

💻 UTS Machine Learning 2023/2024 💻

Daffa Asyqar Ahmad Khalisheka - 1103200034 :

Support Vector Machine Algoirthm

Menggunakan dataset dari website opendatajakarta <https://data.jakarta.go.id/> mengenai kualitas udara di wilayah Provinsi DKI Jakarta dari tahun 2016 - 2021.

The screenshot shows the homepage of the Jakarta Open Data portal. At the top left is the logo 'JAKARTA open data' with the tagline 'Berbagi Data untuk Transparansi'. At the top right are navigation links: Data, Organisasi, Topik, Visualisasi, Infografis, and Tentang. Below the header, a large banner features the text 'Selamat Datang di Portal Data Terbuka Pemerintah Provinsi DKI Jakarta'. To the right of the banner is a 3D illustration of a person interacting with a smartphone displaying various data icons, set against a background of a city skyline and circuit boards. A search bar at the bottom left contains the placeholder 'Cari Data, Tema, Instansi' and a 'Cari Data' button. Below the search bar is a section titled 'Telusuri Berdasarkan Grup / Topik' with five categories: COVID-19, Kependudukan, Kesehatan, Keuangan, and Lingkungan Hidup. To the right of this section is a blue box labeled 'SARAN DATA PUBLIK' containing an image of a person writing. At the bottom of the page is a blue bar with the text 'Statistik Portal Data Terbuka Jakarta'.

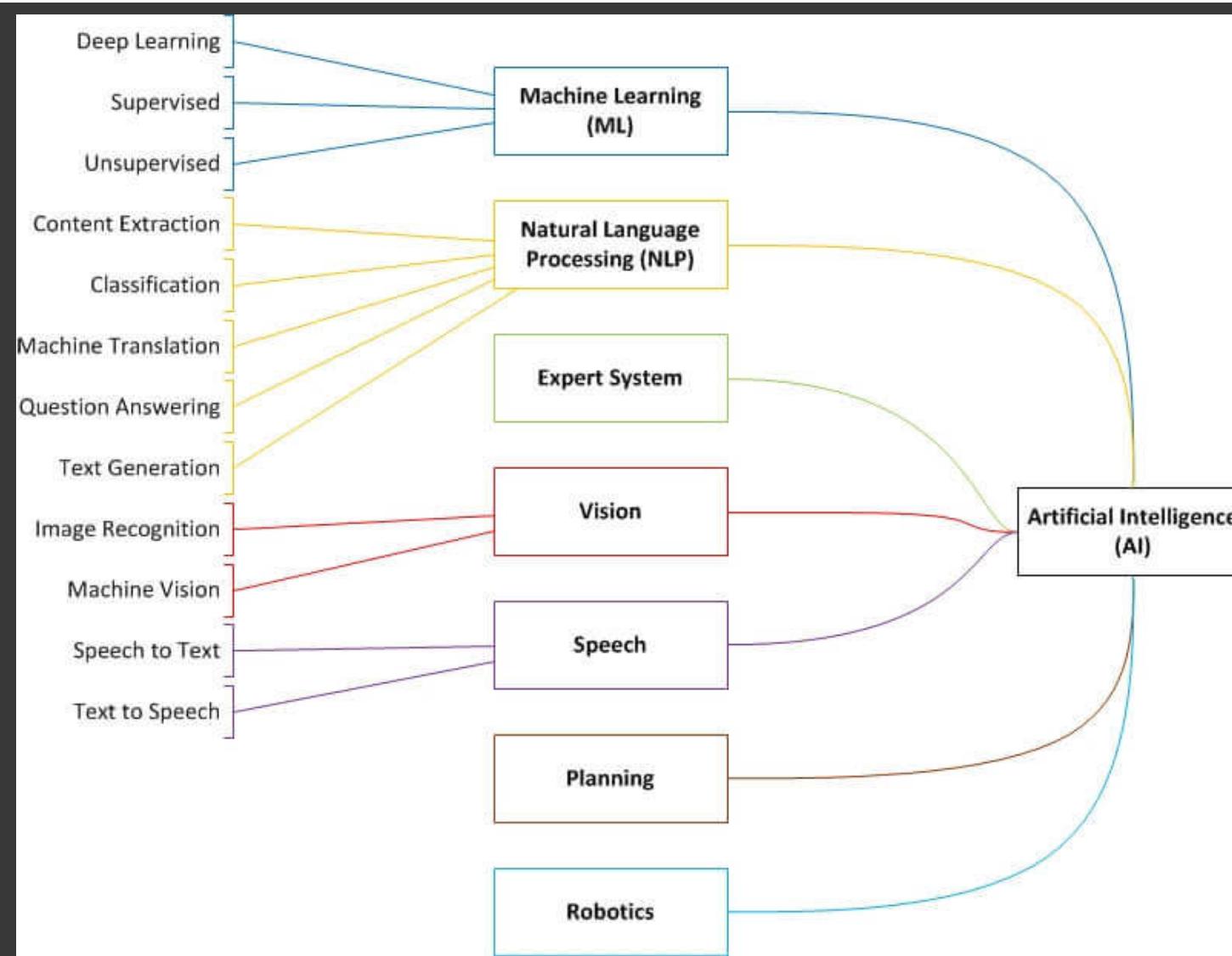
Link dataset:

1. <https://data.jakarta.go.id/dataset/indeks-standar-pencemaran-udara-ispu-tahun-2016>
2. <https://data.jakarta.go.id/dataset/indeks-standar-pencemaran-udara-ispu-tahun-2017>
3. <https://data.jakarta.go.id/dataset/indeks-standar-pencemaran-udara-ispu-tahun-2020>
4. <https://data.jakarta.go.id/dataset/indeks-standar-pencemaran-udara-ispu-tahun-2021>

Penjelasan variabel dari data diatas sebagai berikut :

1. tanggal : Tanggal pengukuran kualitas udara
2. stasiun : Lokasi pengukuran di stasiun
3. pm10 : Partikulat salah satu parameter yang diukur
4. pm25 : Partikulat salah satu parameter yang diukur
5. so2 : Sulfida (dalam bentuk SO₂) salah satu parameter yang diukur
6. co : Carbon Monoksida salah satu parameter yang diukur
7. o₃ : Ozon salah satu parameter yang diukur
8. no₂ : Nitrogen dioksida salah satu parameter yang diukur
9. max : Nilai ukur paling tinggi dari seluruh parameter yang diukur dalam waktu yang sama
10. critical : Parameter yang hasil pengukurannya paling tinggi
11. categori : Kategori hasil perhitungan indeks standar pencemaran udara

▼ Konsep-konsep Machine Learning



Machine learning (ML) adalah cabang dari kecerdasan buatan yang fokus pada pengembangan sistem yang dapat belajar dari data. Tujuan utama dari machine learning adalah memungkinkan komputer untuk belajar dan membuat keputusan atau tugas tanpa diprogram secara eksplisit. Ada beberapa konsep dasar yang perlu dipahami dalam machine learning:

1. Data:

- **Training Data:** Merupakan data yang digunakan untuk melatih model. Data ini harus mencakup contoh-contoh yang representatif dari situasi atau masalah yang ingin dipecahkan.
- **Testing Data:** Data yang digunakan untuk menguji kinerja model setelah dilatih. Data ini seharusnya tidak dikenal oleh model selama proses pelatihan.

2. Model:

- **Algoritma:** Sebuah set prosedur matematis yang digunakan oleh model untuk membuat prediksi atau keputusan berdasarkan data input.
- **Parameter:** Nilai-nilai yang disesuaikan oleh model selama proses pelatihan untuk meningkatkan kemampuannya membuat prediksi yang akurat.

3. Feature (Fitur):

- **Variable Input:** Data yang digunakan sebagai input untuk model. Fitur-fitur ini dapat menjadi atribut atau karakteristik dari data yang diukur atau diamati.

4. Label atau Target:

- **Variable Output:** Hasil yang diharapkan dari model. Dalam pembelajaran terawasi, ini sering disebut sebagai label atau target.

5. Pelatihan (Training):

- **Proses Pengoptimalan:** Model diperbarui atau disesuaikan menggunakan algoritma pelatihan dengan menggunakan data pelatihan. Tujuan utama adalah mengurangi kesalahan model terhadap data pelatihan dan meningkatkan kinerjanya terhadap data yang tidak dikenal.

6. Validasi dan Pengujian:

- **Validasi:** Penggunaan subset data yang tidak digunakan selama pelatihan untuk mengevaluasi kinerja model dan mencegah overfitting.
- **Pengujian (Testing):** Penggunaan data yang sepenuhnya tidak dikenal oleh model untuk mengukur sejauh mana model dapat membuat prediksi yang akurat.

7. Supervised, Unsupervised, dan Reinforcement Learning:

- **Supervised Learning:** Model belajar dari pasangan input-output yang sudah diberi label.
- **Unsupervised Learning:** Model belajar dari data yang tidak diberi label dan mencoba menemukan pola atau struktur di dalamnya.

- **Reinforcement Learning:** Model belajar dari interaksi dengan lingkungannya, menerima umpan balik berdasarkan tindakan yang diambil.

8. Overfitting dan Underfitting:

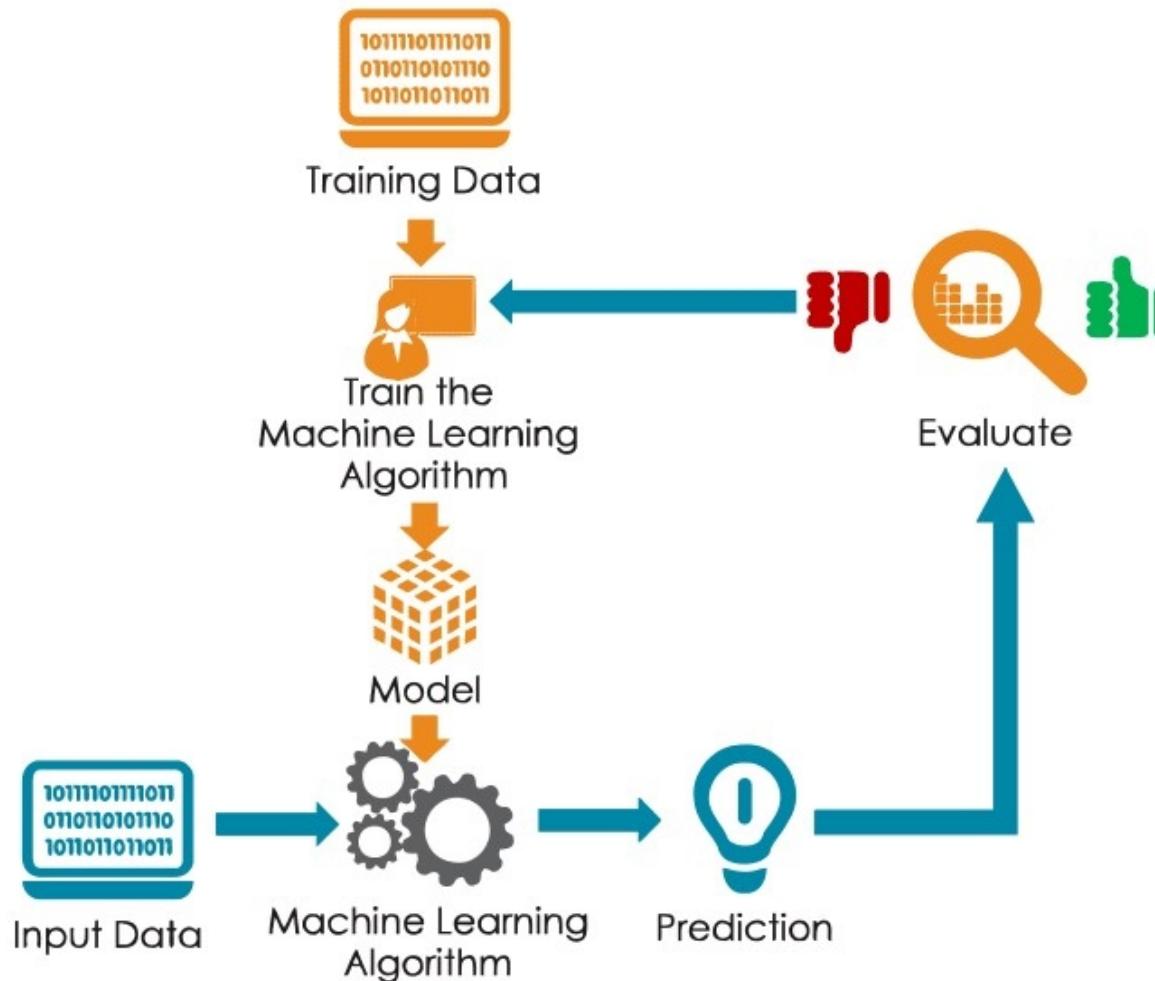
- **Overfitting:** Model terlalu rumit dan "memorizing" data pelatihan sehingga tidak dapat umum pada data baru.
- **Underfitting:** Model terlalu sederhana sehingga gagal menangkap pola dalam data pelatihan dan tidak dapat melakukan prediksi yang baik pada data baru.

Konsep-konsep ini mencakup dasar-dasar machine learning, dan ada berbagai algoritma dan teknik yang dapat diterapkan berdasarkan jenis tugas dan jenis data yang digunakan. Machine learning telah menjadi alat yang sangat penting dalam berbagai aplikasi, seperti pengenalan wajah, penerjemahan bahasa, analisis data besar, dan banyak lagi.

Berikut adalah daftar tools machine learning yang umumnya digunakan oleh para ML engineer:

- TensorFlow
- Auto-WEKA
- KNIME
- Google Cloud AutoML
- BigML
- Scikit-learn
- Azure Machine Learning Studio
- Amazon Machine Learning (AML)
- Accord.NET

Cara Kerja Machine Learning



Konsep dasar cara kerja Machine Learning dapat diuraikan sebagai berikut:

1. **Data Gathering:**

- Machine Learning memerlukan data sebagai bahan bakar untuk pembelajaran. Data ini dapat berupa berbagai jenis informasi tergantung pada jenis masalah yang ingin dipecahkan.

2. Data Preprocessing:

- Data yang dikumpulkan seringkali memerlukan pembersihan dan penyiapan sebelum dapat digunakan untuk pelatihan model. Proses ini melibatkan pemrosesan data untuk menghilangkan kecacatan, mengisi nilai yang hilang, atau mengonversi data ke dalam format yang dapat digunakan.

3. Feature Extraction:

- Fitur-fitur atau atribut dari data yang relevan untuk masalah harus diekstraksi. Ini melibatkan pemilihan informasi yang paling penting untuk melatih model.

4. Model Selection:

- Pemilihan model atau algoritma ML yang sesuai berdasarkan jenis masalah yang dihadapi. Ini bisa mencakup model-model seperti regresi, pohon keputusan, atau jaringan saraf tiruan.

5. Training the Model:

- Model ML dilatih dengan memberikan data latihan yang memungkinkan algoritma untuk menyesuaikan parameter internalnya. Selama pelatihan, model mencoba untuk memahami pola dan hubungan dalam data.

6. Evaluation:

- Setelah pelatihan, model dievaluasi menggunakan data yang tidak digunakan sebelumnya (data uji) untuk mengukur kinerjanya. Metrik evaluasi dapat berbeda-beda tergantung pada jenis masalah (misalnya, akurasi untuk klasifikasi, MSE untuk regresi).

7. Fine-Tuning:

- Berdasarkan hasil evaluasi, model mungkin perlu disesuaikan (fine-tuning) untuk meningkatkan kinerjanya. Ini melibatkan penyesuaian parameter model atau mengubah konfigurasi algoritma.

8. Prediction/Inference:

- Setelah model dianggap memadai, ia dapat digunakan untuk membuat prediksi atau inferensi pada data baru yang tidak pernah dilihat sebelumnya.

9. Deployment:

- Model yang dilatih dan dievaluasi dapat diimplementasikan ke dalam lingkungan produksi, di mana ia dapat digunakan untuk memproses data secara real-time atau batch.

10. Monitoring dan Maintenance:

- Model perlu dipantau secara berkala untuk memastikan bahwa kinerjanya tetap optimal. Jika ada perubahan dalam data atau kebutuhan bisnis, model



Dalam keseluruhan, konsep utama di balik Machine Learning adalah kemampuan model untuk belajar dari data, memahami pola, dan membuat prediksi atau keputusan tanpa instruksi eksplisit. Proses ini menciptakan suatu siklus iteratif di mana model terus-menerus diperbarui dan disempurnakan seiring berjalannya waktu.

▼ Machine Learning dengan Algoritma Support Vector Machine

Machine learning dengan algoritma Support Vector Machine (SVM) di Google Colab melibatkan penggunaan SVM untuk menyelesaikan masalah klasifikasi atau regresi. SVM adalah algoritma pembelajaran yang dapat digunakan untuk memisahkan data ke dalam kategori atau untuk menemukan hubungan dalam data. Berikut adalah langkah-langkah umum untuk menggunakan SVM di Google Colab:

1. Persiapan Data:

- Import library yang diperlukan, seperti NumPy, Pandas, dan scikit-learn, di dalam notebook Colab.
- Load dataset atau persiapkan data yang akan digunakan untuk pelatihan dan pengujian model.

2. Preprocessing Data:

- Lakukan preprocessing pada data, seperti normalisasi atau standarisasi, untuk memastikan bahwa SVM berperforma dengan baik.

3. Pembagian Data:

- Bagi dataset menjadi set pelatihan dan pengujian. Set pelatihan digunakan untuk melatih model, sementara set pengujian digunakan untuk mengukur kinerja model.

4. Import dan Inisialisasi SVM:

- Import SVM dari scikit-learn, misalnya, `from sklearn.svm import SVC`.
- Inisialisasi model SVM dengan parameter yang sesuai, seperti jenis kernel (linear, polinomial, atau RBF) dan parameter C.

5. Pelatihan Model:

- Latih model SVM menggunakan data pelatihan dengan memanggil metode `.fit()` pada objek SVM.

6. Evaluasi Model:

- Gunakan set pengujian untuk mengukur kinerja model, seperti akurasi, presisi, dan recall.

7. Visualisasi Hasil (Opsional):

- Jika diperlukan, visualisasikan hasil prediksi model atau karakteristik pemisahan SVM.

8. Optimasi dan Penyesuaian Model:

- Jika diperlukan, lakukan optimasi dan penyesuaian model SVM dengan mencoba parameter yang berbeda atau menggunakan teknik penalaan (regularization).

9. Penyimpanan Model (Opsional):

- Simpan model SVM yang telah dilatih untuk digunakan di masa depan.

10. Analisis dan Interpretasi:

- Analisis hasil model dan interpretasi keputusan yang dihasilkan oleh SVM.

Google Colab memberikan lingkungan komputasi yang kuat dengan akses ke GPU atau TPU, yang dapat meningkatkan kecepatan pelatihan model, terutama untuk dataset yang lebih besar. Selain itu, Colab memudahkan untuk berbagi dan menyimpan notebook secara online.

Prompt chatGPT:

Buatlah sebuah kode Python menggunakan library Google Colab untuk mengakses Google Drive. Gunakan fungsi dari library tersebut untuk melakukan mounting (pasang) Google Drive pada direktori '[/content/drive](#)'. Pastikan untuk mengimpor library yang diperlukan.

Penjelasan Kode:

```
# Import library Google Colab untuk mengakses Google Drive
from google.colab import drive
```

Baris ini mengimpor modul `drive` dari pustaka Google Colab. Modul ini menyediakan fungsionalitas untuk mengakses dan mengelola Google Drive dari lingkungan Colab.

```
# Mount (pasang) Google Drive pada direktori /content/drive
drive.mount('/content/drive')
```

Baris ini menggunakan fungsi `mount` dari modul `drive` untuk mengaitkan atau "mount" Google Drive ke dalam direktori yang ditentukan, dalam hal ini, '[/content/drive](#)'. Ini memungkinkan akses ke berkas dan direktori di Google Drive dari dalam lingkungan Colab. Setelah baris ini dieksekusi, Google Drive akan tersedia untuk digunakan di bawah direktori '[/content/drive](#)' dalam notebook Colab.

```
1 # Import library Google Colab untuk mengakses Google Drive
2 from google.colab import drive
3
4 # Mount (pasang) Google Drive pada direktori /content/drive
5 drive.mount('/content/drive')
```

Mounted at /content/drive

Catatan/Penjelasan dari Output di atas:

Output tersebut memberikan informasi bahwa Google Drive telah berhasil dimount pada direktori '[/content/drive](#)'. Pada contoh di atas, pesan "Drive already mounted at [/content/drive](#)" mengindikasikan bahwa Google Drive sudah berhasil dimount sebelumnya. Pesan "to attempt to forcibly remount, call drive.mount(\"[/content/drive](#)\", force_remount=True)" memberi tahu bahwa jika Anda ingin memaksa remount Google Drive, Anda dapat menggunakan opsi `force_remount=True` saat memanggil fungsi `drive.mount`. Ini mungkin diperlukan jika terdapat perubahan ke dalam Google Drive dari luar Colab yang belum terdeteksi.

Dengan demikian, pesan tersebut memberikan panduan tambahan tentang cara melakukan remount Google Drive jika diperlukan.

Prompt chatGPT:

Berikan kode Python untuk mengimpor library pandas dengan alias pd, library seaborn dengan alias sns, library matplotlib.pyplot dengan alias plt, dan library numpy dengan alias np. Gunakan komentar untuk memberikan penjelasan singkat tentang setiap pengimporan.

Penjelasan Kode:

1. Mengimpor library pandas dengan alias pd:

```
import pandas as pd
```

Ini adalah baris kode untuk mengimpor library pandas. Pandas adalah library yang sangat populer untuk manipulasi dan analisis data dalam Python. Dengan memberikan alias `pd`, kita dapat menggunakan singkatan ini untuk merujuk ke library pandas di seluruh kode kita.

2. Mengimpor library seaborn dengan alias sns:

```
import seaborn as sns
```

Ini adalah baris kode untuk mengimpor library seaborn. Seaborn adalah library untuk membuat visualisasi data statistik yang indah dan informatif. Dengan memberikan alias `sns`, kita membuat referensi singkat untuk mengakses fungsi dan objek seaborn.

3. Mengimpor library matplotlib.pyplot dengan alias plt:

```
import matplotlib.pyplot as plt
```

Baris ini mengimpor modul `pyplot` dari library `matplotlib` dengan memberikan alias `plt`. `matplotlib.pyplot` menyediakan fungsi-fungsi untuk membuat plot grafik seperti scatter plot, line plot, dan lainnya. Menggunakan alias `plt` mempermudah penggunaan fungsi-fungsi ini.

4. Mengimpor library numpy dengan alias np:

```
import numpy as np
```

Ini adalah baris kode untuk mengimpor library numpy. NumPy adalah library yang sangat kuat untuk komputasi numerik dalam Python. Dengan memberikan alias `np`, kita membuat referensi singkat untuk menggunakan fungsi-fungsi dan objek-objek NumPy tanpa mengetik

nama lengkapnya setiap kali.

```
1 # Mengimpor library pandas dengan alias pd
2 import pandas as pd
3
4 # Mengimpor library seaborn dengan alias sns
5 import seaborn as sns
6
7 # Mengimpor library matplotlib.pyplot dengan alias plt
8 import matplotlib.pyplot as plt
9
10 # Mengimpor library numpy dengan alias np
11 import numpy as np
12
```

Catatan/Penjelasan dari Output di atas:

Baris kode tersebut tidak menghasilkan output langsung atau memberikan insight tertentu saat dijalankan. Sebaliknya, itu hanya memuat ke dalam lingkungan Python empat library yang umum digunakan dalam analisis data dan visualisasi:

1. **pandas (pd)**: Library ini digunakan untuk manipulasi dan analisis data. Dengan memberikan alias `pd`, pengguna dapat mengakses fungsi dan objek pandas dengan menggunakan `pd` sebagai singkatan.
2. **seaborn (sns)**: Seaborn adalah library yang memungkinkan pembuatan visualisasi data statistik dengan mudah. Alias `sns` memudahkan penggunaan fungsi dan objek seaborn.
3. **matplotlib.pyplot (plt)**: Library matplotlib.pyplot menyediakan fungsi untuk membuat berbagai jenis plot, seperti scatter plot, line plot, dan histogram. Alias `plt` membuat referensi singkat untuk memanggil fungsi-fungsi ini.
4. **numpy (np)**: NumPy adalah library yang digunakan untuk komputasi numerik. Memberikan alias `np` memungkinkan pengguna untuk menggunakan fungsi dan objek NumPy dengan sintaks yang lebih ringkas.

Pengimporan ini umum digunakan dalam proyek analisis data di Python, dan dengan memberikan alias, kode menjadi lebih singkat dan mudah dibaca.

Penjelasan lain:

Mengimpor library merupakan langkah penting sebelum melakukan machine learning dengan algoritma Support Vector Machine (SVM) di Google Colab, atau pada umumnya dalam lingkungan pengembangan Python. Berikut adalah penjelasan singkat mengapa mengimpor library tersebut penting:

1. Pandas (pd):

- **Alasan:** Pandas menyediakan struktur data yang kuat, seperti DataFrame, yang sangat berguna untuk menyimpan dan mengelola data tabular. Dalam machine learning, ini memungkinkan pengelolaan dataset dengan mudah, termasuk pembersihan data, pengindeksan, dan pemilihan fitur.

2. Seaborn (sns) dan Matplotlib (plt):

- **Alasan:** Visualisasi data sangat penting dalam tahap eksplorasi data sebelum menerapkan algoritma machine learning. Seaborn dan Matplotlib menyediakan berbagai jenis plot dan grafik yang membantu memahami distribusi data, pola, dan korelasi. Ini membantu pengguna untuk membuat keputusan yang lebih baik tentang pemrosesan data dan pemilihan fitur.

3. NumPy (np):

- **Alasan:** NumPy menyediakan array multidimensional yang efisien dan berbagai fungsi matematika yang mendukung operasi vektor dan matriks. Algoritma SVM, seperti yang diimplementasikan dalam library seperti scikit-learn, seringkali memanfaatkan manipulasi array NumPy untuk performa yang lebih baik dan efisiensi komputasi.

Selain itu, Google Colab menyediakan lingkungan penelitian dan pengembangan yang siap pakai, tetapi kita perlu mengimpor library yang diperlukan karena tidak semua library tersedia secara default. Dengan mengimpor library ini, pengguna dapat memanfaatkan fungsionalitas dan kemampuan yang diperlukan untuk memproses data, melatih model, dan mengevaluasi hasil dengan efisien.

EDA

EDA (Exploratory Data Analysis) dalam machine learning merujuk pada proses pemeriksaan dan pemahaman awal terhadap dataset yang akan digunakan. Tujuan dari EDA adalah untuk mendapatkan wawasan yang mendalam tentang struktur dan karakteristik data sebelum menerapkan model machine learning. Berikut adalah beberapa poin kunci tentang EDA:

1. Pemahaman Data:

- EDA membantu untuk memahami struktur dataset, termasuk jumlah dan tipe variabel, serta nilai-nilai yang ada dalam setiap variabel.

2. Pendekripsi Anomali dan Outlier:

- EDA membantu dalam mendeteksi anomali atau outlier yang mungkin memengaruhi hasil model. Pemahaman ini penting untuk membersihkan data sebelum melatih model.

3. Distribusi Variabel:

- EDA memungkinkan pengamatan terhadap distribusi variabel, membantu pengguna untuk memilih transformasi atau normalisasi yang tepat untuk data.

4. Korelasi antar Variabel:

- Analisis korelasi membantu mengidentifikasi hubungan antara variabel, yang dapat memberikan wawasan tentang fitur yang mungkin memiliki dampak besar pada prediksi.

5. Visualisasi Data:

- EDA melibatkan penggunaan visualisasi data seperti histogram, box plot, scatter plot, dan lainnya untuk mengidentifikasi pola, tren, atau kelainan dalam data.

6. Pemilihan Fitur:

- EDA membantu dalam pemilihan fitur dengan memahami hubungan antar fitur dan dampaknya terhadap target. Fitur yang kurang relevan atau saling terkait dapat diidentifikasi.

7. Persiapan Data:

- Melalui EDA, mungkin diperlukan langkah-langkah persiapan data seperti penanganan missing values, normalisasi, atau transformasi variabel.

EDA memberikan landasan yang kuat sebelum membangun model machine learning, membantu menghindari keputusan yang salah atau pengaturan model yang tidak tepat. Dengan memahami karakteristik data, praktisi machine learning dapat membuat keputusan yang lebih cerdas dan mendapatkan hasil yang lebih baik dari model yang dikembangkan.

Prompt chatGPT:

Buatlah sebuah program Python menggunakan library pandas untuk membaca file Excel yang memiliki beberapa sheet. Tentukan nama-nama sheet yang ingin dibaca, berikan path file Excel, dan gunakan fungsi `pd.read_excel` untuk membaca semua sheet ke dalam sebuah dictionary DataFrame.

Setelah membaca setiap sheet, gunakan loop untuk menampilkan beberapa baris pertama dari setiap DataFrame di dalam dictionary. Terakhir, gabungkan semua DataFrame menjadi satu DataFrame menggunakan fungsi `pd.concat`.

Penjelasan Kode:

```
# Tentukan nama-nama sheet yang ingin dibaca  
sheet_names = ['DKI1', 'DKI2', 'DKI3', 'DKI4', 'DKI5']
```

Pada baris ini, kita membuat sebuah list yang berisi nama-nama sheet yang ingin dibaca dari file Excel.

```
# Berikan path file  
file_path = '/content/drive/MyDrive/Dataset/DATA_ISPU - Impute (1).xlsx'
```

Baris ini digunakan untuk menentukan path atau lokasi file Excel yang akan dibaca. Pastikan path file sudah sesuai dengan lokasi sebenarnya file tersebut.

```
# Baca semua sheet yang telah ditentukan ke dalam sebuah dictionary DataFrame  
all_data = pd.read_excel(file_path, sheet_name=sheet_names)
```

Pada baris ini, kita menggunakan fungsi `pd.read_excel` dari library pandas untuk membaca semua sheet yang telah ditentukan dalam list `sheet_names` ke dalam sebuah dictionary DataFrame yang disimpan dalam variabel `all_data`.

```
# Loop melalui setiap sheet dan DataFrame pada dictionary  
for sheet_name, df in all_data.items():  
    print(f"Beberapa baris pertama dari {sheet_name}:")  
    print(df.head())  
    print("\n")
```

Baris ini menginisialisasi sebuah loop yang akan iterasi melalui setiap item (pasangan key-value) dalam dictionary `all_data`. Di dalam loop, kita mencetak beberapa baris pertama dari setiap DataFrame bersama dengan nama sheet-nya.

```
# Gabungkan semua DataFrames menjadi satu DataFrame menggunakan pd.concat
dataudara = pd.concat(all_data.values(), ignore_index=True)
```

Pada baris ini, kita menggunakan fungsi `pd.concat` untuk menggabungkan semua DataFrame yang ada dalam dictionary `all_data` menjadi satu DataFrame tunggal yang disimpan dalam variabel `dataudara`. Parameter `ignore_index=True` digunakan untuk mereset index DataFrame hasil penggabungan.

```
1 # Tentukan nama-nama sheet yang ingin dibaca
2 sheet_names = ['DKI1', 'DKI2', 'DKI3', 'DKI4', 'DKI5']
3
4 # Berikan path file
5 file_path = '/content/drive/MyDrive/Dataset/DATA ISPU - Impute (1).xlsx'
6
7 # Baca semua sheet yang telah ditentukan ke dalam sebuah dictionary DataFrame
8 all_data = pd.read_excel(file_path, sheet_name=sheet_names)
9
10 # Loop melalui setiap sheet dan DataFrame pada dictionary
11 for sheet_name, df in all_data.items():
12     print(f"Beberapa baris pertama dari {sheet_name}:")
13     print(df.head())
14     print("\n")
15
16 # Gabungkan semua DataFrames menjadi satu DataFrame menggunakan pd.concat
17 dataudara = pd.concat(all_data.values(), ignore_index=True)
18
```

2	2017-01-03	DKI3 (Jagakarsa)	51.0	30.0	7.0	56.0	3.0	51.0	PM10
3	2017-01-04	DKI3 (Jagakarsa)	45.0	33.0	7.0	63.0	7.0	63.0	03
4	2017-01-05	DKI3 (Jagakarsa)	54.0	30.0	16.0	68.0	9.0	68.0	03

Kategori

0	Sedang
1	Baik
2	Sedang
3	Sedang
4	Sedang

Beberapa baris pertama dari DKI4:

	Tanggal	Wilayah	PM10	SO2	CO	O3	NO2	Max	Critical	\
0	2017-01-01	DKI4 (Lubang Buaya)	69.0	27.0	23.0	54.0	5.0	69.0	PM10	
1	2017-01-02	DKI4 (Lubang Buaya)	35.0	26.0	7.0	36.0	4.0	36.0	03	
2	2017-01-03	DKI4 (Lubang Buaya)	55.0	26.0	7.0	42.6	5.0	55.0	PM10	
3	2017-01-04	DKI4 (Lubang Buaya)	51.0	26.0	13.0	87.0	7.0	87.0	03	
4	2017-01-05	DKI4 (Lubang Buaya)	48.0	27.0	16.0	67.0	9.0	67.0	03	

Kategori

0	Sedang
1	Baik
2	Sedang
3	Sedang
4	Sedang

Beberapa baris pertama dari DKI5:

	Tanggal	Wilayah	PM10	SO2	CO	O3	NO2	\
0	2017-01-01	DKI5 (Kebon Jeruk)	Jakarta Barat	28.6	17.0	38.0	54.0	5.0
1	2017-01-02	DKI5 (Kebon Jeruk)	Jakarta Barat	34.0	17.0	33.0	44.0	3.0
2	2017-01-03	DKI5 (Kebon Jeruk)	Jakarta Barat	38.2	16.0	31.0	46.0	4.0
3	2017-01-04	DKI5 (Kebon Jeruk)	Jakarta Barat	35.0	16.0	35.0	48.0	5.0
4	2017-01-05	DKI5 (Kebon Jeruk)	Jakarta Barat	25.8	17.0	35.0	35.0	7.0

Max Critical Kategori

0	54.0	03	Sedang
1	44.0	03	Baik
2	46.0	03	Baik
3	48.0	03	Baik
4	35.0	CO	Baik

Catatan/Penjelasan dari Output di atas:

Dari output tersebut, terlihat beberapa baris pertama dari beberapa dataset yang mewakili tingkat polusi udara (PM10, SO2, CO, O3, NO2) di beberapa wilayah (DKI1, DKI2, DKI3, DKI4, DKI5) di Jakarta pada berbagai tanggal. Berikut adalah beberapa insight dari data tersebut:

1. Variabilitas Wilayah:

- Terdapat variasi dalam tingkat polusi antar wilayah DKI. Sebagai contoh, DKI1 (Bunderan HI) dan DKI2 (Kelapa Gading) mungkin memiliki tingkat polusi yang berbeda pada tanggal yang sama.

2. Jenis Polutan yang Dominan:

- Setiap wilayah memiliki kategori polutan kritis yang berbeda pada beberapa tanggal tertentu. Misalnya, DKI1 memiliki kategori kritis "PM10", sementara DKI2 memiliki kategori kritis "O3" pada beberapa hari.

3. Tren Harian:

- Beberapa wilayah menunjukkan tren harian tertentu dalam tingkat polusi. Sebagai contoh, DKI3 (Jagakarsa) dan DKI4 (Lubang Buaya) memiliki kategori kritis "O3" pada beberapa hari, mungkin menunjukkan kecenderungan peningkatan polutan ini pada hari tertentu.

4. Korelasi antara Variabel:

- Insight tentang korelasi antara berbagai jenis polutan mungkin ditemukan dengan membandingkan nilai-nilai variabel PM10, SO2, CO, O3, dan NO2 pada tanggal-tanggal tertentu.

5. Pemantauan Kategori Kritis:

- Data mencakup kategori kritis (seperti "Sedang" atau "Baik") untuk memantau tingkat polusi. Pemantauan ini dapat membantu dalam pengambilan keputusan terkait dengan kesehatan lingkungan.

6. Perbandingan Kategori:

- Pengguna dapat membandingkan kategori polusi antar wilayah untuk menentukan wilayah yang mungkin membutuhkan tindakan lebih lanjut dalam pengendalian polusi.

Insight dari data ini dapat membantu para peneliti, pengambil keputusan, dan praktisi lingkungan untuk memahami pola polusi udara di Jakarta, yang pada gilirannya dapat membantu dalam pengelolaan dan mitigasi dampaknya terhadap kesehatan dan lingkungan.

Prompt chatGPT:

Berikan kode untuk menampilkan informasi tentang objek 'dataudara'.

Penjelasan Kode:

Kode di atas menggunakan metode `info()` pada objek `dataudara` untuk menampilkan informasi tentang objek tersebut. Berikut adalah penjelasan lengkap untuk setiap baris kode:

```
# Ini adalah komentar dalam kode Python. Tanda '#' menandakan bahwa baris ini adalah komentar dan tidak dieksekusi.  
# Komentar digunakan untuk memberikan penjelasan atau catatan tambahan dalam kode.  
  
# Memanggil metode 'info()' pada objek 'dataudara'.  
# Metode 'info()' umumnya digunakan untuk menampilkan informasi terperinci tentang suatu objek, seperti tipe data, jumlah entri, dan penggunaan memori.  
dataudara.info()
```

Dengan menjalankan kode di atas, informasi terkait objek 'dataudara' akan ditampilkan di konsol atau lingkungan eksekusi Python. Informasi tersebut biasanya mencakup daftar kolom, tipe data setiap kolom, jumlah entri, dan penggunaan memori.

```
1 # Ini adalah kode untuk menampilkan informasi tentang objek 'data'  
2 dataudara.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 8825 entries, 0 to 8824  
Data columns (total 10 columns):  
 #   Column   Non-Null Count  Dtype     
---  --      --          --  
 0   Tanggal  8825 non-null   datetime64[ns]  
 1   Wilayah  8825 non-null   object  
 2   PM10     8825 non-null   float64  
 3   SO2      8825 non-null   float64  
 4   CO       8825 non-null   float64  
 5   O3       8825 non-null   float64  
 6   NO2     8825 non-null   float64  
 7   Max      8825 non-null   float64  
 8   Critical 8716 non-null   object  
 9   Kategori 8813 non-null   object
```

```
dtypes: datetime64[ns](1), float64(6), object(3)
memory usage: 689.6+ KB
```

Catatan/Penjelasan dari Output di atas:

Output tersebut menunjukkan informasi terkait DataFrame 'dataudara'. Berikut adalah insight dari output tersebut:

1. Informasi Umum:

- Tipe objek: `pandas.core.frame.DataFrame`.
- Jumlah baris: 8825 (dengan indeks berjalan dari 0 hingga 8824).
- Jumlah kolom: 10.

2. Kolom dan Tipe Data:

- Kolom 1: 'Tanggal' memiliki tipe data `datetime64[ns]`.
- Kolom 2: 'Wilayah' memiliki tipe data `object`.
- Kolom 3-8: 'PM10', 'SO2', 'CO', 'O3', 'NO2', 'Max' memiliki tipe data `float64`.
- Kolom 9: 'Critical' memiliki tipe data `object`.
- Kolom 10: 'Kategori' memiliki tipe data `object`.

3. Non-Null Count:

- Semua kolom memiliki 8825 entri, kecuali 'Critical' dan 'Kategori' yang memiliki beberapa nilai null.

4. Penggunaan Memori:

- Penggunaan memori oleh DataFrame ini sekitar 689.6 KB.

5. Keterangan Tambahan:

- 'Critical' dan 'Kategori' adalah kolom dengan beberapa nilai null. Dalam kolom 'Critical', terdapat 8716 entri non-null, sedangkan 'Kategori' memiliki 8813 entri non-null.

6. Rentang Tanggal:

- DataFrame ini mencakup data yang melibatkan tanggal.

Insight ini memberikan gambaran tentang struktur dan isian dari DataFrame 'dataudara', termasuk jenis data dalam setiap kolom, keberadaan nilai null, dan informasi umum lainnya yang dapat membantu dalam analisis data dan pengolahan lebih lanjut.

Prompt chatGPT:

Tolong berikan kode Python yang tepat untuk mengubah tipe data kolom 'Kategori' dan 'Critical' pada DataFrame 'dataudara' menjadi tipe data kategori, dan kemudian menampilkan informasi DataFrame setelah perubahan tipe data tersebut.

Penjelasan Kode:

```
# Mengubah kolom 'Kategori' dan 'Critical' menjadi tipe data kategori  
dataudara['Kategori'] = dataudara['Kategori'].astype('category')
```

Pada baris pertama, kita menggunakan metode `astype` untuk mengubah tipe data kolom 'Kategori' dalam DataFrame `dataudara` menjadi tipe data kategori. Tipe data kategori digunakan untuk kolom yang memiliki sejumlah nilai yang terbatas dan sering kali berulang, seperti kategori atau label. Penggunaan tipe data kategori dapat membantu menghemat memori dan meningkatkan efisiensi operasi terkait kategori.

```
# Mengubah kolom 'Critical' menjadi tipe data kategori  
dataudara['Critical'] = dataudara['Critical'].astype('category')
```

Pada baris kedua, kita melakukan hal yang sama dengan baris pertama, tetapi kali ini kita fokus pada kolom 'Critical'. Kami mengonversi tipe data kolom ini menjadi kategori dengan menggunakan metode `astype`.

```
# Menampilkan informasi DataFrame setelah perubahan tipe data  
print(dataudara.info())
```

Baris terakhir digunakan untuk menampilkan informasi tentang DataFrame `dataudara` setelah perubahan tipe data. Fungsi `info()` memberikan ringkasan tentang DataFrame, termasuk informasi tentang tipe data setiap kolom, jumlah nilai non-null, dan penggunaan memori. Dengan mencetak informasi ini, kita dapat memverifikasi apakah perubahan tipe data telah diterapkan dengan sukses dan memahami karakteristik DataFrame yang telah diubah.

```
1 # Mengubah kolom 'Kategori' dan 'Critical' menjadi tipe data kategori  
2 dataudara['Kategori'] = dataudara['Kategori'].astype('category')  
3  
4 # Mengubah kolom 'Critical' menjadi tipe data kategori  
5 dataudara['Critical'] = dataudara['Critical'].astype('category')
```

```
6
7 # Menampilkan informasi DataFrame setelah perubahan tipe data
8 print(dataudara.info())
9

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8825 entries, 0 to 8824
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          ----- 
 0   Tanggal     8825 non-null   datetime64[ns]
 1   Wilayah    8825 non-null   object  
 2   PM10        8825 non-null   float64 
 3   SO2         8825 non-null   float64 
 4   CO          8825 non-null   float64 
 5   O3          8825 non-null   float64 
 6   NO2        8825 non-null   float64 
 7   Max         8825 non-null   float64 
 8   Critical    8716 non-null   category
 9   Kategori   8813 non-null   category
dtypes: category(2), datetime64[ns](1), float64(6), object(1)
memory usage: 569.6+ KB
None
```

Catatan/Penjelasan dari Output di atas:

Output tersebut memberikan informasi tentang DataFrame `dataudara` setelah dilakukan beberapa operasi, termasuk perubahan tipe data kolom 'Critical' dan 'Kategori'. Berikut adalah insight dari output tersebut:

1. Jumlah Baris dan Kolom:

- DataFrame `dataudara` memiliki total 8825 baris dan 10 kolom.

2. Tipe Data Kolom:

- Kolom 'Tanggal' memiliki tipe data `datetime64[ns]`, yang sesuai dengan data tanggal.
- Kolom 'Wilayah' memiliki tipe data `object`, menunjukkan data dalam bentuk teks atau string.
- Kolom 'PM10', 'SO2', 'CO', 'O3', 'NO2', 'Max' memiliki tipe data `float64`, menunjukkan data dalam bentuk angka desimal.
- Kolom 'Critical' dan 'Kategori' telah diubah menjadi tipe data `category`.

3. Jumlah Nilai Non-Null:

- Setiap kolom memiliki 8825 nilai non-null, kecuali kolom 'Critical' dan 'Kategori' yang memiliki beberapa nilai null.

4. Memory Usage:

- DataFrame menggunakan sekitar 569.6 KB memori. Informasi ini dapat membantu dalam memantau penggunaan memori dan mengoptimalkan performa, terutama ketika bekerja dengan dataset yang lebih besar.

5. Data Kategorikal:

- Kolom 'Critical' dan 'Kategori' diidentifikasi sebagai tipe data category, yang sesuai dengan perubahan yang telah kita terapkan sebelumnya. Tipe data ini cocok digunakan untuk kolom yang memiliki sejumlah nilai terbatas dan berulang.

6. Missing Values:

- Kolom 'Critical' memiliki 8716 nilai non-null dari total 8825 baris, sehingga terdapat beberapa nilai null.
- Kolom 'Kategori' memiliki 8813 nilai non-null dari total 8825 baris, juga terdapat beberapa nilai null.

Insight ini membantu kita memahami struktur dan karakteristik DataFrame, termasuk tipe data, jumlah nilai non-null, dan penggunaan memori setelah perubahan yang telah diterapkan. Selain itu, informasi ini dapat menjadi dasar untuk langkah-langkah selanjutnya dalam pemrosesan dan analisis data.

Prompt chatGPT:

Diberikan DataFrame 'dataudara' dengan kolom 'Kategori' dan 'Critical'. Tulis kode untuk mengambil nilai unik dari kolom 'Kategori' dan 'Critical', dan menampilkannya.

Penjelasan Kode:

```
# Mengambil nilai unik dari kolom 'Kategori'  
unique_kategori_values = dataudara['Kategori'].unique()
```

Kode ini mengambil nilai unik dari kolom 'Kategori' dalam DataFrame 'dataudara' menggunakan metode `unique()`. Hasilnya disimpan dalam variabel `unique_kategori_values`.

```
# Mengambil nilai unik dari kolom 'Critical'  
unique_kategori_values1 = dataudara['Critical'].unique()
```

Baris ini mirip dengan yang sebelumnya, tetapi kali ini mengambil nilai unik dari kolom 'Critical' dan hasilnya disimpan dalam variabel `unique_kategori_values1`.

```
# Menampilkan nilai unik untuk kolom 'Kategori'  
print("Unique values for 'Kategori' column:")  
print(unique_kategori_values)
```

Kode ini digunakan untuk mencetak nilai-nilai unik yang telah diambil dari kolom 'Kategori'. Pesan yang dicetak ke layar adalah "Unique values for 'Kategori' column:" diikuti oleh daftar nilai unik.

```
# Menampilkan nilai unik untuk kolom 'Critical'  
print("Unique values for 'Critical' column:")  
print(unique_kategori_values1)
```

Mirip dengan baris sebelumnya, tetapi kali ini mencetak nilai-nilai unik dari kolom 'Critical'. Pesan yang dicetak ke layar adalah "Unique values for 'Critical' column:" diikuti oleh daftar nilai unik.

```
1 # Mengambil nilai unik dari kolom 'Kategori'  
2 unique_kategori_values = dataudara['Kategori'].unique()  
3  
4 # Mengambil nilai unik dari kolom 'Critical'  
5 unique_kategori_values1 = dataudara['Critical'].unique()  
6  
7 # Menampilkan nilai unik untuk kolom 'Kategori'  
8 print("Unique values for 'Kategori' column:")  
9 print(unique_kategori_values)  
10  
11 # Menampilkan nilai unik untuk kolom 'Critical'  
12 print("Unique values for 'Critical' column:")  
13 print(unique_kategori_values1)  
14
```

Unique values for 'Kategori' column:
['Sedang', 'Baik', 'Tidak Ada Data', 'Tidak Sehat', 'BAIK', 'SEDANG', 'TIDAK ADA DATA', 'TIDAK SEHAT', 'SANGAT TIDAK SEHAT', NaN]
Categories (9, object): ['BAIK', 'Baik', 'SANGAT TIDAK SEHAT', 'SEDANG', ..., 'TIDAK ADA DATA',
'TIDAK SEHAT', 'Tidak Ada Data', 'Tidak Sehat']

```
Unique values for 'Critical' column:  
['PM10', 'SO2', NaN, 'O3', 'CO', 'PM25', 'NO2', 'TIDAK ADA DATA']  
Categories (7, object): ['CO', 'NO2', 'O3', 'PM10', 'PM25', 'SO2', 'TIDAK ADA DATA']
```

Catatan/Penjelasan dari Output di atas:

Output menunjukkan nilai unik dari kolom 'Kategori' dan 'Critical' dalam DataFrame 'dataudara':

Untuk kolom 'Kategori':

- Terdapat beberapa kategori yang mungkin memiliki variasi penulisan dan penggunaan huruf besar/kecil (case-insensitive), seperti 'Sedang', 'Baik', 'Tidak Ada Data', 'Tidak Sehat', 'BAIK', 'SEDANG', 'TIDAK ADA DATA', 'TIDAK SEHAT', 'SANGAT TIDAK SEHAT'.
- Selain itu, terdapat nilai NaN yang menunjukkan adanya data yang hilang atau tidak tersedia.
- Terdapat kategori yang teridentifikasi sebagai kategori khusus dengan menggunakan tipe data kategori (Categories), dengan total 9 kategori yang diakui oleh pandas.

Untuk kolom 'Critical':

- Terdapat beberapa jenis polutan udara yang menjadi fokus perhatian, seperti 'PM10', 'SO2', 'O3', 'CO', 'PM25', 'NO2'.
- Selain itu, terdapat nilai NaN yang menunjukkan adanya data yang hilang atau tidak tersedia.
- Terdapat jenis polutan yang teridentifikasi sebagai kategori khusus dengan menggunakan tipe data kategori (Categories), dengan total 7 jenis polutan yang diakui oleh pandas.

Analisis ini dapat membantu dalam pemahaman mengenai variasi kategori dan jenis polutan yang ada dalam dataudara, serta mengidentifikasi potensi masalah seperti data yang hilang atau variasi penulisan yang perlu diharmonisasi.

Prompt chatGPT:

Buatlah kode dalam bahasa pemrograman tertentu yang melakukan beberapa operasi pada DataFrame 'dataudara'. Pertama, konversi kolom 'Kategori' dan 'Critical' menjadi huruf besar. Selanjutnya, hapus semua baris yang memiliki nilai 'TIDAK ADA DATA', 'Tidak Ada Data', atau nilai NaN di kolom 'Kategori' dan 'Critical'. Terakhir, tampilkan nilai unik untuk kolom 'Kategori' setelah modifikasi.

Penjelasan Kode:

```
# Mengasumsikan 'dataudara' adalah DataFrame  
# Mengonversi kolom 'Kategori' dan 'Critical' menjadi huruf besar
```

```
dataudara['Kategori'] = dataudara['Kategori'].str.upper()  
dataudara['Critical'] = dataudara['Critical'].str.upper()
```

Baris pertama mengasumsikan bahwa ada sebuah DataFrame yang disebut 'dataudara'. Selanjutnya, kode ini mengubah isi kolom 'Kategori' dan 'Critical' menjadi huruf besar menggunakan metode `str.upper()` pada objek kolom tersebut.

```
# Menghapus 'TIDAK ADA DATA', 'Tidak Ada Data', dan NaN dari kolom 'Kategori'  
dataudara = dataudara[~dataudara['Kategori'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]
```

Baris ini menghapus baris-baris di DataFrame 'dataudara' dimana nilai pada kolom 'Kategori' adalah 'TIDAK ADA DATA', 'Tidak Ada Data', atau NaN. Penggunaan operator `~` dan `isin()` membuat kita dapat menghilangkan baris-baris tersebut.

```
# Menghapus 'TIDAK ADA DATA', 'Tidak Ada Data', dan NaN dari kolom 'Critical'  
dataudara = dataudara[~dataudara['Critical'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]
```

Sama seperti baris sebelumnya, namun kali ini berfokus pada kolom 'Critical'. Baris ini menghilangkan baris-baris di DataFrame 'dataudara' dimana nilai pada kolom 'Critical' adalah 'TIDAK ADA DATA', 'Tidak Ada Data', atau NaN.

```
# Menampilkan nilai unik untuk kolom 'Kategori' setelah dimodifikasi  
unique_kategori_values = dataudara['Kategori'].unique()  
print("Nilai unik untuk kolom 'Kategori':")  
print(unique_kategori_values)
```

Baris terakhir ini menampilkan nilai unik yang tersisa dalam kolom 'Kategori' setelah dilakukan modifikasi sebelumnya. Nilai-nilai ini kemudian dicetak untuk ditampilkan dalam output.

```
1 # Mengasumsikan 'dataudara' adalah DataFrame  
2 # Mengonversi kolom 'Kategori' dan 'Critical' menjadi huruf besar  
3 dataudara['Kategori'] = dataudara['Kategori'].str.upper()  
4 dataudara['Critical'] = dataudara['Critical'].str.upper()  
5  
6 # Menghapus 'TIDAK ADA DATA', 'Tidak Ada Data', dan NaN dari kolom 'Kategori'  
7 dataudara = dataudara[~dataudara['Kategori'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]  
8
```

```
9 # Menghapus 'TIDAK ADA DATA', 'Tidak Ada Data', dan NaN dari kolom 'Critical'  
10 dataudara = dataudara[~dataudara['Critical'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]  
11  
12 # Menampilkan nilai unik untuk kolom 'Kategori' setelah dimodifikasi  
13 unique_kategori_values = dataudara['Kategori'].unique()  
14 print("Nilai unik untuk kolom 'Kategori':")  
15 print(unique_kategori_values)  
16  
17
```

```
Nilai unik untuk kolom 'Kategori':  
['SEDANG' 'BAIK' 'TIDAK SEHAT' 'SANGAT TIDAK SEHAT']
```

Catatan/Penjelasan dari Output di atas:

Output tersebut menunjukkan nilai unik yang tersisa dalam kolom 'Kategori' setelah dilakukan serangkaian operasi pengolahan data. Berikut adalah penjelasan dari nilai-nilai unik tersebut:

1. **SEDANG:** Ini mungkin merujuk pada kategori kualitas udara yang dinyatakan sebagai "SEDANG". Ini mungkin menunjukkan bahwa kualitas udara pada tingkat ini masih dapat diterima, tetapi perlu diawasi.
2. **BAIK:** Ini mungkin merujuk pada kategori kualitas udara yang dinyatakan sebagai "BAIK". Ini menunjukkan bahwa kualitas udara pada tingkat ini dianggap baik atau sehat.
3. **TIDAK SEHAT:** Ini mungkin merujuk pada kategori kualitas udara yang dinyatakan sebagai "TIDAK SEHAT". Ini menandakan bahwa kualitas udara pada tingkat ini dianggap tidak sehat dan dapat memiliki dampak negatif pada kesehatan.
4. **SANGAT TIDAK SEHAT:** Ini mungkin merujuk pada kategori kualitas udara yang dinyatakan sebagai "SANGAT TIDAK SEHAT". Ini menunjukkan bahwa kualitas udara pada tingkat ini dianggap sangat tidak sehat dan dapat menyebabkan dampak kesehatan serius.

Dengan kata lain, output tersebut memberikan gambaran mengenai tingkat kualitas udara yang diukur dan dikategorikan ke dalam empat kategori berbeda: 'SEDANG', 'BAIK', 'TIDAK SEHAT', dan 'SANGAT TIDAK SEHAT'.

Prompt chatGPT:

Buatlah sebuah kode di Google Colab yang mengambil nilai unik dari kolom 'Critical' setelah dimodifikasi dan menampilkan nilai unik tersebut. Gunakan dataframe 'dataudara' untuk operasi ini. Berikan output dalam bentuk array atau daftar nilai unik. Pastikan untuk mencetak hasilnya dengan menggunakan perintah print.

Penjelasan Kode:

```
# Mengambil nilai unik dari kolom 'Critical' setelah dimodifikasi  
unique_critical_values = dataudara['Critical'].unique()
```

Baris pertama ini adalah sebuah komentar yang memberikan penjelasan singkat tentang tujuan baris kode berikutnya. Pada baris ini, kita menggunakan metode `unique()` dari Pandas untuk mengambil nilai unik dari kolom 'Critical' dalam DataFrame 'dataudara'. Hasilnya disimpan dalam variabel 'unique_critical_values'.

```
# Menampilkan nilai unik untuk kolom 'Critical'  
print("Nilai unik untuk kolom 'Critical':")  
print(unique_critical_values)
```

Baris kedua dan ketiga adalah komentar yang memberikan penjelasan tentang tindakan selanjutnya. Kode ini bertujuan untuk mencetak nilai unik yang telah diambil sebelumnya dari kolom 'Critical'. Fungsi `print()` digunakan untuk menampilkan pesan teks yang menjelaskan kontennya, yaitu "Nilai unik untuk kolom 'Critical':", dan kemudian fungsi `print()` digunakan lagi untuk mencetak nilai unik tersebut ke layar.

```
1 # Mengambil nilai unik dari kolom 'Critical' setelah dimodifikasi  
2 unique_critical_values = dataudara['Critical'].unique()  
3  
4 # Menampilkan nilai unik untuk kolom 'Critical'  
5 print("Nilai unik untuk kolom 'Critical':")  
6 print(unique_critical_values)  
7  
Nilai unik untuk kolom 'Critical':  
['PM10' 'SO2' 'O3' 'CO' 'PM25' 'NO2']
```

Catatan/Penjelasan dari Output di atas:

Output tersebut menunjukkan nilai unik yang terdapat dalam kolom 'Critical' dari DataFrame. Berikut adalah catatan/penjelasan untuk nilai-nilai unik tersebut:

1. '**PM10**': Merupakan singkatan dari Particulate Matter 10, yang mengacu pada partikel udara dengan diameter kurang dari atau sama dengan 10 mikrometer. Partikel ini dapat berasal dari debu jalan, industri, atau aktivitas pembakaran.
2. '**SO₂**': Merupakan singkatan dari sulfur dioksida, gas yang dihasilkan dari pembakaran bahan bakar fosil yang mengandung belerang. SO₂ dapat berkontribusi terhadap polusi udara dan membentuk hujan asam.
3. '**O₃**': Merupakan ozon, gas yang terdapat di atmosfer bumi dan dapat berperan sebagai polutan udara di tingkat permukaan. Ozon troposferik dapat menyebabkan masalah kesehatan dan merugikan tanaman.
4. '**CO**': Merupakan karbon monoksida, gas beracun yang dihasilkan dari pembakaran bahan bakar fosil. Pada kadar tinggi, CO dapat membahayakan kesehatan manusia.
5. '**PM25**': Merupakan singkatan dari Particulate Matter 2.5, yang merujuk pada partikel udara dengan diameter kurang dari atau sama dengan 2.5 mikrometer. Partikel ini dapat mencakup debu halus, asap rokok, dan polutan lainnya.
6. '**NO₂**': Merupakan nitrogen dioksida, gas yang dihasilkan dari berbagai sumber seperti kendaraan bermotor dan pembakaran bahan bakar. NO₂ dapat berkontribusi terhadap polusi udara dan memiliki dampak negatif pada kesehatan manusia.

Prompt chatGPT:

Tuliskan kode Python yang menggunakan fungsi `dataudara.info()` untuk mendapatkan informasi tentang data udara. Pastikan untuk memasukkan kode tersebut ke dalam blok kode Python.

Penjelasan Kode:

Berikut penjelasan kode:

```
# Menggunakan fungsi info() pada objek dataudara
dataudara.info()
```

1. `#` digunakan untuk menandai komentar dalam kode Python. Komentar tidak mempengaruhi eksekusi kode dan hanya digunakan untuk memberikan penjelasan kepada pembaca.
2. `dataudara.info()` adalah pemanggilan fungsi `info()` pada objek `dataudara`. Fungsi ini umumnya digunakan pada objek pandas DataFrame untuk memberikan informasi rinci tentang struktur dan tipe data di dalamnya. Informasi tersebut termasuk jumlah baris dan kolom, tipe data setiap kolom, penggunaan memori, dan sebagainya.

Dengan menjalankan kode ini, Anda akan mendapatkan output yang memberikan informasi rinci tentang dataset yang disimpan dalam objek `dataudara`.

```
1 # Menggunakan fungsi dataudara.info() untuk mendapatkan informasi tentang data udara
2 dataudara.info()
3

<class 'pandas.core.frame.DataFrame'>
Int64Index: 8705 entries, 0 to 8824
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          ----- 
 0   Tanggal     8705 non-null   datetime64[ns]
 1   Wilayah    8705 non-null   object  
 2   PM10        8705 non-null   float64 
 3   SO2         8705 non-null   float64 
 4   CO          8705 non-null   float64 
 5   O3          8705 non-null   float64 
 6   NO2        8705 non-null   float64 
 7   Max         8705 non-null   float64 
 8   Critical    8705 non-null   object  
 9   Kategori   8705 non-null   object  
dtypes: datetime64[ns](1), float64(6), object(3)
memory usage: 748.1+ KB
```

Catatan/Penjelasan dari Output di atas:

Output tersebut adalah hasil dari pemanggilan fungsi `info()` pada objek DataFrame di pandas. Berikut adalah catatan dan penjelasan dari output tersebut:

1. Kelas DataFrame:

- `<class 'pandas.core.frame.DataFrame'>`: Output ini menunjukkan bahwa objek yang ditangani adalah DataFrame dari pustaka pandas.

2. Indeks:

- `Int64Index: 8705 entries, 0 to 8824`: DataFrame memiliki indeks berupa bilangan bulat 64-bit, dimulai dari 0 hingga 8824, dengan total 8705 entri.

3. Kolom Data:

- `Data columns (total 10 columns)` : DataFrame memiliki sepuluh kolom.

4. Informasi Kolom:

- Setiap kolom memiliki informasi sebagai berikut:
 - `Tanggal` : Tipe data datetime64, dengan 8705 entri non-null.
 - `Wilayah` : Tipe data object (teks), dengan 8705 entri non-null.
 - `PM10, SO2, CO, O3, NO2, Max` : Tipe data float64, masing-masing dengan 8705 entri non-null.
 - `Critical, Kategori` : Tipe data object (teks), masing-masing dengan 8705 entri non-null.

5. Tipe Data dan Jumlah Non-Null:

- Setiap kolom memiliki informasi tentang tipe datanya (datetime64, float64, atau object) dan jumlah entri non-null.

6. Penggunaan Memori:

- `memory usage: 748.1+ KB` : Menunjukkan perkiraan penggunaan memori oleh DataFrame.

Output ini memberikan gambaran umum tentang struktur data, tipe data, dan jumlah entri non-null di setiap kolom dalam DataFrame. Ini membantu untuk memahami dataset dan memastikan bahwa data yang diinginkan telah dimuat dengan benar.

Prompt chatGPT:

Tulislah kode Python untuk menghitung matriks korelasi dari DataFrame 'dataudara', membuat sebuah heatmap segitiga termasuk diagonal menggunakan Seaborn, dan menampilkan plot dengan menggunakan Matplotlib. Pastikan untuk mengatur ukuran figur matplotlib menjadi 10x8, menyertakan anotasi, menggunakan peta warna 'coolwarm', dan menampilkan nilai korelasi dengan dua angka desimal.

Penjelasan Kode:

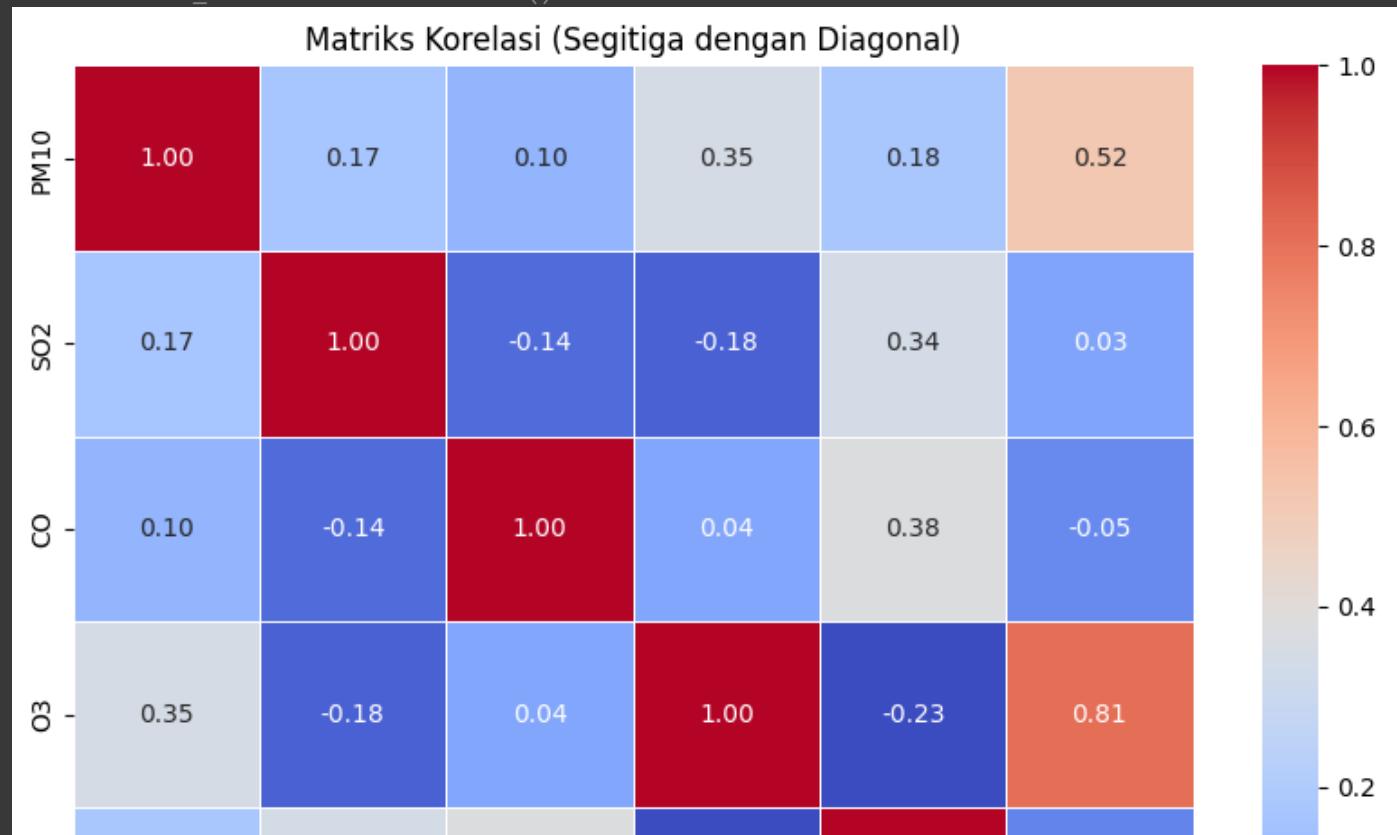
1. `correlation_matrix = dataudara.corr()` : Baris ini menghitung matriks korelasi dari DataFrame `dataudara` menggunakan metode `.corr()`. Matriks korelasi ini memuat nilai korelasi antara semua pasangan variabel dalam DataFrame.
2. `plt.figure(figsize=(10, 8))` : Baris ini membuat sebuah figur matplotlib dengan ukuran 10x8 menggunakan fungsi `figure()` dari modul Matplotlib. Ini menentukan ukuran figur yang akan digunakan untuk menampilkan heatmap.
3. `sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)` : Baris ini menggunakan Seaborn untuk membuat heatmap dari matriks korelasi. Parameter yang digunakan:

- `correlation_matrix`: Matriks korelasi yang telah dihitung sebelumnya.
- `annot=True`: Menampilkan nilai-nilai korelasi pada heatmap.
- `cmap='coolwarm'`: Menggunakan peta warna 'coolwarm' untuk menampilkan variasi korelasi.
- `fmt=".2f"`: Menentukan format angka untuk nilai korelasi dengan dua angka desimal.
- `linewidths=.5`: Menentukan lebar garis antara sel pada heatmap.

4. `plt.title('Matriks Korelasi (Segitiga dengan Diagonal)')`: Baris ini memberikan judul pada plot, dengan teks "Matriks Korelasi (Segitiga dengan Diagonal)".
5. `plt.show()`: Baris ini menampilkan plot yang telah dibuat menggunakan Matplotlib. Ini menunjukkan heatmap matriks korelasi dengan segitiga dan diagonal yang berisi nilai-nilai korelasi antar variabel.

```
1 # Menghitung matriks korelasi dari dataudara
2 correlation_matrix = dataudara.corr()
3
4 # Menyiapkan figur matplotlib dengan ukuran 10x8
5 plt.figure(figsize=(10, 8))
6
7 # Membuat heatmap segitiga termasuk diagonal menggunakan Seaborn
8 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
9
10 # Menampilkan plot
11 plt.title('Matriks Korelasi (Segitiga dengan Diagonal)')
12 plt.show()
13
```

```
<ipython-input-12-e5bc13fb9c7c>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it w
correlation_matrix = dataudara.corr()
```



Catatan/Penjelasan dari Output di atas:

Output gambar tersebut adalah matriks korelasi dari dataset `dataudara`. Matriks ini menggambarkan hubungan antara setiap variabel dalam dataset. Nilai dalam matriks berkisar antara -1 hingga 1, dengan -1 menunjukkan korelasi negatif, 0 menunjukkan tidak ada korelasi, dan 1 menunjukkan korelasi positif.

- **Korelasi positif** berarti jika nilai satu variabel meningkat, nilai variabel lainnya juga cenderung meningkat.
- **Korelasi negatif** berarti jika nilai satu variabel meningkat, nilai variabel lainnya cenderung menurun.
- **Tidak ada korelasi** berarti tidak ada hubungan yang jelas antara perubahan nilai dua variabel.

Warna pada heatmap berkisar dari biru (korelasi negatif) hingga merah (korelasi positif). Jadi, semakin merah suatu sel, semakin kuat korelasi positif antara dua variabel. Sebaliknya, semakin biru suatu sel, semakin kuat korelasi negatifnya.

Nilai diagonal selalu 1 karena mereka mewakili korelasi variabel dengan dirinya sendiri.

Matriks korelasi ini sangat berguna untuk memahami hubungan antara variabel dalam datasetnya dan dapat membantu dalam analisis eksplorasi data dan pemilihan fitur.

Apa itu korelasi?

Korelasi adalah ukuran statistik yang menunjukkan sejauh mana dua variabel bergerak bersama-sama. Dengan kata lain, korelasi mengukur sejauh mana perubahan dalam satu variabel berkaitan dengan perubahan dalam variabel lainnya.

Ada tiga jenis korelasi utama:

1. **Korelasi Positif:** Jika satu variabel meningkat, variabel lainnya juga meningkat. Misalnya, tinggi dan berat badan biasanya memiliki korelasi positif; orang yang lebih tinggi cenderung memiliki berat badan yang lebih besar.
2. **Korelasi Negatif:** Jika satu variabel meningkat, variabel lainnya menurun. Misalnya, jumlah jam belajar dan jumlah kesalahan dalam tes biasanya memiliki korelasi negatif; semakin banyak seseorang belajar, semakin sedikit kesalahan yang mereka buat.
3. **Tidak Ada Korelasi:** Tidak ada hubungan yang jelas antara perubahan dalam satu variabel dan perubahan dalam variabel lainnya.

Nilai korelasi biasanya berkisar antara -1 dan 1. Nilai -1 menunjukkan korelasi negatif sempurna, nilai 0 menunjukkan tidak ada korelasi, dan nilai 1 menunjukkan korelasi positif sempurna. Korelasi adalah alat yang sangat berguna dalam statistik dan analisis data.

Bagaimana cara membaca matriks korelasi?

Matriks korelasi adalah tabel yang menunjukkan koefisien korelasi antara setiap pasangan variabel dalam dataset. Nilai dalam matriks berkisar antara -1 hingga 1, dengan -1 menunjukkan korelasi negatif, 0 menunjukkan tidak ada korelasi, dan 1 menunjukkan korelasi positif.

Berikut adalah cara membaca matriks korelasi:

1. **Identifikasi Variabel:** Variabel biasanya ditampilkan di sumbu x dan y dari matriks. Setiap sel dalam matriks mewakili hubungan antara dua variabel.
2. **Baca Nilai Korelasi:** Nilai dalam setiap sel mewakili koefisien korelasi antara dua variabel. Nilai ini berkisar antara -1 dan 1.
 - Nilai **1** berarti ada korelasi positif sempurna antara dua variabel (ketika satu variabel meningkat, variabel lainnya juga meningkat).
 - Nilai **-1** berarti ada korelasi negatif sempurna (ketika satu variabel meningkat, variabel lainnya menurun).
 - Nilai **0** berarti tidak ada korelasi antara variabel.

3. **Perhatikan Warna:** Dalam heatmap, korelasi positif biasanya ditandai dengan warna hangat seperti merah, dan korelasi negatif ditandai dengan warna dingin seperti biru. Warna yang lebih gelap atau lebih terang menunjukkan kekuatan korelasi.
4. **Diagonal Utama:** Diagonal utama dari matriks (dari kiri atas ke kanan bawah) biasanya memiliki nilai 1, karena ini adalah korelasi variabel dengan dirinya sendiri.

Dengan memahami matriks korelasi, kita dapat mengidentifikasi hubungan antara variabel dalam dataset, yang dapat membantu dalam analisis data dan pemilihan fitur.

Prompt chatGPT:

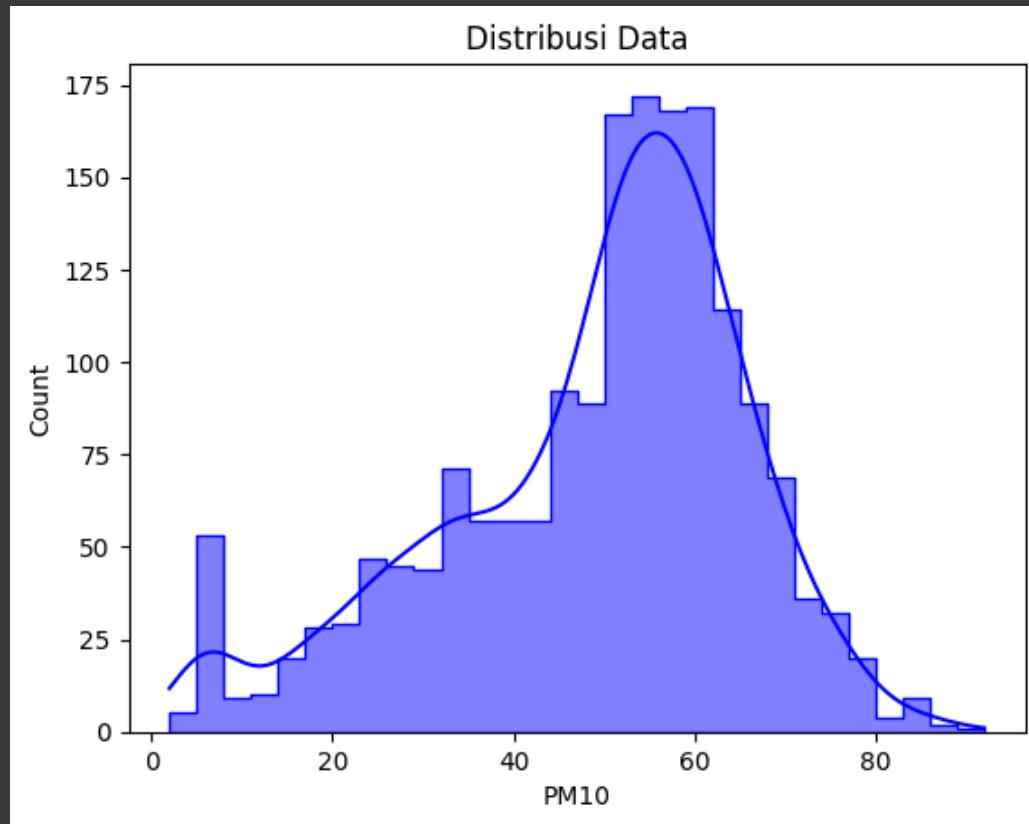
Buatlah kode menggunakan seaborn untuk membuat histogram dari kolom 'PM10' dengan 30 bins. Sertakan juga kernel density estimate (kde) dengan warna biru dan jenis elemen 'step'. Tambahkan judul plot dengan teks 'Distribusi Data' dan terakhir, tampilkan plot.

Penjelasan Kode:

Berikut penjelasan untuk setiap baris kode:

1. `sns.histplot(df['PM10'], bins=30, kde=True, color='blue', element='step')`: Menggunakan seaborn untuk membuat histogram dari kolom 'PM10' pada DataFrame `df`. Parameter `bins=30` menentukan jumlah bin (batang) yang akan digunakan dalam histogram. `kde=True` menambahkan kernel density estimate (kurva kepadatan) pada plot. `color='blue'` menentukan warna histogram menjadi biru, dan `element='step'` menentukan jenis elemen plot sebagai langkah-langkah (step plot).
2. `plt.title('Distribusi Data')`: Menambahkan judul pada plot dengan teks 'Distribusi Data'.
3. `plt.show()`: Menampilkan plot secara keseluruhan. Ini diperlukan untuk melihat hasil visualisasi yang telah dibuat menggunakan seaborn.

```
1 # Menggunakan seaborn untuk membuat histogram dari kolom 'PM10' dengan 30 bins
2 sns.histplot(df['PM10'], bins=30, kde=True, color='blue', element='step')
3
4 # Menambahkan judul plot
5 plt.title('Distribusi Data')
6
7 # Menampilkan plot
8 plt.show()
9
```



Catatan/Penjelasan dari Output di atas:

Histogram telah dibuat dari kolom 'PM10' dalam dataframe menggunakan seaborn. Histogram ini memiliki 30 bins dan data direpresentasikan sebagai garis langkah berwarna biru. Judul plot adalah 'Distribusi Data'.

- **Histogram:** Histogram adalah plot yang memungkinkan untuk menemukan dan menunjukkan distribusi frekuensi (bentuk) yang mendasari serangkaian data.
- **Bins:** Bins adalah interval data yang digunakan dalam histogram. Dalam hal ini, telah digunakan 30 bins, yang berarti data 'PM10' telah dibagi menjadi 30 interval yang sama.
- **PM10:** PM10 adalah partikel yang berdiameter kurang dari atau sama dengan 10 mikrometer. Dalam konteks ini, tampaknya merujuk ke konsentrasi PM10 dalam beberapa sampel data.

- **Distribusi Data:** Dari plot, tampak bahwa data 'PM10' mendekati distribusi normal dengan puncak sekitar 60. Ini berarti sebagian besar nilai 'PM10' berada di sekitar 60, dan nilai cenderung menyebar secara simetris di sekitar nilai ini.

Apa itu seaborn?

Seaborn adalah library Python yang digunakan untuk visualisasi data statistik. Library ini dibangun di atas matplotlib dan terintegrasi erat dengan struktur data pandas. Seaborn menyediakan antarmuka tingkat tinggi untuk membuat grafik statistik yang menarik dan informatif.

Berikut adalah beberapa fitur utama dari Seaborn:

- Dapat bekerja dengan baik dengan DataFrame pandas dan array yang berisi dataset lengkap.
- Menyediakan fungsi plotting tingkat tinggi untuk membuat berbagai jenis plot, seperti histogram, boxplots, violin plots, dan banyak lagi.
- Membantu menjelajahi dan memahami data.
- Fokus pada visualisasi distribusi variabel dan hubungan antara beberapa variabel.

Seaborn sangat berguna untuk analisis dan visualisasi data, dan dapat membantu mengekstrak wawasan penting dari data.

Prompt chatGPT:

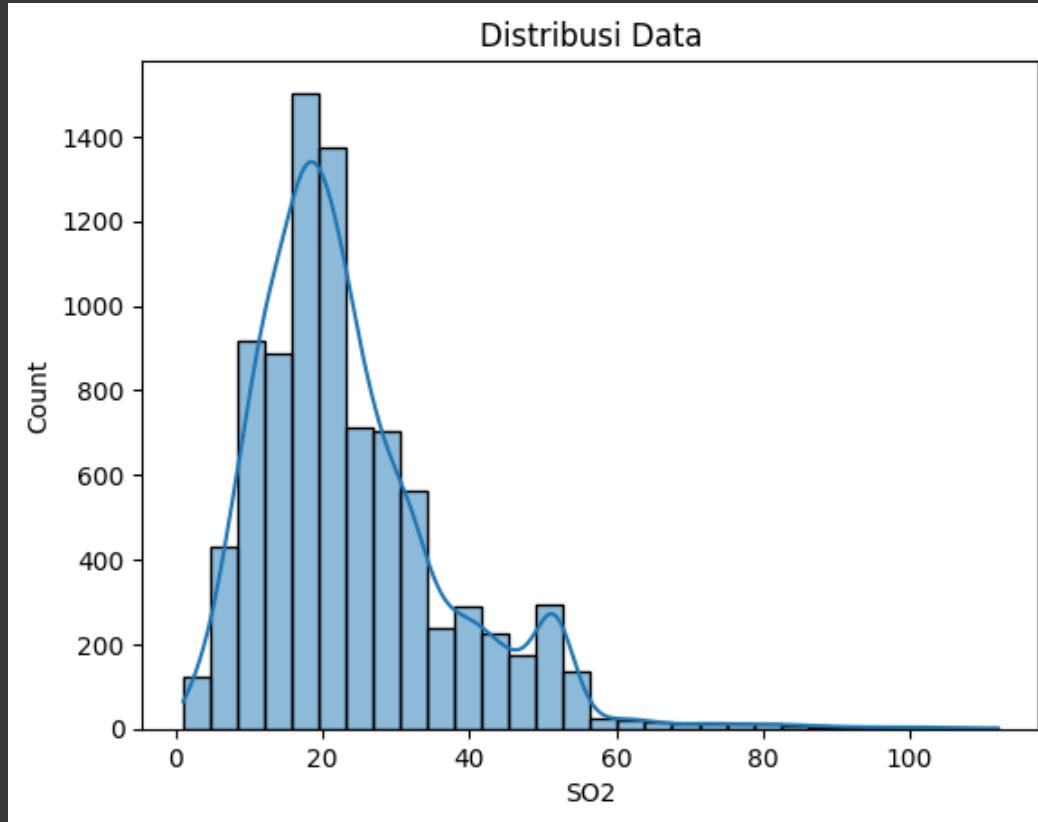
Buatlah kode menggunakan seaborn untuk membuat histogram dengan kernel density estimate (KDE) dari data yang terdapat dalam dataframe 'dataudara'. Gunakan kolom 'SO2' dengan 30 bins, dan tambahkan judul plot 'Distribusi Data'. Terakhir

Penjelasan Kode:

1. `sns.histplot(dataudara['SO2'], bins=30, kde=True)`: Baris ini menggunakan Seaborn untuk membuat histogram dengan kernel density estimate (KDE) dari kolom 'SO2' dalam dataframe 'dataudara'. Parameter `bins=30` menentukan jumlah bin pada histogram, sedangkan `kde=True` menambahkan kurva kernel density estimate.
2. `plt.title('Distribusi Data')`: Baris ini menambahkan judul pada plot yang telah dibuat. Judulnya adalah 'Distribusi Data'.
3. `plt.show()`: Baris ini menampilkan plot yang telah dibuat. Ini diperlukan agar plot dapat ditampilkan setelah sebelumnya telah disiapkan menggunakan Seaborn dan Matplotlib.

```
1 # Menggunakan seaborn untuk membuat histogram dengan kernel density estimate (KDE)
2 sns.histplot(dataudara['SO2'], bins=30, kde=True)
3
4 # Menambahkan judul plot
```

```
4 # Menampilkan judul plot  
5 plt.title('Distribusi Data')  
6  
7 # Menampilkan plot  
8 plt.show()
```



Catatan/Penjelasan dari Output di atas:

Histogram dengan estimasi densitas kernel (KDE) dibuat menggunakan pustaka seaborn. Variabel 'SO2' dari dataset 'dataudara' digunakan sebagai input. Histogram menunjukkan distribusi variabel 'SO2', dengan sumbu x mewakili nilai 'SO2' dan sumbu y mewakili jumlah pengamatan. KDE adalah kurva halus yang mengestimasi fungsi densitas probabilitas dari variabel 'SO2'. Judul plot adalah 'Distribusi Data'.

Dalam konteks ini, histogram digunakan untuk memvisualisasikan distribusi data 'SO2'. Binning data membantu dalam mengidentifikasi pola dalam data yang mungkin tidak terlihat dalam bentuk mentah. KDE, di sisi lain, memberikan perkiraan yang halus dari distribusi probabilitas yang mendasarinya.

Perlu dicatat bahwa pemilihan jumlah bins dalam histogram dapat mempengaruhi interpretasi data. Jumlah bins yang lebih besar dapat mengungkapkan detail lebih lanjut tentang data, tetapi juga dapat menghasilkan noise. Sebaliknya, jumlah bins yang lebih sedikit dapat mengurangi noise, tetapi juga dapat menghilangkan detail penting. Oleh karena itu, penting untuk memilih jumlah bins dengan hati-hati. Secara keseluruhan, plot ini memberikan gambaran yang baik tentang bagaimana data 'SO2' didistribusikan dalam dataset 'dataudara'.

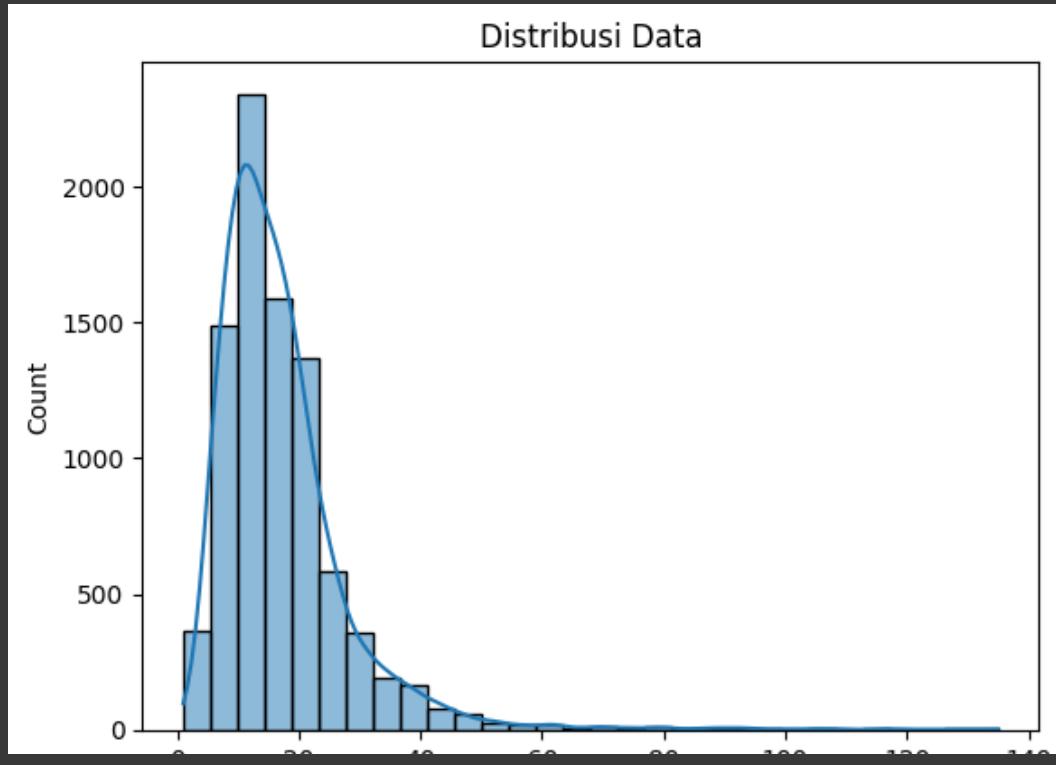
Prompt chatGPT:

Buatlah kode menggunakan sns.histplot untuk membuat histogram dari kolom 'CO' dalam DataFrame dataudara dengan 30 bins dan tambahkan KDE (Kernel Density Estimate). Selanjutnya, tambahkan judul plot dengan teks 'Distribusi Data'. Terakhir, gunakan plt.show() untuk menampilkan plot.

Penjelasan Kode:

1. `sns.histplot(dataudara['CO'], bins=30, kde=True)`: Baris ini menggunakan fungsi `histplot` dari modul `sns` (Seaborn) untuk membuat histogram dari kolom 'CO' dalam DataFrame `dataudara`. Parameter `bins=30` digunakan untuk menentukan jumlah bins (batang-batang) pada histogram, dan `kde=True` menambahkan KDE (Kernel Density Estimate) untuk menunjukkan perkiraan distribusi kontinu dari data.
2. `plt.title('Distribusi Data')`: Baris ini menambahkan judul pada plot dengan menggunakan fungsi `title` dari modul `plt` (Matplotlib). Judul yang ditambahkan pada plot ini adalah 'Distribusi Data'.
3. `plt.show()`: Baris ini digunakan untuk menampilkan plot yang telah dibuat. Fungsi `show` dari modul `plt` digunakan untuk menampilkan visualisasi plot ke layar atau output yang sesuai dengan lingkungan pengembangan yang digunakan.

```
1 # Menggunakan sns.histplot untuk membuat histogram
2 sns.histplot(dataudara['CO'], bins=30, kde=True)
3
4 # Menambahkan judul plot
5 plt.title('Distribusi Data')
6
7 # Menampilkan plot
8 plt.show()
```



Catatan/Penjelasan dari Output di atas:

Berikut ini adalah penjelasan dari output grafik histogram yang dihasilkan:

- Grafik ini menunjukkan distribusi variabel CO dalam dataset dataudara.
- Grafik ini memiliki 30 bins dan garis KDE.
- Sumbu x mewakili variabel CO dan sumbu y mewakili jumlahnya.
- Grafik ini menunjukkan bahwa variabel CO memiliki distribusi miring ke kanan dengan puncak sekitar 20.
- Mayoritas titik data jatuh antara 0 dan 40, dengan beberapa pencilan mencapai hingga 140.
- Judul dari plot ini adalah 'Distribusi Data'.

Bagaimana cara membaca grafik histogram?

Membaca grafik histogram melibatkan pemahaman tentang bagaimana data disajikan. Berikut adalah beberapa langkah untuk membaca grafik histogram:

1. **Identifikasi Judul:** Judul biasanya memberikan informasi tentang apa yang diukur dan bagaimana data tersebut dikumpulkan.
2. **Lihat Sumbu X dan Y:** Sumbu x (horizontal) menunjukkan kisaran data, sedangkan sumbu y (vertikal) menunjukkan jumlah pengamatan dalam setiap interval. Dalam konteks ini, sumbu x mewakili variabel CO dan sumbu y mewakili jumlahnya.
3. **Perhatikan Bins:** Histogram dibagi menjadi segmen yang disebut bins. Setiap bin mewakili rentang data. Dalam kasus ini, ada 30 bins.
4. **Analisis Distribusi:** Lihat bentuk distribusi data. Apakah data tersebut simetris (distribusi normal), miring ke kanan (positif skewness), atau miring ke kiri (negatif skewness)? Dalam kasus ini, variabel CO memiliki distribusi miring ke kanan.
5. **Identifikasi Puncak:** Puncak histogram adalah bin dengan frekuensi tertinggi. Dalam kasus ini, puncaknya berada sekitar 20.
6. **Cari Outliers:** Outliers adalah titik data yang jauh dari sebagian besar data lainnya. Dalam kasus ini, ada beberapa outliers yang mencapai hingga 140.
7. **Ringkasan:** Akhirnya, buat ringkasan dari apa yang telah diamati dari histogram. Misalnya, "Variabel CO memiliki distribusi miring ke kanan dengan mayoritas data antara 0 dan 40 dan beberapa outliers hingga 140."

Prompt chatGPT:

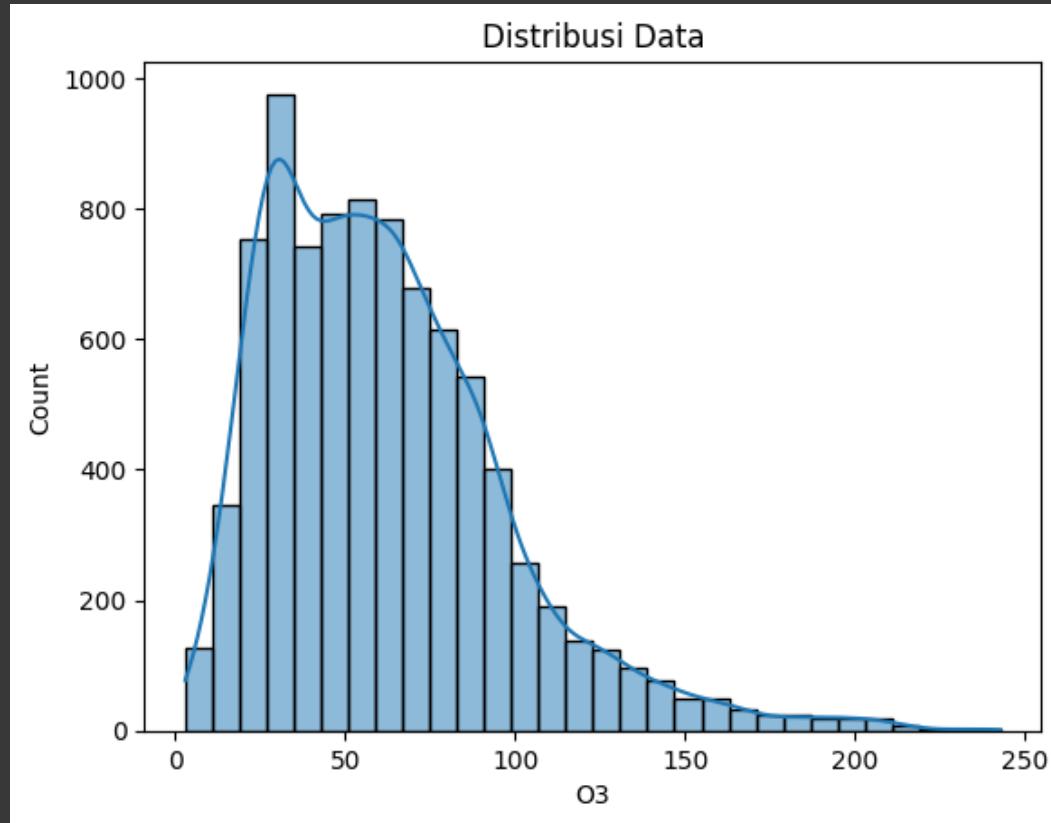
Buatlah kode menggunakan fungsi histplot dari modul seaborn untuk membuat histogram dari kolom 'O3' pada DataFrame bernama dataudara. Tetapkan jumlah bins sebanyak 30 dan tambahkan kurva kepadatan di atas histogram. Berikan judul plot yang menyatakan 'Distribusi Data'. Terakhir, gunakan plt.show() untuk menampilkan plot.

Penjelasan Kode:

1. `sns.histplot(dataudara['O3'], bins=30, kde=True)`: Menggunakan fungsi `histplot` dari modul Seaborn untuk membuat histogram. Dalam hal ini, kita menggunakan data dari kolom 'O3' pada DataFrame yang disebut `dataudara`. Parameter `bins=30` menentukan jumlah bins atau interval dalam histogram, sementara `kde=True` menambahkan kurva kepadatan di atas histogram.
2. `plt.title('Distribusi Data')`: Menambahkan judul ke plot yang telah dibuat. Judul plot ditetapkan sebagai 'Distribusi Data'.
3. `plt.show()`: Menampilkan plot yang telah dibuat menggunakan fungsi `show()` dari modul matplotlib.pyplot. Ini diperlukan agar plot dapat ditampilkan ketika Anda menjalankan skrip atau kode tersebut.

```
1 # Menggunakan fungsi histplot dari modul seaborn untuk membuat histogram
2 # dataudara['O3'] adalah kolom yang akan diplot
3 # bins=30 menentukan jumlah bins atau interval dalam histogram
```

```
4 # kde=True menambahkan kurva kepadatan di atas histogram
5 sns.histplot(dataudara['O3'], bins=30, kde=True)
6
7 # Menambahkan judul plot
8 plt.title('Distribusi Data')
9
10 # Menampilkan plot
11 plt.show()
```



Catatan/Penjelasan dari Output di atas:

Distribusi data O3 dalam dataset 'dataudara' ditampilkan dalam bentuk histogram. Histogram ini dibuat dengan menggunakan fungsi `histplot` dari modul seaborn dan parameter `bins=30` yang menentukan jumlah bins atau interval dalam histogram. Parameter `kde=True` digunakan untuk menambahkan kurva kepadatan di atas histogram.

Berikut adalah beberapa poin penting yang dapat dilihat dari histogram ini:

- Nilai O3 berkisar antara 0 hingga 250.
- Frekuensi tertinggi terletak sekitar 1000 dan frekuensi terendah sekitar 200.
- Kurva kepadatan (ditampilkan dalam garis biru) memberikan gambaran umum tentang distribusi data, yang tampaknya memiliki bentuk unimodal (satu puncak).

Secara keseluruhan, histogram ini memberikan gambaran visual tentang distribusi data O3 dalam dataset 'dataudara'. Ini membantu dalam memahami bagaimana data tersebar dan apakah ada pola tertentu yang dapat diamati.

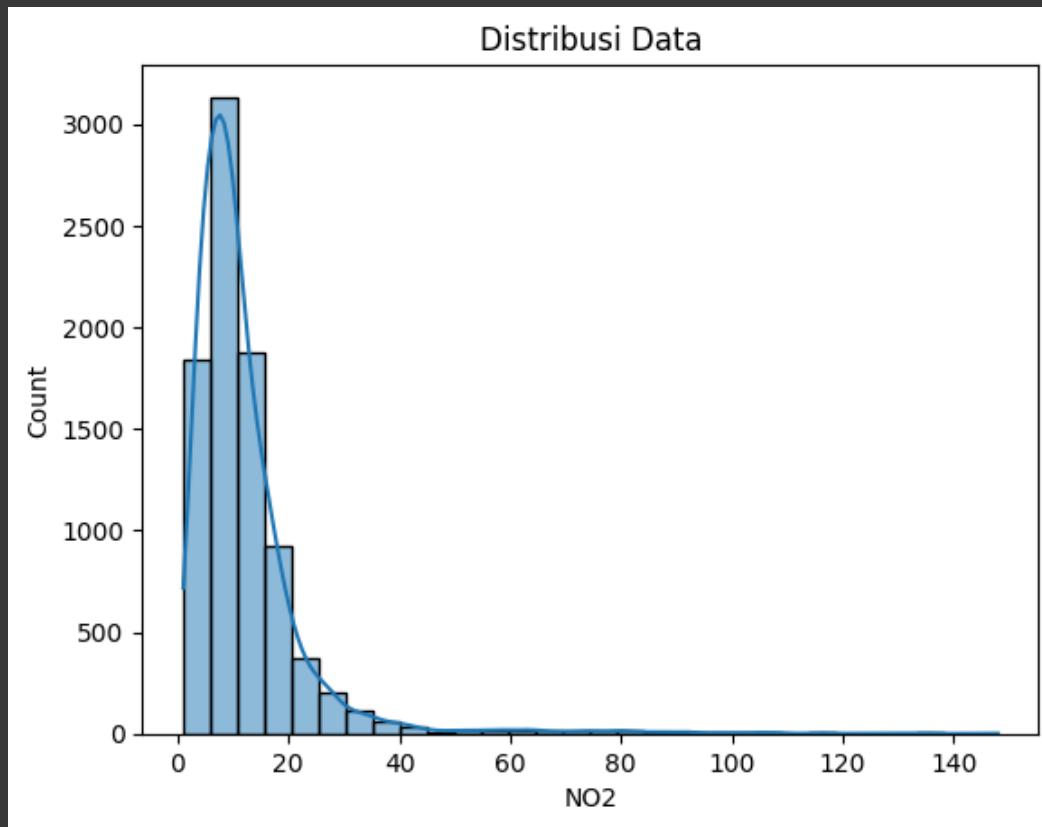
Prompt chatGPT:

Buatlah sebuah histogram plot menggunakan data dari kolom 'NO2' dengan 30 bins dan tambahkan kernel density estimation (kde). Sertakan juga judul pada plot yang berbunyi 'Distribusi Data'. Terakhir, tampilkan plot tersebut.

Penjelasan Kode:

1. `sns.histplot(dataudara['NO2'], bins=30, kde=True)`: Baris ini menggunakan fungsi `histplot` dari library Seaborn (`sns`) untuk membuat histogram plot. Data yang digunakan diambil dari kolom 'NO2' dalam DataFrame `dataudara`. Parameter `bins=30` menentukan jumlah bins atau batang dalam histogram, dan `kde=True` menambahkan kernel density estimation (kde) ke plot.
2. `plt.title('Distribusi Data')`: Baris ini menambahkan judul pada plot menggunakan fungsi `title` dari library Matplotlib (`plt`). Judul yang ditambahkan adalah 'Distribusi Data'.
3. `plt.show()`: Baris ini menggunakan fungsi `show` dari Matplotlib untuk menampilkan plot secara langsung. Ini diperlukan untuk menampilkan hasil plot yang sudah dibuat di atasnya.

```
1 # Membuat histogram plot menggunakan data dari kolom 'NO2', dengan 30 bins dan tambahan kernel density estimation (kde)
2 sns.histplot(dataudara['NO2'], bins=30, kde=True)
3
4 # Menambahkan judul pada plot
5 plt.title('Distribusi Data')
6
7 # Menampilkan plot
8 plt.show()
```



Catatan/Penjelasan dari Output di atas:

Berikut adalah penjelasan mengenai output grafik histogram yang dihasilkan:

- Grafik ini menunjukkan distribusi data NO2 dengan menggunakan 30 bins dan estimasi densitas kernel (kde).
- Sumbu x mewakili nilai NO2 dan sumbu y mewakili jumlah data.
- Mayoritas data berada di antara 0 dan 40, dengan puncak sekitar 20.
- Grafik juga menunjukkan bahwa ada sedikit data antara 80 dan 120.
- Judul dari plot ini adalah "Distribusi Data".

Dengan memahami distribusi data ini, dapat membantu dalam mengidentifikasi adanya data pencilan (outliers).

Prompt chatGPT:

Buatlah histogram dengan menggunakan fungsi histplot dari seaborn pada kolom 'Max' dari DataFrame dataudara. Tentukan jumlah bins sebanyak 30 dan aktifkan kernel density estimation (kde=True).

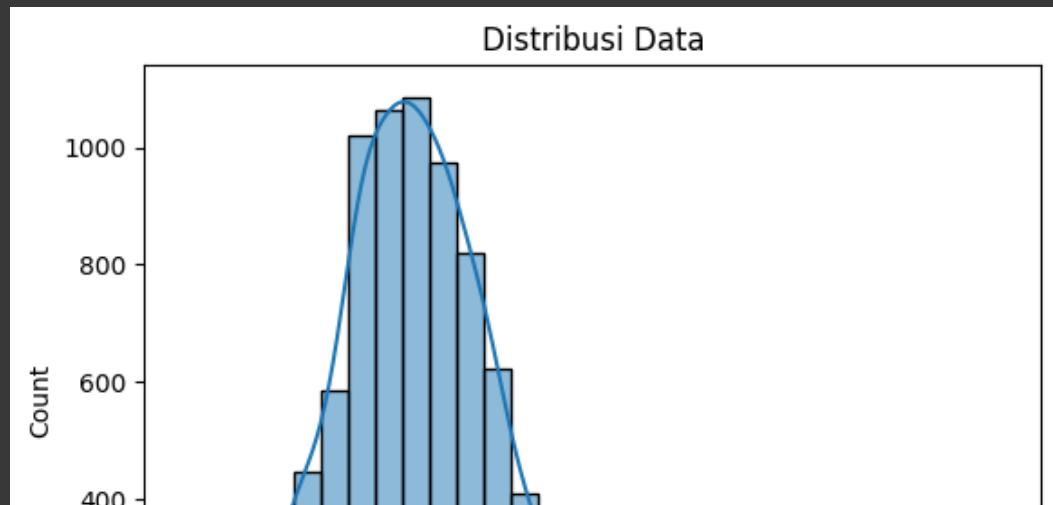
Tambahkan judul pada plot dengan menggunakan plt.title('Distribusi Data').

Terakhir, tampilkan plot dengan plt.show().

Penjelasan Kode:

1. `sns.histplot(dataudara['Max'], bins=30, kde=True)` : Menggunakan fungsi `histplot` dari library Seaborn (`sns`) untuk membuat histogram. `dataudara['Max']` digunakan sebagai data yang akan diplotkan. Parameter `bins=30` menentukan jumlah batang (bins) pada histogram, sedangkan `kde=True` mengaktifkan Kernel Density Estimation (KDE) untuk menunjukkan perkiraan distribusi probabilitas data.
2. `plt.title('Distribusi Data')` : Menambahkan judul pada plot yang telah dibuat. Fungsi `title` dari library Matplotlib (`plt`) digunakan untuk memberikan judul pada plot, dan judul yang ditetapkan di sini adalah 'Distribusi Data'.
3. `plt.show()` : Menampilkan plot yang telah dibuat. Fungsi `show` dari library Matplotlib (`plt`) digunakan untuk menampilkan plot ke layar sehingga dapat dilihat oleh pengguna atau disimpan dalam file gambar jika diinginkan.

```
1 # Menggunakan fungsi histplot dari seaborn untuk membuat histogram
2 sns.histplot(dataudara['Max'], bins=30, kde=True)
3
4 # Menambahkan judul pada plot
5 plt.title('Distribusi Data')
6
7 # Menampilkan plot
8 plt.show()
9
```



Catatan/Penjelasan dari Output di atas:

Histogram ini menunjukkan distribusi variabel 'Max' dalam dataset 'dataudara'. Plot ini dibuat menggunakan fungsi histplot dari pustaka seaborn. Plot ini memiliki 30 bins dan garis estimasi densitas kernel (KDE).

Judul plot adalah 'Distribusi Data'. Sumbu x mewakili variabel 'Max' dan sumbu y mewakili jumlah pengamatan di setiap bin. Plot menunjukkan bahwa variabel 'Max' memiliki distribusi yang kurang lebih normal dengan puncak sekitar 100 dan ekor menuju nilai yang lebih tinggi.

Dalam konteks ini, histogram adalah alat visualisasi data yang efektif untuk menunjukkan distribusi data. Dengan melihat histogram, dapat dilihat apakah data memiliki distribusi normal atau tidak, apakah data memiliki skewness atau kurtosis, dan apakah ada outlier dalam data. Histogram juga dapat digunakan untuk membandingkan distribusi dua set data atau lebih. Dalam hal ini, histogram menunjukkan distribusi variabel 'Max' dalam dataset 'dataudara'.

Prompt chatGPT:

Buatlah sebuah pair plot untuk melihat hubungan antar variabel dalam dataframe 'dataudara' menggunakan library seaborn (sns). Berikan judul pada pair plot dengan te

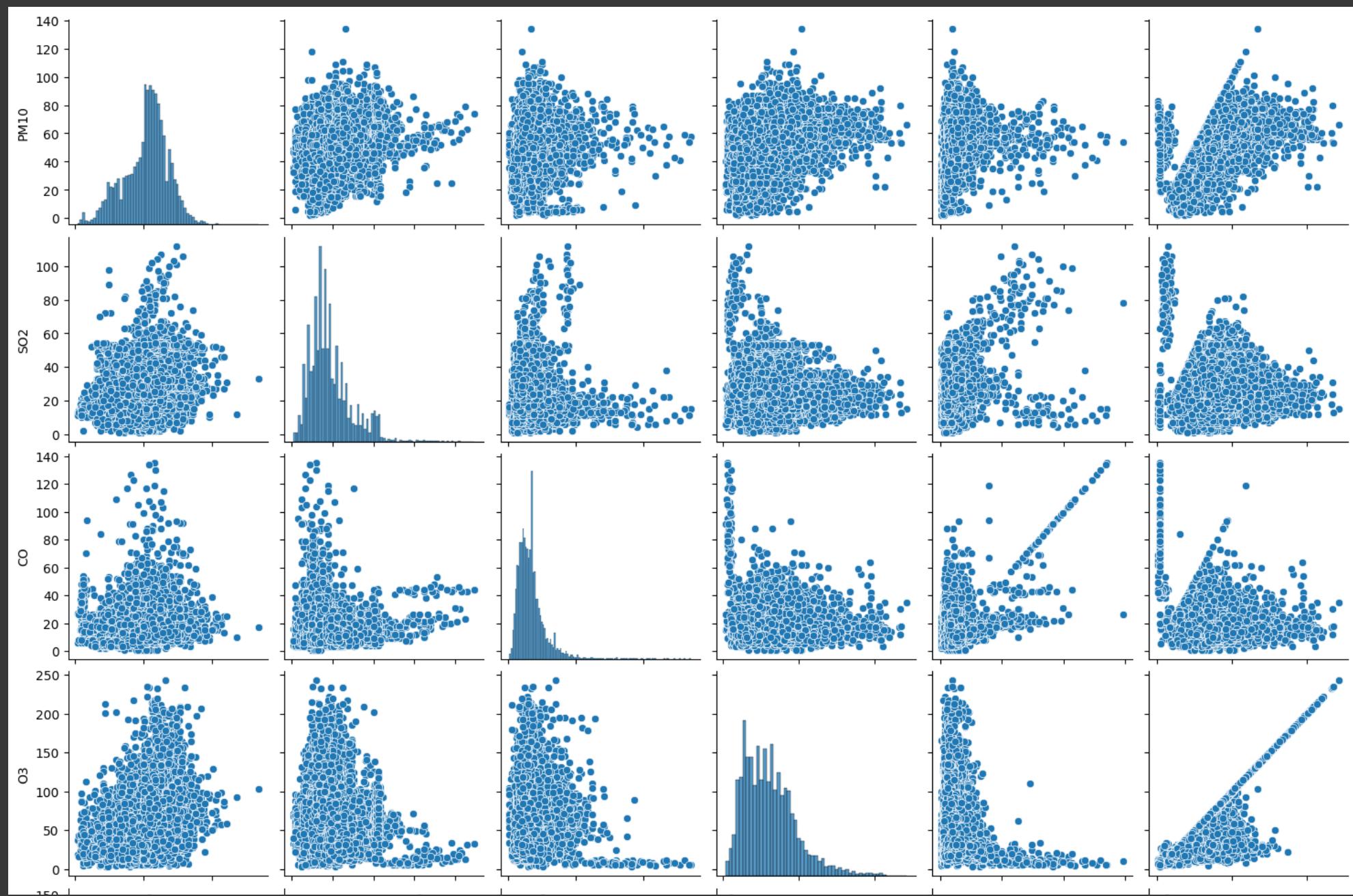
Penjelasan Kode:

1. `sns.pairplot(dataudara)` : Baris ini menggunakan fungsi `pairplot` dari library seaborn (`sns`) untuk membuat sebuah matriks pair plot yang menunjukkan hubungan antar variabel dalam dataframe `dataudara`. Setiap sel pada matriks ini berisi scatter plot antara dua

variabel, dan diagonalnya menunjukkan distribusi univariat dari setiap variabel.

2. `plt.title('Pair Plot')`: Dengan menggunakan fungsi `title` dari library pyplot (`plt`), baris ini menambahkan judul "Pair Plot" ke pair plot yang telah dibuat sebelumnya. Judul ini membantu dalam memberikan informasi singkat tentang isi dari pair plot.
3. `plt.show()`: Baris ini menampilkan pair plot yang telah dibuat. Fungsi `show()` dari library pyplot digunakan untuk menampilkan hasil visualisasi yang telah disiapkan sebelumnya.

```
1 # Membuat pair plot untuk melihat hubungan antar variabel dalam dataframe 'dataudara'  
2 sns.pairplot(dataudara)  
3  
4 # Memberikan judul pada pair plot  
5 plt.title('Pair Plot')  
6  
7 # Menampilkan pair plot  
8 plt.show()
```



Catatan/Penjelasan dari Output di atas:

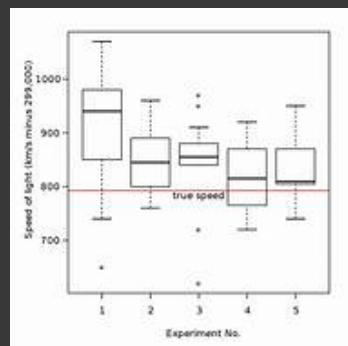
Berikut ini adalah penjelasan dari output pair plot yang telah dibuat:

Pair plot adalah cara visualisasi yang efektif untuk melihat hubungan antara beberapa variabel dalam satu gambar. Dalam hal ini, pair plot dibuat untuk dataframe 'dataudara'.

- Plot diagonal (dari kiri atas ke kanan bawah) menunjukkan distribusi dari variabel tersebut. Ini bisa berupa histogram atau plot densitas, tergantung pada pengaturan.
- Plot off-diagonal (yang bukan bagian dari plot diagonal) adalah scatter plot antara dua variabel. Scatter plot ini menunjukkan hubungan antara dua variabel.
- Dengan melihat scatter plot, dapat dilihat apakah ada korelasi antara dua variabel. Jika titik-titik membentuk garis diagonal dari kiri bawah ke kanan atas, itu menunjukkan korelasi positif. Sebaliknya, jika titik-titik membentuk garis diagonal dari kiri atas ke kanan bawah, itu menunjukkan korelasi negatif.
- Jika titik-titik tersebar secara acak, itu menunjukkan bahwa tidak ada korelasi antara dua variabel tersebut.

Harap dicatat bahwa pair plot hanya memberikan gambaran umum tentang data dan tidak dapat digunakan untuk membuat kesimpulan statistik yang pasti. Untuk analisis lebih lanjut, metode statistik lainnya mungkin perlu digunakan. Selain itu, pair plot bisa menjadi cara yang baik untuk melihat data pada tahap awal analisis data untuk memahami hubungan antar variabel.

▼ Data Visualization Outlier



Data Visualization atau visualisasi data adalah tampilan berupa grafis atau visual dari informasi dan data. Dengan kata lain, data visualization mengubah kumpulan data menjadi hal lebih sederhana untuk ditampilkan. Dengan menggunakan elemen visual tersebut, pembaca akan lebih mudah memahami tren, outliers, dan pola dalam suatu data¹. Dalam bisnis, data visualization memungkinkan para pembuat keputusan untuk melihat analitik yang disajikan secara visual, sehingga mereka dapat memahami konsep yang sulit atau mengidentifikasi pola baru.

Sedangkan **Outlier** atau pencilan adalah data yang nilainya terlalu jauh dengan data lainnya². Keberadaan outlier akan membuat analisis terhadap data tersebut menjadi bias atau dengan kata lain tidak mencerminkan fenomena yang sebenarnya². Outlier ini bisa juga dikenal dengan anomali ini adalah sebuah data atau observasi yang menyimpang secara ekstrim dari rata-rata sekumpulan data yang ada. Penyimpangan ini bisa berupa angkanya terlalu tinggi atau terlalu rendah.

Dalam konteks visualisasi data, outlier ini dapat terlihat sebagai titik data atau elemen visual yang jauh dari kumpulan data lainnya. Dengan memvisualisasikan data, kita dapat lebih mudah mengidentifikasi dan memahami outlier ini. Selain itu, visualisasi data juga membantu kita untuk memahami pola, tren, hubungan, dan outlier yang tersembunyi di dalam data yang besar dan kompleks.

Outlier dalam konteks data visualization mengacu pada nilai yang secara signifikan berbeda dari sebagian besar data. Outlier bisa menjadi titik data yang jauh dari sebagian besar observasi atau dapat mewakili anomali atau kesalahan dalam pengukuran atau pengumpulan data.

Konsep data visualization outlier mencakup cara-cara untuk mengidentifikasi, menyoroti, dan memahami outlier dalam data menggunakan visualisasi grafis. Beberapa teknik umum yang digunakan untuk menangani outlier dalam visualisasi data melibatkan penggunaan grafik seperti:

1. **Box Plot (Diagram Kotak)**: Box plot dapat memberikan gambaran visual tentang sebaran data dan dengan mudah menyoroti nilai-nilai yang berada jauh dari kuartil.
2. **Scatter Plot (Diagram Sebar)**: Scatter plot adalah grafik yang menunjukkan hubungan antara dua variabel. Outlier dapat ditemukan sebagai titik-titik yang jauh dari pola umum pada scatter plot.
3. **Histogram**: Histogram dapat membantu menggambarkan distribusi data, dan outlier dapat terlihat sebagai puncak atau ekor yang mencolok di luar distribusi utama.
4. **Heatmaps**: Heatmap dapat digunakan untuk memvisualisasikan data dalam bentuk matriks warna. Outlier dapat ditemukan sebagai nilai yang menonjol dengan warna yang berbeda.
5. **Violin Plot**: Violin plot merupakan gabungan antara box plot dan kernel density plot. Ini membantu dalam menunjukkan sebaran data dan juga memperlihatkan kepadatan data pada berbagai nilai.
6. **Lorenz Curve**: Ini digunakan terutama dalam ekonomi untuk menunjukkan distribusi kumulatif dari sebuah variabel. Outlier dapat terlihat sebagai ketidaksesuaian besar dari kurva Lorenz yang diharapkan.

Penting untuk dicatat bahwa tidak semua nilai yang dianggap sebagai outlier harus dihapus. Outlier dapat memberikan informasi berharga tentang data, seperti tren yang tidak terduga atau kesalahan pengukuran. Penting untuk memahami konteks data dan memberikan penilaian yang tepat terhadap nilai-nilai outlier tersebut.

Prompt chatGPT:

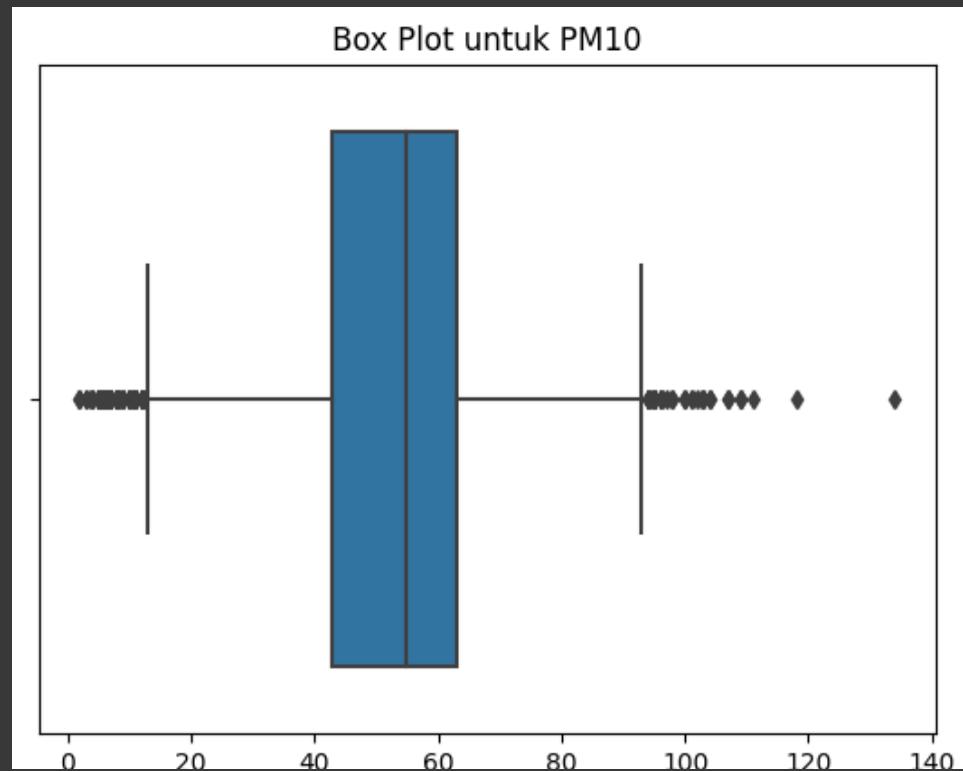
Buatlah sebuah program Python untuk menghasilkan box plot menggunakan library seaborn dan matplotlib.pyplot. Asumsikan bahwa terdapat sebuah DataFrame yang disebut 'dataudara', dan Anda ingin membuat box plot untuk kolom 'PM10'. Berikan judul plot

Penjelasan Kode:

Berikut adalah penjelasan untuk setiap baris kode:

1. `import seaborn as sns`: Mengimpor library seaborn dan memberikan alias `sns` untuk mempermudah penggunaan nantinya.
2. `import matplotlib.pyplot as plt`: Mengimpor library pyplot dari matplotlib dan memberikan alias `plt` untuk mempermudah penggunaan nantinya.
3. `sns.boxplot(x='PM10', data=dataudara)`: Membuat box plot menggunakan seaborn untuk kolom 'PM10' dalam DataFrame 'dataudara'.
Box plot digunakan untuk menampilkan distribusi statistik seperti kuartil dan nilai ekstrem.
4. `plt.title('Box Plot untuk PM10')`: Menambahkan judul pada plot yang telah dibuat. Judulnya adalah 'Box Plot untuk PM10'.
5. `plt.show()`: Menampilkan plot yang telah dibuat. Fungsi ini diperlukan untuk melihat visualisasi yang telah dihasilkan.

```
1 # Mengimpor library seaborn dan matplotlib.pyplot
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4
5 # Mengasumsikan 'dataudara' adalah DataFrame
6 # Membuat box plot untuk kolom 'PM10' dalam DataFrame 'dataudara'
7 sns.boxplot(x='PM10', data=dataudara)
8
9 # Menambahkan judul plot
10 plt.title('Box Plot untuk PM10')
11
12 # Menampilkan plot
13 plt.show()
14
```

**Catatan/Penjelasan dari Output di atas:**

Dari grafik box plot untuk kolom 'PM10' dalam DataFrame 'dataaudara', dapat dilihat bahwa:

- Nilai median (garis tengah kotak) berada di sekitar 60. Ini berarti bahwa setengah dari nilai PM10 berada di bawah 60 dan setengahnya lagi berada di atas 60.
- Kuartil bawah dan atas (batas bawah dan atas kotak) berada di sekitar 40 dan 80. Ini berarti bahwa sekitar 50% dari nilai PM10 berada dalam rentang ini.
- Terdapat beberapa pencilan di atas 120. Ini berarti bahwa terdapat beberapa nilai PM10 yang jauh lebih tinggi dari mayoritas nilai lainnya.
- Dengan demikian, dapat disimpulkan bahwa sebagian besar nilai PM10 berada antara 40 dan 80, dengan beberapa nilai yang lebih tinggi.

Prompt chatGPT:

Tuliskan kode dalam bahasa Python menggunakan Plotly Express untuk membuat box plot dari DataFrame 'dataaudara', dengan kolom 'PM10' sebagai sumbu y. Atur judul plot menjadi 'Box Plot untuk PM10' dan modifikasi layout untuk mengatur lebar dan tinggi plot menjadi 700 dan

500, secara berturut-turut. Terakhir, tampilkan plot yang telah dimodifikasi.

```
1 # Mengimpor library Plotly Express sebagai px
2 import plotly.express as px
3
4 # Mengasumsikan 'dataudara' adalah DataFrame
5 # Membuat box plot untuk kolom 'PM10'
6 fig = px.box(dataudara, y='PM10', title='Box Plot untuk PM10')
7
8 # Mengupdate layout untuk mengatur lebar dan tinggi plot
9 fig.update_layout(width=700, height=500)
10
11 # Menampilkan plot yang telah dimodifikasi
12 fig.show()
13
14
```

Box Plot untuk PM10

Catatan/Penjelasan dari Output di atas:

Berikut ini adalah penjelasan dari output grafik box plot untuk kolom 'PM10':

Box plot adalah metode untuk menggambarkan grup angka melalui lima ukuran ringkasan: minimum, kuartil pertama (Q1), median (atau Q2), kuartil ketiga (Q3), dan maksimum.

Dalam hal ini, box plot menunjukkan distribusi nilai PM10 dalam data. Nilai median adalah 55, yang berarti setengah dari pengamatan berada di bawah nilai ini dan setengahnya di atas. Nilai minimum (lower fence) adalah 13 dan nilai maksimum (upper fence) adalah 93. Nilai maksimum dalam data adalah 134.

Box plot ini dapat membantu memahami penyebaran dan variabilitas nilai PM10 dalam data. Selain itu, box plot juga dapat digunakan untuk mengidentifikasi pencilan (outliers) dalam data. Pencilan adalah pengamatan yang jauh dari sebagian besar pengamatan lainnya. Dalam konteks ini, nilai di atas 93 dapat dianggap sebagai pencilan.

Harap dicatat bahwa interpretasi ini didasarkan pada asumsi bahwa 'dataudara' adalah DataFrame yang berisi data kualitas udara, dan 'PM10' adalah kolom dalam DataFrame tersebut yang mewakili konsentrasi partikel PM10 dalam udara.

Prompt chatGPT:

Buatlah sebuah box plot menggunakan Plotly Express untuk DataFrame yang disebut 'dataudara' dengan menggunakan data kolom 'CO'. Beri judul plot tersebut 'Box Plot for CO'. Setelah itu, modifikasi layout plot dengan menetapkan lebar menjadi 700 dan tinggi menjadi 500. Terakhir, tampilkan plot yang telah dimodifikasi.

Penjelasan Kode:

1. `fig = px.box(dataudara, y='CO', title='Box Plot for CO')`: Baris ini membuat box plot menggunakan Plotly Express. Fungsi `px.box` digunakan untuk membuat box plot. Argumen pertama adalah DataFrame `dataudara`, dan argumen kedua, `y='CO'`, menunjukkan bahwa sumbu y plot akan menggunakan data dari kolom 'CO' dalam DataFrame tersebut. Argumen ketiga, `title='Box Plot for CO'`, memberikan judul plot.
2. `fig.update_layout(width=700, height=500)`: Baris ini mengupdate layout plot yang sudah dibuat sebelumnya. Fungsi `update_layout` digunakan untuk mengubah beberapa properti layout plot. Di sini, kita menetapkan lebar plot (`width`) menjadi 700 dan tinggi plot

(height) menjadi 500.

3. `fig.show()`: Baris ini menampilkan plot yang telah dimodifikasi. Fungsi `show()` digunakan untuk menampilkan plot ke dalam antarmuka pengguna atau notebook sesuai dengan lingkungan kerja yang digunakan.

```
1 # Memasukkan data pada DataFrame 'dataudara' ke dalam box plot menggunakan Plotly Express
2 fig = px.box(dataudara, y='CO', title='Box Plot for CO')
3
4 # Mengupdate layout plot untuk menetapkan lebar dan tinggi
5 fig.update_layout(width=700, height=500)
6
7 # Menampilkan plot yang telah dimodifikasi
8 fig.show()
9
10
```

Box Plot SO₂

Catatan/Penjelasan dari Output di atas:

Box plot untuk level CO dalam DataFrame 'dataudara' memberikan representasi visual dari distribusi level CO. Berikut adalah apa yang diwakili oleh angka-angka tersebut:

- Level **median** CO, diwakili oleh garis di dalam kotak, adalah **15**. Ini adalah nilai tengah dari set data, yang berarti bahwa setengah dari level CO berada di atas nilai ini dan setengahnya berada di bawah.
- **Batas bawah** atau bagian bawah kotak berada pada **10**. Ini mewakili persentil ke-25, yang berarti bahwa 25% dari level CO berada di bawah nilai ini.
- **Batas atas** atau bagian atas kotak berada pada **37**. Ini mewakili persentil ke-75, yang berarti bahwa 75% dari level CO berada di bawah nilai ini.
- Level CO **minimum**, diwakili oleh kumis bawah plot, adalah **1**. Ini adalah level CO terendah dalam set data.
- Level CO **maksimum**, diwakili oleh kumis atas plot, adalah **135**. Ini adalah level CO tertinggi dalam set data.

Box plot adalah alat yang berguna untuk memahami penyebaran dan variabilitas level CO dalam DataFrame 'dataudara'. Ini dapat membantu mengidentifikasi penculan dan memahami distribusi keseluruhan data. Rentang yang luas antara nilai minimum dan maksimum menunjukkan variabilitas tinggi dalam level CO. Median yang relatif tinggi dibandingkan dengan batas bawah menunjukkan bahwa level CO condong ke nilai yang lebih tinggi. Box plot dapat digunakan untuk menginformasikan analisis lebih lanjut dan pengambilan keputusan terkait dengan manajemen kualitas udara.

Prompt chatGPT:

Hasilkan sebuah diagram kotak (box plot) untuk kolom 'SO2' dari DataFrame yang disebut 'dataudara'. Atur judul diagram kotak tersebut sebagai 'Diagram Kotak untuk SO2'. Sesuaikan tata letak untuk mengatur lebar menjadi 700 dan tinggi menjadi 500. Terakhir, tampilkan plot yang telah dimodifikasi.

Penjelasan Kode:

1. `fig = px.box(dataudara, y='SO2', title='Diagram Kotak untuk SO2'):`
 - `px.box`: Ini menggunakan fungsi `box` dari `plotly express (px)` untuk membuat diagram kotak.
 - `dataudara`: Merupakan DataFrame yang diasumsikan sudah ada sebelumnya, dan digunakan sebagai sumber data untuk plot.

- `y='SO2'` : Menunjukkan bahwa sumbu y dari diagram kotak akan mewakili data dari kolom 'SO2' dalam DataFrame.
- `title='Diagram Kotak untuk SO2'` : Menetapkan judul diagram kotak menjadi 'Diagram Kotak untuk SO2'.

2. `fig.update_layout(width=700, height=500)`:

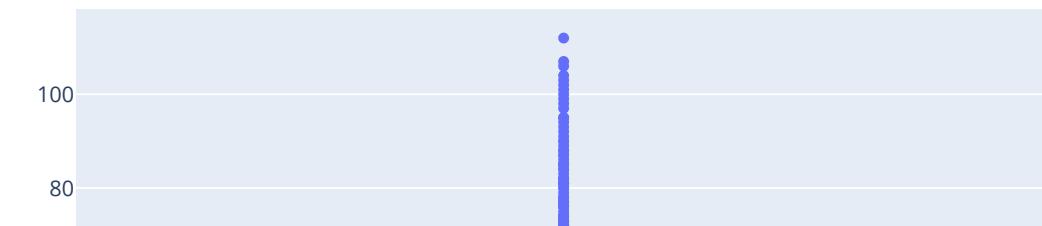
- `fig`: Merujuk pada objek diagram kotak yang telah dibuat sebelumnya.
- `update_layout`: Ini adalah metode untuk memperbarui berbagai properti layout dari plot.
- `width=700, height=500`: Menetapkan lebar plot menjadi 700 dan tinggi plot menjadi 500 piksel.

3. `fig.show()`:

- `fig`: Merujuk kembali pada objek diagram kotak yang telah diperbarui.
- `show()`: Memanggil fungsi `show()` untuk menampilkan plot diagram kotak yang telah dimodifikasi sesuai dengan pengaturan yang telah diperbarui sebelumnya.

```
1 # Mengasumsikan 'dataaudara' sebagai DataFrame
2 fig = px.box(dataaudara, y='SO2', title='Diagram Kotak untuk SO2')
3
4 # Memperbarui tata letak untuk mengatur lebar dan tinggi
5 fig.update_layout(width=700, height=500)
6
7 # Menampilkan plot yang telah dimodifikasi
8 fig.show()
9
```

Diagram Kotak untuk SO2



Catatan/Penjelasan dari Output di atas:

Berikut ini adalah penjelasan tentang output grafik boxplot untuk SO2:

Boxplot atau diagram kotak adalah metode untuk menggambarkan kelompok data numerik melalui kuartil mereka. Dalam hal ini, boxplot digunakan untuk menampilkan distribusi nilai SO2 dalam DataFrame 'dataudara'.

- Nilai minimum SO2 dalam data adalah 1.
- Nilai Q1 atau kuartil pertama (batas bawah kotak biru) adalah 15. Ini berarti 25% dari data berada di bawah nilai ini.
- Nilai median atau Q2 (garis tengah kotak biru) adalah 31. Ini berarti 50% dari data berada di bawah nilai ini.
- Nilai Q3 atau kuartil ketiga (batas atas kotak biru) adalah 52.4. Ini berarti 75% dari data berada di bawah nilai ini.
- Nilai maksimum SO2 dalam data adalah 112.

Dengan demikian, boxplot memberikan gambaran yang jelas tentang bagaimana data SO2 tersebar dan dapat membantu dalam mengidentifikasi outlier atau anomali dalam data. Selain itu, lebar dan tinggi plot telah disesuaikan menjadi 700 dan 500 piksel masing-masing untuk memastikan tampilan yang optimal.

Prompt chatGPT:

Buatlah sebuah box plot menggunakan Plotly Express untuk DataFrame bernama 'dataudara'. Gunakan sumbu y='Max' dan beri judul 'Box Plot for Max'. Setelah itu, perbarui tata letak (layout) plot untuk menetapkan lebar plot sebesar 700 dan tinggi sebesar 500. Terakhir, minta agar kode menampilkan plot yang telah dimodifikasi. Tuliskan kodennya dalam bentuk plaintext.

Penjelasan Kode:

1. `fig = px.box(dataudara, y='Max', title='Box Plot for Max')`: Baris ini membuat sebuah objek figure (`fig`) menggunakan Plotly Express untuk box plot dari DataFrame 'dataudara'. `y='Max'` mengindikasikan bahwa sumbu y dari box plot akan menggunakan kolom 'Max' dari DataFrame tersebut. Selain itu, judul plot diatur menjadi 'Box Plot for Max'.
2. `fig.update_layout(width=700, height=500)`: Baris ini memperbarui tata letak (layout) dari plot yang telah dibuat sebelumnya. `width=700` dan `height=500` mengatur lebar dan tinggi plot secara berturut-turut.
3. `fig.show()`: Baris ini menampilkan plot yang telah dimodifikasi ke layar atau output. Ini adalah langkah akhir untuk melihat visualisasi box plot yang telah dibuat.

```
1 # Menggunakan plotly express untuk membuat box plot dari DataFrame 'dataudara' dengan sumbu y='Max' dan judul 'Box Plot for Max'
2 fig = px.box(dataudara, y='Max', title='Box Plot for Max')
3
4 # Memperbarui tata letak (layout) plot untuk menetapkan lebar dan tinggi
5 fig.update_layout(width=700, height=500)
6
7 # Menampilkan plot yang telah dimodifikasi
8 fig.show()
9
```

Box Plot for Max

250

**Catatan/Penjelasan dari Output di atas:**

Sebuah box plot dibuat menggunakan library Plotly Express untuk variabel 'Max' dari DataFrame 'dataudara'. Box plot ini memberikan gambaran tentang distribusi data 'Max'.

Berikut adalah penjelasan tentang angka-angka pada box plot:

- Nilai Median: Nilai tengah dari data, dalam hal ini adalah 89. Ini berarti bahwa setengah dari nilai 'Max' berada di bawah 89 dan setengahnya lagi berada di atas 89.
- Nilai Minimum: Nilai terendah dalam data, dalam hal ini adalah 1.
- Nilai Maksimum: Nilai tertinggi dalam data, dalam hal ini adalah 243.
- Lower Fence: Batas bawah dari data, dalam hal ini adalah 3. Nilai di bawah ini dianggap sebagai pencilan (outlier).
- Upper Fence: Batas atas dari data. Nilai di atas ini dianggap sebagai pencilan (outlier).
- Pencilan (Outliers): Nilai-nilai yang jauh dari nilai lainnya dalam data. Dalam plot ini, terdapat beberapa pencilan di atas upper fence.

Box plot ini juga telah dimodifikasi untuk memiliki lebar 700 dan tinggi 500. Ini membantu dalam visualisasi data dengan lebih jelas.

Harap dicatat bahwa box plot ini hanya memberikan gambaran umum tentang distribusi data 'Max'. Untuk analisis lebih lanjut, mungkin perlu dilakukan analisis statistik tambahan.

Prompt chatGPT:

Buatlah sebuah box plot untuk kolom 'NO2' dari DataFrame 'dataudara' menggunakan plotly express (px). Atur judul plot menjadi 'Box Plot for NO2'. Setelah itu, perbarui layout plot untuk mengatur lebar menjadi 700 dan tinggi menjadi 500. Terakhir

Penjelasan Kode:

Sure, berikut adalah penjelasan lengkap untuk setiap baris kode:

```
1. fig = px.box(dataudara, y='NO2', title='Box Plot for NO2'):
```

- `px.box` : Membuat sebuah box plot menggunakan Plotly Express.
- `dataaudara` : Nama DataFrame yang digunakan untuk membuat box plot.
- `y='NO2'` : Menentukan bahwa sumbu y dari box plot akan menampilkan data dari kolom 'NO2' DataFrame.
- `title='Box Plot for NO2'` : Menetapkan judul plot sebagai 'Box Plot for NO2'.

2. `fig.update_layout(width=700, height=500)`:

- `fig`: Merujuk pada objek plot yang telah dibuat sebelumnya.
- `update_layout`: Mengizinkan pembaruan pada layout plot.
- `width=700`: Menetapkan lebar plot menjadi 700 piksel.
- `height=500`: Menetapkan tinggi plot menjadi 500 piksel.

3. `fig.show()`:

- Menampilkan plot yang telah dimodifikasi dengan judul, sumbu y, lebar, dan tinggi yang telah diatur sebelumnya.

```
1 # Assuming 'dataaudara' is your DataFrame
2 fig = px.box(dataaudara, y='NO2', title='Box Plot for NO2')
3
4 # Update layout to set width and height
5 fig.update_layout(width=700, height=500)
6
7 # Show the modified plot
8 fig.show()
9
```

Box Plot for NO2**Catatan/Penjelasan dari Output di atas:**

Grafik ini adalah representasi grafis dari distribusi data NO2. Grafik ini menunjukkan nilai minimum, lower fence, Q1, median, Q3, upper fence, dan maksimum untuk NO2.

- Nilai median untuk NO2 adalah 14. Nilai median adalah nilai tengah dari suatu set data.
- Nilai lower fence adalah 9 dan upper fence adalah 26. Nilai ini adalah batas bawah dan atas dari kuartil data, yang digunakan untuk mengidentifikasi outlier.
- Nilai minimum adalah 1 dan nilai maksimum adalah 148. Nilai ini adalah nilai terendah dan tertinggi dalam set data.
- Grafik juga menunjukkan beberapa outlier di atas upper fence. Outlier adalah nilai yang jauh dari sebagian besar nilai lain dalam set data.

Secara keseluruhan, grafik ini memberikan gambaran yang baik tentang bagaimana data NO2 tersebar dan dapat membantu dalam mengidentifikasi pola atau anomali dalam data.

Prompt chatGPT:

Tulis kode dalam Python menggunakan library Plotly Express untuk membuat box plot dari DataFrame 'dataudara'. Tetapkan sumbu y ('O3') dan judul ('Box Plot for O3'). Selanjutnya, perbarui tata letak plot untuk menetapkan lebar menjadi 700 dan tinggi menjadi 500. Terakhir, tampilkan plot yang telah dimodi

Penjelasan Kode:

1. `fig = px.box(dataudara, y='03', title='Box Plot for 03')`: Dalam baris ini, kita menggunakan Plotly Express (diwakili oleh `px`) untuk membuat sebuah box plot. `dataudara` adalah DataFrame yang digunakan untuk plotting, `y='03'` menentukan bahwa sumbu y dari box plot akan menggunakan data kolom '03' dari DataFrame tersebut, dan `title='Box Plot for 03'` memberikan judul pada plot tersebut.
2. `fig.update_layout(width=700, height=500)`: Baris ini digunakan untuk memperbarui tata letak (layout) dari plot. `width=700` dan `height=500` menetapkan lebar dan tinggi plot menjadi masing-masing 700 piksel dan 500 piksel.
3. `fig.show()`: Baris terakhir ini digunakan untuk menampilkan plot yang telah dimodifikasi. Setelah menjalankan semua baris kode di atas, Anda akan melihat box plot dari data '03' dengan lebar 700 piksel dan tinggi 500 piksel.

```
1 # Assuming 'dataudara' is your DataFrame
2 fig = px.box(dataudara, y='03', title='Box Plot for 03')
3
4 # Update layout to set width and height
5 fig.update_layout(width=700, height=500)
6
7 # Show the modified plot
8 fig.show()
9
```

Box Plot for O3

250

**Catatan/Penjelasan dari Output di atas:**

Grafik ini adalah representasi grafis dari distribusi data O3. Grafik ini menunjukkan nilai median, kuartil atas dan bawah, serta nilai minimum dan maksimum.

- Nilai median untuk O3 adalah 57. Nilai median adalah nilai tengah dari suatu set data.
- Kuartil atas adalah 81 dan kuartil bawah adalah 34. Kuartil adalah nilai yang membagi data menjadi empat bagian sama besar.
- Nilai minimum adalah 3 dan nilai maksimum adalah 243. Nilai ini adalah nilai terendah dan tertinggi dalam set data.
- Grafik juga menunjukkan beberapa outlier, yang merupakan titik-titik yang jatuh di luar pagar atas dan bawah. Pagar atas adalah 151 dan pagar bawah adalah 3. Outlier diwakili oleh titik-titik biru di atas pagar atas.

Secara keseluruhan, grafik ini memberikan gambaran yang baik tentang bagaimana data O3 tersebar dan dapat membantu dalam mengidentifikasi pola atau anomali dalam data.

▼ Data Transformation

Data transformation adalah proses mengubah data dari satu bentuk atau format ke bentuk atau format yang lain. Tujuan utama dari data transformation adalah membuat data lebih sesuai untuk analisis atau pengolahan selanjutnya. Proses ini dapat melibatkan beberapa tindakan, seperti penggabungan data, pemfilteran, pemformatan ulang, penghitungan agregat, dan lain sebagainya.

Berikut adalah beberapa konsep utama dalam data transformation:

1. **Pembersihan Data (Data Cleaning):** Proses ini melibatkan identifikasi dan penanganan nilai yang hilang, duplikat, atau tidak valid dalam dataset. Pembersihan data penting untuk memastikan bahwa hasil analisis atau pemodelan didasarkan pada data yang akurat dan dapat diandalkan.

2. **Pemfilteran (Filtering):** Pemfilteran melibatkan pemilihan subset dari data berdasarkan kriteria tertentu. Ini dapat melibatkan penghapusan baris atau kolom tertentu yang tidak diperlukan untuk analisis tertentu.
3. **Penggabungan Data (Data Merging):** Kadang-kadang, data yang relevan tersebar di beberapa sumber atau tabel. Penggabungan data adalah proses menggabungkan data dari beberapa sumber berbeda berdasarkan kunci atau kolom tertentu.
4. **Pemformatan Ulang (Reshaping):** Pemformatan ulang data melibatkan perubahan struktur data. Contohnya, mengubah data dari format panjang menjadi format lebar atau sebaliknya.
5. **Penghitungan Agregat (Aggregation):** Agregasi melibatkan penghitungan statistik ringkasan, seperti rata-rata, total, median, dan lain sebagainya, untuk satu atau lebih variabel. Ini sering digunakan ketika kita ingin melihat data secara keseluruhan daripada secara individu.
6. **Transformasi Variabel (Variable Transformation):** Transformasi variabel melibatkan perubahan skala atau distribusi variabel. Contohnya, mengaplikasikan transformasi logaritma pada variabel untuk mengurangi skewness.
7. **Normalisasi dan Standarisasi (Normalization and Standardization):** Ini adalah proses mengubah nilai-nilai variabel sehingga mereka berada pada skala yang seragam. Normalisasi sering digunakan dalam konteks machine learning untuk memastikan bahwa variabel-variabel dengan skala yang besar atau kecil tidak mendominasi proses pembelajaran.

Data transformation penting dalam analisis data dan pengolahan informasi karena membantu mengoptimalkan data untuk pemodelan, visualisasi, dan pengambilan keputusan. Transformasi data yang baik dapat meningkatkan kualitas analisis dan hasil akhir.

Prompt chatGPT:

Hitung frekuensi relatif dari kolom 'Critical' dalam dataframe 'dataudara', lalu kodekan kolom 'Critical' berdasarkan frekuensi relatif yang telah dihitung. Terakhir, tampilkan dataframe 'dataudara' yang telah diubah dengan penambahan kolom baru 'Critical Encoded'.

Penjelasan Kode:

1. Menghitung Frekuensi Relatif:

```
frequency_map = dataudara['Critical'].value_counts(normalize=True).to_dict()
```

Baris ini menghitung frekuensi relatif dari nilai yang muncul dalam kolom 'Critical' pada dataframe 'dataudara'.

`value_counts(normalize=True)` digunakan untuk menghitung frekuensi relatif, dan `to_dict()` mengonversi hasilnya menjadi bentuk kamus.

2. Mengkodekan Kolom 'Critical':

```
dataudara['Critical Encoded'] = dataudara['Critical'].map(frequency_map)
```

Pada baris ini, kolom baru dengan nama 'Critical Encoded' ditambahkan ke dataframe 'dataudara'. Kolom ini diisi dengan nilai-nilai yang dihasilkan dari pemetaan (mapping) nilai-nilai dalam kolom 'Critical' ke frekuensi relatif yang telah dihitung sebelumnya dan disimpan dalam `frequency_map`.

3. Menampilkan Dataframe yang Telah Diubah:

```
dataudara
```

Baris ini bertujuan untuk menampilkan dataframe 'dataudara' yang telah diubah dengan penambahan kolom baru 'Critical Encoded'. Hasilnya akan muncul sebagai output dari sel kode.

Dengan langkah-langkah tersebut, kita dapat menghitung frekuensi relatif dari kolom 'Critical' dan membuat kolom baru 'Critical Encoded' yang berisi nilai-nilai yang telah dikodekan berdasarkan frekuensi relatif tersebut.

```
1 # Menghitung frekuensi relatif dari kolom 'Critical' dalam dataframe 'dataudara'  
2 frequency_map = dataudara['Critical'].value_counts(normalize=True).to_dict()  
3  
4 # Mengkodekan kolom 'Critical' berdasarkan frekuensi relatif yang telah dihitung  
5 dataudara['Critical Encoded'] = dataudara['Critical'].map(frequency_map)  
6  
7 # Menampilkan dataframe 'dataudara' yang telah diubah dengan kolom baru 'Critical Encoded'  
8 dataudara  
9
```



	Tanggal	Wilayah	PM10	SO2	CO	O3	NO2	Max	Critical	Kategori	Critical Encoded
0	2017-01-01	DKI1 (Bunderan HI)	63.0	34.0	25.0	21.0	2.0	63.0	PM10	SEDANG	0.246410
1	2017-01-02	DKI1 (Bunderan HI)	16.0	32.0	16.0	19.0	1.0	32.0	SO2	BAIK	0.019989
2	2017-01-03	DKI1 (Bunderan HI)	18.0	32.0	17.0	20.0	2.0	32.0	SO2	BAIK	0.019989
3	2017-01-04	DKI1 (Bunderan HI)	24.0	32.0	22.0	20.0	2.0	32.0	SO2	BAIK	0.019989
4	2017-01-05	DKI1 (Bunderan HI)	29.0	32.0	25.0	14.0	2.0	32.0	SO2	BAIK	0.019989
...
8820	2021-10-27	DKI5 (Kebon Jeruk) Jakarta Barat	53.0	32.6	11.0	32.0	17.0	76.0	PM25	SEDANG	0.180701
8821	2021-10-28	DKI5 (Kebon Jeruk) Jakarta Barat	51.0	26.8	16.0	29.0	27.0	77.0	PM25	SEDANG	0.180701
8822	2021-10-29	DKI5 (Kebon Jeruk) Jakarta Barat	50.0	23.8	18.0	23.0	33.0	70.0	PM25	SEDANG	0.180701
8823	2021-10-30	DKI5 (Kebon Jeruk) Jakarta Barat	50.0	28.6	12.0	30.0	24.0	74.0	PM25	SEDANG	0.180701

Catatan/Penjelasan dari Output di atas:

Output di atas adalah tampilan dari dataframe 'dataudara' setelah melakukan beberapa operasi, seperti menghitung frekuensi relatif dari kolom 'Critical' dan mengkodekan kolom 'Critical' berdasarkan frekuensi relatif tersebut. Berikut adalah catatan/penjelasan dari output tersebut:

- Kolom 'Tanggal':** Menunjukkan tanggal pengukuran data udara.
- Kolom 'Wilayah':** Menyatakan wilayah di mana pengukuran dilakukan.
- Kolom 'PM10', 'SO2', 'CO', 'O3', 'NO2':** Merupakan nilai konsentrasi dari masing-masing jenis polutan udara pada suatu tanggal dan wilayah tertentu.
- Kolom 'Max':** Merupakan nilai maksimum dari konsentrasi polutan udara di antara PM10, SO2, CO, O3, dan NO2 pada suatu pengukuran.
- Kolom 'Critical':** Menunjukkan jenis polutan yang memiliki konsentrasi tertinggi pada suatu pengukuran.
- Kolom 'Kategori':** Menunjukkan kategori tingkat pencemaran udara berdasarkan konsentrasi polutan tertinggi pada suatu pengukuran.
- Kolom 'Critical Encoded':** Merupakan kolom baru yang berisi nilai-nilai yang telah dikodekan berdasarkan frekuensi relatif dari kolom 'Critical'. Ini memberikan representasi numerik untuk jenis polutan yang dominan pada setiap pengukuran.

Dengan adanya kolom 'Critical Encoded', kita dapat memanfaatkannya untuk analisis lebih lanjut dengan menggunakan representasi numerik yang mencerminkan tingkat kepentingan relatif dari jenis polutan pada setiap entri data.

Prompt chatGPT:

Berikan kode Python dalam bentuk plaintext untuk menampilkan nama kolom dalam DataFrame dengan nama 'dataudara'. Gunakan fungsi 'print' dan pastikan menyertakan sintaks yang benar untuk mengakses nama kolom dalam DataFrame.

Penjelasan Kode:

1. `print(dataudara.columns)`: Ini adalah baris pertama dalam kode. Fungsi `print()` digunakan untuk mencetak output ke konsol atau output yang ditentukan. Pada kasus ini, kita mencetak nama kolom dari DataFrame 'dataudara'. `dataudara.columns` mengakses atribut kolom dari DataFrame 'dataudara', yang berisi daftar nama-nama kolom dalam DataFrame tersebut. Sehingga, baris ini mencetak daftar nama kolom ke konsol.

```
1 # Check the column names in your DataFrame
2 print(dataudara.columns)
3

Index(['Tanggal', 'Wilayah', 'PM10', 'SO2', 'CO', 'O3', 'NO2', 'Max',
       'Critical', 'Kategori', 'Critical Encoded'],
      dtype='object')
```

Catatan/Penjelasan dari Output di atas:

Output di atas menunjukkan index atau daftar nama kolom dalam DataFrame. Berikut adalah penjelasan untuk setiap kolom:

1. **Tanggal**: Kolom ini mewakili tanggal pengukuran data udara.
2. **Wilayah**: Kolom ini berisi informasi tentang wilayah atau lokasi di mana pengukuran udara dilakukan.
3. **PM10**: Merupakan konsentrasi Particulate Matter (PM10) yang diukur dalam udara. PM10 adalah partikel-partikel kecil yang memiliki diameter kurang dari 10 mikrometer.
4. **SO2**: Merupakan konsentrasi sulfur dioksida (SO2) dalam udara. SO2 adalah gas yang dapat berasal dari aktivitas pembakaran bahan bakar fosil.
5. **CO**: Merupakan konsentrasi karbon monoksida (CO) dalam udara. CO dapat berasal dari pembakaran tidak lengkap bahan bakar.
6. **O3**: Merupakan konsentrasi ozon (O3) dalam udara. Ozon dapat terbentuk sebagai polutan udara.

7. **NO2**: Merupakan konsentrasi nitrogen dioksida (NO₂) dalam udara. NO₂ dapat berasal dari berbagai sumber termasuk kendaraan bermotor.
8. **Max**: Kolom ini mungkin berisi nilai maksimum dari parameter-parameter tertentu dalam suatu pengukuran.
9. **Critical**: Mungkin mengindikasikan apakah nilai suatu parameter kritis atau tidak.
10. **Kategori**: Kolom ini mungkin berisi kategori atau label tertentu yang diberikan berdasarkan nilai parameter tertentu.
11. **Critical Encoded**: Mungkin berisi nilai terenkripsi yang mengindikasikan apakah suatu parameter kritis atau tidak dalam bentuk terenkripsi.

Setiap kolom tersebut memberikan informasi tentang parameter-parameter yang diukur dan variabel-variabel yang terlibat dalam dataset pengukuran kualitas udara.

Prompt chatGPT:

Tulislah kode Python untuk mengimpor kelas LabelEncoder dari pustaka sklearn.preprocessing, membuat objek LabelEncoder, menggunakan LabelEncoder untuk mengkodekan kategori pada kolom 'Kategori', dan menyimpan hasilnya dalam kolom baru bernama 'CategoryEncoded' pada dataframe 'dataudara'. Terakhir, tampilkan dataframe 'dataudara' setelah proses pengkodean.

Penjelasan Kode:

1. **Mengimport kelas LabelEncoder dari pustaka sklearn.preprocessing:**
 - Pada baris pertama, dilakukan impor kelas `LabelEncoder` dari pustaka `sklearn.preprocessing`. Kelas ini digunakan untuk mengkodekan label kategori menjadi nilai numerik.
2. **Membuat objek LabelEncoder:**
 - Baris kedua membuat objek `le` yang merupakan instance dari kelas `LabelEncoder()`. Objek ini akan digunakan untuk melakukan transformasi label kategori menjadi nilai numerik.
3. **Menggunakan LabelEncoder untuk mengkodekan kategori pada kolom 'Kategori' dan menyimpannya dalam kolom baru 'CategoryEncoded':**
 - Baris ketiga menggunakan metode `fit_transform` dari objek `le` untuk mengkodekan nilai pada kolom 'Kategori' dalam dataframe `dataudara`. Hasilnya disimpan dalam kolom baru yang disebut 'CategoryEncoded' pada dataframe `dataudara`. Proses ini mengubah

label kategori menjadi nilai numerik.

4. Menampilkan dataframe 'dataudara' setelah proses pengkodean:

- Baris keempat menampilkan dataframe `dataudara` setelah proses pengkodean dilakukan. Ini memungkinkan untuk melihat perubahan yang terjadi setelah label kategori dikodekan menjadi nilai numerik dan disimpan dalam kolom 'CategoryEncoded'.

```
1 # Mengimport kelas LabelEncoder dari pustaka sklearn.preprocessing
2 from sklearn.preprocessing import LabelEncoder
3
4 # Membuat objek LabelEncoder
5 le = LabelEncoder()
6
7 # Menggunakan LabelEncoder untuk mengkodekan kategori pada kolom 'Kategori' dan menyimpannya dalam kolom baru 'CategoryEncoded'
8 dataudara['CategoryEncoded'] = le.fit_transform(dataudara['Kategori'])
9
10 # Menampilkan dataframe 'dataudara' setelah proses pengkodean
11 dataudara
12
```



Tanggal

Wilayah DM10 SO2 CO NO2 Max Critical Kategori Critical Encoded CategoryEncoded

Catatan/Penjelasan dari Output di atas:

Dari output dataframe yang diberikan, berikut adalah catatan/penjelasan:

1. Kolom 'Tanggal':

- Menunjukkan tanggal pengukuran kualitas udara.

2. Kolom 'Wilayah':

- Merupakan lokasi pengukuran kualitas udara, seperti "DKI1 (Bunderan HI)" atau "DKI5 (Kebon Jeruk) Jakarta Barat".

3. Kolom 'PM10', 'SO2', 'CO', 'O3', 'NO2':

- Merupakan nilai pengukuran untuk masing-masing jenis polutan udara. Nilai-nilai ini mewakili konsentrasi polutan dalam udara.

4. Kolom 'Max':

- Merupakan nilai maksimum dari polutan tertentu pada suatu tanggal.

5. Kolom 'Critical' dan 'Kategori':

- 'Critical' menunjukkan jenis polutan yang memiliki nilai maksimum pada suatu tanggal.
- 'Kategori' memberikan informasi kategori kualitas udara berdasarkan nilai indeks tertentu (seperti 'BAIK' atau 'SEDANG').

6. Kolom 'Critical Encoded' dan 'CategoryEncoded':

- 'Critical Encoded' dan 'CategoryEncoded' adalah kolom yang menyimpan nilai numerik yang mewakili informasi kritis dan kategori kualitas udara, masing-masing. Proses pengkodean ini dilakukan menggunakan LabelEncoder, seperti yang dijelaskan dalam kode sebelumnya.

Catatan tambahan:

- Dataframe memiliki total 8825 baris dan 12 kolom.
- Terdapat variasi dalam kategori kualitas udara, seperti 'BAIK' dan 'SEDANG'.
- Data ini mencakup rentang waktu dari tahun 2017 hingga Oktober 2021.

Prompt chatGPT:

Tuliskan kode dalam bahasa Python untuk memeriksa apakah kolom 'CategoryEncoded' ada dalam DataFrame yang disebut 'dataudara'. Jika kolom tersebut ada, konversi tipe datanya menjadi 'float64' dan cetak pesan yang sesuai. Jika tidak, cetak pesan bahwa kolom tersebut tidak ditemukan.

Penjelasan Kode:

1. `if 'CategoryEncoded' in dataudara.columns:` : Baris ini melakukan pemeriksaan apakah kolom dengan nama 'CategoryEncoded' ada di dalam DataFrame 'dataudara'. Pemeriksaan ini menggunakan pernyataan `in` untuk memeriksa apakah nama kolom tersebut terdapat dalam daftar kolom DataFrame.
2. `dataudara['CategoryEncoded'] = dataudara['CategoryEncoded'].astype('float64')` : Jika kolom 'CategoryEncoded' ditemukan, baris ini mengkonversi tipe data pada kolom tersebut menjadi float64 menggunakan metode `.astype('float64')`. Ini bermanfaat jika kolom 'CategoryEncoded' sebelumnya memiliki tipe data yang berbeda.
3. `print("Kolom 'CategoryEncoded' ditemukan dalam DataFrame.")` : Jika kolom 'CategoryEncoded' ada, pernyataan ini akan mencetak pesan yang memberi tahu bahwa kolom tersebut ditemukan dalam DataFrame.
4. `else:` : Jika kondisi pada baris pertama tidak terpenuhi (artinya, kolom 'CategoryEncoded' tidak ditemukan),
5. `print("Kolom 'CategoryEncoded' tidak ditemukan dalam DataFrame.")` : Pernyataan ini akan mencetak pesan yang memberi tahu bahwa kolom 'CategoryEncoded' tidak ditemukan dalam DataFrame.

```
1 # Periksa apakah 'CategoryEncoded' ada di kolom-kolom
2 if 'CategoryEncoded' in dataudara.columns:
3     # Lakukan konversi tipe data
4     dataudara['CategoryEncoded'] = dataudara['CategoryEncoded'].astype('float64')
5     print("Kolom 'CategoryEncoded' ditemukan dalam DataFrame.")
6 else:
7     print("Kolom 'CategoryEncoded' tidak ditemukan dalam DataFrame.")
8
9
```

Kolom 'CategoryEncoded' ditemukan dalam DataFrame.

Catatan/Penjelasan dari Output di atas:

Output "Kolom 'CategoryEncoded' ditemukan dalam DataFrame." menunjukkan bahwa kolom dengan nama 'CategoryEncoded' ada di dalam DataFrame yang disebut 'dataudara'. Oleh karena itu, kode tersebut berhasil menemukan kolom yang dicari dan kemudian melakukan konversi tipe data kolom tersebut menjadi float64.

Catatan: Output ini hanya akan muncul jika kolom 'CategoryEncoded' ada dalam DataFrame. Jika kolom tersebut tidak ditemukan, maka output yang sesuai akan dicetak berdasarkan blok else dalam struktur pengkondisian.

Prompt chatGPT:

Tampilkan informasi terbaru DataFrame dengan menggunakan perintah print(dataudara.info()).

Penjelasan Kode:

```
# Display the updated DataFrame information
print(dataudara.info())
```

Penjelasan baris kode di atas:

1. `# Display the updated DataFrame information`: Ini adalah komentar dalam kode Python dan tidak memiliki efek eksekusi. Komentar digunakan untuk memberikan penjelasan atau catatan terkait dengan kode yang mengikuti komentar ini.
2. `print(dataudara.info())`: Ini adalah perintah untuk mencetak informasi terkini tentang DataFrame `dataudara`. `dataudara.info()` digunakan untuk mendapatkan informasi tentang struktur DataFrame, termasuk jumlah entri (baris), jumlah kolom, tipe data setiap kolom, dan jumlah nilai non-null dalam setiap kolom. Fungsi `print()` digunakan untuk menampilkan informasi ini di konsol atau output program.

```
1 # Display the updated DataFrame information
2 print(dataudara.info())
3

<class 'pandas.core.frame.DataFrame'\>
Int64Index: 8705 entries, 0 to 8824
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   Tanggal          8705 non-null   datetime64[ns]
 1   Wilayah          8705 non-null   object
```

```
2  PM10           8705 non-null  float64
3  S02           8705 non-null  float64
4  CO            8705 non-null  float64
5  O3            8705 non-null  float64
6  NO2           8705 non-null  float64
7  Max           8705 non-null  float64
8  Critical       8705 non-null  object
9  Kategori       8705 non-null  object
10 Critical Encoded 8705 non-null  float64
11 CategoryEncoded 8705 non-null  float64
dtypes: datetime64[ns](1), float64(8), object(3)
memory usage: 884.1+ KB
None
```

Catatan/Penjelasan dari Output di atas:

Output di atas adalah hasil dari pemanggilan `info()` pada DataFrame `dataudara`. Berikut adalah penjelasan dari setiap bagian output:

1. `<class 'pandas.core.frame.DataFrame'>`: Ini menunjukkan bahwa objek tersebut adalah instance dari kelas DataFrame dari pustaka Pandas.
2. `Int64Index: 8705 entries, 0 to 8824`: Menunjukkan bahwa indeks DataFrame ini adalah tipe data `Int64Index` dengan total 8705 entri, dan indeksnya berkisar dari 0 hingga 8824.
3. `Data columns (total 12 columns)`: Menginformasikan bahwa DataFrame ini memiliki 12 kolom.
4. `# Column ... Dtype`: Bagian ini memberikan rincian untuk setiap kolom.
 - `Tanggal`: Tipe data kolom ini adalah `datetime64[ns]`.
 - `Wilayah`: Tipe data kolom ini adalah `object`.
 - `PM10, S02, CO, O3, NO2, Max, Critical Encoded, dan CategoryEncoded`: Semua kolom ini memiliki tipe data `float64`.
 - `Critical` dan `Kategori`: Kolom ini memiliki tipe data `object`.
5. `Non-Null Count`: Menunjukkan jumlah nilai non-null dalam setiap kolom. Semua kolom memiliki 8705 nilai non-null, menunjukkan tidak ada nilai yang hilang.
6. `memory usage: 884.1+ KB`: Menunjukkan penggunaan memori oleh DataFrame ini.
7. `None`: Ini merupakan nilai yang dikembalikan oleh `print(dataudara.info())`, menandakan bahwa tidak ada nilai yang dikembalikan atau dihasilkan dari pernyataan print ini.

Prompt chatGPT:

Tuliskan kode menggunakan Pandas untuk mendapatkan matriks korelasi antar kolom pada DataFrame yang disebut 'dataudara' menggunakan fungsi corr().

Penjelasan Kode:**1. dataudara.corr():**

- `dataudara` : Merupakan nama DataFrame yang telah dibuat sebelumnya.
- `.corr()` : Adalah metode Pandas yang digunakan untuk menghitung korelasi antar kolom-kolom dalam DataFrame. Korelasi adalah ukuran statistik yang mengukur sejauh mana dua variabel berkaitan satu sama lain.

2. Penjelasan umum:

- Kode tersebut bertujuan untuk menghasilkan matriks korelasi antar kolom-kolom dalam DataFrame 'dataudara'.

3. Perhatian:

- Pastikan DataFrame 'dataudara' sudah didefinisikan dan berisi data numerik, karena fungsi korelasi biasanya hanya dapat dihitung untuk data numerik.

Kode tersebut sangat berguna dalam analisis data untuk memahami sejauh mana hubungan antar variabel dalam DataFrame, yang dapat membantu dalam pemahaman pola dan tren dalam dataset.

```
1 # Jika menggunakan Pandas dan memiliki DataFrame bernama dataudara, berikut adalah contoh penggunaan fungsi corr() untuk mendapatkan korelasi ant
2
3 dataudara.corr()
4
```

```
<ipython-input-32-c60163768185>:3: FutureWarning:
```

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or

	PM10	SO2	CO	O3	NO2	Max	Critical	Encoded	CategoryEncoded
PM10	1.000000	0.168343	0.097841	0.345780	0.177235	0.521499	0.031805	0.616354	
SO2	0.168343	1.000000	-0.144530	-0.175539	0.338922	0.034402	-0.338530	0.146605	
CO	0.097841	-0.144530	1.000000	0.040232	0.376311	-0.052302	0.007513	0.098655	
O3	0.345780	-0.175539	0.040232	1.000000	-0.234030	0.808419	0.643107	0.511922	

Catatan/Penjelasan dari Output di atas:

Matriks korelasi di atas menunjukkan koefisien korelasi antar kolom-kolom dalam suatu DataFrame. Berikut adalah catatan dan penjelasan dari output tersebut:

1. PM10 vs. Kolom Lainnya:

- Korelasi PM10 terhadap SO2 (0.168343), CO (0.097841), O3 (0.345780), NO2 (0.177235), Max (0.521499), Critical Encoded (0.031805), dan CategoryEncoded (0.616354).
- Nilai korelasi yang positif menunjukkan hubungan positif, sedangkan nilai negatif menunjukkan hubungan negatif. Semakin dekat nilai korelasi dengan 1 (positif) atau -1 (negatif), semakin kuat hubungan antar variabelnya.

2. SO2 vs. Kolom Lainnya:

- Korelasi SO2 terhadap PM10 (0.168343), CO (-0.144530), O3 (-0.175539), NO2 (0.338922), Max (0.034402), Critical Encoded (-0.338530), dan CategoryEncoded (0.146605).

3. CO vs. Kolom Lainnya:

- Korelasi CO terhadap PM10 (0.097841), SO2 (-0.144530), O3 (0.040232), NO2 (0.376311), Max (-0.052302), Critical Encoded (0.007513), dan CategoryEncoded (0.098655).

4. O3 vs. Kolom Lainnya:

- Korelasi O3 terhadap PM10 (0.345780), SO2 (-0.175539), CO (0.040232), NO2 (-0.234030), Max (0.808419), Critical Encoded (0.643107), dan CategoryEncoded (0.511922).

5. NO2 vs. Kolom Lainnya:

- Korelasi NO2 terhadap PM10 (0.177235), SO2 (0.338922), CO (0.376311), O3 (-0.234030), Max (-0.070487), Critical Encoded (-0.271178), dan CategoryEncoded (0.204546).

6. Max vs. Kolom Lainnya:

- Korelasi Max terhadap PM10 (0.521499), SO2 (0.034402), CO (-0.052302), O3 (0.808419), NO2 (-0.070487), Critical Encoded (0.335234), dan CategoryEncoded (0.682435).

7. Critical Encoded vs. Kolom Lainnya:

- Korelasi Critical Encoded terhadap PM10 (0.031805), SO2 (-0.338530), CO (0.007513), O3 (0.643107), NO2 (-0.271178), Max (0.335234), dan CategoryEncoded (0.204700).

8. CategoryEncoded vs. Kolom Lainnya:

- Korelasi CategoryEncoded terhadap PM10 (0.616354), SO2 (0.146605), CO (0.098655), O3 (0.511922), NO2 (0.204546), Max (0.682435), dan Critical Encoded (0.204700).

Catatan: Interpretasi korelasi bergantung pada konteks dan jenis data yang diamati. Korelasi tidak menyiratkan sebab-akibat, dan hanya mengukur sejauh mana dua variabel bergerak bersamaan.

Prompt chatGPT:

Tampilkan isi data udara dengan perintah: dataudara

Penjelasan Kode:

1. # Menampilkan isi datanya : Ini adalah komentar dalam kode yang memberikan informasi singkat tentang tujuan atau fungsi kode berikutnya. Pada kasus ini, tujuannya adalah menampilkan isi data udara.
2. dataudara : Ini adalah perintah atau instruksi untuk menampilkan isi data udara. Perintah ini mungkin merujuk pada suatu variabel atau fungsi yang berisi informasi terkait kualitas udara atau parameter udara lainnya.

Jadi, secara keseluruhan, kode ini memberikan instruksi untuk menampilkan isi data udara dengan menggunakan perintah dataudara . Namun, tanpa informasi lebih lanjut tentang implementasi atau konteksnya, sulit untuk memberikan penjelasan yang lebih rinci.

```
1 # Menampilkan isi datanya  
2 dataudara
```

	Tanggal	Wilayah	PM10	SO2	CO	O3	NO2	Max	Critical	Kategori	Critical Encoded	CategoryEncoded
0	2017-01-01	DKI1 (Bunderan HI)	63.0	34.0	25.0	21.0	2.0	63.0	PM10	SEDANG	0.246410	2.0
1	2017-01-02	DKI1 (Bunderan HI)	16.0	32.0	16.0	19.0	1.0	32.0	SO2	BAIK	0.019989	0.0
2	2017-01-03	DKI1 (Bunderan HI)	18.0	32.0	17.0	20.0	2.0	32.0	SO2	BAIK	0.019989	0.0
3	2017-01-04	DKI1 (Bunderan HI)	24.0	32.0	22.0	20.0	2.0	32.0	SO2	BAIK	0.019989	0.0
4	2017-01-05	DKI1 (Bunderan HI)	29.0	32.0	25.0	14.0	2.0	32.0	SO2	BAIK	0.019989	0.0
...
8820	2021-10-27	DKI5 (Kebon Jeruk) Jakarta Barat	53.0	32.6	11.0	32.0	17.0	76.0	PM25	SEDANG	0.180701	2.0
8821	2021-10-28	DKI5 (Kebon Jeruk) Jakarta Barat	51.0	26.8	16.0	29.0	27.0	77.0	PM25	SEDANG	0.180701	2.0
8822	2021-10-29	DKI5 (Kebon Jeruk) Jakarta Barat	50.0	23.8	18.0	23.0	33.0	70.0	PM25	SEDANG	0.180701	2.0
8823	2021-10-30	DKI5 (Kebon Jeruk) Jakarta Barat	50.0	28.6	12.0	30.0	24.0	74.0	PM25	SEDANG	0.180701	2.0
8824	2021-10-31	DKI5 (Kebon Jeruk) Jakarta Barat	49.0	28.4	10.0	31.0	13.0	74.0	PM25	SEDANG	0.180701	2.0

8705 rows × 12 columns

Catatan/Penjelasan dari Output di atas:

- Tanggal:** Kolom ini menunjukkan tanggal pengukuran data kualitas udara.
- Wilayah:** Kolom ini memberikan informasi tentang wilayah atau lokasi di mana pengukuran dilakukan, misalnya, "DKI1 (Bunderan HI) Jakarta Pusat" atau "DKI5 (Kebon Jeruk) Jakarta Barat".
- PM10, SO2, CO, O3, NO2:** Kolom-kolom ini berisi nilai pengukuran untuk berbagai parameter kualitas udara, seperti Particulate Matter (PM10), Sulfur Dioxide (SO2), Carbon Monoxide (CO), Ozone (O3), dan Nitrogen Dioxide (NO2).
- Max:** Kolom ini menunjukkan nilai maksimum dari parameter kualitas udara pada suatu tanggal tertentu.
- Critical:** Kolom ini memberikan informasi tentang parameter kualitas udara yang memiliki nilai paling tinggi pada suatu tanggal tertentu.
- Kategori:** Kolom ini menyajikan kategori kualitas udara berdasarkan nilai parameter tertentu pada suatu tanggal. Misalnya, "BAIK" atau "SEDANG".
- Critical Encoded:** Ini mungkin merupakan representasi numerik dari kolom "Critical" yang dapat digunakan dalam analisis data.

8. **Category Encoded:** Ini mungkin merupakan representasi numerik dari kolom "Kategori" yang dapat digunakan dalam analisis data.

9. **8705 rows × 12 columns:** Ini menunjukkan bahwa dataset ini terdiri dari 8.705 baris dan 12 kolom.

Prompt chatGPT:

Hapus kolom 'Tanggal', 'Wilayah', 'Critical', dan 'Kategori' dari DataFrame 'dataudara' dan tampilkan DataFrame yang telah diperbarui. Gunakan kode Python untuk melakukan hal ini.

Penjelasan Kode:

1. `columns_to_drop = ['Tanggal', 'Wilayah', 'Critical', 'Kategori']`: Di baris ini, kita mendefinisikan sebuah list yang disebut `columns_to_drop`. List ini berisi nama-nama kolom yang ingin kita hapus dari DataFrame.
2. `dataudara = dataudara.drop(columns=columns_to_drop)`: Di sini, kita menggunakan metode `drop` pada DataFrame `dataudara` untuk menghapus kolom-kolom yang telah kita tentukan sebelumnya dalam `columns_to_drop`. Parameter `columns` digunakan untuk menyebutkan kolom-kolom yang akan dihapus.
3. `print(dataudara.head())`: Baris ini bertujuan untuk menampilkan lima baris pertama dari DataFrame yang telah diperbarui. Fungsi `head()` digunakan untuk menampilkan sejumlah baris teratas dari DataFrame. Dalam hal ini, lima baris pertama dari DataFrame `dataudara` setelah kolom-kolom tertentu dihapus akan ditampilkan.

```
1 # Assuming 'dataudara' is your DataFrame
2 columns_to_drop = ['Tanggal', 'Wilayah', 'Critical', 'Kategori']
3 dataudara = dataudara.drop(columns=columns_to_drop)
4
5 # Display the updated DataFrame
6 print(dataudara.head())
7
```

	PM10	S02	CO	O3	N02	Max	Critical	Encoded	CategoryEncoded
0	63.0	34.0	25.0	21.0	2.0	63.0	0.246410	2.0	
1	16.0	32.0	16.0	19.0	1.0	32.0	0.019989	0.0	
2	18.0	32.0	17.0	20.0	2.0	32.0	0.019989	0.0	
3	24.0	32.0	22.0	20.0	2.0	32.0	0.019989	0.0	
4	29.0	32.0	25.0	14.0	2.0	32.0	0.019989	0.0	

Catatan/Penjelasan dari Output di atas:

Catatan/Penjelasan dari Output:

DataFrame di atas mencakup data terkait kualitas udara dengan beberapa parameter yang diukur. Berikut adalah penjelasan untuk setiap kolom:

1. **PM10**: Merupakan nilai partikulat matter 10 mikrometer atau kurang dalam udara, diukur dalam mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
2. **SO2 (Sulfur Dioxide)**: Menunjukkan konsentrasi sulfur dioksida dalam udara, diukur dalam mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
3. **CO (Carbon Monoxide)**: Merepresentasikan konsentrasi karbon monoksida dalam udara, diukur dalam mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
4. **O3 (Ozone)**: Menunjukkan konsentrasi ozon dalam udara, diukur dalam mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
5. **NO2 (Nitrogen Dioxide)**: Merupakan nilai konsentrasi nitrogen dioksida dalam udara, diukur dalam mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
6. **Max**: Merupakan nilai maksimum dari parameter-parameter sebelumnya dalam satu baris tertentu.
7. **Critical Encoded**: Nilai numerik yang mewakili pengkodean dari status kritis (critical) suatu parameter.
8. **CategoryEncoded**: Nilai numerik yang mewakili pengkodean dari kategori suatu parameter.

Setiap baris dalam DataFrame ini menunjukkan pengukuran kualitas udara pada suatu waktu tertentu. Data ini dapat digunakan untuk analisis dan pemodelan terkait kualitas udara.

Prompt chatGPT:

Buatlah perintah prompt yang menghasilkan kode untuk menampilkan informasi terkini DataFrame dengan menggunakan metode `info()`.

Penjelasan Kode:

1. `# Display the updated DataFrame information`: Ini adalah komentar dalam kode Python yang digunakan untuk memberikan penjelasan atau keterangan terhadap baris kode atau blok kode di bawahnya. Komentar diawali dengan tanda pagar (#) dan tidak memengaruhi eksekusi program.
2. `dataudara.info()`: Ini adalah panggilan metode pada objek DataFrame yang disebut `dataudara`. Metode `info()` digunakan untuk menampilkan informasi tentang DataFrame, termasuk jumlah entri, jenis data setiap kolom, jumlah memori yang digunakan, dan

sebagainya. Informasi ini berguna untuk mendapatkan wawasan cepat tentang struktur dan sifat data dalam DataFrame.

Penjelasan tersebut memberikan konteks tentang tujuan dan fungsionalitas dari dua baris kode yang diberikan.

```
1 # Display the updated DataFrame information  
2 dataudara.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 8705 entries, 0 to 8824  
Data columns (total 8 columns):  
 #   Column           Non-Null Count  Dtype     
---  --  
 0   PM10            8705 non-null    float64  
 1   SO2              8705 non-null    float64  
 2   CO               8705 non-null    float64  
 3   O3               8705 non-null    float64  
 4   NO2              8705 non-null    float64  
 5   Max              8705 non-null    float64  
 6   Critical Encoded 8705 non-null    float64  
 7   CategoryEncoded 8705 non-null    float64  
dtypes: float64(8)  
memory usage: 612.1 KB
```

Catatan/Penjelasan dari Output di atas:

Output di atas adalah hasil dari pemanggilan metode `info()` pada objek DataFrame. Berikut adalah catatan dan penjelasan dari output tersebut:

1. **Class 'pandas.core.frame.DataFrame'**: Ini menunjukkan bahwa objek ini adalah instance dari kelas DataFrame dari pustaka Pandas, yang digunakan untuk mengelola data dalam bentuk tabel.
2. **Int64Index: 8705 entries, 0 to 8824**: Ini memberikan informasi tentang indeks DataFrame. Int64Index menunjukkan tipe indeks yang digunakan (integer 64-bit), dan terdapat 8705 entri mulai dari indeks 0 hingga 8824.
3. **Data columns (total 8 columns)**: Menunjukkan bahwa terdapat 8 kolom dalam DataFrame.
4. **Columns Information (with Non-Null Count and Dtype)**:
 - **Column 0 (PM10)**: Terdapat 8705 non-null entries dengan tipe data float64.
 - **Column 1 (SO2)**: Terdapat 8705 non-null entries dengan tipe data float64.
 - **Column 2 (CO)**: Terdapat 8705 non-null entries dengan tipe data float64.

- **Column 3 (O3)**: Terdapat 8705 non-null entries dengan tipe data float64.
- **Column 4 (NO2)**: Terdapat 8705 non-null entries dengan tipe data float64.
- **Column 5 (Max)**: Terdapat 8705 non-null entries dengan tipe data float64.
- **Column 6 (Critical Encoded)**: Terdapat 8705 non-null entries dengan tipe data float64.
- **Column 7 (Category Encoded)**: Terdapat 8705 non-null entries dengan tipe data float64.

5. **dtypes: float64(8)**: Menampilkan jumlah kolom untuk setiap jenis tipe data, dalam hal ini, terdapat 8 kolom dengan tipe data float64.

6. **memory usage: 612.1 KB**: Menunjukkan sekitar 612.1 kilobyte memori yang digunakan oleh DataFrame untuk menyimpan data. Informasi ini berguna untuk pemantauan penggunaan memori, terutama pada dataset besar.

Prompt chatGPT:

Buatlah matriks korelasi dari dataudara dan gambarkan heatmap segitiga termasuk diagonal menggunakan Seaborn. Gunakan matplotlib untuk menampilkan plot. Pastikan matriks korelasi dijelaskan dengan annotasi dan menggunakan skala warna 'coolwarm'. Gunakan ukuran f

Penjelasan Kode:

1. `correlation_matrix = dataudara.corr()`: Baris ini menghitung matriks korelasi dari DataFrame `dataudara`. Fungsi `corr()` digunakan untuk menghasilkan matriks korelasi antar kolom dalam DataFrame.
2. `plt.figure(figsize=(8, 6))`: Baris ini menyiapkan figur untuk plot menggunakan Matplotlib. `figsize=(8, 6)` mengatur ukuran figur menjadi 8 inci lebar dan 6 inci tinggi.
3. `sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)`: Baris ini membuat heatmap menggunakan Seaborn. `correlation_matrix` adalah matriks yang telah dihitung sebelumnya. Parameter `annot=True` menambahkan nilai pada sel heatmap. `cmap='coolwarm'` menentukan skema warna yang digunakan. `fmt=".2f"` memformat nilai dalam sel sebagai angka desimal dengan dua digit di belakang koma. `linewidths=.5` mengatur lebar garis antar sel dalam heatmap.
4. `plt.title('Matriks Korelasi')`: Baris ini menambahkan judul pada plot, dengan teks "Matriks Korelasi".
5. `plt.show()`: Baris ini menampilkan plot yang telah disiapkan sebelumnya menggunakan Matplotlib.

```
1 # Menghitung matriks korelasi
2 correlation_matrix = dataudara.corr()
3
```

```
4 # Menyiapkan figur matplotlib
5 plt.figure(figsize=(8, 6))
6
7 # Membuat heatmap segitiga termasuk diagonal menggunakan Seaborn
8 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
9
10 # Menampilkan plot
11 plt.title('Matriks Korelasi')
12 plt.show()
13
```



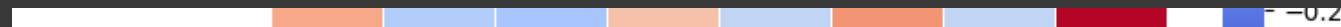
Catatan/Penjelasan dari Output di atas:

Matriks korelasi yang ditampilkan dalam bentuk heatmap ini memberikan gambaran tentang hubungan antara variabel-variabel dalam dataset. Nilai dalam matriks berkisar dari -1 hingga 1, dengan -1 menunjukkan korelasi negatif, 0 menunjukkan tidak ada korelasi, dan 1 menunjukkan korelasi positif.

Warna dalam heatmap berkisar dari merah hingga biru. Merah menunjukkan korelasi negatif, sedangkan biru menunjukkan korelasi positif. Nilai diagonal dalam matriks adalah 1, karena mereka mewakili korelasi variabel dengan dirinya sendiri.

Angka-angka yang ditampilkan dalam heatmap adalah koefisien korelasi antara pasangan variabel. Misalnya, jika dua variabel memiliki koefisien korelasi 0.8, ini menunjukkan bahwa mereka memiliki korelasi positif yang kuat. Sebaliknya, jika dua variabel memiliki koefisien korelasi -0.8, ini menunjukkan bahwa mereka memiliki korelasi negatif yang kuat.

Dengan memahami korelasi antara variabel, dapat membantu dalam analisis data lebih lanjut, seperti pemilihan fitur atau pemahaman hubungan antara variabel. Namun, perlu diingat bahwa korelasi tidak menyiratkan sebab akibat. Variabel yang berkorelasi mungkin memiliki hubungan sebab akibat, atau mereka mungkin dipengaruhi oleh variabel ketiga yang tidak diamati. Selain itu, korelasi yang kuat antara dua variabel tidak selalu berarti bahwa satu variabel dapat diprediksi dengan baik dari variabel lainnya.



Prompt chatGPT:

Tuliskan kode untuk menghitung jumlah masing-masing nilai dalam kolom 'CategoryEncoded', lalu plot distribusinya menggunakan diagram batang. Sertakan judul untuk diagram, label sumbu x sebagai 'CategoryEncoded', label sumbu y sebagai 'Jumlah', dan tampilkan nilai jumlah di atas setiap batang dalam diagram.

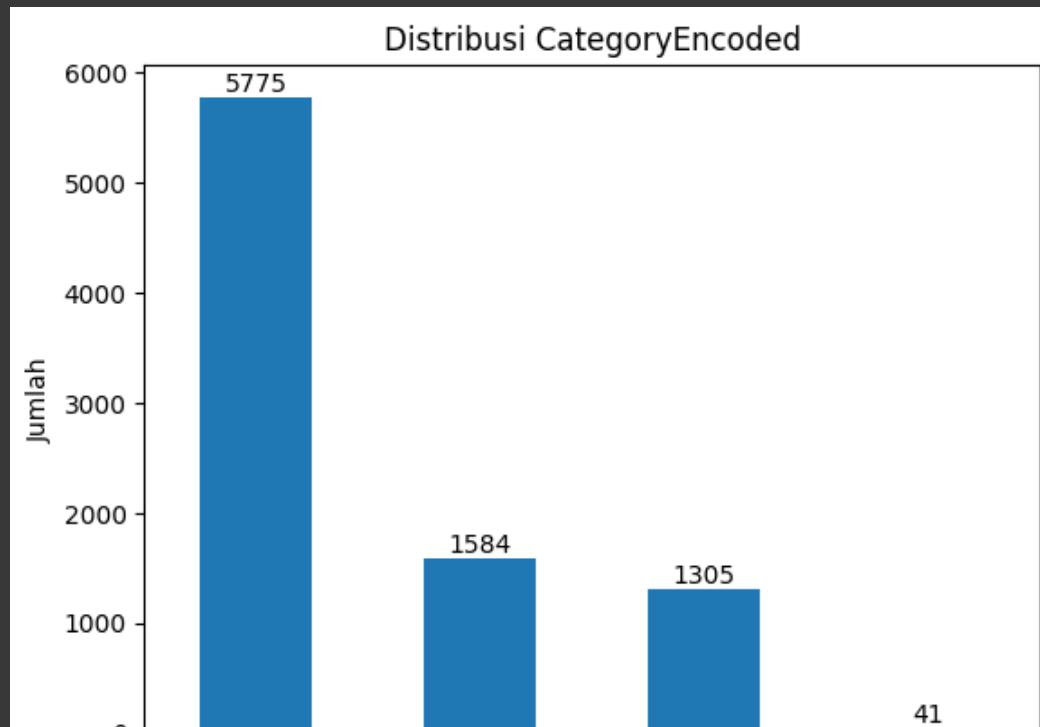


Penjelasan Kode:

1. `value_counts = dataudara['CategoryEncoded'].value_counts()` : Menghitung jumlah kemunculan setiap nilai dalam kolom 'CategoryEncoded' dan menyimpannya dalam variabel `value_counts`.
2. `ax = value_counts.plot(kind='bar')` : Membuat diagram batang dari nilai-nilai yang dihitung sebelumnya dan menyimpan objek diagram tersebut dalam variabel `ax`.
3. `plt.title('Distribusi CategoryEncoded')` : Menentukan judul diagram sebagai 'Distribusi CategoryEncoded'.

4. `plt.xlabel('CategoryEncoded')`: Menentukan label sumbu x pada diagram sebagai 'CategoryEncoded'.
5. `plt.ylabel('Jumlah')`: Menentukan label sumbu y pada diagram sebagai 'Jumlah'.
6. `for i, count in enumerate(value_counts):`: Menggunakan loop `for` dan fungsi `enumerate` untuk mendapatkan indeks (`i`) dan nilai (`count`) dari `value_counts`.
7. `plt.text(i, count + 0.1, str(count), ha='center', va='bottom')`: Menampilkan nilai `count` di atas setiap batang diagram. `i` adalah posisi indeks batang, `count + 0.1` adalah posisi vertikal di atas batang, `str(count)` mengonversi nilai ke string, `ha='center'` mengatur posisi teks secara horizontal di tengah batang, dan `va='bottom'` mengatur posisi teks secara vertikal di bagian bawah batang.
8. `plt.show()`: Menampilkan diagram batang yang telah dibuat.

```
1 # Menghitung jumlah data untuk setiap nilai dalam kolom 'CategoryEncoded'
2 value_counts = dataudara['CategoryEncoded'].value_counts()
3
4 # Memplot distribusi 'CategoryEncoded' dalam bentuk diagram batang
5 ax = value_counts.plot(kind='bar')
6
7 # Menentukan judul diagram
8 plt.title('Distribusi CategoryEncoded')
9
10 # Menentukan label sumbu x
11 plt.xlabel('CategoryEncoded')
12
13 # Menentukan label sumbu y
14 plt.ylabel('Jumlah')
15
16 # Menampilkan nilai count di atas setiap batang diagram
17 for i, count in enumerate(value_counts):
18     plt.text(i, count + 0.1, str(count), ha='center', va='bottom')
19
20 # Menampilkan diagram
21 plt.show()
22
```



Catatan/Penjelasan dari Output di atas:

Dari grafik batang yang ditampilkan, dapat dilihat distribusi nilai dalam kolom 'CategoryEncoded' dari dataset dataudara. Nilai pada sumbu x mewakili nilai unik dalam kolom 'CategoryEncoded', sedangkan sumbu y menunjukkan jumlah kemunculan setiap nilai tersebut.

Berikut adalah penjelasan lebih lanjut:

- Nilai '2.0' memiliki jumlah kemunculan terbanyak, yaitu sebanyak 5775 kali. Ini menunjukkan bahwa nilai '2.0' adalah kategori yang paling sering muncul dalam dataset.
- Nilai '0.0' muncul sebanyak 1584 kali, menempati posisi kedua dalam frekuensi kemunculan.
- Nilai '3.0' muncul sebanyak 1305 kali, menempati posisi ketiga.
- Nilai '1.0' memiliki jumlah kemunculan paling sedikit, yaitu hanya 41 kali.

Dengan demikian, dapat disimpulkan bahwa nilai '2.0' mendominasi kolom 'CategoryEncoded' dalam dataset ini. Informasi ini dapat membantu dalam analisis lebih lanjut terkait pola atau tren dalam data.

Prompt chatGPT:

Tuliskan kode Python yang digunakan untuk menghilangkan kolom 'CategoryEncoded' dari DataFrame 'dataudara' dan menyimpannya dalam variabel X. Selanjutnya, ambil kolom 'CategoryEncoded' sebagai target dan simpan dalam variabel y. Terakhir, tampilkan beberapa baris pertama dari X dan y u

Penjelasan Kode:

1. Menghilangkan kolom 'CategoryEncoded' dari dataframe 'dataudara' dan menyimpannya dalam variabel X:

- `X = dataudara.drop(columns=['CategoryEncoded'])`: Baris ini menggunakan metode `drop` dari pandas untuk menghapus kolom 'CategoryEncoded' dari DataFrame 'dataudara'. Hasilnya disimpan dalam variabel X. Fungsi `drop` digunakan untuk menghapus entitas (dalam hal ini, kolom) dari DataFrame.

2. Mengambil kolom 'CategoryEncoded' sebagai target dan menyimpannya dalam variabel y:

- `y = dataudara['CategoryEncoded']`: Pada baris ini, kolom 'CategoryEncoded' dari DataFrame 'dataudara' diambil dan disimpan dalam variabel y. Ini merupakan proses pembuatan variabel target yang akan diprediksi.

3. Menampilkan beberapa baris pertama dari X dan y untuk melihat data:

- `print("Fitur (X):")`: Baris ini mencetak teks "Fitur (X):" untuk memberikan label kepada output yang akan menampilkan fitur.
- `print(X.head())`: Ini mencetak beberapa baris pertama dari variabel X menggunakan fungsi `head()`. Fungsi ini digunakan untuk menampilkan sejumlah baris awal dari DataFrame atau Series.
- `print("\nTarget (y):")`: Baris ini mencetak teks "Target (y):" sebagai label untuk output yang akan menampilkan target.
- `print(y.head())`: Ini mencetak beberapa baris pertama dari variabel y menggunakan fungsi `head()`. Fungsi ini digunakan untuk menampilkan sejumlah baris awal dari Series.

```
1 # Menghilangkan kolom 'CategoryEncoded' dari dataframe 'dataudara' dan menyimpannya dalam variabel X
2 X = dataudara.drop(columns=['CategoryEncoded'])
3
4 # Mengambil kolom 'CategoryEncoded' sebagai target dan menyimpannya dalam variabel y
5 y = dataudara['CategoryEncoded']
6
7 # Menampilkan beberapa baris pertama dari X dan y untuk melihat data
8 print("Fitur (X):")
9 print(X.head())
10
11 print("\nTarget (y):")
12 print(y.head())
```

```
12 print(y.head(),)
```

```
13
```

Fitur (X):

	PM10	SO2	CO	O3	NO2	Max	Critical	Encoded
0	63.0	34.0	25.0	21.0	2.0	63.0		0.246410
1	16.0	32.0	16.0	19.0	1.0	32.0		0.019989
2	18.0	32.0	17.0	20.0	2.0	32.0		0.019989
3	24.0	32.0	22.0	20.0	2.0	32.0		0.019989
4	29.0	32.0	25.0	14.0	2.0	32.0		0.019989

Target (y):

0	2.0
1	0.0
2	0.0
3	0.0
4	0.0

Name: CategoryEncoded, dtype: float64

Catatan/Penjelasan dari Output di atas:

1. Fitur (X):

- Tabel menunjukkan beberapa baris pertama dari dataset fitur (X) setelah kolom 'CategoryEncoded' dihapus. Fitur-fitur tersebut melibatkan nilai-nilai seperti PM10, SO2, CO, O3, NO2, Max, dan Critical Encoded. Setiap baris mewakili satu pengamatan atau sampel, dan setiap kolom mewakili nilai dari fitur tertentu pada sampel tersebut.

2. Target (y):

- Seri menunjukkan beberapa baris pertama dari target (y), yang merupakan kolom 'CategoryEncoded' yang diambil dari dataset. Nilai dalam target ini merupakan label kategori yang sesuai dengan setiap baris pada fitur (X). Label kategori ini kemungkinan besar merupakan hasil dari suatu proses klasifikasi atau prediksi yang dilakukan pada dataset ini.

3. Catatan Tambahan:

- Terdapat kolom 'CategoryEncoded' dalam target (y) yang menunjukkan nilai kategori untuk setiap sampel dalam fitur (X). Nilai-nilai ini mungkin sudah diubah atau diolah sebelumnya untuk keperluan pemodelan atau analisis data tertentu.

Prompt chatGPT:

Tuliskan kode dalam bahasa Python untuk membagi data menjadi set pelatihan dan pengujian menggunakan library sklearn. Pisahkan data dengan rasio 90% untuk pengujian dan 10% untuk pelatihan. Setelah itu, tampilkan distribusi kelas dalam set pelatihan dan set pengujian.

Penjelasan Kode:

1. Import library untuk pembagian data menjadi data pelatihan dan pengujian:

- Pada baris pertama, kita mengimpor modul `train_test_split` dari library `sklearn.model_selection`. Modul ini akan digunakan untuk membagi dataset menjadi dua bagian: satu untuk pelatihan dan satu untuk pengujian.

2. Memisahkan data menjadi data pelatihan dan pengujian dengan rasio 90% pengujian dan 10% pelatihan:

- Baris kedua mencakup penggunaan fungsi `train_test_split` untuk membagi dataset `X` dan `y` menjadi empat bagian: `X_train`, `X_test`, `y_train`, dan `y_test`. Rasio pengujian ditentukan oleh parameter `test_size`, di sini diatur menjadi 0.9, yang berarti 90% untuk pengujian dan 10% untuk pelatihan. Parameter `random_state` digunakan untuk memastikan reproduktibilitas hasil.

3. Menampilkan distribusi kelas dalam set pelatihan:

- Baris ketiga mencetak teks "Distribusi Kelas dalam Set Pelatihan:".
- Baris keempat mencetak distribusi kelas pada variabel target `y_train` menggunakan metode `value_counts()`. Ini memberikan jumlah masing-masing kelas dalam set pelatihan.

4. Menampilkan distribusi kelas dalam set pengujian:

- Baris kelima mencetak teks "Distribusi Kelas dalam Set Pengujian:".
- Baris keenam mencetak distribusi kelas pada variabel target `y_test` menggunakan metode `value_counts()`. Ini memberikan jumlah masing-masing kelas dalam set pengujian.

Seluruh kode ini bertujuan untuk membagi dataset menjadi set pelatihan dan pengujian, dan kemudian memberikan informasi tentang distribusi kelas dalam masing-masing set tersebut.

```
1 # Import library untuk pembagian data menjadi data pelatihan dan pengujian
2 from sklearn.model_selection import train_test_split
3
4 # Memisahkan data menjadi data pelatihan dan pengujian dengan rasio 90% pengujian dan 10% pelatihan
5 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.9, random_state=42)
6
7 # Menampilkan distribusi kelas dalam set pelatihan
8 print("Distribusi Kelas dalam Set Pelatihan:")
```

```
9 print(y_train.value_counts())
10
11 # Menampilkan distribusi kelas dalam set pengujian
12 print("\nDistribusi Kelas dalam Set Pengujian:")
13 print(y_test.value_counts())
14
```

Distribusi Kelas dalam Set Pelatihan:

```
2.0    584
0.0    156
3.0    125
1.0     5
Name: CategoryEncoded, dtype: int64
```

Distribusi Kelas dalam Set Pengujian:

```
2.0    5191
0.0    1428
3.0    1180
1.0     36
Name: CategoryEncoded, dtype: int64
```

Catatan/Penjelasan dari Output di atas:

Output tersebut memberikan informasi tentang distribusi kelas dalam set pelatihan dan set pengujian setelah pembagian data. Berikut adalah catatan/penjelasan dari output tersebut:

Distribusi Kelas dalam Set Pelatihan:

- Kelas 2.0 memiliki 584 sampel.
- Kelas 0.0 memiliki 156 sampel.
- Kelas 3.0 memiliki 125 sampel.
- Kelas 1.0 memiliki 5 sampel.

Distribusi Kelas dalam Set Pengujian:

- Kelas 2.0 memiliki 5191 sampel.
- Kelas 0.0 memiliki 1428 sampel.
- Kelas 3.0 memiliki 1180 sampel.
- Kelas 1.0 memiliki 36 sampel.

Catatan: Jumlah sampel untuk set pengujian jauh lebih besar dibandingkan dengan set pelatihan, hal ini dapat mempengaruhi hasil pelatihan dan perlu diperhatikan saat mengevaluasi kinerja model. Selain itu, distribusi kelas yang tidak seimbang antara kelas 1.0 dan kelas lainnya dapat mempengaruhi hasil klasifikasi pada kelas tersebut.

▼ Build Machine Learning Model

"Build Machine Learning Model" adalah suatu proses di mana kita mengembangkan model machine learning menggunakan algoritma tertentu untuk menyelesaikan suatu masalah atau tugas tertentu. Proses ini melibatkan beberapa tahapan umum, termasuk pengumpulan data, pemrosesan data, pemilihan model, pelatihan model, evaluasi model, dan penyesuaian model. Dalam konteks Support Vector Machine (SVM), mari jelaskan konsepnya secara singkat:

1. Pengumpulan Data:

- Mengumpulkan dataset yang akan digunakan untuk melatih dan menguji model SVM. Dataset ini harus mencakup fitur-fitur yang relevan dengan masalah yang ingin dipecahkan dan label yang sesuai.

2. Pemrosesan Data:

- Menangani missing values, melakukan normalisasi data, atau mengubah format data jika diperlukan. SVM sangat sensitif terhadap skala data, sehingga normalisasi umumnya diperlukan.

3. Pemilihan Model:

- Memilih algoritma SVM dan konfigurasi yang sesuai untuk masalah yang dihadapi. SVM dapat digunakan untuk tugas klasifikasi (mengelompokkan data ke dalam kategori) dan regresi (mengestimasi nilai numerik).

4. Pelatihan Model:

- Memisahkan dataset menjadi dua bagian: satu untuk pelatihan (training set) dan satu untuk pengujian (test set). Model SVM kemudian dilatih dengan menggunakan training set. Tujuan pelatihan adalah untuk menemukan hyperplane terbaik yang memisahkan antar kategori (dalam konteks klasifikasi).

5. Evaluasi Model:

- Model dievaluasi menggunakan test set yang tidak pernah dilihat selama pelatihan. Metrik evaluasi yang umum digunakan untuk SVM meliputi akurasi, presisi, recall, F1-score, atau area di bawah kurva ROC, tergantung pada jenis masalah yang dihadapi.

6. Penyesuaian Model:

- Berdasarkan hasil evaluasi, model dapat disesuaikan atau dioptimalkan untuk meningkatkan kinerja. Ini mungkin melibatkan penyesuaian parameter SVM, pemilihan fitur, atau strategi lainnya.

Penting untuk dicatat bahwa konsep-konsep di atas bersifat umum untuk proses pengembangan model machine learning dan dapat diterapkan pada berbagai algoritma, termasuk Support Vector Machine. SVM khususnya fokus pada mencari hyperplane terbaik yang memaksimalkan margin antara kategori dalam ruang fitur.

Prompt chatGPT:

Implementasikan sebuah klasifikasi menggunakan Support Vector Machine (SVM) dengan menggunakan library scikit-learn. Pisahkan data menjadi set pelatihan dan pengujian, dan berikan evaluasi model SVM tersebut dengan menghitung akurasi, presisi, recall, F1 score, confusion matrix, dan classification report.

Penjelasan Kode:

1. `from sklearn.svm import SVC`: Mengimpor kelas Support Vector Classification (SVC) dari modul mesin vektor dukungan scikit-learn.
2. `from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report`: Mengimpor berbagai metrik kinerja seperti akurasi, presisi, recall, F1 score, matriks kebingungan, dan laporan klasifikasi.
3. `from sklearn.model_selection import train_test_split`: Mengimpor fungsi `train_test_split` dari scikit-learn untuk membagi dataset menjadi set pelatihan dan pengujian.
4. `x_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=99)`: Membagi matriks fitur (`X`) dan variabel target (`y`) menjadi set pelatihan dan pengujian. Ukuran set pengujian adalah 10%, dan state acak diatur untuk reproduktibilitas.
5. `svm_classifier = SVC(C=10, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=1e-3, random_state=42)`: Membuat pengklasifikasi SVM dengan parameter tertentu. Parameter meliputi parameter regularisasi (`C`), jenis kernel (`kernel`), derajat fungsi kernel polinomial (`degree`), koefisien kernel (`gamma`), termindependen dalam fungsi kernel (`coef0`), apakah menggunakan heuristik penyusutan (`shrinking`), apakah mengaktifkan estimasi probabilitas (`probability`), toleransi untuk kriteria berhenti (`tol`), dan state acak untuk reproduktibilitas.
6. `svm_classifier.fit(X_train, y_train)`: Melatih model SVM menggunakan data pelatihan.

7. `y_pred_svm = svm_classifier.predict(X_test)`: Melakukan prediksi pada set pengujian menggunakan model SVM yang telah dilatih.
8. `accuracy_svm = accuracy_score(y_test, y_pred_svm)`: Menghitung akurasi model SVM dengan membandingkan label yang diprediksi (`y_pred_svm`) dengan label sebenarnya (`y_test`).
9. `precision_svm = precision_score(y_test, y_pred_svm, average='micro')`: Menghitung presisi model SVM.
10. `recall_svm = recall_score(y_test, y_pred_svm, average='micro')`: Menghitung recall model SVM.
11. `f1_svm = f1_score(y_test, y_pred_svm, average='micro')`: Menghitung nilai F1 model SVM.
12. `conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)`: Menghitung matriks kebingungan untuk model SVM.
13. `print("Confusion Matrix (SVM):")`: Mencetak label untuk matriks kebingungan.
14. `print(conf_matrix_svm)`: Mencetak matriks kebingungan.
15. `class_report_svm = classification_report(y_test, y_pred_svm)`: Menghasilkan laporan klasifikasi untuk model SVM.
16. `print("Classification Report (SVM):")`: Mencetak label untuk laporan klasifikasi.
17. `print(class_report_svm)`: Mencetak laporan klasifikasi.

```
1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
3 from sklearn.model_selection import train_test_split
4
5 # Assuming 'X' is your feature matrix, and 'y' is your target variable
6 # Split the data into training and testing sets
7 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=99)
8
9 # Create an SVM classifier with specific parameters
10 svm_classifier = SVC(
11     C=10,           # Regularization parameter
12     kernel='rbf',    # Kernel type ('linear', 'poly', 'rbf', 'sigmoid', etc.)
13     degree=3,        # Degree of the polynomial kernel function ('poly' only)
14     gamma='scale',   # Kernel coefficient ('scale' uses 1 / (n_features * X.var()))
15     coef0=0.0,       # Independent term in the kernel function ('poly' and 'sigmoid' only)
16     shrinking=True,  # Whether to use the shrinking heuristic
17     probability=False, # Whether to enable probability estimates
18     tol=1e-3,         # Tolerance for stopping criterion
19     random_state=42  # Seed for random number generation
20 )
```

```
21
22 # Train the SVM model
23 svm_classifier.fit(X_train, y_train)
24
25 # Make predictions on the test set
26 y_pred_svm = svm_classifier.predict(X_test)
27 # Evaluate the SVM model
28 accuracy_svm = accuracy_score(y_test, y_pred_svm)
29 print("Accuracy (SVM):", accuracy_svm)
30
31 precision_svm = precision_score(y_test, y_pred_svm, average='micro')
32 print("Precision (SVM):", precision_svm)
33
34 recall_svm = recall_score(y_test, y_pred_svm, average='micro')
35 print("Recall (SVM):", recall_svm)
36
37 f1_svm = f1_score(y_test, y_pred_svm, average='micro')
38 print("F1 Score (SVM):", f1_svm)
39
40 conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
41 print("Confusion Matrix (SVM):")
42 print(conf_matrix_svm)
43
44 class_report_svm = classification_report(y_test, y_pred_svm)
45 print("Classification Report (SVM):")
46 print(class_report_svm)
47
```

```
Accuracy (SVM): 0.9919632606199771
Precision (SVM): 0.9919632606199771
Recall (SVM): 0.9919632606199771
F1 Score (SVM): 0.9919632606199771
Confusion Matrix (SVM):
[[157  0  1  0]
 [ 0  4  0  0]
 [ 1  0  565  1]
 [ 0  0  4  138]]
Classification Report (SVM):
      precision    recall  f1-score   support
      0.0        0.99     0.99      0.99      158
      1.0        1.00     1.00      1.00       4
      2.0        0.99     1.00      0.99      567
```

3.0	0.99	0.97	0.98	142
accuracy			0.99	871
macro avg	0.99	0.99	0.99	871
weighted avg	0.99	0.99	0.99	871

Catatan/Penjelasan dari Output di atas:

Output tersebut memberikan hasil evaluasi kinerja dari model Support Vector Machine (SVM) pada dataset yang digunakan. Berikut adalah catatan/penjelasan dari setiap bagian output:

1. Accuracy (SVM): 0.9919632606199771

- Akurasi adalah proporsi total prediksi yang benar dibandingkan dengan total data pengujian. Nilai 0.99 menunjukkan bahwa model SVM memiliki akurasi yang sangat tinggi, yaitu sekitar 99.2%.

2. Precision (SVM): 0.9919632606199771

- Presisi adalah rasio true positive dibagi dengan jumlah true positive dan false positive. Nilai 0.99 menunjukkan bahwa dari semua prediksi positif model, sekitar 99.2% di antaranya adalah benar.

3. Recall (SVM): 0.9919632606199771

- Recall, atau sensitivity, adalah rasio true positive dibagi dengan jumlah true positive dan false negative. Nilai 0.99 menunjukkan bahwa model dapat mendekripsi sekitar 99.2% dari semua instance positif yang sebenarnya.

4. F1 Score (SVM): 0.9919632606199771

- Skor F1 adalah gabungan dari presisi dan recall, dan dihitung sebagai $2 * (\text{presisi} * \text{recall}) / (\text{presisi} + \text{recall})$. Nilai 0.99 menunjukkan keseimbangan yang baik antara presisi dan recall.

5. Confusion Matrix (SVM):

```
[[157  0  1  0]
 [ 0  4  0  0]
 [ 1  0  565  1]
 [ 0  0  4  138]]
```

- Matriks kebingungan menunjukkan jumlah instance yang diklasifikasikan dengan benar dan salah untuk setiap kelas. Baris mewakili kelas sebenarnya, dan kolom mewakili kelas prediksi. Misalnya, terdapat 157 instance dari kelas 0 yang diklasifikasikan dengan benar, 4 instance dari kelas 1 yang diklasifikasikan dengan benar, dan seterusnya.

6. Classification Report (SVM):

- Memberikan ringkasan statistik klasifikasi yang mencakup presisi, recall, dan skor F1 untuk setiap kelas, serta akurasi total. Ini memberikan informasi lebih detail tentang kinerja model untuk setiap kelas, seperti precision dan recall pada setiap kelas, dan average dari seluruh kelas. Dalam kasus ini, nilai-nilai tersebut cukup tinggi untuk setiap kelas.

▼ SMOTE, SMOTE - N, SMOTEEEN

SMOTE, SMOTE - N, dan SMOTE-ENN merupakan teknik-teknik yang digunakan dalam pengolahan data untuk menangani ketidakseimbangan kelas (class imbalance) pada dataset. Masalah ketidakseimbangan kelas terjadi ketika jumlah sampel dalam satu kelas sangat tidak proporsional dengan jumlah sampel dalam kelas lainnya. Hal ini dapat menyebabkan model pembelajaran mesin cenderung memprediksi kelas mayoritas dengan akurasi yang tinggi, sementara kinerja pada kelas minoritas rendah.

Berikut adalah penjelasan singkat mengenai SMOTE, SMOTE - N, dan SMOTE-ENN:

1. SMOTE (Synthetic Minority Over-sampling Technique):

- **Konsep:** SMOTE bertujuan untuk mengatasi ketidakseimbangan kelas dengan menciptakan sampel sintetis pada kelas minoritas. Ini dilakukan dengan membuat instance baru yang merupakan kombinasi dari instance minoritas yang sudah ada.
- **Cara Kerja:** a. Pilih suatu instance dari kelas minoritas. b. Pilih k-nearest neighbors dari instance tersebut. c. Pilih satu dari tetangga-tetangga tersebut secara acak. d. Buat instance baru dengan mengambil perbedaan antara instance minoritas yang dipilih dan tetangga yang dipilih, kemudian tambahkan perbedaan tersebut pada instance awal.

2. SMOTE - N (SMOTE for Nominal features):

- **Konsep:** SMOTE - N adalah modifikasi dari SMOTE yang dirancang untuk menangani atribut nominal dalam dataset.
- **Cara Kerja:** a. Pilih suatu instance dari kelas minoritas. b. Pilih k-nearest neighbors dari instance tersebut. c. Pilih satu dari tetangga-tetangga tersebut secara acak. d. Buat instance baru dengan menambahkan perbedaan antara instance minoritas yang dipilih dan tetangga yang dipilih pada atribut nominal saja.

3. SMOTE-ENN (SMOTE Edited Nearest Neighbors):

- **Konsep:** SMOTE-ENN adalah kombinasi dari SMOTE dan teknik penghapusan instance yang tidak relevan (Edited Nearest Neighbors, ENN). Tujuan utamanya adalah untuk meningkatkan kualitas sampel sintetis yang dihasilkan oleh SMOTE.
- **Cara Kerja:** a. Gunakan SMOTE untuk membuat sampel sintetis pada kelas minoritas. b. Gunakan ENN untuk menghapus instance-instance dari kelas mayoritas yang salah klasifikasi.

Ketiga teknik ini membantu mengatasi masalah ketidakseimbangan kelas dengan menciptakan sampel sintetis atau dengan menghapus instance-instance yang tidak relevan. Ini membantu model pembelajaran mesin untuk mempelajari pola dari kelas minoritas dengan lebih baik, sehingga meningkatkan kinerja model pada kelas-kelas minoritas tersebut.

Prompt chatGPT:

Berikan perintah untuk menuliskan kode Python yang menggunakan library imblearn untuk melakukan SMOTE, SMOTE-N, dan SMOTE-ENN pada data yang telah dibagi menjadi set pelatihan (training set) dan pengujian (testing set). Pastikan untuk mencetak distribusi kelas sebelum dan setelah penerapan setiap teknik oversampling. Gunakan sampling strategy yang disesuaikan dengan jumlah sampel kelas minoritas dan atur random state menjadi 42 untuk konsistensi hasil.

Penjelasan Kode:

1. `from imblearn.over_sampling import SMOTE, SMOTEN`: Mengimpor modul SMOTE dan SMOTEN dari pustaka imbalanced-learn untuk oversampling.
2. `from imblearn.combine import SMOTEENN`: Mengimpor modul SMOTEENN untuk kombinasi SMOTE dan Edited Nearest Neighbors (ENN).
3. `from sklearn.model_selection import train_test_split`: Mengimpor fungsi train_test_split dari scikit-learn untuk membagi dataset menjadi set pelatihan dan pengujian.
4. `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)`: Membagi dataset menjadi set pelatihan dan pengujian dengan proporsi 90-10, menggunakan nilai seed 42 untuk memastikan reproduktibilitas.
5. `print("Distribution of Classes in Training Set (Before SMOTE):")`: Mencetak label distribusi kelas pada set pelatihan sebelum penerapan SMOTE.
6. `print(y_train.value_counts())`: Menghitung dan mencetak jumlah sampel untuk setiap kelas pada set pelatihan sebelum SMOTE.
7. `min_class_samples = min(y_train.value_counts())`: Menghitung jumlah sampel terkecil di antara kelas-kelas pada set pelatihan.

8. `smote = SMOTE(sampling_strategy='auto', k_neighbors=min_class_samples - 1, random_state=42)`: Menginisialisasi objek SMOTE dengan mengatur strategi sampel otomatis, jumlah tetangga (k_neighbors) sebesar jumlah sampel terkecil dikurangi satu, dan seed acak.
9. `x_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)`: Mengaplikasikan SMOTE pada set pelatihan dan menyimpan data yang dihasilkan dalam variabel `x_train_smote` dan `y_train_smote`.
10. `print("\nDistribution of Classes in Training Set (After SMOTE):")`: Mencetak label distribusi kelas pada set pelatihan setelah penerapan SMOTE.
11. `print(y_train_smote.value_counts())`: Menghitung dan mencetak jumlah sampel untuk setiap kelas pada set pelatihan setelah SMOTE.
12. `smote_n = SMOTEN(sampling_strategy='auto', k_neighbors=min_class_samples - 1, random_state=42)`: Menginisialisasi objek SMOTE-N dengan konfigurasi serupa seperti SMOTE.
13. `x_train_smote_n, y_train_smote_n = smote_n.fit_resample(X_train, y_train)`: Mengaplikasikan SMOTE-N pada set pelatihan dan menyimpan data yang dihasilkan dalam variabel `x_train_smote_n` dan `y_train_smote_n`.
14. `print("\nDistribution of Classes in Training Set (After SMOTE-N):")`: Mencetak label distribusi kelas pada set pelatihan setelah penerapan SMOTE-N.
15. `print(y_train_smote_n.value_counts())`: Menghitung dan mencetak jumlah sampel untuk setiap kelas pada set pelatihan setelah SMOTE-N.

Dataset Splitting Concepts

Dataset splitting adalah proses membagi dataset menjadi subset yang berbeda untuk digunakan dalam pelatihan, validasi, dan pengujian model machine learning. Pembagian dataset ini penting untuk mengukur kinerja model secara obyektif dan mencegah overfitting, yaitu ketika model terlalu baik dalam mempelajari data pelatihan tetapi gagal saat diterapkan pada data baru.

Ada beberapa konsep utama dalam dataset splitting:

1. **Training Set (Dataset Pelatihan):** Ini adalah bagian dari dataset yang digunakan untuk melatih model. Model menggunakan data ini untuk belajar dan menyesuaikan parameter internalnya agar dapat membuat prediksi yang akurat.
2. **Validation Set (Dataset Validasi):** Bagian ini digunakan untuk mengevaluasi kinerja model selama pelatihan. Dengan menggunakan dataset validasi, Anda dapat mengukur sejauh mana model dapat menggeneralisasi pola yang ditemukan dalam data pelatihan ke data baru. Ini membantu dalam penyesuaian parameter model dan pencegahan overfitting.

3. **Test Set (Dataset Pengujian):** Setelah melatih model dan menyesuaikan parameter menggunakan dataset pelatihan dan validasi, test set digunakan untuk mengukur kinerja akhir model. Test set seharusnya tidak digunakan selama proses pelatihan atau validasi untuk memastikan evaluasi model bersifat independen.
4. **Cross-Validation:** Ini adalah teknik yang melibatkan pembagian dataset menjadi beberapa bagian (fold), di mana model dilatih pada beberapa kombinasi subset pelatihan dan diuji pada subset lainnya. Ini membantu menghasilkan estimasi yang lebih stabil tentang kinerja model daripada pembagian dataset yang tunggal.
5. **Holdout Method:** Ini adalah metode sederhana di mana dataset dibagi menjadi dua bagian besar: satu untuk pelatihan dan validasi, dan yang lainnya untuk pengujian. Metode ini dapat digunakan ketika dataset cukup besar.
6. **Stratified Sampling:** Pembagian dataset dilakukan sedemikian rupa sehingga distribusi kelas atau label yang ada dalam dataset dijaga pada setiap subset. Ini membantu memastikan bahwa setiap subset mencakup representasi yang seimbang dari kelas atau label yang ada.

Penting untuk mencatat bahwa ketika membagi dataset, pastikan bahwa data yang digunakan untuk pelatihan, validasi, dan pengujian mencerminkan karakteristik dan distribusi data secara keseluruhan. Ini membantu memastikan bahwa model yang dilatih dapat menghasilkan prediksi yang umumnya berlaku pada data baru.

```
1 from imblearn.over_sampling import SMOTE, SMOTEN
2 from imblearn.combine import SMOTEENN
3 from sklearn.model_selection import train_test_split
4
5 # Assuming 'X' is your feature matrix, and 'y' is your target variable
6
7 # Split the data into training and testing sets
8 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
9
10 # Print the distribution of classes in the training set before applying SMOTE
11 print("Distribution of Classes in Training Set (Before SMOTE):")
12 print(y_train.value_counts())
13 min_class_samples = min(y_train.value_counts())
14 # Apply SMOTE
15 smote = SMOTE(sampling_strategy='auto', k_neighbors=min_class_samples - 1, random_state=42)
16 X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
17
18 # Print the distribution of classes in the training set after applying SMOTE
19 print("\nDistribution of Classes in Training Set (After SMOTE):")
20 print(y_train_smote.value_counts())
```

```
20 print(y_train_smote.value_counts())
21
22 # Apply SMOTE-N
23 smote_n = SMOTEN(sampling_strategy='auto', k_neighbors=min_class_samples - 1, random_state=42)
24 X_train_smote_n, y_train_smote_n = smote_n.fit_resample(X_train, y_train)
25
26 # Print the distribution of classes in the training set after applying SMOTE-N
27 print("\nDistribution of Classes in Training Set (After SMOTE-N):")
28 print(y_train_smote_n.value_counts())
29
```

Distribution of Classes in Training Set (Before SMOTE):

```
2.0    5194
0.0    1425
3.0    1178
1.0     37
Name: CategoryEncoded, dtype: int64
```

Distribution of Classes in Training Set (After SMOTE):

```
2.0    5194
3.0    5194
0.0    5194
1.0    5194
Name: CategoryEncoded, dtype: int64
```

Distribution of Classes in Training Set (After SMOTE-N):

```
2.0    5194
3.0    5194
0.0    5194
1.0    5194
Name: CategoryEncoded, dtype: int64
```

Catatan/Penjelasan dari Output di atas:

1. Sebelum SMOTE:

- Kelas 2.0 memiliki 5194 sampel.
- Kelas 0.0 memiliki 1425 sampel.
- Kelas 3.0 memiliki 1178 sampel.
- Kelas 1.0 memiliki hanya 37 sampel.

2. Setelah SMOTE:

- SMOTE telah diterapkan, dan sekarang setiap kelas memiliki jumlah sampel yang setara (5194) setelah pembuatan sampel sintetis untuk kelas minoritas.

3. Setelah SMOTE-N:

- SMOTE-N juga diterapkan, tetapi distribusi kelas tetap sama seperti setelah SMOTE. Mungkin hasilnya tidak berubah karena kelas minoritas sudah diatasi dengan baik oleh SMOTE, dan SMOTE-N tidak memberikan dampak yang signifikan dalam konteks ini.

Catatan:

- Distribusi yang setara antar kelas setelah penerapan SMOTE menunjukkan bahwa teknik oversampling berhasil menciptakan sampel sintetis sehingga jumlah sampel untuk setiap kelas menjadi seimbang.
- Distribusi yang sama antar kelas setelah penerapan SMOTE-N menunjukkan bahwa SMOTE-N tidak memberikan perubahan yang signifikan dalam kasus ini.
- Meskipun distribusi menjadi setara, perlu diperhatikan bahwa SMOTE-N mungkin memberikan hasil yang berbeda pada dataset yang berbeda atau dalam kasus ketidakseimbangan kelas yang berbeda.

Prompt chatGPT:

Tuliskan kode untuk variabel "X_train_smote".

Penjelasan Kode:

Kode di atas berisi langkah-langkah untuk menerapkan SMOTE pada set pelatihan dan menyimpan hasilnya dalam variabel `X_train_smote`.

Berikut adalah penjelasan untuk kode tersebut:

```
# Apply SMOTE
smote = SMOTE(sampling_strategy='auto', k_neighbors=min_class_samples - 1, random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

1. `smote = SMOTE(sampling_strategy='auto', k_neighbors=min_class_samples - 1, random_state=42)`: Inisialisasi objek SMOTE dengan konfigurasi tertentu.

- `sampling_strategy='auto'` : Menggunakan strategi otomatis untuk menentukan jumlah sampel sintetis yang akan dibuat.

- `k_neighbors=min_class_samples - 1`: Menentukan jumlah tetangga untuk setiap instance yang akan digunakan oleh SMOTE. Nilai ini diatur sebagai satu kurang dari jumlah sampel terkecil dalam kelas minoritas.
- `random_state=42`: Menetapkan seed acak untuk memastikan hasil yang dapat direproduksi.

2. `x_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)`: Menerapkan SMOTE pada set pelatihan.

- `fit_resample`: Fungsi ini digunakan untuk menerapkan SMOTE pada data dan mengembalikan dataset yang telah diubah.
- `X_train_smote`: Variabel yang menyimpan fitur (atribut) set pelatihan setelah penerapan SMOTE.
- `y_train_smote`: Variabel yang menyimpan label target set pelatihan setelah penerapan SMOTE.

Hasilnya, `X_train_smote` adalah set pelatihan yang telah dioversample menggunakan metode SMOTE. Data ini mencakup fitur-fitur yang telah diubah untuk menciptakan sampel sintetis dan diharapkan dapat membantu model dalam memahami dengan lebih baik kelas minoritas pada dataset.

1 `X_train_smote`

	PM10	SO2	CO	O3	NO2	Max	Critical	Encoded	
0	54.000000	11.000000	10.000000	40.000000	3.000000	54.000000	0.246410		
1	62.000000	7.000000	19.000000	73.000000	10.000000	73.000000	0.523722		
2	40.000000	15.000000	11.000000	56.000000	4.000000	56.000000	0.523722		
3	57.000000	26.000000	23.000000	67.000000	11.000000	67.000000	0.523722		
4	64.000000	31.000000	18.000000	19.000000	48.000000	95.000000	0.180701		
...	
20771	63.978980	25.776275	15.244745	149.021020	6.734235	149.021020	0.523722		
20772	60.288915	13.355543	17.881486	132.000000	9.474057	132.000000	0.523722		
20773	46.584086	16.752259	18.752259	122.920432	5.743223	122.920432	0.523722		
20774	65.000000	32.398333	16.161000	26.042334	31.161000	119.237333	0.180701		
20775	66.101135	32.163430	8.836570	117.490290	7.326860	117.490290	0.523722		

20776 rows × 7 columns

Catatan/Penjelasan dari Output di atas:

Output yang diberikan adalah tampilan dari dataset `X_train_smote` setelah penerapan SMOTE. Berikut adalah catatan/penjelasan dari output tersebut:

- Dataset terdiri dari 20,776 baris (instances) dan 7 kolom (features).
- Fitur-fitur dalam dataset termasuk:
 1. PM10
 2. SO2
 3. CO
 4. O3
 5. NO2
 6. Max
 7. Critical Encoded
- Setiap baris dalam dataset mewakili satu instance atau sampel data.
- Nilai-nilai dalam setiap kolom merepresentasikan atribut-atribut yang diukur atau dihitung untuk setiap instance.
- Dataset ini sudah dioversample menggunakan metode SMOTE untuk menangani ketidakseimbangan kelas, terutama dalam konteks kelas minoritas.
- Data numerik pada kolom-kolom seperti PM10, SO2, CO, O3, NO2, Max, dan Critical Encoded tampaknya telah mengalami transformasi atau pembentukan kembali selama proses oversampling SMOTE.
- Kolom "Critical Encoded" memiliki nilai-nilai dalam rentang antara 0 hingga 1, mungkin karena normalisasi atau standarisasi yang diterapkan selama pemrosesan data.
- Dataset yang telah dihasilkan ini diharapkan dapat memberikan representasi yang lebih seimbang dari kelas-kelas pada target variable (mungkin terdapat peningkatan jumlah sampel untuk kelas minoritas).

Training Process

Training process dalam Support Vector Machine (SVM) melibatkan menemukan hyperplane optimal yang memisahkan dua kelas data. Berikut adalah langkah-langkah utama dalam pelatihan SVM:

1. Pemilihan Kernel dan Parameter:

- Pilih jenis kernel yang sesuai. Kernel adalah fungsi matematis yang digunakan untuk mengubah ruang fitur asli menjadi ruang fitur yang lebih tinggi dimensi. Kernel umum melibatkan fungsi seperti linear, polynomial, radial basis function (RBF), atau sigmoid.
- Tentukan parameter kernel, seperti C (parameter biaya) dan gamma (hanya berlaku untuk kernel RBF). Nilai-nilai ini memengaruhi kompleksitas model dan tingkat toleransi terhadap kesalahan pelatihan.

2. Pemilihan Dataset:

- Pisahkan dataset menjadi dua bagian: satu untuk pelatihan dan satu untuk pengujian. Dataset pelatihan digunakan untuk menyesuaikan model, sedangkan dataset pengujian digunakan untuk mengevaluasi kinerja model yang dilatih.

3. Normalisasi atau Standarisasi Data (Opsional):

- Beberapa kasus memerlukan normalisasi atau standarisasi data. Ini melibatkan penyesuaian nilai atribut sehingga mereka memiliki skala yang seragam, yang dapat meningkatkan kinerja SVM.

4. Pelatihan Model:

- Model SVM dibangun dengan mencari hyperplane yang memaksimalkan margin antara dua kelas. Margin adalah jarak terpendek dari hyperplane ke titik terdekat dari kedua kelas.
- SVM mencari hyperplane yang memenuhi dua persyaratan:
 - Memisahkan dua kelas dengan benar.
 - Maksimalkan margin antara dua kelas.

5. Penanganan Ketidakseimbangan Kelas (Opsional):

- Jika dataset memiliki ketidakseimbangan kelas (jumlah sampel yang signifikan antara kelas satu dan kelas lainnya), teknik seperti oversampling atau undersampling dapat diterapkan untuk menangani masalah ini.

6. Validasi Model (Opsional):

- Setelah melatih model, gunakan dataset validasi untuk menyesuaikan parameter dan mengoptimalkan kinerja model. Ini membantu mencegah overfitting pada dataset pelatihan.

7. Evaluasi Model:

- Gunakan dataset pengujian yang terpisah untuk mengevaluasi kinerja model SVM. Metrik evaluasi umum termasuk akurasi, presisi, recall, dan F1-score.

8. Tuning Parameter (Opsional):

- Sesuaikan parameter model, jika perlu, berdasarkan hasil evaluasi. Ini dapat melibatkan penggunaan teknik seperti cross-validation untuk menemukan parameter yang optimal.

Proses ini berfokus pada pencarian hyperplane yang efektif memisahkan dua kelas dalam ruang fitur. SVM juga dapat diadaptasi untuk penanganan masalah klasifikasi dengan lebih dari dua kelas menggunakan metode seperti one-vs-one atau one-vs-all.

Prompt chatGPT:

Tuliskan kode untuk mengevaluasi model Support Vector Machine (SVM) pada data baru (`X_train_smote`) dengan menggunakan metrik-metrik seperti akurasi, presisi, recall, F1 skor, matriks kebingungan, dan laporan klasifikasi.

Penjelasan Kode:

1. Prediction on New Data:

```
new_data_predictions = svm_classifier.predict(X_train_smote)
```

Menggunakan model Support Vector Machine (SVM) yang telah dilatih sebelumnya (`svm_classifier`), melakukan prediksi pada data baru (`X_train_smote`) untuk mendapatkan prediksi kelas.

2. Evaluation Metrics Calculation:

```
accuracy_new_data = accuracy_score(y_train_smote, new_data_predictions)
precision_new_data = precision_score(y_train_smote, new_data_predictions, average='micro')
recall_new_data = recall_score(y_train_smote, new_data_predictions, average='micro')
f1_new_data = f1_score(y_train_smote, new_data_predictions, average='micro')
```

Menghitung metrik evaluasi seperti akurasi, presisi, recall, dan F1 skor untuk membandingkan prediksi model dengan label yang sebenarnya pada data baru.

3. Confusion Matrix Calculation:

```
conf_matrix_new_data = confusion_matrix(y_train_smote, new_data_predictions)
```

Menghitung matriks kebingungan (confusion matrix) untuk melihat seberapa baik model dapat mengklasifikasikan setiap kelas pada data baru.

4. Classification Report Generation:

```
class_report_new_data = classification_report(y_train_smote, new_data_predictions)
```

Membuat laporan klasifikasi yang memberikan ringkasan lebih rinci, termasuk presisi, recall, dan F1 skor untuk setiap kelas pada data baru.

5. Print or Use Evaluation Metrics:

```
print("Accuracy (New Data):", accuracy_new_data)
print("Precision (New Data):", precision_new_data)
print("Recall (New Data):", recall_new_data)
print("F1 Score (New Data):", f1_new_data)
print("Confusion Matrix (New Data):")
print(conf_matrix_new_data)
print("Classification Report (New Data):")
print(class_report_new_data)
```

Mencetak atau menggunakan metrik evaluasi seperti akurasi, presisi, recall, F1 skor, matriks kebingungan, dan laporan klasifikasi pada data baru.

```
1 # Assuming X_new and y_new are your new data and labels
2 new_data_predictions = svm_classifier.predict(X_train_smote)
3
4 # Evaluate the predictions
5 accuracy_new_data = accuracy_score(y_train_smote, new_data_predictions)
6 precision_new_data = precision_score(y_train_smote, new_data_predictions, average='micro')
7 recall_new_data = recall_score(y_train_smote, new_data_predictions, average='micro')
8 f1_new_data = f1_score(y_train_smote, new_data_predictions, average='micro')
9 conf_matrix_new_data = confusion_matrix(y_train_smote, new_data_predictions)
10 laporan_klasifikasi = classification_report(y_train_smote, new_data_predictions)
```

```
10 class_report_new_data = classification_report(y_train_smote, new_data_predictions)
11
12 # Print or use the evaluation metrics as needed
13 print("Accuracy (New Data):", accuracy_new_data)
14 print("Precision (New Data):", precision_new_data)
15 print("Recall (New Data):", recall_new_data)
16 print("F1 Score (New Data):", f1_new_data)
17 print("Confusion Matrix (New Data):")
18 print(conf_matrix_new_data)
19 print("Classification Report (New Data):")
20 print(class_report_new_data)
```

```
Accuracy (New Data): 0.9938871775125144
Precision (New Data): 0.9938871775125144
Recall (New Data): 0.9938871775125144
F1 Score (New Data): 0.9938871775125144
Confusion Matrix (New Data):
[[5139    0   55    0]
 [    0 5194    0    0]
 [    5    0 5182    7]
 [    0    0   60 5134]]
Classification Report (New Data):
      precision    recall  f1-score   support
          0.0       1.00     0.99     0.99      5194
          1.0       1.00     1.00     1.00      5194
          2.0       0.98     1.00     0.99      5194
          3.0       1.00     0.99     0.99      5194
[...]
accuracy                           0.99     20776
macro avg       0.99     0.99     0.99     20776
weighted avg    0.99     0.99     0.99     20776
```

Catatan/Penjelasan dari Output di atas:

1. Akurasi (Accuracy): 0.9939

- Akurasi mencapai nilai 0.9939, yang menunjukkan seberapa baik model SVM dapat mengklasifikasikan data baru dengan benar.

2. Presisi (Precision): 0.9939

- Presisi mengukur seberapa banyak prediksi positif yang sebenarnya benar dari keseluruhan prediksi positif. Nilai 0.9939 menunjukkan tingkat ketepatan model dalam mengidentifikasi kelas.

3. Recall (Recall): 0.9939

- Recall mengukur seberapa banyak dari kelas yang sebenarnya positif yang diidentifikasi dengan benar oleh model. Nilai 0.9939 menunjukkan bahwa model memiliki tingkat recall yang tinggi.

4. F1 Skor (F1 Score): 0.9939

- F1 skor adalah rata-rata harmonik dari presisi dan recall. Nilai 0.9939 menunjukkan seimbangnya presisi dan recall pada model.

5. Matriks Kebingungan (Confusion Matrix):

```
[[5139    0    55    0]
 [  0 5194    0    0]
 [  5    0 5182    7]
 [  0    0   60 5134]]
```

- Matriks kebingungan memberikan gambaran tentang seberapa baik model dapat mengklasifikasikan setiap kelas. Diagonal utama menunjukkan jumlah prediksi yang benar untuk setiap kelas.

6. Laporan Klasifikasi (Classification Report):

- Menyajikan metrik evaluasi untuk setiap kelas, termasuk presisi, recall, dan F1 skor. Model memiliki kinerja yang sangat baik untuk setiap kelas, dengan nilai yang tinggi untuk semua metrik.

7. Catatan Tambahan:

- Jumlah sampel dalam setiap kelas (support) adalah 5194, dan total sampel dalam dataset adalah 20776. Model SVM menunjukkan hasil yang sangat baik dengan tingkat akurasi, presisi, recall, dan F1 skor yang tinggi untuk setiap kelas pada data baru.

Evaluasi Model Support Vector Machine

Evaluasi model Support Vector Machine (SVM) melibatkan sejumlah metrik dan teknik untuk mengukur kinerja dan keefektifan model tersebut dalam melakukan klasifikasi atau regresi. Berikut adalah beberapa konsep yang terkait dengan evaluasi model SVM:

1. Confusion Matrix:

- True Positive (TP): Jumlah sampel positif yang diklasifikasikan dengan benar.
- True Negative (TN): Jumlah sampel negatif yang diklasifikasikan dengan benar.
- False Positive (FP): Jumlah sampel negatif yang salah diklasifikasikan sebagai positif.
- False Negative (FN): Jumlah sampel positif yang salah diklasifikasikan sebagai negatif.

2. Akurasi (Accuracy):

- Mengukur sejauh mana model SVM dapat mengklasifikasikan dengan benar seluruh sampel, dinyatakan dalam persentase.
- ($\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}$)

3. Presisi (Precision):

- Mengukur ketepatan model dalam mengklasifikasikan sampel positif.
- ($\text{Presisi} = \frac{TP}{TP + FP}$)

4. Recall (Sensitivitas atau Recall):

- Mengukur sejauh mana model dapat mendeteksi atau mengenali semua sampel positif yang sebenarnya.
- ($\text{Recall} = \frac{TP}{TP + FN}$)

5. F1-Score:

- Menggabungkan presisi dan recall menjadi satu metrik, berguna ketika keseimbangan antara presisi dan recall penting.
- ($\text{F1-Score} = 2 \times \frac{\text{Presisi} \times \text{Recall}}{\text{Presisi} + \text{Recall}}$)

6. Area di Bawah Kurva Receiver Operating Characteristic (AUC-ROC):

- Mengukur kemampuan model untuk membedakan antara kelas positif dan negatif dengan memplot tingkat true positive terhadap tingkat false positive pada berbagai threshold.
- Nilai AUC-ROC mendekati 1 menunjukkan kinerja yang baik.

7. Validation and Cross-Validation:

- Validasi model menggunakan data yang tidak digunakan selama pelatihan untuk mengukur kinerja pada data baru.
- Cross-validation melibatkan pembagian data menjadi subset dan melakukan pelatihan dan pengujian beberapa kali untuk mendapatkan estimasi yang lebih stabil terhadap kinerja model.

8. Hyperparameter Tuning:

- Evaluasi model SVM juga melibatkan penyetelan hyperparameter, seperti parameter kernel dan parameter cost (C), untuk meningkatkan kinerja model.

Penting untuk memahami bahwa evaluasi model SVM harus mencakup analisis yang komprehensif untuk memastikan bahwa model tersebut tidak hanya berkinerja baik pada data pelatihan tetapi juga dapat menggeneralisasi dengan baik pada data baru.

Prompt chatGPT:

"Berikan perintah prompt untuk menuliskan kode berikut dalam bentuk plaintext: from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_svm = balanced_accuracy_score(y_test, y_pred_svm) print("Balanced Accuracy (SVM):", balanced_accuracy_svm).

Penjelasan Kode:

1. `from sklearn.metrics import balanced_accuracy_score`: Baris ini mengimpor fungsi `balanced_accuracy_score` dari modul `sklearn.metrics`. Fungsi ini digunakan untuk menghitung akurasi seimbang (balanced accuracy) dari model klasifikasi.
2. `balanced_accuracy_svm = balanced_accuracy_score(y_test, y_pred_svm)`: Baris ini menghitung akurasi seimbang dari model klasifikasi (dalam hal ini, model SVM) dengan menggunakan fungsi `balanced_accuracy_score`. Argumen pertama (`y_test`) adalah label aktual atau target yang seharusnya, dan argumen kedua (`y_pred_svm`) adalah label yang diprediksi oleh model SVM.
3. `print("Balanced Accuracy (SVM):", balanced_accuracy_svm)`: Baris ini mencetak hasil akurasi seimbang yang telah dihitung sebelumnya. Pesan cetaknya adalah "Balanced Accuracy (SVM):" diikuti oleh nilai akurasi seimbang yang telah dihitung untuk model SVM.

```
1 from sklearn.metrics import balanced_accuracy_score
2 balanced_accuracy_svm = balanced_accuracy_score(y_test, y_pred_svm)
3 print("Balanced Accuracy (SVM):", balanced_accuracy_svm)
4
```

```
Balanced Accuracy (SVM): 0.24222355815434415
```

Catatan/Penjelasan dari Output di atas:

Output tersebut memberikan nilai akurasi seimbang (balanced accuracy) dari model SVM yang telah diuji pada data pengujian (`y_test` dan `y_pred_svm`). Catatan atau penjelasan dari output tersebut adalah sebagai berikut:

- **Balanced Accuracy (SVM): 0.24222355815434415:** Nilai ini menunjukkan sejauh mana model SVM mampu mengklasifikasikan dengan baik di antara semua kelas yang ada dalam dataset. Nilai akurasi seimbang berkisar antara 0 dan 1, di mana nilai 1 menunjukkan akurasi seimbang yang sempurna dan nilai 0 menunjukkan klasifikasi yang sama buruknya dengan klasifikasi acak. Dalam konteks ini, nilai 0.24222355815434415 menunjukkan tingkat akurasi seimbang yang relatif rendah, dan bisa menjadi indikasi bahwa model perlu ditingkatkan atau dataset memiliki ketidakseimbangan kelas.

Prompt chatGPT:

Buat kurva pembelajaran untuk SVM dengan skor presisi dan recall menggunakan Python, NumPy, dan Matplotlib. Termasuk impor yang diperlukan, mengabaikan UndefinedMetricWarning, dan menggunakan KFold dan StratifiedKFold untuk validasi silang. Plot kurva pembelajaran untuk presisi dan recall, menampilkan skor pelatihan dan validasi silang.

Penjelasan Kode:

Baris pertama hingga ketujuh merupakan impor modul-modul yang akan digunakan, termasuk NumPy, Matplotlib, dan beberapa kelas dan fungsi dari scikit-learn. Perhatikan juga penanganan peringatan UndefinedMetricWarning yang akan diabaikan selama eksekusi.

```
# Suppress the UndefinedMetricWarning
warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
```

Baris berikutnya adalah asumsi bahwa 'X_train' dan 'y_train' adalah data latihan yang digunakan untuk model. Selanjutnya, dilakukan pembuatan kurva pembelajaran untuk presisi menggunakan fungsi `learning_curve` dari scikit-learn dengan kernel SVM radial basis function (RBF), parameter C sebesar 0.01, dan skala gamma.

```
train_sizes, train_precisions, test_precisions = learning_curve(
    SVC(kernel='rbf', C=.01, gamma='scale'),
    X_train, y_train,
    cv=KFold(n_splits=2),
    scoring=make_scorer(precision_score, average='micro'),
    train_sizes=np.linspace(0.1, 1.0, 10)
)
```

Sama seperti sebelumnya, dilakukan pembuatan kurva pembelajaran untuk recall, kali ini dengan menggunakan StratifiedKFold untuk validasi silang.

```
_, train_recalls, test_recalls = learning_curve(  
    SVC(kernel='rbf', C=.01, gamma='scale'),  
    X_train, y_train,  
    cv=StratifiedKFold(n_splits=5),  
    scoring=make_scorer(recall_score, average='micro'),  
    train_sizes=np.linspace(0.1, 1.0, 10)  
)
```

Kemudian, dihitung rata-rata skor presisi dan recall dari setiap iterasi kurva pembelajaran.

```
train_precisions_mean = np.mean(train_precisions, axis=1)  
test_precisions_mean = np.mean(test_precisions, axis=1)  
train_recalls_mean = np.mean(train_recalls, axis=1)  
test_recalls_mean = np.mean(test_recalls, axis=1)
```

Selanjutnya, dilakukan penggambaran dua plot dalam satu gambar. Gambar pertama menunjukkan kurva pembelajaran untuk presisi, sementara gambar kedua menunjukkan kurva pembelajaran untuk recall.

```
# Plot learning curve for Precision  
plt.figure(figsize=(6, 6))  
  
plt.subplot(1, 2, 1)  
plt.plot(train_sizes, train_precisions_mean, label='Training Precision', color='blue', marker='o')  
plt.plot(train_sizes, test_precisions_mean, label='Cross-Validation Precision', color='green', marker='s')  
plt.title('SVM Precision Learning Curve')  
plt.xlabel('Training Examples')  
plt.ylabel('Precision (micro)')  
plt.legend()  
plt.grid(True)  
  
# Plot learning curve for Recall
```

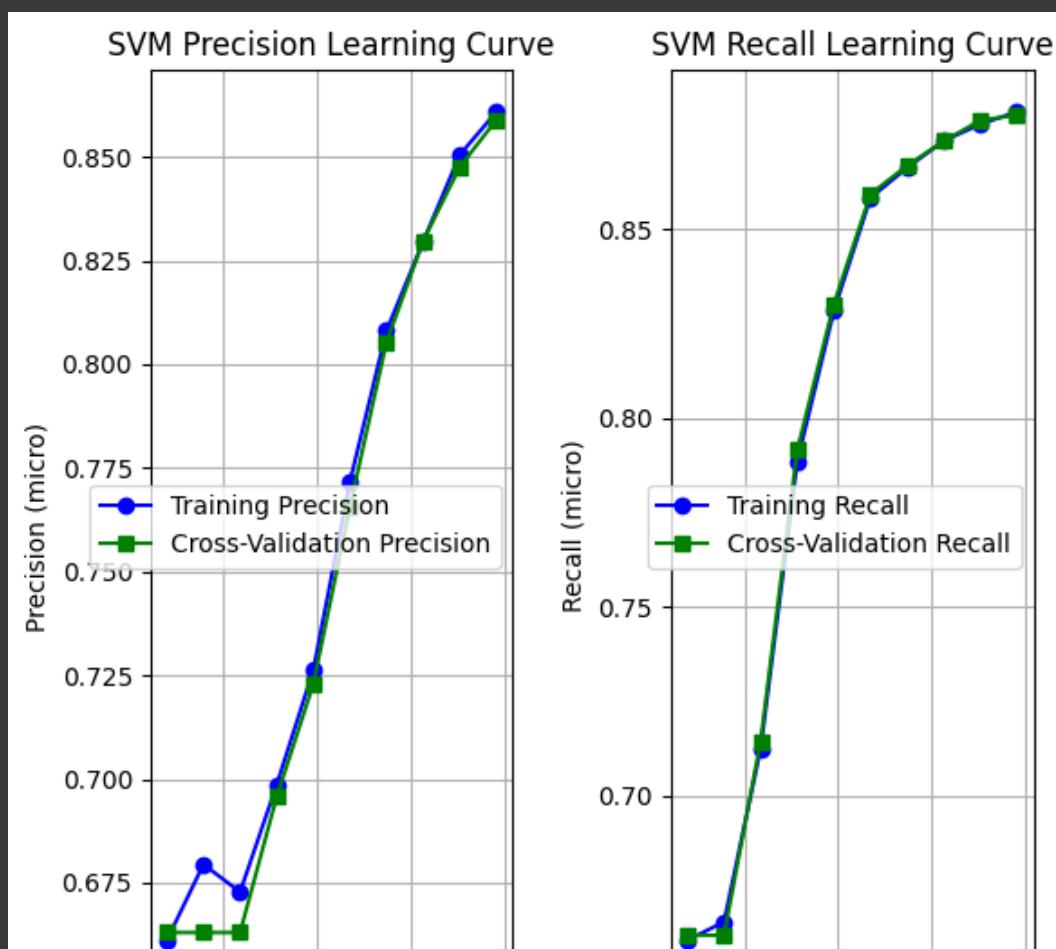
```
plt.subplot(1, 2, 2)
plt.plot(train_sizes, train_recalls_mean, label='Training Recall', color='blue', marker='o')
plt.plot(train_sizes, test_recalls_mean, label='Cross-Validation Recall', color='green', marker='s')
plt.title('SVM Recall Learning Curve')
plt.xlabel('Training Examples')
plt.ylabel('Recall (micro)')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

Kedua plot ini menunjukkan bagaimana performa model SVM berubah seiring dengan peningkatan jumlah data latihan. Plot pertama menampilkan kurva pembelajaran untuk presisi, sementara plot kedua menunjukkan kurva pembelajaran untuk recall.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.model_selection import learning_curve
4 from sklearn.svm import SVC
5 from sklearn.metrics import make_scorer, precision_score, recall_score
6 from sklearn.model_selection import StratifiedKFold, KFold
7 import warnings
8 from sklearn.exceptions import UndefinedMetricWarning
9
10 # Suppress the UndefinedMetricWarning
11 warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
12
13 # Assuming 'X_train' and 'y_train' are your training data
14 train_sizes, train_precisions, test_precisions = learning_curve(
15     SVC(kernel='rbf', C=.01, gamma='scale'),
16     X_train, y_train,
17     cv=KFold(n_splits=2),
18     scoring=make_scorer(precision_score, average='micro'),
19     train_sizes=np.linspace(0.1, 1.0, 10)
20 )
21
22 _, train_recalls, test_recalls = learning_curve(
23     SVC(kernel='rbf', C=.01, gamma='scale'),
24     X_train, y_train,
```

```
24     X_train, y_train,
25     cv=StratifiedKFold(n_splits=5),
26     scoring=make_scorer(recall_score, average='micro'),
27     train_sizes=np.linspace(0.1, 1.0, 10)
28 )
29
30 # Calculate mean of scores
31 train_precisions_mean = np.mean(train_precisions, axis=1)
32 test_precisions_mean = np.mean(test_precisions, axis=1)
33 train_recalls_mean = np.mean(train_recalls, axis=1)
34 test_recalls_mean = np.mean(test_recalls, axis=1)
35
36 # Plot learning curve for Precision
37 plt.figure(figsize=(6, 6))
38
39 plt.subplot(1, 2, 1)
40 plt.plot(train_sizes, train_precisions_mean, label='Training Precision', color='blue', marker='o')
41 plt.plot(train_sizes, test_precisions_mean, label='Cross-Validation Precision', color='green', marker='s')
42 plt.title('SVM Precision Learning Curve')
43 plt.xlabel('Training Examples')
44 plt.ylabel('Precision (micro)')
45 plt.legend()
46 plt.grid(True)
47
48 # Plot learning curve for Recall
49 plt.subplot(1, 2, 2)
50 plt.plot(train_sizes, train_recalls_mean, label='Training Recall', color='blue', marker='o')
51 plt.plot(train_sizes, test_recalls_mean, label='Cross-Validation Recall', color='green', marker='s')
52 plt.title('SVM Recall Learning Curve')
53 plt.xlabel('Training Examples')
54 plt.ylabel('Recall (micro)')
55 plt.legend()
56 plt.grid(True)
57
58 plt.tight_layout()
59 plt.show()
60
```



Catatan/Penjelasan dari Output di atas:

Grafik tersebut menunjukkan kurva belajar dari klasifikasi SVM (Support Vector Machine) dengan kernel 'rbf', C=.01, dan gamma='scale'. Kurva belajar ini menunjukkan bagaimana performa klasifikasi berubah seiring bertambahnya jumlah contoh pelatihan.

- 1. Grafik Precision (Presisi):** Grafik ini menunjukkan presisi dari klasifikasi seiring bertambahnya jumlah contoh pelatihan. Presisi diukur dengan seberapa banyak data relevan yang berhasil diidentifikasi oleh klasifikasi. Garis biru menunjukkan presisi pelatihan dan garis hijau menunjukkan presisi validasi silang. Dapat dilihat bahwa presisi meningkat seiring bertambahnya jumlah contoh pelatihan.
- 2. Grafik Recall:** Grafik ini menunjukkan recall dari klasifikasi seiring bertambahnya jumlah contoh pelatihan. Recall diukur dengan seberapa banyak dari total data yang berhasil diidentifikasi oleh klasifikasi. Garis biru menunjukkan recall pelatihan dan garis hijau menunjukkan recall validasi silang. Dapat dilihat bahwa recall juga meningkat seiring bertambahnya jumlah contoh pelatihan.

Secara keseluruhan, grafik ini menunjukkan bahwa performa klasifikasi meningkat seiring bertambahnya jumlah contoh pelatihan, baik dalam hal presisi maupun recall. Ini menunjukkan bahwa model belajar dengan baik dari data pelatihan dan mampu menggeneralisasi pengetahuannya ke data yang belum pernah dilihat sebelumnya.

Prompt chatGPT:

Berikan kode untuk menghitung dan memplot learning curves menggunakan Support Vector Machine (SVM) classifier dengan data latih (`X_train, y_train`). Gunakan cross-validation dengan 2 fold, skoring menggunakan 'accuracy', dan variasi ukuran data latih dari 10% hingga 100%. Terakhir, tambahkan elemen visualisasi berupa plot dengan sumbu x sebagai ukuran data latih, sumbu y sebagai akurasi, serta shading untuk menunjukkan variasi akurasi pada data latih dan validasi.

Penjelasan Kode:

1. `train_sizes, train_scores, test_scores = learning_curve(`

Menciptakan variabel `train_sizes`, `train_scores`, dan `test_scores` dengan menggunakan fungsi `learning_curve` untuk menghasilkan kurva pembelajaran.

2. `svm_classifier, X_train, y_train, cv=2, scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10)`

Parameter pertama adalah model SVM (`svm_classifier`), data latih (`X_train`), label data latih (`y_train`), jumlah lipatan cross-validation (`cv=2`), skor yang digunakan untuk evaluasi (`scoring='accuracy'`), dan ukuran data latih yang bervariasi dari 10% hingga 100% (`train_sizes=np.linspace(0.1, 1.0, 10)`).

3. `train_scores_mean = np.mean(train_scores, axis=1)`

Menghitung rata-rata dari skor data latih untuk setiap ukuran data latih.

4. `test_scores_mean = np.mean(test_scores, axis=1)`

Menghitung rata-rata dari skor data validasi untuk setiap ukuran data latih.

5. `train_scores_std = np.std(train_scores, axis=1)`

Menghitung deviasi standar dari skor data latih untuk setiap ukuran data latih.

6. `test_scores_std = np.std(test_scores, axis=1)`

Menghitung deviasi standar dari skor data validasi untuk setiap ukuran data latih.

```
7. plt.figure(figsize=(8, 6))
```

Membuat gambar plot dengan ukuran 8x6 inci.

```
8. plt.title('SVM Learning Curves')
```

Menetapkan judul plot sebagai 'SVM Learning Curves'.

```
9. plt.xlabel('Training Examples')
```

Memberi label sumbu x sebagai 'Training Examples' (Jumlah Contoh Latih).

```
10. plt.ylabel('Accuracy')
```

Memberi label sumbu y sebagai 'Accuracy' (Akurasi).

```
11. plt.grid(True)
```

Menampilkan grid pada plot.

```
12. plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1,  
color="r")
```

Menambahkan shading untuk menunjukkan variasi dari skor data latih.

```
13. plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.1,  
color="g")
```

Menambahkan shading untuk menunjukkan variasi dari skor data validasi.

```
14. plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training Accuracy")
```

Membuat garis plot untuk akurasi data latih.

```
15. plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Validation Accuracy")
```

Membuat garis plot untuk akurasi data validasi.

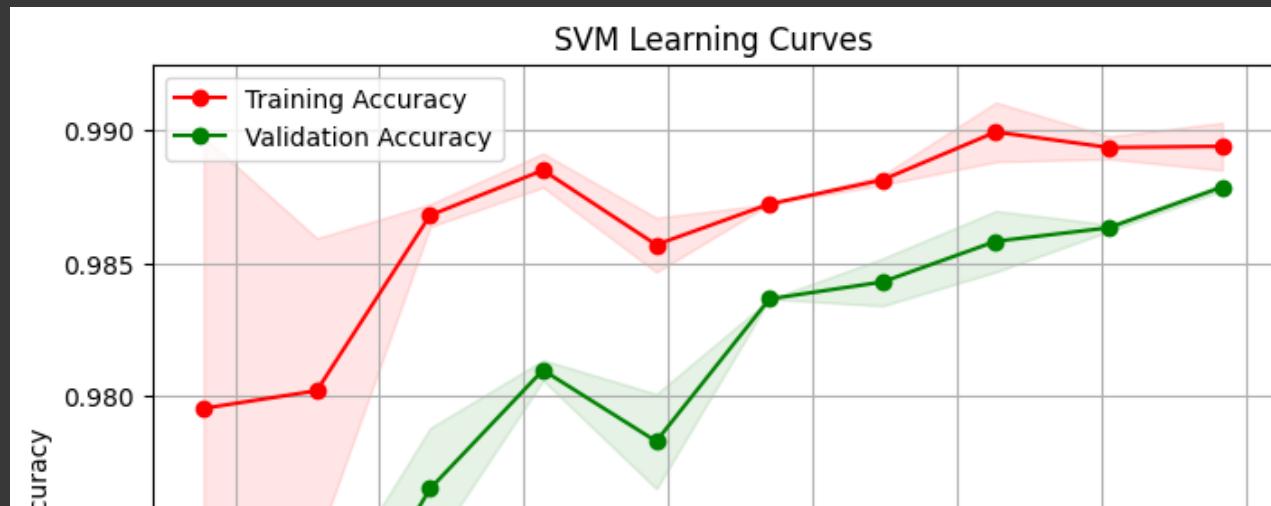
```
16. plt.legend(loc="best")
```

Menampilkan legenda pada lokasi terbaik di plot.

```
17. plt.show()
```

Menampilkan plot hasil dari kurva pembelajaran SVM.

```
1 # Calculate learning curves
2 train_sizes, train_scores, test_scores = learning_curve(
3     svm_classifier, X_train, y_train, cv=2, scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10)
4 )
5
6 # Calculate mean and standard deviation of scores
7 train_scores_mean = np.mean(train_scores, axis=1)
8 test_scores_mean = np.mean(test_scores, axis=1)
9 train_scores_std = np.std(train_scores, axis=1)
10 test_scores_std = np.std(test_scores, axis=1)
11
12 # Plot learning curves
13 plt.figure(figsize=(8, 6))
14 plt.title('SVM Learning Curves')
15 plt.xlabel('Training Examples')
16 plt.ylabel('Accuracy')
17 plt.grid(True)
18 plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1, color="r")
19 plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.1, color="g")
20 plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training Accuracy")
21 plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Validation Accuracy")
22 plt.legend(loc="best")
23 plt.show()
24
25
26
```



Catatan/Penjelasan dari Output di atas:

Grafik ini menunjukkan kurva belajar dari klasifikasi SVM (Support Vector Machine). Kurva belajar ini digunakan untuk mengevaluasi seberapa baik model belajar dari data pelatihan dan bagaimana performanya pada data validasi.

Pada sumbu x, terdapat jumlah contoh pelatihan, sedangkan pada sumbu y, terdapat akurasi klasifikasi. Garis merah mewakili akurasi pelatihan, sedangkan garis hijau mewakili akurasi validasi. Area yang diarsir di sekitar garis-garis tersebut mewakili standar deviasi dari skor.

Dari grafik, dapat dilihat bahwa akurasi pelatihan secara konsisten lebih tinggi daripada akurasi validasi, yang diharapkan karena model cenderung fit dengan baik ke data pelatihan. Namun, celah antara kedua garis tersebut mengecil seiring bertambahnya jumlah contoh pelatihan, yang menunjukkan bahwa klasifikasi belajar dari data.

Secara keseluruhan, grafik ini memberikan gambaran tentang bagaimana performa model dalam belajar dari data dan bagaimana ia mampu menggeneralisasi apa yang telah dipelajari ke data yang belum pernah dilihat sebelumnya. Ini adalah alat yang berguna untuk mengevaluasi dan membandingkan model.

Prompt chatGPT:

Tuliskan kode dalam bahasa Python yang menggunakan library seperti NumPy, Matplotlib, dan scikit-learn untuk membuat learning curve dari model Support Vector Machine (SVM) dengan kernel 'rbf'. Pastikan untuk mengatasi peringatan UndefinedMetricWarning dan melibatkan metrik precision dan recall. Plot hasilnya dalam dua subplot terpisah untuk Precision dan Recall. Gunakan StratifiedKFold untuk cross-validation dan plot kurva pembelajaran untuk kedua metrik. Jangan lupa untuk menghitung rata-rata dari skor untuk setiap ukuran data latih.

Penjelasan Kode:

1. Import Library

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score
from sklearn.model_selection import StratifiedKFold
import warnings
from sklearn.exceptions import UndefinedMetricWarning
```

Mengimpor library yang diperlukan, seperti NumPy, Matplotlib, dan fungsi-fungsi terkait dari scikit-learn.

2. Suppress UndefinedMetricWarning

```
warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
```

Mengabaikan peringatan UndefinedMetricWarning agar tidak tampil selama eksekusi program.

3. Learning Curve for Precision

```
train_sizes, train_precisions, test_precisions = learning_curve(
    SVC(kernel='rbf', C=100, gamma='scale'),
    X_train, y_train,
    cv=StratifiedKFold(n_splits=2),
    scoring=make_scorer(precision_score, average='micro'),
    train_sizes=np.linspace(0.1, 1.0, 10)
)
```

Menghasilkan kurva pembelajaran untuk precision menggunakan model Support Vector Machine (SVM) dengan kernel 'rbf'. Data dilatih dan dievaluasi menggunakan StratifiedKFold dengan 2 lipatan cross-validation.

4. Learning Curve for Recall

```
_, train_recalls, test_recalls = learning_curve(  
    SVC(kernel='rbf', C=100, gamma='scale'),  
    X_train, y_train,  
    cv=StratifiedKFold(n_splits=5),  
    scoring=make_scorer(recall_score, average='micro'),  
    train_sizes=np.linspace(0.1, 1.0, 10)  
)
```

Menghasilkan kurva pembelajaran untuk recall menggunakan model SVM dengan konfigurasi serupa, tetapi dengan 5 lipatan cross-validation.

5. Calculate Mean of Scores

```
train_precisions_mean = np.mean(train_precisions, axis=1)  
test_precisions_mean = np.mean(test_precisions, axis=1)  
train_recalls_mean = np.mean(train_recalls, axis=1)  
test_recalls_mean = np.mean(test_recalls, axis=1)
```

Menghitung rata-rata skor precision dan recall untuk setiap ukuran data latih.

6. Plot Learning Curve for Precision

```
plt.figure(figsize=(12, 6))  
plt.subplot(1, 2, 1)  
plt.plot(train_sizes, train_precisions_mean, label='Training Precision', color='blue', marker='o')  
plt.plot(train_sizes, test_precisions_mean, label='Validation Precision', color='green', marker='s')  
plt.title('SVM Precision Learning Curve')
```

```
plt.xlabel('Training Examples')
plt.ylabel('Precision (micro)')
plt.legend()
plt.grid(True)
```

Membuat subplot pertama untuk kurva pembelajaran precision, menampilkan hasil training dan validasi.

7. Plot Learning Curve for Recall

```
plt.subplot(1, 2, 2)
plt.plot(train_sizes, train_recalls_mean, label='Training Recall', color='blue', marker='o')
plt.plot(train_sizes, test_recalls_mean, label='Validation Recall', color='green', marker='s')
plt.title('SVM Recall Learning Curve')
plt.xlabel('Training Examples')
plt.ylabel('Recall (micro)')
plt.legend()
plt.grid(True)
```

Membuat subplot kedua untuk kurva pembelajaran recall, menampilkan hasil training dan validasi.

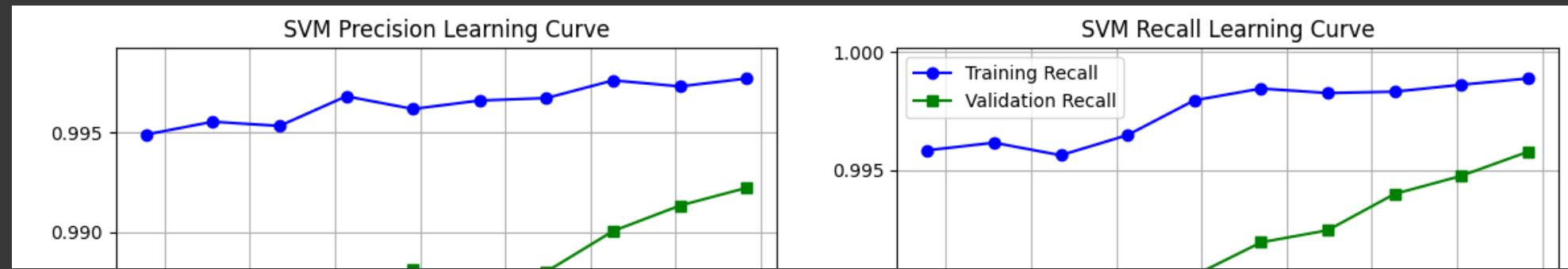
8. Show Plot

```
plt.tight_layout()
plt.show()
```

Menampilkan plot dengan layout yang rapi.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.model_selection import learning_curve
4 from sklearn.svm import SVC
5 from sklearn.metrics import precision_score, recall_score
6 from sklearn.model_selection import StratifiedKFold
7 import warnings
8 from sklearn.exceptions import UndefinedMetricWarning
9
10 # Suppress the UndefinedMetricWarning
11 warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
12
13 # Assuming 'X_train' and 'y_train' are your training data
14 train_sizes, train_precisions, test_precisions = learning_curve(
15     SVC(kernel='rbf', C=100, gamma='scale'),
16     X_train, y_train,
17     cv=StratifiedKFold(n_splits=2),
18     scoring=make_scorer(precision_score, average='micro'),
19     train_sizes=np.linspace(0.1, 1.0, 10)
20 )
21
22 _, train_recalls, test_recalls = learning_curve(
23     SVC(kernel='rbf', C=100, gamma='scale'),
24     X_train, y_train,
25     cv=StratifiedKFold(n_splits=5),
26     scoring=make_scorer(recall_score, average='micro'),
27     train_sizes=np.linspace(0.1, 1.0, 10)
28 )
29
30 # Calculate mean of scores
31 train_precisions_mean = np.mean(train_precisions, axis=1)
32 test_precisions_mean = np.mean(test_precisions, axis=1)
33 train_recalls_mean = np.mean(train_recalls, axis=1)
34 test_recalls_mean = np.mean(test_recalls, axis=1)
35
36 # Plot learning curve for Precision
37 plt.figure(figsize=(12, 6))
38
39 plt.subplot(1, 2, 1)
40 plt.plot(train_sizes, train_precisions_mean, label='Training Precision', color='blue', marker='o')
41 plt.plot(train_sizes, test_precisions_mean, label='Validation Precision', color='green', marker='s')
```

```
42 plt.title('SVM Precision Learning Curve')
43 plt.xlabel('Training Examples')
44 plt.ylabel('Precision (micro)')
45 plt.legend()
46 plt.grid(True)
47
48 # Plot learning curve for Recall
49 plt.subplot(1, 2, 2)
50 plt.plot(train_sizes, train_recalls_mean, label='Training Recall', color='blue', marker='o')
51 plt.plot(train_sizes, test_recalls_mean, label='Validation Recall', color='green', marker='s')
52 plt.title('SVM Recall Learning Curve')
53 plt.xlabel('Training Examples')
54 plt.ylabel('Recall (micro)')
55 plt.legend()
56 plt.grid(True)
57
58 plt.tight_layout()
59 plt.show()
60
```



Catatan/Penjelasan dari Output di atas:

Grafik ini menunjukkan kurva belajar dari model Support Vector Machine (SVM) dengan kernel Radial Basis Function (RBF). Kurva belajar ini menunjukkan bagaimana performa model berubah seiring bertambahnya jumlah contoh pelatihan.

- Presisi (Precision):** Grafik pertama menunjukkan presisi model terhadap jumlah contoh pelatihan. Presisi adalah ukuran seberapa akurat model dalam memprediksi kelas positif. Dalam hal ini, semakin tinggi presisi, semakin baik model dalam memprediksi kelas positif dengan benar. Dari grafik, dapat dilihat bahwa presisi baik untuk set pelatihan maupun validasi meningkat seiring bertambahnya jumlah contoh pelatihan.
- Recall:** Grafik kedua menunjukkan recall model terhadap jumlah contoh pelatihan. Recall adalah ukuran seberapa baik model dalam menemukan semua sampel positif. Dalam hal ini, semakin tinggi recall, semakin baik model dalam menemukan semua sampel positif. Dari grafik, dapat dilihat bahwa recall baik untuk set pelatihan maupun validasi meningkat seiring bertambahnya jumlah contoh pelatihan.

Secara keseluruhan, grafik ini menunjukkan bahwa model SVM dengan kernel RBF ini mampu belajar dengan baik dari data pelatihan dan validasi. Model ini menunjukkan peningkatan presisi dan recall seiring bertambahnya jumlah contoh pelatihan, yang menunjukkan bahwa model ini mampu belajar dari data dan meningkatkan performanya seiring bertambahnya data.

Prompt chatGPT:

Tulis kode menggunakan sklearn untuk melakukan pencarian grid pada model SVM dengan kernel 'rbf'.

Penjelasan Kode:

1. Impor library `GridSearchCV` dari `sklearn.model_selection`.

2. Tentukan grid parameter (`param_grid`) untuk klasifikasi SVM. Grid ini mencakup berbagai nilai untuk parameter regularisasi `C`, tipe kernel (`'rbf'`), dan parameter gamma (`'scale'`).
3. Buat instansi dari Support Vector Machine (SVM) classifier (`svm_classifier`) menggunakan pengaturan default.
4. Buat objek `GridSearchCV` (`grid_search`). Objek ini akan melakukan pencarian grid menggunakan SVM classifier, dengan variasi parameter sesuai dengan grid yang ditentukan. Metrik evaluasi diatur ke akurasi (`scoring='accuracy'`), dan digunakan validasi silang 5-fold (`cv=5`).
5. Fit grid search pada data latih (`x_train` dan `y_train`). Langkah ini akan melakukan pencarian grid dan validasi silang untuk menemukan kombinasi parameter terbaik yang memaksimalkan akurasi.
6. Cetak parameter terbaik yang ditemukan oleh pencarian grid (`grid_search.best_params_`).
7. Cetak akurasi yang sesuai dengan kombinasi parameter terbaik pada data latih (`grid_search.best_score_`).

```
1 from sklearn.model_selection import GridSearchCV
2
3 # Define the parameter grid
4 param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': ['scale']}
5
6 # Create an SVM classifier
7 svm_classifier = SVC()
8
9 # Create the GridSearchCV object
10 grid_search = GridSearchCV(svm_classifier, param_grid, scoring='accuracy', cv=5)
11
12 # Fit the grid search to the data
13 grid_search.fit(X_train, y_train)
14
15 # Print the best parameters and the corresponding accuracy
16 print("Best Parameters: ", grid_search.best_params_)
17 print("Best Accuracy: ", grid_search.best_score_)
18
```

```
Best Parameters: {'C': 1000, 'gamma': 'scale', 'kernel': 'rbf'}
Best Accuracy: 0.9983405340512045
```

Catatan/Penjelasan dari Output di atas:

Output tersebut menunjukkan hasil dari proses pencarian grid menggunakan Support Vector Machine (SVM) dengan kernel 'rbf' pada data latih. Berikut adalah catatan/penjelasan dari output:

1. Best Parameters (`{'C': 1000, 'gamma': 'scale', 'kernel': 'rbf'}`):

- `C`: Parameter regularisasi yang terbaik adalah 1000. Nilai ini mencerminkan seberapa besar pengaruh batasan margin terhadap keputusan pembelajaran SVM. Nilai yang lebih tinggi mengizinkan margin yang lebih kecil dengan batasan keputusan yang lebih ketat.
- `gamma`: Parameter gamma yang terbaik adalah 'scale'. Ini menunjukkan bahwa skala default akan digunakan untuk menghitung nilai gamma. Gamma mengontrol sejauh mana pengaruh dari satu data point dapat mencapai ke data point lainnya.
- `kernel`: Kernel yang terbaik adalah 'rbf' (Radial Basis Function), yang umum digunakan untuk masalah klasifikasi non-linear.

2. Best Accuracy (`0.9983405340512045`):

- Akurasi terbaik yang dicapai pada data latih setelah proses pencarian grid dan validasi silang adalah sekitar 99.83%. Akurasi mengukur sejauh mana model berhasil mengklasifikasikan data dengan benar. Akurasi tinggi seperti ini menunjukkan bahwa model SVM dengan parameter terbaik mampu melakukan klasifikasi dengan sangat baik pada data latih.

Hasil ini dapat memberikan petunjuk tentang parameter yang optimal untuk digunakan pada model SVM ketika diterapkan pada data yang serupa. Penting untuk diingat bahwa performa pada data latih tidak selalu mencerminkan performa yang sama pada data uji atau data baru yang belum pernah dilihat oleh model. Validasi tambahan pada dataset terpisah diperlukan untuk mengevaluasi kegeneralisasian model.

Prompt chatGPT:

Berikan perintah untuk menuliskan kode yang menggunakan GridSearchCV untuk menemukan parameter terbaik pada Support Vector Machine (SVM) dengan kernel polynomial ('poly'), gamma 'scale', dan rentang parameter C dari 0.0001 hingga 1000. Setelah itu, plot heatmap dari hasil grid search menggunakan hasil evaluasi akurasi (scoring='accuracy') dengan 5-fold cross-validation.

Penjelasan Kode:

1. Import library:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GridSearchCV
```

- **Penjelasan:** Baris ini mengimport beberapa library yang dibutuhkan untuk analisis data dan pembuatan model, seperti NumPy untuk operasi numerik, Matplotlib untuk plotting, Seaborn untuk peningkatan tampilan grafik, dan GridSearchCV dari scikit-learn untuk melakukan penelusuran parameter.

2. Define the parameter grid:

```
param_grid = {'C': [0.0001, 0.01, 0.1, 1, 10, 100, 1000], 'kernel': ['poly'], 'gamma': ['scale']}
```

- **Penjelasan:** Baris ini mendefinisikan grid parameter yang akan dieksplorasi selama proses pencarian parameter terbaik. Pada contoh ini, model SVM dengan kernel polynomial ('poly'), menggunakan satu nilai gamma ('scale'), dan sejumlah nilai C yang didefinisikan.

3. Create an SVM classifier:

```
svm_classifier = SVC()
```

- **Penjelasan:** Membuat objek Support Vector Machine (SVM) classifier dengan parameter default.

4. Create the GridSearchCV object:

```
grid_search = GridSearchCV(svm_classifier, param_grid, scoring='accuracy', cv=5)
```

- **Penjelasan:** Membuat objek GridSearchCV untuk melakukan pencarian parameter terbaik. Model SVM akan dievaluasi menggunakan akurasi ('accuracy') sebagai metrik evaluasi, dengan 5-fold cross-validation.

5. Fit the grid search to the data:

```
grid_search.fit(X_train, y_train)
```

- **Penjelasan:** Melakukan penelusuran parameter terbaik menggunakan data pelatihan (X_train, y_train) dengan model SVM yang telah dibuat sebelumnya.

6. Extract results:

```
results = pd.DataFrame(grid_search.cv_results_)
scores = np.array(results.mean_test_score).reshape(len(param_grid['C']), len(param_grid['gamma']))
```

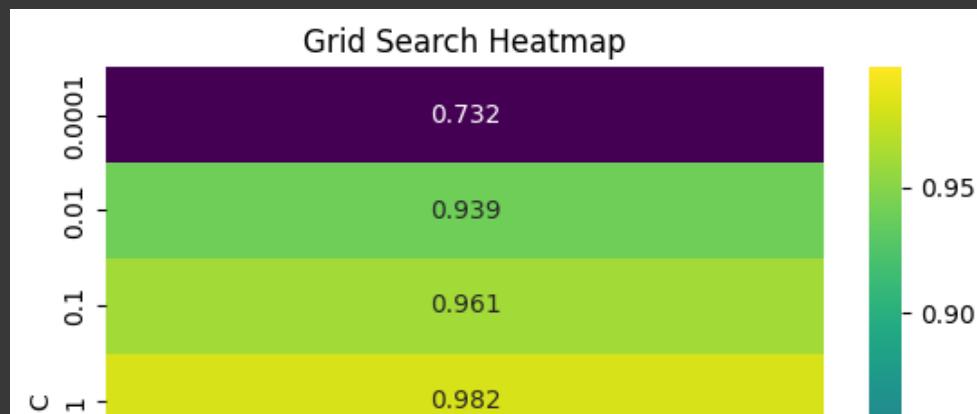
- **Penjelasan:** Mengekstrak hasil dari penelusuran parameter dan menyusunnya ke dalam DataFrame. Nilai evaluasi akurasi diambil dan diubah menjadi matriks yang sesuai dengan kombinasi parameter C dan gamma yang dieksplorasi.

7. Plot heatmap:

```
sns.heatmap(scores, annot=True, fmt=".3f", xticklabels=param_grid['gamma'], yticklabels=param_grid['C'], cmap="viridis")
plt.xlabel('Gamma')
plt.ylabel('C')
plt.title('Grid Search Heatmap')
plt.show()
```

- **Penjelasan:** Membuat heatmap untuk memvisualisasikan hasil evaluasi akurasi dari penelusuran parameter. Anotasi ditambahkan dengan format tiga angka desimal, dan label sumbu x dan y diberikan sesuai dengan nilai gamma dan C yang dieksplorasi. Colormap yang digunakan adalah 'viridis'.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from sklearn.model_selection import GridSearchCV
5
6 # Define the parameter grid
7 param_grid = {'C': [0.0001, 0.01, 0.1, 1, 10, 100, 1000], 'kernel': ['poly'], 'gamma': ['scale']}
8
9 # Create an SVM classifier
10 svm_classifier = SVC()
11
12 # Create the GridSearchCV object
13 grid_search = GridSearchCV(svm_classifier, param_grid, scoring='accuracy', cv=5)
14
15 # Fit the grid search to the data
16 grid_search.fit(X_train, y_train)
17
18 # Extract results
19 results = pd.DataFrame(grid_search.cv_results_)
20 scores = np.array(results.mean_test_score).reshape(len(param_grid['C']), len(param_grid['gamma']))
21
22 # Plot heatmap
23 sns.heatmap(scores, annot=True, fmt=".3f", xticklabels=param_grid['gamma'], yticklabels=param_grid['C'], cmap="viridis")
24 plt.xlabel('Gamma')
25 plt.ylabel('C')
26 plt.title('Grid Search Heatmap')
27 plt.show()
28
```



Catatan/Penjelasan dari Output di atas:

Grafik ini adalah peta panas yang menunjukkan hasil pencarian grid untuk klasifikasi SVM. Pencarian grid dilakukan pada grid parameter dengan nilai-nilai yang berbeda untuk C, kernel, dan gamma. Peta panas menunjukkan skor tes rata-rata untuk setiap kombinasi C dan gamma.

Nilai C dan gamma yang digunakan dalam pencarian grid adalah sebagai berikut:

- C: [0.0001, 0.01, 0.1, 1, 10, 100, 1000]
- Gamma: ['scale']

Dalam hal ini, kernel yang digunakan adalah 'poly'.

Skor tes rata-rata tertinggi adalah 0.998, yang dicapai dengan C=1000 dan gamma=scale. Ini menunjukkan bahwa kombinasi parameter terbaik untuk dataset ini adalah C=1000 dan gamma=scale.

Warna yang lebih gelap pada peta panas menunjukkan skor tes rata-rata yang lebih tinggi. Oleh karena itu, area yang lebih gelap pada peta panas menunjukkan kombinasi parameter yang menghasilkan model dengan performa terbaik.

Harap dicatat bahwa hasil ini mungkin berbeda tergantung pada dataset yang digunakan. Selalu disarankan untuk melakukan pencarian grid atau metode optimasi parameter lainnya saat bekerja dengan model machine learning untuk menemukan set parameter optimal.

Prompt chatGPT:

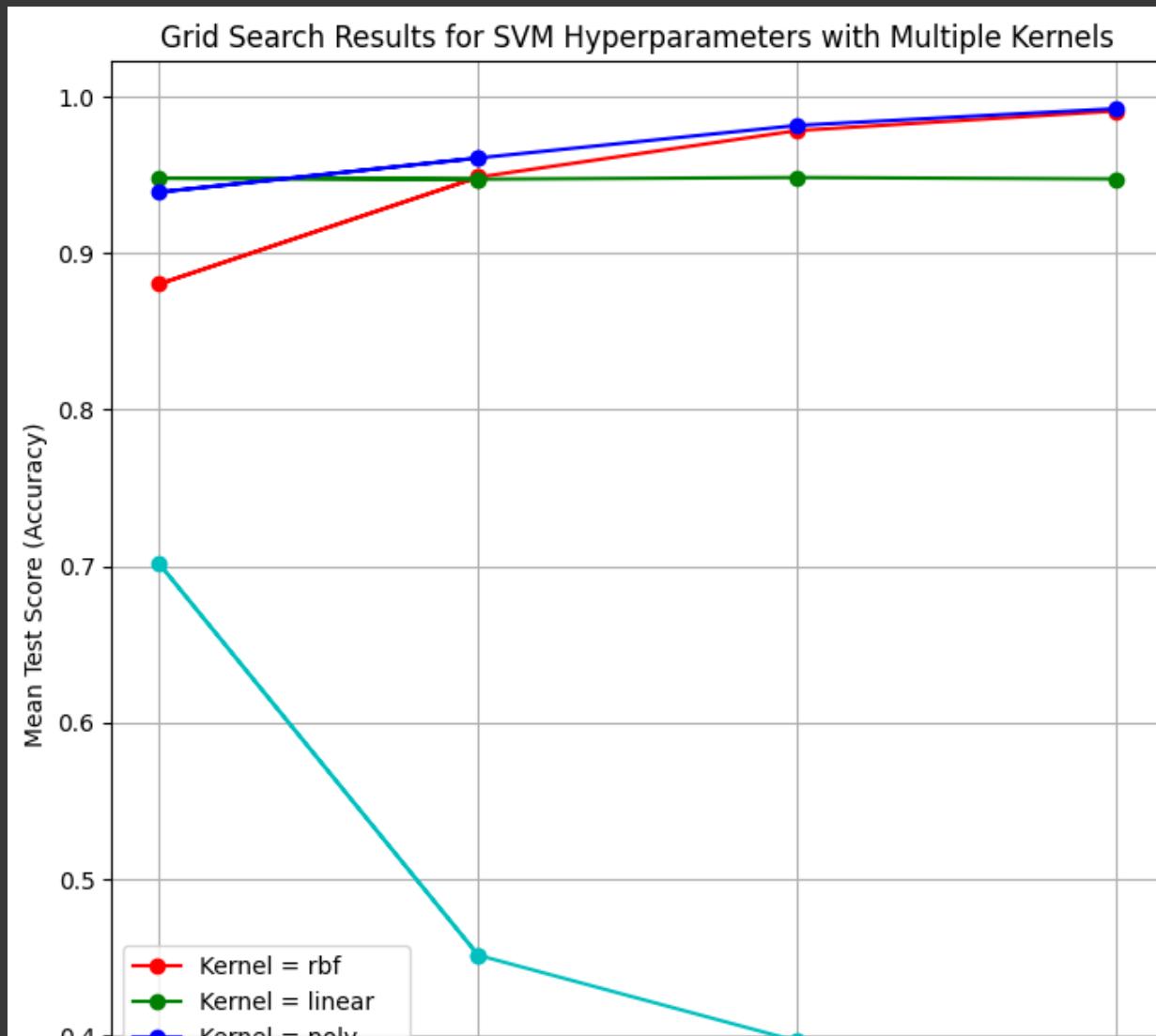
Harap tuliskan kode Python yang menggunakan Matplotlib, NumPy, dan scikit-learn untuk melakukan Grid Search pada model Support Vector Machine (SVM) dengan berbagai nilai hyperparameter. Gunakan parameter grid yang mencakup nilai C, kernel, dan gamma. Setelah itu, plot hasil Grid Search untuk berbagai kombinasi nilai C dan kernel menggunakan skor akurasi sebagai metrik evaluasi. Pastikan juga untuk menggunakan skema validasi silang (cross-validation) dengan 5 lipatan (cv=5).

Penjelasan Kode:

1. `import matplotlib.pyplot as plt`: Mengimpor modul pyplot dari pustaka Matplotlib dengan alias plt untuk membuat plot.
2. `import numpy as np`: Mengimpor modul NumPy dengan alias np untuk pengolahan array dan operasi matematika.
3. `from sklearn.svm import SVC`: Mengimpor kelas SVC (Support Vector Classification) dari pustaka scikit-learn untuk membuat model Support Vector Machine.
4. `from sklearn.model_selection import GridSearchCV`: Mengimpor kelas GridSearchCV dari pustaka scikit-learn untuk melakukan pencarian grid pada hyperparameter model.
5. `param_grid = {'C': [0.1, 0.01, 0.1, 1, 10], 'kernel': ['rbf', 'linear', 'poly', 'sigmoid'], 'gamma': ['scale']}`: Mendefinisikan grid parameter untuk hyperparameter model, termasuk nilai C, kernel, dan gamma yang akan dieksplorasi dalam pencarian grid.
6. `svm_classifier = SVC()`: Membuat objek kelas Support Vector Machine (SVM) untuk digunakan sebagai model klasifikasi.
7. `grid_search = GridSearchCV(svm_classifier, param_grid, scoring='accuracy', cv=5)`: Membuat objek GridSearchCV dengan menggunakan model SVM, grid parameter, skor evaluasi yang diukur adalah akurasi, dan skema validasi silang dengan 5 lipatan.
8. `grid_search.fit(X_train, y_train)`: Melatih model SVM menggunakan pencarian grid pada dataset pelatihan (X_train) dan label pelatihan (y_train).
9. `results = grid_search.cv_results_`: Menyimpan hasil pencarian grid dalam variabel results, termasuk skor evaluasi untuk setiap kombinasi hyperparameter.
10. `plt.figure(figsize=(8, 8))`: Membuat objek gambar (figure) dengan ukuran 8x8 inci untuk menempatkan plot hasil pencarian grid.
11. `for kernel_val, color in zip(param_grid['kernel'], ['r', 'g', 'b', 'c']):`: Mengiterasi melalui nilai kernel dan warna yang telah didefinisikan dalam grid parameter.
12. `mask = results['param_kernel'] == kernel_val`: Membuat mask untuk memfilter hasil pencarian grid berdasarkan nilai kernel saat ini.
13. `plt.plot(results['param_C'][mask], results['mean_test_score'][mask], label=f'Kernel = {kernel_val}', marker='o', color=color)`: Plot hasil pencarian grid untuk kombinasi nilai C dan kernel tertentu, dengan menggunakan skor akurasi sebagai ordinat. Setiap kernel diplot dengan warna yang sesuai.

14. `plt.xscale('log')`: Menggunakan skala logaritmik untuk sumbu x (C) guna meningkatkan visibilitas pada nilai C yang bervariasi secara besar.
15. `plt.xlabel('C')`: Menyertakan label sumbu x sebagai "C".
16. `plt.ylabel('Mean Test Score (Accuracy)')`: Menyertakan label sumbu y sebagai "Mean Test Score (Accuracy)".
17. `plt.title('Grid Search Results for SVM Hyperparameters with Multiple Kernels')`: Menyertakan judul plot.
18. `plt.legend()`: Menampilkan legenda untuk membedakan antara plot hasil pencarian grid untuk setiap kernel.
19. `plt.grid(True)`: Menambahkan grid pada plot untuk memudahkan membaca nilai.
20. `plt.show()`: Menampilkan plot hasil pencarian grid.

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from sklearn.svm import SVC
4 from sklearn.model_selection import GridSearchCV
5
6 # Define the parameter grid with multiple kernels
7 param_grid = {'C': [0.1, 0.01, 0.1, 1, 10, ], 'kernel': ['rbf', 'linear', 'poly', 'sigmoid'], 'gamma': ['scale']}
8
9 # Create an SVM classifier
10 svm_classifier = SVC()
11
12 # Create the GridSearchCV object
13 grid_search = GridSearchCV(svm_classifier, param_grid, scoring='accuracy', cv=5)
14
15 # Fit the grid search to the data
16 grid_search.fit(X_train, y_train)
17
18 # Extract results
19 results = grid_search.cv_results_
20
21 # Plot the results for different values of C and kernels
22 plt.figure(figsize=(8, 8))
23
24 for kernel_val, color in zip(param_grid['kernel'], ['r', 'g', 'b', 'c']):
25     mask = results['param_kernel'] == kernel_val
26     plt.plot(results['param_C'][mask], results['mean_test_score'][mask], label=f'Kernel = {kernel_val}', marker='o', color=color)
27
28 plt.xscale('log') # Use a logarithmic scale for better visibility
29 plt.xlabel('C')
30 plt.ylabel('Mean Test Score (Accuracy)')
31 plt.title('Grid Search Results for SVM Hyperparameters with Multiple Kernels')
32 plt.legend()
33 plt.grid(True)
34 plt.show()
35
```



Catatan/Penjelasan dari Output di atas:

Grafik ini menunjukkan hasil dari pencarian grid untuk hyperparameter dari klasifikasi SVM. Sumbu x adalah nilai C dan sumbu y adalah skor tes rata-rata (akurasi). Kernel yang digunakan adalah rbf, linear, poly, dan sigmoid.

- Skor tes rata-rata tertinggi dicapai dengan nilai C 10 dan kernel rbf. Ini menunjukkan bahwa model SVM dengan kernel rbf dan nilai C 10 memiliki performa terbaik dalam hal akurasi pada data tes.

- Skor tes rata-rata terendah dicapai dengan nilai C 0.01 dan kernel sigmoid. Ini menunjukkan bahwa model SVM dengan kernel sigmoid dan nilai C 0.01 memiliki performa terburuk dalam hal akurasi pada data tes.

Dengan demikian, dapat disimpulkan bahwa pemilihan kernel dan nilai C sangat mempengaruhi performa model SVM. Oleh karena itu, penting untuk melakukan pencarian grid untuk menemukan kombinasi hyperparameter terbaik.

Harap dicatat bahwa hasil ini mungkin berbeda tergantung pada dataset yang digunakan. Oleh karena itu, selalu disarankan untuk melakukan pencarian grid atau metode optimasi hyperparameter lainnya saat bekerja dengan model SVM atau model machine learning lainnya.

▼ Inputting new data

Input data baru dalam mencoba model Support Vector Machine (SVM) dapat dilakukan dengan menyajikan informasi fitur-fitur dari data tersebut ke dalam model. Informasi fitur-fitur ini kemudian digunakan oleh SVM untuk melakukan klasifikasi atau regresi, tergantung pada tujuan pemodelan yang diinginkan. Proses ini melibatkan pemetaan data ke dalam ruang dimensi yang lebih tinggi, di mana SVM akan mencari batas keputusan optimal yang memisahkan kelas-kelas atau memprediksi nilai target.

Langkah-langkah umumnya melibatkan:

1. Memasukkan nilai fitur-fitur data baru ke dalam model SVM.
2. Model SVM akan melakukan transformasi data ke dalam ruang dimensi yang lebih tinggi sesuai dengan fungsi kernel yang digunakan.
3. SVM kemudian menentukan posisi data baru dalam ruang tersebut.
4. Berdasarkan posisi tersebut, SVM membuat keputusan klasifikasi atau regresi sesuai dengan aturan yang telah dipelajari selama pelatihan.

Penting untuk dicatat bahwa data baru yang dimasukkan ke dalam model harus memiliki struktur yang sama dengan data yang digunakan selama pelatihan, yaitu memiliki jumlah fitur yang sama dengan tipe data yang sesuai.

▼ EDA

Prompt chatGPT:

Tuliskan kode untuk membaca beberapa sheet dari file Excel dan menggabungkannya menjadi satu DataFrame menggunakan pandas.

Penjelasan Kode:

1. `sheet_names = ['DKI1', 'DKI2', 'DKI3', 'DKI4', 'DKI5']`: Tentukan daftar nama sheet yang ingin dibaca dari file Excel.
2. `file_path = '/content/drive/MyDrive/Dataset/Data ISPU Tahun 2016.xlsx'`: Tentukan path atau lokasi file Excel yang akan dibaca.
3. `all_data = pd.read_excel(file_path, sheet_name=sheet_names)`: Gunakan pandas untuk membaca file Excel dengan menggunakan `pd.read_excel`. Parameter `sheet_name` digunakan untuk menentukan sheet-sheet yang akan dibaca dan hasilnya disimpan dalam bentuk dictionary of DataFrames.
4. `for sheet_name, df in all_data.items():`: Loop melalui setiap pasangan key-value pada dictionary `all_data`. `sheet_name` akan berisi nama sheet, dan `df` akan berisi DataFrame yang sesuai.
5. `print(f"Beberapa baris pertama dari {sheet_name}:")`: Cetak nama sheet sebagai judul untuk setiap DataFrame.
6. `print(df.head())`: Cetak beberapa baris pertama dari DataFrame menggunakan metode `head()`.
7. `print("\n")`: Cetak baris kosong untuk memisahkan output setiap sheet.
8. `dataudara = pd.concat(all_data.values(), ignore_index=True)`: Gabungkan semua DataFrame dalam dictionary menjadi satu DataFrame tunggal menggunakan `pd.concat`. Parameter `ignore_index=True` digunakan untuk mengatur ulang indeks DataFrame yang dihasilkan.

```
1 # Tentukan nama-nama sheet yang ingin dibaca
2 sheet_names = ['DKI1', 'DKI2', 'DKI3', 'DKI4', 'DKI5']
3
4 # Berikan path file
5 file_path = '/content/drive/MyDrive/Dataset/Data ISPU Tahun 2016.xlsx'
6
7 # Baca semua sheet yang telah ditentukan ke dalam sebuah dictionary DataFrame
8 all_data = pd.read_excel(file_path, sheet_name=sheet_names)
9
10 # Loop melalui setiap sheet dan DataFrame pada dictionary
11 for sheet_name, df in all_data.items():
12     print(f"Beberapa baris pertama dari {sheet_name}:")
13     print(df.head())
14     print("\n")
15
16 # Gabungkan semua DataFrames menjadi satu DataFrame menggunakan pd.concat
17 dataudara = pd.concat(all_data.values(), ignore_index=True)
```

18

Beberapa baris pertama dari DKI2:

```
Tanggal Wilayah PM10 SO2 CO O3 NO2 Max Critical \
0 2016-01-01 DKI2 (Kelapa Gading) 69 19 42 40 10 69 PM10
1 2016-01-02 DKI2 (Kelapa Gading) 40 20 15 44 12 44 03
2 2016-01-03 DKI2 (Kelapa Gading) 32 18 24 43 18 43 03
3 2016-01-04 DKI2 (Kelapa Gading) 61 27 21 103 24 103 03
4 2016-01-05 DKI2 (Kelapa Gading) 61 21 31 67 22 67 03
```

Kategori

```
0 SEDANG
1 BAIK
2 BAIK
3 TIDAK SEHAT
4 SEDANG
```

Beberapa baris pertama dari DKI3:

```
Tanggal Wilayah PM10 SO2 CO O3 NO2 Max Critical Kategori
0 2016-01-01 DKI3 (Jagakarsa) 52 17 53 51 11.0 53 CO SEDANG
1 2016-01-02 DKI3 (Jagakarsa) 52 16 29 50 5.0 50 03 BAIK
2 2016-01-03 DKI3 (Jagakarsa) 52 16 33 50 5.0 50 03 BAIK
3 2016-01-04 DKI3 (Jagakarsa) 52 18 36 71 9.0 71 03 SEDANG
4 2016-01-05 DKI3 (Jagakarsa) 52 16 54 46 11.0 54 CO SEDANG
```

Beberapa baris pertama dari DKI4:

```
Tanggal Wilayah PM10 SO2 CO O3 NO2 Max Critical \

```

	Ranggai	Wilayah	PM10	SO2	CO	O3	NO2	\
0	2016-01-01	DKI5 (Kebon Jeruk)	Jakarta Barat	54	11	44.0	33.0	4
1	2016-01-02	DKI5 (Kebon Jeruk)	Jakarta Barat	43	11	44.0	30.0	5
2	2016-01-03	DKI5 (Kebon Jeruk)	Jakarta Barat	38	10	44.0	37.0	4
3	2016-01-04	DKI5 (Kebon Jeruk)	Jakarta Barat	51	12	44.0	67.0	8
4	2016-01-05	DKI5 (Kebon Jeruk)	Jakarta Barat	68	11	44.0	50.0	7

	Max	Critical	Kategori
0	54	PM10	SEDANG
1	43	PM10	BAIK
2	38	PM10	BAIK
3	67	O3	SEDANG
4	68	PM10	SEDANG

Catatan/Penjelasan dari Output di atas:

Output di atas menunjukkan beberapa baris pertama dari setiap sheet yang telah dibaca dari file Excel. Setiap sheet mewakili data kualitas udara untuk wilayah tertentu di DKI Jakarta pada tahun 2016. Berikut adalah catatan untuk setiap sheet:

DKI1:

- **Wilayah:** DKI1 (Bunderan HI)
- **Variabel yang Diukur:** PM10, SO2, CO, O3, NO2
- **Beberapa Baris Pertama:**
 - Tanggal: 2016-01-01 hingga 2016-01-05
 - Kategori: SEDANG, SEDANG, BAIK, SEDANG, SEDANG

DKI2:

- **Wilayah:** DKI2 (Kelapa Gading)
- **Variabel yang Diukur:** PM10, SO2, CO, O3, NO2
- **Beberapa Baris Pertama:**
 - Tanggal: 2016-01-01 hingga 2016-01-05
 - Kategori: SEDANG, BAIK, BAIK, TIDAK SEHAT, SEDANG

DKI3:

- **Wilayah:** DKI3 (Jagakarsa)
- **Variabel yang Diukur:** PM10, SO2, CO, O3, NO2
- **Beberapa Baris Pertama:**
 - Tanggal: 2016-01-01 hingga 2016-01-05
 - Kategori: SEDANG, BAIK, BAIK, SEDANG, SEDANG

DKI4:

- **Wilayah:** DKI4 (Lubang Buaya)
- **Variabel yang Diukur:** PM10, SO2, CO, O3, NO2
- **Beberapa Baris Pertama:**
 - Tanggal: 2016-01-01 hingga 2016-01-05
 - Kategori: SEDANG, BAIK, BAIK, SEDANG, SEDANG

DKI5:

- **Wilayah:** DKI5 (Kebon Jeruk) Jakarta Barat
- **Variabel yang Diukur:** PM10, SO2, CO, O3, NO2
- **Beberapa Baris Pertama:**
 - Tanggal: 2016-01-01 hingga 2016-01-05
 - Kategori: SEDANG, BAIK, BAIK, SEDANG, SEDANG

Catatan: Setiap sheet mencakup data untuk beberapa hari (dalam contoh ini, lima hari pertama tahun 2016), dan kategori kualitas udara diberikan berdasarkan standar yang ditentukan.

Prompt chatGPT:

Tulis kode untuk menampilkan informasi tentang objek 'dataudara'.

Penjelasan Kode:

Kode di atas adalah perintah dalam bahasa pemrograman untuk menampilkan informasi tentang objek 'dataudara' menggunakan metode 'info()'. Berikut adalah penjelasan rinci untuk setiap baris kode:

1. `dataudara.info()`: Ini adalah pemanggilan metode 'info()' pada objek 'dataudara'. Metode ini biasanya digunakan untuk menampilkan informasi tentang struktur dan tipe data dalam objek DataFrame pada library seperti Pandas di Python.

- `dataudara`: Merupakan nama objek DataFrame yang akan ditampilkan informasinya.
- `info()`: Metode yang digunakan untuk menampilkan ringkasan informasi tentang DataFrame, termasuk jumlah entri, tipe data setiap kolom, dan jumlah nilai non-null.

Penjelasan singkat: Kode ini digunakan untuk mendapatkan gambaran umum tentang struktur dan karakteristik dari objek DataFrame 'dataudara', termasuk informasi tentang kolom-kolomnya dan jenis data yang terkandung di dalamnya.

```
1 # Ini adalah kode untuk menampilkan informasi tentang objek 'data'  
2 dataudara.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1830 entries, 0 to 1829  
Data columns (total 10 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          -----          ----  
 0   Tanggal     1830 non-null   datetime64[ns]  
 1   Wilayah     1830 non-null   object  
 2   PM10        1830 non-null   int64  
 3   SO2          1830 non-null   int64  
 4   CO           1829 non-null   float64  
 5   O3           1827 non-null   float64  
 6   NO2          1826 non-null   float64  
 7   Max          1830 non-null   int64  
 8   Critical     1794 non-null   object  
 9   Kategori     1830 non-null   object  
dtypes: datetime64[ns](1), float64(3), int64(3), object(3)  
memory usage: 143.1+ KB
```

Catatan/Penjelasan dari Output di atas:

Output tersebut adalah hasil dari menjalankan metode `info()` pada objek DataFrame 'dataudara'. Berikut adalah catatan dan penjelasan dari output tersebut:

1. Info Umum DataFrame:

- **Class DataFrame:** Menunjukkan bahwa objek ini adalah instans dari kelas DataFrame dari library Pandas.

- **RangIndex:** Memberikan informasi tentang indeks DataFrame, dalam hal ini, merupakan RangIndex dengan total entri sebanyak 1830, dimulai dari 0 hingga 1829.
- **Jumlah Kolom dan Entri:** Terdapat 10 kolom dalam DataFrame ini dengan total 1830 entri.

2. Informasi Kolom:

- **Column:** Nama kolom dalam DataFrame.
- **Non-Null Count:** Jumlah nilai non-null (tidak kosong) dalam setiap kolom.
- **Dtype:** Tipe data dari nilai dalam kolom.

3. Detail Kolom:

- **Tanggal:** Kolom dengan tipe data datetime64[ns], dan memiliki 1830 entri non-null.
- **Wilayah:** Kolom dengan tipe data object (teks), dan memiliki 1830 entri non-null.
- **PM10, SO2, Max:** Kolom-kolom dengan tipe data int64, dan memiliki 1830 entri non-null.
- **CO, O3, NO2:** Kolom-kolom dengan tipe data float64, dengan beberapa nilai null (kurang dari 1830 non-null).
- **Critical:** Kolom dengan tipe data object (teks), dan memiliki 1794 entri non-null.
- **Kategori:** Kolom dengan tipe data object (teks), dan memiliki 1830 entri non-null.

4. Memory Usage:

- **Memory Usage:** Menunjukkan perkiraan penggunaan memori oleh DataFrame, dalam hal ini, sekitar 143.1 KB.

Catatan: Output ini memberikan wawasan tentang struktur DataFrame, tipe data, serta keberadaan nilai null pada setiap kolomnya. Dengan informasi ini, pengguna dapat mengevaluasi kualitas dan integritas data dalam DataFrame 'dataudara'.

Prompt chatGPT:

Konversikan kolom 'Kategori' dan 'Critical' dalam DataFrame 'dataudara' menjadi tipe data kategori. Setelah itu, tampilkan informasi DataFrame setelah perubahan tipe data.

Penjelasan Kode:

1. `dataudara['Kategori'] = dataudara['Kategori'].astype('category')`: Baris ini mengubah tipe data kolom 'Kategori' dalam DataFrame 'dataudara' menjadi tipe data kategori. Dengan mengubah tipe data menjadi kategori, kita dapat menghemat ruang memori dan meningkatkan efisiensi dalam pemrosesan data kategori.
2. `dataudara['Critical'] = dataudara['Critical'].astype('category')`: Baris ini mengubah tipe data kolom 'Critical' dalam DataFrame 'dataudara' menjadi tipe data kategori. Seperti pada langkah sebelumnya, ini bertujuan untuk mengoptimalkan penggunaan memori dan meningkatkan kinerja.
3. `print(dataudara.info())`: Baris ini mencetak informasi tentang DataFrame 'dataudara' setelah perubahan tipe data dilakukan. Fungsi `info()` menyediakan informasi rinci tentang DataFrame, termasuk tipe data setiap kolom, jumlah nilai non-null, dan penggunaan memori. Dengan mencetak informasi ini, kita dapat memverifikasi apakah perubahan tipe data telah dilakukan dengan sukses dan memahami properti DataFrame secara lebih detail.

```
1 # Mengubah kolom 'Kategori' dan 'Critical' menjadi tipe data kategori
2 dataudara['Kategori'] = dataudara['Kategori'].astype('category')
3
4 # Mengubah kolom 'Critical' menjadi tipe data kategori
5 dataudara['Critical'] = dataudara['Critical'].astype('category')
6
7 # Menampilkan informasi DataFrame setelah perubahan tipe data
8 print(dataudara.info())
9
```

```
<class 'pandas.core.frame.DataFrame'\>
RangeIndex: 1830 entries, 0 to 1829
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          ----- 
 0   Tanggal     1830 non-null   datetime64[ns]
 1   Wilayah    1830 non-null   object  
 2   PM10        1830 non-null   int64   
 3   SO2         1830 non-null   int64   
 4   CO          1829 non-null   float64 
 5   O3          1827 non-null   float64 
 6   NO2        1826 non-null   float64 
 7   Max         1830 non-null   int64   
 8   Critical    1794 non-null   category 
 9   Kategori    1830 non-null   category 
dtypes: category(2), datetime64[ns](1), float64(3), int64(3), object(1)
```

```
memory usage: 118.5+ KB
None
```

Catatan/Penjelasan dari Output di atas:

Output tersebut adalah informasi dari DataFrame 'dataudara' setelah perubahan tipe data pada kolom 'Critical' dan 'Kategori' dilakukan. Berikut adalah catatan dan penjelasan dari output tersebut:

1. RangeIndex dan Jumlah Baris:

- RangeIndex: 1830 entries, 0 to 1829: Menunjukkan bahwa DataFrame memiliki 1830 baris data, dengan indeks berupa rentang dari 0 hingga 1829.

2. Info Kolom:

- Data columns (total 10 columns): Menyajikan total 10 kolom dalam DataFrame.

3. Informasi Setiap Kolom:

- Tanggal: Tipe data datetime64[ns]. Semua 1830 entri memiliki nilai non-null.
- Wilayah: Tipe data object (mungkin string atau mixed types). Semua 1830 entri memiliki nilai non-null.
- PM10, SO2, Max: Tipe data int64. Semua 1830 entri memiliki nilai non-null.
- CO, O3, NO2: Tipe data float64. Beberapa kolom memiliki nilai null, seperti 'CO' (1 null), 'O3' (3 null), dan 'NO2' (4 null).
- Critical, Kategori: Tipe data category. Semua 1830 entri memiliki nilai non-null pada kolom 'Kategori', tetapi 'Critical' memiliki 36 nilai null.

4. Memory Usage:

- memory usage: 118.5+ KB: Menunjukkan penggunaan memori DataFrame setelah perubahan tipe data. Penggunaan memori ini dapat membantu memahami efisiensi penggunaan memori setelah melakukan perubahan tipe data menjadi kategori pada kolom tertentu.

Prompt chatGPT:

Ubahlah tipe data kolom 'PM10' menjadi float, 'SO2' menjadi float, dan 'Max' menjadi float dalam DataFrame 'dataudara'. Setelah itu, tampilkan informasi DataFrame setelah perubahan tipe data.

Penjelasan Kode:

1. `dataudara['PM10'] = dataudara['PM10'].astype('float')`: Baris ini mengubah tipe data kolom 'PM10' dalam DataFrame 'dataudara' menjadi float. Fungsi `astype('float')` digunakan untuk mengkonversi tipe data kolom menjadi float.
2. `dataudara['SO2'] = dataudara['SO2'].astype('float')`: Baris ini melakukan hal yang serupa dengan baris pertama, namun kali ini untuk kolom 'SO2'.
3. `dataudara['Max'] = dataudara['Max'].astype('float')`: Baris ini kembali melakukan hal yang sama, mengubah tipe data kolom 'Max' menjadi float.
4. `print(dataudara.info())`: Baris ini menampilkan informasi tentang DataFrame 'dataudara' setelah perubahan tipe data. Fungsi `info()` memberikan ringkasan informasi seperti jumlah entri, tipe data, dan penggunaan memori untuk setiap kolom dalam DataFrame. Output dari `info()` akan dicetak dan ditampilkan.

```
1 # Mengubah kolom 'PM10' menjadi tipe data float
2 dataudara['PM10'] = dataudara['PM10'].astype('float')
3
4 # Mengubah kolom 'SO2' menjadi tipe data float
5 dataudara['SO2'] = dataudara['SO2'].astype('float')
6
7 # Mengubah kolom 'Max' menjadi tipe data float
8 dataudara['Max'] = dataudara['Max'].astype('float')
9
10 # Menampilkan informasi DataFrame setelah perubahan tipe data
11 print(dataudara.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1830 entries, 0 to 1829
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   Tanggal     1830 non-null   datetime64[ns]
 1   Wilayah    1830 non-null   object  
 2   PM10        1830 non-null   float64 
 3   SO2         1830 non-null   float64 
 4   CO          1829 non-null   float64 
 5   O3          1827 non-null   float64 
 6   NO2        1826 non-null   float64 
 7   Max         1830 non-null   float64
```

```
8   Critical  1794 non-null  category
9   Kategori  1830 non-null  category
dtypes: category(2), datetime64[ns](1), float64(6), object(1)
memory usage: 118.5+ KB
None
```

Catatan/Penjelasan dari Output di atas:

1. **DataFrame Info:** Output pertama menunjukkan informasi tentang DataFrame, termasuk:

- **RangIndex:** Menyajikan informasi tentang indeks DataFrame. Di sini, RangIndex menunjukkan bahwa indeksnya berupa rentang nilai dari 0 hingga 1829.
- **Total Entri:** Terdapat 1830 entri dalam DataFrame.
- **Total Kolom:** Terdapat 10 kolom dalam DataFrame.
- **Tipe Data Kolom:** Menampilkan tipe data untuk setiap kolom, seperti datetime64, float64, object, dan category.

2. **Detail Kolom:**

- **Column:** Nama kolom dalam DataFrame.
- **Non-Null Count:** Jumlah non-null (tidak bernilai null) untuk setiap kolom.
- **Dtype:** Tipe data untuk setiap kolom.

3. **Tipe Data Khusus:** Terdapat beberapa kolom dengan tipe data khusus:

- **datetime64[ns]:** Kolom 'Tanggal' memiliki tipe data datetime64, yang umumnya digunakan untuk data tanggal dan waktu.
- **category:** Kolom 'Critical' dan 'Kategori' memiliki tipe data category, yang cocok untuk kolom dengan sejumlah nilai yang terbatas.

4. **Missing Values (NaN):** Beberapa kolom memiliki nilai null (NaN):

- Kolom 'CO' memiliki 1 nilai null.
- Kolom 'O3' memiliki 3 nilai null.
- Kolom 'NO2' memiliki 4 nilai null.
- Kolom 'Critical' memiliki 36 nilai null.

5. **Memory Usage:** Menunjukkan penggunaan memori oleh DataFrame. Di sini, DataFrame menggunakan sekitar 118.5 KB memori.

Output ini memberikan pemahaman yang baik tentang struktur dan kualitas data dalam DataFrame 'dataudara', termasuk tipe data, nilai null, dan penggunaan memori.

Prompt chatGPT:

Download file CSV dari Google Drive menggunakan Colab.

Penjelasan Kode:

Baris pertama dari kode menggunakan perintah `from google.colab import files` untuk mengimpor modul `files` dari pustaka Colab Google. Modul ini menyediakan fungsionalitas untuk mengelola file dalam lingkungan Colab.

Baris kedua merupakan komentar yang memberikan penjelasan terhadap tindakan yang akan dilakukan selanjutnya, yaitu mengunduh file CSV. Komentar ini digunakan untuk dokumentasi dan penjelasan kode.

Baris ketiga adalah pemanggilan fungsi `download` dari modul `files`. Fungsi ini digunakan untuk mengunduh file dari Colab, dan sebagai argumennya, kita menyediakan path atau lokasi file yang ingin diunduh. Dalam hal ini, path tersebut adalah

```
'/content/drive/MyDrive/Dataset/Data ISPU Tahun 2016.xlsx'.
```

Jadi, secara keseluruhan, kode tersebut digunakan untuk mengunduh file dengan path tertentu dari Google Drive dalam lingkungan Colab menggunakan modul `files`.

```
1 from google.colab import files  
2  
3 # Unduh file CSV  
4 files.download('/content/drive/MyDrive/Dataset/Data ISPU Tahun 2016.xlsx')
```

Catatan/Penjelasan dari Output di atas:

Kode yang diberikan berfungsi untuk mengunduh file dengan path `/content/drive/MyDrive/Dataset/Data ISPU Tahun 2016.xlsx` dari Google Drive dalam lingkungan Colab. Namun, karena fungsi `files.download` hanya berinteraksi dengan server Colab dan tidak memberikan output yang dapat ditampilkan di notebook, maka tidak akan ada output yang terlihat secara langsung dalam notebook tersebut.

Sebagai gantinya, tindakan ini akan memicu unduhan file ke perangkat lokal pengguna. Pengguna dapat memeriksa direktori unduhan di perangkat mereka untuk menemukan file yang telah diunduh. Jadi, output sebenarnya terjadi di luar notebook Colab, di perangkat pengguna yang menjalankan kode ini.

Prompt chatGPT:

Buatlah kode Python untuk mendapatkan nilai unik dari kolom 'Kategori' dan 'Critical' dalam DataFrame 'dataudara'.

Double-click (or enter) to edit

Penjelasan Kode:

1. `unique_kategori_values = dataudara['Kategori'].unique()`

- Ini adalah baris pertama yang digunakan untuk mengambil nilai unik dari kolom 'Kategori' dalam DataFrame 'dataudara'.
- `dataudara['Kategori']` mengakses kolom 'Kategori' dari DataFrame.
- Metode `.unique()` digunakan untuk mendapatkan nilai unik dari kolom tersebut, dan hasilnya disimpan dalam variabel `unique_kategori_values`.

2. `unique_kategori_values1 = dataudara['Critical'].unique()`

- Ini adalah baris kedua yang digunakan untuk mengambil nilai unik dari kolom 'Critical' dalam DataFrame 'dataudara'.
- `dataudara['Critical']` mengakses kolom 'Critical' dari DataFrame.
- Metode `.unique()` digunakan untuk mendapatkan nilai unik dari kolom tersebut, dan hasilnya disimpan dalam variabel `unique_kategori_values1`.

3. `print("Unique values for 'Kategori' column:")`

- Baris ini mencetak pesan yang memberi tahu bahwa nilai unik untuk kolom 'Kategori' akan ditampilkan.

4. `print(unique_kategori_values)`

- Baris ini mencetak nilai unik yang telah diambil dari kolom 'Kategori'. Hasilnya adalah daftar nilai unik tersebut.

5. `print("Unique values for 'Critical' column:")`

- Baris ini mencetak pesan yang memberi tahu bahwa nilai unik untuk kolom 'Critical' akan ditampilkan.

6. `print(unique_kategori_values1)`

- Baris ini mencetak nilai unik yang telah diambil dari kolom 'Critical'. Hasilnya adalah daftar nilai unik tersebut.

```
1 # Mengambil nilai unik dari kolom 'Kategori'  
2 unique_kategori_values = dataudara['Kategori'].unique()  
3  
4 # Mengambil nilai unik dari kolom 'Critical'  
5 unique_kategori_values1 = dataudara['Critical'].unique()  
6  
7 # Menampilkan nilai unik untuk kolom 'Kategori'  
8 print("Unique values for 'Kategori' column:")  
9 print(unique_kategori_values)  
10  
11 # Menampilkan nilai unik untuk kolom 'Critical'  
12 print("Unique values for 'Critical' column:")  
13 print(unique_kategori_values1)  
14
```

```
Unique values for 'Kategori' column:  
['SEDANG', 'BAIK', 'TIDAK SEHAT', 'TIDAK ADA DATA', 'SANGAT TIDAK SEHAT']  
Categories (5, object): ['BAIK', 'SANGAT TIDAK SEHAT', 'SEDANG', 'TIDAK ADA DATA',  
 'TIDAK SEHAT']  
Unique values for 'Critical' column:  
['PM10', 'O3', 'SO2', 'CO', NaN, ' ', 'NO2']  
Categories (6, object): [' ', 'CO', 'NO2', 'O3', 'PM10', 'SO2']
```

Catatan/Penjelasan dari Output di atas:

Output di atas memberikan informasi tentang nilai unik yang diambil dari kolom 'Kategori' dan 'Critical' dalam DataFrame 'dataudara'. Berikut adalah catatan/penjelasan dari output tersebut:

▼ Untuk kolom 'Kategori':

- Nilai unik yang ditemukan: 'SEDANG', 'BAIK', 'TIDAK SEHAT', 'TIDAK ADA DATA', 'SANGAT TIDAK SEHAT'.
- Ada kategori yang muncul dalam output yang tidak terlihat dalam nilai unik ('Categories' baris): 'BAIK', 'SANGAT TIDAK SEHAT', 'SEDANG', 'TIDAK ADA DATA', 'TIDAK SEHAT'.
- Kategori memiliki tipe data kategori ('Categories (5, object)').

Untuk kolom 'Critical':

- Nilai unik yang ditemukan: 'PM10', 'O3', 'SO2', 'CO', NaN, ' ', 'NO2'.
- Ada kategori yang muncul dalam output yang tidak terlihat dalam nilai unik ('Categories' baris): ' ', 'CO', 'NO2', 'O3', 'PM10', 'SO2'.

- Kolom 'Critical' memiliki beberapa nilai yang tidak jelas, seperti 'NaN', '', dan memiliki tipe data kategori ('Categories (6, object)').

Catatan: Kategori pada umumnya harus bersih dan sesuai, sehingga perlu diperhatikan dan mungkin diperlukan pemrosesan lebih lanjut tergantung pada kebutuhan analisis data.

Prompt chatGPT:

Ubahlah kolom 'Kategori' dan 'Critical' dalam DataFrame 'dataudara' menjadi huruf besar. Kemudian, hapus entri 'TIDAK ADA DATA', 'Tidak Ada Data', dan nilai NaN dari kedua kolom tersebut. Terakhir, tampilkan nilai unik dari kolom 'Kategori' setelah modifikasi.

Penjelasan Kode:

1. `dataudara['Kategori'] = dataudara['Kategori'].str.upper()`: Ini adalah perintah untuk mengonversi semua nilai dalam kolom 'Kategori' DataFrame 'dataudara' menjadi huruf besar (uppercase).
2. `dataudara['Critical'] = dataudara['Critical'].str.upper()`: Sama seperti baris sebelumnya, ini mengonversi semua nilai dalam kolom 'Critical' DataFrame 'dataudara' menjadi huruf besar (uppercase).
3. `dataudara = dataudara[~dataudara['Kategori'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]`: Baris ini melakukan penghapusan baris yang memiliki nilai 'TIDAK ADA DATA', 'Tidak Ada Data', atau NaN dalam kolom 'Kategori' DataFrame 'dataudara'. Operator `~` digunakan untuk membalik seleksi, sehingga hanya baris dengan nilai lain yang tetap.
4. `dataudara = dataudara[~dataudara['Critical'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]`: Mirip dengan baris sebelumnya, ini menghapus baris dengan nilai 'TIDAK ADA DATA', 'Tidak Ada Data', atau NaN dalam kolom 'Critical' DataFrame 'dataudara'.
5. `unique_kategori_values = dataudara['Kategori'].unique()`: Baris ini membuat variabel `unique_kategori_values` yang berisi nilai unik dari kolom 'Kategori' setelah dilakukan modifikasi.
6. `print("Nilai unik untuk kolom 'Kategori':")`: Menampilkan teks yang memberikan informasi bahwa nilai yang akan ditampilkan adalah nilai unik dari kolom 'Kategori'.
7. `print(unique_kategori_values)`: Menampilkan nilai unik dari kolom 'Kategori' setelah modifikasi dalam DataFrame 'dataudara'.

```
1 # Mengasumsikan 'dataudara' adalah DataFrame
2 # Mengonversi kolom 'Kategori' dan 'Critical' menjadi huruf besar
3 dataudara['Kategori'] = dataudara['Kategori'].str.upper()
4 dataudara['Critical'] = dataudara['Critical'].str.upper()
5
6 # Menghapus 'TIDAK ADA DATA', 'Tidak Ada Data', dan NaN dari kolom 'Kategori'
7 dataudara = dataudara[~dataudara['Kategori'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]
8
9 # Menghapus 'TIDAK ADA DATA', 'Tidak Ada Data', dan NaN dari kolom 'Critical'
10 dataudara = dataudara[~dataudara['Critical'].isin(['TIDAK ADA DATA', 'Tidak Ada Data', np.nan])]
11
12 # Menampilkan nilai unik untuk kolom 'Kategori' setelah dimodifikasi
13 unique_kategori_values = dataudara['Kategori'].unique()
14 print("Nilai unik untuk kolom 'Kategori':")
15 print(unique_kategori_values)
16
17
```

```
Nilai unik untuk kolom 'Kategori':
['SEDANG' 'BAIK' 'TIDAK SEHAT' 'SANGAT TIDAK SEHAT']
```

Catatan/Penjelasan dari Output di atas:

Output tersebut menunjukkan nilai unik yang tersisa setelah dilakukan serangkaian operasi pada kolom 'Kategori' dalam DataFrame 'dataudara'.

Berikut adalah penjelasan dari nilai unik tersebut:

1. 'SEDANG': Menunjukkan kategori kualitas udara yang sedang, artinya kualitas udara pada tingkat ini dianggap cukup baik.
2. 'BAIK': Merupakan kategori untuk kualitas udara yang baik, menandakan bahwa tingkat polutan dalam udara pada tingkat ini dianggap rendah.
3. 'TIDAK SEHAT': Menunjukkan bahwa kualitas udara pada tingkat ini dianggap tidak sehat, yang berarti terdapat tingkat polutan yang dapat berdampak buruk pada kesehatan.
4. 'SANGAT TIDAK SEHAT': Merupakan kategori tertinggi yang menandakan bahwa kualitas udara pada tingkat ini dianggap sangat tidak sehat, dengan tingkat polutan yang sangat tinggi dan berpotensi memberikan dampak serius pada kesehatan manusia.

Prompt chatGPT:

Tulislah perintah untuk mengambil nilai unik dari kolom 'Critical' setelah dimodifikasi dan menampilkan nilai unik tersebut.

Penjelasan Kode:

1. Mengambil nilai unik dari kolom 'Critical' setelah dimodifikasi:

```
unique_critical_values = dataudara['Critical'].unique()
```

Baris ini membuat variabel `unique_critical_values` yang akan menyimpan nilai unik dari kolom 'Critical' setelah dimodifikasi. Fungsi `.unique()` digunakan untuk mengambil nilai unik dari kolom tersebut.

2. Menampilkan nilai unik untuk kolom 'Critical':

```
print("Nilai unik untuk kolom 'Critical':")
print(unique_critical_values)
```

Baris ini mencetak teks "Nilai unik untuk kolom 'Critical': " sebagai judul atau label, kemudian mencetak nilai unik yang telah disimpan di dalam variabel `unique_critical_values`. Ini memberikan informasi tentang nilai-nilai unik yang ada dalam kolom 'Critical'.

```
1 # Mengambil nilai unik dari kolom 'Critical' setelah dimodifikasi
2 unique_critical_values = dataudara['Critical'].unique()
3
4 # Menampilkan nilai unik untuk kolom 'Critical'
5 print("Nilai unik untuk kolom 'Critical':")
6 print(unique_critical_values)
7
```

```
Nilai unik untuk kolom 'Critical':
['PM10' 'O3' 'SO2' 'CO' 'NO2']
```

Catatan/Penjelasan dari Output di atas:

Output tersebut menunjukkan nilai unik yang ditemukan dalam kolom 'Critical'. Berikut adalah catatan/penjelasan dari output tersebut:

- 'Critical' memiliki lima nilai unik, yaitu 'PM10', 'O3', 'SO2', 'CO', dan 'NO2'.

- Setiap nilai merepresentasikan jenis atau kategori tertentu dari data yang mungkin dianggap kritis dalam konteks kualitas udara, seperti partikulat matter (PM10), ozon (O₃), sulfur dioxide (SO₂), karbon monoksida (CO), dan nitrogen dioksida (NO₂).
- Output ini memberikan pemahaman tentang variasi jenis kategori yang mungkin menjadi fokus penting atau kritis dalam analisis atau pemantauan kualitas udara.

Prompt chatGPT:

"Berikan kode yang menggunakan fungsi dataudara.info() untuk mendapatkan informasi tentang data udara.

Penjelasan Kode:

1. `dataudara` : Ini adalah suatu objek atau variabel yang mengandung data udara. Nama ini mungkin merujuk pada suatu struktur data seperti DataFrame jika Anda menggunakan library seperti Pandas.
2. `.info()` : Ini adalah metode yang digunakan pada objek `dataudara`. Metode ini biasanya digunakan untuk memberikan informasi terkait dengan struktur dan konten dari objek data, terutama jika objek tersebut adalah DataFrame Pandas. Informasi tersebut dapat mencakup jumlah baris dan kolom, tipe data setiap kolom, dan jumlah nilai yang tidak null.

Jadi, perintah tersebut dapat diartikan sebagai permintaan untuk menampilkan informasi terperinci tentang data udara yang disimpan dalam objek `dataudara`.

```
1 # Menggunakan fungsi dataudara.info() untuk mendapatkan informasi tentang data udara
2 dataudara.info()
3
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1793 entries, 0 to 1829
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          ----- 
 0   Tanggal     1793 non-null   datetime64[ns]
 1   Wilayah    1793 non-null   object  
 2   PM10        1793 non-null   float64 
 3   SO2         1793 non-null   float64 
 4   CO          1792 non-null   float64 
 5   O3          1790 non-null   float64 
 6   NO2        1790 non-null   float64 
 7   Max         1793 non-null   float64
```

```
8   Critical 1793 non-null object
9   Kategori 1793 non-null object
dtypes: datetime64[ns](1), float64(6), object(3)
memory usage: 154.1+ KB
```

Catatan/Penjelasan dari Output di atas:

Output yang diberikan adalah hasil dari pemanggilan metode `info()` pada objek DataFrame (sepertinya) yang disimpan dalam variabel `dataudara`. Berikut adalah catatan dan penjelasan dari output tersebut:

1. Struktur DataFrame:

- DataFrame tersebut memiliki 1793 baris (entri) dan 10 kolom.
- Indeks barisnya adalah bilangan bulat (`Int64Index`) dan berkisar dari 0 hingga 1829.

2. Nama Kolom dan Tipe Data:

- `Tanggal`: Tipe data kolom ini adalah `datetime64[ns]`, yang menunjukkan bahwa kolom ini berisi data tanggal dan waktu.
- `Wilayah`: Tipe data kolom ini adalah `object`, yang mungkin menunjukkan bahwa kolom ini berisi data teks atau string.
- `PM10`, `S02`, `CO`, `O3`, `N02`, `Max`: Kelima kolom ini memiliki tipe data `float64`, yang menunjukkan bahwa kolom-kolom tersebut berisi nilai desimal.
- `Critical` dan `Kategori`: Kedua kolom ini memiliki tipe data `object`, yang mungkin menunjukkan bahwa kolom-kolom tersebut berisi data teks atau string.

3. Jumlah Data Non-Null:

- Setiap kolom memiliki jumlah data non-null yang kurang lebih sama dengan jumlah baris total (1793), yang menunjukkan bahwa tidak ada nilai null yang mencolok pada pandangan pertama.

4. Informasi Tambahan:

- Total penggunaan memori untuk DataFrame ini adalah sekitar 154.1 KB.
- Ada satu nilai null pada kolom `CO`.
- Ada beberapa kolom yang mungkin berkaitan dengan kualitas udara, seperti `PM10`, `S02`, `CO`, `O3`, dan `N02`.

Catatan: Analisis yang lebih mendalam tentang data ini mungkin diperlukan untuk memahami konteks dan signifikansi kolom serta nilai-nilai spesifik dalam DataFrame.

▼ Data Transformation

Prompt chatGPT:

Tuliskan kode untuk menghitung frekuensi relatif dari kolom 'Critical' dalam dataframe 'dataudara' dan mengkodekan kolom tersebut berdasarkan frekuensi relatif yang telah dihitung. Tampilkan dataframe 'dataudara' yang telah diubah dengan menambahkan kolom baru 'Critical Encoded'.

Penjelasan Kode:

1. Menghitung Frekuensi Relatif:

```
frequency_map = dataudara['Critical'].value_counts(normalize=True).to_dict()
```

Dalam baris ini, kita menggunakan metode `value_counts()` pada kolom 'Critical' dari dataframe 'dataudara' untuk menghitung frekuensi relatif dari setiap nilai dalam kolom tersebut. Parameter `normalize=True` digunakan untuk mendapatkan proporsi relatif daripada jumlah absolut. Hasilnya kemudian dikonversi menjadi dictionary menggunakan `to_dict()` dan disimpan dalam variabel `frequency_map`.

2. Mengkodekan Kolom 'Critical':

```
dataudara['Critical Encoded'] = dataudara['Critical'].map(frequency_map)
```

Pada baris ini, kita menggunakan metode `map()` untuk menggantikan setiap nilai dalam kolom 'Critical' dengan nilai frekuensi relatif yang sesuai dari dictionary `frequency_map`. Hasilnya disimpan dalam kolom baru bernama 'Critical Encoded' di dalam dataframe 'dataudara'.

3. Menampilkan DataFrame yang Telah Diubah:

```
dataudara
```

Baris ini hanya bertujuan untuk menampilkan dataframe 'dataudara' yang telah diubah dengan menambahkan kolom baru 'Critical Encoded', sehingga kita dapat melihat hasil transformasi tersebut.

```

1 # Menghitung frekuensi relatif dari kolom 'Critical' dalam dataframe 'dataudara'
2 frequency_map = dataudara['Critical'].value_counts(normalize=True).to_dict()
3
4 # Mengkodekan kolom 'Critical' berdasarkan frekuensi relatif yang telah dihitung
5 dataudara['Critical Encoded'] = dataudara['Critical'].map(frequency_map)
6
7 # Menampilkan dataframe 'dataudara' yang telah diubah dengan kolom baru 'Critical Encoded'
8 dataudara
9

```

	Tanggal	Wilayah	PM10	SO2	CO	O3	NO2	Max	Critical	Kategori	Critical Encoded
0	2016-01-01	DKI1 (Bunderan HI)	59.0	19.0	27.0	31.0	1.0	59.0	PM10	SEDANG	0.445622
1	2016-01-02	DKI1 (Bunderan HI)	52.0	19.0	40.0	33.0	2.0	52.0	PM10	SEDANG	0.445622
2	2016-01-03	DKI1 (Bunderan HI)	44.0	19.0	40.0	36.0	2.0	44.0	PM10	BAIK	0.445622
3	2016-01-04	DKI1 (Bunderan HI)	58.0	21.0	40.0	46.0	5.0	58.0	PM10	SEDANG	0.445622
4	2016-01-05	DKI1 (Bunderan HI)	70.0	19.0	40.0	41.0	4.0	70.0	PM10	SEDANG	0.445622
...
1825	2016-12-27	DKI5 (Kebon Jeruk) Jakarta Barat	48.0	17.0	34.0	54.0	3.0	54.0	O3	SEDANG	0.504183
1826	2016-12-28	DKI5 (Kebon Jeruk) Jakarta Barat	11.0	18.0	30.0	20.0	2.0	30.0	CO	BAIK	0.032348
1827	2016-12-29	DKI5 (Kebon Jeruk) Jakarta Barat	11.0	17.0	33.0	28.0	4.0	33.0	CO	BAIK	0.032348
1828	2016-12-30	DKI5 (Kebon Jeruk) Jakarta Barat	8.0	17.0	40.0	31.0	4.0	40.0	CO	BAIK	0.032348
1829	2016-12-31	DKI5 (Kebon Jeruk) Jakarta Barat	24.0	18.0	44.0	44.0	4.0	44.0	CO	BAIK	0.032348

1793 rows × 11 columns

Catatan/Penjelasan dari Output di atas:

Dari output di atas, kita dapat melihat dataframe yang telah mengalami transformasi dengan penambahan kolom baru, yaitu 'Critical Encoded'.

Berikut adalah catatan/penjelasan dari beberapa kolom:

1. **Tanggal:** Menunjukkan tanggal pengukuran polusi udara.

2. **Wilayah:** Menunjukkan lokasi pengukuran polusi udara.
3. **PM10, SO2, CO, O3, NO2, Max:** Kadar polutan udara seperti Particulate Matter (PM10), Sulfur Dioxide (SO2), Carbon Monoxide (CO), Ozone (O3), Nitrogen Dioxide (NO2), dan nilai maksimum dari polutan tersebut.
4. **Critical:** Kategori kritis yang diberikan berdasarkan jenis polutan tertentu.
5. **Kategori:** Menunjukkan kategori tingkat pencemaran udara yang dihasilkan dari konsentrasi polutan.
6. **Critical Encoded:** Kolom baru yang dibuat berdasarkan pengkodean frekuensi relatif dari kolom 'Critical'. Nilai ini memberikan representasi numerik untuk kategori kritis yang semula bersifat kualitatif. Misalnya, pada baris pertama, nilai 'Critical Encoded' untuk kategori 'PM10' adalah 0.445622.

Prompt chatGPT:

Tolong berikan perintah prompt untuk menuliskan kode yang mencakup kalimat "Check the column names in your DataFrame" dan "print(dataudara.columns)".

Penjelasan Kode:

1. `# Check the column names in your DataFrame`: Ini adalah komentar dalam kode Python yang digunakan untuk memberikan penjelasan atau catatan. Baris ini memberikan informasi bahwa kita akan memeriksa nama kolom dalam DataFrame.
2. `print(dataudara.columns)`: Baris ini mencetak nama-nama kolom dari DataFrame yang disimpan dalam variabel `dataudara`. Fungsi `print()` digunakan untuk menampilkan informasi ke konsol, dan `dataudara.columns` digunakan untuk mengakses atribut kolom DataFrame, yang memberikan daftar nama kolom.

```
1 # Check the column names in your DataFrame
2 print(dataudara.columns)
3

Index(['Tanggal', 'Wilayah', 'PM10', 'SO2', 'CO', 'O3', 'NO2', 'Max',
       'Critical', 'Kategori', 'Critical Encoded'],
      dtype='object')
```

Catatan/Penjelasan dari Output di atas:

Output di atas adalah hasil dari pemanggilan atribut `columns` pada DataFrame `dataudara`. Berikut adalah penjelasan untuk setiap nama kolom:

1. `Tanggal`: Mendefinisikan kolom yang berisi informasi tentang tanggal pengukuran data udara.
2. `Wilayah`: Merupakan kolom yang berisi informasi tentang wilayah atau lokasi di mana pengukuran dilakukan.
3. `PM10`: Kolom yang berisi nilai partikulat matter dengan diameter kurang dari 10 mikrometer.
4. `SO2`: Mengacu pada kolom yang berisi nilai konsentrasi sulfur dioksida dalam udara.
5. `CO`: Kolom yang berisi nilai konsentrasi karbon monoksida dalam udara.
6. `O3`: Merupakan kolom yang berisi nilai konsentrasi ozon dalam udara.
7. `NO2`: Kolom yang berisi nilai konsentrasi nitrogen dioksida dalam udara.
8. `Max`: Kolom ini mungkin menunjukkan nilai maksimum dari parameter tertentu dalam satu periode pengukuran.
9. `Critical`: Menunjukkan nilai kritis tertentu yang mungkin memiliki dampak buruk terhadap kualitas udara.
10. `Kategori`: Kolom yang mungkin berisi kategori atau level kualitas udara berdasarkan pengukuran parameter tertentu.
11. `Critical_Encoded`: Kemungkinan kolom yang berisi nilai yang dikenkoding terkait dengan nilai kritis pada kolom "Critical". Encoding dapat digunakan untuk representasi numerik dari kategori.

Tentu saja, penjelasan ini bersifat umum dan tergantung pada konteks spesifik dataset dan tujuan analisisnya.

Prompt chatGPT:

Tulis perintah Python untuk mengimpor kelas LabelEncoder dari pustaka `sklearn.preprocessing`, membuat objek LabelEncoder bernama 'le', menggunakan LabelEncoder untuk mengkodekan kategori pada kolom 'Kategori' dari dataframe 'dataudara', dan menyimpan hasilnya dalam kolom baru 'CategoryEncoded'. Terakhir, tampilkan dataframe 'dataudara' setelah proses pengkodean.

Penjelasan Kode:

1. **Mengimport Kelas LabelEncoder dari pustaka `sklearn.preprocessing`:**

```
from sklearn.preprocessing import LabelEncoder
```

Pada baris ini, kita mengimpor kelas LabelEncoder dari pustaka `sklearn.preprocessing`. Kelas ini digunakan untuk mengkodekan label kategori menjadi nilai numerik.

2. Membuat Objek LabelEncoder:

```
le = LabelEncoder()
```

Baris ini membuat objek LabelEncoder baru yang disimpan dalam variabel `le`. Objek ini akan digunakan untuk melakukan pengkodean kategori.

3. Menggunakan LabelEncoder untuk Mengkodekan Kategori pada Kolom 'Kategori':

```
dataudara[ 'CategoryEncoded' ] = le.fit_transform(dataudara[ 'Kategori' ])
```

Pada baris ini, LabelEncoder digunakan untuk mengkodekan nilai pada kolom 'Kategori' dari dataframe `dataudara`. Hasil pengkodean disimpan dalam kolom baru bernama 'CategoryEncoded' pada dataframe yang sama.

4. Menampilkan DataFrame 'dataudara' Setelah Proses Pengkodean:

```
dataudara
```

Baris ini bertujuan untuk menampilkan dataframe `dataudara` setelah proses pengkodean. Ini memungkinkan kita melihat hasil pengkodean kategori yang telah disimpan dalam kolom 'CategoryEncoded'.

```

1 # Mengimport kelas LabelEncoder dari pustaka sklearn.preprocessing
2 from sklearn.preprocessing import LabelEncoder
3
4 # Membuat objek LabelEncoder
5 le = LabelEncoder()
6
7 # Menggunakan LabelEncoder untuk mengkodekan kategori pada kolom 'Kategori' dan menyimpannya dalam kolom baru 'CategoryEncoded'
8 dataudara['CategoryEncoded'] = le.fit_transform(dataudara['Kategori'])
9
10 # Menampilkan dataframe 'dataudara' setelah proses pengkodean
11 dataudara
12

```

	Tanggal	Wilayah	PM10	SO2	CO	O3	NO2	Max	Critical	Kategori	Critical Encoded	CategoryEncoded
0	2016-01-01	DKI1 (Bunderan HI)	59.0	19.0	27.0	31.0	1.0	59.0	PM10	SEDANG	0.445622	2
1	2016-01-02	DKI1 (Bunderan HI)	52.0	19.0	40.0	33.0	2.0	52.0	PM10	SEDANG	0.445622	2
2	2016-01-03	DKI1 (Bunderan HI)	44.0	19.0	40.0	36.0	2.0	44.0	PM10	BAIK	0.445622	0
3	2016-01-04	DKI1 (Bunderan HI)	58.0	21.0	40.0	46.0	5.0	58.0	PM10	SEDANG	0.445622	2
4	2016-01-05	DKI1 (Bunderan HI)	70.0	19.0	40.0	41.0	4.0	70.0	PM10	SEDANG	0.445622	2
...
1825	2016-12-27	DKI5 (Kebon Jeruk) Jakarta Barat	48.0	17.0	34.0	54.0	3.0	54.0	O3	SEDANG	0.504183	2
1826	2016-12-28	DKI5 (Kebon Jeruk) Jakarta Barat	11.0	18.0	30.0	20.0	2.0	30.0	CO	BAIK	0.032348	0
1827	2016-12-29	DKI5 (Kebon Jeruk) Jakarta Barat	11.0	17.0	33.0	28.0	4.0	33.0	CO	BAIK	0.032348	0
1828	2016-12-30	DKI5 (Kebon Jeruk) Jakarta Barat	8.0	17.0	40.0	31.0	4.0	40.0	CO	BAIK	0.032348	0
1829	2016-12-31	DKI5 (Kebon Jeruk) Jakarta Barat	24.0	18.0	44.0	44.0	4.0	44.0	CO	BAIK	0.032348	0

1793 rows × 12 columns

Catatan/Penjelasan dari Output di atas:

- Tanggal:** Data pada kolom ini menyajikan tanggal pengukuran kualitas udara.
- Wilayah:** Menunjukkan wilayah tempat pengukuran dilakukan, misalnya "DKI1 (Bunderan HI)" atau "DKI5 (Kebon Jeruk) Jakarta Barat".

3. **PM10, SO2, CO, O3, NO2:** Kolom-kolom ini berisi tingkat konsentrasi berbagai jenis polutan udara (partikulat, sulfur dioksida, karbon monoksida, ozon, nitrogen dioksida) yang diukur dalam udara.
4. **Max:** Menunjukkan nilai tertinggi di antara konsentrasi PM10, SO2, CO, O3, dan NO2 pada suatu pengukuran.
5. **Critical:** Indikator kategori kualitas udara berdasarkan nilai Max. Misalnya, "PM10" atau "CO".
6. **Kategori:** Menyajikan tingkat kualitas udara berdasarkan kriteria tertentu, seperti "BAIK" atau "SEDANG".
7. **Critical Encoded:** Kategori kualitas udara yang telah diubah menjadi nilai numerik.
8. **CategoryEncoded:** Kategori kualitas udara pada kolom 'Kategori' yang telah diubah menjadi nilai numerik menggunakan LabelEncoder. Nilai ini tampaknya diberikan sebagai hasil dari pengkodean kategori untuk kolom 'Kategori'. Misalnya, "BAIK" dapat diwakili sebagai 0, "SEDANG" sebagai 2, dan sebagainya.

Prompt chatGPT:

"Buatkan kode Python yang memeriksa apakah kolom 'CategoryEncoded' ada dalam DataFrame 'dataudara'. Jika ada, konversi tipe data kolom tersebut menjadi float64; jika tidak, tampilkan pesan bahwa kolom tersebut tidak ditemukan."

Penjelasan Kode:

1. Baris 1: `if 'CategoryEncoded' in dataudara.columns:`
Perintah ini memeriksa apakah kolom dengan nama 'CategoryEncoded' ada dalam DataFrame 'dataudara'. `dataudara.columns` mengembalikan daftar nama-nama kolom dalam DataFrame, dan `'CategoryEncoded' in dataudara.columns` mengevaluasi ke `True` jika kolom tersebut ada.
2. Baris 3-4:
Jika kolom 'CategoryEncoded' ditemukan (evaluasi di baris 1 adalah `True`), baris ini akan dieksekusi. Di dalam blok ini, tipe data kolom 'CategoryEncoded' diubah menjadi float64 menggunakan `.astype('float64')`. Ini berguna jika kolom tersebut sebelumnya memiliki tipe data yang berbeda dan kita ingin mengonversinya menjadi tipe data float64.
3. Baris 5: `print("Kolom 'CategoryEncoded' ditemukan dalam DataFrame.")`
Ini adalah pernyataan cetak yang akan dieksekusi jika kolom 'CategoryEncoded' ditemukan. Pesan ini memberi tahu pengguna bahwa kolom tersebut ada dalam DataFrame.

4. Baris 6-7:

Jika evaluasi pada baris 1 adalah `False` (kolom 'CategoryEncoded' tidak ditemukan), maka blok `else` ini akan dieksekusi. Pada baris 6, dicetak pesan yang memberi tahu pengguna bahwa kolom 'CategoryEncoded' tidak ditemukan dalam DataFrame.

```
1 # Periksa apakah 'CategoryEncoded' ada di kolom-kolom
2 if 'CategoryEncoded' in dataudara.columns:
3     # Lakukan konversi tipe data
4     dataudara['CategoryEncoded'] = dataudara['CategoryEncoded'].astype('float64')
5     print("Kolom 'CategoryEncoded' ditemukan dalam DataFrame.")
6 else:
7     print("Kolom 'CategoryEncoded' tidak ditemukan dalam DataFrame.")
8
9
```

Kolom 'CategoryEncoded' ditemukan dalam DataFrame.

Catatan/Penjelasan dari Output di atas:

Output tersebut menunjukkan bahwa kolom dengan nama 'CategoryEncoded' ditemukan dalam DataFrame. Pesan "Kolom 'CategoryEncoded' ditemukan dalam DataFrame." dicetak sebagai respons dari pernyataan `print` pada baris 5 dalam kode. Ini menandakan bahwa proses pemeriksaan pada baris 1 menghasilkan nilai `True`, yang berarti kolom 'CategoryEncoded' ada dalam DataFrame 'dataudara'. Selanjutnya, program melanjutkan untuk mengonversi tipe data kolom tersebut menjadi float64, seperti yang ditunjukkan pada baris 3-4.

Prompt chatGPT:

Berikan perintah dalam Python untuk menampilkan informasi terkini DataFrame, termasuk tipe data dan jumlah entri, dengan menggunakan fungsi `info()` pada DataFrame yang disebut `dataudara`.

Penjelasan Kode:

Berikan perintah dalam Python untuk menampilkan informasi terkini DataFrame, termasuk tipe data dan jumlah entri, dengan menggunakan fungsi `info()` pada DataFrame yang disebut `dataudara`.

```
1 # Display the updated DataFrame information
2 print(dataudara.info())
3
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1793 entries, 0 to 1829
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   Tanggal          1793 non-null    datetime64[ns]
 1   Wilayah          1793 non-null    object  
 2   PM10             1793 non-null    float64 
 3   SO2              1793 non-null    float64 
 4   CO               1792 non-null    float64 
 5   O3               1790 non-null    float64 
 6   NO2              1790 non-null    float64 
 7   Max              1793 non-null    float64 
 8   Critical          1793 non-null    object  
 9   Kategori          1793 non-null    object  
 10  Critical Encoded  1793 non-null    float64 
 11  CategoryEncoded   1793 non-null    float64 
dtypes: datetime64[ns](1), float64(8), object(3)
memory usage: 182.1+ KB
None
```

Catatan/Penjelasan dari Output di atas:

Output tersebut adalah hasil dari fungsi `info()` pada DataFrame yang disebut `dataudara`. Berikut adalah penjelasan dari setiap bagian output:

1. **Header:** Menunjukkan bahwa DataFrame memiliki 1793 entri dengan indeks bertipe `Int64Index` dari 0 hingga 1829.
2. **Data columns:** Memberikan informasi tentang jumlah kolom (12 kolom) dalam DataFrame dan atribut-atribut kolom tersebut:
 - **Column:** Nama kolom.
 - **Non-Null Count:** Jumlah nilai non-null (tidak bernilai null) dalam kolom.
 - **Dtype:** Tipe data dari nilai dalam kolom.
3. **Info pada setiap kolom:**
 - **Tanggal:** Memiliki 1793 non-null nilai dengan tipe data `datetime64`.
 - **Wilayah:** Memiliki 1793 non-null nilai dengan tipe data `object` (teks).
 - **PM10, SO2, CO, O3, NO2, Max, Critical, Kategori, Critical Encoded, CategoryEncoded:** Memiliki 1793 non-null nilai dengan tipe data `float64` atau `object`, tergantung pada kolomnya.
4. **memory usage:** Menunjukkan penggunaan memori DataFrame, dalam hal ini sekitar 182.1+ KB.

Output ini memberikan informasi penting tentang struktur DataFrame, tipe data setiap kolom, dan jumlah nilai non-null dalam setiap kolom, yang dapat membantu dalam pemahaman data dan penanganan nilai null (jika ada).

Prompt chatGPT:

Berikan contoh kode menggunakan Pandas untuk mendapatkan korelasi antar kolom-kolom dalam DataFrame "dataudara" menggunakan fungsi corr().

Penjelasan Kode:

```
# Menggunakan Pandas untuk menghitung korelasi antar kolom-kolom dalam DataFrame "dataudara".  
# Fungsi corr() digunakan untuk menghasilkan matriks korelasi antar kolom.  
dataudara.corr()
```

Penjelasan baris per baris:

1. # Menggunakan Pandas untuk menghitung korelasi antar kolom-kolom dalam DataFrame "dataudara". : Komentar untuk menjelaskan bahwa kita akan menggunakan Pandas untuk menghitung korelasi antar kolom-kolom dalam DataFrame yang disebut "dataudara".
2. # Fungsi corr() digunakan untuk menghasilkan matriks korelasi antar kolom. : Komentar untuk menjelaskan bahwa kita menggunakan fungsi corr() dari Pandas, yang menghasilkan matriks korelasi antar kolom dalam DataFrame. Ini memberikan gambaran tentang seberapa kuat hubungan antar kolom-kolom tersebut; nilai korelasi berkisar antara -1 hingga 1. Nilai positif menunjukkan korelasi positif, nilai negatif menunjukkan korelasi negatif, dan nilai 0 menunjukkan tidak adanya korelasi.

```
1 # Jika menggunakan Pandas dan memiliki DataFrame bernama dataudara, berikut adalah contoh penggunaan fungsi corr() untuk mendapatkan korelasi ant  
2  
3 dataudara.corr()  
4
```

```
<ipython-input-91-c60163768185>:3: FutureWarning:
```

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or

	PM10	SO2	CO	O3	NO2	Max	Critical	Encoded	CategoryEncoded
PM10	1.000000	0.144936	0.191395	0.202042	0.281972	0.462001	0.152448	0.540241	
SO2	0.144936	1.000000	-0.333884	-0.220688	-0.302556	-0.067987	-0.024420	-0.013739	
CO	0.191395	-0.333884	1.000000	0.153370	0.147185	0.216699	-0.141599	0.206184	
O3	0.202042	-0.220688	0.153370	1.000000	0.390923	0.869898	0.361654	0.584893	
NO2	0.281972	-0.302556	0.147185	0.390923	1.000000	0.332694	0.150217	0.271917	

Catatan/Penjelasan dari Output di atas:

Output di atas adalah matriks korelasi antar kolom-kolom dalam DataFrame. Setiap angka dalam matriks menunjukkan nilai korelasi antara dua kolom yang sesuai. Berikut adalah catatan/penjelasan dari matriks korelasi tersebut:

- **PM10 dan Kolom Lainnya:**
 - PM10 memiliki korelasi positif yang cukup kuat dengan Max (0.462001) dan CategoryEncoded (0.540241). Ini menunjukkan adanya hubungan positif antara PM10 dengan nilai Max dan CategoryEncoded.
 - Korelasi positif yang moderat terdapat antara PM10 dan O3 (0.202042), NO2 (0.281972), dan CO (0.191395).
 - Korelasi positif yang lemah dengan Critical Encoded (0.152448) dan SO2 (0.144936).
- **SO2 dan Kolom Lainnya:**
 - SO2 memiliki korelasi negatif yang cukup kuat dengan CO (-0.333884) dan NO2 (-0.302556), menunjukkan adanya hubungan negatif antara SO2 dengan nilai CO dan NO2.
 - Korelasi negatif yang lemah terdapat dengan Max (-0.067987) dan Critical Encoded (-0.024420).
- **CO dan Kolom Lainnya:**
 - CO memiliki korelasi positif yang lemah dengan O3 (0.153370) dan CategoryEncoded (0.206184).
 - Korelasi negatif yang cukup kuat dengan SO2 (-0.333884), menunjukkan adanya hubungan negatif antara CO dan SO2.
- **O3 dan Kolom Lainnya:**
 - O3 memiliki korelasi positif yang moderat dengan PM10 (0.202042), NO2 (0.281972), dan CategoryEncoded (0.584893).
 - Korelasi negatif yang lemah terdapat dengan SO2 (-0.220688).
- **NO2 dan Kolom Lainnya:**
 - NO2 memiliki korelasi negatif yang cukup kuat dengan SO2 (-0.302556) dan CO (0.147185).
 - Korelasi positif yang lemah terdapat dengan PM10 (0.281972) dan CategoryEncoded (0.271917).

- O3 memiliki korelasi positif yang cukup kuat dengan Max (0.869898) dan CategoryEncoded (0.584893), menunjukkan adanya hubungan positif antara O3 dengan nilai Max dan CategoryEncoded.
- Korelasi positif yang moderat terdapat dengan NO2 (0.390923) dan PM10 (0.202042).

- **NO2 dan Kolom Lainnya:**

- NO2 memiliki korelasi positif yang cukup kuat dengan O3 (0.390923) dan Max (0.332694), menunjukkan adanya hubungan positif antara NO2 dengan nilai O3 dan Max.
- Korelasi positif yang moderat terdapat dengan PM10 (0.281972) dan CO (0.147185).

- **Max dan Kolom Lainnya:**

- Max memiliki korelasi positif yang sangat kuat dengan O3 (0.869898) dan CategoryEncoded (0.751145), menunjukkan adanya hubungan positif yang sangat kuat antara Max dengan nilai O3 dan CategoryEncoded.
- Korelasi positif yang cukup kuat terdapat dengan PM10 (0.462001) dan NO2 (0.332694).

- **Critical Encoded dan Kolom Lainnya:**

- Korelasi positif yang cukup kuat dengan CategoryEncoded (0.321413), menunjukkan adanya hubungan positif antara Critical Encoded dengan CategoryEncoded.
- Korelasi positif yang moderat terdapat dengan O3 (0.361654) dan Max (0.345071).

- **CategoryEncoded dan Kolom Lainnya:**

- CategoryEncoded memiliki korelasi positif yang sangat kuat dengan Max (0.751145) dan PM10 (0.540241), menunjukkan adanya hubungan positif yang sangat kuat antara CategoryEncoded dengan nilai Max dan PM10.
- Korelasi positif yang cukup kuat terdapat dengan O3 (0.584893) dan NO2 (0.271917).

Prompt chatGPT:

Berikan kode yang diperlukan untuk menampilkan isi dari variabel 'dataudara'.

Penjelasan Kode:

```
# Menampilkan isi datanya
print(dataudara)
```

1. `# Menampilkan isi datanya`: Ini adalah komentar di Python yang digunakan untuk memberikan penjelasan singkat tentang tujuan atau fungsi dari baris kode berikutnya. Komentar tidak memengaruhi jalannya program dan hanya berfungsi sebagai panduan untuk pembaca.
2. `print(dataudara)`: Ini adalah perintah cetak pada Python yang digunakan untuk menampilkan nilai dari suatu variabel ke konsol atau output program. Pada kasus ini, variabel yang ingin ditampilkan adalah 'dataudara'. `print()` adalah fungsi bawaan Python yang mencetak nilai atau teks ke layar.

```
1 # Menampilkan isi datanya
2 dataudara
```

	Tanggal	Wilayah	PM10	SO2	CO	O3	NO2	Max	Critical	Kategori	Critical Encoded	CategoryEncoded
0	2016-01-01	DKI1 (Bunderan HI)	59.0	19.0	27.0	31.0	1.0	59.0	PM10	SEDANG	0.445622	2.0
1	2016-01-02	DKI1 (Bunderan HI)	52.0	19.0	40.0	33.0	2.0	52.0	PM10	SEDANG	0.445622	2.0
2	2016-01-03	DKI1 (Bunderan HI)	44.0	19.0	40.0	36.0	2.0	44.0	PM10	BAIK	0.445622	0.0
3	2016-01-04	DKI1 (Bunderan HI)	58.0	21.0	40.0	46.0	5.0	58.0	PM10	SEDANG	0.445622	2.0
4	2016-01-05	DKI1 (Bunderan HI)	70.0	19.0	40.0	41.0	4.0	70.0	PM10	SEDANG	0.445622	2.0
...
1825	2016-12-27	DKI5 (Kebon Jeruk) Jakarta Barat	48.0	17.0	34.0	54.0	3.0	54.0	O3	SEDANG	0.504183	2.0
1826	2016-12-28	DKI5 (Kebon Jeruk) Jakarta Barat	11.0	18.0	30.0	20.0	2.0	30.0	CO	BAIK	0.032348	0.0
1827	2016-12-29	DKI5 (Kebon Jeruk) Jakarta Barat	11.0	17.0	33.0	28.0	4.0	33.0	CO	BAIK	0.032348	0.0
1828	2016-12-30	DKI5 (Kebon Jeruk) Jakarta Barat	8.0	17.0	40.0	31.0	4.0	40.0	CO	BAIK	0.032348	0.0
1829	2016-12-31	DKI5 (Kebon Jeruk) Jakarta Barat	24.0	18.0	44.0	44.0	4.0	44.0	CO	BAIK	0.032348	0.0

1793 rows × 12 columns

Catatan/Penjelasan dari Output di atas:

Output di atas adalah tampilan dari suatu data tabular yang mungkin berasal dari sebuah DataFrame di dalam lingkungan pemrograman Python, seperti pandas. Berikut adalah catatan/penjelasan dari output tersebut:

1. Header/Data Columns:

- `Tanggal` : Menunjukkan tanggal pengukuran.
- `Wilayah` : Merupakan lokasi atau wilayah pengukuran.
- `PM10, SO2, CO, O3, NO2` : Merupakan nilai tingkat polusi udara untuk setiap jenis polutan pada saat pengukuran.
- `Max` : Menunjukkan nilai maksimum dari kelima jenis polutan tersebut.
- `Critical` : Menunjukkan jenis polutan yang memiliki nilai kritis tertinggi.
- `Kategori` : Kategori tingkat polusi berdasarkan standar tertentu, seperti "BAIK", "SEDANG", dll.
- `Critical Encoded` : Representasi encoded dari nilai kritis.
- `Category Encoded` : Representasi encoded dari kategori tingkat polusi.

2. Data Rows:

- Setiap baris mewakili satu entri data pada suatu tanggal dan wilayah tertentu.
- Nilai-nilai pada kolom-kolom seperti `PM10, SO2`, dst., menunjukkan tingkat konsentrasi polutan pada saat pengukuran.

3. Pembatasan Waktu:

- Data ini tampaknya berisi pengukuran polusi udara untuk setiap hari selama satu tahun (2016), berdasarkan kolom `Tanggal`.

4. Informasi Geografis:

- Kolom `Wilayah` memberikan informasi tentang lokasi atau wilayah di mana pengukuran dilakukan, seperti "DKI1 (Bunderan HI) Jakarta Pusat" atau "DKI5 (Kebon Jeruk) Jakarta Barat".

5. Kategori dan Encoding:

- Terdapat kolom `Kategori` dan `Critical` yang memberikan informasi tentang kategori dan nilai kritis tingkat polusi pada suatu waktu tertentu.
- Kolom `Critical Encoded` dan `Category Encoded` tampaknya adalah representasi encoded dari informasi kritis dan kategori tersebut, mungkin untuk keperluan analisis atau pemrosesan lebih lanjut.

Prompt chatGPT:

Hapus kolom 'Tanggal', 'Wilayah', 'Critical', dan 'Kategori' dari DataFrame 'dataudara' dan tampilkan DataFrame yang telah diperbarui.

Penjelasan Kode:

1. `columns_to_drop = ['Tanggal', 'Wilayah', 'Critical', 'Kategori']`: Pada baris ini, sebuah daftar kolom yang akan dihapus dari DataFrame disiapkan. Kolom-kolom yang dimaksud adalah 'Tanggal', 'Wilayah', 'Critical', dan 'Kategori'.
2. `dataudara = dataudara.drop(columns=columns_to_drop)`: Pada baris ini, metode `.drop()` dari pandas digunakan untuk menghapus kolom-kolom yang telah ditentukan dalam variabel `columns_to_drop` dari DataFrame 'dataudara'. DataFrame yang telah diperbarui kemudian disimpan kembali ke dalam variabel 'dataudara'.
3. `print(dataudara.head())`: Pada baris ini, perintah `print()` digunakan untuk menampilkan lima baris pertama dari DataFrame yang telah diperbarui ('dataudara') menggunakan metode `.head()`. Ini memberikan gambaran singkat tentang struktur dan isi DataFrame setelah penghapusan kolom.

```

1 # Assuming 'dataudara' is your DataFrame
2 columns_to_drop = ['Tanggal', 'Wilayah', 'Critical', 'Kategori']
3 dataudara = dataudara.drop(columns=columns_to_drop)
4
5 # Display the updated DataFrame
6 print(dataudara.head())
7

```

	PM10	SO2	CO	O3	N02	Max	Critical	Encoded	CategoryEncoded
0	59.0	19.0	27.0	31.0	1.0	59.0	0.445622	2.0	
1	52.0	19.0	40.0	33.0	2.0	52.0	0.445622	2.0	
2	44.0	19.0	40.0	36.0	2.0	44.0	0.445622	0.0	
3	58.0	21.0	40.0	46.0	5.0	58.0	0.445622	2.0	
4	70.0	19.0	40.0	41.0	4.0	70.0	0.445622	2.0	

Catatan/Penjelasan dari Output di atas:

Output di atas adalah representasi dari lima baris pertama dari DataFrame yang mungkin berasal dari data kualitas udara. Berikut adalah catatan/penjelasan dari setiap kolom:

1. **PM10**: Kandungan Partikulat Materi dengan diameter kurang dari 10 mikrometer (μm) dalam udara pada satuan mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
2. **SO2**: Kandungan Dioxide belerang (SO₂) dalam udara pada satuan mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
3. **CO**: Kandungan Carbon Monoxide (CO) dalam udara pada satuan mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
4. **O3**: Kandungan Ozon (O₃) dalam udara pada satuan mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).

5. **NO2**: Kandungan Nitrogen Dioxide (NO₂) dalam udara pada satuan mikrogram per meter kubik ($\mu\text{g}/\text{m}^3$).
6. **Max**: Nilai maksimum dari parameter kualitas udara pada suatu waktu tertentu.
7. **Critical Encoded**: Sebuah nilai numerik yang mungkin telah dihasilkan dari suatu metode encoding atau proses transformasi untuk mengkodekan nilai kritis atau batasan tertentu dalam parameter kualitas udara.
8. **CategoryEncoded**: Nilai numerik yang mungkin telah dihasilkan dari suatu metode encoding untuk mewakili kategori tertentu yang terkait dengan kualitas udara.

Setiap baris mewakili pengukuran kualitas udara pada waktu tertentu, dengan nilai numerik yang mencerminkan konsentrasi atau tingkat berbagai polutan udara yang diukur.

Prompt chatGPT:

Tuliskan perintah untuk menampilkan informasi terkini dari DataFrame 'dataudara' di Python.

Penjelasan Kode:

1. `# Display the updated DataFrame information`: Ini adalah komentar dalam kode Python. Komentar tidak memengaruhi jalannya program, tetapi digunakan untuk memberikan penjelasan atau dokumentasi kepada pembaca kode.
2. `dataudara.info()`: Ini adalah pemanggilan metode `info()` pada objek DataFrame yang disebut `dataudara`. Metode ini digunakan untuk menampilkan informasi tentang DataFrame, termasuk jumlah baris dan kolom, tipe data setiap kolom, dan jumlah nilai yang tidak null dalam setiap kolom. Ini berguna untuk menganalisis struktur data dan mendeteksi nilai yang hilang.

Penjelasan singkat ini mencakup maksud dan fungsi setiap baris kode dalam potongan yang Anda berikan. Jika Anda memiliki pertanyaan lebih lanjut tentang fungsi atau konsep tertentu, jangan ragu untuk bertanya!

```
1 # Display the updated DataFrame information
2 dataudara.info()

<class 'pandas.core.frame.DataFrame'\>
Int64Index: 1793 entries, 0 to 1829
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   PM10            1793 non-null    float64
```

```
1  SO2          1793 non-null  float64
2  CO           1792 non-null  float64
3  O3           1790 non-null  float64
4  NO2          1790 non-null  float64
5  Max          1793 non-null  float64
6  Critical Encoded  1793 non-null  float64
7  CategoryEncoded  1793 non-null  float64
dtypes: float64(8)
memory usage: 126.1 KB
```

Catatan/Penjelasan dari Output di atas:

Output di atas adalah informasi tentang DataFrame yang ditampilkan oleh metode `info()` dari Pandas. Berikut adalah penjelasan untuk setiap bagian:

1. `<class 'pandas.core.frame.DataFrame'>`: Ini menunjukkan bahwa objek tersebut adalah instansi dari kelas DataFrame dalam pustaka Pandas.
2. `Int64Index: 1793 entries, 0 to 1829`: Menyediakan informasi tentang indeks DataFrame. `Int64Index` adalah tipe indeks, dan terdapat 1793 entri dari indeks 0 hingga 1829.
3. `Data columns (total 8 columns)`: Menunjukkan bahwa DataFrame ini memiliki total 8 kolom.
4. `# Column Non-Null Count Dtype`: Membuat header untuk bagian selanjutnya yang menyajikan informasi kolom.
5. `---`: Garis pemisah antara header dan informasi kolom.
6. `0 PM10 1793 non-null float64`: Memberikan informasi tentang kolom pertama, yaitu 'PM10'. Ada 1793 nilai non-null (tidak ada nilai yang hilang) dan tipe data kolom tersebut adalah float64.
7. `1 SO2 1793 non-null float64`: Informasi serupa untuk kolom kedua, 'SO2'.
8. `2 CO 1792 non-null float64`: Informasi untuk kolom ketiga, 'CO'. Terdapat 1792 nilai non-null.
9. `3 O3 1790 non-null float64`: Informasi untuk kolom keempat, 'O3'. Terdapat 1790 nilai non-null.
10. `4 NO2 1790 non-null float64`: Informasi untuk kolom kelima, 'NO2'. Terdapat 1790 nilai non-null.
11. `5 Max 1793 non-null float64`: Informasi untuk kolom keenam, 'Max'. Ada 1793 nilai non-null.
12. `6 Critical Encoded 1793 non-null float64`: Informasi untuk kolom ketujuh, 'Critical Encoded'. Ada 1793 nilai non-null.
13. `7 CategoryEncoded 1793 non-null float64`: Informasi untuk kolom kedelapan, 'CategoryEncoded'. Ada 1793 nilai non-null.

14. `dtypes: float64(8)` : Menyajikan tipe data dari setiap kolom, yaitu float64.

15. `memory usage: 126.1 KB` : Memberikan perkiraan penggunaan memori untuk DataFrame ini, yaitu sekitar 126.1 kilobita.

Prompt chatGPT:

Berikan perintah untuk menuliskan kode dalam Python untuk memisahkan kolom 'CategoryEncoded' dari dataframe 'dataudara' menjadi variabel X dan y, dan menampilkan beberapa baris pertama dari keduanya.

Penjelasan Kode:

1. Menghilangkan kolom 'CategoryEncoded' dari dataframe 'dataudara' dan menyimpannya dalam variabel X:

- `X1 = dataudara.drop(columns=['CategoryEncoded'])` : Dengan menggunakan fungsi `drop`, baris ini menghasilkan variabel `x1` yang merupakan salinan dari dataframe `dataudara` tanpa kolom 'CategoryEncoded'. Ini dilakukan untuk membuat dataset fitur (features) yang akan digunakan dalam pembuatan model.

2. Mengambil kolom 'CategoryEncoded' sebagai target dan menyimpannya dalam variabel y:

- `y1 = dataudara['CategoryEncoded']` : Pada baris ini, kolom 'CategoryEncoded' dari dataframe `dataudara` diambil dan disimpan dalam variabel `y1`. Variabel ini akan digunakan sebagai target atau output yang ingin diprediksi dalam model.

3. Menampilkan beberapa baris pertama dari X dan y untuk melihat data:

- `print("Fitur (X):")` : Baris ini mencetak teks "Fitur (X):" untuk memberikan label pada output yang akan ditampilkan, menunjukkan bahwa yang dicetak setelahnya adalah dataset fitur.
- `print(X1.head())` : Ini mencetak beberapa baris pertama dari dataset fitur (`x1`), memberikan pandangan awal tentang struktur dan isi dari dataset tersebut.
- `print("\nTarget (y):")` : Mencetak teks "Target (y):" sebagai label untuk output selanjutnya, menunjukkan bahwa yang dicetak setelahnya adalah dataset target.
- `print(y1.head())` : Ini mencetak beberapa baris pertama dari dataset target (`y1`), memberikan informasi awal tentang nilai yang ingin diprediksi dalam model.

```
1 # Menghilangkan kolom 'CategoryEncoded' dari dataframe 'dataudara' dan menyimpannya dalam variabel X
2 X1 = dataudara.drop(columns=['CategoryEncoded'])
3
4 # Mengambil kolom 'CategoryEncoded' sebagai target dan menyimpannya dalam variabel y
5 y1 = dataudara['CategoryEncoded']
6
7 # Menampilkan beberapa baris pertama dari X dan y untuk melihat data
8 print("Fitur (X):")
9 print(X1.head())
10
11 print("\nTarget (y):")
12 print(y1.head())
13
```

Fitur (X):

	PM10	S02	CO	O3	NO2	Max	Critical	Encoded
0	59.0	19.0	27.0	31.0	1.0	59.0		0.445622
1	52.0	19.0	40.0	33.0	2.0	52.0		0.445622
2	44.0	19.0	40.0	36.0	2.0	44.0		0.445622
3	58.0	21.0	40.0	46.0	5.0	58.0		0.445622
4	70.0	19.0	40.0	41.0	4.0	70.0		0.445622

Target (y):

0	2.0
1	2.0
2	0.0
3	2.0
4	2.0

Name: CategoryEncoded, dtype: float64

Catatan/Penjelasan dari Output di atas:

Output:

Fitur (X):

- **PM10:** Nilai kolom PM10 untuk setiap baris dalam dataset.
- **S02:** Nilai kolom S02 untuk setiap baris dalam dataset.
- **CO:** Nilai kolom CO untuk setiap baris dalam dataset.
- **O3:** Nilai kolom O3 untuk setiap baris dalam dataset.
- **NO2:** Nilai kolom NO2 untuk setiap baris dalam dataset.

- **Max:** Nilai kolom Max untuk setiap baris dalam dataset.
- **Critical Encoded:** Nilai kolom Critical Encoded untuk setiap baris dalam dataset.

Target (y):

- **CategoryEncoded:** Nilai target yang ingin diprediksi atau dijelaskan oleh model. Nilai ini diambil dari kolom 'CategoryEncoded' dalam dataframe asli 'dataaudara'. Nilai ini mewakili kategori atau kelas yang diatributkan kepada setiap baris data dalam dataset. Dalam hal ini, nilai target dapat berupa angka desimal, seperti 0.0 atau 2.0.

▼ Uji Data Baru

Uji Data Baru atau pengujian pada model Support Vector Machine (SVM) merupakan bagian penting dalam evaluasi kinerja model. Proses ini bertujuan untuk mengukur sejauh mana model yang telah dilatih mampu menggeneralisasi pola yang telah dipelajari dari data latih terhadap data yang belum pernah dilihat sebelumnya.

Berikut adalah penjelasan umum tentang Uji Data Baru pada pengujian model Support Vector Machine (SVM):

1. **Data Uji (Testing Data):** Sebelum melibatkan model SVM dalam uji data baru, perlu memisahkan data menjadi dua bagian utama: data latih (training data) dan data uji (testing data). Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk mengukur kinerja model di luar sampel data yang telah dilihat sebelumnya.
2. **Proses Pelatihan (Training Process):** SVM merupakan algoritma pembelajaran yang membangun sebuah hyperplane untuk memisahkan kelas-kelas dalam data latih. Hyperplane ini ditentukan berdasarkan vektor pendukung (support vectors) yang memiliki jarak terdekat dengan batas keputusan. Proses pelatihan berfokus pada menemukan hyperplane yang optimal.
3. **Pengujian Model dengan Data Uji Baru:**
 - Setelah model SVM dilatih dengan data latih, langkah berikutnya adalah menguji model menggunakan data uji yang belum pernah dilihat sebelumnya.
 - Data uji diberikan kepada model, dan prediksi dibuat untuk setiap sampel dalam data uji.
 - Hasil prediksi ini kemudian dibandingkan dengan label sebenarnya dari data uji untuk mengukur sejauh mana model dapat memprediksi dengan benar.
4. **Metrik Evaluasi Kinerja:**

- Beberapa metrik evaluasi umum yang digunakan untuk mengukur kinerja model SVM pada data uji baru melibatkan perhitungan akurasi, presisi, recall, F1-score, dan matriks konfusi.
- **Akurasi** mengukur sejauh mana model dapat membuat prediksi yang benar secara keseluruhan.
- **Presisi dan recall** memberikan wawasan lebih rinci tentang kualitas prediksi model, terutama dalam konteks kelas-kelas yang tidak seimbang.
- **F1-score** memberikan keseimbangan antara presisi dan recall.

5. **Penyetelan Hyperparameter (Optional)**: Kadang-kadang, dilakukan penyetelan ulang terhadap hyperparameter model SVM untuk meningkatkan kinerja pada data uji. Ini dapat melibatkan menggunakan teknik seperti validasi silang untuk menemukan kombinasi hyperparameter yang optimal.

Dengan melakukan uji data baru, kita dapat menilai sejauh mana model SVM mampu menggeneralisasi pola yang telah dipelajari selama pelatihan dan seberapa baik model tersebut dapat berperforma pada data yang belum pernah dilihat sebelumnya.

Prompt chatGPT:

Gantilah nilai NaN dalam dataset dengan menggunakan mean dari setiap kolomnya menggunakan SimpleImputer dari scikit-learn. Simpan hasil transformasi ke dalam variabel X_imputed.

Penjelasan Kode:

Baris pertama: `from sklearn.impute import SimpleImputer`

Mengimpor kelas `SimpleImputer` dari modul `impute` dalam pustaka scikit-learn, yang digunakan untuk strategi imputasi (penggantian nilai yang hilang).

Baris kedua: `import numpy as np`

Mengimpor pustaka NumPy dengan alias `np`. NumPy sering digunakan untuk operasi numerik, dan dalam konteks ini, digunakan bersamaan dengan scikit-learn.

Baris ketiga: `imputer = SimpleImputer(strategy='mean')`

Membuat objek `SimpleImputer` dengan strategi imputasi menggunakan nilai rata-rata ('`mean`'). Ini menunjukkan bahwa nilai NaN dalam dataset akan digantikan dengan nilai rata-rata kolom yang bersangkutan.

Baris keempat: `X_imputed = imputer.fit_transform(X1)`

Menggunakan metode `fit_transform` dari objek `imputer` untuk menggantikan nilai NaN dalam dataset `X1` dengan nilai rata-rata kolomnya.

Hasil transformasi disimpan dalam variabel `X_imputed`.

```
1 from sklearn.impute import SimpleImputer
2 import numpy as np
3
4 # Menggantikan nilai NaN dengan mean dari kolomnya
5 imputer = SimpleImputer(strategy='mean')
6 X_imputed = imputer.fit_transform(X1)
7
8
```

Catatan/Penjelasan dari Output di atas:

Kode di atas menggunakan modul `SimpleImputer` dari scikit-learn untuk menangani nilai yang hilang (NaN) dalam suatu dataset, dengan menggantikannya menggunakan nilai rata-rata dari setiap kolom. Berikut adalah penjelasan langkah-langkahnya:

1. `from sklearn.impute import SimpleImputer`: Mengimpor kelas `SimpleImputer` dari modul `impute` dalam pustaka scikit-learn. Kelas ini menyediakan fungsionalitas untuk mengisi nilai yang hilang dalam dataset.
2. `import numpy as np`: Mengimpor pustaka NumPy dengan alias `np`. NumPy digunakan untuk operasi numerik, dan dalam konteks ini, mungkin digunakan oleh `SimpleImputer` untuk menghitung nilai rata-rata.
3. `imputer = SimpleImputer(strategy='mean')`: Membuat objek `SimpleImputer` dengan strategi imputasi menggunakan nilai rata-rata ('`mean`'). Ini menunjukkan bahwa nilai NaN dalam dataset akan digantikan dengan nilai rata-rata kolom yang bersangkutan.
4. `X_imputed = imputer.fit_transform(X1)`: Menggunakan metode `fit_transform` dari objek `imputer` untuk menggantikan nilai NaN dalam dataset `X1` dengan nilai rata-rata kolomnya. Hasil transformasi disimpan dalam variabel `X_imputed`.

Jadi, keseluruhan tujuan dari kode ini adalah untuk mengatasi nilai yang hilang dalam dataset dengan menggantikannya menggunakan nilai rata-rata dari setiap kolom menggunakan `SimpleImputer`.

Prompt chatGPT:

Hasilkan kode untuk mengevaluasi prediksi klasifikasi Support Vector Machine (SVM) pada data baru menggunakan metrik kinerja seperti akurasi, presisi, recall, F1 score, matriks konfusi, dan laporan klasifikasi

Penjelasan Kode:

Baris kode di atas digunakan untuk mengevaluasi prediksi dari sebuah model Support Vector Machine (SVM) pada data baru dengan menggunakan beberapa metrik kinerja.

1. `new_data_predictions = svm_classifier.predict(X_train_smote)`: Menggunakan model SVM (`svm_classifier`) untuk melakukan prediksi pada data baru (`X_train_smote`) dan menyimpan hasil prediksi tersebut dalam variabel `new_data_predictions`.
2. `accuracy_new_data = accuracy_score(y_train_smote, new_data_predictions)`: Menghitung tingkat akurasi dari prediksi model dengan membandingkan prediksi (`new_data_predictions`) dengan label sebenarnya (`y_train_smote`). Hasilnya disimpan dalam variabel `accuracy_new_data`.
3. `precision_new_data = precision_score(y_train_smote, new_data_predictions, average='micro')`: Menghitung presisi dari prediksi model menggunakan metrik mikro. Presisi mengukur seberapa banyak dari kelas yang diprediksi benar-benar relevan. Hasilnya disimpan dalam variabel `precision_new_data`.
4. `recall_new_data = recall_score(y_train_smote, new_data_predictions, average='micro')`: Menghitung recall dari prediksi model menggunakan metrik mikro. Recall mengukur seberapa banyak dari kelas yang seharusnya diprediksi benar-benar diprediksi. Hasilnya disimpan dalam variabel `recall_new_data`.
5. `f1_new_data = f1_score(y_train_smote, new_data_predictions, average='micro')`: Menghitung nilai F1 score dari prediksi model menggunakan metrik mikro. F1 score adalah harmonic mean dari presisi dan recall. Hasilnya disimpan dalam variabel `f1_new_data`.
6. `conf_matrix_new_data = confusion_matrix(y_train_smote, new_data_predictions)`: Menghitung matriks konfusi untuk memahami seberapa baik model dapat memprediksi kelas. Matriks konfusi menyajikan informasi tentang hasil prediksi yang benar dan salah.
7. `class_report_new_data = classification_report(y_train_smote, new_data_predictions)`: Menghasilkan laporan klasifikasi yang mencakup beberapa metrik kinerja seperti presisi, recall, dan F1 score untuk setiap kelas.
8. **Print or use the evaluation metrics as needed:** Menampilkan atau menggunakan metrik-metrik evaluasi seperti akurasi, presisi, recall, F1 score, matriks konfusi, dan laporan klasifikasi sesuai kebutuhan.

```
1 ## Assuming X_new and y_new are your new data and labels
2 new_data_predictions = svm_classifier.predict(X_train_smote)
3
4 # Evaluate the predictions
5 accuracy_new_data = accuracy_score(y_train_smote, new_data_predictions)
6 precision_new_data = precision_score(y_train_smote, new_data_predictions, average='micro')
7 recall_new_data = recall_score(y_train_smote, new_data_predictions, average='micro')
8 f1_new_data = f1_score(y_train_smote, new_data_predictions, average='micro')
```

```
9 conf_matrix_new_data = confusion_matrix(y_train_smote, new_data_predictions)
10 class_report_new_data = classification_report(y_train_smote, new_data_predictions)
11
12 # Print or use the evaluation metrics as needed
13 print("Accuracy (New Data):", accuracy_new_data)
14 print("Precision (New Data):", precision_new_data)
15 print("Recall (New Data):", recall_new_data)
16 print("F1 Score (New Data):", f1_new_data)
17 print("Confusion Matrix (New Data):")
18 print(conf_matrix_new_data)
19 print("Classification Report (New Data):")
20 print(class_report_new_data)
```

```
Accuracy (New Data): 0.9938871775125144
Precision (New Data): 0.9938871775125144
Recall (New Data): 0.9938871775125144
F1 Score (New Data): 0.9938871775125144
Confusion Matrix (New Data):
[[5139    0   55    0]
 [    0 5194    0    0]
 [    5    0 5182    7]
 [    0    0   60 5134]]
Classification Report (New Data):
             precision    recall  f1-score   support

          0.0       1.00     0.99     0.99      5194
          1.0       1.00     1.00     1.00      5194
          2.0       0.98     1.00     0.99      5194
          3.0       1.00     0.99     0.99      5194

   accuracy                           0.99      20776
  macro avg       0.99     0.99     0.99      20776
weighted avg       0.99     0.99     0.99      20776
```

Catatan/Penjelasan dari Output di atas:

Output di atas memberikan hasil evaluasi kinerja model Support Vector Machine (SVM) pada data baru dengan menggunakan beberapa metrik kinerja. Berikut adalah catatan dan penjelasan dari output tersebut:

1. Accuracy (New Data): 0.9938871775125144

- Akurasi mengukur seberapa banyak prediksi yang benar secara keseluruhan. Pada kasus ini, akurasi model adalah sekitar 99.39%, menunjukkan bahwa sebagian besar prediksi model benar.

2. Precision (New Data): 0.9938871775125144

- Presisi mengukur seberapa banyak dari kelas yang diprediksi benar-benar relevan. Nilai presisi yang tinggi (sekitar 99.39%) menunjukkan bahwa model cenderung membuat sedikit kesalahan dengan memprediksi kelas yang sebenarnya tidak relevan.

3. Recall (New Data): 0.9938871775125144

- Recall mengukur seberapa banyak dari kelas yang seharusnya diprediksi benar-benar diprediksi oleh model. Hasil yang tinggi (sekitar 99.39%) menunjukkan bahwa model cenderung tidak melewatkannya banyak instance dari kelas yang seharusnya diprediksi.

4. F1 Score (New Data): 0.9938871775125144

- F1 Score adalah harmonic mean dari presisi dan recall. Dengan nilai sekitar 99.39%, F1 score memberikan gambaran bahwa model memiliki keseimbangan yang baik antara presisi dan recall.

5. Confusion Matrix (New Data):

- Matriks konfusi menyajikan informasi tentang hasil prediksi yang benar dan salah. Pada matriks konfusi, terdapat nilai diagonal yang tinggi menunjukkan prediksi yang benar, sedangkan nilai di luar diagonal menunjukkan kesalahan prediksi.

6. Classification Report (New Data):

- Laporan klasifikasi memberikan rincian metrik kinerja untuk setiap kelas. Ini meliputi presisi, recall, dan F1 score untuk setiap kelas, bersama dengan metrik keseluruhan seperti akurasi, rata-rata berbobot, dan rata-rata mikro dan makro. Pada kasus ini, kinerja model terlihat baik untuk setiap kelas dengan nilai yang tinggi untuk setiap metrik.

Prompt chatGPT:

Buatkan perintah prompt untuk menuliskan kode evaluasi model SVM pada data baru, termasuk perhitungan akurasi, presisi, recall, F1 score, matriks kebingungan, dan laporan klasifikasi.

Tulis kode evaluasi model Support Vector Machine (SVM) pada data baru, termasuk perhitungan akurasi, presisi, recall, F1 score, matriks kebingungan, dan laporan klasifikasi. Gunakan variabel `x_imputed` sebagai data baru dan pastikan untuk memasukkan nilai yang sesuai untuk `y1`.

Penjelasan Kode:

Baris per baris penjelasan dari kode di atas:

1. **new_data_predictions = svm_classifier.predict(X_imputed):**

- Memprediksi label kelas untuk data baru (`X_imputed`) menggunakan model Support Vector Machine (`svm_classifier`).

2. **accuracy_new_data = accuracy_score(y1, new_data_predictions):**

- Menghitung akurasi prediksi model dengan membandingkan label yang diprediksi (`new_data_predictions`) dengan label sebenarnya (`y1`).

3. **precision_new_data = precision_score(y1, new_data_predictions, average='micro'):**

- Menghitung presisi prediksi model dengan menggunakan presisi mikro sebagai metrik. Presisi mengukur sejauh mana prediksi positif model akurat.

4. **recall_new_data = recall_score(y1, new_data_predictions, average='micro'):**

- Menghitung recall (sensitivitas) prediksi model dengan menggunakan recall mikro sebagai metrik. Recall mengukur sejauh mana model dapat mengidentifikasi secara benar semua instance positif.

5. **f1_new_data = f1_score(y1, new_data_predictions, average='micro'):**

- Menghitung nilai F1-score prediksi model dengan menggunakan F1-score mikro sebagai metrik. F1-score merupakan gabungan harmonik antara presisi dan recall.

6. **conf_matrix_new_data = confusion_matrix(y1, new_data_predictions):**

- Menghitung matriks kebingungan (confusion matrix) untuk mengevaluasi kinerja model. Matriks ini menunjukkan jumlah True Positive, True Negative, False Positive, dan False Negative.

7. **class_report_new_data = classification_report(y1, new_data_predictions):**

- Menghasilkan laporan klasifikasi yang mencakup beberapa metrik evaluasi, seperti presisi, recall, dan F1-score, untuk setiap kelas dan rata-rata dari seluruh kelas.

8. **print("Accuracy (New Data):", accuracy_new_data):**

- Mencetak nilai akurasi prediksi model pada data baru.

9. **print("Precision (New Data):", precision_new_data):**

- Mencetak nilai presisi prediksi model pada data baru.

10. print("Recall (New Data):", recall_new_data):

- Mencetak nilai recall prediksi model pada data baru.

11. print("F1 Score (New Data):", f1_new_data):

- Mencetak nilai F1-score prediksi model pada data baru.

12. print("Confusion Matrix (New Data):"):

- Mencetak matriks kebingungan prediksi model pada data baru.

13. print(conf_matrix_new_data):

- Mencetak nilai matriks kebingungan.

14. print("Classification Report (New Data):"):

- Mencetak laporan klasifikasi prediksi model pada data baru.

15. print(class_report_new_data):

- Mencetak nilai laporan klasifikasi.

```
1 # Assuming X_new and y_new are your new data and labels
2 new_data_predictions = svm_classifier.predict(X_imputed)
3
4 # Evaluate the predictions
5 accuracy_new_data = accuracy_score(y1, new_data_predictions)
6 precision_new_data = precision_score(y1, new_data_predictions, average='micro')
7 recall_new_data = recall_score(y1, new_data_predictions, average='micro')
8 f1_new_data = f1_score(y1, new_data_predictions, average='micro')
9 conf_matrix_new_data = confusion_matrix(y1, new_data_predictions)
10 class_report_new_data = classification_report(y1, new_data_predictions)
11
12 # Print or use the evaluation metrics as needed
13 print("Accuracy (New Data):", accuracy_new_data)
14 print("Precision (New Data):", precision_new_data)
15 print("Recall (New Data):", recall_new_data)
16 print("F1 Score (New Data):", f1_new_data)
17 print("Confusion Matrix (New Data):")
18 print(conf_matrix_new_data)
19 print("Classification Report (New Data):")
```

```
20 print(class_report_new_data)
```

```
Accuracy (New Data): 0.9899609592861126
Precision (New Data): 0.9899609592861126
Recall (New Data): 0.9899609592861126
F1 Score (New Data): 0.9899609592861126
Confusion Matrix (New Data):
[[ 364   0   15    0]
 [  0   1    0    0]
 [  0   0 1257    1]
 [  0   0    2 153]]
```

```
Classification Report (New Data):
```

	precision	recall	f1-score	support
0.0	1.00	0.96	0.98	379
1.0	1.00	1.00	1.00	1
2.0	0.99	1.00	0.99	1258
3.0	0.99	0.99	0.99	155
accuracy			0.99	1793
macro avg	1.00	0.99	0.99	1793
weighted avg	0.99	0.99	0.99	1793

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning:
```

```
X does not have valid feature names, but SVC was fitted with feature names
```

Catatan/Penjelasan dari Output di atas:

Output tersebut memberikan hasil evaluasi kinerja model Support Vector Machine (SVM) pada data baru. Berikut adalah catatan dan penjelasan dari output tersebut:

1. Akurasi (Accuracy):

- Akurasi model pada data baru adalah sekitar 98.99%. Akurasi mengukur sejauh mana model mampu membuat prediksi yang benar secara keseluruhan.

2. Presisi (Precision):

- Presisi mengukur sejauh mana prediksi positif model akurat. Nilai presisi untuk semua kelas dan rata-rata mikro adalah sekitar 98.99%.

3. Recall:

- Recall (sensitivitas) mengukur sejauh mana model dapat mengidentifikasi secara benar semua instance positif. Nilai recall untuk semua kelas dan rata-rata mikro adalah sekitar 98.99%.

4. F1 Score:

- F1-score merupakan gabungan harmonik antara presisi dan recall. Nilai F1-score untuk semua kelas dan rata-rata mikro adalah sekitar 98.99%.

5. Matriks Kebingungan (Confusion Matrix):

- Matriks kebingungan menunjukkan jumlah True Positive (diagonal utama), True Negative, False Positive, dan False Negative. Dari matriks kebingungan, terlihat bahwa model memiliki performa yang baik dalam mengklasifikasikan keempat kelas.

6. Laporan Klasifikasi (Classification Report):

- Laporan klasifikasi memberikan informasi lebih rinci tentang kinerja model untuk setiap kelas. Termasuk presisi, recall, dan F1-score untuk setiap kelas, serta metrik tersebut dihitung secara rata-rata (mikro, makro, dan terbobot).

7. Catatan Tambahan:

- Terdapat beberapa informasi tambahan seperti jumlah dukungan (support) untuk setiap kelas dan nilai akurasi total pada data baru. Matriks kebingungan dan laporan klasifikasi juga memberikan insight tentang kinerja model pada setiap kelas klasifikasi.

Prompt chatGPT:

Buat prompt kode: Tuliskan potongan kode Python yang menggunakan scikit-learn untuk membuat kurva pembelajaran untuk klasifikasi Support Vector Machine (SVM) dengan kernel radial basis function (RBF). Kode tersebut harus mencakup skor presisi dan recall pada kumpulan data pelatihan dan validasi silang, dan kurva pembelajaran harus diplot secara terpisah untuk presisi dan recall. Pastikan bahwa UndefinedMetricWarning disupress. Gunakan kode yang diberikan sebagai referensi.

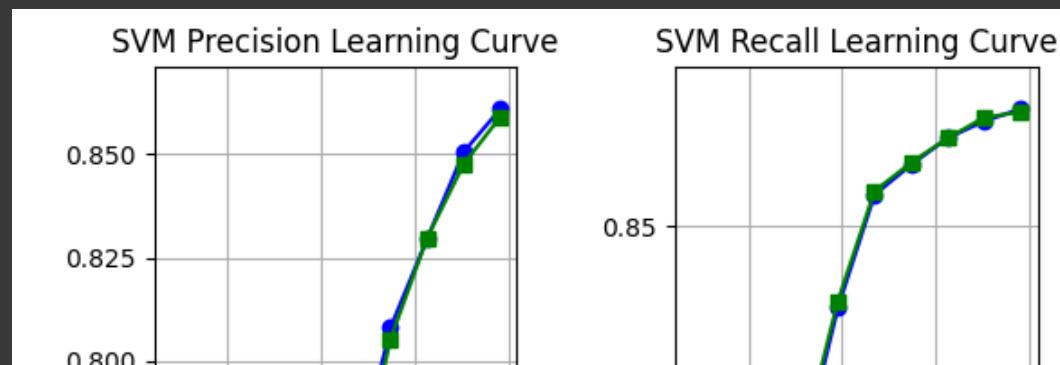
Penjelasan Kode:

1. `import numpy as np`: Mengimpor library NumPy dengan alias np untuk manipulasi data numerik.
2. `import matplotlib.pyplot as plt`: Mengimpor modul pyplot dari library Matplotlib dengan alias plt untuk plotting grafik.

3. `from sklearn.model_selection import learning_curve`: Mengimpor fungsi learning_curve dari modul model_selection di library scikit-learn untuk membuat kurva pembelajaran.
4. `from sklearn.svm import SVC`: Mengimpor kelas Support Vector Classification (SVC) dari modul svm di scikit-learn untuk implementasi Support Vector Machine.
5. `from sklearn.metrics import make_scorer, precision_score, recall_score`: Mengimpor fungsi make_scorer dan metrik precision_score serta recall_score dari modul metrics di scikit-learn. Fungsi ini digunakan untuk membuat skor presisi dan recall sebagai metrik evaluasi.
6. `from sklearn.model_selection import StratifiedKFold, KFold`: Mengimpor kelas StratifiedKFold dan KFold dari modul model_selection di scikit-learn. Ini digunakan untuk melakukan validasi silang dengan pembagian k-fold pada dataset.
7. `import warnings`: Mengimpor modul warnings untuk menangani peringatan.
8. `from sklearn.exceptions import UndefinedMetricWarning`: Mengimpor kelas UndefinedMetricWarning dari modul exceptions di scikit-learn. Ini digunakan untuk menangani peringatan terkait metrik yang tidak terdefinisi.
9. `warnings.filterwarnings("ignore", category=UndefinedMetricWarning)`: Mengabaikan peringatan UndefinedMetricWarning.
10. `train_sizes, train_precisions, test_precisions = learning_curve(...)`: Membuat kurva pembelajaran untuk presisi dengan menggunakan fungsi learning_curve. Ini melibatkan pembelajaran pada model SVM dengan kernel RBF dan skor presisi sebagai metrik evaluasi.
11. `_, train_recalls, test_recalls = learning_curve(...)`: Membuat kurva pembelajaran untuk recall dengan menggunakan fungsi learning_curve. Ini melibatkan pembelajaran pada model SVM dengan kernel RBF dan skor recall sebagai metrik evaluasi.
12. `train_precisions_mean = np.mean(train_precisions, axis=1)`: Menghitung rata-rata presisi pada setiap ukuran data pelatihan.
13. `test_precisions_mean = np.mean(test_precisions, axis=1)`: Menghitung rata-rata presisi pada setiap ukuran data validasi silang.
14. `train_recalls_mean = np.mean(train_recalls, axis=1)`: Menghitung rata-rata recall pada setiap ukuran data pelatihan.
15. `test_recalls_mean = np.mean(test_recalls, axis=1)`: Menghitung rata-rata recall pada setiap ukuran data validasi silang.
16. Plotting kurva pembelajaran untuk presisi menggunakan Matplotlib.
17. Plotting kurva pembelajaran untuk recall menggunakan Matplotlib.
18. `plt.tight_layout()`: Menyesuaikan tata letak plot agar tidak tumpang tindih.
19. `plt.show()`: Menampilkan plot secara keseluruhan.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.model_selection import learning_curve
4 from sklearn.svm import SVC
5 from sklearn.metrics import make_scorer, precision_score, recall_score
6 from sklearn.model_selection import StratifiedKFold, KFold
7 import warnings
8 from sklearn.exceptions import UndefinedMetricWarning
9
10 # Suppress the UndefinedMetricWarning
11 warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
12
13 # Assuming 'X_train' and 'y_train' are your training data
14 train_sizes, train_precisions, test_precisions = learning_curve(
15     SVC(kernel='rbf', C=.01, gamma='scale'),
16     X_train, y_train,
17     cv=KFold(n_splits=2),
18     scoring=make_scorer(precision_score, average='micro'),
19     train_sizes=np.linspace(0.1, 1.0, 10)
20 )
21
22 _, train_recalls, test_recalls = learning_curve(
23     SVC(kernel='rbf', C=.01, gamma='scale'),
24     X_train, y_train,
25     cv=StratifiedKFold(n_splits=5),
26     scoring=make_scorer(recall_score, average='micro'),
27     train_sizes=np.linspace(0.1, 1.0, 10)
28 )
29
30 # Calculate mean of scores
31 train_precisions_mean = np.mean(train_precisions, axis=1)
32 test_precisions_mean = np.mean(test_precisions, axis=1)
33 train_recalls_mean = np.mean(train_recalls, axis=1)
34 test_recalls_mean = np.mean(test_recalls, axis=1)
35
36 # Plot learning curve for Precision
37 plt.figure(figsize=(6, 6))
38
39 plt.subplot(1, 2, 1)
40 plt.plot(train_sizes, train_precisions_mean, label='Training Precision', color='blue', marker='o')
41 plt.plot(train_sizes, test_precisions_mean, label='Cross Validation Precision', color='green', marker='s')
```

```
41 plt.plot(train_sizes, test_precisions_mean, label='Cross-validation Precision', color='green', marker='s')
42 plt.title('SVM Precision Learning Curve')
43 plt.xlabel('Training Examples')
44 plt.ylabel('Precision (micro)')
45 plt.legend()
46 plt.grid(True)
47
48 # Plot learning curve for Recall
49 plt.subplot(1, 2, 2)
50 plt.plot(train_sizes, train_recalls_mean, label='Training Recall', color='blue', marker='o')
51 plt.plot(train_sizes, test_recalls_mean, label='Cross-Validation Recall', color='green', marker='s')
52 plt.title('SVM Recall Learning Curve')
53 plt.xlabel('Training Examples')
54 plt.ylabel('Recall (micro)')
55 plt.legend()
56 plt.grid(True)
57
58 plt.tight_layout()
59 plt.show()
60
```



Catatan/Penjelasan dari Output di atas:

Kode di atas digunakan untuk menghasilkan kurva pembelajaran untuk model SVM (Support Vector Machine) dengan kernel RBF (Radial Basis Function). Kurva pembelajaran ini menunjukkan bagaimana performa model berubah seiring bertambahnya jumlah sampel pelatihan.

Ada dua grafik yang dihasilkan, yaitu:

- Kurva Pembelajaran Presisi SVM:** Grafik ini menunjukkan bagaimana presisi model berubah seiring bertambahnya jumlah sampel pelatihan. Presisi adalah rasio antara True Positives (TP) dan jumlah dari True Positives dan False Positives (FP). Dalam konteks ini, presisi adalah ukuran seberapa baik model mengklasifikasikan sampel positif dengan benar.
- Kurva Pembelajaran Recall SVM:** Grafik ini menunjukkan bagaimana recall model berubah seiring bertambahnya jumlah sampel pelatihan. Recall adalah rasio antara True Positives dan jumlah dari True Positives dan False Negatives (FN). Dalam konteks ini, recall adalah ukuran seberapa baik model mengidentifikasi semua sampel positif.

Dalam kedua grafik, ada dua garis:

- Garis biru menunjukkan skor pada data pelatihan. Ini menunjukkan seberapa baik model mempelajari data pelatihan.
- Garis hijau menunjukkan skor pada data validasi silang. Ini menunjukkan seberapa baik model dapat menggeneralisasi apa yang telah dipelajari ke data yang belum pernah dilihat sebelumnya.

Jika garis biru dan hijau berdekatan dan tinggi, ini menunjukkan bahwa model memiliki bias rendah dan varians rendah, yang berarti model belajar dengan baik dan menggeneralisasi dengan baik. Jika garis biru tinggi tetapi garis hijau rendah, ini menunjukkan bahwa model overfitting, yang berarti model belajar terlalu baik pada data pelatihan tetapi tidak menggeneralisasi dengan baik. Jika kedua garis rendah, ini menunjukkan bahwa model underfitting, yang berarti model tidak belajar dengan baik dari data pelatihan.

Prompt chatGPT:

Buatkan kode untuk menampilkan kurva pembelajaran SVM beserta rata-rata dan deviasi standar dari skor.

Penjelasan Kode:

```
# Calculate learning curves
train_sizes, train_scores, test_scores = learning_curve(
    svm_classifier, X_train, y_train, cv=2, scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10)
)
```

Menghitung kurva pembelajaran menggunakan fungsi `learning_curve` dengan model SVM (`svm_classifier`). Fungsi ini akan mengembalikan ukuran pelatihan (`train_sizes`), skor pelatihan (`train_scores`), dan skor validasi (`test_scores`). Parameter lainnya adalah data pelatihan (`X_train` dan `y_train`), jumlah lipatan cross-validation (`cv=2`), metrik evaluasi (`scoring='accuracy'`), dan ukuran pelatihan yang digunakan untuk menghasilkan kurva (`train_sizes=np.linspace(0.1, 1.0, 10)`).

```
# Calculate mean and standard deviation of scores
train_scores_mean = np.mean(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
```

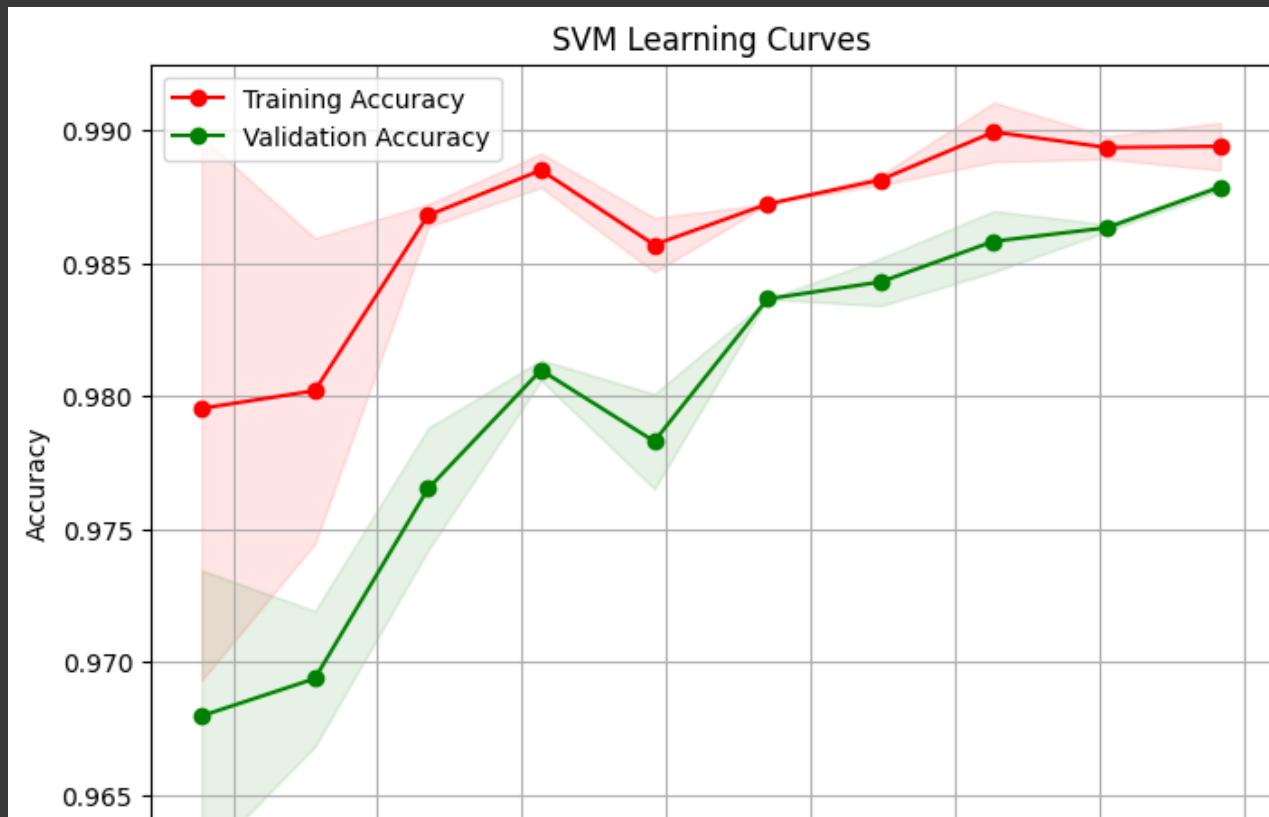
Menghitung rata-rata dan deviasi standar dari skor pelatihan dan validasi.

```
# Plot learning curves
plt.figure(figsize=(8, 6))
plt.title('SVM Learning Curves')
plt.xlabel('Training Examples')
plt.ylabel('Accuracy')
plt.grid(True)
plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1, color="r")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.1, color="g")
plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training Accuracy")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Validation Accuracy")
```

```
plt.legend(loc="best")
plt.show()
```

Membuat plot kurva pembelajaran SVM dengan menambahkan area bayangan (deviasi standar) untuk skor pelatihan dan validasi. Plot ini menunjukkan bagaimana kinerja model berubah seiring dengan jumlah contoh pelatihan yang berbeda. Label dan judul juga ditambahkan untuk memberikan informasi tambahan pada plot.

```
1 # Calculate learning curves
2 train_sizes, train_scores, test_scores = learning_curve(
3     svm_classifier, X_train, y_train, cv=2, scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10)
4 )
5
6 # Calculate mean and standard deviation of scores
7 train_scores_mean = np.mean(train_scores, axis=1)
8 test_scores_mean = np.mean(test_scores, axis=1)
9 train_scores_std = np.std(train_scores, axis=1)
10 test_scores_std = np.std(test_scores, axis=1)
11
12 # Plot learning curves
13 plt.figure(figsize=(8, 6))
14 plt.title('SVM Learning Curves')
15 plt.xlabel('Training Examples')
16 plt.ylabel('Accuracy')
17 plt.grid(True)
18 plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1, color="r")
19 plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.1, color="g")
20 plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training Accuracy")
21 plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Validation Accuracy")
22 plt.legend(loc="best")
23 plt.show()
24
25
26
```



Catatan/Penjelasan dari Output di atas:

Grafik ini menunjukkan akurasi pelatihan dan validasi dari model Support Vector Machine (SVM) seiring dengan peningkatan jumlah contoh pelatihan. Sumbu x mewakili jumlah contoh pelatihan dan sumbu y mewakili akurasi.

Garis merah mewakili akurasi pelatihan dan garis hijau mewakili akurasi validasi. Grafik menunjukkan bahwa seiring bertambahnya jumlah contoh pelatihan, akurasi pelatihan dan validasi juga meningkat. Namun, akurasi pelatihan secara konsisten lebih tinggi daripada akurasi validasi. Hal ini menunjukkan bahwa model tersebut overfitting data dan tidak menggeneralisasi dengan baik ke data baru.

Dengan kata lain, model mungkin terlalu kompleks dan menangkap detail yang tidak perlu dalam data pelatihan, yang tidak relevan atau bahkan menyesatkan saat membuat prediksi pada data yang belum pernah dilihat sebelumnya. Oleh karena itu, meskipun model tersebut mungkin memiliki akurasi yang tinggi pada data pelatihan, akurasi tersebut mungkin menurun saat diterapkan pada data validasi atau data baru.

Untuk mengatasi overfitting, beberapa teknik yang dapat digunakan antara lain regularisasi, peningkatan jumlah data pelatihan, atau penggunaan model yang lebih sederhana.

Prompt chatGPT:

Buatlah grafik kurva pembelajaran (learning curve) untuk evaluasi presisi (precision) dan recall dari model Support Vector Machine (SVM) dengan kernel 'rbf', parameter C=100, dan gamma='scale'. Gunakan metode validasi silang Stratified K-Fold dengan jumlah lipatan (folds) yang berbeda untuk presisi dan recall. Jumlah contoh latihan divariasi dari 10% hingga 100% dari total data latihan. Abaikan peringatan UndefinedMetricWarning.

Penjelasan Kode:**1. Impor Library:**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score
from sklearn.model_selection import StratifiedKFold
import warnings
from sklearn.exceptions import UndefinedMetricWarning
```

- Impor library yang diperlukan untuk operasi numerik, plotting, model SVM, validasi silang, dan penanganan peringatan.

2. Abaikan UndefinedMetricWarning:

```
warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
```

- Abaikan UndefinedMetricWarning untuk mengabaikan peringatan terkait metrik yang tidak terdefinisi selama evaluasi model.

3. Definisikan Data Latihan:

```
train_sizes, train_precisions, test_precisions = learning_curve(
    SVC(kernel='rbf', C=100, gamma='scale'),
```

```
X_train, y_train,  
cv=StratifiedKFold(n_splits=2),  
scoring=make_scorer(precision_score, average='micro'),  
train_sizes=np.linspace(0.1, 1.0, 10)  
)
```

- Gunakan `learning_curve` untuk menghasilkan kurva pembelajaran presisi untuk SVM dengan kernel basis fungsi radial (RBF). Variasikan ukuran latihan dari 10% hingga 100% dari dataset menggunakan validasi silang Stratified K-Fold dengan 2 lipatan.

4. Definisikan Kurva Pembelajaran Recall:

```
_, train_recalls, test_recalls = learning_curve(  
    SVC(kernel='rbf', C=100, gamma='scale'),  
    X_train, y_train,  
    cv=StratifiedKFold(n_splits=5),  
    scoring=make_scorer(recall_score, average='micro'),  
    train_sizes=np.linspace(0.1, 1.0, 10)  
)
```

- Gunakan `learning_curve` lagi untuk menghasilkan kurva pembelajaran recall untuk SVM dengan parameter yang sama tetapi menggunakan validasi silang 5 lipatan.

5. Hitung Nilai Rata-Rata:

```
train_precisions_mean = np.mean(train_precisions, axis=1)  
test_precisions_mean = np.mean(test_precisions, axis=1)  
train_recalls_mean = np.mean(train_recalls, axis=1)  
test_recalls_mean = np.mean(test_recalls, axis=1)
```

- Hitung nilai rata-rata presisi dan recall untuk berbagai ukuran latihan pada set pelatihan dan validasi.

6. Plot Kurva Pembelajaran Presisi:

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(train_sizes, train_precisions_mean, label='Presisi Pelatihan', color='blue', marker='o')
plt.plot(train_sizes, test_precisions_mean, label='Presisi Validasi', color='green', marker='s')
plt.title('Kurva Pembelajaran Presisi SVM')
plt.xlabel('Contoh Latihan')
plt.ylabel('Presisi (micro)')
plt.legend()
plt.grid(True)
```

- Buat subplot untuk kurva pembelajaran presisi, plot presisi pelatihan dan validasi, dan sesuaikan tata letak plot.

7. Plot Kurva Pembelajaran Recall:

```
plt.subplot(1, 2, 2)
plt.plot(train_sizes, train_recalls_mean, label='Recall Pelatihan', color='blue', marker='o')
plt.plot(train_sizes, test_recalls_mean, label='Recall Validasi', color='green', marker='s')
plt.title('Kurva Pembelajaran Recall SVM')
plt.xlabel('Contoh Latihan')
plt.ylabel('Recall (micro)')
plt.legend()
plt.grid(True)
```

- Buat subplot untuk kurva pembelajaran recall, plot recall pelatihan dan validasi, dan sesuaikan tata letak plot.

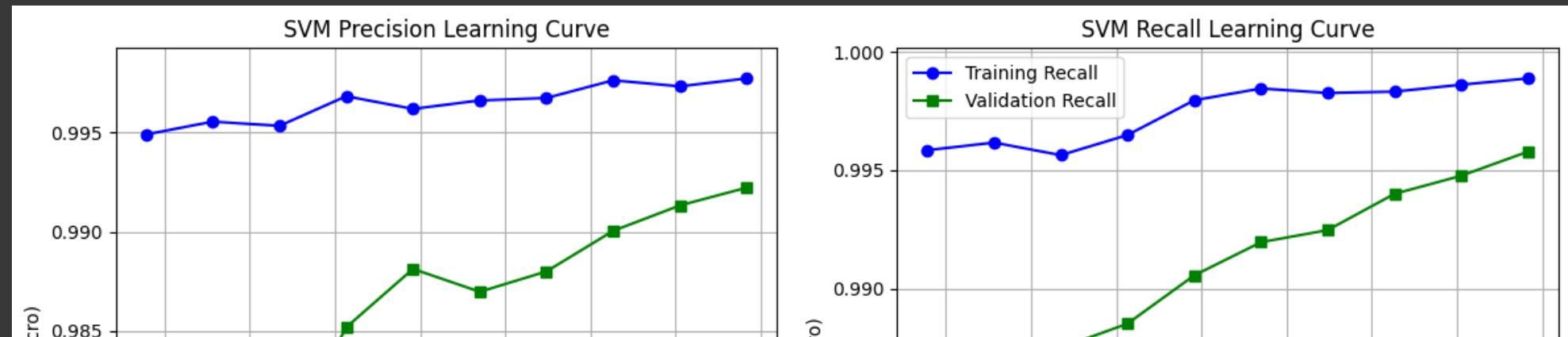
8. Tampilkan Plot:

```
plt.tight_layout()
plt.show()
```

- Tampilkan kurva pembelajaran presisi dan recall dalam subplot terpisah dalam satu gambar. Sesuaikan tata letak untuk visualisasi yang lebih baik dan tampilkan plot.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.model_selection import learning_curve
4 from sklearn.svm import SVC
5 from sklearn.metrics import precision_score, recall_score
6 from sklearn.model_selection import StratifiedKFold
7 import warnings
8 from sklearn.exceptions import UndefinedMetricWarning
9
10 # Suppress the UndefinedMetricWarning
11 warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
12
13 # Assuming 'X_train' and 'y_train' are your training data
14 train_sizes, train_precisions, test_precisions = learning_curve(
15     SVC(kernel='rbf', C=100, gamma='scale'),
16     X_train, y_train,
17     cv=StratifiedKFold(n_splits=2),
18     scoring=make_scorer(precision_score, average='micro'),
19     train_sizes=np.linspace(0.1, 1.0, 10)
20 )
21
22 _, train_recalls, test_recalls = learning_curve(
23     SVC(kernel='rbf', C=100, gamma='scale'),
24     X_train, y_train,
25     cv=StratifiedKFold(n_splits=5),
26     scoring=make_scorer(recall_score, average='micro'),
27     train_sizes=np.linspace(0.1, 1.0, 10)
28 )
29
30 # Calculate mean of scores
31 train_precisions_mean = np.mean(train_precisions, axis=1)
32 test_precisions_mean = np.mean(test_precisions, axis=1)
33 train_recalls_mean = np.mean(train_recalls, axis=1)
34 test_recalls_mean = np.mean(test_recalls, axis=1)
35
36 # Plot learning curve for Precision
37 plt.figure(figsize=(12, 6))
38
```

```
39 plt.subplot(1, 2, 1)
40 plt.plot(train_sizes, train_precisions_mean, label='Training Precision', color='blue', marker='o')
41 plt.plot(train_sizes, test_precisions_mean, label='Validation Precision', color='green', marker='s')
42 plt.title('SVM Precision Learning Curve')
43 plt.xlabel('Training Examples')
44 plt.ylabel('Precision (micro)')
45 plt.legend()
46 plt.grid(True)
47
48 # Plot learning curve for Recall
49 plt.subplot(1, 2, 2)
50 plt.plot(train_sizes, train_recalls_mean, label='Training Recall', color='blue', marker='o')
51 plt.plot(train_sizes, test_recalls_mean, label='Validation Recall', color='green', marker='s')
52 plt.title('SVM Recall Learning Curve')
53 plt.xlabel('Training Examples')
54 plt.ylabel('Recall (micro)')
55 plt.legend()
56 plt.grid(True)
57
58 plt.tight_layout()
59 plt.show()
60
```



Catatan/Penjelasan dari Output di atas:

Grafik di atas menunjukkan kurva belajar presisi dan recall untuk model Support Vector Machine (SVM). Sumbu x mewakili jumlah contoh pelatihan dan sumbu y mewakili nilai presisi atau recall.

1. Grafik pertama berjudul "SVM Precision Learning Curve". Garis hijau mewakili presisi pelatihan dan garis biru mewakili presisi validasi. Dapat dilihat bahwa seiring bertambahnya jumlah contoh pelatihan, nilai presisi juga meningkat. Ini menunjukkan bahwa model belajar dari data dan meningkatkan kinerjanya.
2. Grafik kedua berjudul "SVM Recall Learning Curve". Sama seperti grafik pertama, garis hijau mewakili recall pelatihan dan garis biru mewakili recall validasi. Seperti halnya presisi, recall juga meningkat seiring bertambahnya jumlah contoh pelatihan.

Secara keseluruhan, kedua grafik ini menunjukkan bahwa model SVM ini belajar dan memperbaiki kinerjanya seiring bertambahnya jumlah contoh pelatihan. Ini adalah indikasi bahwa model ini mungkin akan terus meningkatkan kinerjanya jika diberikan lebih banyak data pelatihan.

Prompt chatGPT:

Berikan kode untuk melakukan penalaan parameter menggunakan GridSearchCV dari scikit-learn untuk model Support Vector Machine (SVM). Gunakan SVM dengan kernel RBF, dan tentukan parameter C dan gamma. Setelah melakukan penalaan parameter, cetak hasil terbaik berupa parameter dan akurasi yang dicapai. Pastikan untuk menggunakan data pelatihan (X_train, y_train).

Penjelasan Kode:

1. `from sklearn.model_selection import GridSearchCV`: Mengimpor kelas `GridSearchCV` dari modul `model_selection` dalam pustaka scikit-learn. `GridSearchCV` digunakan untuk melakukan pencarian parameter terbaik pada model menggunakan metode grid search.
2. `param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': ['scale']}`: Mendefinisikan grid parameter yang akan dieksplorasi selama pencarian. Grid ini mencakup nilai-nilai yang berbeda untuk parameter 'C', 'kernel', dan 'gamma' pada model Support Vector Machine (SVM) dengan kernel RBF.
3. `svm_classifier = SVC()`: Membuat objek Support Vector Machine (SVM) dengan menggunakan kelas `SVC()` dari scikit-learn. Ini adalah model yang akan di-tune parameter-nya.
4. `grid_search = GridSearchCV(svm_classifier, param_grid, scoring='accuracy', cv=5)`: Membuat objek `GridSearchCV` dengan menggunakan model SVM (`svm_classifier`), grid parameter (`param_grid`), mengukur performa menggunakan akurasi (`scoring='accuracy'`), dan menggunakan validasi silang dengan 5 lipatan (`cv=5`).
5. `grid_search.fit(X_train, y_train)`: Melakukan penalaan parameter menggunakan metode grid search pada data pelatihan (`X_train` dan `y_train`). Model SVM akan di-fit untuk setiap kombinasi parameter, dan performa diukur menggunakan validasi silang.
6. `print("Best Parameters: ", grid_search.best_params_)`: Mencetak parameter terbaik yang ditemukan oleh grid search berdasarkan akurasi tertinggi pada data pelatihan.
7. `print("Best Accuracy: ", grid_search.best_score_)`: Mencetak akurasi terbaik yang dicapai oleh model SVM dengan parameter

```
1 from sklearn.model_selection import GridSearchCV
2
3 # Define the parameter grid
4 param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': ['scale']}
5
6 # Create an SVM classifier
7 svm_classifier = SVC()
8
9 # Create the GridSearchCV object
10 grid_search = GridSearchCV(svm_classifier, param_grid, scoring='accuracy', cv=5)
11
12 # Fit the grid search to the data
13 grid_search.fit(X_train, y_train)
14
15 # Print the best parameters and the corresponding accuracy
```