Pembukaan (perkenalan diri, clasifikasi menggunakan model linear regresi, dataset word…)

Lanjut ke dataset. Pengambilan dataset dari mana, jelaskan kolom apa saja, lanjut

Agar file dapat dibaca dengan mudah gunakan library

```
# Contoh membaca file CSV dari Google Drive menggunakan pandas
import pandas as pd
```

fungsi library pandas tersebut agar memudahkan untuk memanipulasi data yang digunakan. Pandas adalah library open-source yang menyediakan struktur data dan alat analisis data untuk bahasa pemrograman Python. Pandas sering digunakan dalam analisis data, pengolahan data, dan kegiatan ilmu data. Library ini menyediakan dua struktur data utama: Series dan DataFrame.

Selanjutnya tahap EDA

EDA atau Exploratory Data Analysis adalah suatu pendekatan untuk menganalisis dataset dengan merinci dan memahami karakteristik utama dari data tersebut. Tujuan dari EDA adalah untuk merumuskan hipotesis, mengidentifikasi pola, menemukan anomali, dan memahami struktur data sebelum menerapkan model statistik atau algoritma machine learning.

Untuk menampilkan info dataframe digunakan syntax df.info

```
#menampilkan info dataframe
df.info()
```

Selanjutnya agar mengetahui jumlah banyaknya data yang kosong atau Nan digunakan syntax

```
# Menghitung jumlah NaN pada setiap kolom
nan_count = df.isna().sum()
nan_count
```

df.isna(): Metode ini menghasilkan DataFrame yang memiliki nilai True di setiap sel yang berisi NaN dan False untuk sel yang memiliki nilai valid.

sum(): Setelah mendapatkan DataFrame dengan nilai True dan False, kita menggunakan metode sum() untuk menghitung jumlah nilai True (yang merepresentasikan NaN) untuk setiap kolom.

```
# Menghapus ketiga kolom
kolom_yang_dihapus = ['Revealed comparative advantage', 'World Growth
(%)', 'Country Growth (%)']
df = df.drop(kolom_yang_dihapus, axis=1)

# Menampilkan DataFrame setelah penghapusan kolom
df.columns
```

df.drop(kolom_yang_dihapus, axis=1): Metode drop digunakan untuk menghapus kolom-kolom yang tercantum dalam kolom_yang_dihapus. Argumen axis=1 menunjukkan bahwa yang dihapus adalah kolom, bukan baris.

Setelah menghapus beberapa kolom, sisanya masih terdapat kolom yang memiliki null atau kosong. Nilai null tersebut dapat diisi dengan beberapa metode seperti mean, modus, nilai sesuai keinginan dan beberapa metode lainnya. Dalam kasus ini digunakan dengan metode mean.

```
# Mengisi nilai NaN dengan nilai rata-rata setiap kolom
df = df.fillna(df.mean())

# Menampilkan DataFrame setelah pengisian nilai NaN
df
```

pengisian nilai NaN dengan nilai rata-rata adalah salah satu cara umum untuk mengatasi missing values. Namun, pendekatan ini dapat bervariasi tergantung pada konteks proyek dan tujuan analisis data.

Pada plot histogram ini

```
# Set the size of the plot
plt.figure(figsize=(18, 10))

# Create histograms for all columns in the DataFrame
df.hist(bins=20, figsize=(20,15), edgecolor='black', grid=False,
color='blue')

# Adjust layout for better visualization
plt.tight_layout()

# Set plot title
plt.suptitle('Histograms for All Columns in the DataFrame', y=1.02)
```

plt.figure(figsize=(18, 10)): Ini mengatur ukuran total gambar plot.

output tersebut menampilkan representasi visual dari distribusi frekuensi nilai-nilai dalam suatu kolom.

```
# Buat pair plot
sns.pairplot(df, height=2, aspect=1.5)
plt.suptitle('Pair Plot untuk Beberapa Kolom', y=1.02)

# Tampilkan plot
plt.show()
```

syntax ini digunakan untuk membuat pair plot menggunakan library Seaborn (sns) dan matplotlib. Pair plot memperlihatkan scatter plot untuk setiap pasangan kolom numerik dalam DataFrame dan histogram untuk distribusi masing-masing kolom pada diagonal. Pair plot sangat berguna untuk melihat hubungan antar kolom dalam DataFrame, terutama ketika kita memiliki beberapa kolom numerik. Scatter plot di bagian atas memberikan gambaran tentang korelasi antar kolom, sementara histogram diagonal memberikan informasi tentang distribusi masing-masing kolom.

Tahap selanjutnya adalah data visualization

Data visualisasi adalah representasi grafis dari informasi atau data yang dimaksudkan untuk memudahkan pemahaman, analisis, dan komunikasi konsep atau pola dalam dataset. Dengan menggunakan elemen-elemen visual seperti grafik, diagram, dan plot, data visualisasi mengubah data abstrak menjadi bentuk yang lebih tangibel, memungkinkan orang untuk melihat hubungan, distribusi, dan tren dalam data dengan lebih jelas. Tujuan utama dari data visualisasi adalah menyajikan informasi dengan cara yang dapat diinterpretasikan dengan cepat dan efektif, sehingga pemirsa dapat mengidentifikasi pola, anomali, atau wawasan penting dengan lebih baik.

```
import pandas as pd

# Buat DataFrame baru hanya untuk baris 'Europe & Central Asia'
new_df = df[df['Partner Name'] == 'Europe & Central
Asia'].reset_index(drop=True)

# Tampilkan DataFrame baru
new_df.head()
```

.reset_index(drop=True): Metode ini digunakan untuk mereset indeks DataFrame agar dimulai dari 0 dan menghapus indeks sebelumnya. Parameter drop=True menghapus kolom indeks sebelumnya yang akan dibuat sebagai kolom tambahan.

Hasilnya adalah DataFrame baru yang hanya berisi data yang terkait dengan wilayah 'Europe & Central Asia'

Kode ini membuat plot sekaligus mengecek banyaknya data yang terisi pada setiap tahun berupa blok diagram.

```
# Set ukuran plot
plt.figure(figsize=(15, 6))

# Buat count plot untuk kolom 'Year'
sns.countplot(x='Year', data=new_df,
order=sorted(new_df['Year'].unique()))

# Atur label sumbu x agar lebih terbaca
plt.xticks(rotation=90)

# Tentukan judul plot
plt.title('Count Plot untuk Kolom "Year"')

# Tampilkan plot
plt.show()
```

sns.countplot(x='Year', data=new_df, order=sorted(new_df['Year'].unique())): Fungsi countplot dari Seaborn digunakan untuk membuat count plot. x='Year' menunjukkan bahwa kita ingin memplot kolom 'Year', data=new_df adalah DataFrame yang akan digunakan, dan order=sorted(new_df['Year'].unique()) mengatur urutan kategori pada sumbu x.

Hasilnya adalah count plot yang menunjukkan jumlah observasi untuk setiap tahun pada kolom 'Year'.

```
# Hitung matriks korelasi
correlation_matrix = new_df.corr()

# Set ukuran plot
plt.figure(figsize=(12, 10))

# Buat heatmap untuk matriks korelasi
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=0.5)

# Atur judul plot
plt.title('Heatmap Korelasi DataFrame')

# Tampilkan plot
plt.show()
```

dimana sebelum membuat gambar pertama tama dihitung terlebih dahulu corelasinya tiap kolom menggunakan .corr() Matriks korelasi adalah tabel yang menunjukkan sejauh mana variabel-variabel numerik berkorelasi satu sama lain.

Setelah dapat dibuat diagram heatmapnya.

Dalam pembuatan boxplot. Kode ini

```
# Membuat fungsi untuk mengelompokkan tahun menjadi kategori abad
def categorize_century(year):
    if 1800 <= year < 1900:
        return 'Abad 19'
    elif 1900 <= year <= 2000:
        return 'Abad 20'
    else:
        return 'Lainnya'

# Menambahkan kolom baru 'Century' ke DataFrame
new_df['Century'] = new_df['Year'].apply(categorize_century)

# Pilih kolom yang ingin diplot pada sumbu y
selected_column_y = 'Export (US$ Thousand)'  # Ganti dengan kolom yang
sesuai

# Set ukuran plot
plt.figure(figsize=(12, 8))

# Buat boxplot dengan variabel kategorikal 'Century' pada sumbu x dan
variabel numerik pada sumbu y
sns.boxplot(x='Century', y=selected_column_y, data=new_df)

# Atur label sumbu x
```

```
plt.xlabel('Century')

# Tentukan judul plot
plt.title(f'Boxplot untuk {selected_column_y} di Europa dan Central
Asia')

# Tampilkan plot
plt.show()
```

Boxplot berisikan data tiap tahun terhadap nilai export. Dimana nilai export tersebut dibagi menjadi 2 bagian yaitu abad setelah tahun 2000 dan sebelumnya. Dimulai dari

categorize_century(year): Ini adalah fungsi yang digunakan untuk mengelompokkan tahun menjadi kategori abad.

new_df['Century'] = new_df['Year'].apply(categorize_century): Ini menambahkan kolom baru 'Century' ke DataFrame new_df dengan menggunakan fungsi categorize_century untuk mengelompokkan tahun.

sns.boxplot(x='Century', y=selected_column_y, data=new_df): Fungsi boxplot dari Seaborn digunakan untuk membuat boxplot. Variabel kategorikal 'Century' ditempatkan pada sumbu x, dan variabel numerik selected_column_y ditempatkan pada sumbu y.

output yang dihasilkan menggambarkan nilai max,min quartil atas dan bawah serta median dari masing masing baris dalam kolom Year terhadap nilai export.

```
# Pilih kolom yang ingin diplot pada sumbu y
selected_column_y = 'Export (US$ Thousand)'  # Ganti dengan kolom yang
sesuai

# Set ukuran plot
plt.figure(figsize=(12, 8))

# Buat violin plot dengan variabel kategorikal 'Century' pada sumbu x
dan variabel numerik pada sumbu y
sns.violinplot(x='Century', y=selected_column_y, data=new_df)

# Atur label sumbu x
plt.xlabel('Century')

# Tentukan judul plot
plt.title(f'Violin Plot untuk {selected_column_y} di Angola')

# Tampilkan plot
plt.show()
```

kode ini adalah kode untuk mempresentasikan grap violin. Violin plot digunakan untuk memvisualisasikan distribusi data numerik di dalam kategori atau kelompok-kelompok tertentu. Plot ini menggabungkan elemen-elemen dari plot densitas kernel dengan boxplot, sehingga memberikan gambaran lebih rinci tentang distribusi data.

sns.violinplot(x='Century', y=selected_column_y, data=new_df): Fungsi violinplot dari Seaborn digunakan untuk membuat violin plot. Variabel kategorikal 'Century' ditempatkan pada sumbu x, dan variabel numerik selected_column_y ditempatkan pada sumbu y.

Training regresi linear

Regresi linear adalah metode statistik yang digunakan untuk memodelkan hubungan linier antara satu atau lebih variabel independen (fitur) dengan variabel dependen (target). Model regresi linear mengasumsikan bahwa hubungan antara variabel-variabel tersebut dapat dijelaskan dengan persamaan garis lurus.

Pada tahap training regresi linear ini, mengregresikan features dan target dengan pemilihan kolomnya yaitu Year dan export.

```
# Pilih fitur (variabel independen) dan target (variabel dependen)
features = ['Year']  # Ganti dengan fitur-fitur yang sesuai
target = 'Export (US$ Thousand)'  # Ganti dengan target yang sesuai
```

Dalam syntax

```
model = LinearRegression()
```

menggunakan kelas LinearRegression() dari library scikit-learn untuk inisialisasi model regresi linear. Langkah ini merupakan bagian penting dalam proses pembuatan model, di mana kita menentukan jenis model yang akan digunakan untuk memahami dan memodelkan hubungan antara fitur dan target pada dataset. LinearRegression() adalah implementasi regresi linear yang sederhana di scikit-learn, dan model ini akan belajar untuk menyesuaikan garis lurus yang terbaik dengan data pelatihan.

Setelah inisialisasi, model akan dilatih menggunakan data pelatihan dengan menggunakan metode .fit(X_train, y_train), di mana X_train adalah fitur pelatihan dan y_train adalah target pelatihan. Model akan berusaha menyesuaikan koefisien dan intercept terbaik untuk menciptakan garis linear yang dapat memprediksi nilai target berdasarkan fitur yang diberikan.

Setelah data diproses dengan training dan testing kemudian data yang telah diolah tersebut divisualisasikan menggunakan plot scatter untuk membentuk regresi linear.

Evaluasi

Metrik evaluasi pada regresi linear, seperti RMSE (Root Mean Squared Error), R-squared (R2), dan lain-lain, digunakan untuk mengukur seberapa baik model regresi dapat memprediksi nilai target (variabel dependen) berdasarkan fitur-fitur yang ada.

Dalam tahap evaluasi ini digunakan kolom year untuk features dan kolom export untuk target yang nantinya akan dihitung akurasinya menggunakan metode MAE (mean absolute error), RMSE (root mean squared error), R2 (R-squared), dan MSE (mean squared error).

```
# Pisahkan data menjadi set pelatihan dan pengujian
```

```
X_train, X_test, y_train, y_test = train_test_split(new_df[features],
new_df[target], test_size=0.2, random_state=42)

# Inisialisasi model regresi linear
model = LinearRegression()
```

Dalam tahap persiapan data untuk pengembangan model regresi linear, kita pertama-tama memisahkan dataset menjadi dua bagian utama: set pelatihan dan set pengujian. Penggunaan fungsi train_test_split memastikan bahwa kita memiliki data yang tidak pernah dilihat oleh model selama proses pelatihan, yang esensial untuk mengukur seberapa baik model dapat menggeneralisasi pada data baru.

Setelah data terbagi, langkah selanjutnya adalah inisialisasi model regresi linear menggunakan kelas LinearRegression(). Inisialisasi model ini merupakan langkah awal dalam pengembangan model regresi, di mana kita memilih algoritma yang akan digunakan untuk memahami dan memodelkan hubungan antara variabel independen (fitur) dan variabel dependen (target) dalam dataset.

```
# Latih model pada set pelatihan
model.fit(X_train, y_train)

# Lakukan prediksi pada set pelatihan
y_train_pred = model.predict(X_train)

# Lakukan prediksi pada set pengujian
y_test_pred = model.predict(X_test)
```

Latih Model pada Set Pelatihan (model.fit(X_train, y_train)):

Dengan menggunakan fungsi fit, model regresi linear belajar dari data pelatihan untuk menyesuaikan parameter-parameter (koefisien dan intercept) sehingga garis regresi yang paling sesuai dapat ditemukan.

Proses ini melibatkan optimasi parameter-parameter tersebut untuk meminimalkan selisih antara nilai prediksi dan nilai aktual pada data pelatihan.

Lakukan Prediksi pada Set Pelatihan dan Pengujian:

Setelah model dilatih, kita dapat menggunakan metode .predict() untuk melakukan prediksi pada set pelatihan (y_train_pred) dan set pengujian (y_test_pred).

Hasil prediksi ini memberikan perkiraan nilai target yang dihasilkan oleh model berdasarkan fitur yang diberikan.

```
mse_train = mean_squared_error(y_train, y_train_pred)
r2_train = r2_score(y_train, y_train_pred)
mse_test = mean_squared_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)
```

Setelah melatih model regresi linear, langkah selanjutnya adalah mengevaluasi performa model pada kedua set data: pelatihan dan pengujian. Evaluasi ini memberikan pemahaman tentang seberapa baik model dapat memprediksi nilai target. Berikut adalah penjelasan mengenai metrik evaluasi yang digunakan:

Mean Absolute Error (MAE - Kesalahan Rata-Rata Mutlak):

mae_train mengukur rata-rata absolute selisih antara nilai prediksi dan nilai aktual pada set pelatihan dan pengujian.

Root Mean Squared Error (RMSE - Akar Kesalahan Kuadrat Rata-Rata):

rmse_train mengukur akar rata-rata dari kuadrat selisih antara nilai prediksi dan nilai aktual pada set pelatihan dan pengujian.

R-squared (R2 Score):

r2_train mengukur seberapa baik variabilitas dalam data target dapat dijelaskan oleh model pada set pelatihan dan pengujian.

Mean Squared Error (MSE - Kesalahan Kuadrat Rata-Rata):

mse_train mengukur rata-rata dari kuadrat selisih antara nilai prediksi dan nilai aktual pada set pelatihan dan pengujian.

Metrik evaluasi ini memberikan gambaran lengkap tentang seberapa baik model dapat memprediksi nilai target pada kedua set data. Semakin kecil nilai MAE, RMSE, dan MSE, serta semakin mendekati 1 nilai R2, menunjukkan performa model yang lebih baik. Evaluasi ini penting untuk memastikan bahwa model dapat menggeneralisasi dengan baik pada data yang tidak pernah dilihat sebelumnya (set pengujian).