1 . Fitting Linear regression model on real data set

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.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Load dataset

df = pd.read\_csv('path\_to\_your\_dataset.csv')

# Define features and target variable

X = df[['feature1', 'feature2', 'feature3']] # replace with your features

y = df['target'] # replace with your target variable

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)

# Make predictions

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

print(f'R^2 Score: {r2}')

2. Learn Decision trees for regression and classification problem

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

df = pd.read\_csv('path\_to\_your\_regression\_dataset.csv')

print(df.head())

X = df[['feature1', 'feature2', 'feature3']] # Replace with relevant feature names

y = df['target'] # Replace with your target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

dt\_regressor = DecisionTreeRegressor(random\_state=42)

dt\_regressor.fit(X\_train, y\_train)

y\_pred = dt\_regressor.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print('Mean Squared Error:', mse)

print('R^2 Score:', r2)

plt.scatter(y\_test, y\_pred)

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title('Decision Tree Regression: Actual vs Predicted')

plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red')

plt.show()

3. Prediction based on KNN Regression

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

df = pd.read\_csv('path\_to\_your\_regression\_dataset.csv')

# Define features (X) and target variable (y)

X = df[['feature1', 'feature2', 'feature3']] # Replace with your feature column names

y = df['target'] # Replace with your target column name

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

knn\_regressor = KNeighborsRegressor(n\_neighbors=5) # You can choose the number of neighbors

knn\_regressor.fit(X\_train, y\_train)

y\_pred = knn\_regressor.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print('Mean Squared Error:', mse)

print('R² Score:', r2)

4. Fitting Logistic regression model on real data set for classification problem

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.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

from sklearn.preprocessing import StandardScaler

df = pd.read\_csv('path\_to\_your\_dataset.csv')

print(df.head())

# df = pd.get\_dummies(df, columns=['categorical\_column\_name'])

X = df.drop('target\_column', axis=1) # Replace with your target column name

y = df['target\_column'] # Replace with your target column name

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

log\_reg = LogisticRegression()

y\_pred = log\_reg.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print('Accuracy:', accuracy)

5. Application of Clustering Techniques in Data Analytics

### 1. \*\*Customer Segmentation\*\*

- \*\*Application\*\*: Businesses use clustering to segment their customers into groups based on purchasing behavior, demographics, and preferences.

### 3. \*\*Anomaly Detection\*\*

- \*\*Application\*\*: Clustering techniques can help detect outliers or anomalies in data (e.g., fraudulent transactions in finance).

### 5. \*\*Social Network Analysis\*\*

- \*\*Application\*\*: Clustering helps analyze social networks by identifying communities or groups of people with similar interests or behaviors.

### 6. \*\*Document Clustering\*\*

- \*\*Application\*\*: Clustering is utilized in text mining to group similar documents or articles based on content.

### 7. \*\*Recommendation Systems\*\*

- \*\*Application\*\*: Clustering techniques are employed to improve recommendation engines by grouping users or items based on preferences and interactions.

### 8. \*\*Healthcare and Bioinformatics\*\*

- \*\*Application\*\*: Clustering is used in genomics to group genes or patients with similar characteristics based on genetic data.

### 9. \*\*Geospatial Data Analysis\*\*

- \*\*Application\*\*: Clustering assist in analyzing geographical data to identify patterns, such as crime hotspots or optimal locations for new stores.

### 10. \*\*Time Series Clustering\*\*

- \*\*Application\*\*: Clustering can help group time series data into patterns to identify trends or behaviors over time.