1. Preprocessing

```
import pandas as pd
from google.colab import files
uploaded = files.upload()

# Baca data
df = pd.read_csv('data_pelanggan.csv')
```

```
<IPython.core.display.HTML object>

Saving data_pelanggan.csv to data_pelanggan (8).csv
```

a. Tampilkan ringkasan statistik

```
import pandas as pd

# Load data
df = pd.read_csv('data_pelanggan.csv')

# Ringkasan statistik
print(df.describe())

# Cek missing value
print(df.isnull().sum())

# Cek tipe data
print(df.dtypes)
```

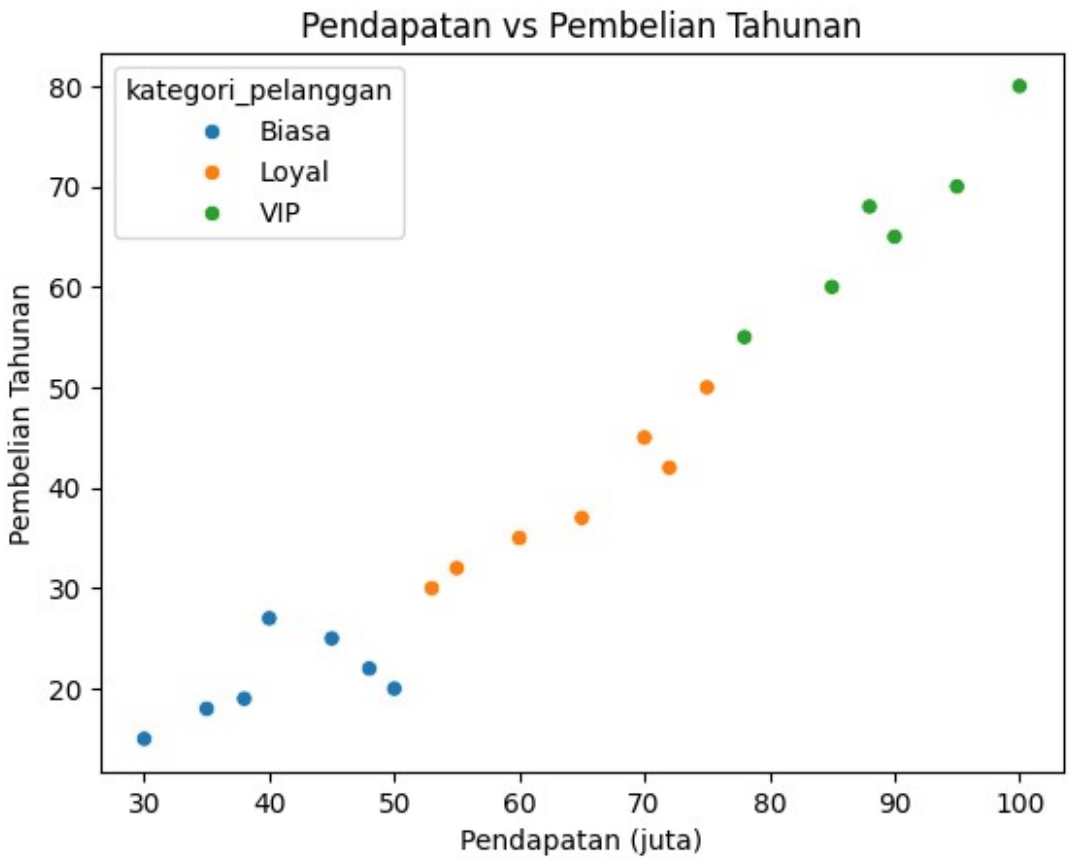
```
       id_pelanggan       umur  pendapatan  pembelian_tahunan
count      20.00000  20.000000   20.000000          20.000000
mean       10.50000  34.100000   63.600000          40.750000
std         5.91608   7.114847   21.330482          19.891482
min         1.00000  23.000000   30.000000          15.000000
25%         5.75000  28.750000   47.250000          24.250000
50%        10.50000  33.500000   62.500000          36.000000
75%        15.25000  38.500000   79.750000          56.250000
max        20.00000  50.000000  100.000000          80.000000
id_pelanggan          0
nama                  0
umur                  0
pendapatan            0
pembelian_tahunan     0
kategori_pelanggan    0
dtype: int64
id_pelanggan           int64
nama                  object
umur                   int64
pendapatan             int64
pembelian_tahunan      int64
```

```
kategori_pelanggan    object
dtype: object
```

b. Visualisasikan hubungan antara pendapatan dan pembelian_tahunan.

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

# Load data
df = pd.read_csv('data_pelanggan.csv')

# Check if 'pendapatan' column exists
print(df.columns)  # Print all column names to check for 'pendapatan'

# If 'pendapatan' is not found, check for possible typos and correct
them
# For example, if the column is named 'Pendapatan', rename it:
# df = df.rename(columns={'Pendapatan': 'pendapatan'})

# Create the scatter plot
sns.scatterplot(data=df, x='pendapatan', y='pembelian_tahunan',
hue='kategori_pelanggan')
plt.title('Pendapatan vs Pembelian Tahunan')
plt.xlabel('Pendapatan (juta)')
plt.ylabel('Pembelian Tahunan')
plt.show()
```

```
Index(['id_pelanggan', 'nama', 'umur', 'pendapatan',
'pembelian_tahunan',
       'kategori_pelanggan'],
      dtype='object')
```



1. Implementasi Algoritma

Preprocessing untuk klasifikasi

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load data
df = pd.read_csv('data_pelanggan.csv')

# Tampilkan semua nama kolom
print("Kolom yang tersedia:", df.columns)

# Pastikan nama kolom seragam (huruf kecil semua, tanpa spasi)
df.columns = df.columns.str.strip().str.lower()

# Label encoding untuk target
le = LabelEncoder()
df['kategori_pelanggan'] = le.fit_transform(df['kategori_pelanggan'])

# Pisahkan fitur dan label
fitur = ['umur', 'pendapatan', 'pembelian_tahunan']
X = df[fitur]
```

```
y = df['kategori_pelanggan']

# Standarisasi fitur
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split dataset: 80% train, 20% test
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
```

```
Kolom yang tersedia: Index(['id_pelanggan', 'nama', 'umur',
'pendapatan', 'pembelian_tahunan',
       'kategori_pelanggan'],
      dtype='object')
```

a. Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score

tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)

print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_tree))
print(classification_report(y_test, y_pred_tree))
```

```
Decision Tree Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         2

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4
```

b. K-Nearest Neighbor (KNN)

```
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)

print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
```

```
KNN Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         2

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4
```

1. Analisis Perbandingan

a. Model mana yang lebih akurat?

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score

tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
y_pred_dt = tree_model.predict(X_test) # Assign the predictions to
y_pred_dt

print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))

print("Akurasi Decision Tree:", accuracy_score(y_test, y_pred_dt))
print("Akurasi KNN:", accuracy_score(y_test, y_pred_knn))
```

```
Decision Tree Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         2

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4

Akurasi Decision Tree: 1.0
Akurasi KNN: 1.0
```

Jawaban : Jika dilihat dari akurasinya kedua jenis model klasifikasi tersebut memiliki tingkat akurasi yang sama yaitu 1.00 baik Decision Tree maupun KNN.

b. Apakah hasil klasifikasi konsisten?

```
import numpy as np

consistency = np.mean(y_pred_dt == y_pred_knn)
```

```
print("Konsistensi antara Decision Tree dan KNN:", consistency)
```

```
Konsistensi antara Decision Tree dan KNN: 1.0
```

Jawaban : dilihat dari hasil run kode diatas antara Decision Tree dan KNN memiliki konsistensi yang sama yaitu 1.0.

1. Kesimpulan

Decision Tree menunjukkan akurasi sebesar 1.00 dan juga KNN menunjukkan akurasi sebesar 1.00. Jika Decision Tree lebih akurat, maka model ini lebih cocok untuk dataset ini yang kemungkinan memiliki pola keputusan yang eksplisit. Keunggulan Decision Tree adalah interpretabilitas yang tinggi dan tidak memerlukan scaling data. Kelemahannya adalah rawan overfitting. sedangkan Kelebihan KNN adalah sederhana dan efektif untuk data yang memiliki distribusi yang seimbang, tetapi kelemahannya adalah sensitif terhadap outlier dan fitur dengan skala besar. Namun karena untuk dataset saat ini kedua klasifikasi tersebut menunjukkan akurasi yang sama maka kita bisa menyesuaikan untuk penggunaannya. Kemudian jika ada perbedaan akurasi dan konsistensi maka dipengaruhi oleh kompleksitas model, ukuran data, distribusi kelas target, serta hubungan antara fitur. Maka dapat disimpulkan bahwa penggunaan dari kedua jenis model klasifikasi tersebut dapat digunakan sesuai dengan kebutuhannya dan harus dicocokkan kembali dengan dataset yang kita miliki.