarkan urutan waktu, memastikan bahwa model hanya dilatih pada data dari masa lalu dan diuji pada data dari masa depan. Keuntungan: Menghormati urutan temporal dan cocok untuk data deret waktu.
6.

Contoh Implementasi K-Fold Cross-Validation di Python

Berikut adalah contoh bagaimana mengimplementasikan K-Fold Cross-Validation menggunakan scikit-learn di Python:

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LinearRegression
import numpy as np

# Contoh dataset
X = np.array([[1, 2], [2, 3], [3, 4], [4, 5], [5, 6]])
y = np.array([2, 3, 4, 5, 6])

# Model
model = LinearRegression()

# K-Fold Cross-Validation
kf = KFold(n_splits=5)
```

```
scores = cross_val_score(model, X, y, cv=kf)

print("K-Fold Cross-Validation Scores:", scores)
print("Mean Score:", np.mean(scores))
```

Output:

```
K-Fold Cross-Validation Scores: [nan nan nan nan nan]
Mean Score: nan
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:1211:
UndefinedMetricWarning: R^2 score is not well-defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:1211:
UndefinedMetricWarning: R^2 score is not well-defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:1211:
UndefinedMetricWarning: R^2 score is not well-defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:1211:
UndefinedMetricWarning: R^2 score is not well-defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:1211:
UndefinedMetricWarning: R^2 score is not well-defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
```

Outputnya adalah skor kinerja model untuk setiap fold dan rata-rata skornya, yang memberikan gambaran umum tentang kinerja model Validasi model adalah langkah kritis dalam machine learning yang membantu memastikan bahwa model yang dikembangkan adalah akurat, andal, dan mampu menangani data yang tidak terlihat dengan baik. Dengan menggunakan teknik validasi yang tepat, kita dapat memilih model terbaik dan mengoptimalkan kinerjanya untuk aplikasi dunia nyata.

5.4. K-Fold Cross-Validation

K-Fold Cross-Validation adalah salah satu teknik validasi model yang umum digunakan dalam machine learning untuk mengevaluasi kinerja model secara lebih akurat. Teknik ini membagi dataset menjadi k subset (folds) yang sama besar, di mana setiap fold digunakan sebagai set pengujian satu kali sementara k-1 folds lainnya digunakan sebagai set pelatihan. Berikut ini adalah langkah-langkah dan manfaat utama dari K-Fold Cross-Validation:

Langkah-langkah K-Fold Cross-Validation

1. Pembagian Dataset: Dataset dibagi menjadi k subset atau folds yang sama besar.
2. Iterasi Training dan Testing: Model dilatih k kali menggunakan kombinasi berbeda dari k-1 folds untuk pelatihan dan 1 fold untuk pengujian setiap kali.
3. Evaluasi Kinerja: Untuk setiap iterasi, kinerja model dievaluasi menggunakan metrik evaluasi yang relevan (misalnya, akurasi, presisi, recall).

4. Perhitungan Rata-rata: Hasil evaluasi dari setiap iterasi diambil rata-ratanya untuk memberikan estimasi akhir dari kinerja model.

Manfaat K-Fold Cross-Validation

1. Menggunakan Data Secara Efisien: Memanfaatkan seluruh dataset untuk pelatihan dan pengujian, menghasilkan estimasi kinerja model yang lebih stabil dan akurat.
2. Pencegahan Overfitting: Dengan menggunakan k-1 folds untuk pelatihan pada setiap iterasi, K-Fold Cross-Validation membantu mengurangi risiko overfitting model terhadap data pelatihan tertentu.
3. Penggunaan yang Luas: Teknik ini dapat diterapkan pada berbagai jenis model dan dataset, termasuk data yang tidak seimbang atau data deret waktu.

```
# Implementasi K-Fold Cross-Validation di Python
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Model
model = LogisticRegression(max_iter=1000)

# K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)  # Misalnya, menggunakan 5
folds dengan shuffle dan seed random 42
scores = cross_val_score(model, X, y, cv=kf, scoring='accuracy')

print("K-Fold Cross-Validation Scores:", scores)
print("Mean Score:", scores.mean())
```

Output:

```
K-Fold Cross-Validation Scores: [1.          1.          0.93333333 0.96666667
0.96666667]
Mean Score: 0.9733333333333334
```

Dalam contoh di atas, dataset Iris dibagi menjadi 5 folds dengan shuffle dan seed random yang ditentukan. Model Regresi Logistik dievaluasi menggunakan metrik akurasi, dan hasil dari setiap fold dievaluasi dan dihitung rata-ratanya. K-Fold Cross-Validation adalah alat yang penting dalam evaluasi model machine learning untuk memastikan bahwa model yang dikembangkan dapat diandalkan dan dapat digeneralisasikan dengan baik pada data yang tidak terlihat sebelumnya.

5.5. Confussion Matrix

Confusion Matrix (matriks kebingungan) adalah tabel yang digunakan untuk mengevaluasi kinerja suatu model klasifikasi pada sebuah dataset yang sudah diketahui labelnya. Matriks ini menggambarkan jumlah hasil prediksi yang benar dan yang salah dalam empat kategori berbeda. Berikut ini adalah komponen confusion matrix:

1. True Positive (TP): Kasus di mana model memprediksi positif (1) dan kenyataannya positif (1).
2. True Negative (TN): Kasus di mana model memprediksi negatif (0) dan kenyataannya negatif (0).
3. False Positive (FP) (Type I Error): Kasus di mana model memprediksi positif (1) tetapi kenyataannya negatif (0). Juga dikenal sebagai kesalahan tipe I.
4. False Negative (FN) (Type II Error): Kasus di mana model memprediksi negatif (0) tetapi kenyataannya positif (1). Juga dikenal sebagai kesalahan tipe II.

Interpretasi Confusion Matrix

Matriks kebingungan memberikan gambaran yang lebih komprehensif tentang kinerja model klasifikasi daripada metrik evaluasi tunggal seperti akurasi. Dengan menggunakan komponen-komponen tersebut, beberapa metrik evaluasi lain dapat dihitung, seperti:

Contoh Confusion Matrix

Misalkan kita memiliki sebuah confusion matrix sebagai berikut:

```lua
   Predicted
  | 0   | 1  |
```

Actual |------|-----| 0 | 50 | 10 | 1 | 5 | 35 |

Dari matriks ini:

```
True Positive (TP) = 50
True Negative (TN) = 35
False Positive (FP) = 10
False Negative (FN) = 5
```

Dengan informasi ini, kita bisa menghitung akurasi, presisi, recall, dan F1-score untuk mengevaluasi performa model klasifikasi tersebut. Confusion matrix adalah

alat yang sangat berguna untuk memahami seberapa baik model klasifikasi kita bekerja pada dataset tertentu, terutama dalam konteks klasifikasi yang tidak seimbang atau ketika perlu menilai dampak dari kesalahan prediksi tertentu

5.6. Bootstrap Sampling

Bootstrap sampling adalah teknik resampling yang digunakan dalam statistik untuk mengevaluasi keandalan estimasi dari sebuah sampel data. Teknik ini memungkinkan kita untuk membuat estimasi dari sebaran (distribution) suatu statistik, seperti mean, median, atau deviasi standar, bahkan jika distribusi dari populasi tidak diketahui.

Konsep Dasar Bootstrap Sampling

1. Resampling: Bootstrap sampling melibatkan pengambilan sampel dari dataset yang ada dengan mengembalikan (dengan penggantian) observasi yang sudah diambil. Ini berarti setiap observasi dalam dataset asli memiliki kesempatan untuk muncul di dalam sampel bootstrap lebih dari satu kali atau bahkan tidak muncul sama sekali.
2. Estimasi Sebaran: Dengan membuat banyak sampel bootstrap (biasanya ribuan sampai jutaan), kita dapat membangun distribusi dari statistik yang ingin diestimasi. Misalnya, kita dapat menghitung mean dari setiap sampel bootstrap, dan distribusi dari mean tersebut akan memberi kita perkiraan tentang sebaran mean dari populasi.
3. Keuntungan: Bootstrap sampling mengatasi masalah ketidakpastian tentang distribusi dari populasi, karena kita tidak harus bergantung pada asumsi tertentu tentang distribusi. Teknik ini juga memberikan interval kepercayaan (confidence intervals) yang lebih akurat.

Langkah-langkah Implementasi Bootstrap Sampling

1. Ambil Sampel: Ambil sampel dari dataset asli sebanyak n kali, dengan penggantian.
2. Hitung Statistik: Hitung statistik dari setiap sampel bootstrap, seperti mean, median, deviasi standar, atau persentil.
3. Bangun Distribusi: Dengan menggunakan hasil statistik dari banyak sampel bootstrap, bangun distribusi dari statistik tersebut.
4. Interval Kepercayaan: Dengan distribusi yang telah dibangun, hitung interval kepercayaan untuk estimasi statistik tertentu, misalnya 95% confidence interval.

Contoh Bootstrap Sampling

Misalkan kita memiliki dataset berikut yang berisi nilai-nilai tinggi dari suatu populasi:

Data: [10, 15, 8, 12, 14, 20, 18, 16, 11, 13]

Bootstrap sampling adalah teknik resampling yang digunakan dalam statistik untuk mengevaluasi keandalan estimasi dari sebuah sampel data. Teknik ini memungkinkan kita untuk membuat estimasi dari sebaran (distribution) suatu statistik, seperti mean, median, atau deviasi standar, bahkan jika distribusi dari populasi tidak diketahui.

Konsep Dasar Bootstrap Sampling

1. Resampling: Bootstrap sampling melibatkan pengambilan sampel dari dataset yang ada dengan mengembalikan (dengan penggantian) observasi yang sudah diambil. Ini berarti setiap observasi dalam dataset asli memiliki kesempatan untuk muncul di dalam sampel bootstrap lebih dari satu kali atau bahkan tidak muncul sama sekali.
2. Estimasi Sebaran: Dengan membuat banyak sampel bootstrap (biasanya ribuan sampai jutaan), kita dapat membangun distribusi dari statistik yang ingin diestimasi. Misalnya, kita dapat menghitung mean dari setiap sampel bootstrap, dan distribusi dari mean tersebut akan memberi kita perkiraan tentang sebaran mean dari populasi.
3. Keuntungan: Bootstrap sampling mengatasi masalah ketidakpastian tentang distribusi dari populasi, karena kita tidak harus bergantung pada asumsi tertentu tentang distribusi. Teknik ini juga memberikan interval kepercayaan (confidence intervals) yang lebih akurat.

Langkah-langkah Implementasi Bootstrap Sampling

1. Ambil Sampel: Ambil sampel dari dataset asli sebanyak n kali, dengan penggantian.
2. Hitung Statistik: Hitung statistik dari setiap sampel bootstrap, seperti mean, median, deviasi standar, atau persentil.
3. Bangun Distribusi: Dengan menggunakan hasil statistik dari banyak sampel bootstrap, bangun distribusi dari statistik tersebut.
4. Interval Kepercayaan: Dengan distribusi yang telah dibangun, hitung interval kepercayaan untuk estimasi statistik tertentu, misalnya 95% confidence interval.

Contoh Bootstrap Sampling

Misalkan kita memiliki dataset berikut yang berisi nilai-nilai tinggi dari suatu populasi:

Data: [10, 15, 8, 12, 14, 20, 18, 16, 11, 13]

Kita ingin mengevaluasi rata-rata (mean) dari populasi ini menggunakan bootstrap sampling:

1. Ambil Sampel: Ambil n sampel dengan penggantian dari dataset asli. Misalnya, kita ambil 1000 sampel bootstrap.
2. Hitung Mean: Hitung mean dari setiap sampel bootstrap.
3. Bangun Distribusi: Dengan menggunakan mean dari sampel bootstrap, kita dapat membangun distribusi mean dari populasi.
4. Interval Kepercayaan: Hitung 95% confidence interval dari distribusi mean yang dibangun.

Berikut adalah contoh implementasi sederhana menggunakan Python untuk menghitung mean dan confidence interval dengan bootstrap sampling:

```
import numpy as np

# Data
data = np.array([10, 15, 8, 12, 14, 20, 18, 16, 11, 13])

# Bootstrap sampling
n_samples = 1000
bootstrap_means = np.zeros(n_samples)

for i in range(n_samples):
    bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
    bootstrap_means[i] = np.mean(bootstrap_sample)

# Confidence interval (95%)
ci_lower = np.percentile(bootstrap_means, 2.5)
ci_upper = np.percentile(bootstrap_means, 97.5)

print("Mean:", np.mean(data))
print("95% Confidence Interval:", ci_lower, "-", ci_upper)
```

Output:

```
Mean: 13.7
95% Confidence Interval: 11.6975 - 15.9
```

Dalam contoh ini, kita mengambil 1000 sampel bootstrap dari data, menghitung mean dari setiap sampel, dan kemudian menghitung 95% confidence interval dari distribusi mean yang dibangun. Bootstrap sampling adalah alat yang powerful untuk memperoleh estimasi yang stabil dan distribusi statistik dari data yang ada.

5.7. Evaluasi Kinerja Model

Evaluasi kinerja model adalah proses penting dalam machine learning untuk menilai seberapa baik model dapat melakukan prediksi terhadap data yang belum pernah dilihat sebelumnya. Tujuannya adalah untuk memastikan bahwa model dapat digeneralisasikan dengan baik dan bekerja dengan akurat pada situasi dunia nyata. Berikut ini adalah langkah-langkah umum dalam evaluasi kinerja model:

Langkah-langkah Evaluasi Kinerja Model

1. Pembagian Dataset: Bagi dataset menjadi set pelatihan (training set) dan set pengujian (test set) secara acak. Data pelatihan digunakan untuk melatih model, sementara data pengujian digunakan untuk mengevaluasi kinerja model.
2. Pemilihan Metrik Evaluasi: Pilih metrik evaluasi yang sesuai tergantung pada jenis masalah yang sedang diselesaikan. Beberapa metrik umum termasuk:
    - Classification: Akurasi, presisi, recall, F1-score, ROC-AUC.
    - Regression: Mean Squared Error (MSE), R-squared, Mean Absolute Error (MAE).
3. Pembuatan Model: Pilih algoritma model yang sesuai dengan jenis masalah (klasifikasi, regresi, clustering, dll.) dan latih model menggunakan data pelatihan.
4. Evaluasi pada Data Pengujian: Evaluasi model pada data pengujian menggunakan metrik yang telah dipilih. Ini memberikan gambaran tentang seberapa baik model akan berperforma pada data baru yang belum pernah dilihat sebelumnya.
5. Validasi Silang (Cross-Validation): Gunakan teknik validasi silang seperti K-Fold Cross-Validation untuk mendapatkan estimasi yang lebih baik tentang kinerja model. Ini membantu dalam mengurangi variabilitas hasil evaluasi yang mungkin disebabkan oleh pembagian acak data.
6. Analisis Confusion Matrix (untuk klasifikasi): Jika model adalah model klasifikasi, analisis confusion matrix membantu dalam memahami seberapa baik model dapat mengklasifikasikan instance dari setiap kelas.
7. Penyetelan Hyperparameter: Lakukan penyetelan hyperparameter untuk meningkatkan kinerja model. Gunakan teknik seperti Grid Search atau Random Search untuk menemukan kombinasi hyperparameter terbaik.

Contoh Implementasi Evaluasi Kinerja Model di Python

Berikut adalah contoh sederhana menggunakan Python untuk evaluasi kinerja model klasifikasi menggunakan metrik akurasi dan confusion matrix:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Bagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inisialisasi dan latih model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Prediksi dengan data pengujian
y_pred = model.predict(X_test)

# Evaluasi kinerja model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
```

Output:

```
Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
# Contoh Confussion Matrix di NLP
# Impor library yang diperlukan
import nltk
from nltk.corpus import movie_reviews
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import confusion_matrix, classification_report

# Unduh dataset sentimen ulasan film dari NLTK
nltk.download('movie_reviews')

# Ambil ulasan dan label dari dataset
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

# Pisahkan teks ulasan dan label
texts = [' '.join(document) for document, category in documents]
```

```
labels = [category for document, category in documents]

# Ubah teks menjadi vektor fitur TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(texts)

# Bagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2,
random_state=42)

# Inisialisasi dan latih model klasifikasi (misalnya, Linear SVM)
classifier = LinearSVC()
classifier.fit(X_train, y_train)

# Prediksi kelas pada data pengujian
y_pred = classifier.predict(X_test)

# Evaluasi model menggunakan confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Evaluasi model menggunakan classification report
report = classification_report(y_test, y_pred)
print("\nClassification Report:")
print(report)
```

Output:

```
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Unzipping corpora/movie_reviews.zip.
Confusion Matrix:
[[170  29]
 [ 37 164]]

Classification Report:
              precision    recall  f1-score   support

         neg       0.82      0.85      0.84       199
         pos       0.85      0.82      0.83       201

    accuracy                           0.83       400
   macro avg       0.84      0.84      0.83       400
weighted avg       0.84      0.83      0.83       400
```

5.8. Teknik-teknik Normalisasi Data

Dalam pembahasan ini, kita akan merangkum berbagai teknik normalisasi data yang krusial dalam proses pengolahan data untuk machine learning. Normalisasi data adalah langkah penting yang bertujuan untuk mengubah skala fitur ke dalam rentang tertentu, sehingga memungkinkan model untuk belajar dengan lebih efektif dan efisien. Dengan adanya variasi dalam skala fitur, model machine learning sering kali menghadapi tantangan dalam menghasilkan prediksi yang akurat.

Teknik normalisasi yang tepat dapat membantu mengurangi bias yang dihasilkan oleh perbedaan skala antar fitur dan memastikan bahwa setiap fitur berkontribusi secara proporsional terhadap proses pelatihan model. Selain itu, normalisasi juga dapat meningkatkan konvergensi algoritma optimasi yang digunakan dalam pelatihan model.

Dalam rangkuman ini, kita akan membahas beberapa metode normalisasi yang umum digunakan, seperti Min-Max Scaling, Z-score Normalization, dan Robust Scaling. Setiap metode memiliki keunggulan dan kekurangan tersendiri, serta dapat diterapkan tergantung pada karakteristik data yang dimiliki. Dengan pemahaman yang mendalam tentang teknik-teknik normalisasi ini, diharapkan kita dapat memilih metode yang paling sesuai untuk meningkatkan kinerja model machine learning yang sedang dibangun. Mari kita eksplorasi lebih lanjut tentang normalisasi data dan teknik-tekniknya.

5.9. Pengantar Analisis Sentimen

Dalam video ini, kita akan memberikan pengantar tentang analisis sentimen, yang merupakan salah satu aplikasi penting dalam Natural Language Processing (NLP). Analisis sentimen bertujuan untuk menentukan apakah suatu teks mengungkapkan perasaan positif, negatif, atau netral. Kita akan membahas berbagai teknik yang digunakan dalam analisis sentimen, mulai dari metode sederhana seperti penghitungan kata hingga algoritma machine learning yang lebih kompleks. Selain itu, kita juga akan mengeksplorasi bagaimana analisis sentimen dapat diterapkan di berbagai bidang, termasuk pemasaran, layanan pelanggan, dan penelitian sosial. Dengan pemahaman dasar ini, Anda akan dipersiapkan untuk menjelajahi lebih

dalam mengenai cara menganalisis dan memahami emosi yang terkandung dalam data teks.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

5.10. Underfitting vs Overfitting

Dalam video ini, kita akan membahas dua konsep penting dalam machine learning, yaitu underfitting dan overfitting. Underfitting terjadi ketika model terlalu sederhana untuk menangkap pola dalam data, sehingga menghasilkan prediksi yang buruk baik pada data pelatihan maupun data uji. Hal ini sering disebabkan oleh pemilihan model yang tidak tepat atau kurangnya fitur yang relevan. Sebaliknya, overfitting terjadi ketika model terlalu kompleks, sehingga ia belajar pola dari data pelatihan dengan sangat baik, termasuk noise atau fluktuasi yang tidak relevan. Akibatnya, model ini berkinerja buruk pada data uji karena tidak mampu menggeneralisasi dengan baik. Dengan memahami kedua konsep ini, Anda dapat lebih baik dalam merancang model machine learning yang efektif dan efisien.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

5.11. Bootstrap Sampling

Bootstrap adalah teknik resampling yang digunakan untuk memperkirakan statistik dari sampel dengan cara mengambil beberapa subset dari data asli secara acak, dengan pengembalian. Dalam video ini, kita akan mempelajari konsep dasar bootstrap, cara kerjanya, serta manfaatnya dalam mengukur ketidakpastian model dan validasi data. Kita akan menjelajahi bagaimana Bootstrap Sampling dapat membantu meningkatkan akurasi prediksi pada berbagai model statistik dan machine learning. Dengan pemahaman ini, Anda akan melihat bagaimana teknik ini berperan penting dalam analisis data dan pengembangan model yang lebih andal.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 6. Pertemuan 6 - Organisasi dan Analisis Dataset

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan feature engineering.
2. Teknik data augmentation.
3. Metode normalisasi data.
4. Penggunaan ensemble methods.
5. Integrasi dataset dari berbagai sumber.

6.1. Feature Engineering: definisi, teknik, dan implementasi

Feature engineering adalah proses di mana kita menggunakan pengetahuan domain untuk membuat fitur-fitur baru yang lebih informatif dari data mentah (raw data) yang ada. Tujuannya adalah untuk meningkatkan kualitas data yang digunakan dalam model machine learning, sehingga model dapat membuat prediksi yang lebih akurat.

Teknik Feature Engineering:

1. Penggabungan Fitur (Feature Combination): Menggabungkan beberapa fitur untuk menciptakan fitur baru yang lebih informatif. Contohnya, dalam data geospasial, bisa digabungkan koordinat latitude dan longitude menjadi fitur jarak atau fitur lain yang lebih bermakna.
2. Pengurangan Dimensi (Dimensionality Reduction): Mengurangi jumlah fitur dalam dataset dengan teknik seperti Principal Component Analysis (PCA) atau t-SNE untuk menghilangkan fitur yang kurang relevan atau redundan.
3. Penambahan Informasi (Feature Augmentation): Menambahkan informasi tambahan ke dalam dataset, misalnya menambahkan fitur tanggal menjadi hari kerja atau hari libur, yang dapat memberikan wawasan tambahan pada model.
4. Transformasi Fitur (Feature Transformation): Mengubah skala atau distribusi dari fitur untuk memperbaiki kinerja model. Contoh umumnya adalah normalisasi atau transformasi logaritmik.
5. Seleksi Fitur (Feature Selection): Memilih subset fitur yang paling penting atau relevan untuk meningkatkan performa model. Teknik seperti chi-square, information gain, atau recursive feature elimination (RFE) digunakan untuk seleksi fitur.
6. Encoding Kategori (Category Encoding): Mengubah variabel kategori menjadi bentuk yang dapat dimengerti oleh model, seperti menggunakan teknik one-hot encoding atau label encoding.

```
# Implementasi:
# Berikut adalah contoh implementasi sederhana beberapa teknik feature engineering
menggunakan Python dan scikit-learn:
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Contoh dataset
data = {
    'City': ['New York', 'Los Angeles', 'Chicago', 'Boston', 'Miami'],
    'Temperature': [25, 30, 22, 18, 28],
    'Category': ['A', 'B', 'A', 'C', 'B']
}
df = pd.DataFrame(data)

# Contoh 1: Encoding kategori
encoder = OneHotEncoder()
encoded_category = encoder.fit_transform(df[['Category']])

# Contoh 2: Transformasi fitur numerik
scaler = StandardScaler()
scaled_temperature = scaler.fit_transform(df[['Temperature']])

# Contoh 3: Penggabungan fitur
df['City_Temperature'] = df['City'] + '_' + df['Temperature'].astype(str)

# Contoh 4: Pengurangan dimensi dengan PCA
pca = PCA(n_components=1)
pca_result = pca.fit_transform(encoded_category.toarray())

# Contoh 5: Ekstraksi fitur teks dengan TF-IDF
# NOTE: corpus has been modified to have the same number of samples (5) as
df['Category']
corpus = [
    'This is the first document.',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
    'This is the fifth document.' # Added a fifth document to match the number
of samples in df['Category']
]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)

# Memisahkan fitur dan label, dan membagi dataset
X_train, X_test, y_train, y_test = train_test_split(X, df['Category'],
test_size=0.2, random_state=42)

# Contoh menggunakan model machine learning setelah feature engineering
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Accuracy: 0.0
```

Dalam contoh di atas:

a. Kita mulai dengan dataset sederhana yang berisi fitur kategori, numerik, dan teks.

b. Dilakukan encoding kategori menggunakan OneHotEncoder, transformasi skala fitur numerik menggunakan StandardScaler, penggabungan fitur dengan menambahkan fitur baru City_Temperature, dan pengurangan dimensi dengan menggunakan PCA.

c. Fitur teks diekstraksi menggunakan TfidfVectorizer.

d. Setelah melakukan feature engineering, dataset dibagi menjadi data pelatihan dan data pengujian, dan digunakan model RandomForestClassifier untuk memprediksi kategori dari contoh data pengujian.

e. Akurasi dari model diukur menggunakan metrik accuracy.

Feature engineering adalah langkah penting dalam pengembangan model machine learning yang dapat meningkatkan performa model dengan memanfaatkan informasi tambahan atau lebih relevan dari dataset yang tersedia.

6.2. Data Augmentation

Data augmentation adalah teknik yang digunakan untuk meningkatkan variasi dataset dengan membuat salinan data yang dimodifikasi dari data asli. Tujuan utamanya adalah untuk memperluas dataset yang tersedia tanpa mengumpulkan lebih banyak data, yang dapat membantu mengurangi overfitting dan meningkatkan generalisasi model.

Tujuan Data Augmentation

1. Meningkatkan Keanekaragaman Data: Dengan membuat variasi data yang lebih besar, model dapat mempelajari pola yang lebih umum dan lebih baik dalam data yang belum pernah dilihat sebelumnya.

2. Mengurangi Overfitting: Dengan menggunakan augmentasi, kita dapat mengurangi risiko model mempelajari detail yang spesifik dari data pelatihan yang mungkin tidak relevan untuk data baru.

3. Memperbaiki Kinerja Model: Dengan dataset yang lebih besar dan lebih bervariasi, kita dapat meningkatkan performa model dalam tugas-tugas seperti klasifikasi gambar atau teks.

Data Augmentation Gambar

- Rotasi: Memutar gambar dalam berbagai sudut.
- Flip: Membalikkan gambar secara horizontal atau vertikal.
- Zoom: Memperbesar atau memperkecil bagian-bagian gambar.
- Pergeseran: Memindahkan gambar ke arah horizontal atau vertikal.
- Cropping: Memotong bagian dari gambar.

Implementasi menggunakan imgaug (Python):

imgaug adalah pustaka Python yang kuat untuk augmentasi gambar dengan berbagai teknik. Berikut adalah contoh penggunaan imgaug untuk melakukan beberapa teknik augmentasi gambar:

```
import numpy as np
import imgaug.augmenters as iaa
import matplotlib.pyplot as plt
from PIL import Image

# Contoh gambar
image = np.array(Image.open('example_image.jpg'))

# Definisi augmentor
seq = iaa.Sequential([
    iaa.Fliplr(0.5),  # Flip horizontal dengan peluang 50%
    iaa.Affine(rotate=(-10, 10)),  # Rotasi gambar dalam rentang -10 sampai 10
derajat
    iaa.GaussianBlur(sigma=(0, 1.0))  # Blur Gaussian dengan sigma antara 0 dan
1.0
])

# Augmentasi gambar
augmented_image = seq(image=image)

# Tampilkan hasil augmentasi
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Gambar Asli')

plt.subplot(1, 2, 2)
plt.imshow(augmented_image)
plt.title('Gambar Setelah Augmentasi')

plt.tight_layout()
plt.show()
```

Data Augmentation Teks

a. Pertambahan Kata (Word Insertion): Menambahkan kata-kata baru ke dalam teks.

b. Penggantian Kata (Word Replacement): Mengganti kata-kata dengan sinonim atau kata-kata yang serupa.

c. Pemotongan Kata (Word Truncation): Memotong atau menghapus kata-kata dari teks.

d. Pengacakan Urutan Kata (Word Reordering): Mengacak urutan kata dalam teks.

Implementasi menggunakan TextBlob (Python):

TextBlob adalah pustaka Python yang memudahkan analisis teks dengan menyediakan akses mudah ke operasi linguistik. Berikut adalah contoh penggunaan TextBlob untuk beberapa teknik augmentasi teks:

```
from textblob import TextBlob
import random
import nltk
nltk.download('punkt')

# Contoh teks
text = "Ini adalah contoh kalimat untuk augmentasi teks."

# Objek TextBlob untuk teks
blob = TextBlob(text)

# Pertambahan kata
augmented_text = blob.words + ['baru', 'kata']

# Penggantian kata acak
for i in range(len(blob.words)):
    if random.random() < 0.3:  # Probabilitas 30% untuk penggantian kata
        augmented_text[i] = 'kata_baru'

# Konversi kembali ke teks
augmented_text = ' '.join(augmented_text)

print("Teks Asli:", text)
print("Teks Setelah Augmentasi:", augmented_text)
```

Output:

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Teks Asli: Ini adalah contoh kalimat untuk augmentasi teks.
Teks Setelah Augmentasi: Ini kata_baru contoh kalimat untuk augmentasi teks baru
kata
```

6.3. Normalisasi Data

Normalisasi data adalah proses mengubah nilai-nilai dalam dataset sehingga memiliki skala yang seragam. Tujuannya adalah untuk menghindari dominasi oleh fitur-fitur dengan skala besar dan memastikan setiap fitur memiliki kontribusi yang seimbang terhadap hasil akhir model.

Implementasi menggunakan Scikit-Learn

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Contoh dataset
data = np.array([[1.0, 2.0, 3.0],
                 [4.0, 5.0, 6.0],
                 [7.0, 8.0, 9.0]])

# 1. Min-Max Scaling (Normalization)
scaler_minmax = MinMaxScaler()
data_minmax_scaled = scaler_minmax.fit_transform(data)

print("Min-Max Scaled Data:")
print(data_minmax_scaled)
print()

# 2. Z-Score Normalization (Standardization)
scaler_standard = StandardScaler()
data_standard_scaled = scaler_standard.fit_transform(data)

print("Z-Score Standardized Data:")
print(data_standard_scaled)
```

Output:

```
Min-Max Scaled Data:
[[0.  0.  0. ]
 [0.5 0.5 0.5]
 [1.  1.  1. ]]

Z-Score Standardized Data:
[[-1.22474487 -1.22474487 -1.22474487]
 [ 0.          0.          0.        ]
 [ 1.22474487  1.22474487  1.22474487]]
```

Dalam contoh di atas, data_minmax_scaled dan data_standard_scaled adalah hasil dari normalisasi data menggunakan Min-Max Scaling dan Z-Score Normalization, masing-masing. Hasilnya dapat digunakan sebagai input untuk model machine learning untuk memastikan setiap fitur memiliki pengaruh yang setara terhadap hasil akhir model. Dengan menggunakan teknik normalisasi, kita dapat mempersiapkan data dengan cara yang optimal untuk berbagai jenis model machine learning, meningkatkan interpretabilitas, konvergensi, dan performa model secara keseluruhan.

6.4. Ensemble Methods

Ensemble Methods adalah teknik dalam machine learning di mana beberapa model (yang disebut "learners" atau "base models") digabungkan bersama untuk meningkatkan kinerja prediksi secara keseluruhan. Ide dasarnya adalah bahwa gabungan dari beberapa model yang berbeda sering kali lebih baik daripada model individu yang digunakan secara terpisah.

Jenis Ensemble Methods

1. Bagging (Bootstrap Aggregating): Menggunakan beberapa dataset bootstrap (sampel acak dengan penggantian dari dataset pelatihan) untuk melatih beberapa model serentak. Contoh: Random Forest.
2. Boosting: Mengurangi bias model dengan memusatkan perhatian pada data yang salah diperkirakan oleh model sebelumnya. Contoh: AdaBoost, Gradient Boosting Machines (GBM), XGBoost, LightGBM
3. Stacking (Stacked Generalization): Menggabungkan output dari beberapa model berbeda sebagai input untuk model meta-learner (model yang lebih tinggi). Contoh: Menggunakan hasil prediksi dari SVM, Random Forest, dan Neural Network sebagai input untuk model klasifikasi logistic regression.

Implementasi Ensemble Methods

Berikut adalah contoh implementasi ensemble methods menggunakan Python dengan scikit-learn:

```
# Contoh dengan Random Forest (Bagging):
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh dataset
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)

# Bagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inisialisasi dan latih model Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Prediksi menggunakan model yang sudah dilatih
y_pred = clf.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Accuracy: 0.9
```

```
# Contoh dengan AdaBoost (Boosting)
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh dataset
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)

# Bagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inisialisasi dan latih model AdaBoost
clf = AdaBoostClassifier(n_estimators=50, random_state=42)
clf.fit(X_train, y_train)

# Prediksi menggunakan model yang sudah dilatih
y_pred = clf.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
/usr/local/lib/python3.10/dist-
packages/sklearn/ensemble/_weight_boosting.py:527: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
Accuracy: 0.87
```

Keunggulan Ensemble Methods:

1. Meningkatkan Kinerja: Ensemble methods sering kali menghasilkan performa yang lebih baik daripada model tunggal karena mampu mengurangi varians (overfitting) dan meningkatkan akurasi prediksi.
2. Stabilitas: Mengurangi risiko kesalahan yang disebabkan oleh variasi dalam dataset pelatihan atau noise.
3. Fleksibilitas: Dapat digunakan dengan berbagai jenis model dasar, memungkinkan untuk mengkombinasikan kekuatan dari berbagai pendekatan pemodelan.

6.5. Identifikasi Pola dan Tren

Identifikasi pola dan tren merupakan bagian penting dalam analisis data, terutama dalam konteks analisis time series atau dataset yang mengandung data kronologis. Pola dan tren mencerminkan perilaku atau perubahan dalam data dari waktu ke waktu, yang dapat memberikan wawasan berharga untuk pengambilan keputusan atau prediksi masa depan. Berikut adalah cara umum untuk mengidentifikasi pola dan tren dalam data.

Identifikasi Pola

1. Visualisasi Data: Gunakan grafik seperti line plot, scatter plot, atau histogram untuk memvisualisasikan data. Pola dapat terlihat sebagai pengelompokan data, pola siklus, atau pola lain yang menunjukkan hubungan antara variabel.
2. Analisis Deskriptif: Hitung statistik deskriptif seperti mean, median, dan mode untuk memahami distribusi data. Pola bisa terlihat sebagai pola konsentrasi nilai di sekitar nilai tengah atau pola distribusi yang asimetris.
3. Analisis Cluster: Gunakan teknik clustering seperti k-means untuk mengidentifikasi kelompok data yang serupa. Pola mungkin muncul sebagai kelompok data yang terpisah dengan karakteristik yang berbeda-beda.

Identifikasi Tren

1. Trendline: Buat trendline atau garis tren menggunakan metode seperti regresi linier. Tren dapat terlihat sebagai pola umum dari data yang menunjukkan arah pergerakan yang jelas ke atas (tren naik), ke bawah (tren turun), atau datar (tren stabil).
2. Moving Average: Gunakan moving average untuk meratakan fluktuasi jangka pendek dan menyoroti tren jangka panjang. Tren bisa terlihat sebagai kecenderungan nilai rata-rata yang berubah dari waktu ke waktu.
3. Decomposition: Lakukan dekomposisi time series untuk memisahkan data menjadi komponen tren, musiman, dan residual. Tren dapat terlihat sebagai komponen yang menunjukkan perubahan sistematis dari waktu ke waktu.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

# Load data from CSV
data = pd.read_csv('contoh-data.csv', parse_dates=['date'], index_col='date')

# Visualize data with line plot
plt.figure(figsize=(10, 6))
plt.plot(data.index, data['value'], marker='o', linestyle='-')
plt.title('Line Plot of Time Series Data')
plt.xlabel('Date')
```

```
plt.ylabel('Value')
plt.grid(True)
plt.show()

# Decompose time series data
result = seasonal_decompose(data['value'], model='additive', period=1)

# Visualize trend component
plt.figure(figsize=(10, 6))
plt.plot(result.trend)
plt.title('Trend Component')
plt.xlabel('Date')
plt.ylabel('Trend Value')
plt.grid(True)
plt.show()
```

Dengan menjalankan script ini, Anda akan melihat visualisasi pola dan tren dari data yang ada dalam data.csv. Pastikan untuk menginstal library yang diperlukan seperti Pandas, Matplotlib, dan statsmodels jika belum terinstal di lingkungan Python Anda (pip install pandas matplotlib statsmodels).

Contoh Kasus NLP: Analisis Sentimen Sederhana

Kita akan membuat sebuah contoh kasus di mana kita akan melakukan analisis sentimen terhadap ulasan film. Kita akan menggunakan dataset yang sudah disediakan oleh NLTK dan menghitung nilai positif atau negatif dari setiap ulasan.

```
!pip install nltk
```

Output:

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-
packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.6)
```

```
import nltk
from nltk.corpus import movie_reviews
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download dataset movie_reviews jika belum ada
```

```
nltk.download('movie_reviews')
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')

# Ambil dataset ulasan film dari NLTK
reviews = []
for fileid in movie_reviews.fileids():
    review = movie_reviews.raw(fileid)
    reviews.append(review)

# Ambil contoh ulasan
sample_review = reviews[0]

# Tokenisasi kata-kata dalam ulasan
tokens = word_tokenize(sample_review)

# Hilangkan stop words (kata-kata umum yang tidak memiliki makna penting)
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

# Analisis sentimen menggunakan VADER (Valence Aware Dictionary and sEntiment
Reasoner)
sid = SentimentIntensityAnalyzer()
sentiment_score = sid.polarity_scores(sample_review)

# Tampilkan hasil
print("Contoh Ulasan Film:")
print(sample_review)
print("\nTokenisasi Kata-kata:")
print(tokens[:20])  # Tampilkan 20 token pertama
print("\nTokenisasi Kata-kata setelah filtering Stop Words:")
print(filtered_tokens[:20])  # Tampilkan 20 token pertama setelah filtering stop
words
print("\nAnalisis Sentimen:")
print(sentiment_score)
```

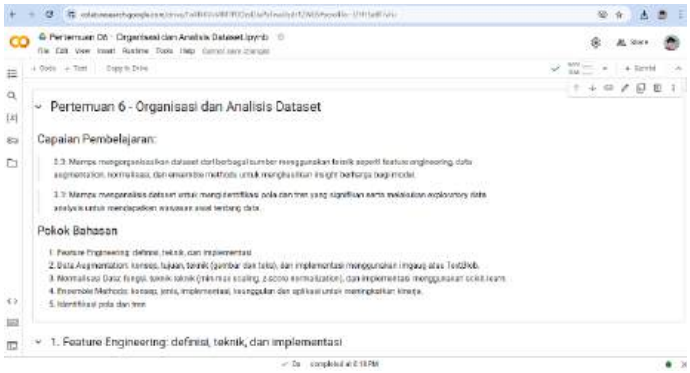
Output:

```
Contoh Ulasan Film:
plot : two teen couples go to a church party , drink and then drive .
they get into an accident .
one of the guys dies , but his girlfriend continues to see him in her life , and
has nightmares .
what's the deal ?
watch the movie and " sorta " find out . . .
critique : a mind-fuck movie for the teen generation that touches on a very cool
idea , but presents it in a very bad package .
which is what makes this review an even harder one to write , since i generally
applaud films which attempt to break the mold , mess with your head and such (
lost highway & memento ) , but there are good and bad ways of making all types
of films , and these folks just didn't snag this one correctly .
they seem to have taken this pretty neat concept , but executed it terribly .
so what are the problems with the movie ?
well , its main problem is that it's simply too jumbled .
it starts off " normal " but then downshifts into this " fantasy " world in which
you , as an audience member , have no idea what's going on .
there are dreams , there are characters coming back from the dead , there are
others who look like the dead , there are strange apparitions , there are
```

```
disappearances , there are a looooot of chase scenes , there are tons of weird
things that happen , and most of it is simply not explained .
now i personally don't mind trying to unravel a film every now and then , but
when all it does is give me the same clue over and over again , i get kind of fed
up after a while , which is this film's biggest problem .
it's obviously got this big secret to hide , but it seems to want to hide it
completely until its final five minutes .
and do they make things entertaining , thrilling or even engaging , in the
meantime ?
not really .
the sad part is that the arrow and i both dig on flicks like this , so we actually
figured most of it out by the half-way point , so all of the strangeness after
that did start to make a little bit of sense , but it still didn't the make the
film all that more entertaining .
i guess the bottom line with movies like this is that you should always make sure
that the audience is " into it " even before they are given the secret password
to enter your world of understanding .
i mean , showing melissa sagemiller running away from visions for about 20 minutes
throughout the movie is just plain lazy ! !
okay , we get it . . . there
are people chasing her and we don't know who they are .
do we really need to see it over and over again ?
how about giving us different scenes offering further insight into all of the
strangeness going down in the movie ?
apparently , the studio took this film away from its director and chopped it up
themselves , and it shows .
there might've been a pretty decent teen mind-fuck movie in here somewhere , but
i guess " the suits " decided that turning it into a music video with little edge
, would make more sense .
the actors are pretty good for the most part , although wes bentley just seemed
to be playing the exact same character that he did in american beauty , only in
a new neighborhood .
but my biggest kudos go out to sagemiller , who holds her own throughout the
entire film , and actually has you feeling her character's unraveling .
overall , the film doesn't stick because it doesn't entertain , it's confusing ,
it rarely excites and it feels pretty redundant for most of its runtime , despite
a pretty cool ending and explanation to all of the craziness that came before it
.
oh , and by the way , this is not a horror or teen slasher flick . . . it's
just packaged to look that way because someone is apparently assuming that the
genre is still hot with the kids .
it also wrapped production two years ago and has been sitting on the shelves ever
since .
whatever . . . skip
it !
where's joblo coming from ?
a nightmare of elm street 3 ( 7/10 ) - blair witch 2 ( 7/10 ) - the crow ( 9/10
) - the crow : salvation ( 4/10 ) - lost highway ( 10/10 ) - memento ( 10/10 ) -
the others ( 9/10 ) - stir of echoes ( 8/10 )


Tokenisasi Kata-kata:
['plot', ':', 'two', 'teen', 'couples', 'go', 'to', 'a', 'church', 'party', ',',
'drink', 'and', 'then', 'drive', '.', 'they', 'get', 'into', 'an']

Tokenisasi Kata-kata setelah filtering Stop Words:
['plot', ':', 'two', 'teen', 'couples', 'go', 'church', 'party', ',', 'drink',
'drive', '.', 'get', 'accident', '.', 'one', 'guys', 'dies', ',', 'girlfriend']

Analisis Sentimen:
{'neg': 0.093, 'neu': 0.762, 'pos': 0.145, 'compound': 0.9924}
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
```

```
[nltk_data]   Package movie_reviews is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



6.6. Perbedaan Overfitting dan Underfitting

Pada kesempatan ini, kita akan membahas konsep penting dalam machine learning, yaitu perbedaan antara overfitting dan underfitting. Kedua istilah ini merujuk pada masalah yang sering dihadapi saat membangun model prediktif. Overfitting terjadi ketika model terlalu kompleks dan terlalu sesuai dengan data pelatihan, sehingga tidak dapat menggeneralisasi dengan baik pada data baru. Sebaliknya, underfitting terjadi ketika model terlalu sederhana untuk menangkap pola yang ada dalam data, sehingga kinerjanya buruk baik pada data pelatihan maupun data baru. Memahami perbedaan antara kedua kondisi ini sangat penting untuk membangun model yang efektif dan akurat. Selanjutnya, kita akan melihat ilustrasi yang menggambarkan perbedaan antara overfitting dan underfitting.

6.7. Ringkasan Natural Language Processing

Pengantar berikut ini akan memberikan gambaran singkat tentang Natural Language Processing (NLP). NLP adalah bidang ilmu komputer dan kecerdasan buatan yang berfokus pada interaksi antara komputer dan bahasa manusia. Dengan menggunakan algoritma dan model matematis, NLP memungkinkan mesin untuk memahami, menganalisis, dan menghasilkan teks dalam bahasa alami. Teknologi ini telah banyak diterapkan dalam berbagai aplikasi, seperti penerjemahan bahasa, analisis sentimen, dan chatbot, yang semakin memudahkan komunikasi antara manusia dan mesin. Mari kita simak ringkasan lebih lanjut tentang NLP.

6.8. Natural Language Processing

Dalam video ini, kita akan menjelaskan konsep Natural Language Processing (NLP) dan menjadikannya salah satu bidang yang paling menarik dalam kecerdasan buatan. NLP merupakan cabang dari machine learning yang berfokus pada interaksi antara komputer dan bahasa manusia. Berbagai tugas yang dapat dilakukan dengan NLP, seperti analisis sentimen, terjemahan otomatis, pengenalan suara, dan pembuatan teks, akan dibahas secara mendetail. Kita juga akan mengulas beberapa teknik dasar dalam NLP, seperti tokenisasi, stemming, dan lemmatization. Selain itu, penggunaan model bahasa seperti Word2Vec dan Transformer akan menjadi sorotan utama dalam video ini. Dengan pemahaman ini, Anda akan lebih siap untuk mengeksplorasi aplikasi NLP dalam berbagai konteks.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

6.9. Peran Data dalam Kesuksesan Model Machine Learning

Dalam video ini, kita akan membahas pentingnya data dalam kesuksesan model machine learning. Data berfungsi sebagai bahan bakar utama yang memungkinkan model untuk belajar dan membuat prediksi yang akurat. Kita akan melihat bagaimana kualitas data—seperti kelengkapan, relevansi, dan kebersihan—sangat mempengaruhi performa model. Data yang berkualitas baik membantu model untuk mengenali pola dengan lebih tepat, sedangkan data yang buruk dapat menyebabkan kesalahan seperti overfitting atau underfitting. Dengan data yang tepat, model machine learning dapat memberikan hasil yang lebih andal dan akurat, menjadikannya elemen kunci dalam keberhasilan proyek machine learning.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

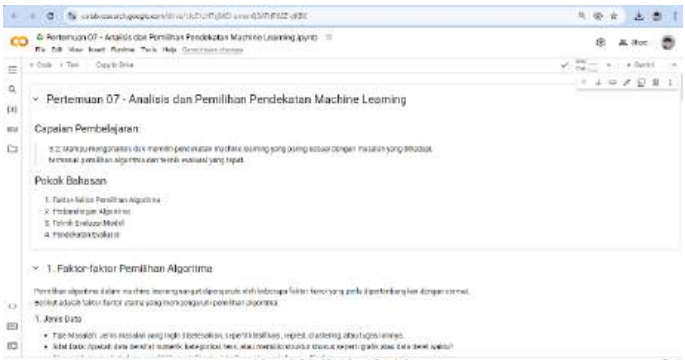
Bab 7. Analisis dan Pemilihan Pendekatan Machine Learning

Hal-hal yang dibahas pada Bab ini:

1. Analisis masalah dan kebutuhan.
2. Pemilihan algoritma machine learning.
3. Teknik evaluasi model.
4. Penyesuaian pendekatan dengan karakteristik data.
5. Studi kasus penerapan machine learning.

7.1. Faktor-faktor Pemilihan Algoritma

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



Pemilihan algoritma dalam machine learning sangat dipengaruhi oleh beberapa faktor kunci yang perlu dipertimbangkan dengan cermat. Berikut adalah faktor-faktor utama yang mempengaruhi pemilihan algoritma:

1. Jenis Data

   a. Tipe Masalah: Jenis masalah yang ingin diselesaikan, seperti klasifikasi, regresi, clustering, atau tugas lainnya.

   b. Sifat Data: Apakah data bersifat numerik, kategorikal, teks, atau memiliki struktur khusus seperti grafik atau data deret waktu?

   c. Dimensi Data: Apakah data memiliki banyak fitur (variabel) atau hanya beberapa fitur?

2. Tujuan Analisis

   a. Ketersediaan Label: Apakah data memiliki label yang jelas (supervised learning) atau tidak (unsupervised learning)?

b. Performa yang Diinginkan: Apakah yang lebih penting, kecepatan eksekusi atau akurasi prediksi? Apakah Anda lebih peduli dengan performa secara keseluruhan atau dengan performa pada kelas-kelas minoritas (misalnya, dalam klasifikasi)?

3. Kompleksitas Model

a. Interpretabilitas: Apakah penting untuk memahami dan menjelaskan bagaimana model membuat prediksi?

b. Kapasitas Model: Apakah Anda memerlukan model yang bisa menangani masalah yang sangat kompleks dan fleksibel atau cukup dengan model yang lebih sederhana dan mudah diinterpretasi?

c. Overfitting dan Underfitting: Bagaimana algoritma menangani risiko overfitting (terlalu cocok dengan data pelatihan) atau underfitting (tidak cocok dengan data)?

Contoh Penerapan

1. Contoh 1: Untuk klasifikasi teks dalam data dengan label, seperti klasifikasi spam email, algoritma seperti Naive Bayes atau Support Vector Machines (SVM) sering digunakan karena efektif dalam menangani data teks dan memiliki kemampuan yang baik dalam mengatasi dimensi yang tinggi.

2. Contoh 2: Untuk data deret waktu dalam prediksi harga saham, algoritma seperti ARIMA (Autoregressive Integrated Moving Average) atau LSTM (Long Short-Term Memory) dalam neural networks sering digunakan karena dapat menangani karakteristik khusus deret waktu seperti tren dan musiman.

3. Contoh 3: Untuk clustering data tanpa label dalam data customer segmentation, algoritma seperti K-Means atau DBSCAN sering digunakan karena mampu mengelompokkan data dengan baik berdasarkan pola yang ada dalam dataset.

Pemilihan algoritma yang tepat sangat penting untuk mencapai hasil yang optimal dalam machine learning. Hal ini melibatkan pemahaman yang mendalam tentang karakteristik data, tujuan analisis yang ingin dicapai, dan tingkat kompleksitas model yang diperlukan. Dengan mempertimbangkan faktor-faktor ini secara menyeluruh, Anda dapat memilih algoritma yang sesuai untuk memecahkan masalah spesifik yang Anda hadapi.

7.2. Perbandingan Algoritma

Perbandingan antara berbagai algoritma dalam machine learning sangat penting untuk memilih pendekatan yang tepat sesuai dengan masalah yang dihadapi. Berikut ini perbandingan beberapa algoritma yang umum digunakan berdasarkan karakteristik dan aplikasi.

a. Regresi Linier

Karakteristik: Algoritma sederhana yang cocok untuk regresi yang memodelkan hubungan linier antara variabel input dan output. Mencari garis regresi terbaik untuk meminimalkan kesalahan kuadrat.

Kelebihan: Mudah diinterpretasi dan cepat dilatih dan cocok untuk kasus dengan hubungan linier yang jelas antara variabel.

Kekurangan: Tidak mampu menangani hubungan non-linier di antara variabel dan rentan terhadap overfitting jika jumlah fitur (variabel) besar.

b. Decision Trees

Karakteristik: Algoritma non-parametrik yang membangun model dalam bentuk pohon keputusan dan memprediksi nilai target dengan membagi data berdasarkan aturan keputusan yang dipelajari dari data pelatihan.

Kelebihan: Mudah diinterpretasi dan dapat mengatasi hubungan non-linier dan interaksi antara variabel dan tidak memerlukan normalisasi data.

Kekurangan: Rentan terhadap overfitting, terutama dengan pohon yang sangat dalam dan tidak cocok untuk data dengan banyak fitur dan nilai yang sangat bervariasi.

c. Support Vector Machines (SVM)

Karakteristik: Algoritma yang digunakan untuk klasifikasi dan regresi dan memisahkan kelas dengan mencari hiperplane terbaik yang memiliki margin maksimum di antara kelas.

Kelebihan: Efektif dalam ruang fitur yang tinggi dan dapat menangani data non-linier dengan kernel yang tepat.

Kekurangan: Membutuhkan penyetelan parameter (seperti kernel) yang baik dan tidak cocok untuk dataset besar karena memerlukan waktu pelatihan yang lama.

d. K-Nearest Neighbors (KNN)

Karakteristik: Algoritma yang berdasarkan pada kumpulan data pelatihan untuk mengklasifikasikan data baru berdasarkan mayoritas dari k-nearest neighbors (tetangga terdekat) dalam ruang fitur.

Kelebihan: Mudah diimplementasikan dan tidak memerlukan pelatihan dan cocok untuk data yang tidak linier atau tidak terstruktur.

Kekurangan: Sensitif terhadap skala data dan jumlah dimensi dan performa lambat dalam memprediksi untuk dataset besar.

7.3. Teknik Evaluasi Model

Evaluasi model dalam machine learning sangat penting untuk memastikan bahwa model yang dikembangkan memiliki kinerja yang baik dan dapat diandalkan. Berikut adalah beberapa teknik dan metrik yang umum digunakan untuk evaluasi model:

A. K-Fold Cross-Validation

Teknik validasi di mana data dibagi menjadi k subset (folds). Model dilatih pada k-1 subset dan divalidasi pada subset yang tersisa. Proses ini diulang k kali dengan setiap fold sebagai subset validasi sekali. Kelebihannya adalah memberikan estimasi yang lebih stabil dan andal dari kinerja model dengan memanfaatkan seluruh dataset untuk pelatihan dan validasi. Kekurangannya adalah memerlukan waktu komputasi yang lebih lama dibandingkan dengan pembagian data sederhana.

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
import numpy as np

# Contoh K-Fold Cross-Validation
model = RandomForestClassifier()
scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
print(f"Accuracy: {np.mean(scores):.2f} (+/- {np.std(scores):.2f})")
```

B. Confusion Matrix

Tabel yang digunakan untuk mengevaluasi kinerja model klasifikasi. Menunjukkan jumlah true positives (TP), true negatives (TN), false positives (FP), dan false negatives (FN). Kelebihannya adalah memberikan informasi detail tentang jenis kesalahan yang dibuat oleh model. Namun kekurangannya adalah sulit diinterpretasikan pada dataset dengan banyak kelas.

```
from sklearn.metrics import confusion_matrix

# Prediksi
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

Interpretasi Hasil Evaluasi

a. Accuracy: Digunakan ketika dataset seimbang. Tingkat akurasi yang tinggi menunjukkan model yang baik, tetapi tidak cukup informatif untuk dataset yang tidak seimbang.

b. Precision dan Recall: Digunakan untuk memahami bagaimana model menangani kesalahan positif dan negatif. Berguna ketika ada biaya yang berbeda untuk kesalahan jenis yang berbeda.

c. F1 Score: Berguna ketika ada trade-off antara precision dan recall dan keduanya sama pentingnya.

d. ROC-AUC: Digunakan untuk mengevaluasi kinerja model klasifikasi secara keseluruhan pada berbagai threshold.

e. Confusion Matrix: Memberikan wawasan mendetail tentang kesalahan spesifik yang dibuat oleh model, membantu dalam memperbaiki dan memahami kelemahan model.

7.4. Pendekatan Evaluasi

Evaluasi model dalam machine learning tidak hanya mencakup penggunaan metrik dan teknik dasar seperti yang telah dijelaskan sebelumnya, tetapi juga melibatkan pendekatan evaluasi lanjutan yang bertujuan untuk mengoptimalkan model secara menyeluruh. Berikut adalah beberapa teknik evaluasi lanjutan, perbandingan teknik evaluasi, serta aplikasi praktis dan studi kasusnya. Berikut ini adalah beberapa Advanced Evaluation Techniques:

A. Grid Search

Teknik pencarian hiperparameter yang mencoba setiap kombinasi dari serangkaian parameter yang ditentukan pengguna untuk menemukan kombinasi terbaik. Grid search dimulai proses mengatur grid dari parameter, melatih model pada setiap kombinasi, dan memilih kombinasi dengan kinerja terbaik berdasarkan metrik evaluasi yang dipilih. Kelebihan Grid Search adalah memberikan jaminan menemukan kombinasi parameter yang optimal dalam ruang pencarian yang ditentukan. Namun, kekurangannya adalah memerlukan waktu komputasi yang sangat besar, terutama jika ruang pencarian parameter besar.

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Definisikan model dan parameter grid
model = RandomForestClassifier()
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30]
}
```

```
# Grid Search
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

Random Search

Teknik pencarian hiperparameter yang memilih kombinasi parameter secara acak dari ruang pencarian yang telah ditentukan pengguna. Random searh dimulai dengan mengatur distribusi dari parameter, melatih model pada sejumlah kombinasi acak, dan memilih kombinasi dengan kinerja terbaik berdasarkan metrik evaluasi yang dipilih. Kelebihannya adalah lebih efisien daripada grid search, terutama pada ruang pencarian parameter yang besar. Kekurangannya adalah tidak memberikan jaminan menemukan kombinasi parameter yang optimal tetapi bisa mendekati hasil optimal dengan waktu komputasi lebih sedikit.

```
from sklearn.model_selection import RandomizedSearchCV

# Definisikan model dan parameter distribusi
model = RandomForestClassifier()
param_dist = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'bootstrap': [True, False]
}

# Random Search
random_search = RandomizedSearchCV(estimator=model,
param_distributions=param_dist, n_iter=10, cv=5, scoring='accuracy')
random_search.fit(X_train, y_train)

print("Best Parameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)
```

7.5. Perbandingan Teknik-Teknik Evaluasi

Kelebihan dan Kekurangan

- Grid Search: Memastikan semua kombinasi parameter diuji, tetapi bisa sangat lambat jika ruang pencarian besar.
- Random Search: Lebih cepat karena tidak menguji semua kombinasi, tetapi tidak memberikan jaminan menemukan kombinasi optimal.

Penggunaan yang Direkomendasikan

- Grid Search: Digunakan ketika ruang pencarian parameter relatif kecil dan Anda memiliki sumber daya komputasi yang cukup.

- Random Search: Digunakan ketika ruang pencarian parameter besar atau tidak diketahui sebelumnya dan Anda ingin solusi yang lebih cepat.

Practical Applications and Case Studies

Aplikasi Praktis

Hyperparameter Tuning untuk Model Prediksi Cuaca:

- Menggunakan Grid Search untuk menemukan kombinasi optimal dari parameter dalam model regresi linear atau decision tree
- Menggunakan Random Search pada model deep learning untuk mengatur parameter seperti learning rate, batch size, dan jumlah neuron pada setiap lapisan.

Optimasi Model Deteksi Penipuan:

- Menggunakan Grid Search untuk mengatur parameter pada model random forest atau SVM untuk mengidentifikasi transaksi penipuan dengan akurasi tinggi.
- Menggunakan Random Search untuk model gradient boosting yang memerlukan tuning parameter yang lebih kompleks.

Studi Kasus

Klasifikasi Teks:

- Studi Kasus: Mengklasifikasikan email sebagai spam atau tidak spam.
- Pendekatan: Menggunakan Grid Search untuk mengatur parameter seperti alpha pada Naive Bayes dan C pada SVM untuk mendapatkan model terbaik.
- Hasil: Meningkatkan akurasi klasifikasi dengan optimasi parameter.

Prediksi Harga Rumah:

- Studi Kasus: Memprediksi harga rumah berdasarkan fitur seperti ukuran, lokasi, dan tahun bangunan.
- Pendekatan: Menggunakan Random Search untuk mengoptimalkan parameter pada model random forest dan gradient boosting.
- Hasil: Mengurangi kesalahan prediksi (MAE, RMSE) dengan menemukan kombinasi parameter yang optimal.

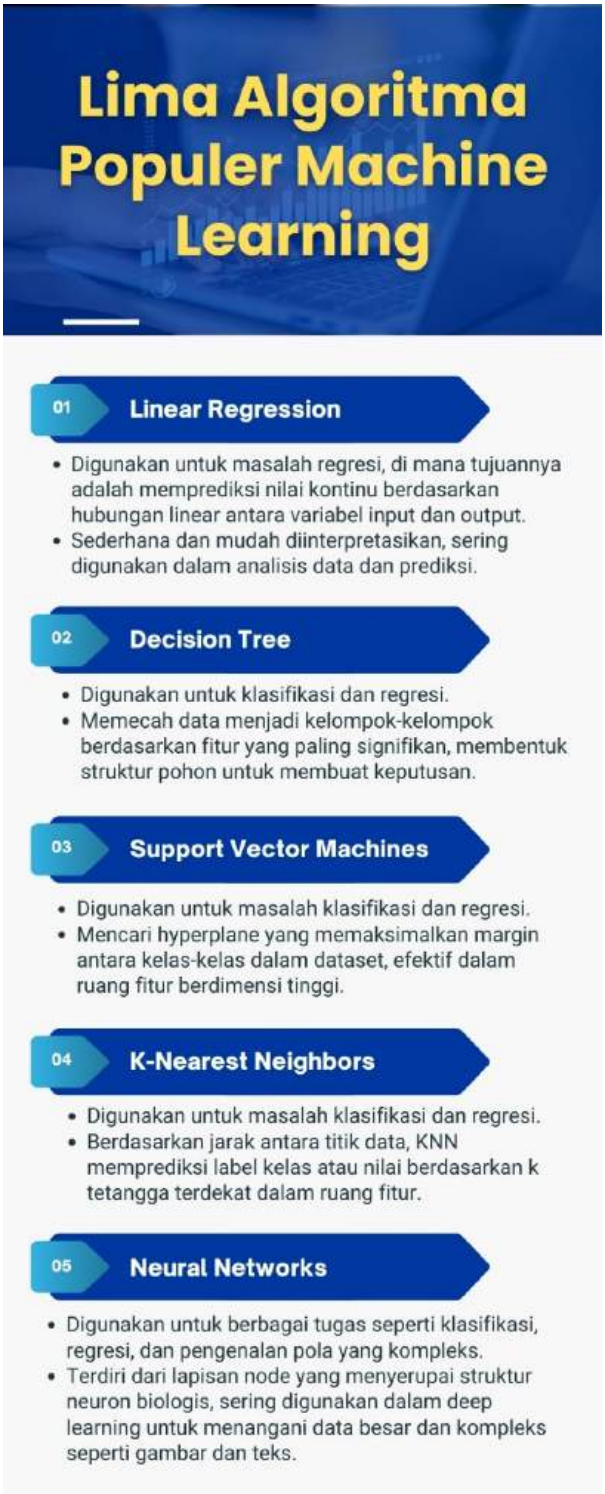
7.6. Linear Regression vs Logistic Regression

Pengantar berikut akan membahas perbedaan antara Linear Regression dan Logistic Regression. Kedua algoritma ini sama-sama digunakan dalam analisis data dan machine learning, namun memiliki fungsi dan pendekatan yang berbeda. Linear Regression digunakan untuk memprediksi nilai kontinu, sedangkan Logistic Regression digunakan untuk klasifikasi biner atau multikelas. Pemahaman mengenai perbedaan utama antara keduanya akan membantu dalam memilih model yang sesuai dengan tipe data dan tujuan analisis.

7.7. Lima Algoritma Populer di Machine Learning

Dalam bagian ini, kita akan membahas lima algoritma populer yang sering digunakan dalam machine learning. Algoritma-algoritma ini memiliki aplikasi yang

luas dalam berbagai bidang, termasuk klasifikasi, regresi, dan klastering. Setiap algoritma memiliki karakteristik unik dan dapat digunakan untuk menyelesaikan berbagai jenis masalah berdasarkan data yang tersedia. Memahami algoritma-algoritma ini akan membantu Anda memilih pendekatan yang tepat untuk proyek machine learning Anda. Selanjutnya, akan ditampilkan foto-foto yang relevan untuk memberikan gambaran lebih jelas mengenai setiap algoritma tersebut.



7.8. Linear vs Logistik Regression

Pada video ini, akan dijelaskan perbedaan utama antara Linear Regression dan Logistic Regression, dua algoritma yang sering digunakan dalam machine learning. Linear Regression berfokus pada prediksi nilai kontinu dengan membangun hubungan linear antara variabel independen dan dependen. Model ini menghasilkan garis lurus yang paling cocok untuk data dalam rentang numerik. Sementara itu, Logistic Regression lebih cocok untuk klasifikasi, di mana model ini memprediksi probabilitas kategori tertentu. Model Logistic Regression membentuk kurva sigmoid untuk memetakan hasil ke rentang antara 0 dan 1. Dengan memahami perbedaan mendasar ini, Anda akan dapat memilih metode yang sesuai berdasarkan jenis data dan tujuan analisis.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

7.9. Contoh Kasus Supervised dan Unsupervised

Pada video ini, akan dijelaskan contoh penerapan Supervised Learning dan Unsupervised Learning dalam machine learning. Dalam Supervised Learning, kita akan menggunakan contoh prediksi harga rumah, di mana model dilatih dengan dataset yang mencakup berbagai fitur, seperti ukuran rumah, lokasi, dan jumlah kamar tidur, serta label harga yang sudah diketahui. Dengan data ini, model dapat memprediksi harga rumah baru berdasarkan informasi yang diberikan. Di sisi lain, Unsupervised Learning akan diilustrasikan melalui contoh pengelompokan pelanggan. Dalam skenario ini, model bekerja pada data tanpa label, seperti pola transaksi pembelian, untuk mengidentifikasi kelompok dan pola tersembunyi dalam data. Pendekatan ini bermanfaat bagi perusahaan dalam segmentasi pasar dan perencanaan strategi pemasaran.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 8. Optimalisasi Kinerja Model Machine Learning

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan hyperparameter tuning.
2. Teknik-teknik feature engineering.
3. Implementasi ensemble methods.
4. Evaluasi kinerja model.
5. Studi kasus optimasi model.

8.1. Definisi, Dampak, dan Pentingnya Optimalisasi Model

Optimalisasi model dalam machine learning adalah proses menyesuaikan parameter dan hiperparameter model untuk meningkatkan kinerja model dalam membuat prediksi atau pengklasifikasian. Tujuan dari optimalisasi model adalah untuk menemukan konfigurasi terbaik yang memaksimalkan akurasi atau performa model sesuai dengan metrik evaluasi yang dipilih. Ini sering melibatkan teknik seperti grid search, random search, bayesian optimization, dan lainnya.

Dampak Optimalisasi Model

a. Meningkatkan Akurasi dan Kinerja

Optimalisasi model dapat secara signifikan meningkatkan akurasi prediksi atau klasifikasi, menghasilkan model yang lebih andal dan efektif. Misalnya, dengan menyesuaikan parameter seperti jumlah pohon dalam random forest atau learning rate dalam gradient boosting, kinerja model dapat ditingkatkan.

b. Mengurangi Kesalahan:

Dengan mengoptimalkan model, tingkat kesalahan seperti false positives dan false negatives dapat dikurangi, yang sangat penting dalam aplikasi kritis seperti deteksi penipuan atau diagnosis medis.Optimalisasi dapat membantu menemukan keseimbangan yang tepat antara bias dan varians, mengurangi risiko overfitting atau underfitting.

c. Efisiensi Komputasi:

Proses optimalisasi juga dapat mengarah pada penggunaan sumber daya komputasi yang lebih efisien, dengan menemukan konfigurasi parameter yang memberikan kinerja terbaik dengan biaya komputasi yang lebih rendah. Optimalisasi hyperparameter dapat membantu mengurangi waktu pelatihan dan inferensi.

Pentingnya Optimalisasi Model

a. Meningkatkan Keandalan Model

Optimalisasi model memastikan bahwa model dapat diandalkan dalam berbagai kondisi dan tidak hanya bekerja dengan baik pada data pelatihan tetapi juga pada data yang belum pernah dilihat sebelumnya. Dengan menguji berbagai kombinasi parameter, optimalisasi membantu mengidentifikasi set parameter yang memberikan kinerja konsisten.

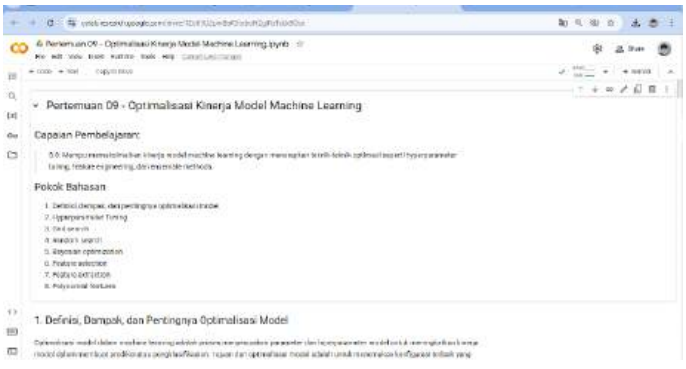
b. Menghadapi Tantangan di Dunia Nyata

Dalam aplikasi dunia nyata, data seringkali tidak sempurna dan memiliki ketidakseimbangan kelas, noise, atau outliers. Optimalisasi model membantu model beradaptasi dan bekerja dengan baik dalam kondisi yang beragam ini. Misalnya, dalam tugas klasifikasi medis, optimalisasi dapat memastikan bahwa model memberikan hasil yang akurat meskipun ada variasi dalam data pasien.

c. Mendukung Pengambilan Keputusan:

Model yang dioptimalkan memberikan prediksi yang lebih akurat dan dapat diandalkan, yang sangat penting dalam pengambilan keputusan bisnis, seperti rekomendasi produk, analisis risiko, dan manajemen rantai pasokan. Optimalisasi model dapat membantu bisnis mengidentifikasi peluang dan risiko dengan lebih baik, mendukung strategi yang lebih efektif.

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



8.2. Hyperparameter Tuning

Hyperparameter tuning adalah proses memilih set parameter terbaik untuk model machine learning untuk meningkatkan kinerjanya. Berbeda dengan parameter model yang dipelajari selama pelatihan (misalnya, bobot dalam regresi linier),

hyperparameter adalah pengaturan yang harus ditentukan sebelum pelatihan model, seperti learning rate dalam neural network, jumlah pohon dalam random forest, atau nilai k dalam k-nearest neighbors (KNN).

Pentingnya Hyperparameter Tuning

1. Meningkatkan Akurasi Model: Hyperparameter yang tepat dapat meningkatkan akurasi model secara signifikan. Model dengan set hyperparameter yang dioptimalkan akan lebih baik dalam memprediksi data yang belum pernah dilihat sebelumnya.
2. Menghindari Overfitting dan Underfitting: Hyperparameter tuning membantu dalam menemukan keseimbangan yang tepat antara bias dan varians. Ini membantu menghindari overfitting (model terlalu sesuai dengan data pelatihan) dan underfitting (model tidak cukup belajar dari data pelatihan).
3. Efisiensi Komputasi: Dengan memilih hyperparameter yang optimal, Anda dapat meningkatkan efisiensi komputasi, mengurangi waktu pelatihan, dan penggunaan sumber daya yang lebih efisien.

8.3. Grid Search

Grid search adalah metode yang secara sistematis menjelajahi ruang hyperparameter yang telah ditentukan oleh pengguna dengan mencoba setiap kombinasi yang mungkin. Proses: Mengatur grid dari parameter, melatih model pada setiap kombinasi, dan memilih kombinasi dengan kinerja terbaik berdasarkan metrik evaluasi yang dipilih. Kelebihan: Memberikan jaminan menemukan kombinasi parameter yang optimal dalam ruang pencarian yang ditentukan. Kekurangan: Memerlukan waktu komputasi yang sangat besar, terutama jika ruang pencarian parameter besar.

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Definisikan model dan parameter grid
model = RandomForestClassifier()
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30]
}

# Grid Search
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

Output:

```
Best Parameters: {'max_depth': 20, 'n_estimators': 200}
Best Score: 0.9462450592885375
```

8.4. Random Search

Definisi: Random search memilih kombinasi hyperparameter secara acak dari ruang pencarian yang telah ditentukan pengguna Proses: Mengatur distribusi dari parameter, melatih model pada sejumlah kombinasi acak, dan memilih kombinasi dengan kinerja terbaik berdasarkan metrik evaluasi yang dipilih. Kelebihan: Lebih efisien daripada grid search, terutama pada ruang pencarian parameter yang besar. Kekurangan: Tidak memberikan jaminan menemukan kombinasi parameter yang optimal tetapi bisa mendekati hasil optimal dengan waktu komputasi lebih sedikit.

```
from sklearn.model_selection import RandomizedSearchCV

# Definisikan model dan parameter distribusi
model = RandomForestClassifier()
param_dist = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'bootstrap': [True, False]
}

# Random Search
random_search                =                RandomizedSearchCV(estimator=model,
param_distributions=param_dist, n_iter=10, cv=5, scoring='accuracy')
random_search.fit(X_train, y_train)

print("Best Parameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)
```

Output:

```
Best Parameters: {'n_estimators': 50, 'max_depth': 20, 'bootstrap': True}
Best Score: 0.9458498023715414
```

8.5. Bayesian Optimization

Definisi: Bayesian optimization menggunakan model probabilistik untuk memilih hyperparameter yang diharapkan menghasilkan kinerja terbaik berdasarkan evaluasi sebelumnya. Proses: Menggunakan proses Gaussian atau model lainnya untuk memperkirakan distribusi dari fungsi tujuan dan memilih hyperparameter yang memaksimalkan ekspektasi perbaikan. Kelebihan: Lebih efisien dalam menemukan hyperparameter optimal dibandingkan grid search dan random search. Kekurangan: Memerlukan pemahaman yang lebih dalam tentang statistik dan probabilitas.

```
!pip install scikit-optimize # Install the skopt package

from skopt import BayesSearchCV
from sklearn.ensemble import RandomForestClassifier
```

```
# Definisikan model dan parameter distribusi
model = RandomForestClassifier()
param_dist = {
    'n_estimators': (50, 200),
    'max_depth': (10, 30),
    'bootstrap': [True, False]
}

# Bayesian Optimization
bayes_search    =    BayesSearchCV(estimator=model,    search_spaces=param_dist,
n_iter=10, cv=5, scoring='accuracy')
bayes_search.fit(X_train, y_train)

print("Best Parameters:", bayes_search.best_params_)
print("Best Score:", bayes_search.best_score_)
```

Output:

```
Collecting scikit-optimize
  Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-
packages (from scikit-optimize) (1.4.2)
Collecting pyaml>=16.9 (from scikit-optimize)
  Downloading pyaml-24.9.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.10/dist-
packages (from scikit-optimize) (1.26.4)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-optimize) (1.13.1)
Requirement       already       satisfied:       scikit-learn>=1.0.0       in
/usr/local/lib/python3.10/dist-packages (from scikit-optimize) (1.5.2)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-
packages (from scikit-optimize) (24.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages
(from pyaml>=16.9->scikit-optimize) (6.0.2)
Requirement       already       satisfied:       threadpoolctl>=3.1.0       in
/usr/local/lib/python3.10/dist-packages    (from    scikit-learn>=1.0.0->scikit-
optimize) (3.5.0)
Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl (107 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
107.8/107.8 kB 3.1 MB/s eta 0:00:00
Downloading pyaml-24.9.0-py3-none-any.whl (24 kB)
Installing collected packages: pyaml, scikit-optimize
Successfully installed pyaml-24.9.0 scikit-optimize-0.10.2
/usr/local/lib/python3.10/dist-packages/numpy/ma/core.py:2820:   RuntimeWarning:
invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
/usr/local/lib/python3.10/dist-packages/numpy/ma/core.py:2820:   RuntimeWarning:
invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
Best   Parameters:   OrderedDict([('bootstrap',   True),   ('max_depth',   29),
('n_estimators', 178)])
Best Score: 0.9458498023715414
```

8.6. Feature Selection

Feature selection adalah proses memilih subset fitur (variabel) yang paling relevan dari data untuk digunakan dalam membangun model machine learning.

Tujuannya adalah untuk meningkatkan kinerja model dengan mengurangi dimensi data, mengurangi risiko overfitting, mempercepat waktu pelatihan, dan meningkatkan interpretabilitas model.

Pentingnya Feature Selection

1. Meningkatkan Kinerja Model: Dengan memilih fitur yang paling relevan, model dapat memfokuskan perhatian pada informasi yang paling berguna, meningkatkan akurasi dan mengurangi noise yang dapat mempengaruhi hasil prediksi.
2. Mengurangi Overfitting: Menggunakan terlalu banyak fitur dapat membuat model terlalu rumit dan mudah overfit pada data pelatihan. Feature selection membantu mengurangi kompleksitas model dengan hanya menggunakan fitur yang penting.
3. Mempercepat Waktu Pelatihan: Mengurangi jumlah fitur dapat mempercepat proses pelatihan dan inferensi model, terutama untuk dataset besar dengan banyak fitur.
4. Meningkatkan Interpretabilitas: Dengan mengurangi jumlah fitur, model menjadi lebih mudah dipahami dan diinterpretasikan, yang penting dalam aplikasi di mana transparansi model sangat penting.

Teknik-Teknik Feature Selection

a. Filter Methods

Definisi: Teknik filter memilih fitur berdasarkan statistik atau metrik individu sebelum proses pelatihan model. Ini dilakukan tanpa mempertimbangkan model machine learning. Teknik Umum: Chi-Square Test: Mengukur independensi antara fitur dan target. Correlation Coefficient: Mengukur hubungan linier antara fitur dan target. Mutual Information: Mengukur ketergantungan antara fitur dan target.

```
# Contoh Implementasi (Chi-Square Test)
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.datasets import load_iris
import pandas as pd

# Load dataset
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

# Chi-Square feature selection
selector = SelectKBest(score_func=chi2, k=2)
X_new = selector.fit_transform(X, y)

print("Selected features:", X.columns[selector.get_support()].tolist())
```

Output:

```
Selected features: ['petal length (cm)', 'petal width (cm)']
```

b. Wrapper Methods

Definisi: Teknik wrapper memilih fitur berdasarkan kinerja model machine learning. Ini melibatkan pelatihan model berulang kali dengan subset fitur yang berbeda untuk menemukan yang terbaik. Teknik Umum: Recursive Feature Elimination (RFE): Menghapus fitur terburuk secara iteratif dan membangun model hingga fitur terbaik ditemukan. Forward Selection: Memulai dengan fitur kosong dan menambahkan fitur satu per satu berdasarkan peningkatan kinerja model. Backward Elimination: Memulai dengan semua fitur dan menghapus fitur satu per satu berdasarkan penurunan kinerja model.

```
# Contoh Implementasi (Recursive Feature Elimination)
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

# Load dataset
data = load_iris()
X = data.data
y = data.target

# Recursive Feature Elimination
model = LogisticRegression(max_iter=1000)
rfe = RFE(model, n_features_to_select=2)
fit = rfe.fit(X, y)

print("Selected features:", fit.support_)
print("Feature ranking:", fit.ranking_)
```

Output:

```
Selected features: [False False  True  True]
Feature ranking: [3 2 1 1]
```

c. Embedded Methods

Definisi: Teknik embedded memilih fitur selama proses pelatihan model, menggabungkan seleksi fitur dan pelatihan model dalam satu langkah. Teknik Umum: Lasso Regression (L1 Regularization): Memilih fitur dengan memberikan penalti pada koefisien regresi, membuat beberapa koefisien menjadi nol. Decision Trees: Menggunakan pentingnya fitur yang dihitung selama pelatihan model decision tree.

```
# Contoh Implementasi (Lasso Regression):
from sklearn.linear_model import Lasso
from sklearn.datasets import load_iris
import pandas as pd

# Load dataset
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

# Lasso feature selection
model = Lasso(alpha=0.1)
model.fit(X, y)

print("Selected features:", X.columns[model.coef_ != 0].tolist())
```

Output:

```
Selected features: ['petal length (cm)']
```

8.7. Feature Extraction

Feature extraction adalah proses mengubah data mentah menjadi fitur yang dapat digunakan oleh model machine learning. Tujuan dari feature extraction adalah untuk menyederhanakan representasi data tanpa kehilangan informasi penting. Ini sangat berguna ketika bekerja dengan data berdimensi tinggi, seperti teks, gambar, atau sinyal, di mana transformasi ke dalam fitur yang bermakna dapat meningkatkan kinerja model dan mempermudah interpretasi.

Pentingnya Feature Extraction

1. Mengurangi Dimensi Data: Feature extraction membantu mengurangi jumlah fitur yang diperlukan, mengurangi beban komputasi dan mempercepat pelatihan model.
2. Meningkatkan Kinerja Model: Dengan mengekstrak fitur yang relevan dan bermakna, model dapat bekerja lebih baik dalam memprediksi atau mengklasifikasikan data, karena data yang tidak relevan atau berisik telah dikurangi.
3. Menyederhanakan Model: Model yang dibangun dari fitur yang diekstraksi seringkali lebih sederhana dan lebih mudah diinterpretasikan, yang penting untuk aplikasi yang membutuhkan transparansi.
4. Menangani Data Berdimensi Tinggi: Dalam kasus data berdimensi tinggi seperti gambar atau teks, feature extraction membantu merangkum informasi penting ke dalam bentuk yang lebih terstruktur dan ringkas.

Teknik-Teknik Feature Extraction

A. Principal Component Analysis (PCA)

Definisi: PCA adalah teknik statistik yang mengubah data mentah menjadi komponen utama yang merupakan kombinasi linier dari variabel asli dengan varians maksimum. Proses: Mengurangi dimensi data dengan memproyeksikan data ke ruang baru yang terdiri dari komponen utama. Contoh Implementasi:

```
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X = data.data

# PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)
print("PCA Components:\n", pca.components_)
```

Output:

```
Explained Variance Ratio: [0.92461872 0.05306648]
PCA Components:
 [[ 0.36138659 -0.08452251  0.85667061  0.3582892 ]
 [ 0.65658877  0.73016143 -0.17337266 -0.07548102]]
```

B. Linear Discriminant Analysis (LDA)

Definisi: LDA adalah teknik yang mencari kombinasi linear dari fitur yang memaksimalkan separasi antar kelas. Proses: Mengurangi dimensi data dengan memproyeksikan data ke ruang yang memaksimalkan jarak antara kelas. Contoh Implementasi:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X = data.data
y = data.target

# LDA
lda = LinearDiscriminantAnalysis(n_components=2)
X_lda = lda.fit_transform(X, y)

print("Explained Variance Ratio:", lda.explained_variance_ratio_)
```

```

```

Output:

```
Explained Variance Ratio: [0.9912126 0.0087874]
```

C. Bag of Words (BoW)

Definisi: BoW adalah teknik representasi teks di mana teks diubah menjadi vektor yang menghitung frekuensi kemunculan kata Proses: Mengubah teks menjadi representasi vektor yang dapat digunakan oleh model machine learning. Contoh Implementas

```
from sklearn.feature_extraction.text import CountVectorizer

# Contoh data teks
corpus = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?"
]

# Bag of Words
vectorizer = CountVectorizer()
X_bow = vectorizer.fit_transform(corpus)

print("Feature Names:", vectorizer.get_feature_names_out())
print("Bag of Words:\n", X_bow.toarray())
```

Output:

```
Feature Names: ['and' 'document' 'first' 'is' 'one' 'second' 'the' 'third' 'this']
Bag of Words:
 [[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

D. Term Frequency-Inverse Document Frequency (TF-IDF)

Definisi: TF-IDF adalah teknik yang memberi bobot pada kata-kata dalam teks berdasarkan frekuensi kemunculannya dalam dokumen dan seberapa jarang kata tersebut muncul di seluruh dokumen. Proses: Mengubah teks menjadi representasi vektor yang mempertimbangkan pentingnya kata dalam konteks dokumen dan korpus. Contoh Implementasi:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Contoh data teks
corpus = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?"
]

# TF-IDF
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(corpus)

print("Feature Names:", vectorizer.get_feature_names_out())
print("TF-IDF:\n", X_tfidf.toarray())
```

Output:

```
Feature Names: ['and' 'document' 'first' 'is' 'one' 'second' 'the' 'third' 'this']
TF-IDF:
 [[0.         0.46979139 0.58028582 0.38408524 0.         0.
  0.38408524 0.         0.38408524]
 [0.         0.6876236  0.         0.28108867 0.         0.53864762
  0.28108867 0.         0.28108867]
 [0.51184851 0.         0.         0.26710379 0.51184851 0.
  0.26710379 0.51184851 0.26710379]
 [0.         0.46979139 0.58028582 0.38408524 0.         0.
  0.38408524 0.         0.38408524]]
```

E. Convolutional Neural Networks (CNNs) for Images

Definisi: CNN adalah jenis neural network yang dirancang untuk memproses data grid seperti gambar. Proses: Menggunakan lapisan konvolusi untuk mengekstrak fitur dari gambar, seperti tepi, tekstur, dan objek. Contoh Implementasi:

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Contoh model CNN sederhana
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# Menampilkan arsitektur model
model.summary()
```

Output:

```
/usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer
using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param
# ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 62, 62, 32)          │             896
│
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
─┤
│ max_pooling2d (MaxPooling2D)         │ (None, 31, 31, 32)          │               0
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
─┤
│ conv2d_1 (Conv2D)                    │ (None, 29, 29, 64)          │          18,496
│
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
─┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 14, 14, 64)          │               0
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
─┤
│ conv2d_2 (Conv2D)                    │ (None, 12, 12, 64)          │          36,928
│
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
─┘
 Total params: 56,320 (220.00 KB)
 Trainable params: 56,320 (220.00 KB)
 Non-trainable params: 0 (0.00 B)
```

8.8. Polynomial Features

Polynomial features adalah teknik transformasi yang menambah fitur baru ke dataset dengan menghitung semua kombinasi polinomial dari fitur yang ada hingga derajat tertentu. Teknik ini memungkinkan model untuk menangkap hubungan non-linier antara fitur, yang bisa meningkatkan kemampuan prediktif model dalam beberapa kasus.

Tujuan Polynomial Features

1. Menangkap Hubungan Non-Linear: Polynomial features memungkinkan model linear untuk menangkap hubungan non-linear antara fitur dan target, yang meningkatkan fleksibilitas dan kinerja model.
2. Meningkatkan Kinerja Model: Dengan menambah fitur baru yang merepresentasikan interaksi dan eksponen dari fitur asli, model bisa menjadi lebih kuat dalam menangkap pola yang kompleks dalam data.

3. Meningkatkan Kekuatan Prediktif: Polynomial features bisa membantu model untuk lebih baik dalam memprediksi data yang memiliki hubungan non-linear, yang tidak bisa ditangkap oleh model linear sederhana.

Teknik-Teknik Polynomial Features

PolynomialFeatures dari scikit-learn. Definisi: PolynomialFeatures adalah kelas dalam scikit-learn yang digunakan untuk menghasilkan fitur polinomial dari dataset yang ada. Proses: Menghitung semua kombinasi polinomial dari fitur yang ada hingga derajat tertentu. Contoh Implementasi:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures

# Contoh dataset
data = {
    'Feature1': [1, 2, 3],
    'Feature2': [4, 5, 6]
}
df = pd.DataFrame(data)

# Polynomial Features
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(df)

# Menampilkan fitur polinomial
poly_feature_names = poly.get_feature_names_out(input_features=df.columns)
df_poly = pd.DataFrame(poly_features, columns=poly_feature_names)

print(df_poly)
```

Output:

```
   Feature1  Feature2  Feature1^2  Feature1 Feature2  Feature2^2
0       1.0       4.0         1.0                4.0        16.0
1       2.0       5.0         4.0               10.0        25.0
2       3.0       6.0         9.0               18.0        36.0
```

Dalam contoh di atas, kita menggunakan PolynomialFeatures dari scikit-learn untuk menghasilkan fitur polinomial dari dataset yang memiliki dua fitur (Feature1 dan Feature2). Dengan menetapkan derajat polinomial ke 2, PolynomialFeatures akan menghasilkan fitur baru yang mencakup semua kombinasi linier, kuadrat, dan produk silang dari fitur asli.

8.9. Apa itu Computer Vision

Pada video ini, akan dibahas tentang konsep Computer Vision dan cara kerjanya. Computer Vision merupakan cabang kecerdasan buatan yang memungkinkan komputer untuk "melihat" serta memahami konten dalam gambar dan video. Video ini mencakup berbagai aplikasi Computer Vision, seperti pengenalan wajah, deteksi objek, pengenalan teks dari gambar (OCR), dan analisis video. Selain itu, dijelaskan juga teknik-teknik dasar dalam Computer Vision, antara lain pengolahan citra, convolutional neural networks (CNN), dan deep learning. Dengan kemampuannya untuk menganalisis data visual, Computer Vision diterapkan di berbagai bidang industri, seperti keamanan, kesehatan, dan otomotif, yang membantu meningkatkan efisiensi serta membuka inovasi baru dalam interaksi manusia dan mesin.

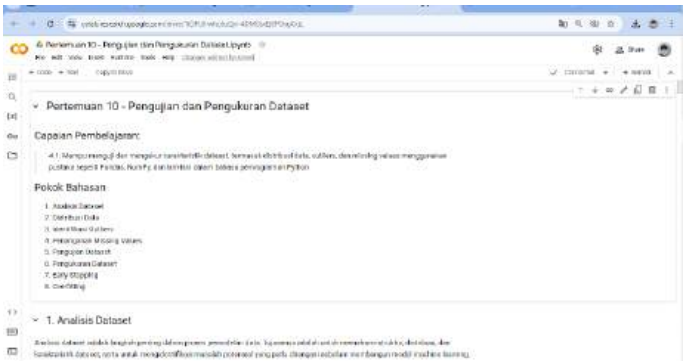
Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 9. Pengujian dan Pengukuran Dataset

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan Pandas dan NumPy untuk analisis data.
2. Analisis distribusi data.
3. Identifikasi dan penanganan outliers.
4. Penanganan missing values.
5. Teknik pengukuran karakteristik dataset.

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



9.1. Analisis Dataset

Analisis dataset adalah langkah penting dalam proses pemodelan data. Tujuannya adalah untuk memahami struktur, distribusi, dan karakteristik dataset, serta untuk mengidentifikasi masalah potensial yang perlu ditangani sebelum membangun model machine learning. Analisis dataset melibatkan beberapa langkah utama seperti eksplorasi data, visualisasi data, analisis statistik, dan penanganan data yang hilang atau tidak valid.

Langkah-langkah Analisis Dataset

1. Eksplorasi Data (Data Exploration): Memahami Struktur Data: Melihat bentuk dan ukuran dataset, jenis fitur (numerik, kategorikal), dan tipe data. Melihat Data Sekilas: Menggunakan fungsi seperti head(), tail(), info(), dan describe() untuk melihat ringkasan dataset.

2. Visualisasi Data (Data Visualization): Visualisasi Distribusi: Menggunakan histogram, boxplot, dan density plot untuk melihat distribusi setiap fitur. Visualisasi Hubungan: Menggunakan scatter plot, pair plot, dan heatmap untuk melihat hubungan antara fitur. Visualisasi Kategori: Menggunakan bar plot dan count plot untuk melihat distribusi data kategorikal.

3. Analisis Statistik: Statistik Deskriptif: Menghitung statistik deskriptif seperti mean, median, modus, standar deviasi, dan korelasi. Uji Hipotesis: Menggunakan uji statistik seperti uji t, uji chi-square, dan ANOVA untuk memahami hubungan antara fitur.
4. Penanganan Data yang Hilang dan Tidak Valid: Mengidentifikasi Missing Values: Melihat jumlah dan distribusi nilai yang hilang. Mengimputasi Missing Values: Mengisi nilai yang hilang dengan mean, median, modus, atau menggunakan teknik imputasi lanjutan. Mengidentifikasi dan Mengatasi Outliers: Menggunakan teknik statistik dan visualisasi untuk mengidentifikasi dan menangani outliers.
5. Feature Engineering: Membuat Fitur Baru: Membuat fitur baru yang dapat membantu model memahami data dengan lebih baik Normalisasi dan Standarisasi: Mengubah skala fitur untuk memastikan model bekerja dengan baik.

Contoh Implementasi Analisis Dataset

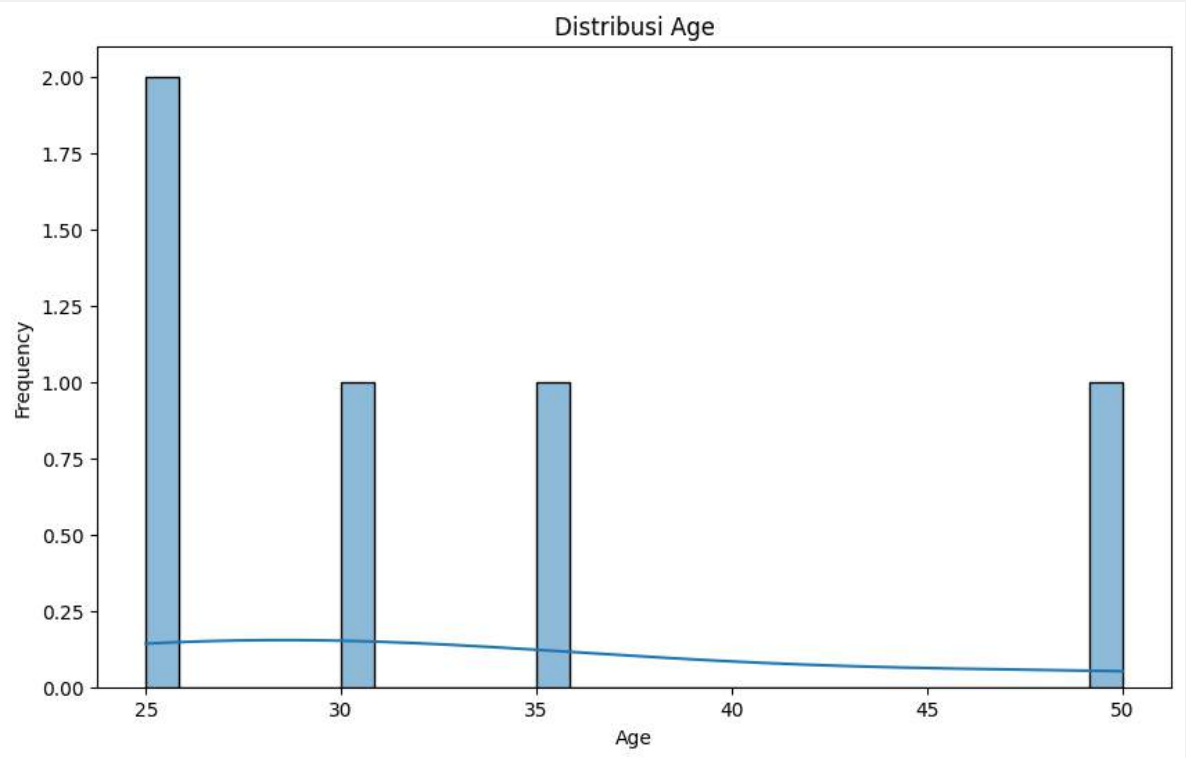
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Membaca dataset
df = pd.read_csv('data.csv')

# Melihat struktur data
print("Shape of the dataset:", df.shape)
print("\nInfo about the dataset:")
print(df.info())

# Melihat data sekilas
print("\nFirst few rows of the dataset:")
print(df.head())
print("\nStatistical summary of the dataset:")
print(df.describe())

# Visualisasi distribusi
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], bins=30, kde=True)
plt.title('Distribusi Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

# Visualisasi hubungan
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Income', data=df)
plt.title('Hubungan Age dan Income')
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()

# Visualisasi kategori
```

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Gender', data=df)
plt.title('Distribusi Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

# Analisis statistik
print("\nCorrelation matrix:")
print(df.corr())

# Mengidentifikasi missing values
print("\nMissing values in each column:")
print(df.isnull().sum())

# Mengimputasi missing values dengan mean
df['Age'].fillna(df['Age'].mean(), inplace=True)

# Mengidentifikasi outliers dengan boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Age'])
plt.title('Boxplot Age')
plt.xlabel('Age')
plt.show()

# Normalisasi fitur numerik
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['Age', 'Income']] = scaler.fit_transform(df[['Age', 'Income']])

# Membuat fitur baru
df['Age_Income_Ratio'] = df['Age'] / df['Income']

# Melihat data setelah preprocessing
print("\nFirst few rows of the dataset after preprocessing:")
print(df.head())
```

Output:

```
Shape of the dataset: (6, 3)

Info about the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    5 non-null      object
 1   Age     5 non-null      float64
 2   City    6 non-null      object
dtypes: float64(1), object(2)
memory usage: 272.0+ bytes
None

First few rows of the dataset:
      Name   Age         City
0    Alice  25.0     New York
1      Bob  30.0  Los Angeles
2  Charlie  35.0      Chicago
```

```
3    Alice  25.0     New York
4      Eve   NaN        Miami

Statistical summary of the dataset:
             Age
count   5.000000
mean   33.000000
std    10.368221
min    25.000000
25%    25.000000
50%    30.000000
75%    35.000000
max    50.000000
```



9.2. Distribusi Data

Distribusi data mengacu pada cara nilai-nilai dalam dataset tersebar atau terdistribusi di sepanjang rentang nilainya. Memahami distribusi data penting dalam analisis statistik karena dapat memberikan wawasan tentang karakteristik data seperti pusat data (mean, median), sebaran data (deviasi standar), serta kecenderungan atau anomali dalam data.

Jenis Distribusi Data

a. Distribusi Normal (Gaussian)

Distribusi normal adalah distribusi simetris di sekitar mean, di mana sebagian besar nilai berpusat di sekitar mean dengan sedikit nilai yang tersebar di kedua ekstrem. Contoh: Tinggi badan manusia, berat badan dalam populasi tertentu.

b. Distribusi Uniform

Distribusi uniform terjadi ketika setiap nilai dalam rentang memiliki probabilitas yang sama untuk muncul, sehingga bentuknya menjadi persegi atau datar. Contoh: Hasil dari pelemparan koin (nilai 0 atau 1), atau hasil pelemparan dadu (nilai 1 hingga 6).

c. Distribusi Binomial

Distribusi binomial muncul ketika ada dua hasil yang mungkin dari suatu percobaan (berhasil atau gagal), dengan sejumlah percobaan yang tetap. Contoh: Hasil dari serangkaian percobaan yang terdiri dari dua kemungkinan hasil (misalnya, hasil ujian dengan jawaban benar atau salah).

d. Distribusi Poisson

Distribusi Poisson digunakan untuk menggambarkan jumlah peristiwa yang terjadi dalam interval waktu atau ruang tertentu, dengan angka rata-rata kejadian yang diketahui dan independen dari waktu sebelumnya. Contoh: Jumlah panggilan ke pusat layanan pelanggan dalam satu jam.

Mengidentifikasi Distribusi Data

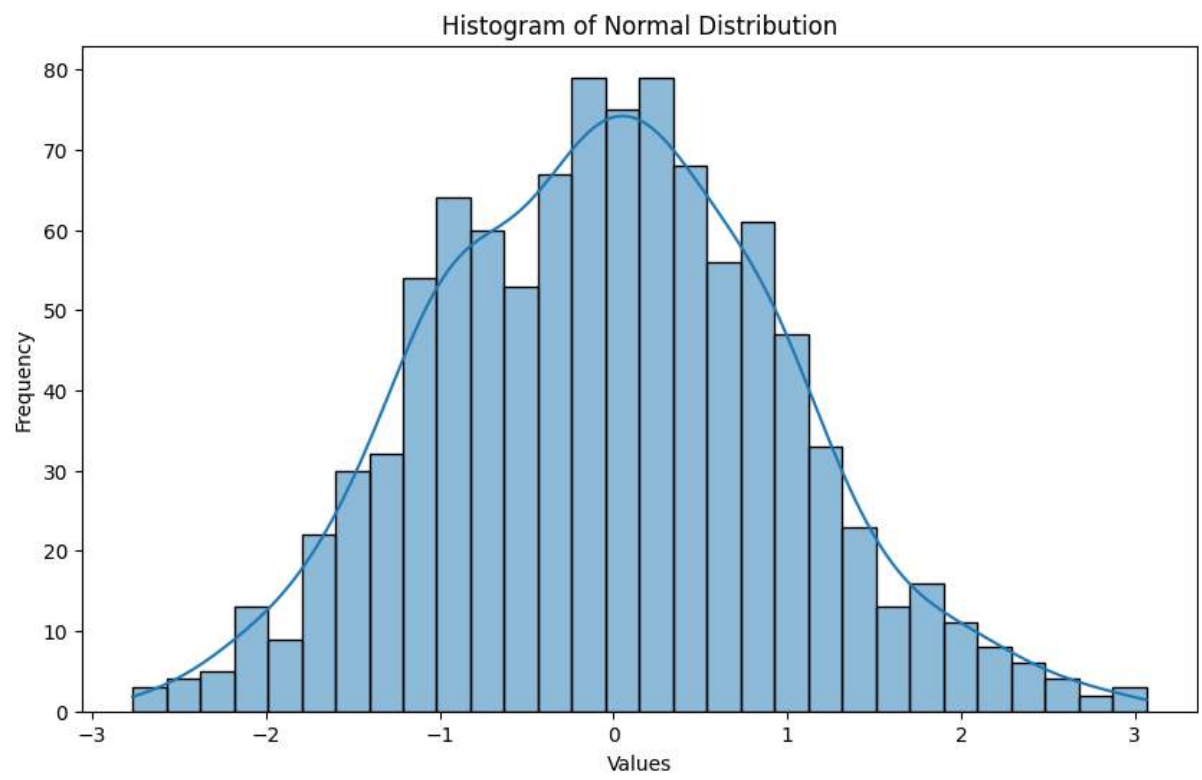
Untuk mengidentifikasi distribusi data, Anda dapat menggunakan beberapa teknik berikut:

a. Histogram

Digunakan untuk melihat distribusi data numerik dengan membagi rentang nilai menjadi beberapa interval dan menghitung jumlah observasi di setiap interval.

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Generate data from a normal distribution
data_normal = np.random.normal(loc=0, scale=1, size=1000)

# Plot histogram
plt.figure(figsize=(10, 6))
sns.histplot(data_normal, bins=30, kde=True)
plt.title('Histogram of Normal Distribution')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```

Output:



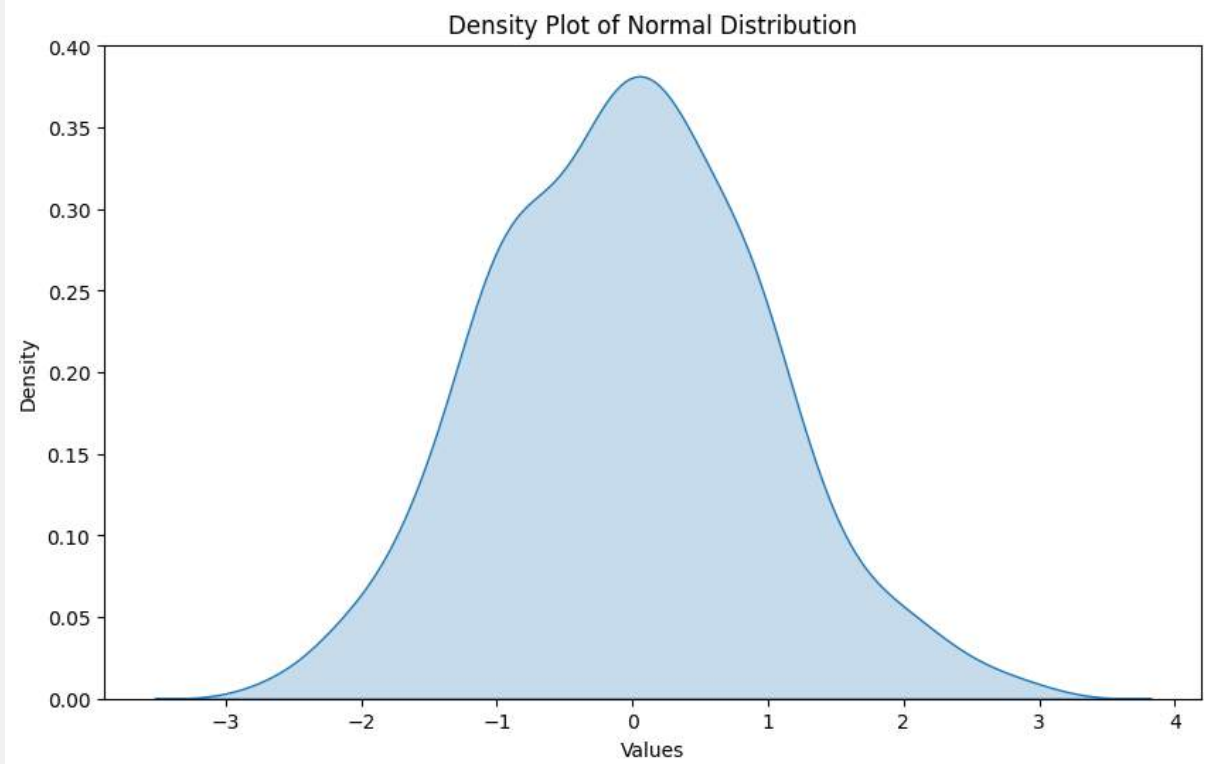
b. Density Plot

Mirip dengan histogram, tetapi menggunakan kurva kernel (kernel density estimation, KDE) untuk menunjukkan distribusi data secara kontinu.

```
# Plot density plot
plt.figure(figsize=(10, 6))
sns.kdeplot(data_normal, shade=True)
plt.title('Density Plot of Normal Distribution')
plt.xlabel('Values')
plt.ylabel('Density')
plt.show()
```

Output:

```
<ipython-input-4-08b2a097c5ff>:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data_normal, shade=True)
```

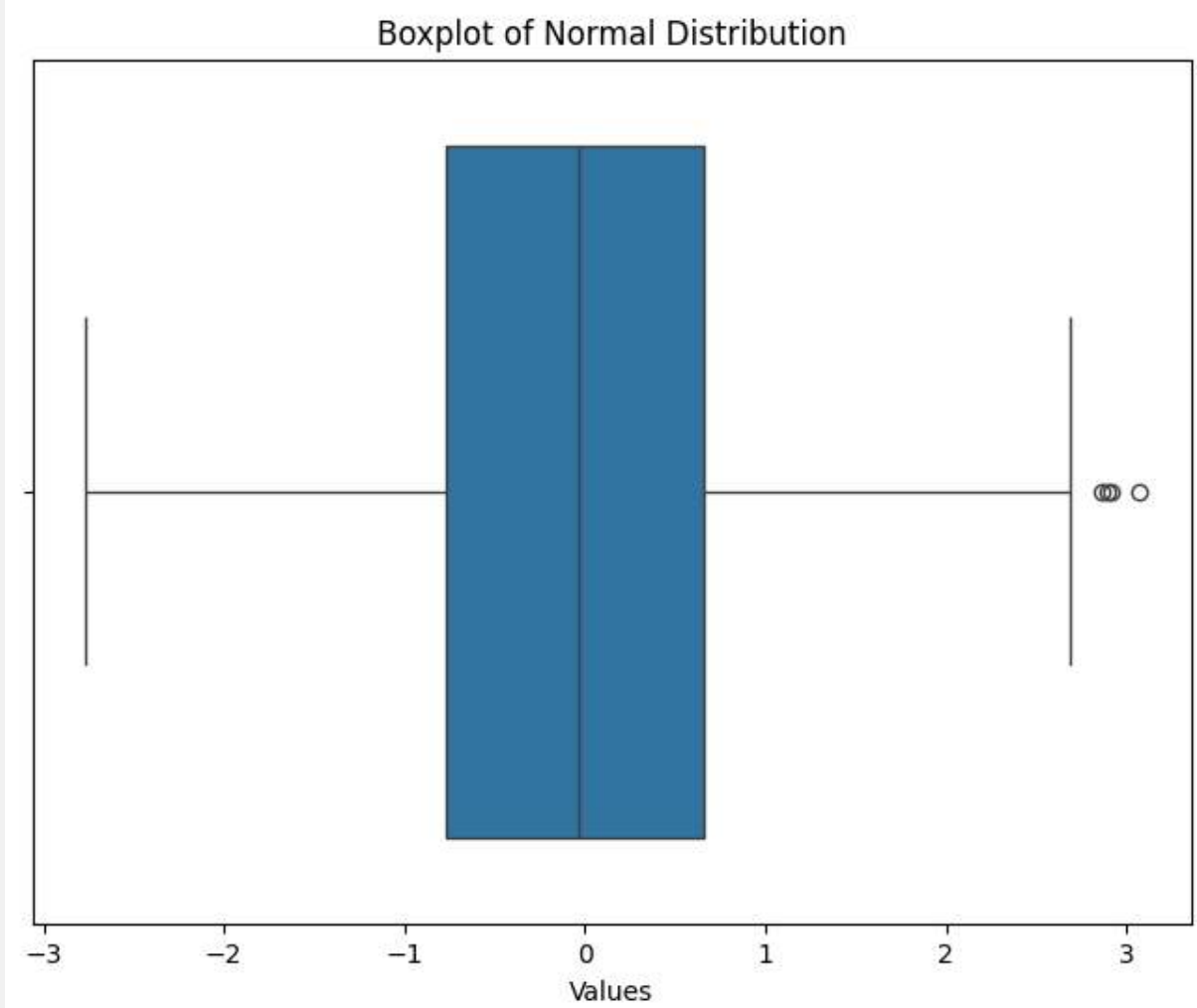


c. Boxplot

Boxplot Digunakan untuk melihat distribusi data serta mengidentifikasi outlier.

```
# Plot boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x=data_normal)
plt.title('Boxplot of Normal Distribution')
plt.xlabel('Values')
plt.show()
```

Output:



Memahami distribusi data adalah langkah kunci dalam analisis data, karena memungkinkan kita untuk mengidentifikasi pola, anomali, dan kecenderungan yang ada dalam dataset. Dengan memilih metode visualisasi yang tepat, seperti histogram, density plot, atau boxplot, Anda dapat mendapatkan wawasan yang lebih dalam tentang bagaimana data didistribusikan dan menentukan langkah-langkah selanjutnya dalam analisis atau pemodelan data.

9.3. Identifikasi Outliers

Outliers adalah nilai-nilai dalam dataset yang berada jauh dari sebagian besar nilai lainnya. Mereka dapat menunjukkan kesalahan data, variabilitas yang luar biasa, atau poin data yang sangat berbeda. Mengidentifikasi outliers penting karena mereka dapat sangat mempengaruhi analisis statistik dan model machine learning.

Mengapa Outliers Penting?

a. Pengaruh pada Statistik: Outliers dapat mempengaruhi mean, standard deviation, dan nilai lainnya.

b. Pengaruh pada Model Machine Learning: Outliers dapat mengurangi akurasi model dengan membuat model overfit atau underfit.

c. Kesalahan Pengukuran: Outliers bisa menjadi indikator adanya kesalahan dalam proses pengumpulan data.

Metode untuk Mengidentifikasi Outliers

a. Visualisasi Data

- Boxplot: Menampilkan data berdasarkan kuartil. Outliers biasanya terlihat sebagai titik-titik yang terpisah dari box.
- Scatter Plot: Berguna untuk data dua dimensi, outliers terlihat sebagai titik yang jauh dari pola umum data.

b. Histogram: Membantu melihat distribusi data dan mengidentifikasi nilai yang berada jauh dari distribusi normal.

c. Metode Statistik

- IQR (Interquartile Range): Nilai yang berada di bawah Q1 - 1.5IQR atau di atas Q3 + 1.5IQR dianggap sebagai outliers.
- Z-Score: Menunjukkan seberapa jauh nilai dari mean. Biasanya, nilai dengan z-score di atas 3 atau di bawah -3 dianggap sebagai outliers.

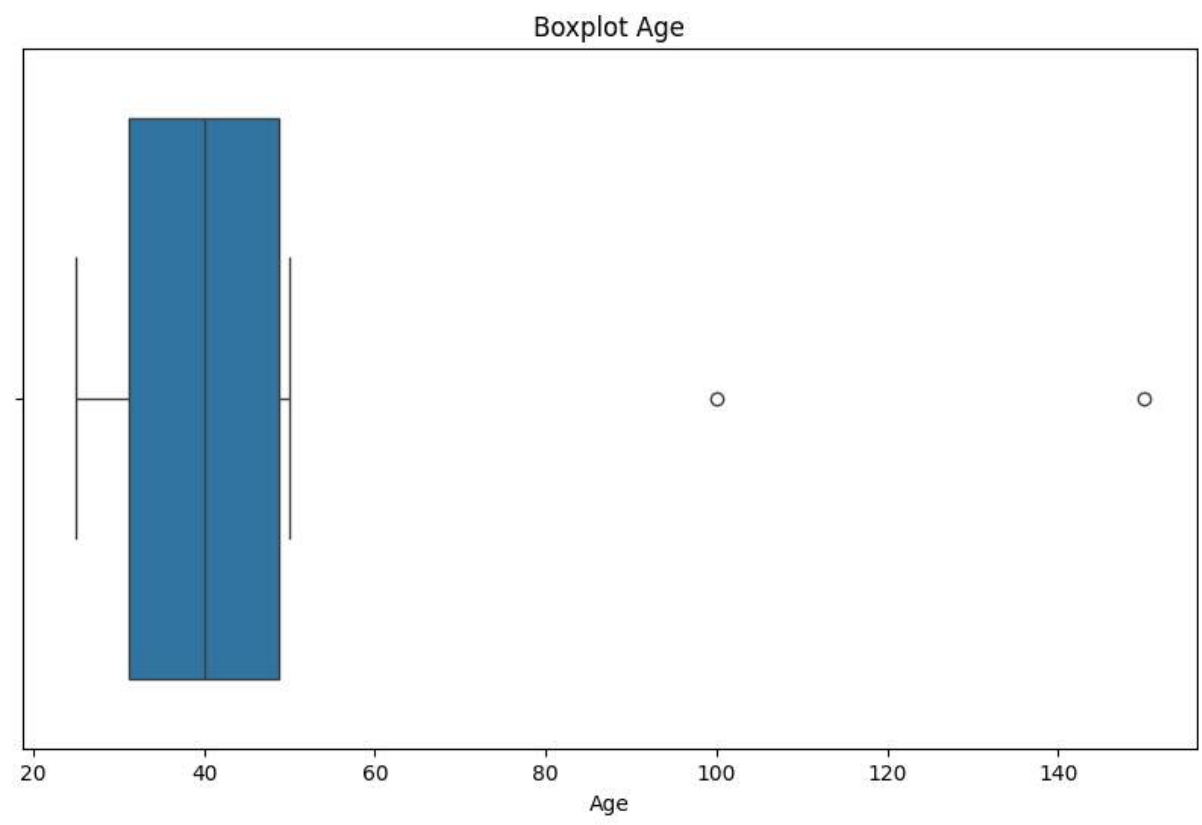
Contoh Implementasi Identifikasi Outliers

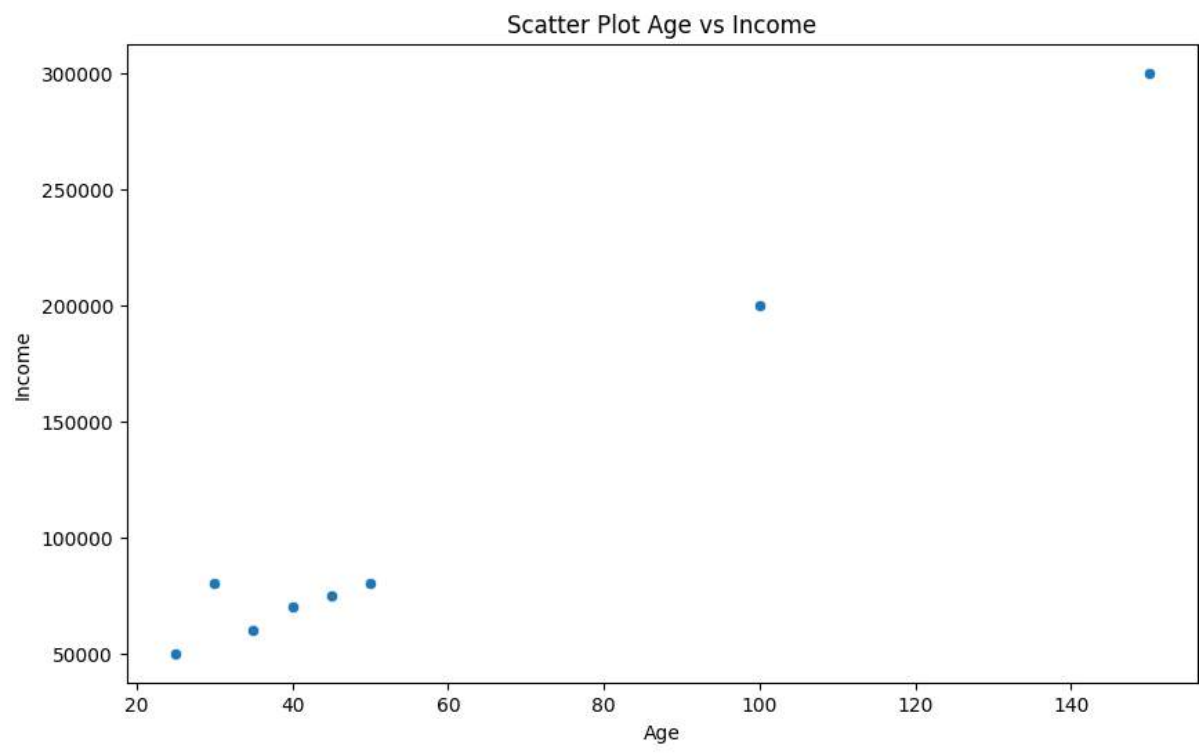
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Membuat data sampel
data = {
    'Age': [25, 30, 35, 40, 100, 45, 50, 25, 30, 35, 40, 45, 50, 150],
    'Income': [50000, 80000, 60000, 70000, 200000, 75000, 80000, 50000, 80000,
60000, 70000, 75000, 80000, 300000]
}

# Membuat dataframe
df = pd.DataFrame(data)

# Menampilkan data
print(df)

# Visualisasi Boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Age'])
plt.title('Boxplot Age')
plt.xlabel('Age')
plt.show()
```

```
# Visualisasi Scatter Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Income', data=df)
plt.title('Scatter Plot Age vs Income')
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()

# Identifikasi Outliers dengan IQR
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1

outliers_IQR = df[(df['Age'] < (Q1 - 1.5 * IQR)) | (df['Age'] > (Q3 + 1.5 * IQR))]
print("\nOutliers berdasarkan IQR:")
print(outliers_IQR)

# Identifikasi Outliers dengan Z-Score
from scipy import stats

z_scores = np.abs(stats.zscore(df['Age']))
outliers_z = df[(z_scores > 3)]
print("\nOutliers berdasarkan Z-Score:")
print(outliers_z)
```

Output:

```
    Age  Income
0    25   50000
1    30   80000
2    35   60000
3    40   70000
4   100  200000
5    45   75000
6    50   80000
7    25   50000
8    30   80000
9    35   60000
10   40   70000
11   45   75000
12   50   80000
13  150  300000
```






```
Outliers berdasarkan IQR:
    Age  Income
4   100  200000
13  150  300000

Outliers berdasarkan Z-Score:
    Age  Income
```

```
13  150  300000
```

Penjelasan Coding

1. Boxplot: Boxplot menunjukkan distribusi Age dan outliers sebagai titik-titik di luar garis whisker.
2. Scatter Plot: Scatter plot menunjukkan hubungan antara Age dan Income dan memungkinkan identifikasi outliers berdasarkan penyimpangan dari pola umum.
3. IQR: Menghitung IQR untuk Age, kemudian menentukan nilai-nilai yang berada di luar Q1 - 1.5IQR dan Q3 + 1.5IQR sebagai outliers.
4. Z-Score: Menghitung z-score untuk Age dan menentukan nilai-nilai dengan z-score lebih besar dari 3 atau kurang dari -3 sebagai outliers.

Outliers yang diidentifikasi dengan metode IQR dan z-score mungkin berbeda tergantung pada distribusi data. Dengan menggunakan kombinasi metode ini, Anda dapat memperoleh pemahaman yang lebih komprehensif tentang outliers dalam dataset Anda.

9.4. Penanganan Missing Values

Penanganan missing values adalah langkah penting dalam preprocessing data. Data yang hilang dapat menyebabkan bias dalam analisis dan model machine learning, sehingga perlu ditangani dengan benar. Berikut adalah beberapa teknik umum untuk menangani missing values beserta contohnya:

Teknik Penanganan Missing Values

a. Menghapus Missing Values

- Jika jumlah missing values kecil dan tidak signifikan, data yang hilang dapat dihapus.
- df.dropna() menghapus semua baris yang memiliki missing values.
- df.dropna(axis=1) menghapus semua kolom yang memiliki missing values.

b. Mengisi Missing Values

Mean/Median/Mode Imputation: Mengisi missing values dengan mean, median, atau mode dari kolom tersebut.

- df['column'].fillna(df['column'].mean(), inplace=True)
- df['column'].fillna(df['column'].median(), inplace=True)
- df['column'].fillna(df['column'].mode()[0], inplace=True)

Forward Fill (ffill): Mengisi missing values dengan nilai sebelumnya.

- df.fillna(method='ffill', inplace=True)

Backward Fill (bfill): Mengisi missing values dengan nilai setelahnya.

- df.fillna(method='bfill', inplace=True)

Interpolate: Mengisi missing values dengan interpolasi linear.

- df.interpolate(method='linear', inplace=True)

Menggunakan Model Prediktif Menggunakan model machine learning untuk memprediksi missing values berdasarkan fitur lain.

```python from sklearn.impute import SimpleImputer imputer = SimpleImputer(strategy='mean') df[['column']] = imputer.fit_transform(df[['column']])

Contoh Implementasi

```
import pandas as pd
import numpy as np

# Membuat data sampel dengan missing values
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [25, np.nan, 35, 45, np.nan],
    'Income': [50000, 60000, np.nan, 80000, 90000],
    'Gender': ['F', 'M', np.nan, 'M', 'F']
}

# Membuat dataframe
df = pd.DataFrame(data)

# Menampilkan data
print("Original DataFrame:")
print(df)

# Menghapus baris dengan missing values
df_dropna = df.dropna()
print("\nDataFrame setelah menghapus baris dengan missing values:")
print(df_dropna)

Mengisi missing values dengan mean (untuk kolom numerik)
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['Income'].fillna(df['Income'].mean(), inplace=True)
print("\nDataFrame setelah mengisi missing values dengan mean:")
print(df)

Mengisi missing values dengan mode (untuk kolom kategorikal)
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
print("\nDataFrame setelah mengisi missing values dengan mode:")
print(df)

# Mengisi missing values dengan metode forward fill
df_ffill = df.fillna(method='ffill')
print("\nDataFrame setelah mengisi missing values dengan forward fill:")
```
```

```
print(df_ffill)

# Mengisi missing values dengan metode backward fill
df_bfill = df.fillna(method='bfill')
print("\nDataFrame setelah mengisi missing values dengan backward fill:")
print(df_bfill)

# Mengisi missing values dengan interpolasi
df_interpolate = df.interpolate(method='linear')
print("\nDataFrame setelah mengisi missing values dengan interpolasi:")
print(df_interpolate)
```

Output:

```
      Name   Age   Income Gender
0    Alice  25.0  50000.0      F
1      Bob  35.0  60000.0      M
2  Charlie  35.0  70000.0      F
3    David  45.0  80000.0      M
4      Eve  35.0  90000.0      F

DataFrame setelah mengisi missing values dengan forward fill:
      Name   Age   Income Gender
0    Alice  25.0  50000.0      F
1      Bob  35.0  60000.0      M
2  Charlie  35.0  70000.0      F
3    David  45.0  80000.0      M
4      Eve  35.0  90000.0      F

DataFrame setelah mengisi missing values dengan backward fill:
      Name   Age   Income Gender
0    Alice  25.0  50000.0      F
1      Bob  35.0  60000.0      M
2  Charlie  35.0  70000.0      F
3    David  45.0  80000.0      M
4      Eve  35.0  90000.0      F

DataFrame setelah mengisi missing values dengan interpolasi:
      Name   Age   Income Gender
0    Alice  25.0  50000.0      F
1      Bob  35.0  60000.0      M
2  Charlie  35.0  70000.0      F
3    David  45.0  80000.0      M
4      Eve  35.0  90000.0      F
<ipython-input-7-96a7ade40086>:25: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].mean(), inplace=True)
<ipython-input-7-96a7ade40086>:26: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
```

```
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Income'].fillna(df['Income'].mean(), inplace=True)
<ipython-input-7-96a7ade40086>:31: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
<ipython-input-7-96a7ade40086>:36: FutureWarning: DataFrame.fillna with 'method'
is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
  df_ffill = df.fillna(method='ffill')
<ipython-input-7-96a7ade40086>:41: FutureWarning: DataFrame.fillna with 'method'
is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
  df_bfill = df.fillna(method='bfill')
<ipython-input-7-96a7ade40086>:46: FutureWarning: DataFrame.interpolate with
object dtype is deprecated and will raise in a future version. Call
obj.infer_objects(copy=False) before interpolating instead.
  df_interpolate = df.interpolate(method='linear')
```

9.5. Pengujian Dataset

Pengujian dataset adalah langkah penting dalam alur kerja analisis data dan machine learning. Ini mencakup eksplorasi awal untuk memahami struktur data, pemeriksaan konsistensi dan integritas, serta evaluasi kinerja model terhadap data tersebut. Berikut adalah beberapa langkah dan teknik yang digunakan dalam pengujian dataset. Langkah-langkah Pengujian Dataset:

a. Eksplorasi Data (Data Exploration)

Memahami Struktur Data: Mengidentifikasi tipe data, jumlah fitur, dan baris. Statistik Deskriptif: Menggunakan statistik deskriptif untuk mendapatkan wawasan awal tentang data. Visualisasi Data: Menggunakan berbagai plot untuk memahami distribusi dan hubungan antar variabel.

b. Pembersihan Data (Data Cleaning)

Menangani Missing Values: Mengisi atau menghapus data yang hilang. Mengidentifikasi dan Menghapus Outliers: Menggunakan teknik statistik dan

visualisasi untuk mengidentifikasi outliers. Menghapus Duplikat: Menghapus baris yang duplikat untuk memastikan integritas data.

c. Transformasi Data (Data Transformation)

Normalisasi/Standarisasi Data: Mengubah skala data sehingga berada dalam rentang tertentu atau memiliki distribusi normal Mengubah Data Kategorikal ke Numerik: Menggunakan teknik seperti one-hot encoding atau label encoding.

d. Splitting Dataset

Train-Test Split: Memisahkan dataset menjadi set pelatihan dan pengujian untuk mengevaluasi kinerja model. K-Fold Cross-Validation: Membagi dataset menjadi beberapa bagian untuk validasi silang.

e. Evaluasi Model

Metrik Evaluasi: Menggunakan metrik seperti accuracy, precision, recall, F1 score, dan ROC-AUC untuk mengevaluasi model Confusion Matrix: Menyediakan visualisasi performa model dalam klasifikasi.

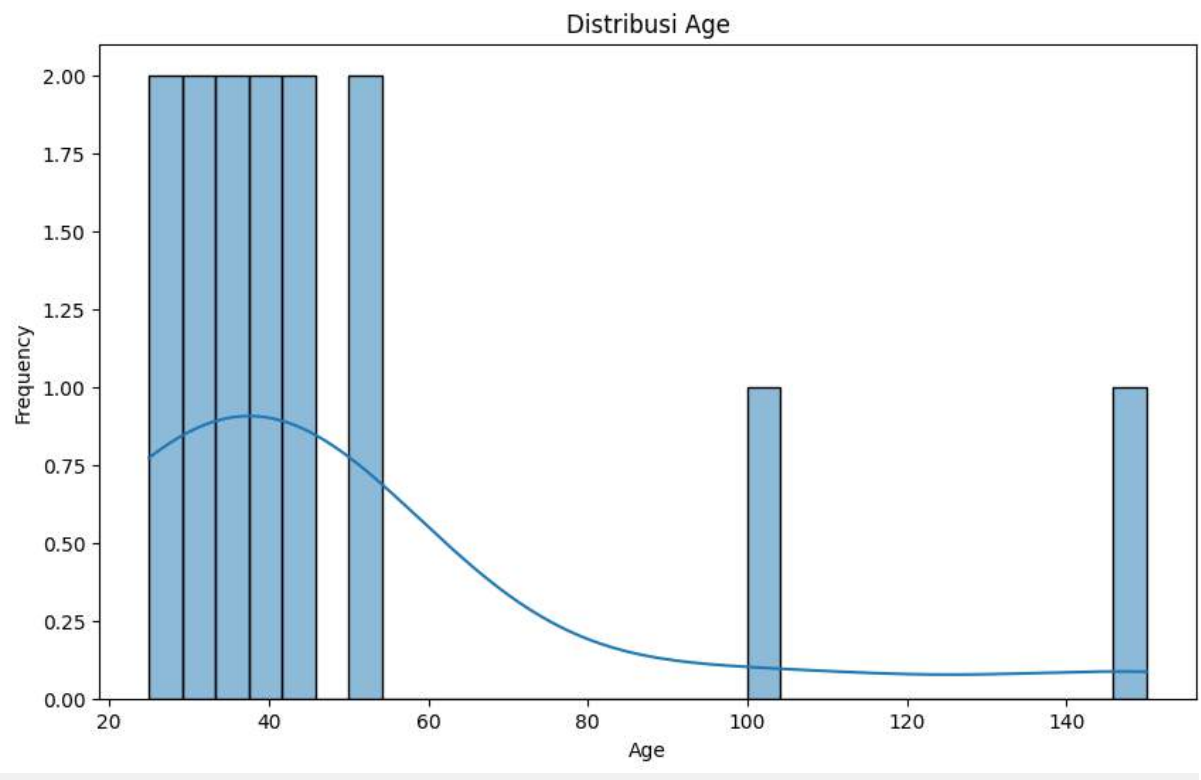
```
# Contoh Implementasi Pengujian Dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression

# Membuat data sampel
data = {
    'Age': [25, 30, 35, 40, 100, 45, 50, 25, 30, 35, 40, 45, 50, 150],
    'Income': [50000, 80000, 60000, 70000, 200000, 75000, 80000, 50000, 80000,
60000, 70000, 75000, 80000, 300000],
     'Gender': ['Male', 'Female', 'Male', 'Female', 'Female', 'Male', 'Male',
'Male', 'Female', 'Male', 'Female', 'Female', 'Male', 'Male'],
    'Purchased': ['No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes',
'No', 'Yes', 'No', 'Yes', 'No']
}

# Membuat dataframe
df = pd.DataFrame(data)

# Menampilkan data sekilas
print("Original DataFrame:")
print(df.head())

# Eksplorasi Data
print("\nStatistical summary of the dataset:")
print(df.describe())

# Visualisasi Data
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], bins=30, kde=True)
```

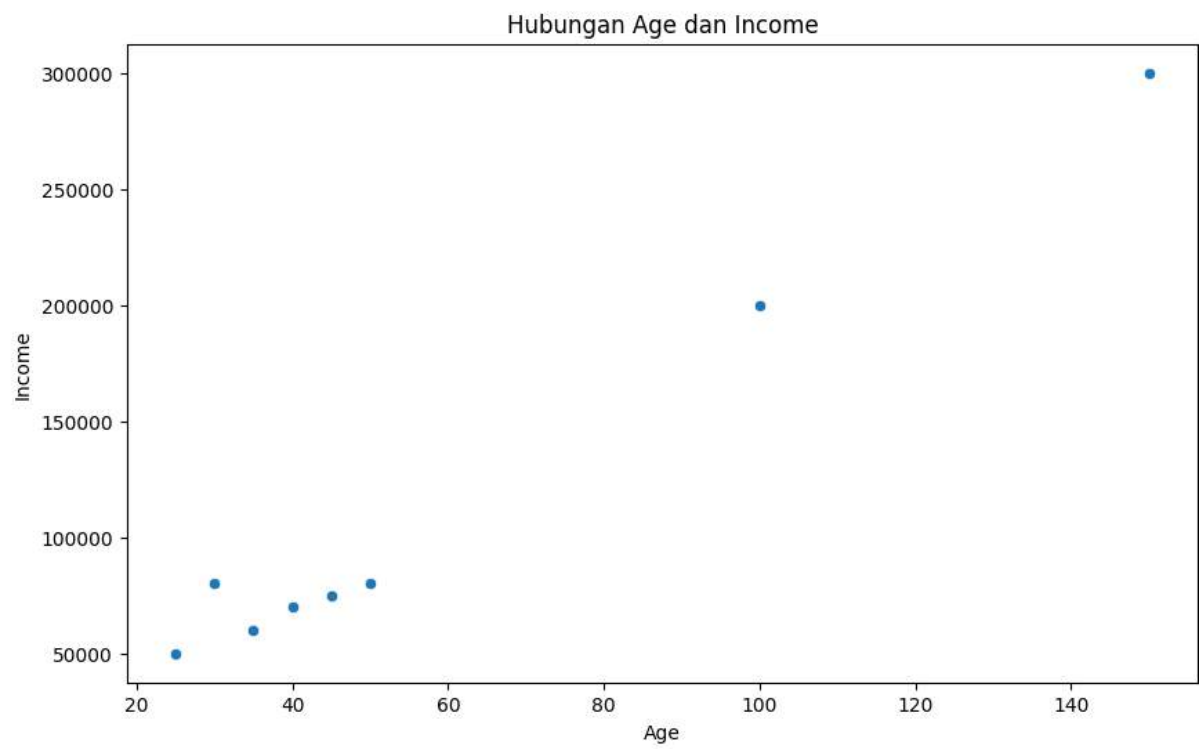
```
plt.title('Distribusi Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Income', data=df)
plt.title('Hubungan Age dan Income')
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='Gender', data=df)
plt.title('Distribusi Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

# Menangani Missing Values dan Outliers
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['Income'].fillna(df['Income'].mean(), inplace=True)

Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df['Age'] < (Q1 - 1.5 * IQR)) | (df['Age'] > (Q3 + 1.5 * IQR)))]

# Transformasi Data
X = df[['Age', 'Income', 'Gender']]
y = df['Purchased']

# Mengubah Data Kategorikal ke Numerik
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), ['Gender'])],
remainder='passthrough')
X = ct.fit_transform(X)

# Normalisasi Data
sc = StandardScaler()
X = sc.fit_transform(X)

# Splitting Dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

# Training Model
classifier = LogisticRegression()
classifier.fit(X_train, y_train)

# Prediksi
y_pred = classifier.predict(X_test)

# Evaluasi Model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='Yes')
recall = recall_score(y_test, y_pred, pos_label='Yes')
f1 = f1_score(y_test, y_pred, pos_label='Yes')
cm = confusion_matrix(y_test, y_pred)

print("\nModel Performance:")
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
```

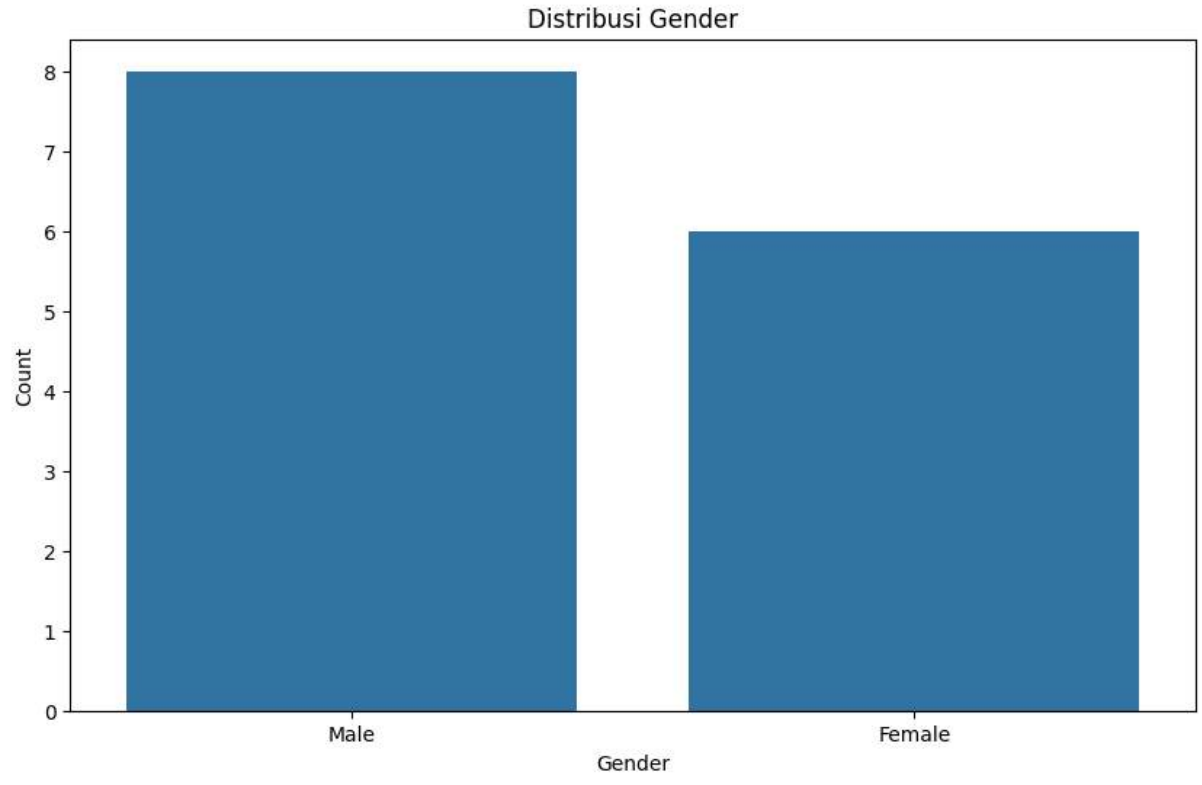
```
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print("Confusion Matrix:")
print(cm)
```

Output:

```
Original DataFrame:
   Age  Income  Gender Purchased
0   25   50000    Male        No
1   30   80000  Female       Yes
2   35   60000    Male        No
3   40   70000  Female       Yes
4  100  200000  Female       Yes

Statistical summary of the dataset:
              Age         Income
count   14.000000      14.000000
mean    50.000000   95000.000000
std     34.250211   69337.524528
min     25.000000   50000.000000
25%     31.250000   62500.000000
50%     40.000000   75000.000000
75%     48.750000   80000.000000
max    150.000000  300000.000000
```








```
Model Performance:
Accuracy: 0.6666666666666666
Precision: 0.5
Recall: 1.0
F1 Score: 0.6666666666666666
Confusion Matrix:
[[1 1]
 [0 1]]
```

```
<ipython-input-8-f521774bd6c7>:54: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to
perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].mean(), inplace=True)
<ipython-input-8-f521774bd6c7>:55: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to
perform the operation inplace on the original object.


  df['Income'].fillna(df['Income'].mean(), inplace=True)
```

9.6. Pengukuran Dataset

Pengukuran dataset adalah proses penting dalam memahami karakteristik dan kualitas data yang kita miliki. Ini melibatkan berbagai langkah untuk mengevaluasi dan memahami struktur, distribusi, serta hubungan antar variabel dalam dataset. Berikut ini adalah langkah-langkah yang biasanya dilakukan dalam pengukuran dataset:

a. Memahami Struktur Data

Langkah pertama adalah memahami struktur dataset, termasuk jumlah baris dan kolom, tipe data dari setiap kolom, dan melihat beberapa baris pertama untuk mendapatkan gambaran umum.

b. Statistik Deskriptif

Statistik deskriptif memberikan ringkasan kuantitatif tentang data. Ini termasuk mean, median, standar deviasi, minimum, maksimum, dan kuartil. Statistik deskriptif membantu dalam memahami distribusi data.

c. Visualisasi Data

Visualisasi data adalah cara efektif untuk melihat distribusi, pola, dan hubungan dalam data. Beberapa visualisasi umum termasuk histogram, box plot, scatter plot, dan heatmap.

d. Identifikasi Missing Values

Mengecek missing values penting untuk memahami kualitas data. Ini termasuk menghitung jumlah missing values di setiap kolom.

e. Identifikasi Outliers

Outliers dapat mempengaruhi analisis data dan model machine learning. Mengidentifikasi outliers dapat dilakukan dengan box plot atau metode statistik lainnya.

9.7. Early Stopping

Early stopping adalah teknik regulasi yang digunakan untuk mencegah overfitting dalam training model machine learning, terutama dalam neural networks dan gradient boosting models. Teknik ini memonitor performa model pada validation set selama training dan menghentikan training ketika performa pada validation set mulai menurun, meskipun performa pada training set masih meningkat. Tujuan utama dari early stopping adalah untuk menemukan titik optimal di mana model tidak lagi meningkatkan kemampuannya untuk generalisasi ke data baru, tetapi malah mulai overfit terhadap data training. Dengan menghentikan training pada titik ini, model diharapkan akan memiliki performa terbaik ketika diaplikasikan ke data baru yang tidak terlihat sebelumnya. Early stopping dapat diimplementasikan dengan berbagai framework machine learning seperti TensorFlow/Keras dan Scikit-learn. Berikut adalah contoh implementasi early stopping dengan Keras dan Scikit-learn.

```
# Contoh Implementasi dengan Keras
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping

# Membuat data sampel
X_train = np.random.rand(1000, 20)
y_train = np.random.randint(2, size=(1000, 1))
X_val = np.random.rand(200, 20)
y_val = np.random.randint(2, size=(200, 1))

# Membangun model sederhana
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Kompilasi model
model.compile(optimizer='adam',                    loss='binary_crossentropy',
metrics=['accuracy'])

# Menggunakan early stopping
early_stopping       =       EarlyStopping(monitor='val_loss',       patience=5,
restore_best_weights=True)

# Melatih model
history    =    model.fit(X_train,    y_train,    epochs=100,    batch_size=32,
validation_data=(X_val, y_val), callbacks=[early_stopping])

# Menampilkan ringkasan model
print(model.summary())
```

Output:

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the
model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - accuracy: 0.5070 - loss:
0.6966 - val_accuracy: 0.5000 - val_loss: 0.7071
Epoch 2/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.5382 - loss:
0.6888 - val_accuracy: 0.4350 - val_loss: 0.7052
Epoch 3/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5397 - loss:
0.6875 - val_accuracy: 0.4400 - val_loss: 0.7072
Epoch 4/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.5358 - loss:
0.6852 - val_accuracy: 0.4800 - val_loss: 0.7166
Epoch 5/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5851 - loss:
0.6746 - val_accuracy: 0.4450 - val_loss: 0.7069
Epoch 6/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5638 - loss:
0.6830 - val_accuracy: 0.4550 - val_loss: 0.7147
Epoch 7/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5622 - loss:
0.6798 - val_accuracy: 0.4600 - val_loss: 0.7143
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param
# ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense (Dense)                        │ (None, 64)                  │         1,344
│
├──────────────────────────────────────┼─────────────────────────────┼───────────────
──┤
│ dense_1 (Dense)                      │ (None, 64)                  │         4,160
│
├──────────────────────────────────────┼─────────────────────────────┼───────────────
──┤
│ dense_2 (Dense)                      │ (None, 1)                   │            65
│
└──────────────────────────────────────┴─────────────────────────────┴───────────────
──┘
 Total params: 16,709 (65.27 KB)
 Trainable params: 5,569 (21.75 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 11,140 (43.52 KB)
None
```

```
# Contoh Implementasi dengan Scikit-learn
import numpy as np
from sklearn.datasets import make_classification
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score

# Membuat data sampel
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15,
n_redundant=5, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

# Membangun model dengan early stopping
model = GradientBoostingClassifier(n_estimators=1000, learning_rate=0.01,
validation_fraction=0.2, n_iter_no_change=10)

# Melatih model
model.fit(X_train, y_train)

# Prediksi dan evaluasi
y_pred = model.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)
print(f'Validation Accuracy: {accuracy:.4f}')
```

Output:

```
Validation Accuracy: 0.9000
```

Early stopping adalah teknik regulasi yang efektif untuk mencegah overfitting dalam training model machine learning. Dengan menghentikan training pada saat yang tepat, model dapat mencapai generalisasi yang lebih baik pada data baru. Implementasinya mudah dilakukan dengan menggunakan framework seperti Keras dan Scikit-learn.

9.8. Overfitting

Overfitting adalah fenomena dalam machine learning di mana model belajar terlalu banyak detail dan noise dari data training, sehingga performanya sangat baik pada data training tetapi buruk pada data baru atau data uji (test data). Model overfit memiliki kompleksitas tinggi dan cenderung mengikuti data training terlalu ketat.

Penyebab Overfitting

- Model Terlalu Kompleks: Model dengan terlalu banyak parameter atau layer cenderung overfit, terutama jika datanya tidak cukup besar untuk mendukung kompleksitas tersebut.
- Data Terbatas: Ketika data training tidak cukup banyak atau tidak cukup bervariasi, model dapat menangkap noise dan outliers sebagai pola penting.
- Training Terlalu Lama: Melatih model terlalu lama tanpa mekanisme regulasi dapat menyebabkan model overfit karena ia akan terus mengoptimalkan dirinya untuk data training.

- Fitur Tidak Relevan: Menggunakan terlalu banyak fitur yang tidak relevan atau sangat berkorelasi dapat menyebabkan overfitting.

Dampak Overfitting

- Generalization Error: Model tidak mampu generalisasi dengan baik ke data baru, sehingga performa di dunia nyata menjadi buruk.
- Kurang Robust: Model menjadi tidak stabil dan sensitif terhadap data baru atau sedikit perubahan dalam data.

Cara Mengatasi Overfitting

- Regularisasi: Menambahkan penalti terhadap kompleksitas model dalam fungsi loss, seperti L1 (Lasso) dan L2 (Ridge) regularization.
- Cross-Validation: Menggunakan teknik seperti k-fold cross-validation untuk memastikan model tidak hanya perform baik pada subset data tertentu.
- Early Stopping: Menghentikan training lebih awal jika performa pada validation set tidak meningkat lagi.
- Data Augmentation: Memperbanyak data training dengan variasi yang dihasilkan dari data yang ada, terutama dalam tugas pengenalan gambar.
- Dropout: Dalam neural networks, menggunakan teknik dropout di mana neuron dipilih secara acak untuk "drop out" selama training, yang membantu mencegah model dari terlalu bergantung pada neuron tertentu.
- Sederhanakan Model: Mengurangi jumlah parameter atau layer dalam model untuk menurunkan kompleksitas.
- Pruning: Dalam decision tree, memotong cabang yang tidak memberikan informasi penting untuk prediksi.

```
# Regularisasi dengan L2 (Ridge Regression) di Scikit-learn
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

# Membuat data sampel
X = np.random.rand(100, 5)
y = 3*X[:, 0] + 2*X[:, 1] + np.random.randn(100) * 0.1

# Split data menjadi train dan test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Melatih model dengan Ridge Regression
ridge_reg = Ridge(alpha=1.0)
ridge_reg.fit(X_train, y_train)
```

```
# Prediksi dan evaluasi
y_pred_train = ridge_reg.predict(X_train)
y_pred_test = ridge_reg.predict(X_test)

print("Train MSE:", mean_squared_error(y_train, y_pred_train))
print("Test MSE:", mean_squared_error(y_test, y_pred_test))
```

Output:

```
Train MSE: 0.027999533704899403
Test MSE: 0.042291235269870935
```

```
# Early Stopping dengan Keras
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping

# Membuat data sampel
X_train = np.random.rand(1000, 20)
y_train = np.random.randint(2, size=(1000, 1))
X_val = np.random.rand(200, 20)
y_val = np.random.randint(2, size=(200, 1))

# Membangun model sederhana
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Kompilasi model
model.compile(optimizer='adam',                    loss='binary_crossentropy',
metrics=['accuracy'])

# Menggunakan early stopping
early_stopping      =      EarlyStopping(monitor='val_loss',      patience=5,
restore_best_weights=True)

# Melatih model
history    =    model.fit(X_train,    y_train,    epochs=100,    batch_size=32,
validation_data=(X_val, y_val), callbacks=[early_stopping])

# Menampilkan ringkasan model
print(model.summary())
```

Output:

```
Epoch 1/100
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the
model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.5141 - loss: 0.6997 - val_accuracy: 0.4950 - val_loss: 0.6950
Epoch 2/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5009 - loss: 0.6926 - val_accuracy: 0.4800 - val_loss: 0.6956
Epoch 3/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5397 - loss: 0.6902 - val_accuracy: 0.5050 - val_loss: 0.6926
Epoch 4/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5749 - loss: 0.6874 - val_accuracy: 0.5300 - val_loss: 0.6938
Epoch 5/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5528 - loss: 0.6863 - val_accuracy: 0.5400 - val_loss: 0.6918
Epoch 6/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5930 - loss: 0.6803 - val_accuracy: 0.5200 - val_loss: 0.6922
Epoch 7/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5748 - loss: 0.6796 - val_accuracy: 0.4950 - val_loss: 0.6924
Epoch 8/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6284 - loss: 0.6701 - val_accuracy: 0.5300 - val_loss: 0.6923
Epoch 9/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5895 - loss: 0.6730 - val_accuracy: 0.4900 - val_loss: 0.7073
Epoch 10/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5817 - loss: 0.6681 - val_accuracy: 0.5100 - val_loss: 0.6950
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense_3 (Dense)                 │ (None, 64)             │         1,344 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_4 (Dense)                 │ (None, 64)             │         4,160 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_5 (Dense)                 │ (None, 1)              │            65 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 16,709 (65.27 KB)
 Trainable params: 5,569 (21.75 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 11,140 (43.52 KB)
None
```

```
# Droout dengan Keras
import numpy as np
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Dropout

# Membuat data sampel
X_train = np.random.rand(1000, 20)
y_train = np.random.randint(2, size=(1000, 1))
X_val = np.random.rand(200, 20)
y_val = np.random.randint(2, size=(200, 1))

# Membangun model dengan Dropout
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

# Kompilasi model
model.compile(optimizer='adam',                    loss='binary_crossentropy',
metrics=['accuracy'])

# Melatih model
history    =    model.fit(X_train,    y_train,    epochs=100,    batch_size=32,
validation_data=(X_val, y_val))

# Menampilkan ringkasan model
print(model.summary())
```

Output:

```
Epoch 1/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.4918 - loss:
0.7318 - val_accuracy: 0.5500 - val_loss: 0.6911
Epoch 2/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4773 - loss:
0.7264 - val_accuracy: 0.5150 - val_loss: 0.6922
Epoch 3/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4870 - loss:
0.7114 - val_accuracy: 0.5600 - val_loss: 0.6899
Epoch 4/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5240 - loss:
0.7050 - val_accuracy: 0.5200 - val_loss: 0.6916
Epoch 5/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4964 - loss:
0.7014 - val_accuracy: 0.5300 - val_loss: 0.6916
Epoch 6/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4988 - loss:
0.7012 - val_accuracy: 0.5400 - val_loss: 0.6918
Epoch 7/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5103 - loss:
0.7003 - val_accuracy: 0.5150 - val_loss: 0.6929
Epoch 8/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5327 - loss:
0.6930 - val_accuracy: 0.5250 - val_loss: 0.6935
Epoch 9/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5414 - loss:
0.6911 - val_accuracy: 0.5350 - val_loss: 0.6927
Epoch 10/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5253 - loss:
0.6880 - val_accuracy: 0.5500 - val_loss: 0.6933
```

```
Epoch 11/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5105 - loss:
0.6924 - val_accuracy: 0.5400 - val_loss: 0.6929
Epoch 12/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5592 - loss:
0.6949 - val_accuracy: 0.5400 - val_loss: 0.6927
Epoch 13/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5194 - loss:
0.6899 - val_accuracy: 0.4900 - val_loss: 0.6956
Epoch 14/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5509 - loss:
0.6903 - val_accuracy: 0.4800 - val_loss: 0.6954
Epoch 15/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5495 - loss:
0.6895 - val_accuracy: 0.5300 - val_loss: 0.6949
Epoch 16/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5491 - loss:
0.6891 - val_accuracy: 0.4900 - val_loss: 0.6961
Epoch 17/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5563 - loss:
0.6924 - val_accuracy: 0.5100 - val_loss: 0.6964
Epoch 18/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5610 - loss:
0.6802 - val_accuracy: 0.4800 - val_loss: 0.6982
Epoch 19/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5541 - loss:
0.6849 - val_accuracy: 0.5250 - val_loss: 0.6955
Epoch 20/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5672 - loss:
0.6874 - val_accuracy: 0.5100 - val_loss: 0.6974
Epoch 21/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5495 - loss:
0.6811 - val_accuracy: 0.5200 - val_loss: 0.6972
Epoch 22/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5331 - loss:
0.6970 - val_accuracy: 0.4950 - val_loss: 0.6978
Epoch 23/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5406 - loss:
0.6834 - val_accuracy: 0.4850 - val_loss: 0.6989
Epoch 24/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5654 - loss:
0.6861 - val_accuracy: 0.5200 - val_loss: 0.6988
Epoch 25/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5390 - loss:
0.6864 - val_accuracy: 0.5000 - val_loss: 0.6994
Epoch 26/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5230 - loss:
0.6845 - val_accuracy: 0.5250 - val_loss: 0.7003
Epoch 27/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5381 - loss:
0.6818 - val_accuracy: 0.5250 - val_loss: 0.6993
Epoch 28/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5760 - loss:
0.6758 - val_accuracy: 0.5100 - val_loss: 0.6985
Epoch 29/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5701 - loss:
0.6762 - val_accuracy: 0.5150 - val_loss: 0.7006
Epoch 30/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5506 - loss:
0.6815 - val_accuracy: 0.5050 - val_loss: 0.7018
```

```
Epoch 31/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5592 - loss:
0.6756 - val_accuracy: 0.4900 - val_loss: 0.6998
Epoch 32/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5674 - loss:
0.6778 - val_accuracy: 0.5050 - val_loss: 0.7005
Epoch 33/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5442 - loss:
0.6806 - val_accuracy: 0.5100 - val_loss: 0.7028
Epoch 34/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6090 - loss:
0.6702 - val_accuracy: 0.4750 - val_loss: 0.7027
Epoch 35/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5789 - loss:
0.6762 - val_accuracy: 0.5050 - val_loss: 0.7023
Epoch 36/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5781 - loss:
0.6810 - val_accuracy: 0.5100 - val_loss: 0.7016
Epoch 37/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5630 - loss:
0.6765 - val_accuracy: 0.5000 - val_loss: 0.7041
Epoch 38/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5740 - loss:
0.6811 - val_accuracy: 0.4750 - val_loss: 0.7100
Epoch 39/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5671 - loss:
0.6717 - val_accuracy: 0.4900 - val_loss: 0.7060
Epoch 40/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5917 - loss:
0.6693 - val_accuracy: 0.4800 - val_loss: 0.7080
Epoch 41/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5906 - loss:
0.6759 - val_accuracy: 0.4700 - val_loss: 0.7108
Epoch 42/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6058 - loss:
0.6721 - val_accuracy: 0.4900 - val_loss: 0.7097
Epoch 43/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5793 - loss:
0.6669 - val_accuracy: 0.5250 - val_loss: 0.7069
Epoch 44/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6071 - loss:
0.6656 - val_accuracy: 0.5050 - val_loss: 0.7099
Epoch 45/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6068 - loss:
0.6634 - val_accuracy: 0.5050 - val_loss: 0.7067
Epoch 46/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5643 - loss:
0.6735 - val_accuracy: 0.4950 - val_loss: 0.7088
Epoch 47/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6127 - loss:
0.6637 - val_accuracy: 0.4950 - val_loss: 0.7089
Epoch 48/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5774 - loss:
0.6746 - val_accuracy: 0.5000 - val_loss: 0.7099
Epoch 49/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5905 - loss:
0.6716 - val_accuracy: 0.4950 - val_loss: 0.7116
Epoch 50/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5972 - loss:
0.6605 - val_accuracy: 0.4700 - val_loss: 0.7127
```

```
Epoch 51/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5985 - loss:
0.6606 - val_accuracy: 0.5000 - val_loss: 0.7154
Epoch 52/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5805 - loss:
0.6620 - val_accuracy: 0.4900 - val_loss: 0.7119
Epoch 53/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6259 - loss:
0.6470 - val_accuracy: 0.4750 - val_loss: 0.7188
Epoch 54/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6134 - loss:
0.6571 - val_accuracy: 0.5000 - val_loss: 0.7170
Epoch 55/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6126 - loss:
0.6466 - val_accuracy: 0.5050 - val_loss: 0.7229
Epoch 56/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6153 - loss:
0.6510 - val_accuracy: 0.4950 - val_loss: 0.7217
Epoch 57/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5929 - loss:
0.6566 - val_accuracy: 0.4900 - val_loss: 0.7110
Epoch 58/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6125 - loss:
0.6524 - val_accuracy: 0.5200 - val_loss: 0.7196
Epoch 59/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6170 - loss:
0.6599 - val_accuracy: 0.5350 - val_loss: 0.7166
Epoch 60/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5751 - loss:
0.6603 - val_accuracy: 0.5000 - val_loss: 0.7232
Epoch 61/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6247 - loss:
0.6357 - val_accuracy: 0.4850 - val_loss: 0.7277
Epoch 62/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6303 - loss:
0.6436 - val_accuracy: 0.5300 - val_loss: 0.7237
Epoch 63/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6244 - loss:
0.6435 - val_accuracy: 0.4850 - val_loss: 0.7203
Epoch 64/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6034 - loss:
0.6414 - val_accuracy: 0.5000 - val_loss: 0.7235
Epoch 65/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6148 - loss:
0.6407 - val_accuracy: 0.4850 - val_loss: 0.7260
Epoch 66/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6467 - loss:
0.6232 - val_accuracy: 0.4900 - val_loss: 0.7367
Epoch 67/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6046 - loss:
0.6366 - val_accuracy: 0.5000 - val_loss: 0.7273
Epoch 68/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6262 - loss:
0.6348 - val_accuracy: 0.4950 - val_loss: 0.7306
Epoch 69/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6514 - loss:
0.6253 - val_accuracy: 0.4900 - val_loss: 0.7327
Epoch 70/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6434 - loss:
0.6254 - val_accuracy: 0.5100 - val_loss: 0.7324
```

```
Epoch 71/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5987 - loss:
0.6521 - val_accuracy: 0.4900 - val_loss: 0.7319
Epoch 72/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6589 - loss:
0.6247 - val_accuracy: 0.4800 - val_loss: 0.7344
Epoch 73/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6421 - loss:
0.6196 - val_accuracy: 0.4650 - val_loss: 0.7389
Epoch 74/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6350 - loss:
0.6362 - val_accuracy: 0.5050 - val_loss: 0.7435
Epoch 75/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6532 - loss:
0.6144 - val_accuracy: 0.4800 - val_loss: 0.7437
Epoch 76/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6217 - loss:
0.6240 - val_accuracy: 0.4950 - val_loss: 0.7436
Epoch 77/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6465 - loss:
0.6220 - val_accuracy: 0.4750 - val_loss: 0.7434
Epoch 78/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6563 - loss:
0.6178 - val_accuracy: 0.4700 - val_loss: 0.7473
Epoch 79/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6284 - loss:
0.6290 - val_accuracy: 0.4850 - val_loss: 0.7527
Epoch 80/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6408 - loss:
0.6186 - val_accuracy: 0.4750 - val_loss: 0.7512
Epoch 81/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6742 - loss:
0.6035 - val_accuracy: 0.5150 - val_loss: 0.7435
Epoch 82/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6461 - loss:
0.6125 - val_accuracy: 0.4950 - val_loss: 0.7435
Epoch 83/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6562 - loss:
0.6093 - val_accuracy: 0.4650 - val_loss: 0.7579
Epoch 84/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6309 - loss:
0.6154 - val_accuracy: 0.4950 - val_loss: 0.7509
Epoch 85/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6633 - loss:
0.6092 - val_accuracy: 0.4950 - val_loss: 0.7556
Epoch 86/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6676 - loss:
0.6005 - val_accuracy: 0.5000 - val_loss: 0.7579
Epoch 87/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6253 - loss:
0.6312 - val_accuracy: 0.4900 - val_loss: 0.7579
Epoch 88/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6458 - loss:
0.6131 - val_accuracy: 0.4700 - val_loss: 0.7515
Epoch 89/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6555 - loss:
0.6090 - val_accuracy: 0.4850 - val_loss: 0.7550
Epoch 90/100
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6702 - loss:
0.6077 - val_accuracy: 0.4750 - val_loss: 0.7679
```

```
Epoch 91/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6641 - loss: 0.6024 - val_accuracy: 0.4750 - val_loss: 0.7699
Epoch 92/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6460 - loss: 0.6206 - val_accuracy: 0.4600 - val_loss: 0.7569
Epoch 93/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6829 - loss: 0.5952 - val_accuracy: 0.4600 - val_loss: 0.7631
Epoch 94/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6536 - loss: 0.6138 - val_accuracy: 0.4650 - val_loss: 0.7497
Epoch 95/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6749 - loss: 0.5889 - val_accuracy: 0.4750 - val_loss: 0.7578
Epoch 96/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6946 - loss: 0.5745 - val_accuracy: 0.4700 - val_loss: 0.7695
Epoch 97/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6733 - loss: 0.5863 - val_accuracy: 0.4850 - val_loss: 0.7562
Epoch 98/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6684 - loss: 0.6009 - val_accuracy: 0.4600 - val_loss: 0.7522
Epoch 99/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6584 - loss: 0.5999 - val_accuracy: 0.4600 - val_loss: 0.7559
Epoch 100/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6613 - loss: 0.5945 - val_accuracy: 0.4950 - val_loss: 0.7695
Model: "sequential_2"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense_6 (Dense)                 │ (None, 64)             │         1,344 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_7 (Dense)                 │ (None, 64)             │         4,160 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_8 (Dense)                 │ (None, 1)              │            65 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 16,709 (65.27 KB)
 Trainable params: 5,569 (21.75 KB)
 Non-trainable params: 0 (0.00 B)
```

```
 Optimizer params: 11,140 (43.52 KB)
None
```

Overfitting adalah masalah umum dalam machine learning yang dapat mengurangi kemampuan model untuk generalisasi ke data baru. Namun, dengan teknik seperti regularisasi, cross-validation, early stopping, data augmentation, dropout, dan penyederhanaan model, kita dapat mengurangi risiko overfitting dan meningkatkan performa model pada data baru.

9.9. Ringkasan Aktivitas Machine Learning (Awal sampai dengan Deployment)

Ringkasan aktivitas machine learning mencakup seluruh tahapan yang dimulai dari pengumpulan data hingga penerapan model ke lingkungan produksi. Proses ini melibatkan berbagai langkah, seperti persiapan data, pemilihan model, pelatihan, evaluasi, optimasi, hingga deployment. Setiap tahapan memiliki peran penting dalam memastikan model machine learning dapat bekerja dengan efektif dan akurat.


5 STEPS
DEPLOYMENT MACHINE LEARNING
1
PERSIAPAN MODEL
Langkah pertama adalah memastikan model yang telah dilatih siap untuk diterapkan.
2
PEMBUATAN API
Model machine learning perlu dikemas dalam antarmuka yang memungkinkan interaksi dengan aplikasi lain.
3
PENGATURAN INFRASTRUKTUR
Pilih dan konfigurasikan infrastruktur untuk menjalankan model. Ini bisa berupa server fisik, mesin virtual di cloud, atau layanan container.
4
PENGUJIAN DAN VALIDASI
Sebelum meluncurkan model, lakukan pengujian menyeluruh untuk memastikan model berfungsi baik.
5
PEMANTAUAN DAN PEMELIHARAAN
Model harus dipantau kinerjanya secara berkala untuk memastikan bahwa model tetap akurat dan efisien.

Bab 10. Evaluasi Efektivitas Pendekatan Machine Learning

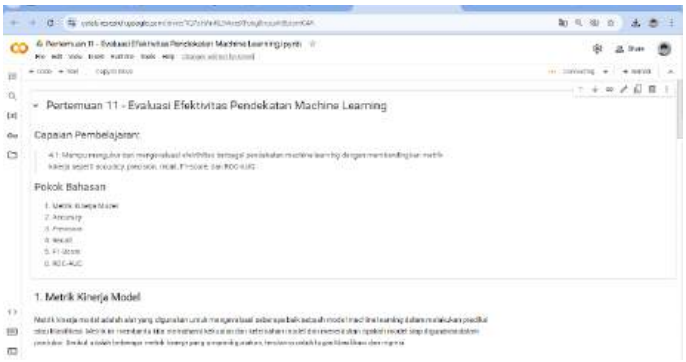
Hal-hal yang dibahas pada Bab ini:

1. Pengenalan metrik kinerja model.
2. Pengukuran accuracy.
3. Evaluasi precision dan recall.
4. Perhitungan F1-score.
5. Analisis ROC-AUC.
6. Perbandingan efektivitas model.

10.1. Metrik Kinerja Model

Metrik kinerja model adalah alat yang digunakan untuk mengevaluasi seberapa baik sebuah model machine learning dalam melakukan prediksi atau klasifikasi. Metrik ini membantu kita memahami kekuatan dan kelemahan model dan menentukan apakah model siap digunakan dalam produksi. Berikut adalah beberapa metrik kinerja yang umum digunakan, terutama untuk tugas klasifikasi dan regresi.

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



10.2. Accuracy

Akurasi adalah persentase prediksi yang benar dari keseluruhan prediksi. Akurasi sering digunakan sebagai metrik awal, tetapi mungkin tidak cukup jika dataset tidak seimbang.

10.3. Precision

Presisi adalah rasio prediksi positif yang benar terhadap semua prediksi positif. Ini menunjukkan seberapa banyak prediksi positif yang benar-benar relevan.

10.4. Recall

Recall adalah rasio prediksi positif yang benar terhadap semua sampel yang sebenarnya positif. Ini menunjukkan seberapa baik model dalam menemukan semua sampel yang relevan.

10.5. F1-Score

F1 Score adalah rata-rata harmonis dari presisi dan recall. Ini memberikan keseimbangan antara presisi dan recall.

10.6. ROC-AUC

AUC-ROC adalah metrik yang menunjukkan kinerja model klasifikasi pada berbagai threshold. ROC adalah kurva yang menunjukkan trade-off antara true positive rate (recall) dan false positive rate. AUC adalah area di bawah kurva ini, dengan nilai maksimum 1 menunjukkan kinerja sempurna.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Membuat data sampel untuk klasifikasi
X_class = np.random.rand(1000, 10)
y_class = np.random.randint(2, size=1000)

# Split data menjadi train dan test
X_train_class, X_test_class, y_train_class, y_test_class =
train_test_split(X_class, y_class, test_size=0.2, random_state=42)

# Melatih model klasifikasi
model_class = LogisticRegression()
model_class.fit(X_train_class, y_train_class)

# Prediksi
y_pred_class = model_class.predict(X_test_class)
y_prob_class = model_class.predict_proba(X_test_class)[:, 1]

# Metrik Klasifikasi
print("Accuracy:", accuracy_score(y_test_class, y_pred_class))
print("Precision:", precision_score(y_test_class, y_pred_class))
print("Recall:", recall_score(y_test_class, y_pred_class))
print("F1 Score:", f1_score(y_test_class, y_pred_class))
print("AUC-ROC:", roc_auc_score(y_test_class, y_prob_class))

# Membuat data sampel untuk regresi
X_reg = np.random.rand(1000, 10)
y_reg = 3 * X_reg[:, 0] + 2 * X_reg[:, 1] + np.random.randn(1000) * 0.1

# Split data menjadi train dan test
```

```
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg,
test_size=0.2, random_state=42)

# Melatih model regresi
model_reg = LinearRegression()
model_reg.fit(X_train_reg, y_train_reg)

# Prediksi
y_pred_reg = model_reg.predict(X_test_reg)

# Metrik Regresi
print("Mean Absolute Error:", mean_absolute_error(y_test_reg, y_pred_reg))
print("Mean Squared Error:", mean_squared_error(y_test_reg, y_pred_reg))
print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test_reg,
y_pred_reg)))
```

Output:

```
Accuracy: 0.52
Precision: 0.5
Recall: 0.5729166666666666
F1 Score: 0.5339805825242718
AUC-ROC: 0.530448717948718
```

10.7. Cross Validation

Video ini akan membahas tentang konsep Cross Validation dan pentingnya teknik ini dalam machine learning. Cross Validation adalah metode evaluasi yang membagi dataset menjadi beberapa bagian (subset) untuk memastikan bahwa model diuji secara menyeluruh pada data yang beragam. Salah satu pendekatan yang sering digunakan adalah k-fold cross validation, di mana data dipecah menjadi k bagian. Model dilatih dan diuji sebanyak k kali, dengan setiap bagian secara bergantian menjadi data uji. Pendekatan ini membantu mengurangi risiko overfitting dan memberikan gambaran yang lebih akurat tentang kinerja model pada data baru, sehingga model yang dihasilkan lebih dapat diandalkan dalam penerapannya.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Metrik kinerja model sangat penting untuk mengevaluasi dan memahami performa model machine learning. Metrik-metrik seperti akurasi, presisi, recall, F1 score, AUC-ROC untuk klasifikasi, serta MAE, MSE, dan RMSE untuk regresi,

memberikan pandangan yang komprehensif tentang seberapa baik model melakukan tugasnya dan membantu dalam perbaikan dan optimisasi model.

Bab 11. Pengujian dan Evaluasi Kinerja Model

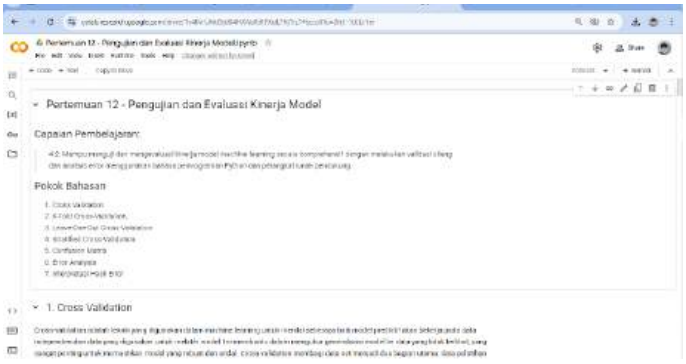
Hal-hal yang dibahas pada Bab ini:

1. Pengenalan validasi silang (cross-validation).
2. Implementasi validasi silang dalam Python.
3. Teknik analisis error.
4. Penggunaan perangkat lunak pendukung untuk evaluasi model.
5. Interpretasi hasil validasi dan analisis error.
6.

11.1. Cross Validation

Cross-validation adalah teknik yang digunakan dalam machine learning untuk menilai seberapa baik model prediktif akan bekerja pada data independen dari data yang digunakan untuk melatih model. Ini membantu dalam mengukur generalisasi model ke data yang tidak terlihat, yang sangat penting untuk memastikan model yang robust dan andal. Cross-validation membagi data set menjadi dua bagian utama: data pelatihan (training) dan data pengujian (testing). Dalam praktik, data sering kali dibagi menjadi beberapa subset (folds), dan proses pelatihan serta pengujian dilakukan beberapa kali dengan subset yang berbeda.

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



A. Jenis-Jenis Cross Validation

a. K-Fold Cross-Validation

- Proses: Data dibagi menjadi k subset (atau folds) yang hampir sama besar. Model dilatih k kali, setiap kali menggunakan k-1 subset sebagai data pelatihan dan 1 subset sebagai data pengujian.

- Keunggulan: Memastikan setiap data point diuji dan digunakan dalam pelatihan.
- Kelemahan: Memerlukan komputasi yang lebih besar dibandingkan metode lain.

b. Stratified K-Fold Cross-Validation

- Proses: Mirip dengan K-Fold, tetapi memastikan setiap fold memiliki distribusi yang sama dari kelas yang berbeda.
- Keunggulan: Sangat berguna untuk dataset yang tidak seimbang, menjaga proporsi kelas dalam setiap fold.

c. Leave-One-Out Cross-Validation (LOOCV)

- Proses: Setiap data point digunakan sekali sebagai data pengujian dan sisanya sebagai data pelatihan.
- Keunggulan: Menggunakan seluruh data untuk pelatihan dan pengujian.
- Kelemahan: Sangat mahal dalam komputasi untuk dataset besar.

d. Hold-Out Method

- Proses: Data dibagi sekali menjadi dua set, satu untuk pelatihan dan satu untuk pengujian.
- Keunggulan: Cepat dan sederhana.
- Kelemahan: Dapat menghasilkan hasil yang bervariasi tergantung pada bagaimana data dibagi.

e. Time Series Cross-Validation

- Proses: Data dibagi berdasarkan urutan waktu. Data awal digunakan untuk pelatihan dan data berikutnya untuk pengujian.
- Keunggulan: Cocok untuk data time series.
- Kelemahan: Tidak dapat diacak seperti metode lain.

B. Manfaat Cross-Validation

- Akurasi Model: Memberikan estimasi yang lebih akurat tentang kinerja model pada data tidak terlihat.
- Bias dan Variance: Membantu mengidentifikasi masalah bias tinggi (underfitting) atau variance tinggi (overfitting).
- Model Selection: Membantu dalam pemilihan model terbaik dari beberapa kandidat.

C. Kelemahan

- Komputasi Intensif: Memerlukan komputasi yang lebih besar, terutama pada dataset besar atau model kompleks.
- Tidak Menangani dengan Baik Data Time Series: Metode standar cross-validation tidak cocok untuk data time series yang memerlukan pemesanan temporal.

Implementasi

Dalam implementasi, kita menggunakan library seperti scikit-learn di Python yang menyediakan fungsi-fungsi untuk melakukan cross-validation dengan mudah.

```
from sklearn.model_selection import cross_val_score, KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Define model
model = RandomForestClassifier()

# Define cross-validation method
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Perform cross-validation
scores = cross_val_score(model, X, y, cv=kf)

print("Cross-Validation Scores:", scores)
print("Mean Score:", scores.mean())
```

Output:

```
Cross-Validation Scores: [1.         0.96666667 0.93333333 0.93333333 0.96666667]
Mean Score: 0.9600000000000002
```

Cross-validation adalah teknik yang penting dan luas digunakan dalam machine learning untuk mengukur kinerja model secara lebih akurat. Memahami dan menerapkan cross-validation dengan benar dapat sangat meningkatkan keandalan dan generalisasi model, yang pada akhirnya menghasilkan prediksi yang lebih baik pada data tidak terlihat.

11.2. K-Fold Cross-Validation

K-Fold Cross-Validation adalah salah satu teknik cross-validation yang paling populer dan sering digunakan dalam machine learning. Teknik ini membagi dataset menjadi k subset atau "folds" yang hampir sama besar, dan proses pelatihan serta

pengujian dilakukan sebanyak k kali, setiap kali menggunakan k-1 subset sebagai data pelatihan dan 1 subset sebagai data pengujian.

Langkah-Langkah K-Fold Cross-Validation

1. Pembagian Data: Dataset dibagi menjadi k subset yang hampir sama besar. Misalnya, jika k = 5, maka dataset dibagi menjadi 5 bagian.
2. Proses Pelatihan dan Pengujian:

- Pada iterasi pertama, fold pertama digunakan sebagai data pengujian, dan fold kedua hingga kelima digunakan sebagai data pelatihan.
- Pada iterasi kedua, fold kedua digunakan sebagai data pengujian, dan fold pertama, serta fold ketiga hingga kelima digunakan sebagai data pelatihan.
- Proses ini diulang hingga setiap fold telah digunakan sekali sebagai data pengujian.

3. Evaluasi Model: Pada setiap iterasi, model dilatih menggunakan data pelatihan dan dievaluasi menggunakan data pengujian. Hasil evaluasi dari setiap iterasi dicatat.
4. Perhitungan Skor Rata-Rata: Setelah semua iterasi selesai, hasil evaluasi (misalnya akurasi, presisi, recall, dll.) dari setiap iterasi dirata-ratakan untuk memberikan estimasi kinerja model.

Contoh Implementasi K-Fold Cross-Validation

Berikut adalah contoh implementasi K-Fold Cross-Validation menggunakan scikit-learn di Python:

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Define model
model = RandomForestClassifier()

# Define K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Perform K-Fold Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

print("Cross-Validation Scores:", scores)
print("Mean Score:", scores.mean())
```

Output:

```
Cross-Validation Scores: [1.         0.96666667 0.93333333 0.93333333 0.96666667]
Mean Score: 0.9600000000000002
```

Keunggulan K-Fold Cross-Validation

- Menggunakan Seluruh Data: Setiap data point digunakan sebagai bagian dari data pelatihan dan data pengujian, sehingga memaksimalkan penggunaan dataset.
- Mengurangi Variansi: Hasil evaluasi yang diperoleh lebih stabil karena dilakukan pada beberapa subset yang berbeda.
- Generalizability: Memberikan estimasi yang lebih baik tentang kinerja model pada data yang tidak terlihat dibandingkan metode hold-out sederhana.

Kelemahan K-Fold Cross-Validation

- Komputasi yang Lebih Besar: Memerlukan pelatihan model k kali, yang bisa menjadi mahal dari segi komputasi terutama untuk dataset yang besar dan model yang kompleks.
- Tidak Cocok untuk Data Time Series: K-Fold Cross-Validation tidak cocok untuk data time series yang memerlukan urutan temporal tertentu.

Variasi K-Fold Cross-Validation

- Stratified K-Fold: Memastikan bahwa setiap fold memiliki distribusi kelas yang sama dengan distribusi kelas asli. Ini sangat berguna untuk dataset dengan kelas yang tidak seimbang.
- Repeated K-Fold: Mengulang proses K-Fold Cross-Validation beberapa kali dengan pembagian data yang berbeda pada setiap pengulangan untuk mendapatkan estimasi kinerja yang lebih akurat.

11.3. Leave-One-Out Cross-Validation

Leave-One-Out Cross-Validation (LOOCV) adalah variasi dari K-Fold Cross-Validation yang sangat ekstrem di mana jumlah fold sama dengan jumlah data point dalam dataset. Dalam LOOCV, setiap data point digunakan sekali sebagai data pengujian dan semua data point lainnya digunakan sebagai data pelatihan. Ini berarti bahwa jika Anda memiliki n data point, model akan dilatih n kali.

Langkah-Langkah LOOCV

1. Pembagian Data: Dataset dibagi menjadi n subset, di mana n adalah jumlah total data point dalam dataset. Setiap subset hanya terdiri dari satu data point.

2. Oroses Pelatihan dan Pengujian:

- Pada setiap iterasi, satu data point dipisahkan sebagai data pengujian, dan data point yang tersisa digunakan sebagai data pelatihan.
- Model dilatih menggunakan data pelatihan dan dievaluasi menggunakan data pengujian.
- Proses ini diulang hingga setiap data point telah digunakan sekali sebagai data pengujian.

3. Evaluasi Model: Hasil evaluasi (misalnya akurasi, kesalahan, dll.) dari setiap iterasi dicatat.
4. Perhitungan Skor Rata-Rata: Setelah semua iterasi selesai, hasil evaluasi dari setiap iterasi dirata-ratakan untuk memberikan estimasi kinerja model.

```
#### Contoh Implementasi LOOCV
##Berikut adalah contoh implementasi LOOCV menggunakan scikit-learn di Python:
from sklearn.model_selection import LeaveOneOut, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Define model
model = RandomForestClassifier()

# Define Leave-One-Out Cross-Validation
loo = LeaveOneOut()

# Perform LOOCV
scores = cross_val_score(model, X, y, cv=loo)

print("Cross-Validation Scores:", scores)
print("Mean Score:", scores.mean())
```

Output:

```
Cross-Validation Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1.]
Mean Score: 0.96
```

Keunggulan LOOCV

- Penggunaan Maksimal Data: Setiap data point digunakan untuk pelatihan dan pengujian, sehingga memaksimalkan penggunaan dataset.

- Tidak Ada Pembagian Data Acak: Menghilangkan variabilitas yang berasal dari pembagian data acak, memberikan hasil yang sangat deterministik.

Kelemahan LOOCV

- Komputasi yang Sangat Intensif: Memerlukan pelatihan model sebanyak n kali, yang bisa sangat mahal dari segi komputasi terutama untuk dataset yang besar.
- High Variance: Hasil dari LOOCV dapat memiliki variansi yang tinggi karena evaluasi dilakukan hanya pada satu data point setiap kali, yang dapat menyebabkan hasil yang sangat bervariasi jika data point tersebut adalah outlier.

Penggunaan LOOCV dalam Praktik

LOOCV sering digunakan dalam situasi di mana dataset sangat kecil, dan penting untuk memaksimalkan penggunaan setiap data point. Namun, untuk dataset yang lebih besar, K-Fold Cross-Validation (dengan k yang lebih kecil dari n) lebih sering digunakan karena mengurangi beban komputasi sambil tetap memberikan estimasi kinerja yang baik. Dengan LOOCV, kita mendapatkan estimasi yang sangat akurat tentang bagaimana model akan berperforma pada data yang belum pernah dilihat, tetapi dengan biaya komputasi yang sangat tinggi. Oleh karena itu, pemilihan antara LOOCV dan metode cross-validation lainnya harus mempertimbangkan ukuran dataset dan sumber daya komputasi yang tersedia.

11.4. Stratified Cross-Validation

Stratified Cross-Validation adalah variasi dari K-Fold Cross-Validation yang memastikan bahwa setiap fold atau subset memiliki distribusi kelas yang serupa dengan distribusi kelas pada dataset asli. Hal ini sangat berguna untuk dataset dengan kelas yang tidak seimbang, di mana jumlah data dalam satu kelas jauh lebih sedikit dibandingkan kelas lainnya. Dengan menggunakan Stratified Cross-Validation, kita bisa mendapatkan evaluasi kinerja model yang lebih representatif.

Langkah-Langkah Stratified Cross-Validation

1. Pembagian Data: Dataset dibagi menjadi k subset (folds) seperti pada K-Fold Cross-Validation, tetapi pembagian ini dilakukan dengan menjaga distribusi kelas dalam setiap subset tetap serupa dengan distribusi kelas pada dataset asli.
2. Proses Pelatihan dan Pengujian:

- Pada iterasi pertama, fold pertama digunakan sebagai data pengujian, dan fold kedua hingga k digunakan sebagai data pelatihan.
- Pada iterasi kedua, fold kedua digunakan sebagai data pengujian, dan fold pertama serta fold ketiga hingga k digunakan sebagai data pelatihan.
- Proses ini diulang hingga setiap fold telah digunakan sekali sebagai data pengujian.

3. Evaluasi Model: Pada setiap iterasi, model dilatih menggunakan data pelatihan dan dievaluasi menggunakan data pengujian. Hasil evaluasi dari setiap iterasi dicatat.
4. Perhitungan Skor Rata-Rata: Setelah semua iterasi selesai, hasil evaluasi dari setiap iterasi dirata-ratakan untuk memberikan estimasi kinerja model.

Contoh Implementasi Stratified Cross-Validation

Berikut adalah contoh implementasi Stratified Cross-Validation menggunakan scikit-learn di Python:

```
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Define model
model = RandomForestClassifier()

# Define Stratified K-Fold Cross-Validation
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform Stratified K-Fold Cross-Validation
scores = cross_val_score(model, X, y, cv=skf)

print("Stratified Cross-Validation Scores:", scores)
print("Mean Score:", scores.mean())
```

Output:


```
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Define model
model = RandomForestClassifier()

# Define Stratified K-Fold Cross-Validation
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform Stratified K-Fold Cross-Validation
scores = cross_val_score(model, X, y, cv=skf)

print("Stratified Cross-Validation Scores:", scores)
print("Mean Score:", scores.mean())
```

Keunggulan Stratified Cross-Validation

- Distribusi Kelas yang Seimbang: Menjaga proporsi kelas yang sama di setiap fold, yang sangat penting untuk dataset dengan kelas yang tidak seimbang.
- Estimasi Kinerja yang Akurat: Memberikan estimasi kinerja model yang lebih representatif karena distribusi kelas yang terjaga.

Kelemahan Stratified Cross-Validation

- Kompleksitas Tambahan: Memerlukan langkah tambahan untuk memastikan bahwa setiap fold memiliki distribusi kelas yang sama, yang bisa lebih rumit dibandingkan K-Fold Cross-Validation biasa.

Penggunaan dalam Praktik

Stratified Cross-Validation sangat berguna dalam situasi di mana dataset memiliki kelas yang tidak seimbang, seperti dalam masalah klasifikasi di mana satu kelas jauh lebih banyak atau lebih sedikit daripada kelas lainnya. Dengan memastikan bahwa setiap fold memiliki distribusi kelas yang sama, kita dapat menghindari bias dalam evaluasi kinerja model dan mendapatkan gambaran yang lebih akurat tentang bagaimana model akan berperforma pada data yang tidak terlihat. Dalam prakteknya, Stratified Cross-Validation sering digunakan dalam klasifikasi biner dan multi-kelas, serta dalam berbagai domain seperti kedokteran, keuangan, dan ilmu sosial di mana dataset sering kali tidak seimbang.

11.5. Confusion Matrix

Confusion Matrix adalah alat evaluasi yang digunakan dalam klasifikasi machine learning untuk memahami kinerja model. Matrix ini menyajikan hasil klasifikasi dalam bentuk tabel yang menunjukkan jumlah prediksi benar dan salah untuk setiap kelas. Confusion Matrix memberikan gambaran lengkap tentang bagaimana model melakukan prediksi dan di mana model membuat kesalahan.

Struktur Confusion Matrix

Confusion Matrix untuk masalah klasifikasi biner terdiri dari empat elemen utama:

1. True Positive (TP): Jumlah data yang benar-benar positif dan diprediksi sebagai positif oleh model.
2. True Negative (TN): Jumlah data yang benar-benar negatif dan diprediksi sebagai negatif oleh model.
3. False Positive (FP): Jumlah data yang benar-benar negatif tetapi diprediksi sebagai positif oleh model (juga dikenal sebagai Type I Error).
4. False Negative (FN): Jumlah data yang benar-benar positif tetapi diprediksi sebagai negatif oleh model (juga dikenal sebagai Type II Error).

Contoh Implementasi Confusion Matrix di Python

Berikut adalah contoh menghitung Confusion Matrix dan metrik terkait menggunakan scikit-learn di Python:

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Binary classification (taking only two classes for simplicity)
X = X[y != 2]
y = y[y != 2]

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define model
model = RandomForestClassifier()

# Train model
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

Output:

```
Confusion Matrix:
 [[17  0]
 [ 0 13]]
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-Score: 1.0
```

Interpretasi Confussion Matrix

Confusion Matrix dan metrik terkait memberikan gambaran yang komprehensif tentang kinerja model, menunjukkan tidak hanya seberapa baik model memprediksi secara keseluruhan tetapi juga bagaimana model menangani berbagai jenis kesalahan.

11.6. Error Analysis

Error analysis dalam machine learning adalah proses sistematis untuk mengidentifikasi dan memahami kesalahan yang dilakukan oleh model prediksi. Ini membantu dalam mengevaluasi kinerja model secara mendalam dan mengidentifikasi area untuk perbaikan. Berikut ini adalah beberapa langkah dan teknik yang umum digunakan dalam error analysis:

a. Menganalisis Confusion Matrix

Confusion Matrix menyediakan cara yang komprehensif untuk melihat bagaimana model mengelompokkan prediksi ke dalam kategori benar dan salah. Dengan melihat nilai TP, TN, FP, dan FN, kita bisa memahami pola kesalahan.

b. Menghitung Metrik Evaluasi

Selain metrik umum seperti akurasi, precision, recall, dan F1-score, penting untuk melihat metrik lainnya tergantung pada konteks dan tujuan model.

c. Visualisasi Kesalahan

Visualisasi kesalahan bisa membantu dalam memahami distribusi kesalahan dan mengidentifikasi pola yang tidak terdeteksi hanya dengan melihat metrik numerik.

- ROC Curve: Menunjukkan trade-off antara true positive rate dan false positive rate
- Precision-Recall Curve: Berguna untuk dataset dengan kelas tidak seimbang.
- Error Distribution Plots: Menunjukkan distribusi kesalahan prediksi.

d. Analisis Kesalahan Per Kelas

Mengidentifikasi kelas mana yang paling sering salah diprediksi bisa membantu dalam memahami ketidakseimbangan kelas atau kesulitan spesifik dalam membedakan kelas tertentu.

e. Analisis Kesalahan Berdasarkan Fitur

Melihat kesalahan berdasarkan fitur atau subset dari data bisa membantu dalam menemukan pola atau karakteristik spesifik yang menyebabkan kesalahan.

f. Menggunakan Teknik Interpretasi Model

Teknik seperti SHAP (SHapley Additive exPlanations) dan LIME (Local Interpretable Model-agnostic Explanations) dapat membantu dalam memahami mengapa model membuat prediksi tertentu dan di mana model mungkin salah.

g. Cross-Validation dan Train/Test Split

- Cross-Validation: Membantu dalam memastikan bahwa hasil error analysis tidak terlalu bergantung pada satu split dari data.
- Train/Test Split: Membandingkan kinerja model pada set pelatihan dan pengujian bisa menunjukkan apakah model overfitting atau underfitting.

h. Analisis Residual

Dalam masalah regresi, menganalisis residual (selisih antara nilai yang diprediksi dan nilai aktual) bisa menunjukkan pola yang tidak diprediksi dengan baik oleh model.

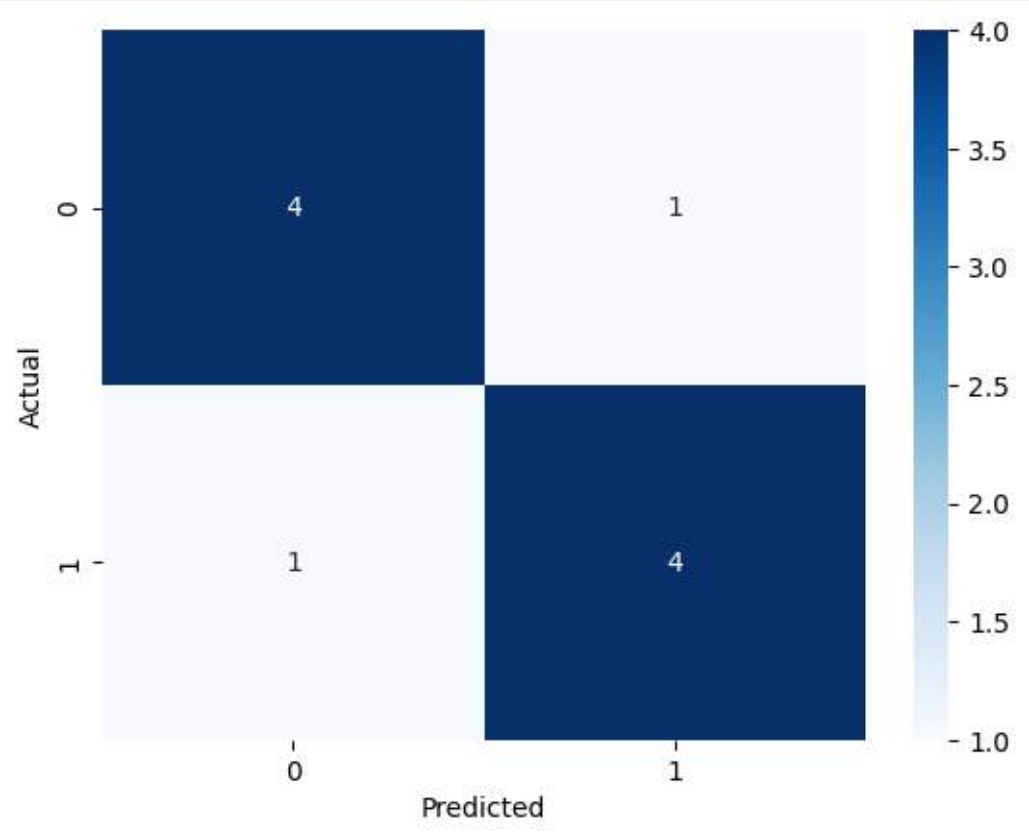
Contoh Implementasi Error Analysis

Berikut adalah contoh implementasi dasar error analysis menggunakan confusion matrix dan beberapa metrik evaluasi di Python:

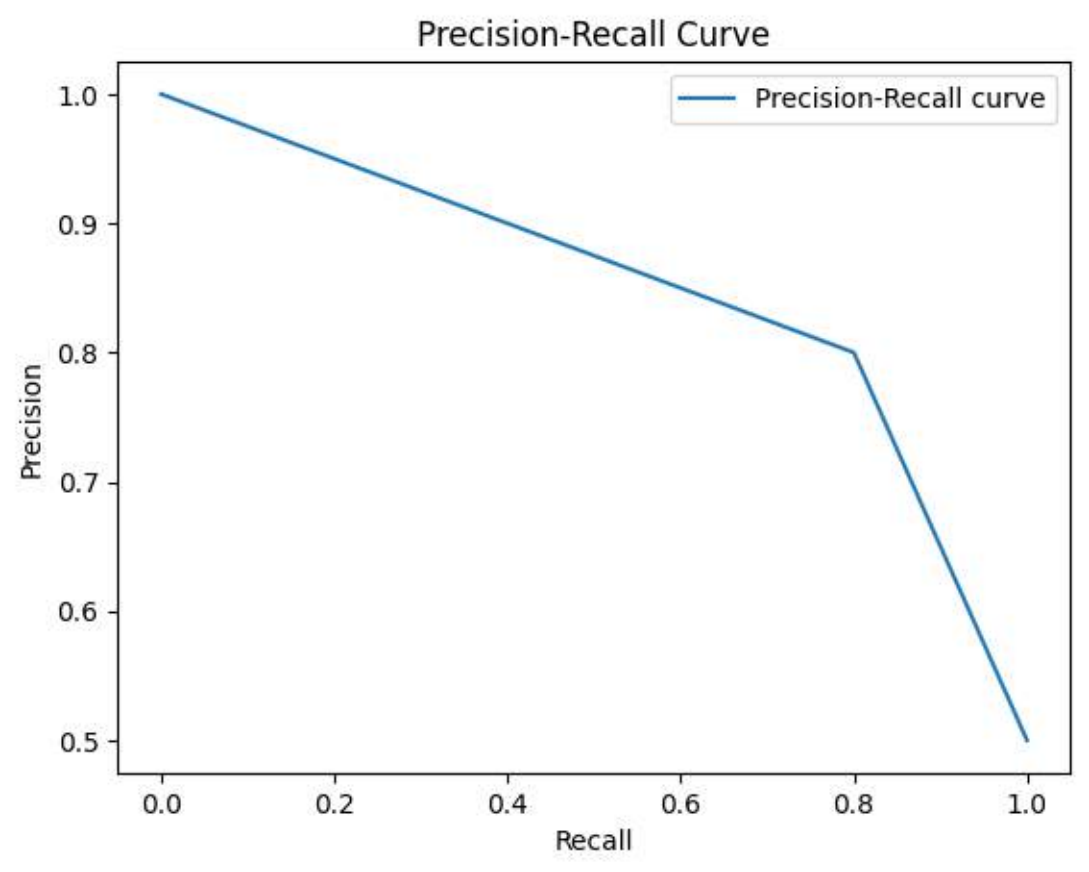
```
from sklearn.metrics import confusion_matrix, classification_report,
roc_auc_score, roc_curve, precision_recall_curve
import matplotlib.pyplot as plt
import seaborn as sns

# Misalkan kita sudah memiliki y_test dan y_pred
y_test = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
y_pred = [0, 1, 0, 0, 0, 1, 1, 1, 0, 1]

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Classification Report
print(classification_report(y_test, y_pred))

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr, label='ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()

# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(y_test, y_pred)
plt.plot(recall, precision, label='Precision-Recall curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
```

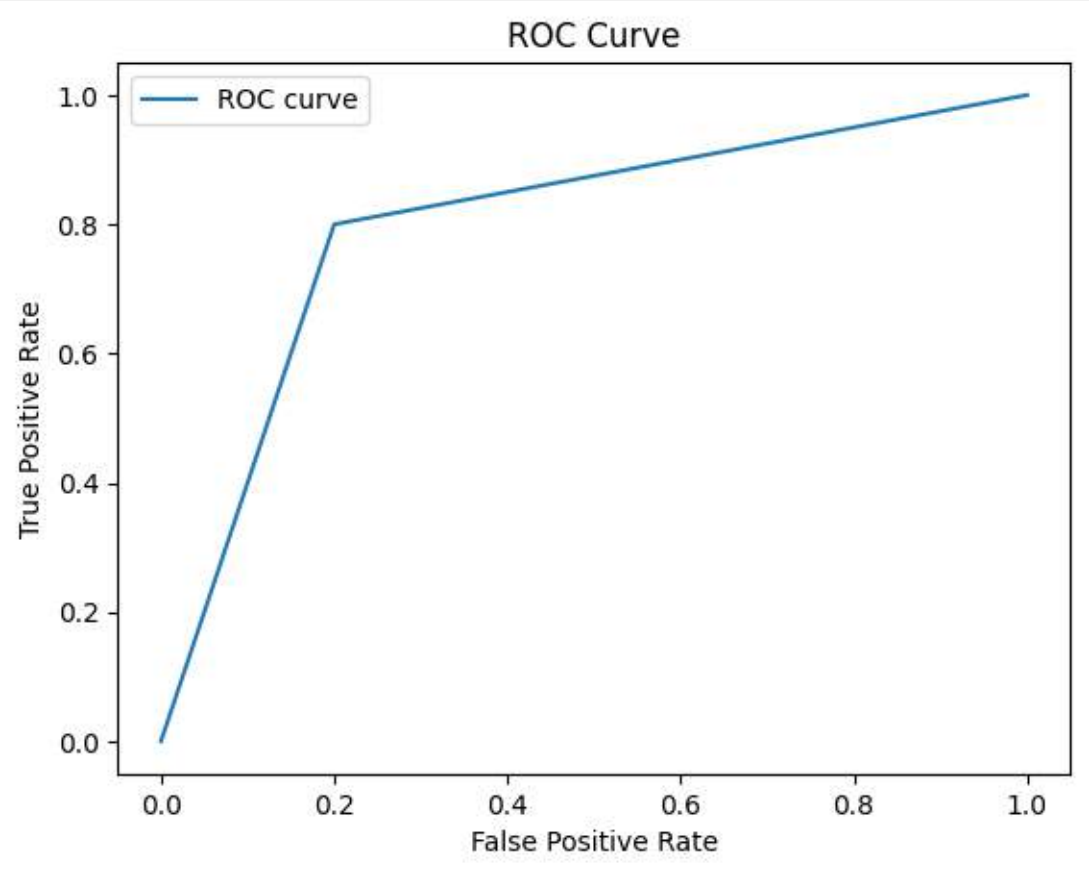
```
plt.legend(loc='best')
plt.show()

# AUC Score
auc_score = roc_auc_score(y_test, y_pred)
print("AUC Score:", auc_score)
```

Output:




```
              precision    recall  f1-score   support

           0       0.80      0.80      0.80         5
           1       0.80      0.80      0.80         5

    accuracy                           0.80        10
   macro avg       0.80      0.80      0.80        10
weighted avg       0.80      0.80      0.80        10
```


Precision-Recall Curve
Precision-Recall curve
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve
ROC curve
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
AUC Score: 0.8
```

11.7. Interpretasi Hasil Error

Menginterpretasi hasil dari analisis error membantu dalam memahami kekuatan dan kelemahan model. Berikut ini adalah beberapa langkah untuk menginterpretasi hasil dari contoh yang telah diberikan, termasuk confusion matrix, classification report, ROC curve, precision-recall curve, dan AUC score.

- Kesalahan Prediksi (False Positives dan False Negatives): Dari confusion matrix, kita dapat melihat bahwa model membuat beberapa kesalahan prediksi. False negatives menunjukkan bahwa model kehilangan beberapa data positif, sedangkan false positives menunjukkan bahwa model salah memprediksi data negatif sebagai positif. Kedua jenis kesalahan ini memiliki implikasi yang berbeda tergantung pada konteks masalah.
- Kinerja pada Kelas Tidak Seimbang: Classification report menunjukkan bahwa model memiliki precision dan recall yang cukup baik untuk kedua kelas, tetapi ada ruang untuk perbaikan terutama jika satu kelas lebih penting daripada yang lain.
- Kinerja Keseluruhan: AUC score dan precision-recall curve menunjukkan bahwa model memiliki kinerja yang cukup baik dalam membedakan antara kelas positif dan negatif. Namun, jika AUC score atau precision-recall curve menunjukkan kinerja yang rendah, ini mungkin menunjukkan perlunya peningkatan model atau penanganan khusus untuk kelas yang tidak seimbang.

11.8. Teknik Deployment Machine Learning

Pada materi ini, kita akan membahas teknik deployment machine learning, yaitu proses menerapkan model yang sudah dilatih ke dalam lingkungan produksi sehingga dapat diakses dan digunakan oleh pengguna atau sistem lainnya. Deployment bertujuan untuk memastikan model machine learning dapat memberikan prediksi atau rekomendasi secara real-time atau sesuai kebutuhan, memungkinkan integrasi dengan aplikasi atau layanan. Teknik deployment meliputi berbagai pendekatan dan metode untuk menyajikan model secara efisien dan skalabel dalam lingkungan operasional.

11.9. Teknik dan Metode Deployment Machine Learning

Pada video ini, kita akan mempelajari berbagai teknik dan metode dalam deployment machine learning, yaitu proses menempatkan model yang telah dilatih ke dalam lingkungan produksi agar dapat dimanfaatkan secara langsung. Video ini akan menjelaskan metode deployment, mulai dari deployment lokal di server internal, deployment berbasis cloud dengan platform seperti AWS atau Google Cloud, hingga edge deployment, di mana model dijalankan pada perangkat pengguna seperti ponsel atau sensor IoT. Selain itu, kita juga akan melihat konsep API untuk menghubungkan model dengan aplikasi, containerization menggunakan Docker untuk pengelolaan yang lebih mudah, serta orkestrasi dengan Kubernetes untuk pengelolaan deployment dalam skala besar. Dengan memahami teknik-teknik ini, Anda akan dapat memilih metode deployment yang paling sesuai untuk memastikan model machine learning dapat berjalan dengan optimal di berbagai lingkungan produksi.

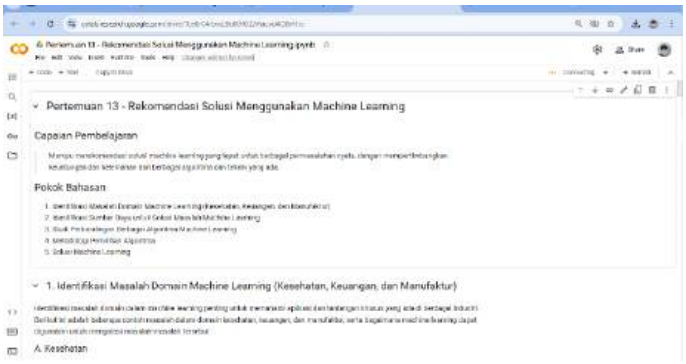
Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 12. Rekomendasi Solusi Menggunakan Machine Learning

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan berbagai algoritma machine learning.
2. Analisis keuntungan dan kelemahan algoritma.
3. Pertimbangan dalam memilih teknik yang tepat.
4. Studi kasus permasalahan nyata dan solusi machine learning.
5. Rekomendasi solusi berdasarkan analisis algoritma.

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



12.1. Identifikasi Masalah Domain Machine Learning (Kesehatan, Keuangan, dan Manufaktur)

Identifikasi masalah domain dalam machine learning penting untuk memahami aplikasi dan tantangan khusus yang ada di berbagai industri. Berikut ini adalah beberapa contoh masalah dalam domain kesehatan, keuangan, dan manufaktur, serta bagaimana machine learning dapat digunakan untuk mengatasi masalah-masalah tersebut:

A. Kesehatan

a. Deteksi Penyakit:

- Contoh: Mendiagnosis kanker dari gambar medis (misalnya, mammogram, MRI, atau CT scan).
- Tantangan: Data medis sering kali tidak seimbang (lebih sedikit contoh kasus positif), dan data privasi menjadi isu penting.

- Pendekatan: Menggunakan deep learning untuk analisis gambar, seperti Convolutional Neural Networks (CNNs), dan teknik augmentasi data untuk mengatasi ketidakseimbangan data.

b. Prediksi Keluaran Pasien:

- Contoh: Memprediksi tingkat kelangsungan hidup pasien setelah operasi.
- Tantangan: Data medis yang tidak lengkap dan variabilitas antar pasien.
- Pendekatan: Menggunakan model prediksi seperti Random Forest atau Gradient Boosting untuk menangani data yang tidak lengkap dan variabel.

c. Pengembangan Obat:

- Contoh: Menemukan kombinasi bahan kimia yang efektif untuk mengembangkan obat baru.
- Tantangan: Ruang pencarian yang sangat besar dan mahalnya eksperimen laboratorium.
- Pendekatan: Menggunakan algoritma machine learning untuk mempercepat proses penemuan obat dengan memprediksi interaksi antara molekul dan target biologis.

B. Keuangan (Finance)

a. Deteksi Penipuan:

- Contoh: Mendeteksi transaksi kartu kredit yang mencurigakan.
- Tantangan: Penipu selalu mengubah metode mereka, dan data transaksi penipuan sering kali tidak seimbang.
- Pendekatan: Menggunakan algoritma supervised learning seperti Logistic Regression atau Random Forest, serta teknik unsupervised learning untuk mendeteksi anomali.

b. Prediksi Kredit:

- Contoh: Menilai kelayakan kredit pemohon pinjaman.
- Tantangan: Data pemohon yang beragam dan potensi bias dalam data historis.
- Pendekatan: Menggunakan model seperti Support Vector Machines (SVM) atau Neural Networks untuk memprediksi risiko gagal bayar berdasarkan data demografis dan keuangan pemohon.

3. Portofolio Investasi:

- Contoh: Mengoptimalkan alokasi aset dalam portofolio investasi.
- Tantangan: Volatilitas pasar dan ketidakpastian dalam pergerakan harga.

- Pendekatan: Menggunakan teknik reinforcement learning dan algoritma optimasi untuk mengembangkan strategi alokasi aset yang adaptif.

C. Manufaktur (Manufacturing)

a. Pemeliharaan Prediktif:

- Contoh: Memprediksi kapan mesin akan gagal untuk melakukan pemeliharaan sebelum terjadi kerusakan.
- Tantangan: Data sensor yang beragam dan dalam jumlah besar, serta deteksi anomali yang sulit.
- Pendekatan: Menggunakan machine learning untuk analisis data sensor, seperti Time Series Analysis dan Recurrent Neural Networks (RNNs), untuk memprediksi kegagalan mesin.

b. Kontrol Kualitas:

- Contoh: Mendeteksi cacat produk dalam lini produksi.
- Tantangan: Variasi dalam jenis cacat dan kecepatan produksi yang tinggi.
- Pendekatan: Menggunakan computer vision dan deep learning (CNNs) untuk mendeteksi cacat pada produk berdasarkan gambar yang diambil dari lini produksi.

3. Optimalisasi Proses Produksi:

- Contoh: Mengoptimalkan aliran bahan baku dan proses produksi untuk meningkatkan efisiensi.
- Tantangan: Kompleksitas proses manufaktur dan interaksi antara berbagai variabel.
- Pendekatan: Menggunakan algoritma machine learning seperti reinforcement learning dan simulasi untuk mengoptimalkan berbagai parameter proses produksi.

Implementasi

Berikut adalah contoh implementasi sederhana untuk deteksi penipuan di domain keuangan menggunakan Logistic Regression di Python:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Load dataset (misalnya dataset transaksi penipuan)
data = pd.read_csv('creditcard.csv')

# Preprocessing
```

```
X = data.drop('Class', axis=1)  # Fitur
y = data['Class']  # Target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Define model
model = LogisticRegression()

# Train model
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluation
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

print("Classification Report:\n", classification_report(y_test, y_pred))
```

12.2. Identifikasi Sumber Daya untuk Solusi Masalah Machine Learning

Untuk mengatasi masalah dalam domain machine learning, penting untuk mengidentifikasi dan memanfaatkan berbagai sumber daya yang diperlukan. Sumber daya ini dapat berupa data, perangkat keras, perangkat lunak, keahlian manusia, serta komunitas dan literatur yang relevan. Berikut adalah identifikasi sumber daya untuk solusi masalah machine learning dalam konteks kesehatan, keuangan, dan manufaktur:

A. Sumber Daya Data

a. Kesehatan

- Data Medis: Gambar medis (MRI, CT scan), rekam medis elektronik (EMR), data genetik.
- Dataset Publik: NIH Chest X-rays, MIMIC-III (rekam medis ICU).
- Organisasi: Rumah sakit, klinik, laboratorium penelitian.

b. Keuangan

- Data Transaksi: Catatan transaksi kartu kredit/debit, data pasar saham, laporan keuangan.
- Dataset Publik: Kaggle's Credit Card Fraud Detection dataset, Yahoo Finance.
- Organisasi: Bank, lembaga keuangan, platform perdagangan.

c. Manufaktur

- Data Sensor: Data dari sensor mesin, catatan pemeliharaan.

- Dataset Publik: UCI Machine Learning Repository (datasets for manufacturing processes).
- Organisasi: Pabrik, lini produksi.

B. Perangkat Keras

- Komputer: Workstation dengan CPU yang kuat, RAM besar, dan penyimpanan cepat.
- GPU: Untuk mempercepat pelatihan model deep learning (misalnya, NVIDIA GPUs).
- Cluster Komputasi: Untuk pemrosesan data skala besar dan pelatihan model (misalnya, AWS, Google Cloud, Azure).

C. Perangkat Lunak dan Alat

Framework Machine Learning:

- TensorFlow: Untuk deep learning.
- PyTorch: Untuk penelitian dan pengembangan model deep learning.
- scikit-learn: Untuk algoritma machine learning klasik.

Libraries:

- Pandas: Untuk manipulasi dan analisis data.
- NumPy: Untuk komputasi numerik.
- Matplotlib/Seaborn: Untuk visualisasi data.

Tools:

- Jupyter Notebook: Untuk eksperimen interaktif.
- TensorBoard: Untuk visualisasi pelatihan model.
- DVC: Untuk versioning data dan model.
- Docker: Untuk kontainerisasi aplikasi.

D. Keahlian dan Tenaga Ahli

- Data Scientists: Ahli dalam analisis data dan pengembangan model machine learning.
- Machine Learning Engineers: Ahli dalam implementasi dan deployment model.
- Domain Experts: Spesialis dalam bidang kesehatan, keuangan, atau manufaktur untuk memberikan wawasan dan pengetahuan kontekstual.

- Software Engineers: Untuk integrasi sistem dan pengembangan perangkat lunak yang diperlukan.

5. Komunitas dan Sumber Daya Pembelajaran

Kursus Online:

- Coursera: Kursus dari universitas ternama (misalnya, Machine Learning oleh Andrew Ng).
- edX: Kursus tentang machine learning dan AI.
- Udacity: Nanodegree tentang deep learning, data engineering.

Forum dan Komunitas:

- Kaggle: Kompetisi data science, forum diskusi.
- Stack Overflow: Untuk bertanya dan menjawab pertanyaan teknis.
- GitHub: Untuk berbagi kode dan kolaborasi proyek.

Literatur:

- Buku: "Deep Learning" oleh Ian Goodfellow, Yoshua Bengio, Aaron Courville; "Pattern Recognition and Machine Learning" oleh Christopher M. Bishop.
- Makalah Penelitian: Akses melalui arXiv, Google Scholar.

6. Infrastruktur dan Alat Pengelolaan Proyek

- Version Control: Git untuk pengelolaan versi kode.
- Project Management Tools: Trello, Jira untuk pengelolaan tugas dan kolaborasi tim.
- CI/CD Pipelines: Jenkins, GitLab CI/CD untuk otomatisasi testing dan deployment model.

Contoh Implementasi Sumber Daya dalam Proyek Machine Learning

Berikut adalah contoh sederhana menggunakan beberapa sumber daya yang telah disebutkan untuk proyek deteksi penipuan dalam domain keuangan:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset (misalnya dari Kaggle)
data = pd.read_csv('creditcard.csv')
```

```
# Preprocessing
X = data.drop('Class', axis=1)  # Fitur
y = data['Class']  # Target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Define model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train model
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluation
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print("Classification Report:\n", classification_report(y_test, y_pred))
```

12.3. Studi Perbandingan Berbagai Algoritma Machine Learning

Membandingkan berbagai algoritma machine learning dapat membantu menentukan algoritma mana yang paling cocok untuk masalah tertentu berdasarkan kinerja, kompleksitas, interpretabilitas, dan faktor lainnya. Berikut adalah studi perbandingan beberapa algoritma machine learning yang umum digunakan dalam domain kesehatan, keuangan, dan manufaktur.

Algoritma yang Dibandingkan

- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- k-Nearest Neighbors (k-NN)
- Naive Bayes
- Gradient Boosting Machines (GBM)
- Neural Networks (NN)

Kriteria Perbandingan

- Kinerja (Accuracy, Precision, Recall, F1-Score)

- Kompleksitas Model
- Kecepatan Pelatihan dan Prediksi
- Interpretabilitas
- Robustness terhadap Overfitting
- Skalabilitas
- Kebutuhan Data
- Kemampuan Mengatasi Data Tidak Seimbang

Studi Perbandingan

a. Logistic Regression

- Kinerja: Baik untuk masalah linear, tetapi mungkin kurang baik untuk data kompleks
- Kompleksitas Model: Rendah, mudah diinterpretasi.
- Kecepatan: Cepat untuk pelatihan dan prediksi.
- Interpretabilitas: Sangat mudah diinterpretasi, koefisien menunjukkan kontribusi masing-masing fitur.
- Robustness: Cenderung overfitting pada data dengan banyak fitur jika tidak diberi regularisasi.
- Skalabilitas: Baik untuk data skala besar.
- Kebutuhan Data: Membutuhkan data yang cukup banyak untuk generalisasi yang baik.
- Data Tidak Seimbang: Kurang optimal tanpa penyesuaian (misalnya, penyesuaian threshold).

b. Decision Trees

- Kinerja: Bagus untuk data dengan relasi non-linear, tetapi rentan terhadap overfitting.
- Kompleksitas Model: Sedang, bisa menjadi sangat kompleks.
- Kecepatan: Cepat untuk pelatihan, prediksi sangat cepat.
- Interpretabilitas: Mudah diinterpretasi, visualisasi pohon keputusan.
- Robustness: Rentan terhadap overfitting.
- Skalabilitas: Kurang optimal untuk dataset sangat besar.
- Kebutuhan Data: Tidak membutuhkan banyak data.
- Data Tidak Seimbang: Rentan terhadap ketidakseimbangan data.

c. Random Forests

- Kinerja: Sangat baik, mengurangi overfitting yang sering terjadi pada decision trees.
- Kompleksitas Model: Tinggi, ensemble dari banyak decision trees.
- Kecepatan: Relatif lambat untuk pelatihan, cepat untuk prediksi.
- Interpretabilitas: Kurang diinterpretasi, tetapi pentingnya fitur dapat diukur.
- Robustness: Sangat robust, mengurangi overfitting dengan ensemble.
- Skalabilitas: Baik, tetapi pelatihan dapat memakan waktu lama untuk dataset besar.
- Kebutuhan Data: Membutuhkan data yang cukup besar.
- Data Tidak Seimbang: Lebih baik dalam menangani data tidak seimbang daripada pohon keputusan tunggal.

d. Support Vector Machines (SVM)

- Kinerja: Sangat baik untuk data dengan dimensi tinggi, terutama dengan kernel non-linear.
- Kompleksitas Model: Tinggi, terutama dengan kernel non-linear.
- Kecepatan: Lambat untuk pelatihan, cepat untuk prediksi.
- Interpretabilitas: Sulit diinterpretasi, terutama dengan kernel non-linear.
- Robustness: Cenderung overfitting dengan banyak fitur atau parameter kernel yang salah.
- Skalabilitas: Kurang baik untuk dataset sangat besar.
- Kebutuhan Data: Membutuhkan data yang cukup banyak.
- Data Tidak Seimbang: Memerlukan penyesuaian (misalnya, kelas berbobot).

e. k-Nearest Neighbors (k-NN)

- Kinerja: Baik untuk masalah dengan pola yang jelas, buruk untuk data berisik.
- Kompleksitas Model: Rendah, sangat sederhana.
- Kecepatan: Lambat untuk prediksi, cepat untuk pelatihan.
- Interpretabilitas: Mudah diinterpretasi.
- Robustness: Rentan terhadap overfitting, sensitif terhadap noise.
- Skalabilitas: Kurang baik untuk dataset sangat besar.
- Kebutuhan Data: Membutuhkan data yang cukup banyak.

- Data Tidak Seimbang: Kurang optimal tanpa penyesuaian.

f. Naive Bayes

- Kinerja: Baik untuk teks atau masalah dengan fitur independen.
- Kompleksitas Model: Rendah, sederhana.
- Kecepatan: Sangat cepat untuk pelatihan dan prediksi.
- Interpretabilitas: Mudah diinterpretasi, terutama dengan asumsi independensi fitur.
- Robustness: Tidak rentan terhadap overfitting, tetapi asumsi independensi mungkin tidak realistis.
- Skalabilitas: Sangat baik untuk dataset besar.
- Kebutuhan Data: Tidak membutuhkan banyak data.
- Data Tidak Seimbang: Relatif baik dalam menangani data tidak seimbang.

g. Gradient Boosting Machines (GBM)

- Kinerja: Sangat baik, mengatasi masalah overfitting dengan membangun model secara bertahap.
- Kompleksitas Model: Tinggi, ensemble dari banyak model sederhana.
- Kecepatan: Lambat untuk pelatihan, cepat untuk prediksi.
- Interpretabilitas: Sulit diinterpretasi, tetapi pentingnya fitur dapat diukur.
- Robustness: Sangat robust, mengurangi overfitting.
- Skalabilitas: Baik, tetapi memerlukan tuning yang cermat.
- Kebutuhan Data: Membutuhkan data yang cukup banyak.
- Data Tidak Seimbang: Baik dalam menangani data tidak seimbang dengan penyesuaian.

h. Neural Networks (NN)

- Kinerja: Sangat baik untuk data yang kompleks dan besar, terutama dengan deep learning.
- Kompleksitas Model: Sangat tinggi, dengan banyak lapisan dan parameter.
- Kecepatan: Lambat untuk pelatihan, cepat untuk prediksi.
- Interpretabilitas: Sangat sulit diinterpretasi, sering dianggap sebagai "black box".
- Robustness: Cenderung overfitting tanpa regularisasi yang tepat.

- Skalabilitas: Sangat baik untuk dataset sangat besar dengan infrastruktur yang tepat.
- Kebutuhan Data: Membutuhkan data dalam jumlah besar untuk performa yang baik.
- Data Tidak Seimbang: Dapat menangani data tidak seimbang dengan teknik seperti oversampling atau penyesuaian loss function.

Contoh Perbandingan Empiris

Berikut adalah contoh perbandingan empiris menggunakan dataset "Credit Card Fraud Detection":

```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix

# Load dataset
data = pd.read_csv('creditcard.csv')

# Preprocessing
X = data.drop('Class', axis=1)
y = data['Class']

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Define models
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "Support Vector Machine": SVC(),
    "k-NN": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB()
}

# Train and evaluate models
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results[name] = {
        "Confusion Matrix": confusion_matrix(y_test, y_pred),
        "Classification Report": classification_report(y_test, y_pred)
    }
```

```
# Print results
for name, metrics in results.items():
    print(f"Model: {name}")
    print("Confusion Matrix:\n", metrics["Confusion Matrix"])
    print("Classification Report:\n", metrics["Classification Report"])
```

12.4. Metodologi Pemilihan Algoritma

Pemilihan algoritma machine learning yang tepat sangat penting untuk mendapatkan model yang berkinerja baik dan memenuhi kebutuhan spesifik dari masalah yang dihadapi. Berikut adalah metodologi langkah-demi-langkah untuk memilih algoritma machine learning yang sesuai:

Definisikan Masalah

- Klasifikasi: Memisahkan data ke dalam kategori yang berbeda.
- Regresi: Memprediksi nilai numerik.
- Clustering: Mengelompokkan data tanpa label.
- Anomaly Detection: Mendeteksi data yang tidak normal atau anomali.
- Rekomendasi: Merekomendasikan item atau konten berdasarkan data pengguna.

Pahami Data

- Jumlah Data: Banyak atau sedikit data.
- Kualitas Data: Apakah data bersih atau banyak missing values?
- Fitur Data: Apakah data memiliki banyak fitur atau sedikit?
- Distribusi Data: Apakah data seimbang atau tidak seimbang?
- Kompleksitas Data: Apakah data memiliki relasi linier atau non-linier?

Pilih Algoritma Berdasarkan Kriteria

- Kinerja: Tingkat akurasi, presisi, recall, F1-score yang diinginkan.
- Kecepatan: Waktu pelatihan dan prediksi.
- Skalabilitas: Kemampuan menangani data besar.
- Interpretabilitas: Seberapa mudah model untuk dipahami.
- Robustness: Ketahanan terhadap overfitting.
- Kebutuhan Data: Jumlah data yang diperlukan untuk performa yang baik.
- Kemampuan Mengatasi Data Tidak Seimbang: Apakah algoritma bisa mengatasi data tidak seimbang dengan baik?

12.5. Solusi Machine Learning

a. Definisikan Masalah

Langkah pertama dalam solusi machine learning adalah mendefinisikan masalah yang ingin dipecahkan. Identifikasi jenis masalah yang dihadapi:

- Klasifikasi: Memisahkan data ke dalam kategori yang berbeda, seperti memprediksi apakah email adalah spam atau bukan.
- Regresi: Memprediksi nilai numerik, misalnya memprediksi harga rumah berdasarkan fitur-fitur tertentu.
- Clustering: Mengelompokkan data tanpa label, misalnya segmentasi pelanggan berdasarkan pola pembelian.
- Anomaly Detection: Mendeteksi data yang tidak normal atau anomali, seperti mendeteksi transaksi keuangan yang mencurigakan.
- Rekomendasi: Merekomendasikan item atau konten berdasarkan data pengguna, seperti rekomendasi film atau produk.

b. Pahami Data

Setelah mendefinisikan masalah, langkah berikutnya adalah memahami data yang tersedia. Beberapa aspek yang perlu dipertimbangkan meliputi:

- Jumlah Data: Menentukan apakah data yang tersedia banyak atau sedikit. Ini akan mempengaruhi pemilihan algoritma dan pendekatan.
- Kualitas Data: Mengevaluasi apakah data bersih atau banyak missing values. Data yang bersih akan memudahkan proses analisis dan pemodelan.
- Fitur Data: Menilai apakah data memiliki banyak fitur atau sedikit. Jumlah fitur akan mempengaruhi kompleksitas model.
- Distribusi Data: Memeriksa apakah data seimbang atau tidak seimbang. Data yang tidak seimbang membutuhkan teknik khusus untuk memastikan model tidak bias.
- Kompleksitas Data: Menentukan apakah data memiliki relasi linier atau non-linier. Ini akan membantu dalam memilih algoritma yang tepat.

c. Pilih Algoritma Berdasarkan Kriteria

Berdasarkan pemahaman tentang data, pilih algoritma yang sesuai dengan kriteria tertentu:

- Kinerja: Menentukan tingkat akurasi, presisi, recall, dan F1-score yang diinginkan.

- Kecepatan: Mempertimbangkan waktu yang dibutuhkan untuk pelatihan dan prediksi.
- Skalabilitas: Kemampuan algoritma untuk menangani data besar.
- Interpretabilitas: Seberapa mudah model untuk dipahami oleh manusia.
- Robustness: Ketahanan model terhadap overfitting.
- Kebutuhan Data: Jumlah data yang diperlukan untuk mencapai performa yang baik.
- Kemampuan Mengatasi Data Tidak Seimbang: Menilai apakah algoritma bisa mengatasi data tidak seimbang dengan baik.

d. Proses Eksperimen

Langkah ini melibatkan eksperimen untuk membangun dan mengevaluasi model:

- Pra-Pemrosesan Data: Melakukan normalisasi atau standarisasi data, penanganan missing values, dan feature selection atau engineering.
- Splitting Data: Membagi dataset menjadi data latih (train) dan data uji (test). Pertimbangkan juga penggunaan cross-validation untuk evaluasi yang lebih stabil.
- Pelatihan Model: Melatih beberapa algoritma yang relevan berdasarkan pemahaman data dan kriteria yang ditentukan.
- Evaluasi Model: Menggunakan metrik evaluasi yang sesuai seperti accuracy, precision, recall, dan F1-score untuk klasifikasi. Cross-validation dilakukan untuk mendapatkan gambaran performa yang lebih baik.
- Tuning Hyperparameter: Menggunakan grid search atau random search untuk menemukan kombinasi hyperparameter terbaik.
- Analisis Hasil: Membandingkan hasil dari berbagai model dan memilih model terbaik berdasarkan kinerja dan kriteria lain yang telah ditentukan.

e. Implementasi dan Monitoring

Setelah model terbaik dipilih, langkah terakhir adalah implementasi dan monitoring:

- Implementasi: Melakukan deploy model terbaik ke lingkungan produksi. Buat pipeline otomatis untuk pemrosesan data dan prediksi.
- Monitoring: Memantau performa model di lingkungan produksi. Lakukan pembaruan dan retrain model secara berkala jika diperlukan untuk menjaga performa.

12.6. Deteksi Anomali Serangan Siber Menggunakan Machine Learning

Video ini akan menjelaskan penerapan machine learning dalam mendeteksi anomali sebagai langkah penting untuk mencegah dan mengidentifikasi serangan siber. Deteksi anomali melibatkan pengenalan perilaku atau aktivitas yang menyimpang dari pola normal dalam jaringan atau sistem, yang sering kali menandakan adanya potensi ancaman atau serangan. Dalam video ini, Anda akan melihat contoh konkret, seperti deteksi serangan DDoS dan malware, untuk memahami cara teknik ini diterapkan di dunia nyata. Dengan penerapan machine learning, organisasi dapat memperkuat sistem keamanan mereka, memantau ancaman secara real-time, dan merespons serangan dengan lebih cepat dan efektif.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

12.7. AI Menggantikan Manusia?

Video ini akan membahas pertanyaan besar di era kecerdasan buatan: "Apakah AI akan menggantikan manusia?" Seiring berkembangnya AI, teknologi ini semakin mampu menangani berbagai tugas yang sebelumnya hanya bisa dilakukan manusia, seperti menganalisis data, mengenali gambar, dan membuat keputusan. Namun, penting untuk diingat bahwa AI diciptakan untuk melengkapi, bukan menggantikan manusia. AI memang dapat mengotomatisasi tugas-tugas yang berulang dan meningkatkan efisiensi, tetapi kemampuan seperti kreativitas, empati, dan pengambilan keputusan kompleks masih tetap menjadi keahlian yang unik bagi manusia.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 13. Arsitektur Model Machine Learning

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan desain arsitektur model machine learning.
2. Pemilihan fitur yang relevan.
3. Pilihan algoritma yang sesuai dengan masalah.
4. Penentuan parameter yang optimal.
5. Contoh penerapan arsitektur model untuk kebutuhan spesifik.

13.1. Arsitektur Machine Learning

Arsitektur machine learning mencakup desain dan struktur sistem yang mengelola aliran data, pemrosesan, dan prediksi dari model machine learning. Berikut adalah penjelasan tentang arsitektur machine learning secara umum:

1. Pengumpulan Data

Proses ini melibatkan pengumpulan data dari berbagai sumber, seperti database, file, API, atau streaming data. Data ini harus relevan dengan masalah yang ingin diselesaikan dan berkualitas tinggi.

2. Pra-Pemrosesan Data

Data yang dikumpulkan biasanya memerlukan pra-pemrosesan sebelum dapat digunakan untuk pelatihan model. Langkah-langkah pra-pemrosesan meliputi:

- Pembersihan Data: Menghapus atau memperbaiki data yang hilang, duplikat, atau tidak konsisten.
- Transformasi Data: Normalisasi atau standarisasi fitur untuk memastikan skala data konsisten.
- Pengkodean Data Kategorikal: Mengonversi fitur kategorikal menjadi format numerik.
- Pembagian Data: Membagi data menjadi subset pelatihan, validasi, dan pengujian.

3. Pemilihan Model

Memilih algoritma machine learning yang sesuai berdasarkan jenis masalah (klasifikasi, regresi, clustering, dll.) dan kriteria pemilihan model (kinerja, kecepatan, interpretabilitas, dll.).

4. Pelatihan Model

Melatih model dengan menggunakan data pelatihan. Selama pelatihan, model belajar untuk mengidentifikasi pola dalam data. Proses ini melibatkan:

- Optimasi: Menggunakan algoritma optimisasi seperti Gradient Descent untuk memperbarui parameter model.
- Validasi: Menggunakan data validasi untuk menghindari overfitting dan memilih hyperparameter terbaik.

5. Evaluasi Model

Setelah pelatihan, model diuji menggunakan data pengujian untuk mengevaluasi kinerjanya. Metrik evaluasi yang umum termasuk akurasi, precision, recall, F1-score, dan AUC-ROC untuk masalah klasifikasi, serta mean squared error (MSE) dan R^2 untuk regresi.

6. Tuning Hyperparameter

Menyesuaikan hyperparameter model untuk meningkatkan kinerja. Ini sering dilakukan dengan teknik seperti grid search atau random search.

7. Deploy dan Integrasi

Mengimplementasikan model ke lingkungan produksi, di mana model dapat digunakan untuk prediksi dengan data baru. Ini melibatkan integrasi model dengan sistem yang ada, serta pengaturan pipeline otomatis untuk inferensi.

8. Monitoring dan Pemeliharaan

Memantau kinerja model di lingkungan produksi dan melakukan pembaruan atau retrain model secara berkala untuk memastikan bahwa model tetap relevan dan akurat.

Arsitektur Machine Learning Berbasis Deep Learning

a. Arsitektur Jaringan Saraf

- Jaringan Saraf Feedforward: Termasuk layer input, hidden layers, dan output layer. Data mengalir dalam satu arah dari input ke output.
- Jaringan Saraf Konvolusi (CNN): Digunakan terutama untuk pengolahan citra dan video, mengintegrasikan layer konvolusi, pooling, dan fully connected layers.
- Jaringan Saraf Rekuren (RNN): Digunakan untuk data sekuensial, seperti teks dan time series, dengan kemampuan untuk menangkap ketergantungan temporal.
- Transformers: Arsitektur yang populer untuk pemrosesan bahasa alami (NLP), menggunakan mekanisme perhatian (attention) untuk menangkap hubungan antar kata dalam kalimat.

b. Pipeline Deep Learning

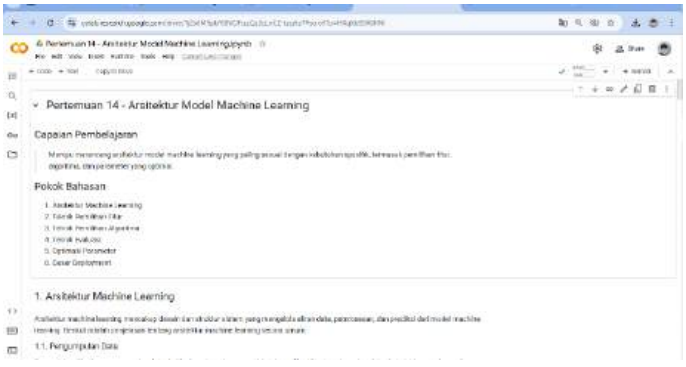
- Preprocessing: Melakukan transformasi data khusus untuk deep learning, seperti augmentasi citra.
- Training: Menggunakan teknik khusus seperti backpropagation untuk melatih jaringan saraf dengan data besar.
- Evaluation: Metrik evaluasi khusus untuk deep learning, termasuk loss function yang relevan dengan task (cross-entropy untuk klasifikasi, MSE untuk regresi).
- Deployment: Mengintegrasikan model deep learning ke dalam aplikasi dengan efisiensi dan skalabilitas.

c. Arsitektur Machine Learning dalam Skala Besar

- Data Pipeline: Mengelola aliran data besar dengan menggunakan alat dan platform seperti Apache Kafka, Apache Spark, dan Google Dataflow.
- Training dan Inferensi di Cloud: Menggunakan layanan cloud seperti AWS SageMaker, Google AI Platform, dan Azure Machine Learning untuk pelatihan dan inferensi skala besar dengan dukungan GPU/TPU.
- Model Management: Mengelola versi model, metadata, dan artefak model menggunakan platform seperti MLflow, DVC, atau ModelDB.
- Scalability dan Distributed Training: Memanfaatkan teknik pelatihan terdistribusi dan parallel untuk mempercepat pelatihan model pada dataset besar dan infrastruktur komputasi besar.

Arsitektur machine learning yang efektif mengintegrasikan semua elemen ini, dari pengumpulan data hingga deployment dan pemeliharaan, untuk membangun sistem yang kuat dan skalabel yang dapat memberikan wawasan dan prediksi yang berharga.

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



13.2. Teknik Pemilihan Fitur

Pemilihan fitur (feature selection) adalah proses penting dalam machine learning yang bertujuan untuk memilih subset fitur yang paling relevan dari data untuk digunakan dalam membangun model. Teknik pemilihan fitur membantu

meningkatkan kinerja model, mengurangi overfitting, dan mengurangi waktu pelatihan. Berikut adalah berbagai teknik pemilihan fitur yang sering digunakan:

a. Filter Methods

Filter methods menilai fitur secara individual dan memilih fitur berdasarkan metrik statistik. Teknik ini tidak melibatkan model pembelajaran mesin.

- Chi-Square Test: Mengukur ketergantungan antara fitur dan label target. Fitur yang memiliki nilai chi-square tinggi dianggap lebih relevan.
- Mutual Information: Mengukur seberapa besar ketergantungan informasi antara fitur dan label target. Fitur dengan mutual information tinggi dianggap lebih informatif.
- Correlation Coefficient: Mengukur kekuatan hubungan linier antara fitur dan label target. Fitur dengan korelasi tinggi dianggap lebih penting.
- ANOVA F-Value: Untuk fitur numerik dengan label kategorikal, ANOVA F-test mengukur variasi antar kelompok dibandingkan dengan variasi dalam kelompok.

b. Wrapper Methods

Wrapper methods menggunakan model pembelajaran mesin untuk menilai subset fitur. Ini melibatkan pelatihan model berulang kali dengan berbagai subset fitur.

- Forward Selection: Memulai dengan fitur kosong dan menambahkan fitur satu per satu yang paling meningkatkan kinerja model. Proses ini berhenti ketika penambahan fitur tidak meningkatkan kinerja secara signifikan.
- Backward Elimination: Memulai dengan semua fitur dan secara iteratif menghapus fitur yang paling sedikit mempengaruhi kinerja model. Proses ini berhenti ketika menghapus fitur mulai menurunkan kinerja model.
- Recursive Feature Elimination (RFE): Menggunakan model pembelajaran mesin untuk menghapus fitur yang kurang penting secara iteratif hingga mencapai jumlah fitur yang diinginkan.

c. Embedded Methods

Embedded methods mengintegrasikan pemilihan fitur sebagai bagian dari proses pelatihan model. Teknik ini sering melibatkan model yang secara otomatis memilih fitur yang relevan selama pelatihan.

- Lasso Regression (L1 Regularization): Menggunakan regularisasi L1 untuk memaksa beberapa koefisien fitur menjadi nol, sehingga secara otomatis memilih fitur yang lebih relevan.
- Decision Trees and Random Forests: Menggunakan pentingnya fitur yang dihitung berdasarkan bagaimana fitur membagi data pada node dalam pohon

keputusan. Fitur yang sering digunakan untuk membuat pembagian signifikan dianggap lebih penting.

- Gradient Boosting Machines (GBM): Mengukur pentingnya fitur berdasarkan kontribusi fitur terhadap pengurangan kesalahan model dalam proses boosting.

1. Dimensionality Reduction

Teknik ini mengurangi jumlah fitur dengan mentransformasi data ke dalam dimensi yang lebih rendah, biasanya dengan menjaga informasi sebanyak mungkin.

- Principal Component Analysis (PCA): Mengurangi dimensi data dengan mentransformasikan fitur asli ke dalam komponen utama yang memiliki variansi terbesar. PCA menghasilkan fitur baru yang merupakan kombinasi linier dari fitur asli.
- Linear Discriminant Analysis (LDA): Mengurangi dimensi dengan memaksimalkan pemisahan antar kelas dalam data. LDA mencari kombinasi fitur yang memaksimalkan variansi antar kelas dan meminimalkan variansi dalam kelas.
- t-Distributed Stochastic Neighbor Embedding (t-SNE): Mengurangi dimensi data dengan mempertahankan struktur lokal data. Ini berguna untuk visualisasi data dengan dimensi tinggi.

2. Feature Engineering

Feature engineering melibatkan penciptaan fitur baru yang dapat lebih baik mewakili informasi dalam data.

- Interaction Terms: Membuat fitur baru yang merupakan hasil perkalian dari fitur yang ada untuk menangkap hubungan interaksi antara fitur.
- Polynomial Features: Membuat fitur baru dengan mengkuadratkan atau mengalikan fitur yang ada untuk menangkap hubungan non-linier.
- Binning: Mengelompokkan nilai fitur kontinu ke dalam bin diskret untuk menangkap informasi yang terlewatkan oleh model.

3. Domain Knowledge

Memanfaatkan pengetahuan domain untuk memilih fitur yang dianggap relevan berdasarkan pemahaman mendalam tentang masalah yang sedang dipecahkan.

- Expert Input: Mengandalkan ahli domain untuk memberikan wawasan tentang fitur mana yang dianggap penting.
- Manual Selection: Memilih fitur berdasarkan pengetahuan dan pengalaman dalam bidang yang relevan.

Teknik pemilihan fitur yang tepat bergantung pada jenis data, tujuan model, dan algoritma yang digunakan. Kombinasi dari beberapa teknik sering kali memberikan hasil terbaik dalam mengidentifikasi fitur yang paling relevan dan meningkatkan kinerja model machine learnin

13.3. Teknik Pemilihan Algoritma

Pemilihan algoritma dalam machine learning memerlukan pertimbangan yang lebih spesifik dibandingkan pemilihan algoritma pada umumnya. Berikut adalah beberapa teknik dan faktor yang perlu diperhatikan dalam pemilihan algoritma machine learning:

a. Tipe Masalah

Klasifikasi: Untuk masalah di mana tujuan utamanya adalah mengelompokkan data ke dalam kategori yang telah ditentukan, seperti Logistic Regression, Decision Trees, Random Forests, atau Support Vector Machines (SVM). Regresi: Untuk masalah di mana tujuan utamanya adalah memprediksi nilai numerik, seperti Linear Regression, Ridge Regression, atau Lasso. Clustering: Untuk mengelompokkan data yang tidak berlabel, seperti K-Means, DBSCAN, atau Hierarchical Clustering. Dimensionality Reduction: Untuk mengurangi jumlah fitur dalam data, seperti Principal Component Analysis (PCA) atau t-Distributed Stochastic Neighbor Embedding (t-SNE).

b. Ukuran dan Jenis Data

Jumlah Data: Beberapa algoritma lebih cocok untuk dataset besar, seperti Neural Networks atau Gradient Boosting Machines. Algoritma lain seperti K-Nearest Neighbors (KNN) mungkin tidak efisien pada dataset yang sangat besar. Dimensionalitas: Data dengan banyak fitur mungkin memerlukan algoritma yang dapat menangani dimensi tinggi, seperti SVM dengan kernel trik atau Random Forest yang dapat menangani fitur yang tidak relevan. Data yang Hilang: Algoritma seperti Random Forest atau KNN dapat menangani data yang hilang lebih baik daripada algoritma lain seperti SVM.

c. Kompleksitas Model

Model yang Sederhana: Algoritma seperti Linear Regression atau Logistic Regression mudah diinterpretasi dan diimplementasikan Model yang Kompleks: Algoritma seperti Neural Networks atau Gradient Boosting Machines dapat menangani hubungan yang lebih kompleks dalam data, tetapi sering kali lebih sulit diinterpretasi.

d. Waktu dan Sumber Daya Komputasi

Training Time: Beberapa algoritma membutuhkan waktu pelatihan yang lama, seperti Neural Networks atau SVM dengan kernel yang kompleks. Inference Time: Untuk aplikasi yang membutuhkan prediksi cepat, seperti dalam sistem real-time,

algoritma seperti Logistic Regression atau Decision Trees mungkin lebih sesuai. Ketersediaan Hardware: Algoritma seperti Neural Networks memerlukan hardware khusus seperti GPU untuk mempercepat training.

e. Overfitting dan Generalisasi

Overfitting: Algoritma seperti Decision Trees atau Neural Networks rentan terhadap overfitting. Teknik seperti cross-validation, regularisasi, dan pruning dapat membantu mengurangi overfitting. Generalization: Algoritma seperti Random Forest atau SVM cenderung memiliki generalisasi yang baik jika di-tuning dengan benar.

f. Interpretabilitas Model

High Interpretability: Algoritma seperti Linear Regression, Logistic Regression, dan Decision Trees mudah diinterpretasi dan dijelaskan kepada non-teknis. Low Interpretability: Algoritma seperti Neural Networks atau ensemble methods seperti Random Forests dan Gradient Boosting Machines lebih sulit diinterpretasi.

g. Sifat Data

Linear vs Non-Linear: Untuk data yang memiliki hubungan linear, algoritma seperti Linear Regression atau SVM dengan kernel linear cocok. Untuk hubungan non-linear, Neural Networks atau SVM dengan kernel non-linear bisa lebih efektif. Outliers and Noise: Algoritma seperti Robust Regression atau Tree-based methods dapat lebih baik dalam mengatasi outliers dan noise dalam data.

h. Cross-validation and Hyperparameter Tuning

Cross-Validation: Teknik seperti k-fold cross-validation digunakan untuk mengevaluasi performa model secara keseluruhan dan memastikan bahwa model tidak overfit. Hyperparameter Tuning: Algoritma seperti Grid Search atau Random Search digunakan untuk menemukan kombinasi hyperparameter yang optimal untuk model.

i. Use Case Spesifik

Image Recognition: Convolutional Neural Networks (CNNs) sangat cocok untuk data gambar. Sequential Data: Recurrent Neural Networks (RNNs) atau Long Short-Term Memory (LSTM) cocok untuk data berurutan seperti teks atau data waktu.

13.4. Teknik Evaluasi

Teknik evaluasi dalam machine learning adalah langkah penting untuk menilai kinerja model dan memastikan bahwa model tersebut akan bekerja dengan baik pada data baru yang belum pernah dilihat sebelumnya. Berikut beberapa teknik evaluasi yang umum digunakan:

a. Train-Test Split

Deskripsi: Memisahkan dataset menjadi dua bagian: data pelatihan dan data pengujian. Model dilatih pada data pelatihan dan dievaluasi pada data pengujian. Kelebihan: Sederhana dan cepat. Kekurangan: Hasil evaluasi bisa bergantung pada bagaimana data dipisahkan.

b. Cross-Validation

K-Fold Cross-Validation

Deskripsi: Membagi data menjadi k bagian (folds), melatih model pada k-1 folds, dan menguji pada 1 fold yang tersisa. Proses ini diulang k kali, setiap fold digunakan sebagai data pengujian sekali. Kelebihan: Mengurangi ketergantungan pada satu set data pengujian, memberikan estimasi kinerja yang lebih stabil. Kekurangan: Lebih memakan waktu dan sumber daya komputasi.

Stratified K-Fold Cross-Validation

Deskripsi: Mirip dengan k-fold, tetapi membagi data sehingga setiap fold memiliki distribusi kelas yang sama. Kelebihan: Lebih baik untuk dataset yang tidak seimbang.

c. Leave-One-Out Cross-Validation (LOOCV)

Deskripsi: Setiap data poin digunakan sebagai data pengujian satu kali, sementara sisanya digunakan sebagai data pelatihan. Kelebihan: Menggunakan maksimal data untuk pelatihan. Kekurangan: Sangat memakan waktu dan sumber daya komputasi.

d. Bootstrap Sampling

Deskripsi: Mengambil sampel dengan penggantian dari dataset untuk membuat beberapa set data pelatihan, dan menguji model pada data yang tidak termasuk dalam sampel. Kelebihan: Berguna untuk mengestimasi variabilitas model. Kekurangan: Dapat memerlukan banyak sampel untuk estimasi yang baik.

e. Holdout Method

Deskripsi: Dataset dibagi menjadi tiga bagian: training set, validation set, dan test set. Model dilatih pada training set, di-tuning pada validation set, dan dievaluasi pada test set. Kelebihan: Memberikan cara untuk men-tune hyperparameter sebelum evaluasi akhir. Kekurangan: Membutuhkan lebih banyak data untuk membagi menjadi tiga set.

f. Time Series Cross-Validation

Deskripsi: Teknik khusus untuk data time series di mana data dibagi berdasarkan waktu untuk memastikan bahwa data pengujian selalu lebih baru daripada data pelatihan. Kelebihan: Menghormati urutan temporal data. Kekurangan: Bisa lebih rumit untuk diterapkan.

g. Hyperparameter Tuning dan Model Selection

Grid Search: Mencari kombinasi hyperparameter terbaik dengan mencoba setiap kemungkinan kombinasi. Random Search: Mencari kombinasi hyperparameter terbaik dengan mencoba kombinasi secara acak. Bayesian Optimization: Mencari kombinasi hyperparameter terbaik dengan pendekatan probabilistik.

13.5. Optimasi Parameter

Optimasi parameter (juga dikenal sebagai tuning hyperparameter) adalah proses penting dalam machine learning yang bertujuan untuk menemukan set parameter yang optimal untuk model sehingga kinerjanya dapat ditingkatkan. Berikut adalah beberapa metode umum yang digunakan untuk optimasi parameter:

1. Grid Search: Mencari kombinasi optimal dari hyperparameter dengan mencoba setiap kombinasi dalam ruang parameter yang telah ditentukan.
2. Random Search: Memilih kombinasi hyperparameter secara acak dalam ruang pencarian yang telah ditentukan.
3. Bayesian Optimization: Menggunakan metode probabilistik untuk memilih kombinasi hyperparameter berdasarkan kinerja sebelumnya.
4. Hyperband: Metode optimasi hyperparameter yang efisien dengan menggunakan teknik early-stopping.
5. genetic Algorithms: Menggunakan prinsip evolusi dan seleksi alam untuk mencari kombinasi hyperparameter yang optimal.
6. Simulated Annealing: Menggunakan prinsip fisika dari annealing untuk menemukan kombinasi hyperparameter yang optimal.
7. Tree-structured Parzen Estimator (TPE): Menggunakan model probabilistik berbasis pohon untuk mencari kombinasi hyperparameter yang optimal.

13.6. Dasar-dasar Deployment

Deployment adalah proses mengeluarkan aplikasi atau sistem dari lingkungan pengembangan ke lingkungan produksi di mana aplikasi tersebut dapat diakses dan digunakan oleh pengguna akhir. Berikut adalah dasar-dasar yang perlu dipahami dalam proses deployment:

Persiapan Lingkungan

Deployment machine learning melibatkan membawa model yang telah dilatih dan dioptimalkan ke dalam lingkungan produksi di mana ia dapat digunakan untuk membuat prediksi pada data baru. Proses ini mencakup beberapa langkah dan pertimbangan khusus. Berikut adalah langkah-langkah dasar dalam deployment machine learning:

a. Persiapan Model

Model Training: Melatih model menggunakan data pelatihan yang tersedia dan menyimpan model yang terlatih. Model Serialization: Menyimpan model yang telah dilatih dalam format yang dapat di-load kembali, seperti menggunakan pickle di Python atau format HDF5 untuk Keras/TensorFlow.

b. Environment Setup

Dependencies: Mengidentifikasi dan menginstal semua dependensi yang diperlukan untuk menjalankan model, termasuk library machine learning (misalnya, TensorFlow, PyTorch), preprocessing tools (misalnya, pandas, scikit-learn), dan runtime (misalnya, Python). Virtual Environments: Menggunakan virtual environments atau container (misalnya, Docker) untuk mengisolasi dan mengelola dependensi.

c. Model Serving

REST APIs: Menyediakan model sebagai layanan web dengan API menggunakan framework seperti Flask, FastAPI, atau Django. Ini memungkinkan aplikasi lain untuk mengirim data dan menerima prediksi melalui permintaan HTTP. Model Server: Menggunakan alat khusus untuk serving model seperti TensorFlow Serving, TorchServe, atau MLflow. Alat ini dirancang untuk mengelola dan mengoptimalkan serving model machine learning. Batch Processing: Untuk prediksi dalam jumlah besar, batch processing dapat digunakan, di mana data dikumpulkan dan diproses sekaligus, misalnya menggunakan Apache Spark.

d. Scalability and Performance

Horizontal Scaling: Menambah lebih banyak instance server untuk menangani beban yang meningkat. Load Balancing: Menggunakan load balancer untuk mendistribusikan lalu lintas ke berbagai instance server secara efisien. Caching: Mengimplementasikan caching untuk menyimpan hasil prediksi yang sering diminta untuk mengurangi beban komputasi.

e. Monitoring and Logging

Model Monitoring: Mengawasi kinerja model di produksi untuk mendeteksi degradasi performa, bias, atau drift data. Alat seperti Prometheus dan Grafana dapat digunakan untuk monitoring. Logging: Menyimpan log prediksi, kesalahan, dan metrik kinerja untuk analisis dan pemecahan masalah. Alat seperti ELK Stack (Elasticsearch, Logstash, Kibana) atau Splunk sering digunakan.

f. Security and Compliance

Data Privacy: Memastikan data pengguna dilindungi dan sesuai dengan regulasi seperti GDPR. Authentication and Authorization: Mengamankan API dengan mekanisme otentikasi dan otorisasi seperti OAuth. Model Security: Melindungi model dari serangan seperti model inversion dan membership inference.

Contoh Deployment dengan Flask dan Docker

Berikut adalah contoh sederhana tentang bagaimana meng-deploy model machine learning menggunakan Flask dan Docker:

```
# Langkah 1: Membuat API dengan Flask
from flask import Flask, request, jsonify
import pickle

# Load model
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict([data['features']])
    return jsonify({'prediction': prediction[0]})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Deployment machine learning mencakup berbagai aspek teknis dan non-teknis, mulai dari persiapan model hingga monitoring di produksi. Dengan pemahaman dan penerapan yang tepat dari konsep-konsep ini, proses deployment dapat dilakukan secara efisien dan aman.

13.7. Aristektur Machine Learning

Pengantar ini bertujuan untuk memberikan gambaran mengenai arsitektur machine learning, yang merupakan struktur dasar yang mendukung pengembangan model machine learning. Arsitektur ini mencakup berbagai komponen, seperti pengumpulan data, pemrosesan data, pelatihan model, dan evaluasi kinerja. Memahami arsitektur machine learning sangat penting karena membantu dalam merancang dan mengimplementasikan solusi yang efektif untuk berbagai masalah. Dengan pengetahuan ini, diharapkan Anda dapat lebih siap dalam menghadapi tantangan dalam pengembangan model machine learning.

13.8. Bagaimana Memilih CPU, GPU, dan TPU

Video ini akan membahas perbedaan antara CPU, GPU, dan TPU, serta konteks penggunaan masing-masing dalam machine learning. CPU, atau unit pemrosesan pusat, berfungsi sebagai otak komputer yang cocok untuk menangani tugas komputasi sehari-hari dan aplikasi machine learning yang lebih sederhana. Di sisi lain, GPU, atau unit pemrosesan grafis, dirancang khusus untuk pemrosesan paralel, menjadikannya pilihan ideal untuk pelatihan model deep learning dengan dataset besar, di mana kecepatan pelatihan dapat meningkat secara signifikan. Terakhir, TPU, atau tensor processing unit, adalah perangkat keras khusus yang dikembangkan oleh Google, dioptimalkan untuk TensorFlow dan memberikan kecepatan serta efisiensi tinggi dalam pelatihan model skala besar. Dengan memahami perbedaan ini, Anda

dapat memilih perangkat yang paling sesuai untuk memastikan proyek machine learning Anda berjalan dengan optimal.

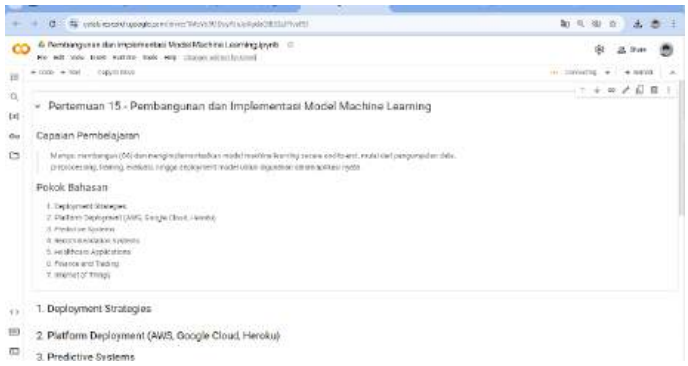
Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran

Bab 14. Pembangunan dan Implementasi Model Machine Learning

Hal-hal yang dibahas pada Bab ini:

1. Pengumpulan dan pemahaman data.
2. Teknik preprocessing dan pembersihan data.
3. Proses training model machine learning.
4. Evaluasi dan tuning model.
5. Deployment model untuk aplikasi nyata.
6. Studi kasus implementasi end-to-end.

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



14.1. Deployment Strategies

14.2. Platform Deployment (AWS, Google Cloud, Heroku)

14.3. Predictive Systems

14.4. Recommendation Systems

14.5. Healthcare Applications

14.6. Finance and Trading

14.7. Internet of Things

14.8. Peran Machine Learning pada Bidang Kesehatan, Keamanan, dan Pendidikan

Dalam video ini, kita akan mengeksplorasi peran machine learning dalam tiga bidang yang sangat penting: kesehatan, keamanan, dan pendidikan. Di bidang kesehatan, machine learning berperan dalam mempercepat diagnosis penyakit, meningkatkan akurasi deteksi, dan mendukung penemuan obat dengan menganalisis data medis dalam skala besar.

Sementara itu, dalam sektor keamanan, machine learning digunakan untuk mendeteksi serangan siber secara real-time, menganalisis pola-pola anomali, serta memperkuat sistem keamanan jaringan agar lebih responsif terhadap ancaman. Di bidang pendidikan, machine learning berkontribusi pada personalisasi pengalaman belajar, membantu menganalisis kemajuan siswa, dan memberikan umpan balik yang lebih efektif untuk meningkatkan proses pembelajaran. Dengan penerapan yang semakin meluas, machine learning berpotensi untuk terus meningkatkan efisiensi dan kualitas dalam ketiga bidang ini.

Scan disi atau klik gambar di samping untuk mengakses konten pembelajaran