PENJELASAN COODINGAN NAIVE BAYES :

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
```

PENJELASAN :

- import pandas as pd : Library untuk analisis dan manipulasi data berbasis tabel (DataFrame).
- import numpy as np : Library untuk komputasi numerik yang efisien, seperti operasi array dan aljabar linier.
- import matplotlib.pyplot as plt : Library untuk membuat visualisasi data, seperti grafik dan diagram.
- import seaborn as sns : Library untuk visualisasi data yang dibangun di atas Matplotlib, memberikan tampilan grafik yang lebih estetis.
- from sklearn.model_selection import train_test_split : Digunakan untuk membagi dataset menjadi data latih (training) dan data uji (testing).
- from sklearn.preprocessing import LabelEncoder : Digunakan untuk mengonversi data kategorikal (seperti teks) menjadi data numerik.
- from sklearn.naive_bayes import GaussianNB : Algoritma klasifikasi Naïve Bayes dengan asumsi distribusi data Gaussian (normal).
- from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix :
  - ccuracy_score: Menghitung akurasi (berapa persen prediksi benar).
  - precision_score: Menghitung presisi (berapa banyak prediksi positif yang benar dari semua prediksi positif).
  - recall_score: Menghitung sensitivitas/recall (berapa banyak kasus positif yang berhasil diprediksi dari semua kasus positif).
  - f1_score: Menggabungkan presisi dan recall dalam satu metrik.

- confusion_matrix: Menampilkan tabel yang membandingkan hasil prediksi dengan data aktual (true positive, false positive, true negative, false negative).

Load dataset

df = pd.read_csv("ecommerce_dataset.csv", sep=";")

df

PENJELASAN :

- pd.read_csv : Fungsi dari library pandas yang digunakan untuk membaca file dataset dalam format CSV (Comma-Separated Values).
- "ecommerce_dataset.csv" : Nama file dataset yang akan dibaca. File ini diasumsikan berada di direktori kerja saat ini (current working directory).
- sep=";" : Parameter sep digunakan untuk menentukan pembatas (delimiter) kolom dalam file CSV.
- df : Menampilkan isi DataFrame pada output.

Menghapus data yang tidak relevan

df.drop(columns=['User_ID', 'Product_ID'], inplace=True)

print("Kolom setelah penghapusan data yang tidak relevan:")

print(df.columns)

PENJELASAN :

- df.drop : untuk menghapus baris atau kolom tertentu dari DataFrame.
- columns=['User_ID', 'Product_ID'] : User_ID dan Product_ID adalah nama kolom dalam DataFrame yang dianggap tidak relevan atau tidak diperlukan untuk analisis, sehingga akan dihapus.
- inplace=True : Jika parameter inplace diset ke True, maka penghapusan dilakukan langsung pada DataFrame df, tanpa perlu membuat salinan baru dan Jika inplace=False (default), perubahan hanya diterapkan pada salinan DataFrame, dan variabel asli tetap tidak berubah.
- Print : Menampilkan pesan teks "Kolom setelah penghapusan data yang tidak relevan:" agar pengguna mengetahui hasil dari proses penghapusan kolom.
- df.columns : columns adalah atribut DataFrame yang menunjukkan daftar nama kolom yang tersisa setelah penghapusan. Dengan print(df.columns), semua nama kolom yang masih ada di DataFrame akan ditampilkan.

df

- PENJELASAN : df : Menampilkan isi DataFrame pada output.

Label Encoding untuk Payment_Method

le_payment = LabelEncoder()

df['Payment_Method'] = le_payment.fit_transform(df['Payment_Method'])

le_category = LabelEncoder()

df['Category'] = le_category.fit_transform(df['Category'])

print("Nilai unik dalam Payment_Method setelah encoding:")

print(df['Payment_Method'].unique())

print("Nilai unik dalam Category setelah encoding:")

print(df['Category'].unique())

PENJELASAN :

- le_payment = LabelEncoder() : Membuat objek dari LabelEncoder yang akan digunakan untuk mengonversi data kategorikal (teks) menjadi data numerik.
- df['Payment_Method'] : Merujuk pada kolom Payment_Method dalam DataFrame df, yang berisi metode pembayaran dalam bentuk kategorikal, seperti "Credit Card", "PayPal", "Cash", dll.
- le_payment.fit_transform(df['Payment_Method']) : fit_transform melakukan dua langkah sekaligus:
  - fit: Memetakan setiap nilai unik pada kolom Payment_Method ke angka.
  - transform: Mengonversi nilai pada kolom Payment_Method menjadi angka berdasarkan pemetaan tersebut.
  - Hasil encoding ini menggantikan nilai asli pada kolom Payment_Method.
- le_category = LabelEncoder() : Membuat objek LabelEncoder untuk kolom Category.
- df['Category'] : Merujuk pada kolom Category, yang berisi kategori produk seperti "Electronics", "Fashion", "Home", dll.
- le_category.fit_transform(df['Category']) : Sama seperti pada Payment_Method, langkah ini memetakan nilai unik pada kolom Category menjadi angka dan menggantinya dengan nilai numerik tersebut.
- Print : Menampilkan pesan teks ” Nilai unik dalam Payment_Method setelah encoding”

- df['Payment_Method'].unique() : Menampilkan nilai-nilai unik yang terdapat dalam kolom Payment_Method setelah encoding.
- Print : Menampilkan pesan teks ” Nilai unik dalam Category setelah encoding”
- df['Category'].unique() : Menampilkan nilai-nilai unik dalam kolom Category setelah encoding.

Feature Selection

selected_features = ['Category', 'Price (Rs.)', 'Discount (%)']

X = df[selected_features]

y = df['Payment_Method']

PENJELASAN :

- selected_features = ['Category', 'Price (Rs.)', 'Discount (%)'] : Variabel ini adalah sebuah daftar Python yang berisi nama-nama kolom yang akan digunakan sebagai fitur (independent variables) untuk memprediksi sesuatu. Untuk fitur yang dipilih adalah:
  - Category: Kategori produk (sudah diubah menjadi nilai numerik sebelumnya menggunakan LabelEncoder).
  - Price (Rs.): Harga produk dalam satuan Rupee.
  - Discount (%): Diskon yang diberikan dalam persentase.
- X: Variabel X adalah DataFrame baru yang hanya berisi kolom-kolom yang ada dalam daftar selected_features.
- df[selected_features]: Mengambil subset DataFrame df dengan kolom-kolom yang disebutkan dalam selected_features. Setelah baris ini dijalankan, variabel X akan memiliki data dari kolom Category, Price (Rs.), dan Discount (%).
- y: Variabel y adalah target (dependent variable) yang ingin diprediksi oleh model.
- df['Payment_Method'] : untuk mengakses kolom tertentu dalam DataFrame df menggunakan nama kolom Payment_Method.

Splitting Data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

PENJELASAN :

- ini membagi dataset menjadi data pelatihan (X_train, y_train) dan data pengujian (X_test, y_test).
- 80% data digunakan untuk melatih model, sementara 20% data digunakan untuk menguji performanya.
- Dengan random_state=42, pembagian dataset akan selalu konsisten, memungkinkan hasil yang dapat direproduksi.

Pembentukan Model Naïve Bayes

model = GaussianNB() :

model.fit(X_train, y_train)

PENJELASAN :

- Model : Variabel ini menyimpan model Gaussian Naive Bayes yang akan digunakan untuk melakukan pelatihan dan prediksi.
- GaussianNB() : adalah fungsi untuk membuat model Naive Bayes menggunakan Gaussian Naive Bayes dari library sklearn.naive_bayes.
- model.fit(X_train, y_train) : Metode fit digunakan untuk melatih model Naive Bayes menggunakan data pelatihan. Untuk input :
  - X_train: Data fitur pelatihan (misalnya: Category, Price (Rs.), dan Discount (%)).
  - y_train: Data target pelatihan (misalnya: Payment_Method).
  - hubungan antara fitur-fitur di X_train dan target di y_train, sehingga dapat digunakan untuk memprediksi nilai target berdasarkan data baru.

Analisis Akurasi Model

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred, average='weighted')

recall = recall_score(y_test, y_pred, average='weighted')

f1 = f1_score(y_test, y_pred, average='weighted')

print(f"Akurasi: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

print(f"F1-score: {f1:.2f}")

PENJELASAN :

- y_pred adalah array yang berisi prediksi kelas target untuk setiap baris data di X_test.
- model.predict(X_test): Model yang sudah dilatih sebelumnya digunakan untuk memprediksi nilai target (Payment_Method) berdasarkan fitur dari X_test (data pengujian).
- accuracy=accuracy_score(y_test, y_pred):Menghitung akurasi, yaitu persentase prediksi yang benar dibandingkan total prediksi.

- precision = precision_score(y_test, y_pred: Precision mengukur seberapa tepat model memprediksi kelas tertentu dibandingkan dengan semua prediksi untuk kelas tersebut.
- average='weighted': Precision dihitung untuk setiap kelas, lalu dirata-rata dengan mempertimbangkan proporsi jumlah data di setiap kelas.
- recall = recall_score(y_test, y_pred, average='weighted'): Recall mengukur seberapa baik model mendeteksi semua data positif sebenarnya untuk setiap kelas, Sama seperti precision, recall dihitung untuk setiap kelas, lalu dirata-rata secara tertimbang dengan average='weighted'.
- f1 = f1_score(y_test, y_pred, average='weighted'): F1-score adalah harmonic mean dari precision dan recall. F1-score memberikan keseimbangan antara precision dan recall.
- print(f"....") digunakan untuk menampilkan nilai hasil evaluasi model dalam format yang rapi.
- .2f memastikan hasil metrik ditampilkan dengan 2 angka desimal, membuatnya lebih mudah dibaca.
- Setiap metrik (accuracy, precision, recall, f1-score) memberikan gambaran yang berbeda tentang kinerja model.

Pengujian Model

comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

print("Perbandingan nilai aktual dan prediksi:")

print(comparison.head())

PENJELASAN :

- comparison digunakan untuk membuat tabel yang membandingkan nilai aktual (data sebenarnya dari data pengujian) dengan nilai prediksi (hasil prediksi model).
- pd.DataFrame(): Fungsi ini digunakan untuk membuat sebuah tabel (DataFrame) menggunakan library Pandas.
- DataFrame akan memiliki dua kolom:
  - Actual: Berisi nilai sebenarnya (y_test) dari data pengujian.
  - Predicted: Berisi nilai prediksi (y_pred) yang dihasilkan oleh model.
- {'Actual': y_test, 'Predicted': y_pred}: Membuat sebuah dictionary, di mana:
  - Kunci 'Actual' menyimpan nilai target aktual dari y_test.
  - Kunci 'Predicted' menyimpan nilai prediksi dari y_pred.
- print(“Perbandingan nilai aktual dan prediksi ”) : Menampilkan teks "Perbandingan nilai aktual dan prediksi:"
- comparison.head(): Fungsi head() digunakan untuk menampilkan 5 baris pertama dari DataFrame dan Jika DataFrame memiliki kurang dari 5 baris, semua baris akan ditampilkan.

```
# Visualisasi Confusion Matrix

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

cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le_payment.classes_,
yticklabels=le_payment.classes_)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()
```

PENJELASAN :

- plt.figure(figsize=(8, 6)): Fungsi plt.figure() dari matplotlib digunakan untuk membuat figure (plot area) yang akan digunakan untuk menampilkan grafik.
- figsize=(8, 6): Menentukan ukuran figure, di sini 8 adalah lebar (width) dan 6 adalah tinggi (height) dalam satuan inci. Ukuran ini membantu agar grafik terlihat lebih jelas dan mudah dibaca.
- cm=confusion_matrix(y_test, y_pred): Fungsi confusion_matrix() dari scikit-learn digunakan untuk menghitung confusion matrix antara nilai aktual (y_test) dan nilai prediksi (y_pred).
- sns.heatmap(): Fungsi ini digunakan untuk menampilkan heatmap dari confusion matrix menggunakan seaborn.
- cm adalah data yang akan digambarkan sebagai heatmap, yang dalam hal ini adalah confusion matrix yang sudah dihitung sebelumnya.
- annot=True: Menambahkan nilai pada setiap kotak di heatmap, sehingga nilai-nilai (seperti TP, TN, FP, FN) akan ditampilkan di dalam kotak.
- fmt='d': Menyatakan bahwa nilai yang ditampilkan di kotak akan diformat sebagai angka bulat (integer) ('d' untuk decimal).
- cmap='Blues': Menentukan skema warna untuk heatmap. Blues berarti warna akan diambil dari palet biru.
- xticklabels=le_payment.classes_: Menetapkan label sumbu X (kolom pada confusion matrix) berdasarkan kelas yang ada di objek le_payment.classes_.
- le_payment.classes_ adalah array yang berisi nama-nama kelas untuk variabel Payment_Method yang sudah di-encode sebelumnya menggunakan LabelEncoder.
- yticklabels=le_payment.classes_: Menetapkan label sumbu Y (baris pada confusion matrix) dengan cara yang sama seperti sumbu X.
- plt.xlabel("Predicted"): Menambahkan label pada sumbu X dengan teks "Predicted", yang menunjukkan bahwa nilai di sumbu X adalah nilai prediksi.
- plt.ylabel("Actual"): Menambahkan label pada sumbu Y dengan teks "Actual", yang menunjukkan bahwa nilai di sumbu Y adalah nilai aktual (sesungguhnya).

- plt.title("Confusion Matrix"): Menambahkan judul grafik dengan teks "Confusion Matrix".
- plt.show() : Fungsi ini digunakan untuk menampilkan grafik yang telah disiapkan (dalam hal ini, heatmap dari confusion matrix).

PENJELASAN COODINGAN REGRESI :

Sel 1: Import Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

Penjelasan:

- **pandas** dan **numpy**: Digunakan untuk manipulasi dan analisis data.
- **matplotlib.pyplot** dan **seaborn**: Untuk visualisasi data (grafik dan plot).
- **sklearn**: Digunakan untuk pemodelan:
    - **train_test_split**: Membagi data menjadi training dan testing.
    - **LabelEncoder**: Mengubah data kategori menjadi numerik.
    - **MinMaxScaler**: Normalisasi data.
    - **LinearRegression**: Membangun model regresi linear.
    - **mean_squared_error, r2_score, mean_absolute_error**: Evaluasi kinerja model.

Sel 2: Membaca Dataset

```
# Load dataset
df = pd.read_csv("ecommerce_dataset.csv", sep=";")
df
```

Penjelasan:

- Dataset dimuat dari file CSV bernama **ecommerce_dataset.csv**, dengan delimiter ;.

- **df**: Objek DataFrame yang memuat data tersebut.

Sel 3: Menghapus Nilai yang Hilang

```
# Menghapus nilai yang hilang
df.dropna(inplace=True)
print("Jumlah nilai yang hilang setelah dihapus:")
print(df.isnull().sum())
```

Penjelasan:

- **dropna**: Menghapus baris yang memiliki nilai kosong/missing.
- **isnull().sum()**: Menampilkan jumlah nilai yang kosong di setiap kolom setelah penghapusan.

Sel 4: Konversi Format Tanggal

```
# Mengubah format tanggal
df['Purchase_Date'] = pd.to_datetime(df['Purchase_Date'], format='%d/%m/%Y')
print("Format tanggal setelah konversi:")
print(df['Purchase_Date'].head())
```

Penjelasan:

- **pd.to_datetime**: Mengonversi kolom **Purchase_Date** menjadi format datetime (%d/%m/%Y berarti hari/bulan/tahun).
- Menampilkan lima data pertama dari kolom yang telah dikonversi.

Sel 5: Konversi Kategori ke Numerik

```
# Mengubah kategori menjadi numerik
label_encoders = {}
categorical_columns = ['Category', 'Payment_Method']
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

```
    label_encoders[col] = le
print("Kategori setelah dikonversi:")
print(df[categorical_columns].head())
```

Penjelasan:

- Mengonversi kolom kategori (**Category** dan **Payment_Method**) menjadi numerik menggunakan **LabelEncoder**.
- **fit_transform**: Menyesuaikan dan mengubah data secara bersamaan.
- **label_encoders**: Menyimpan objek encoder untuk setiap kolom jika diperlukan untuk decoding kembali.

Sel 6: Normalisasi Data

```
# Normalisasi Data
scaler = MinMaxScaler()
df[['Price (Rs.)', 'Final_Price(Rs.)']] = scaler.fit_transform(df[['Price (Rs.)', 'Final_Price(Rs.)']])
print("Data harga setelah normalisasi:")
print(df[['Price (Rs.)', 'Final_Price(Rs.)']].head())
```

Penjelasan:

- **MinMaxScaler**: Menskalakan nilai kolom **Price (Rs.)** dan **Final_Price(Rs.)** ke rentang [0, 1].
- **fit_transform**: Menyesuaikan dan mengubah data secara bersamaan.
- Menampilkan lima data pertama setelah normalisasi.

Sel 7: Deteksi Outlier

```
# Deteksi Outlier menggunakan Boxplot
plt.figure(figsize=(10, 5))
sns.boxplot(data=df[['Price (Rs.)', 'Discount (%)']])
plt.title("Boxplot Harga dan Diskon")
plt.show()
```

Penjelasan:

- Membuat boxplot untuk mendeteksi outlier pada kolom **Price (Rs.)** dan **Discount (%)**.
- **sns.boxplot**: Membuat boxplot menggunakan Seaborn.

Sel 8: Membangun Model Regresi Linear

```
# Pembentukan Model Regresi Linear
X = df[['Price (Rs.)', 'Discount (%)']]
y = df['Final_Price(Rs.)']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
```

Penjelasan:

- **X**: Fitur (Harga awal dan diskon).
- **y**: Target (Harga akhir).
- **train_test_split**: Membagi data menjadi 80% training dan 20% testing.
- **LinearRegression**: Membuat dan melatih model regresi linear menggunakan data training.

Sel 9: Evaluasi Kinerja Model

```
# Analisis Akurasi Model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r2}")
print(f"Mean Absolute Error (MAE): {mae}")
```

Penjelasan:

- **predict**: Memprediksi harga berdasarkan data testing.
- Menghitung metrik evaluasi:
    - **MSE**: Rata-rata kesalahan kuadrat.
    - **R^2**: Koefisien determinasi.
    - **MAE**: Rata-rata kesalahan absolut.

Sel 10: Perbandingan Nilai Aktual dan Prediksi

```
# Pengujian Model
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print("Perbandingan nilai aktual dan prediksi:")
print(comparison.head())
```

Penjelasan:

- Membandingkan nilai aktual dan prediksi dalam DataFrame baru.
- Menampilkan lima data pertama dari hasil perbandingan.

Sel 11: Visualisasi Model

```
# Visualisasi Model
plt.figure(figsize=(10, 5))
sns.scatterplot(x=X_test['Price (Rs.)'], y=y_test, label='Actual')
sns.scatterplot(x=X_test['Price (Rs.)'], y=y_pred, label='Predicted')
plt.title("Scatter Plot Harga Awal vs Harga Akhir")
plt.xlabel("Price (Rs.)")
plt.ylabel("Final Price (Rs.)")
plt.legend()
plt.show()
```

Penjelasan:

- Membuat scatter plot untuk membandingkan harga aktual dan prediksi berdasarkan harga awal.

Sel 12: Residual Plot

```
# Residual Plot
plt.figure(figsize=(10, 5))
sns.residplot(x=y_pred, y=y_test - y_pred, lowess=True, color='blue')
plt.title("Residual Plot")
plt.xlabel("Predicted Value")
plt.ylabel("Residuals")
plt.show()
```

Penjelasan:

- Membuat residual plot untuk memvisualisasikan selisih antara nilai aktual dan prediksi.
- **Residuals**: Nilai aktual - nilai prediksi.

Kesimpulan

Notebook ini berhasil menunjukkan proses pembangunan model regresi linear untuk memprediksi harga akhir (Final_Price(Rs.)) berdasarkan harga awal (Price (Rs.)) dan diskon (Discount (%)). Langkah awal dimulai dengan membersihkan data dari nilai yang hilang dan mengonversi format tanggal menjadi lebih sesuai.

Selanjutnya, data kategori seperti Category dan Payment_Method diubah menjadi bentuk numerik menggunakan label encoding, sementara kolom numerik dinormalisasi ke dalam rentang [0,1] menggunakan Min-Max Scaling.

Model dibangun menggunakan data yang telah dibagi menjadi set pelatihan dan pengujian, dengan hasil evaluasi menunjukkan metrik yang memadai, seperti Mean Squared Error (MSE), R-squared (R^2), dan Mean Absolute Error (MAE).

Visualisasi scatter plot dan residual plot mengindikasikan bahwa model mampu menangkap pola hubungan antara harga awal, diskon, dan harga akhir dengan cukup baik.

Meskipun demikian, deteksi outlier dalam data menyoroti perlunya perhatian lebih untuk meningkatkan akurasi model di masa depan. Secara keseluruhan, model ini dapat menjadi dasar yang kuat untuk prediksi harga dalam aplikasi e-commerce atau studi serupa.