Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Exploratory Data Analysis (EDA)
Pokok Pemb : - Mengenal Library ML
- Mengenal Statistik Deskriptif
- Mengenal EDA Data Numerik
- Mengenal EDA Data Kategori
- Mengenal EDA data Multivariabel
- Mengenal Univariate Analysis
- Mengenal Bivariate Analysis

1. Mengenal Library Machine Learning

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

```
df = pd.read_csv("adult.data.csv")
```

```
df.head()
```

```
df.info()
```

2. Mengenal Statistik Deskriptif

```
# Menghitung statistik deskriptif untuk kolom numerik
numerical_stats = df.describe()
print("Statistik Deskriptif untuk Kolom Numerik:\n", numerical_stats)
```

```
# Menghitung frekuensi untuk kolom kategorikal
categorical_stats = df.select_dtypes(include=['category']).apply(lambda x: x.value_counts(normalize=True) * 100)
print("\nPersentase Frekuensi untuk Kolom Kategorikal:\n", categorical_stats)
```

```
# Menampilkan jumlah missing values pada tiap kolom
missing_values = df.isnull().sum()
print("\nJumlah Missing Values pada tiap kolom:\n", missing_values)
```

3. Mengenal Bentuk EDA dengan Data Numerik

```
# Histogram
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], kde=True)
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

```
# Box Plot
plt.figure(figsize=(8, 6))
sns.boxplot(x='sex', y='hours-per-week', data=df)
plt.title('Box Plot of Hours per Week by Sex')
plt.xlabel('Sex')
plt.ylabel('Hours per Week')
plt.show()
```

```
# Scatter Plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='capital-gain', data=df)
plt.title('Scatter Plot of Age vs Capital Gain')
plt.xlabel('Age')
plt.ylabel('Capital Gain')
plt.show()
```

4. Mengenal Bentuk EDA dengan Data Kategori

```
# Line Plot (Example: Average capital gain by age)
avg_capital_gain_by_age = df.groupby('age')['capital-gain'].mean()
plt.figure(figsize=(8, 6))
plt.plot(avg_capital_gain_by_age.index, avg_capital_gain_by_age.values)
plt.title('Line Plot of Average Capital Gain by Age')
plt.xlabel('Age')
plt.ylabel('Average Capital Gain')
plt.show()
```

```
# Bar Chart
# Contoh: Melihat jumlah orang berdasarkan tingkat pendidikan
education_counts = df['education'].value_counts()
plt.figure(figsize=(10, 5))
plt.bar(education_counts.index, education_counts.values)
plt.xlabel('Tingkat Pendidikan')
plt.ylabel('Jumlah Orang')
plt.title('Jumlah Orang Berdasarkan Tingkat Pendidikan')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
# Pie Chart
# Contoh: Melihat persentase jenis kelamin
sex_counts = df['sex'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(sex_counts.values, labels=sex_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Persentase Jenis Kelamin')
plt.show()
```

```
# Donut Chart
# Contoh: Melihat persentase ras
race_counts = df['race'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(race_counts.values, labels=race_counts.index, autopct='%1.1f%%', startangle=90, wedgeprops=dict(width=0.4))
plt.title('Persentase Ras')
plt.show()
```

5. Mengenal Bentuk EDA dengan Data Multivariabel

```
# Create a heatmap
# Select numerical features for the heatmap
numerical_features = ['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']
heatmap_data = df[numerical_features].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
# Create a pair plot
sns.pairplot(df[['age', 'education-num', 'hours-per-week', 'capital-gain', 'capital-loss']])
plt.show()
```

```
# Create a bubble chart
# Select features for the bubble chart
bubble_data = df[['age', 'hours-per-week', 'capital-gain']]

plt.figure(figsize=(10, 8))
plt.scatter(x='age', y='hours-per-week', s='capital-gain', data=bubble_data, alpha=0.5)
plt.xlabel('Age')
plt.ylabel('Hours per Week')
plt.title('Bubble Chart: Age, Hours per Week, and Capital Gain')
plt.show()
```

6. Mengenal Univarate Analysis

```
# Univariate Analysis for Age
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

```python
# Univariate Analysis for Education (Categorical)
plt.figure(figsize=(10, 6))
sns.countplot(x='education', data=df)
plt.title('Count of Individuals by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```python
# Univariate Analysis for Salary (Categorical)
plt.figure(figsize=(6, 6))
sns.countplot(x='salary', data=df)
plt.title('Count of Individuals by Salary')
plt.xlabel('Salary')
plt.ylabel('Count')
plt.show()
```

7. Mengenal Bivariate Analysis

```python
# 1. Age vs. Salary
# Deskripsi: Melihat hubungan antara umur dan tingkat pendapatan.
# Kita mengharapkan bahwa orang yang lebih tua mungkin memiliki pendapatan yang lebih tinggi.
plt.figure(figsize=(8, 6))
sns.boxplot(x='salary', y='age', data=df)
plt.title('Hubungan antara Umur dan Tingkat Pendapatan')
plt.show()
```

```python
# 2. Education vs. Salary
# Deskripsi: Melihat hubungan antara tingkat pendidikan dan pendapatan.
# Kita mengharapkan bahwa orang dengan pendidikan lebih tinggi cenderung memiliki pendapatan lebih tinggi.
plt.figure(figsize=(12, 6))
sns.countplot(x='education', hue='salary', data=df)
plt.title('Hubungan antara Tingkat Pendidikan dan Tingkat Pendapatan')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
# 3. Hours-per-week vs. Salary
# Deskripsi: Melihat hubungan antara jumlah jam kerja per minggu dan pendapatan.
# Kita mengharapkan bahwa orang yang bekerja lebih banyak jam cenderung memiliki pendapatan lebih tinggi.
plt.figure(figsize=(8, 6))
sns.boxplot(x='salary', y='hours-per-week', data=df)
plt.title('Hubungan antara Jam Kerja per Minggu dan Tingkat Pendapatan')
plt.show()
```

```
# 4. Sex vs. Salary
# Deskripsi: Melihat perbedaan pendapatan antara pria dan wanita.
plt.figure(figsize=(8, 6))
sns.countplot(x='sex', hue='salary', data=df)
plt.title('Perbedaan Tingkat Pendapatan Berdasarkan Jenis Kelamin')
plt.show()
```

```
# 5. Occupation vs. Salary
# Deskripsi: Melihat hubungan antara jenis pekerjaan dan pendapatan.
# Kita mengharapkan bahwa beberapa jenis pekerjaan memiliki pendapatan yang lebih tinggi daripada yang lain.
plt.figure(figsize=(12, 6))
sns.countplot(x='occupation', hue='salary', data=df)
plt.title('Hubungan antara Jenis Pekerjaan dan Tingkat Pendapatan')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
# 6. Correlation Matrix
# Deskripsi: Melihat korelasi antara variabel-variabel numerik.
# Kita dapat melihat variabel mana yang memiliki korelasi kuat dengan variabel lainnya.
numeric_cols = df.select_dtypes(include=['number']).columns
correlation_matrix = df[numeric_cols].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

Latihan

1. Tampilkan Plot Education berdasarkan Salarynya
2. Tampilkan jumlah pendidikan Bachelors dengan gaji <=50k
3. Tampilkan data occupation dengan gaji diatas 50k
4. Tampilkan Plot occupation dengan gaji diatas 50k
5. Tampilkan rata-rata hours per week untuk occupation sales?
6. Tampilkan rata-rata hours per week untuk occupation Prof-Speciality?

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Simple Linear Regresion
Pokok Pemb : - Mengenal Regresi Linier
- Membangun Model Regresi Linier
- Simulasi Algoritma Regresi Linier
- Evaluasi Algoritma Regresi Linier

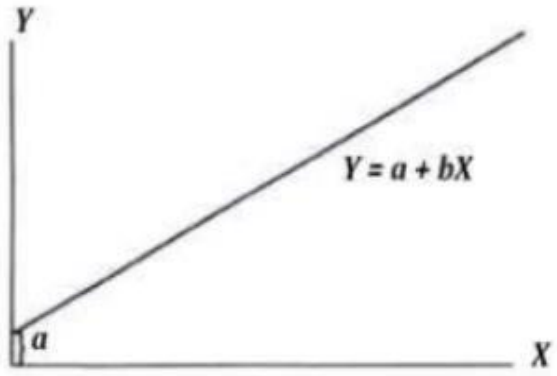
1. Mengenal Regresi Linier

a. Konsep Regresi Linier

Regresi linier berusaha menemukan garis lurus yang terbaik (disebut line of best fit) untuk mendekati data dan memprediksi nilai target. Garis ini dinyatakan dengan persamaan:

$$Y = a + bX$$

Y = Varibel Response atau Variabel Akibat (Dependent)
X = Variabel Predictor atau Variabel Faktor Penyebab (Independent)
a = Konstanta / intercept
b = Koefisien regresi (kemiringan/slope); besaran response yang ditimbulkan oleh predictor



b. Langkah-langkah regresi linier

- Membuat dataset

Nama	Kalori/hari(X)	Berat Badan (Y)
Adi	530	89
Rudi	300	48
Didi	358	56
Budi	510	72
Intan	302	54
Putu	300	42
Parta	387	60
Esti	527	85
Wike	415	63
Elis	512	74

- Menghitung nilai variabel

Nama	Kalori/ hari(X)	X2	Berat Badan (Y)	Y2	XY
Adi	530	280900	89	7921	47170
Rudi	300	90000	48	2304	14400
Didi	358	128164	56	3136	20048
Budi	510	260100	72	5184	36720
Intan	302	91204	54	2916	16308
Putu	300	90000	42	1764	12600
Parta	387	149769	60	3600	23220
Esti	527	277729	85	7225	44795
Wike	415	172225	63	3969	26145
Elis	512	262144	74	5476	37888
Σ	**4141**	**1802235**	**643**	**43495**	**279294**

- Perhitungan koefisien a dan b

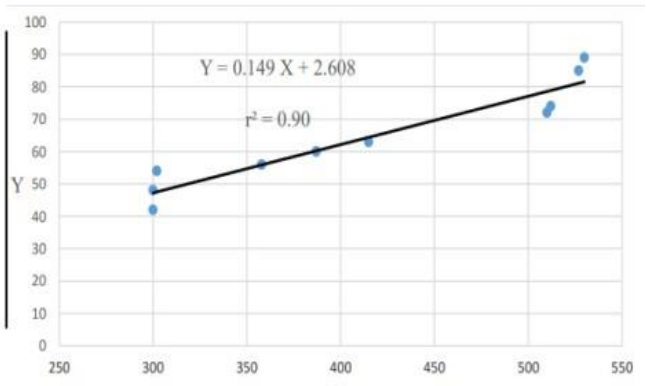
$$a = \frac{(\sum Y_i)(\sum X_i^2) - (\sum X_i)(\sum X_i Y_i)}{n\sum X_i^2 - (\sum X_i)^2}$$

$$= \frac{(643)(1802235) - (4141)(279294)}{10(1802235) - (4141)^2} = \frac{2280651}{874469} \cong 2{,}608$$

$$b = \frac{n(\sum X_i Y_i) - (\sum X_i)(\sum Y_i)}{n\sum X_i^2 - (\sum X_i)^2}$$

$$= \frac{10(279294) - (4141)(643)}{10(1802235) - (4141)^2} = \frac{130227}{874469} \cong 0{,}14892 \approx 0{,}149$$

- Persamaan garis regresi



Setelah didapat koefisien a dan b, maka persamaan garisnya adalah **Y = 2,608 + 0.149X**

- Hitung estimasi
  Melakukan Estimasi terhadap variable predictor atau response
  Estimasi berat badan mahasiswa jika **asupan kalori** adalah **600 kalori/hari :**

Y = 2,608 + 0,149 X

Y = 2,608 + (0,149 * 600)

Estimasi Y = **92 Kilogram**

Estimasi asupan kalori mahasiswa, jika **berat badan mahasiwa** adalah **40 kilogram :**

40 = 2,608 + 0,149 X

40 – 2,608 = 0,149 X

37, 392 = 0,149 X

Estimasi X = **250.59 kalori/hari**

2. Membangun model regresi linier

- Import Library yang dibutuhkan

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error, r2_score
5 import matplotlib.pyplot as plt
6 import seaborn as sns
```

- Deskripsi Dataset

```
1 df = pd.read_csv('konsumsi-bbm.csv')
```

```
1 df.head()
```

```
1 df.info()
```

```
1 df.describe()
```

- Exploratory Data Analysis

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Jarak_Tempuh', y='Konsumsi_BBM', data=df)
plt.title('Hubungan Jarak Tempuh dan Konsumsi BBM')
plt.xlabel('Jarak Tempuh')
plt.ylabel('Konsumsi BBM')
plt.show()
```

```
plt.figure(figsize=(8, 6))
sns.boxplot(data=df[['Jarak_Tempuh', 'Konsumsi_BBM']])
plt.title('Box Plot Jarak Tempuh dan Konsumsi BBM')
plt.show()
```

```
plt.figure(figsize=(8, 6))
sns.histplot(df['Jarak_Tempuh'], kde=True)
plt.title('Distribusi jarak tempuh')
plt.xlabel('jarak tempuh')
plt.ylabel('Frekuensi')
plt.show()
```

- Membangun Model Regresi

```
X = df[['Jarak_Tempuh']]
y = df['Konsumsi_BBM']
```

```
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)
```

```
y_pred = model.predict(X_test)
```

3. Simulasi algoritma regresi linier

```
jarak_baru = [60, 70, 80]  # Contoh jarak tempuh baru
konsumsi_prediksi = model.predict([[jarak] for jarak in jarak_baru])
```

```
print('\nSimulasi Konsumsi BBM:')
for i in range(len(jarak_baru)):
    print(f'Jarak Tempuh: {jarak_baru[i]}, Konsumsi BBM Prediksi: {konsumsi_prediksi[i]}')
```

```
plt.scatter(X_train, y_train, color='blue', label='Data Pelatihan')
plt.plot(X_train, model.predict(X_train), color='red', label='Garis Regresi')
plt.scatter(jarak_baru, konsumsi_prediksi, color='green', label='Data Baru (Prediksi)')

plt.xlabel('jarak')
plt.ylabel('konsumsi bbm')
plt.title('Grafik Regresi Linier')
plt.legend()
plt.show()
```

4. Evaluasi algoritma regresi linier

```
# Menghitung MSE dan R-squared
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Simple Linear Regresion
Pokok Pemb : - Regresi Linier Berganda
- Membangun Model Regresi Linier Berganda
- Simulasi Algoritma Regresi Linier Berganda
- Evaluasi Algoritma Regresi Linier Berganda

1. Regresi Linier Berganda

a. Konsep Regresi Linier Berganda

Regresi Linier Berganda adalah salah satu teknik statistik yang digunakan untuk memodelkan hubungan antara satu variabel dependen (output/label) dan lebih dari satu variabel independen (input/fitur). Regresi linier berganda memperluas konsep regresi linier sederhana, yang hanya melibatkan satu variabel independen, dengan memasukkan beberapa variabel independen dalam analisis.

Tujuan utama dari regresi linier berganda adalah untuk menemukan hubungan linear antara variabel-variabel independen dan variabel dependen, sehingga kita dapat memprediksi nilai variabel dependen berdasarkan nilai-nilai variabel independen.

b. Aplikasi Regresi Linier Berganda

Regresi linier berganda sering digunakan dalam berbagai bidang seperti:

- **Ekonomi**: untuk memprediksi pendapatan berdasarkan berbagai faktor seperti tingkat pendidikan, pengalaman kerja, dan usia.
- **Pemasaran**: untuk menganalisis dampak harga, iklan, dan lokasi terhadap penjualan produk.
- **Kesehatan**: untuk memprediksi jumlah kalori yang dibakar berdasarkan usia, berat badan, tinggi badan, dan durasi olahraga.

c. Rumus Regresi Linier Berganda

Secara matematis, persamaan regresi linier berganda dapat dinyatakan sebagai berikut:

$$\mathbf{Y = a + b_1 X_1 + b_2 X_2 + b_3 X_3 + \ldots b_n X_n}$$

Di mana:

- Y = adalah variabel dependen (yang ingin diprediksi).
- $X_1, X_2, X_3, \ldots, X_n$ = adalah variabel independen (faktor yang memengaruhi variabel dependen).
- a = adalah konstanta atau intercept, yaitu nilai YYY saat semua variabel independen bernilai nol.
- $b_1, b_2, b_3, \ldots, b_n$ adalah koefisien regresi, yang menunjukkan seberapa besar pengaruh setiap variabel independen terhadap variabel dependen.

2. Membangun Model Regresi Linier Berganda

```python
import pandas as pd
import numpy as np

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

```python
df = pd.read_csv('kalori_ganda.csv')
```

```python
# 1. Scatter plot untuk melihat hubungan antara Umur dan Kalori
plt.figure(figsize=(7,5))
plt.scatter(x='Umur', y='Kalori', data=df, color='blue')
plt.title('Age vs Calorie Intake')
plt.xlabel('Age (years)')
plt.ylabel('Calorie Intake (kcal)')
plt.grid(True)
plt.show()
```

```
# 2. Berat Badan vs Konsumsi Kalori
plt.figure(figsize=(7,5))
plt.scatter(x='BB', y='Kalori', data=df, color='green')
plt.title('Weight vs Calorie Intake')
plt.xlabel('Weight (kg)')
plt.ylabel('Calorie Intake (kcal)')
plt.grid(True)
plt.show()
```

```
# Histogram untuk melihat distribusi BB
plt.figure(figsize=(8, 6))
sns.histplot(x='BB', data=df)
plt.title('Distribusi BB')
plt.show()
```

```
# menampilkan plot animasi 3d
fig = px.scatter_3d(df, x='Umur', y='BB', z='TB',
                    color='Olahraga', size='Kalori',
                    hover_name='Umur')
fig.show()
```

3. Simulasi Algoritma Regresi Linier Berganda

```
# Memisahkan fitur (X) dan target (y)
x = df[['Umur','BB','TB','Olahraga']]
y = df['Kalori']
```

```
# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42)
```

```
# Membuat model regresi linier
model = LinearRegression()
model.fit(x, y)
```

```
# Menampilkan intercept regresi dari model
intercept = model.intercept_
intercept
```

```python
# Menampilkan koefisien regresi dari model
coefficients = model.coef_
coefficients
```

```
manual = intercept+(25*3.05138649)+(75*22.34269713)+(170*0.0466017)+(60*0.06622248)
manual
```

```python
# Data input untuk prediksi
input_data = np.array([[25, 75, 170, 60]])
```

```python
# Melakukan prediksi
predicted_calories = model.predict(input_data)
predicted_calories
```

4. Evaluasi Regresi Linier Berganda

```
# Memprediksi nilai BB dengan data uji
y_pred = model.predict(X_test)
```

```
# Menghitung MSE dan R2 score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
# Menampilkan MSE dan R2 score
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

5. Simpan Model

```
# Menyimpan model menggunakan pickle
import pickle

filename = 'regression_model.pkl'
pickle.dump(model, open(filename, 'wb'))
```

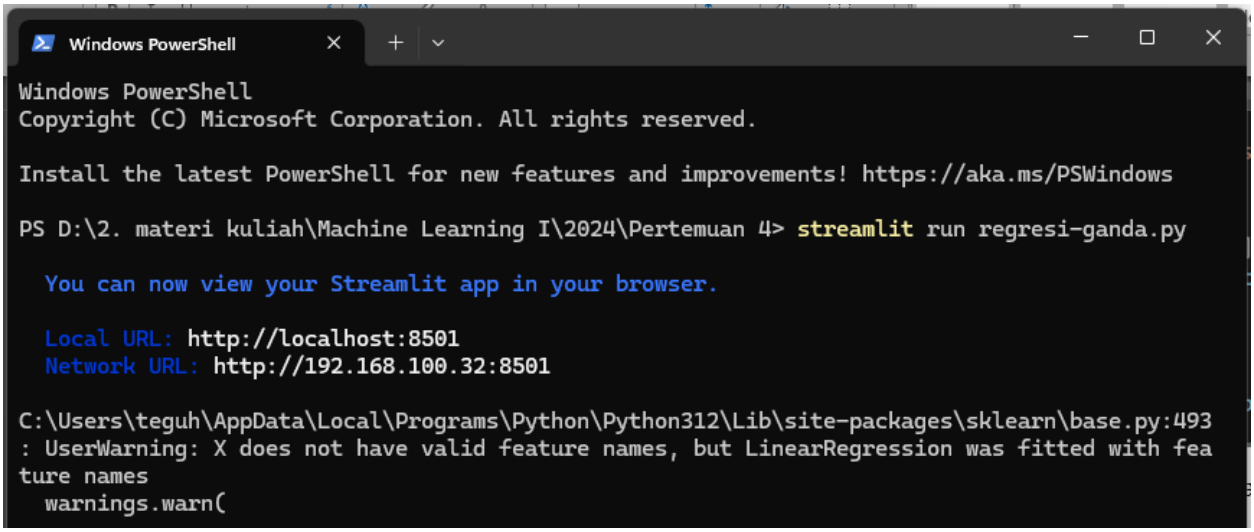
Dari perintah ini akan menghasilkan file baru yaitu **regression_model.pkl**. file ini akan otomatis muncul di direktori project kalian. Apabila berhasil, file model ini bisa kita gunakan untuk membuat aplikasi machine learning berbasis web menggunakan streamlit.

6. Membuat Aplikasi

Buat file **regresi-ganda.py** pada direktori yang sama.

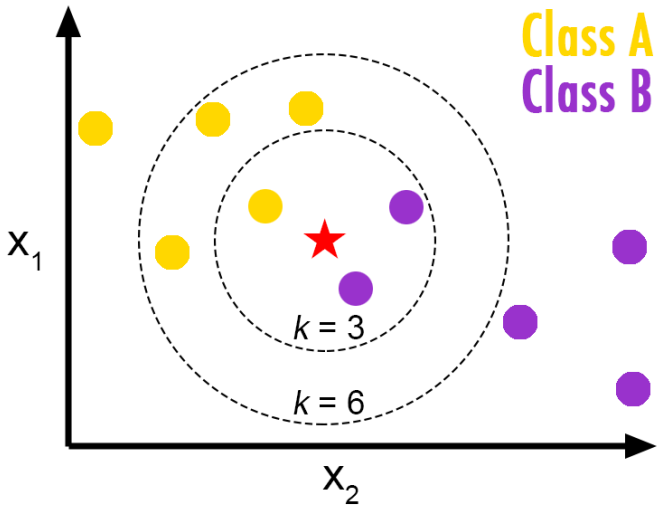
```python
import pickle
import streamlit as st

# Aplikasi Streamlit
st.title('Prediksi Kalori')

# Form input data
st.header('Masukkan Data')
umur = st.number_input('Umur', min_value=25, max_value=55)
bb = st.number_input('Berat Badan (BB)', min_value=60, max_value=95)
tb = st.number_input('Tinggi Badan (TB)', min_value=155, max_value=180)
olahraga = st.number_input('Durasi Olahraga (menit)', min_value=20, max_value=90)

# Tombol prediksi
if st.button('Prediksi'):
    # Memuat model dari file pickle
    loaded_model = pickle.load(open('regression_model.pkl', 'rb'))

    # Melakukan prediksi
    input_data = [[umur, bb, tb, olahraga]]
    prediction = loaded_model.predict(input_data)

    # Menampilkan hasil prediksi
    st.header('Hasil Prediksi')
    st.write(f'Kalori yang diperkirakan: {prediction[0]:.2f}')
```

Untuk menjalankan aplikasinya buka command promt, lalu arahkan ke direktory project dan masukan perintah **streamlit run nama_aplikasi.py**



Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : K Nearest Neighbors
Pokok Pemb : - Konsep Algoritma KNN
- Membangun Model KNN
- Simulasi Algoritma KNN
- Evaluasi Algoritma KNN
- Aplikasi menggunakan algoritma KNN

1. Konsep Algoritma KNN



- k-NN mengidentifikasi nilai “k” dalam data pelatihan yang merupakan “tetangga terdekat” (Lantz, 2013)
- Metode untuk melakukan klasifikasi terhadap objek berdasarkan pembelajaran yang jaraknya paling dekat dengan objek tersebut

Model algoritma KNN

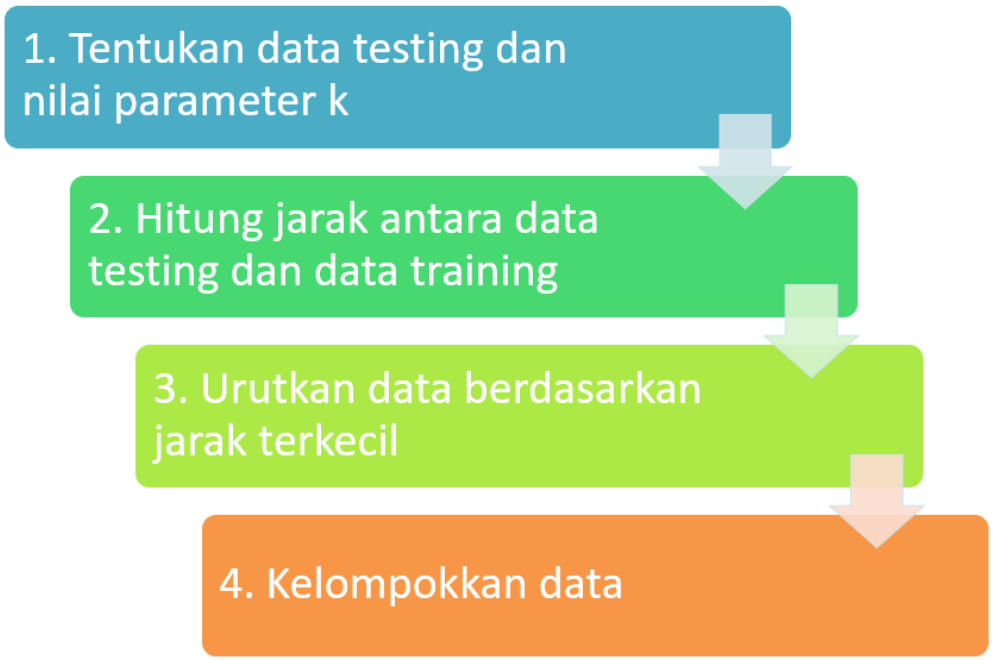
- Euclidean Distance

$$Ed(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Manhattan Distance

$$Md(x,y) = \sum_{i=1}^{n}|x_i - y_i|$$

Langkah-langkah Algoritma KNN :



2. Membangun Model KNN

```
import numpy as np
import pandas as pd

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

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

```
df = pd.read_csv('Pinjaman.csv')
```

```
df.head()
```

```python
# Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['Usia'], bins=10, kde=True, color='blue')
plt.title('Distribution of Age')
plt.xlabel('Usia')
plt.ylabel('Frequency')
plt.show()
```

```python
# Boxplot of Income (Pendapatan) vs Lulus Kredit
plt.figure(figsize=(8, 6))
sns.boxplot(x='Lulus_Kredit', y='Pendapatan', data=df, palette='Set1')
plt.title('Pendapatan vs Kelayakan Kredit')
plt.xlabel('Lulus Kredit')
plt.ylabel('Pendapatan (Juta)')
plt.xticks(ticks=[0, 1], labels=['Tidak Layak', 'Layak'])
plt.show()
```

```python
# Countplot of Marital Status
plt.figure(figsize=(8, 6))
sns.countplot(x='Status_Perkawinan', data=df, palette='Set2')
plt.title('Count of Marital Status')
plt.xticks(ticks=[0, 1], labels=['Belum Menikah', 'Menikah'])
plt.ylabel('Frequency')
plt.show()
```

```python
# Pairplot for the numerical features
plt.figure(figsize=(8, 6))
sns.pairplot(df[['Usia', 'Pendapatan',
                 'Jumlah_Pinjaman',
                 'Durasi_Pinjaman',
                 'Lulus_Kredit']],
             hue='Lulus_Kredit', palette='coolwarm')
plt.show()
```

```python
# Violin plot of Loan Amount vs Credit Eligibility
plt.figure(figsize=(8, 6))
sns.violinplot(x='Lulus_Kredit', y='Jumlah_Pinjaman', data=df, palette='Set2')
plt.title('Distribusi Jumlah Pinjaman berdasarkan Kelayakan Kredit')
plt.xlabel('Lulus Kredit')
plt.ylabel('Jumlah Pinjaman (Juta)')
plt.xticks(ticks=[0, 1], labels=['Tidak Layak', 'Layak'])
plt.show()
```

```python
label_encoder_status_perkawinan = LabelEncoder()
label_encoder_status_pekerjaan = LabelEncoder()
label_encoder_lulus_kredit = LabelEncoder()
```

```python
df['Status_Perkawinan'] = label_encoder_status_perkawinan.fit_transform(df['Status_Perkawinan'])
df['Status_Pekerjaan'] = label_encoder_status_pekerjaan.fit_transform(df['Status_Pekerjaan'])
df['Lulus_Kredit'] = label_encoder_lulus_kredit.fit_transform(df['Lulus_Kredit'])
```

Status Perkawinan (0:Belum Menikah) | (1:Menikah)

Status Pekerjaan (0:Karyawan Kontrak) | (1:Karyawan Tetap) | (2:Pensiunan) | (3:Wirausaha)

Lulus Kredit (0:layak) | (1:tidak layak)

```
df.to_csv('Pinjaman_modif.csv', index=False)
```

```
# Splitting the dataset into features and labels
X = df.drop(columns=['Lulus_Kredit'])
y = df['Lulus_Kredit']
```

```
# Splitting data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

```
# Creating and training the KNN model
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

```
# Predicting the test data
y_pred = knn_model.predict(X_test)
```

3. Simulasi Algoritma KNN

```
    # Example:
    # [Usia : 30,
    # Pendapatan : 80,
    # Status Perkawinan : 0,
    # Jumlah Pinjaman : 70,
    # Durasi Pinjaman : 10,
    # Status Pekerjaan : 1]
new_data = np.array([[30, 80, 0, 70, 10, 1]])
new_pred = knn_model.predict(new_data)
```

```
print('Hasil prediksi: ', new_pred)
```

```
label_encoder_lulus_kredit.inverse_transform(new_pred)
```

4. Evaluasi Algoritma KNN

```
# Calculating the accuracy
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

```
# Menyimpan model menggunakan pickle
import pickle

filename = 'knn_pinjam_mod.pkl'
pickle.dump(knn_model, open(filename, 'wb'))
```

5. Aplikasi Menggunakan Algoritma KNN

Buat file **knn_pinjaman.py** pada direktori project

```python
import streamlit as st
import numpy as np
import pickle
import plotly.express as px
import pandas as pd

# Load the trained KNN model
with open('knn_pinjam_mod.pkl', 'rb') as model_file:
    knn = pickle.load(model_file)

# Create a title for the web app
st.title('K-Nearest Neighbor (KNN) Loan Eligibility Prediction')

# Input fields for new data
usia = st.sidebar.number_input('Usia', min_value=18, max_value=70, value=30)
pendapatan = st.sidebar.number_input('Pendapatan', min_value=10, max_value=200, value=50)
status_perkawinan = st.sidebar.selectbox('Status_Perkawinan',['Belum Menikah','Menikah'])
if status_perkawinan == 'Belum Menikah':
    status_perkawinan = 0
else:
    status_perkawinan = 1
jumlah_pinjaman = st.sidebar.number_input('Jumlah_Pinjaman', min_value=10, max_value=500, value=100)
durasi_pinjaman = st.sidebar.number_input('Durasi_Pinjaman', min_value=1, max_value=30, value=10)
#status_pekerjaan = st.sidebar.number_input('Status_Pekerjaan', min_value=0, max_value=3, value=0)
status_pekerjaan = st.sidebar.selectbox('Status_Pekerjaan',['Karyawan Kontrak','Karyawan Tetap','Pensiunan','Wirausaha'])
if status_pekerjaan == 'Karyawan Kontrak':
    status_pekerjaan = 0
elif status_pekerjaan == 'Karyawan Tetap':
    status_pekerjaan = 1
elif status_pekerjaan == 'Pensiunan':
    status_pekerjaan = 2
elif status_pekerjaan == 'Wirausaha':
    status_pekerjaan = 3

# New data for prediction
new_data = np.array([[usia, pendapatan, status_perkawinan, jumlah_pinjaman, durasi_pinjaman, status_pekerjaan]])

# Predict button
if st.sidebar.button('Predict'):
    # Predict the result
    prediction = knn.predict(new_data)
    result = 'Layak' if prediction == 0 else 'Tidak Layak'

    # Show the prediction result
    st.write(f'Hasil prediksi: **{result}**')


    # 3D Plot for KNN
    df = pd.DataFrame({
        'Usia': np.random.randint(18, 70, size=60),
        'Pendapatan': np.random.randint(10, 200, size=60),
        'Jumlah_Pinjaman': np.random.uniform(10, 500, size=60),
        'Lulus_Kredit': np.random.choice(['0','1'], size=60)
    })
    fig = px.scatter_3d(df, x='Usia', y='Pendapatan', z='Jumlah_Pinjaman', color='Lulus_Kredit',
                        title=f'KNN Prediction')
    st.plotly_chart(fig)
```

Untuk menjalankan aplikasinya buka command promt, lalu arahkan ke direktory project dan masukan perintah streamlit run nama_aplikasi.py

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Naïve Bayes
Pokok Pemb : - Konsep Algoritma Naïve Bayes
- Membangun Model Naïve Bayes
- Simulasi Algoritma Naïve Bayes
- Evaluasi Algoritma Naïve Bayes
- Aplikasi menggunakan algoritma Naïve Bayes

1. Konsep Algoritma Naïve Bayes

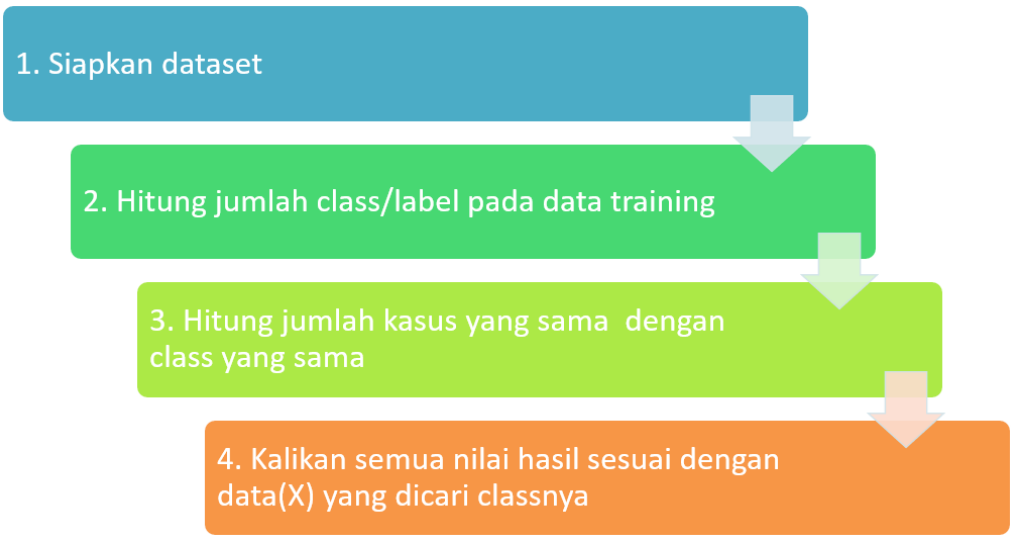
Naïve Bayes Classifier merupakan sebuah metoda klasifikasi yang berakar pada teorema Bayes . Metode pengklasifikasian dg menggunakan metode probabilitas dan statistik yg dikemukakan oleh ilmuwan Inggris Thomas Bayes , yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai Teorema Bayes . Ciri utama dr Naïve Bayes Classifier ini adalah asumsi yg sangat kuat (naïf) akan independensi dari masing-masing kondisi / kejadian.

Model Algoritma Naïve Bayes

$$P(H|X)=\frac{P(X|H)P(H)}{P(X)}=P(X|H)\times P(H)/P(X)$$

x = Data dari class scoring yang belum diketahui
H = Hipotesis data X yang merupakan suatu class yang lebih spesifik
P(H|X) = Probabilitas hipotesis H berdasarkan kondisi X
P(H) = Probabilitas hipotesis H
P(X|H) = Probabilitas hipotesis X berdasarkan kondisi H
P(X) = Probabilitas X

Tahapan Algoritma Naïve Bayes



a. Siapkan Dataset

- Data training buys_computer dataset

Age	Income	Student	Credit Rating	Buys_Computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

- Data testing

Age	Income	Student	Credit Rating	Buys_Computer
Senior	high	no	fair	**?**

Apakah calon pembeli akan membeli komputer?

b. Hitung jumlah class/label pada data training

Class 1 (C1)→ Buys_Computer yes → 9

Class 2 (C2) → Buys_Computer no → 5

Total Record: 14

Maka:

P(C1) = 9/14 = 0.642857143

P(C2) = 5/14 = 0.357142857

c. Hitung jumlah kasus yang sama dengan class yang sama

Contoh penghitungan P(X|Ci) pada kasus Income dimana

i=1 dan i=2

Maka :

P(Income="high"|Buys_Computer="yes") = 2/9 = 0.22222222

P(Income="high"|Buys_Computer="no") = 2/5 = 0.4

d. Kalikan semua nilai hasil sesuai dengan data(X) yang dicari classnya

Jika semua data aribut pada data training dihitung, maka:

Atribut	Parameter	yes	no
Age	youth	0.222222222	0.6
Age	middle_aged	0.444444444	0
Age	**senior**	**0.333333333**	**0.4**
Income	low	0.333333333	0.2
Income	medium	0.444444444	0.4
Income	**high**	**0.222222222**	**0.4**
Student	yes	0.666666667	0.2
Student	**no**	**0.333333333**	**0.8**
Credit Rating	**fair**	**0.666666667**	**0.4**
Credit Rating	excellent	0.333333333	0.6

P(X|Buys_Computer="yes") = 0.016460905

P(X|Buys_Computer="no") = 0.0512

P(X|Buys_Computer="yes")*P(C1) = 0.010582011

P(X|Buys_Computer="no")*P(C2) = 0.018285714

Nilai "no" lebih besar daripada nilai "yes". Sehingga pembeli dengan atribut X tidak membeli komputer

2. Membangun Model Naïve Bayes

```
1 import pandas as pd
2 import numpy as np
3
4 from sklearn.model_selection import train_test_split
5 from sklearn.naive_bayes import CategoricalNB
6 from sklearn.metrics import accuracy_score, confusion_matrix
7
8 import matplotlib.pyplot as plt
9 import seaborn as sns
```

```
df = pd.read_csv('Diabetes1.csv')
```

	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	45	28.7	135	Yes	Poor	Yes
1	50	31.2	145	No	Moderate	Yes
2	30	22.0	95	No	Good	No
3	35	25.4	105	Yes	Poor	Yes
4	60	33.5	155	Yes	Poor	Yes

```
df.head()
```

```
df.info()
```

```
# 1. Bar Chart: Jumlah pasien diabetes berdasarkan jenis diet
plt.figure(figsize=(8, 5))
sns.countplot(x='Diet', hue='Diabetes', data=df)
plt.title("Jumlah pasien diabetes berdasarkan jenis diet")
plt.xlabel("Diet Type")
plt.ylabel("Count")
plt.legend(title="Diabetes")
plt.show()
```

```python
# 2. Pie Chart: Persentase pasien dengan riwayat keluarga
family_history_counts = df['Family_History'].value_counts()
plt.figure(figsize=(5, 5))
family_history_counts.plot.pie(autopct='%1.1f%%', startangle=140, colors=['#ff9999','#66b3ff'])
plt.title("Persentase pasien dengan riwayat keluarga")
plt.ylabel("")
plt.show()
```

```python
# 3. Line Chart: Tren kadar gula darah berdasarkan usia
df_sorted = df.sort_values(by='Age')
plt.figure(figsize=(8, 5))
plt.plot(df_sorted['Age'], df_sorted['Blood_Sugar_Level'], marker='o')
plt.title("Tren kadar gula darah berdasarkan usia")
plt.xlabel("Age")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.grid(True)
plt.show()
```

```python
# 4. Scatter Plot: Hubungan antara BMI dan Kadar Gula Darah
plt.figure(figsize=(8, 5))
sns.scatterplot(x='BMI', y='Blood_Sugar_Level', hue='Diabetes', data=df)
plt.title("Hubungan antara BMI dan Kadar Gula Darah")
plt.xlabel("BMI")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.legend(title="Diabetes")
plt.show()
```

```python
# 5. Histogram: Distribusi BMI
plt.figure(figsize=(8, 5))
plt.hist(df['BMI'], bins=5, color='skyblue', edgecolor='black')
plt.title("Distribusi BMI")
plt.xlabel("BMI")
plt.ylabel("Frequency")
plt.show()
```

```python
# 6. Box Plot: Tingkat Gula Darah berdasarkan Riwayat Keluarga
plt.figure(figsize=(8, 5))
sns.boxplot(x='Family_History', y='Blood_Sugar_Level', data=df)
plt.title("Tingkat Gula Darah berdasarkan Riwayat Keluarga")
plt.xlabel("Family History")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.show()
```

```python
# 7. Heatmap: Korelasi antar variabel numerik
plt.figure(figsize=(8, 5))
sns.heatmap(df[['Age', 'BMI', 'Blood_Sugar_Level']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

```python
# 8. Area Chart: Kasus diabetes kumulatif berdasarkan usia (untuk tujuan ilustrasi)
age_cum_cases = df[df['Diabetes'] == 'Yes'].groupby('Age').size().cumsum()
plt.figure(figsize=(8, 5))
plt.fill_between(age_cum_cases.index, age_cum_cases.values, color='lightgreen', alpha=0.6)
plt.title("Kasus Diabetes Kumulatif Berdasarkan Usia")
plt.xlabel("Age")
plt.ylabel("Cumulative Cases")
plt.grid(True)
plt.show()
```

```python
# Membuat DataFrame
df_encoded = pd.DataFrame(df)
```

```python
# Encode kolom categorical
df_encoded['Family_History'] = df_encoded['Family_History'].map({'Yes': 1, 'No': 0})
df_encoded['Diet'] = df_encoded['Diet'].map({'Poor': 0, 'Moderate': 1, 'Good': 2})
df_encoded['Diabetes'] = df_encoded['Diabetes'].map({'Yes': 1, 'No': 0})
```

```
df_encoded.head()
```

	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	45	28.7	135	1	0	1
1	50	31.2	145	0	1	1
2	30	22.0	95	0	2	0
3	35	25.4	105	1	0	1
4	60	33.5	155	1	0	1

```
# mempersiapkan data untuk Categorical Naive Bayes
categorical_df = df_encoded.copy()
categorical_df['Age'] = pd.cut(categorical_df['Age'], bins=[0, 30, 40, 50, 60, 80], labels=[0, 1, 2, 3, 4])
categorical_df['BMI'] = pd.cut(categorical_df['BMI'], bins=[0, 18.5, 24.9, 29.9, 50], labels=[0, 1, 2, 3])
categorical_df['Blood_Sugar_Level'] = pd.cut(categorical_df['Blood_Sugar_Level'], bins=[0, 100, 125, 150, 200], labels=[0, 1, 2, 3])
```

```
categorical_df.head()
```

	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	2	2	2	1	0	1
1	2	3	2	0	1	1
2	0	1	0	0	2	0
3	1	2	1	1	0	1
4	3	3	3	1	0	1

```
# tentukan features dan target untuk CategoricalNB
X_categorical = categorical_df[['Age', 'BMI', 'Blood_Sugar_Level', 'Family_History', 'Diet']]
y_categorical = categorical_df['Diabetes']
```

```
# membagi (split) dataset
X_train_cat, X_test_cat, y_train_cat, y_test_cat = train_test_split(X_categorical, y_categorical, test_size=0.3, random_state=42)
```

```
# Creating and training the Categorical Naive Bayes model
cat_nb_model = CategoricalNB()
cat_nb_model.fit(X_train_cat, y_train_cat)
```

```
# Predicting on the test set
y_pred_cat = cat_nb_model.predict(X_test_cat)
```

3. Simulasi Algoritma Naïve Bayes

```
# Simulating with new input data (categorical values)
# Example: Age 45, BMI 27.5, Blood Sugar Level 125, Family History Yes, Diet Poor
# Convert these values to categorical codes
new_data_categorical = np.array([[3, 2, 1, 1, 0]])
new_prediction_cat = cat_nb_model.predict(new_data_categorical)
```

```
(new_prediction_cat)
```

```
if (new_prediction_cat[0]==0):
    print ('Pasien tidak terkena diabetes')
else:
    print ('Pasien terkena diabetes')
```

4. Evaluasi Algoritma Naïve Bayes

```
# Calculating accuracy for Categorical Naive Bayes
cat_accuracy = accuracy_score(y_test_cat, y_pred_cat)
```

```
# Generating the confusion matrix
cat_conf_matrix = confusion_matrix(y_test_cat, y_pred_cat)
```

```
(cat_accuracy, cat_conf_matrix)
```

```
# Menyimpan model menggunakan pickle
import pickle

filename = 'cat_diabetes_mod.pkl'
pickle.dump(cat_nb_model, open(filename, 'wb'))
```

5. Aplikasi Menggunakan algoritma naïve bayes

```python
import streamlit as st
import pickle
import numpy as np
import matplotlib.pyplot as plt

# Load the model from the .pkl file
with open('cat_diabetes_mod.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

# Streamlit app title
st.title("Categorical Naive Bayes - Diabetes Prediction")

# Sidebar inputs for user data
st.sidebar.header("Input Data Baru")
age = st.sidebar.selectbox("Age", ["<30", "30-40", "40-50", "50-60", "60+"])
bmi = st.sidebar.selectbox("BMI", ["<18.5", "18.5-24.9", "25-29.9", "30+"])
blood_sugar = st.sidebar.selectbox("Blood Sugar Level", ["<100", "100-125", "125-150", "150+"])
family_history = st.sidebar.selectbox("Family History of Diabetes", ["No", "Yes"])
diet = st.sidebar.selectbox("Diet", ["Good", "Moderate", "Poor"])

# Map inputs to numerical values (based on categories used for CategoricalNB)
age_map = {"<30": 0, "30-40": 1, "40-50": 2, "50-60": 3, "60+": 4}
bmi_map = {"<18.5": 0, "18.5-24.9": 1, "25-29.9": 2, "30+": 3}
blood_sugar_map = {"<100": 0, "100-125": 1, "125-150": 2, "150+": 3}
family_history_map = {"No": 0, "Yes": 1}
diet_map = {"Good": 2, "Moderate": 1, "Poor": 0}

# Convert user input to model input format
input_data = np.array([[age_map[age], bmi_map[bmi], blood_sugar_map[blood_sugar],
                        family_history_map[family_history], diet_map[diet]]])

# Make prediction
prediction = model.predict(input_data)
prediction_prob = model.predict_proba(input_data)

# Display results
st.subheader("Prediction Result")
result = "Diabetes (Yes)" if prediction[0] == 1 else "No Diabetes"
st.write(f"The model predicts: **{result}**")

# Plot prediction probabilities
fig, ax = plt.subplots()
labels = ['No Diabetes', 'Diabetes']
ax.bar(labels, prediction_prob[0], color=['green', 'red'])
ax.set_ylabel('Probability')
ax.set_title('Prediction Probability')
st.pyplot(fig)

# Display input summary
st.subheader("Input Summary")
st.write(f"**Age**: {age}")
st.write(f"**BMI**: {bmi}")
st.write(f"**Blood Sugar Level**: {blood_sugar}")
st.write(f"**Family History**: {family_history}")
st.write(f"**Diet**: {diet}")
```

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Naïve Bayes
Pokok Pemb : - Konsep Algoritma Naïve Bayes
- Membangun Model Naïve Bayes
- Simulasi Algoritma Naïve Bayes
- Evaluasi Algoritma Naïve Bayes
- Aplikasi menggunakan algoritma Naïve Bayes

1. Konsep Algoritma Naïve Bayes

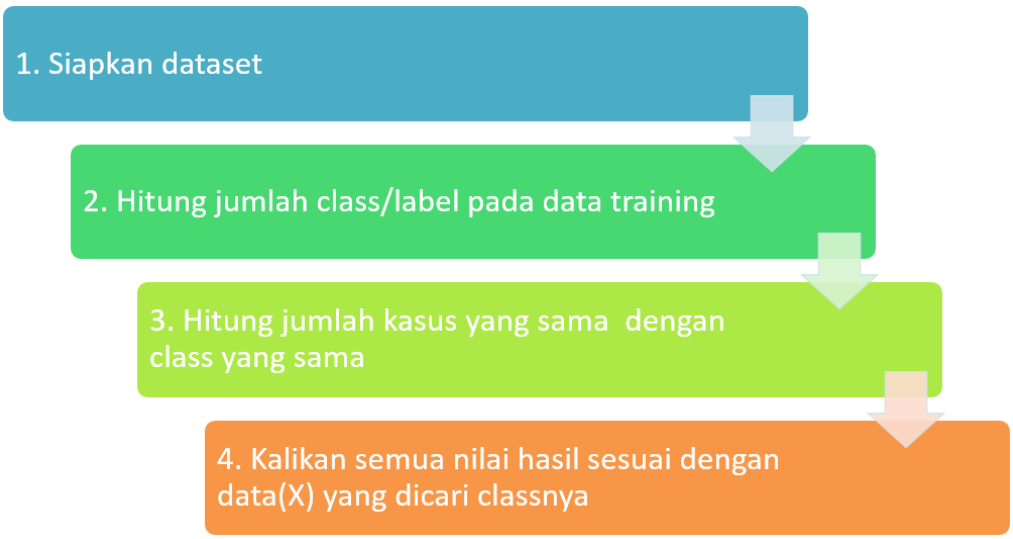
Naïve Bayes Classifier merupakan sebuah metoda klasifikasi yang berakar pada teorema Bayes . Metode pengklasifikasian dg menggunakan metode probabilitas dan statistik yg dikemukakan oleh ilmuwan Inggris Thomas Bayes , yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai Teorema Bayes . Ciri utama dr Naïve Bayes Classifier ini adalah asumsi yg sangat kuat (naïf) akan independensi dari masing-masing kondisi / kejadian.

Model Algoritma Naïve Bayes

$$P(H|X)=\frac{P(X|H)P(H)}{P(X)}=P(X|H)\times P(H)/P(X)$$

x = Data dari class scoring yang belum diketahui
H = Hipotesis data X yang merupakan suatu class yang lebih spesifik
P(H|X) = Probabilitas hipotesis H berdasarkan kondisi X
P(H) = Probabilitas hipotesis H
P(X|H) = Probabilitas hipotesis X berdasarkan kondisi H
P(X) = Probabilitas X

Tahapan Algoritma Naïve Bayes



a. Siapkan Dataset

- Data training buys_computer dataset

Age	Income	Student	Credit Rating	Buys_Computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

- Data testing

Age	Income	Student	Credit Rating	Buys_Computer
Senior	high	no	fair	**?**

Apakah calon pembeli akan membeli komputer?

b. Hitung jumlah class/label pada data training

Class 1 (C1)→ Buys_Computer yes → 9

Class 2 (C2) → Buys_Computer no → 5

Total Record: 14

Maka:

P(C1) = 9/14 = 0.642857143

P(C2) = 5/14 = 0.357142857

c. Hitung jumlah kasus yang sama dengan class yang sama

Contoh penghitungan P(X|Ci) pada kasus Income dimana

i=1 dan i=2

Maka :

P(Income="high"|Buys_Computer="yes") = 2/9 = 0.22222222

P(Income="high"|Buys_Computer="no") = 2/5 = 0.4

d. Kalikan semua nilai hasil sesuai dengan data(X) yang dicari classnya

Jika semua data aribut pada data training dihitung, maka:

Atribut	Parameter	yes	no
Age	youth	0.222222222	0.6
Age	middle_aged	0.444444444	0
Age	**senior**	**0.333333333**	**0.4**
Income	low	0.333333333	0.2
Income	medium	0.444444444	0.4
Income	**high**	**0.222222222**	**0.4**
Student	yes	0.666666667	0.2
Student	**no**	**0.333333333**	**0.8**
Credit Rating	**fair**	**0.666666667**	**0.4**
Credit Rating	excellent	0.333333333	0.6

P(X|Buys_Computer="yes") = 0.016460905

P(X|Buys_Computer="no") = 0.0512

P(X|Buys_Computer="yes")*P(C1) = 0.010582011

P(X|Buys_Computer="no")*P(C2) = 0.018285714

Nilai "no" lebih besar daripada nilai "yes". Sehingga pembeli dengan atribut X tidak membeli komputer

2. Membangun Model Naïve Bayes

```
1 import pandas as pd
2 import numpy as np
3
4 from sklearn.model_selection import train_test_split
5 from sklearn.naive_bayes import CategoricalNB
6 from sklearn.metrics import accuracy_score, confusion_matrix
7
8 import matplotlib.pyplot as plt
9 import seaborn as sns
```

```
df = pd.read_csv('Diabetes1.csv')
```

	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	45	28.7	135	Yes	Poor	Yes
1	50	31.2	145	No	Moderate	Yes
2	30	22.0	95	No	Good	No
3	35	25.4	105	Yes	Poor	Yes
4	60	33.5	155	Yes	Poor	Yes

```
df.head()
```

```
df.info()
```

```
# 1. Bar Chart: Jumlah pasien diabetes berdasarkan jenis diet
plt.figure(figsize=(8, 5))
sns.countplot(x='Diet', hue='Diabetes', data=df)
plt.title("Jumlah pasien diabetes berdasarkan jenis diet")
plt.xlabel("Diet Type")
plt.ylabel("Count")
plt.legend(title="Diabetes")
plt.show()
```

```python
# 2. Pie Chart: Persentase pasien dengan riwayat keluarga
family_history_counts = df['Family_History'].value_counts()
plt.figure(figsize=(5, 5))
family_history_counts.plot.pie(autopct='%1.1f%%', startangle=140, colors=['#ff9999','#66b3ff'])
plt.title("Persentase pasien dengan riwayat keluarga")
plt.ylabel("")
plt.show()
```

```python
# 3. Line Chart: Tren kadar gula darah berdasarkan usia
df_sorted = df.sort_values(by='Age')
plt.figure(figsize=(8, 5))
plt.plot(df_sorted['Age'], df_sorted['Blood_Sugar_Level'], marker='o')
plt.title("Tren kadar gula darah berdasarkan usia")
plt.xlabel("Age")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.grid(True)
plt.show()
```

```python
# 4. Scatter Plot: Hubungan antara BMI dan Kadar Gula Darah
plt.figure(figsize=(8, 5))
sns.scatterplot(x='BMI', y='Blood_Sugar_Level', hue='Diabetes', data=df)
plt.title("Hubungan antara BMI dan Kadar Gula Darah")
plt.xlabel("BMI")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.legend(title="Diabetes")
plt.show()
```

```python
# 5. Histogram: Distribusi BMI
plt.figure(figsize=(8, 5))
plt.hist(df['BMI'], bins=5, color='skyblue', edgecolor='black')
plt.title("Distribusi BMI")
plt.xlabel("BMI")
plt.ylabel("Frequency")
plt.show()
```

```python
# 6. Box Plot: Tingkat Gula Darah berdasarkan Riwayat Keluarga
plt.figure(figsize=(8, 5))
sns.boxplot(x='Family_History', y='Blood_Sugar_Level', data=df)
plt.title("Tingkat Gula Darah berdasarkan Riwayat Keluarga")
plt.xlabel("Family History")
plt.ylabel("Blood Sugar Level (mg/dL)")
plt.show()
```

```python
# 7. Heatmap: Korelasi antar variabel numerik
plt.figure(figsize=(8, 5))
sns.heatmap(df[['Age', 'BMI', 'Blood_Sugar_Level']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

```python
# 8. Area Chart: Kasus diabetes kumulatif berdasarkan usia (untuk tujuan ilustrasi)
age_cum_cases = df[df['Diabetes'] == 'Yes'].groupby('Age').size().cumsum()
plt.figure(figsize=(8, 5))
plt.fill_between(age_cum_cases.index, age_cum_cases.values, color='lightgreen', alpha=0.6)
plt.title("Kasus Diabetes Kumulatif Berdasarkan Usia")
plt.xlabel("Age")
plt.ylabel("Cumulative Cases")
plt.grid(True)
plt.show()
```

```python
# Membuat DataFrame
df_encoded = pd.DataFrame(df)
```

```python
# Encode kolom categorical
df_encoded['Family_History'] = df_encoded['Family_History'].map({'Yes': 1, 'No': 0})
df_encoded['Diet'] = df_encoded['Diet'].map({'Poor': 0, 'Moderate': 1, 'Good': 2})
df_encoded['Diabetes'] = df_encoded['Diabetes'].map({'Yes': 1, 'No': 0})
```

```
df_encoded.head()
```

	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	45	28.7	135	1	0	1
1	50	31.2	145	0	1	1
2	30	22.0	95	0	2	0
3	35	25.4	105	1	0	1
4	60	33.5	155	1	0	1

```
# mempersiapkan data untuk Categorical Naive Bayes
categorical_df = df_encoded.copy()
categorical_df['Age'] = pd.cut(categorical_df['Age'], bins=[0, 30, 40, 50, 60, 80], labels=[0, 1, 2, 3, 4])
categorical_df['BMI'] = pd.cut(categorical_df['BMI'], bins=[0, 18.5, 24.9, 29.9, 50], labels=[0, 1, 2, 3])
categorical_df['Blood_Sugar_Level'] = pd.cut(categorical_df['Blood_Sugar_Level'], bins=[0, 100, 125, 150, 200], labels=[0, 1, 2, 3])
```

```
categorical_df.head()
```

	Age	BMI	Blood_Sugar_Level	Family_History	Diet	Diabetes
0	2	2	2	1	0	1
1	2	3	2	0	1	1
2	0	1	0	0	2	0
3	1	2	1	1	0	1
4	3	3	3	1	0	1

```
# tentukan features dan target untuk CategoricalNB
X_categorical = categorical_df[['Age', 'BMI', 'Blood_Sugar_Level', 'Family_History', 'Diet']]
y_categorical = categorical_df['Diabetes']
```

```
# membagi (split) dataset
X_train_cat, X_test_cat, y_train_cat, y_test_cat = train_test_split(X_categorical, y_categorical, test_size=0.3, random_state=42)
```

```
# Creating and training the Categorical Naive Bayes model
cat_nb_model = CategoricalNB()
cat_nb_model.fit(X_train_cat, y_train_cat)
```

```
# Predicting on the test set
y_pred_cat = cat_nb_model.predict(X_test_cat)
```

3. Simulasi Algoritma Naïve Bayes

```
# Simulating with new input data (categorical values)
# Example: Age 45, BMI 27.5, Blood Sugar Level 125, Family History Yes, Diet Poor
# Convert these values to categorical codes
new_data_categorical = np.array([[3, 2, 1, 1, 0]])
new_prediction_cat = cat_nb_model.predict(new_data_categorical)
```

```
(new_prediction_cat)
```

```
if (new_prediction_cat[0]==0):
    print ('Pasien tidak terkena diabetes')
else:
    print ('Pasien terkena diabetes')
```

4. Evaluasi Algoritma Naïve Bayes

```
# Calculating accuracy for Categorical Naive Bayes
cat_accuracy = accuracy_score(y_test_cat, y_pred_cat)
```

```
# Generating the confusion matrix
cat_conf_matrix = confusion_matrix(y_test_cat, y_pred_cat)
```

```
(cat_accuracy, cat_conf_matrix)
```

```
# Menyimpan model menggunakan pickle
import pickle

filename = 'cat_diabetes_mod.pkl'
pickle.dump(cat_nb_model, open(filename, 'wb'))
```

5. Aplikasi Menggunakan algoritma naïve bayes

```python
import streamlit as st
import pickle
import numpy as np
import matplotlib.pyplot as plt

# Load the model from the .pkl file
with open('cat_diabetes_mod.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

# Streamlit app title
st.title("Categorical Naive Bayes - Diabetes Prediction")

# Sidebar inputs for user data
st.sidebar.header("Input Data Baru")
age = st.sidebar.selectbox("Age", ["<30", "30-40", "40-50", "50-60", "60+"])
bmi = st.sidebar.selectbox("BMI", ["<18.5", "18.5-24.9", "25-29.9", "30+"])
blood_sugar = st.sidebar.selectbox("Blood Sugar Level", ["<100", "100-125", "125-150", "150+"])
family_history = st.sidebar.selectbox("Family History of Diabetes", ["No", "Yes"])
diet = st.sidebar.selectbox("Diet", ["Good", "Moderate", "Poor"])

# Map inputs to numerical values (based on categories used for CategoricalNB)
age_map = {"<30": 0, "30-40": 1, "40-50": 2, "50-60": 3, "60+": 4}
bmi_map = {"<18.5": 0, "18.5-24.9": 1, "25-29.9": 2, "30+": 3}
blood_sugar_map = {"<100": 0, "100-125": 1, "125-150": 2, "150+": 3}
family_history_map = {"No": 0, "Yes": 1}
diet_map = {"Good": 2, "Moderate": 1, "Poor": 0}

# Convert user input to model input format
input_data = np.array([[age_map[age], bmi_map[bmi], blood_sugar_map[blood_sugar],
                        family_history_map[family_history], diet_map[diet]]])

# Make prediction
prediction = model.predict(input_data)
prediction_prob = model.predict_proba(input_data)

# Display results
st.subheader("Prediction Result")
result = "Diabetes (Yes)" if prediction[0] == 1 else "No Diabetes"
st.write(f"The model predicts: **{result}**")

# Plot prediction probabilities
fig, ax = plt.subplots()
labels = ['No Diabetes', 'Diabetes']
ax.bar(labels, prediction_prob[0], color=['green', 'red'])
ax.set_ylabel('Probability')
ax.set_title('Prediction Probability')
st.pyplot(fig)

# Display input summary
st.subheader("Input Summary")
st.write(f"**Age**: {age}")
st.write(f"**BMI**: {bmi}")
st.write(f"**Blood Sugar Level**: {blood_sugar}")
st.write(f"**Family History**: {family_history}")
st.write(f"**Diet**: {diet}")
```

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Decision Tree (C.45)
Pokok Pemb : - Membangun Model Decision Tree
- Simulasi Algoritma Decision Tree
- Evaluasi Algoritma Decision Tree
- Aplikasi menggunakan algoritma Decision Tree

1. Membangun Model Decision Tree

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

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
```

```
df = pd.read_csv('heart.csv')
```

```
df.head()
```

	Age	Cholesterol Level	Blood Pressure	Smoking	Physical Activity	BMI	Heart Disease Risk
0	55	High	High	Yes	Low	Overweight	High
1	43	Normal	Normal	No	Moderate	Normal	Low
2	60	High	High	Yes	Low	Obese	High
3	35	Normal	Low	No	High	Normal	Low
4	50	High	High	No	Low	Overweight	Medium

```python
# 1. Histogram of Age
plt.figure(figsize=(10, 6))
plt.hist(df['Age'], bins=10, edgecolor='black')
plt.title("Distribution of Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

```python
# 2. Bar plot of Cholesterol Level Counts
plt.figure(figsize=(10, 6))
df['Cholesterol Level'].value_counts().plot(kind='bar')
plt.title("Count of Cholesterol Levels")
plt.xlabel("Cholesterol Level")
plt.ylabel("Count")
plt.show()
```

```python
# 3. Bar plot of Blood Pressure Counts
plt.figure(figsize=(10, 6))
df['Blood Pressure'].value_counts().plot(kind='bar', color='orange')
plt.title("Count of Blood Pressure Levels")
plt.xlabel("Blood Pressure Level")
plt.ylabel("Count")
plt.show()
```

```python
# 4. Pie chart of Smoking Status
plt.figure(figsize=(8, 8))
df['Smoking'].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=140)
plt.title("Proportion of Smoking Status")
plt.ylabel("")  # Removes 'Smoking' label from pie
plt.show()
```

```python
# 5. Count plot of BMI categories
plt.figure(figsize=(10, 6))
df['BMI'].value_counts().plot(kind='bar', color='purple')
plt.title("Count of BMI Categories")
plt.xlabel("BMI Category")
plt.ylabel("Count")
plt.show()
```

```python
# 6. Violin Plot of Age by Cholesterol Level
plt.figure(figsize=(10, 6))
sns.violinplot(x='Cholesterol Level', y='Age', data=df)
plt.title("Violin Plot of Age by Cholesterol Level")
plt.xlabel("Cholesterol Level")
plt.ylabel("Age")
plt.show()
```

```python
# 7. Swarm Plot of Age by BMI Category
plt.figure(figsize=(10, 6))
sns.swarmplot(x='BMI', y='Age', data=df)
plt.title("Swarm Plot of Age by BMI Category")
plt.xlabel("BMI Category")
plt.ylabel("Age")
plt.show()
```

```python
# 8. Box Plot of Age by Blood Pressure
plt.figure(figsize=(10, 6))
sns.boxplot(x='Blood Pressure', y='Age', data=df)
plt.title("Box Plot of Age by Blood Pressure Level")
plt.xlabel("Blood Pressure Level")
plt.ylabel("Age")
plt.show()
```

```python
# 9. Scatter Plot of Age vs Physical Activity with BMI Category as Hue
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Age', y='Physical Activity', hue='BMI')
plt.title("Scatter Plot of Age vs Physical Activity by BMI Category")
plt.xlabel("Age")
plt.ylabel("Physical Activity")
plt.legend(title="BMI Category")
plt.show()
```

```python
# membuat fungsi untuk mengkategorikan usia
def categorize_age(age):
    if age < 35:
        return "Young"
    elif age < 55:
        return "Middle-aged"
    else:
        return "Senior"
```

```python
# terapkan fungsi pada kolom 'Age (Usia)' dan buat kolom baru 'Age Category'
df['Age'] = df['Age'].apply(categorize_age)
```

```
df.head()
```

	Age	Cholesterol Level	Blood Pressure	Smoking	Physical Activity	BMI	Heart Disease Risk
0	Senior	High	High	Yes	Low	Overweight	High
1	Middle-aged	Normal	Normal	No	Moderate	Normal	Low
2	Senior	High	High	Yes	Low	Obese	High
3	Middle-aged	Normal	Low	No	High	Normal	Low
4	Middle-aged	High	High	No	Low	Overweight	Medium

```
# Encode semua variabel kategori
label_encoders = {}
for column in df.columns:
    if df[column].dtype == 'object':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        label_encoders[column] = le
```

```
df.head()
```

	Age	Cholesterol Level	Blood Pressure	Smoking	Physical Activity	BMI	Heart Disease Risk
0	1	0	0	1	1	2	0
1	0	1	2	0	2	0	1
2	1	0	0	1	1	1	0
3	0	1	1	0	0	0	1
4	0	0	0	0	1	2	2

```python
# Split the dataset
X = df.drop('Heart Disease Risk', axis=1)
y = df['Heart Disease Risk']
```

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

```python
# Train the C4.5 model (Decision Tree)
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)
```

```python
# Make predictions
y_pred = model.predict(X_test)
```

2. Simulasi Algoritma Decision Tree

```
# Simulate new input data
new_data = pd.DataFrame({
    'Age': [label_encoders['Age'].transform(['Middle-aged'])[0]],
    'Cholesterol Level': [label_encoders['Cholesterol Level'].transform(['High'])[0]],
    'Blood Pressure': [label_encoders['Blood Pressure'].transform(['High'])[0]],
    'Smoking': [label_encoders['Smoking'].transform(['Yes'])[0]],
    'Physical Activity': [label_encoders['Physical Activity'].transform(['Low'])[0]],
    'BMI': [label_encoders['BMI'].transform(['Overweight'])[0]]
})
```

```
new_prediction = model.predict(new_data)
```

```
# Decode the new prediction
new_prediction_decoded = label_encoders['Heart Disease Risk'].inverse_transform(new_prediction)
```

```
new_prediction_decoded
```

3. Evaluasi Algoritma Decision Tree

```
# Calculate accuracy and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
accuracy, conf_matrix
```

4. Plot Hasil Decision Tree

```python
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(20, 20))
plot_tree(model, feature_names=['Age',
                                'Cholesterol Level',
                                'Blood Pressure',
                                'Smoking',
                                'Physical Activity',
                                'BMI'], class_names=['High Risk',
                                                     'Low Risk',
                                                     'Medium Risk'],
                                                     filled=True, rounded=True)
plt.title("pohon keputusan")
plt.show()
```

5. Aplikasi menggunakan algoritma Decision Tree

```python
import streamlit as st
import pickle
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Load the trained model from the .pkl file
model_path = 'c45_pinjam_mod.pkl'
with open(model_path, 'rb') as model_file:
    loaded_model = pickle.load(model_file)

st.title('C4.5 Decision Tree Model for Heart Disease Prediction')

# Input for new data
st.sidebar.header('Input Features')
age = st.sidebar.selectbox('Age', ('Senior', 'Middle-aged', 'Young'))
cholesterol = st.sidebar.selectbox('Cholesterol Level', ('High', 'Normal', 'Low'))
blood_pressure = st.sidebar.selectbox('Blood Pressure', ('High', 'Normal', 'Low'))
smoking = st.sidebar.selectbox('Smoking', ('Yes', 'No'))
physical_activity = st.sidebar.selectbox('Physical Activity', ('Low', 'Moderate', 'High'))
bmi = st.sidebar.selectbox('BMI', ('Overweight', 'Normal', 'Obese'))

# Create a mapping for the labels to numbers (as per how the model was trained)
label_encodings = {
    'Age': {'Senior': 1, 'Middle-aged': 0, 'Young': 2},
    'Cholesterol Level': {'High': 0, 'Normal': 1, 'Low': 2},
    'Blood Pressure': {'High': 0, 'Normal': 1, 'Low': 2},
    'Smoking': {'Yes': 1, 'No': 0},
    'Physical Activity': {'Low': 0, 'Moderate': 1, 'High': 2},
    'BMI': {'Overweight': 1, 'Normal': 0, 'Obese': 2}
}

# Encode the input data
new_data = np.array([[label_encodings['Age'][age],
                      label_encodings['Cholesterol Level'][cholesterol],
                      label_encodings['Blood Pressure'][blood_pressure],
                      label_encodings['Smoking'][smoking],
                      label_encodings['Physical Activity'][physical_activity],
                      label_encodings['BMI'][bmi]]])

# Predict using the loaded model
if st.button('Predict'):
    prediction = loaded_model.predict(new_data)
    risk_mapping = {0: 'High Risk', 1: 'Low Risk', 2: 'Medium Risk'}
    st.write(f'The predicted heart disease risk is: **{risk_mapping[prediction[0]]}**')

    # Plot the decision tree
    st.write("### Decision Tree Visualization")
    fig = plt.figure(figsize=(20, 10))
    plot_tree(loaded_model, feature_names=['Age',
                                           'Cholesterol Level',
                                           'Blood Pressure',
                                           'Smoking',
                                           'Physical Activity',
                                           'BMI'],
                                           class_names=['High Risk',
                                                        'Low Risk',
                                                        'Medium Risk'],
                                                        filled=True, rounded=True)
    st.pyplot(fig)
```

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : K-Means Clustering
Pokok Pemb : - Membangun Model K-Means Clustering
- Simulasi Algoritma K-means

1. Membangun Model K-means

```
1 mport pandas as pd
2
3 from sklearn.cluster import KMeans
4 from sklearn.preprocessing import StandardScaler
5 import matplotlib.pyplot as plt
6
7 import seaborn as sns
8 from sklearn.decomposition import PCA
```

```
1 # Memuat dataset
2 df = pd.read_csv("pelanggan.csv")
```

```
1 # Pie chart distribusi gender pelanggan
2 gender_counts = df['Gender'].value_counts()
3 plt.pie(gender_counts,
4         labels=gender_counts.index,
5         autopct='%1.1f%%',
6         colors=['skyblue', 'pink'],
7         startangle=90)
8 plt.title("Distribusi Gender Pelanggan")
9 plt.show()
```

```python
# Boxplot pengeluaran rata-rata per transaksi berdasarkan kategori produk
sns.boxplot(data=df,
            x='Preferred_Product_Category',
            y='Average_Spending_Per_Transaction',
            palette='Set3')
plt.title("Pengeluaran Rata-Rata Per Transaksi Berdasarkan Kategori Produk")
plt.xlabel("Kategori Produk")
plt.ylabel("Rata-Rata Pengeluaran Per Transaksi (Rp)")
plt.xticks(rotation=45)
plt.show()
```

```python
# Frekuensi pembelian berdasarkan kota
sns.barplot(data=df,
            x='City',
            y='Purchase_Frequency',
            estimator=sum,
            palette='pastel')
plt.title("Frekuensi Pembelian Berdasarkan Kota")
plt.xlabel("Kota")
plt.ylabel("Total Frekuensi Pembelian")
plt.show()
```

```python
# Bar plot kategori produk yang disukai
sns.countplot(data=df,
              x='Preferred_Product_Category',
              order=df['Preferred_Product_Category'].value_counts().index,
              palette='coolwarm')
plt.title("Kategori Produk yang Disukai Pelanggan")
plt.xlabel("Kategori Produk")
plt.ylabel("Jumlah Pelanggan")
plt.show()
```

```python
# Distribusi frekuensi pembelian pelanggan
sns.histplot(df['Purchase_Frequency'],
             kde=True,
             color='orange',
             bins=15)
plt.title("Distribusi Frekuensi Pembelian Pelanggan")
plt.xlabel("Frekuensi Pembelian")
plt.ylabel("Jumlah Pelanggan")
plt.show()
```

2. Simulasi Algoritma K-Means

```python
# Memilih fitur untuk clustering
features = ["Annual_Income",
            "Purchase_Frequency",
            "Average_Spending_Per_Transaction",
            "Total_Spent"]
```

```python
# Standarisasi data
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df[features])
```

```python
# Menentukan jumlah cluster menggunakan metode Elbow
inertia = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

```python
# Plot Elbow Method
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o')
plt.title("Elbow Method for Optimal k")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.grid()
plt.show()
```

```python
# Menggunakan jumlah cluster optimal
optimal_k = 4  # Berdasarkan elbow method (misalnya)
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)
```

```python
# Mengurangi dimensi data untuk visualisasi
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(scaled_features)
```

```python
# Warna untuk setiap cluster
colors = ['blue', 'green', 'red', 'purple']
```

```python
# Plot hasil clustering
plt.figure(figsize=(10, 7))
for cluster in range(optimal_k):
    cluster_points = reduced_features[df['Cluster'] == cluster]
    plt.scatter(cluster_points[:, 0],
                cluster_points[:, 1],
                s=50,
                label=f'Cluster {cluster}',
                color=colors[cluster])

# Menambahkan centroid pada plot
centroids_reduced = pca.transform(kmeans.cluster_centers_)
plt.scatter(centroids_reduced[:, 0],
            centroids_reduced[:, 1],
            s=200,
            c='black',
            marker='X',
            label='Centroids')

# Menambahkan detail plot
plt.title("Hasil Clustering K-Means (2D PCA Projection)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.grid()
plt.show()
```

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Hierarchical Clustering
Pokok Pemb : - Membangun Model Hierarchical Clustering
- Simulasi Hierarchical Clustering
- Profiling Hasil CLustering

1. Load Library dan dataset pada file Notebook

```python
import pandas as pd

from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree

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

```python
# Membaca dataset
data = pd.read_csv('dataset_nasabah.csv')
```

```python
data.head()
```

✓ 0.0s Python

	ID Nasabah	Usia	Jenis Kelamin	Status Merokok	BMI	Jumlah Klaim	Total Biaya Klaim
0	1	58	Pria	Ya	30.7	1	4744854
1	2	26	Pria	Ya	28.2	0	0
2	3	19	Pria	Ya	22.3	4	5780796
3	4	53	Pria	Tidak	22.1	4	22669060
4	5	69	Pria	Ya	30.0	2	11323814

2. Buatkan 5 variasi Exploratory data analysis dan berikan penjelasannya yang terdiri dari

a. **Pie Plot**
b. **Box plot**
c. **Violin plot**
d. **Scatter plot**
e. **Count plot**

3. Buat model Hierarchical Clustering

```
# Pra-pemrosesan data
# Mengonversi kolom kategorikal menjadi numerik
data['Jenis Kelamin'] = data['Jenis Kelamin'].map({'Pria': 0, 'Wanita': 1})
data['Status Merokok'] = data['Status Merokok'].map({'Tidak': 0, 'Ya': 1})
```

```
data.head()
```

✓ 0.0s Python

	ID Nasabah	Usia	Jenis Kelamin	Status Merokok	BMI	Jumlah Klaim	Total Biaya Klaim
0	1	58	0	1	30.7	1	4744854
1	2	26	0	1	28.2	0	0
2	3	19	0	1	22.3	4	5780796
3	4	53	0	0	22.1	4	22669060
4	5	69	0	1	30.0	2	11323814

```
# Select features for clustering
features = ['Usia', 'Jenis Kelamin', 'Status Merokok', 'BMI', 'Jumlah Klaim', 'Total Biaya Klaim']
X = data[features]
```

```
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```python
# Create linkage matrix for dendrogram
linked = linkage(X_scaled, method='ward')
```

```python
# Plot dendrogram
plt.figure(figsize=(10, 7))
plt.title("Dendrogram for Hierarchical Clustering")
dendrogram(linked, truncate_mode='lastp', p=30, leaf_rotation=90, leaf_font_size=10)
plt.xlabel("Cluster Size")
plt.ylabel("Distance")
plt.show()
```

```python
# membuat cluster label
cluster_labels = cut_tree(linked, n_clusters=5).reshape(-1, )
cluster_labels
```

```python
# gabungkan cluster label dengan data
data['Cluster_Labels'] = cluster_labels
```

```python
data.head()
```

✓ 0.0s Python

	ID Nasabah	Usia	Jenis Kelamin	Status Merokok	BMI	Jumlah Klaim	Total Biaya Klaim	Cluster_Labels
0	1	58	0	1	30.7	1	4744854	0
1	2	26	0	1	28.2	0	0	0
2	3	19	0	1	22.3	4	5780796	0
3	4	53	0	0	22.1	4	22669060	1
4	5	69	0	1	30.0	2	11323814	0

4. **Buatkan profiling masing-masing cluster**
   a. **Buatkan profiling menggunakan scaterplot, lakukan perbandingan dengan merubah nilai x dan y lalu berikan kesimpulannya**

```
plt.figure()
sns.scatterplot(data=data,
                x='Usia',
                y='Total Biaya Klaim',
                hue='Cluster_Labels',
                palette='viridis')
plt.title("Hubungan Usia dan Total Biaya Klaim")
plt.xlabel("Usia")
plt.ylabel("Total Biaya Klaim")
plt.show()
```

   b. **Buatkan profiling menggunakan boxplot, lakukan perbandingan dengan merubah nilai y lalu berikan kesimpulannya**

```
sns.boxplot(data=data,
            x='Cluster_Labels',
            y='Usia',
            palette='Set3')
plt.title("rata-rata usia berdasarkan cluster")
plt.xlabel("Cluster")
plt.ylabel("Usia")
plt.show()
```

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
Pokok Pemb : - Membangun Model DBSCAN
- Simulasi Algoritma DBSCAN
- Profiling Hasil Clustering DBSCAN

1. Load Library dan dataset pada file Notebook

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
df = pd.read_csv('net_detection.csv')
```

```python
df.head()
```

0.0s Python

	src_ip	dst_ip	protocol	src_port	dst_port	bytes_sent	bytes_received	duration	packet_count	attack_type
0	39.170.115.188	133.204.219.238	TCP	1762	62458	3422	5989	213	131	normal
1	80.35.125.105	246.113.106.207	TCP	32718	9699	3736	989	277	96	normal
2	49.134.137.30	151.26.62.67	TCP	1225	43970	2865	5943	305	89	ddos
3	157.51.229.193	175.153.3.55	TCP	20804	303	1852	9389	552	80	normal
4	121.123.112.174	72.234.63.118	UDP	15457	17942	8318	5160	533	46	normal

2. **Exploratory data analysis**
   a. **Menghitung jumlah data yang ada pada kolom protocol**

```
# menghitung jumlah data yang ada pada kolom protocol
protocol_counts = df['protocol'].value_counts()
protocol_counts
```

```
# membuat bar plot hasil perhitungan kolom protocol
plt.figure(figsize=(10, 6))
protocol_counts.plot(kind='bar')
plt.title('Number of Data Points per Protocol')
plt.xlabel('Protocol')
plt.ylabel('Count')
plt.show()
```

Lakukan perhitungan untuk mendapatkan jumlah pada kolom “attack_type” dan buatkan bar plot nya

   b. **Membuat box plot untuk menampilkan rata-rata nilai source plot berdasarkan jenis attack type**

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='attack_type', y='src_port', data=df)
plt.title('Source Port Distribution by Attack Type')
plt.xlabel('Attack Type')
plt.ylabel('Source Port')
plt.show()
```

Buatkan boxplot untuk menampilkan bytes_sent dan berdasarkan senis protocol

3. Buat model DBSCAN

```
# pilih fitur untuk clustering
features = ['src_port','dst_port','bytes_sent','bytes_received']
X = df[features]
```

```
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# implementasi algoritma DBSCAN
# tentukan nilai epsilon dan min sample
dbscan = DBSCAN(eps=0.5, min_samples=5)
df['dbscan_cluster'] = dbscan.fit_predict(X_scaled)
```

```
# Plot hasil cluster
plt.figure(figsize=(10, 6))
sns.scatterplot(x='bytes_sent', y='bytes_received',
                hue='dbscan_cluster', data=df,
                palette='viridis')
plt.title('DBSCAN Clustering Results')
plt.show()
```

```
df.head()
```
✓ 0.0s

	src_ip	dst_ip	protocol	src_port	dst_port	bytes_sent	bytes_received	duration	packet_count	attack_type	dbscan_cluster
0	39.170.115.188	133.204.219.238	TCP	1762	62458	3422	5989	213	131	normal	-1
1	80.35.125.105	246.113.106.207	TCP	32718	9699	3736	989	277	96	normal	0
2	49.134.137.30	151.26.62.67	TCP	1225	43970	2865	5943	305	89	ddos	-1
3	157.51.229.193	175.153.3.55	TCP	20804	303	1852	9389	552	80	normal	-1
4	121.123.112.174	72.234.63.118	UDP	15457	17942	8318	5160	533	46	normal	0

```python
# memisahkan data anomaly yang bernilai -1
df['anomaly'] = df['dbscan_cluster'] == -1
```

```python
# Plot hasil deteksi anomaly
plt.figure(figsize=(10, 6))
sns.scatterplot(x='bytes_sent', y='bytes_received',
                hue='anomaly', data=df,
                palette=['blue', 'red'])
plt.title('Anomaly Detection')
plt.show()
```

```
df.head()
```
✓ 0.0s Python

	src_ip	dst_ip	protocol	src_port	dst_port	bytes_sent	bytes_received	duration	packet_count	attack_type	dbscan_cluster	anomaly
0	39.170.115.188	133.204.219.238	TCP	1762	62458	3422	5989	213	131	normal	-1	True
1	80.35.125.105	246.113.106.207	TCP	32718	9699	3736	989	277	96	normal	0	False
2	49.134.137.30	151.26.62.67	TCP	1225	43970	2865	5943	305	89	ddos	-1	True
3	157.51.229.193	175.153.3.55	TCP	20804	303	1852	9389	552	80	normal	-1	True
4	121.123.112.174	72.234.63.118	UDP	15457	17942	8318	5160	533	46	normal	0	False

```python
# simpan hasil cluster
df.to_csv('hasil_dbscan.csv', index=False)
```

4. Buatkan profiling hasil cluster

```
# membuat df untuk type serangan ddos
ddos_df = df[df['attack_type'] == 'ddos']
```

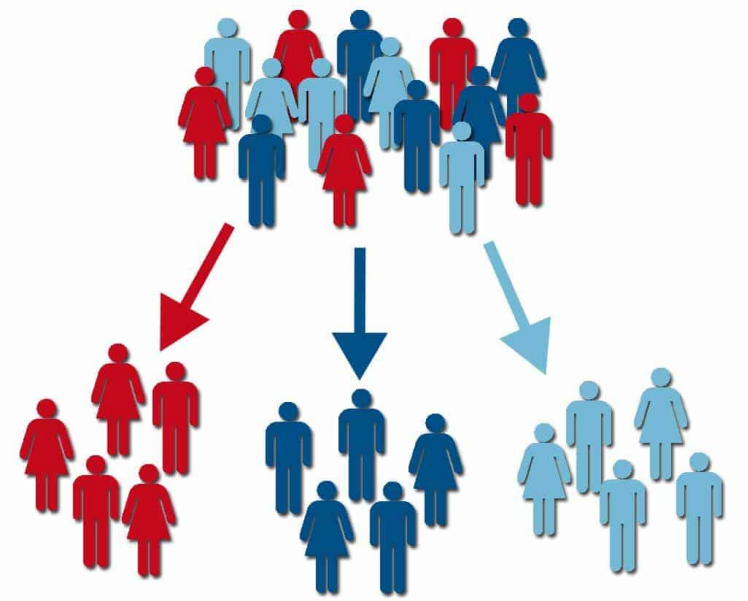
```
# hitung jumlah anomalynya
anomaly_counts = ddos_df['anomaly'].value_counts()
anomaly_counts
```

```
# bar plot untuk serangan DDOS
plt.figure(figsize=(8, 6))
anomaly_counts.plot(kind='bar')
plt.title('Jumlah Anomaly untu tipe DDoS')
plt.xlabel('Anomaly')
plt.ylabel('Count')
plt.show()
```

Lakukan analisis untuk deteksi anomaly pada jenis serangan lainnya seperti brute force, icmp flood, dan port scan.

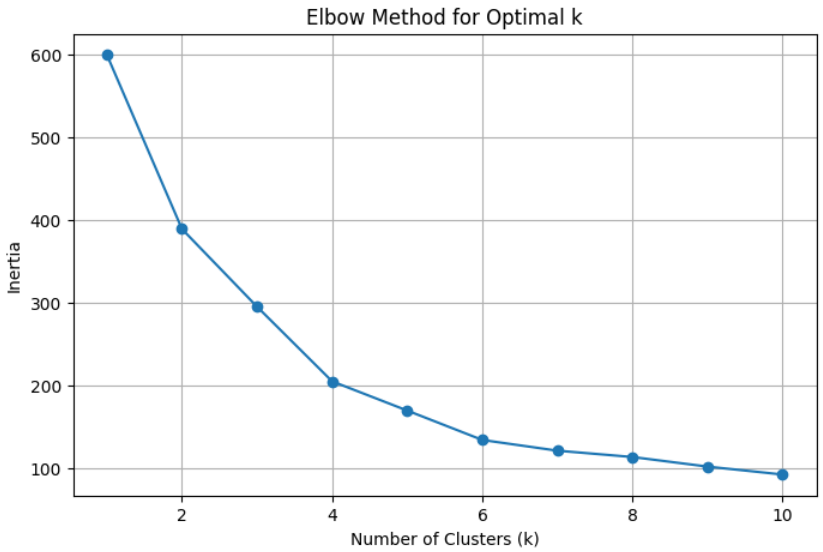
Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Komparasi Algoritma Clustering
Pokok Pemb : - Membangun Model K-Means
- Membangun Model Hierarchical Clustering
- Membangun Model DBSCAN

1. Load dataset pada file Notebook

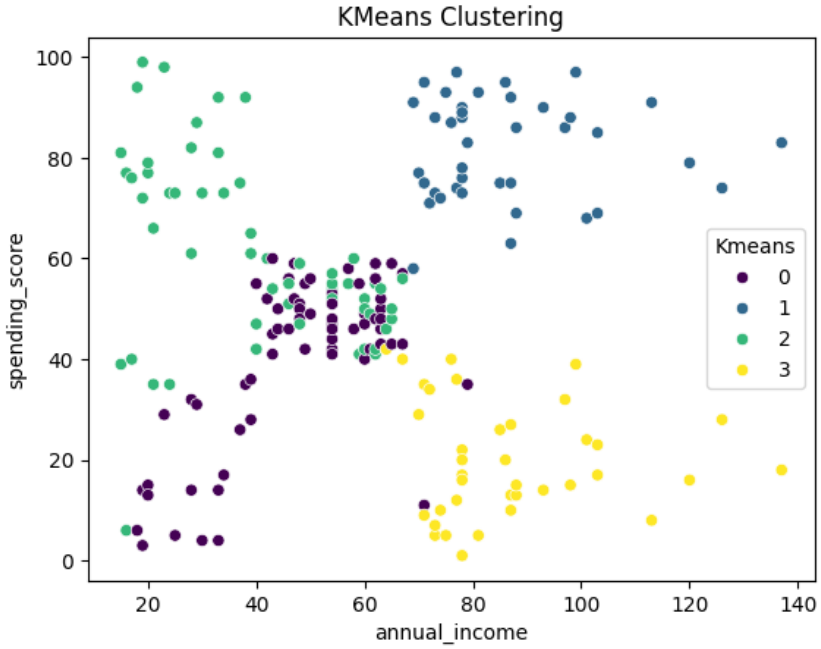


	customer_id	gender	age	annual_income	spending_score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

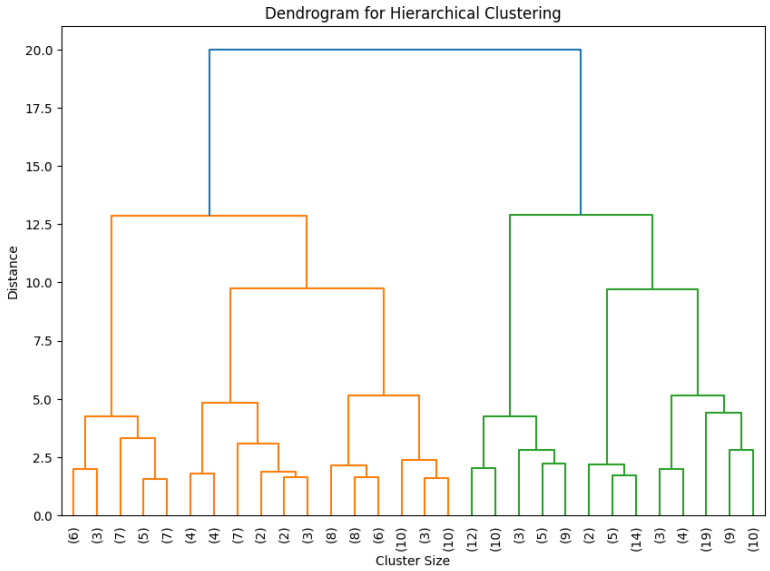
2. Membangun Model K-Means



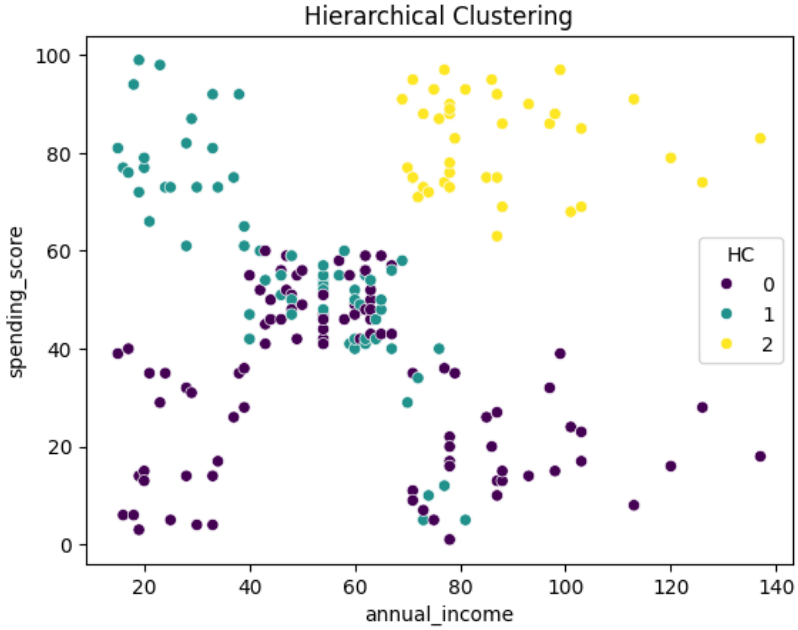
	customer_id	gender	age	annual_income	spending_score	Kmeans
0	1	Male	19	15	39	2
1	2	Male	21	15	81	2
2	3	Female	20	16	6	2
3	4	Female	23	16	77	2
4	5	Female	31	17	40	2



3. Membangun model Hierarchical Clusering

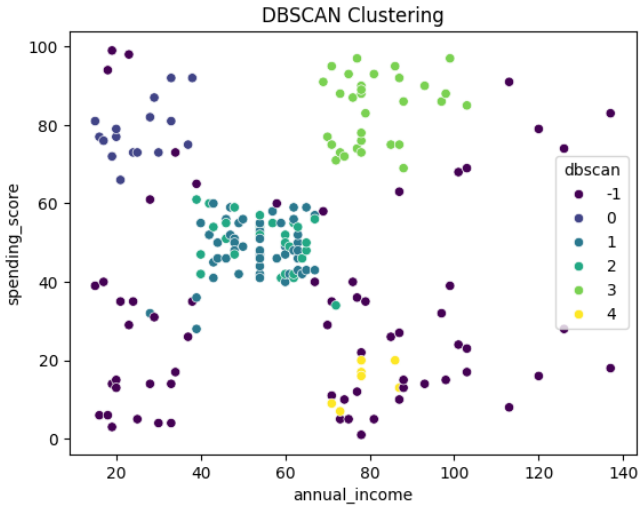


	customer_id	gender	age	annual_income	spending_score	Kmeans	HC
0	1	Male	19	15	39	2	0
1	2	Male	21	15	81	2	1
2	3	Female	20	16	6	2	0
3	4	Female	23	16	77	2	1
4	5	Female	31	17	40	2	0

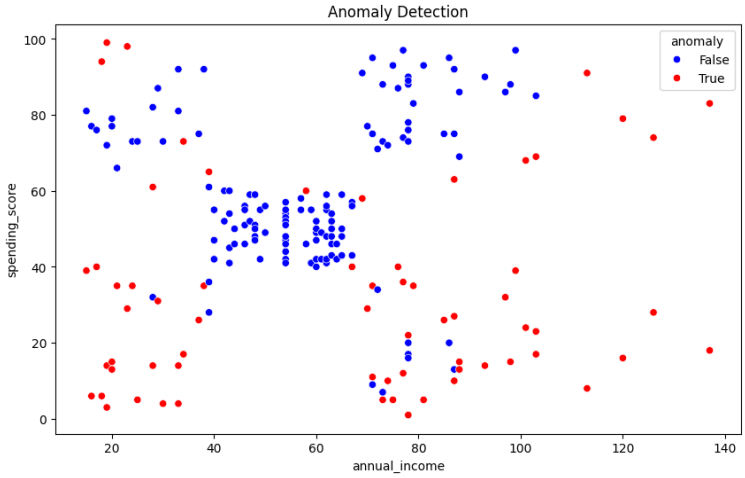


4. Membangun Model DBSCAN

	customer_id	gender	age	annual_income	spending_score	Kmeans	HC	dbscan
0	1	Male	19	15	39	2	0	-1
1	2	Male	21	15	81	2	1	0
2	3	Female	20	16	6	2	0	-1
3	4	Female	23	16	77	2	1	0
4	5	Female	31	17	40	2	0	-1

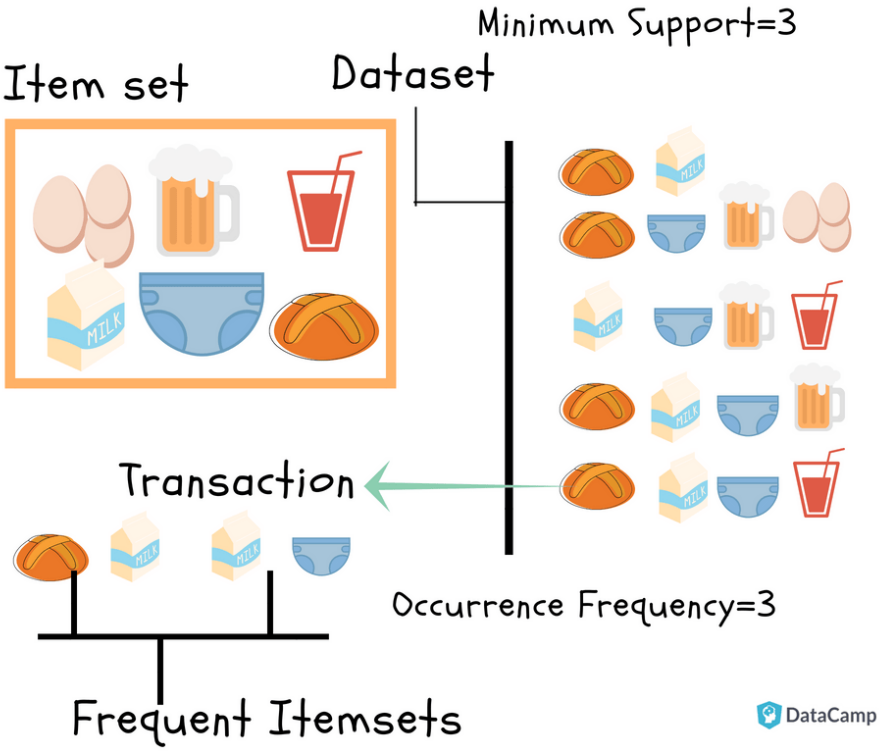


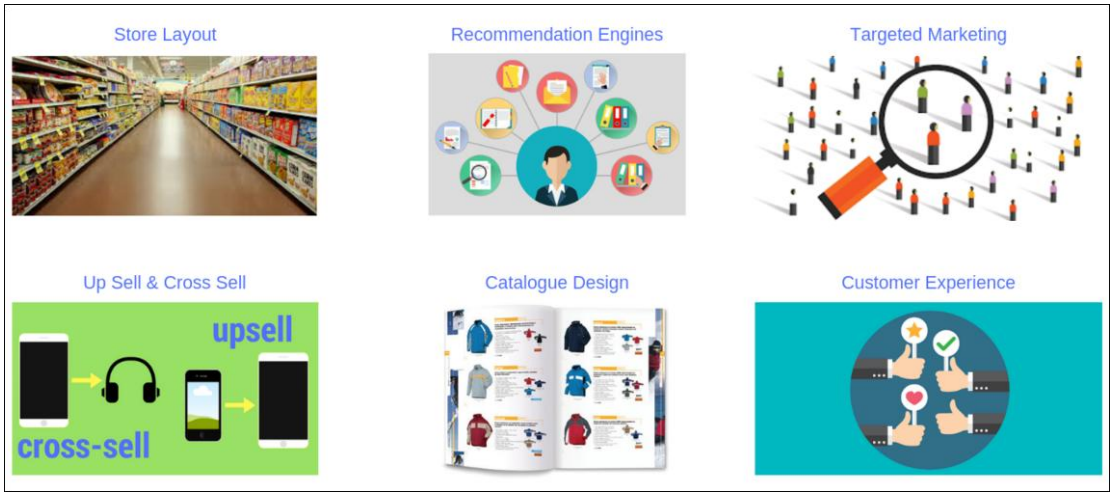
	customer_id	gender	age	annual_income	spending_score	Kmeans	HC	dbscan	anomaly
0	1	Male	19	15	39	2	0	-1	True
1	2	Male	21	15	81	2	1	0	False
2	3	Female	20	16	6	2	0	-1	True
3	4	Female	23	16	77	2	1	0	False
4	5	Female	31	17	40	2	0	-1	True



Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Algoritma Association Rules
Pokok Pemb : - Mengenal Market Basket analysis
- Membangun Model Algoritma Apriori
- Membangun Model Algoritma FP-Growth

1. Market Basket Analysis





2. Load dataset pada file Notebook

```python
import pandas as pd
import matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori, fpgrowth
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
```

```python
# Load data
file_path = "Groceries_dataset.csv"
data = pd.read_csv(file_path)
```

3. Analisa Data

```python
# Tampilkan informasi awal dataset
print("Preview Dataset:")
print(data.head())
print("\n\nInfo Dataset:")
print(data.info())
print("\nStatistik Deskriptif Kolom Numerik:")
print(data.describe())
```

```python
# 1. Analisis kolom 'Member_number'
# Menghitung total member
total_member = data['Member_number'].nunique()
print(f"\n\nTotal member: {total_member}")
```

```python
# Top 5 anggota dengan transaksi terbanyak
top_members = data['Member_number'].value_counts().head()
print("\nTop 5 anggota dengan transaksi terbanyak:")
print(top_members)
```

```python
# 2. Analisis Kolom `Date`
data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')

# Rentang waktu dataset
start_date = data['Date'].min()
end_date = data['Date'].max()
print(f"\nRentang waktu data: {start_date} hingga {end_date}")
```

```python
# Hitung jumlah transaksi per bulan
monthly_transactions = data['Date'].dt.to_period('M').value_counts().sort_index()
print("\nJumlah transaksi per bulan:")
print(monthly_transactions)
```

```python
# Visualisasi jumlah transaksi per bulan
plt.figure(figsize=(12, 6))
monthly_transactions.plot(kind='bar',
                          title='Transaksi per Bulan',
                          color='skyblue')
plt.xlabel('Bulan')
plt.ylabel('Jumlah Transaksi')
plt.xticks(rotation=45)
plt.show()
```

```python
# 3. Analisis Kolom `itemDescription`
# Barang paling sering dibeli
item_counts = data['itemDescription'].value_counts()
print("\nTop 10 barang paling sering dibeli:")
print(item_counts.head(10))
```

```python
# Visualisasi 10 barang teratas
plt.figure(figsize=(10, 6))
item_counts.head(10).plot(kind='bar',
                          title='Top 10 Barang Paling Sering Dibeli',
                          color='orange')
plt.xlabel('Barang')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)
plt.show()
```

```python
# 4. Analisis Gabungan
# Barang yang paling sering dibeli oleh anggota tertentu
member_id = 1808
member_items = data[data['Member_number'] == member_id]['itemDescription'].value_counts()
print(f"\nBarang paling sering dibeli oleh anggota {member_id}:")
print(member_items)
```

```python
# Pola pembelian barang pada bulan tertentu
month = '2015-07'
monthly_items = data[data['Date'].dt.to_period('M') == month]['itemDescription'].value_counts()
print(f"\nBarang paling sering dibeli pada {month}:")
print(monthly_items)
```

```python
# Statistik tambahan: jumlah barang unik
print("\nJumlah barang unik:")
print(data['itemDescription'].nunique())
```

4. Data Preprocessing

```
# Persiapan data: Mengelompokan item berdasarkan Member_number dan Date
transactions = data.groupby(['Member_number', 'Date'])['itemDescription'].apply(list)
```

```
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
```

5. Modeling

```
# --- Algoritma Apriori ---
print("\n===== Apriori =====")
frequent_itemsets_apriori = apriori(df_encoded, min_support=0.005, use_colnames=True)
print("Frequent Itemsets (Apriori):")
print(frequent_itemsets_apriori)
```

```
# --- Algoritma FP-Growth ---
print("\n===== FP-Growth =====")
frequent_itemsets_fpgrowth = fpgrowth(df_encoded, min_support=0.005, use_colnames=True)
print("Frequent Itemsets (FP-Growth):")
print(frequent_itemsets_fpgrowth)
```

```
# Menampilkan hasil aturan asosiasi
print("\nAturan Asosiasi (Apriori):")
rules_apriori = association_rules(frequent_itemsets_apriori, metric="confidence", min_threshold=0.1)
print(rules_apriori[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
print("\nAturan Asosiasi (FP-Growth):")
rules_fpgrowth = association_rules(frequent_itemsets_fpgrowth, metric="confidence", min_threshold=0.1)
print(rules_fpgrowth[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```