slip4

Q1)

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

df = pd.read_csv('mall_customers.csv')

print("\nFirst few rows of the dataset:")
print(df.head())

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X_scaled)

print("\nClusters assigned to each customer:")
print(df[['CustomerID', 'Cluster']].head())

plt.figure(figsize=(8, 6))
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'],
c=df['Cluster'], cmap='viridis')
plt.title('K-Means Clustering of Mall Customers')
```

```
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.colorbar(label='Cluster')
plt.show()
```

Q2)

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('house_price.csv')
print("\nFirst few rows of the dataset:")
print(df.head())
df = df.dropna()

X = df[['SquareFootage']]
y = df['Price']
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)
```

```
mse = mean_squared_error(y_test, y_pred)
print("\nMean Squared Error (MSE):", mse)

plt.figure(figsize=(8,6))
plt.scatter(X_test, y_test, color='blue', label='Actual Prices')
plt.plot(X_test, y_pred, color='red', label='Predicted Prices')
plt.title('Simple Linear Regression: House Price Prediction')
plt.xlabel('Square Footage')
plt.ylabel('Price')
plt.legend()
plt.show()
```

SLIP 5

Q1)

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('fuel_consumption.csv')
print("\nMissing values in the dataset:")
print(df.isnull().sum())
```

```
X = df[['EngineSize', 'Cylinders', 'Horsepower', 'Weight', 'Acceleration']]
y = df['FuelConsumption']  # 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)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("\nMean Squared Error (MSE):", mse)

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
linewidth=2)
plt.title('Actual vs Predicted Fuel Consumption')
plt.xlabel('Actual Fuel Consumption')
plt.ylabel('Predicted Fuel Consumption')
plt.show()
```

Q2)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target

X = df.drop('species', axis=1)  # Features
y = df['species']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the KNN model: {accuracy:.2f}")
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

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

```
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for KNN Classifier')
plt.show()
```

SLIP 6

Q1)

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
df['PRICE'] = housing.target
print(df.head())

X = df.drop('PRICE', axis=1)  # Drop the target column to get features
y = df['PRICE']  # Target variable (house prices)
```

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

poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model = LinearRegression()
model.fit(X_train_poly, y_train)
y_pred = model.predict(X_test_poly)

mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")

plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices (Polynomial Regression)")
plt.show()
```

Q2)

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import warnings
```

```
warnings.filterwarnings("ignore", category=FutureWarning)
df = pd.read_csv('employees.csv')

print("Checking for missing values in the dataset:")
print(df.isnull().sum())

df_cleaned = df.dropna()
print("\nData after removing rows with missing values:")
print(df_cleaned.isnull().sum())

X = df_cleaned[['Income', 'Age']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, n_init=10, random_state=42)
kmeans.fit(X_scaled)
df_cleaned['Cluster'] = kmeans.labels_

print("\nClustered data with labels:")
print(df_cleaned.head())
df_cleaned.to_csv('employees_with_clusters.csv', index=False)

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(df_cleaned['Income'], df_cleaned['Age'], c=df_cleaned['Cluster'],
cmap='viridis')
```

```
plt.title('K-Means Clustering of Employees')
plt.xlabel('Income')
plt.ylabel('Age')
plt.colorbar(label='Cluster')
plt.show()
```

SLIP 7

Q1)

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

df = pd.read_csv('Salary_positions.csv')
print(df.isnull().sum())
df = df.dropna()

X = df[['Position_Level']]
y = df['Salary']
model = LinearRegression()
model.fit(X, y)

levels_to_predict = pd.DataFrame([[11], [12]], columns=['Position_Level'])
predicted_salaries = model.predict(levels_to_predict)

for level, salary in zip([11, 12], predicted_salaries):
```

```python
    print(f"Predicted salary for Level {level}: ${salary:.2f}")

plt.scatter(X, y, color='blue')
plt.plot(X, model.predict(X), color='red')
plt.title('Salary vs Position Level')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()
```

Q2)

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('weather.csv')
label_encoder = LabelEncoder()

df['Outlook'] = label_encoder.fit_transform(df['Outlook'])
df['Temperature'] = label_encoder.fit_transform(df['Temperature'])
df['Humidity'] = label_encoder.fit_transform(df['Humidity'])
df['Wind'] = label_encoder.fit_transform(df['Wind'])
df['PlayTennis'] = label_encoder.fit_transform(df['PlayTennis'])
```

```
X = df.drop('PlayTennis', axis=1)
y = df['PlayTennis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)
y_pred = naive_bayes_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy of Naive Bayes Model: {accuracy * 100:.2f}%")
print("Confusion Matrix:")
print(conf_matrix)
```

SLIP 8

Q1)

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score

newsgroups = fetch_20newsgroups(subset='all')
```

```
X, y = newsgroups.data, newsgroups.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

vectorizer = TfidfVectorizer(stop_words='english', max_df=0.5)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

model = MultinomialNB()
model.fit(X_train_tfidf, y_train)
y_pred = model.predict(X_test_tfidf)
print("Classification Report:\n", classification_report(y_test, y_pred,
target_names=newsgroups.target_names))

accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")

def categorize_news(text):
    text_tfidf = vectorizer.transform([text])
    category_index = model.predict(text_tfidf)[0]
    return newsgroups.target_names[category_index]

sample_text = "The government has announced new tax reforms for the next
fiscal year."
category = categorize_news(sample_text)
print(f"The category of the given text is: {category}")
```

SLIP 9

Q1)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error

df = pd.read_csv('boston_houses.csv')
X = df[['RM']]
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

ridge = Ridge(alpha=1.0)
lasso = Lasso(alpha=0.1)
ridge.fit(X_train, y_train)
lasso.fit(X_train, y_train)

ridge_predictions = ridge.predict(X_test)
lasso_predictions = lasso.predict(X_test)
ridge_mse = mean_squared_error(y_test, ridge_predictions)
lasso_mse = mean_squared_error(y_test, lasso_predictions)

print(f"Ridge Regression Mean Squared Error: {ridge_mse}")
print(f"Lasso Regression Mean Squared Error: {lasso_mse}")
room_count = pd.DataFrame([[5]], columns=['RM'])
```

```
ridge_price = ridge.predict(room_count)
lasso_price = lasso.predict(room_count)

print(f"Predicted Price of a house with 5 rooms (Ridge): {ridge_price[0]}")
print(f"Predicted Price of a house with 5 rooms (Lasso): {lasso_price[0]}")
```

Q2)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

df = pd.read_csv('UniversalBank.csv')
df = df.drop(['ID', 'ZIP Code'], axis=1)
label_encoder = LabelEncoder()
df['Education'] = label_encoder.fit_transform(df['Education'])

X = df.drop('PersonalLoan', axis=1)
y = df['PersonalLoan']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train_scaled, y_train)
y_pred = svm.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

SLIP 10

Q1)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
```

```
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

pca_df = pd.DataFrame(X_pca, columns=['Principal Component 1', 'Principal Component 2'])
pca_df['Target'] = y

plt.figure(figsize=(8, 6))
plt.scatter(pca_df['Principal Component 1'], pca_df['Principal Component 2'], c=pca_df['Target'], cmap='viridis', edgecolor='k', s=50)
plt.title('PCA of Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Target (Species)')
plt.show()

print("Explained variance ratio of the components:")
print(pca.explained_variance_ratio_)
```

```
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
print("\nCumulative explained variance:")
print(cumulative_variance)
```

Q2)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import LabelEncoder

iris = load_iris()
X = iris.data
y = iris.target

label_encoder = LabelEncoder()
y_numeric = label_encoder.fit_transform(y)

df = pd.DataFrame(X, columns=iris.feature_names)
df['Species'] = y_numeric
plt.figure(figsize=(10, 6))

plt.subplot(1, 2, 1)
plt.scatter(df['sepal length (cm)'], df['sepal width (cm)'], c=df['Species'],
cmap='viridis')
plt.title('Sepal Length vs Sepal Width')
```

```
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')

plt.subplot(1, 2, 2)
plt.scatter(df['petal length (cm)'], df['petal width (cm)'], c=df['Species'],
cmap='viridis')
plt.title('Petal Length vs Petal Width')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')

plt.tight_layout()
plt.show()
```

SLIP 11

Q1)

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
```

```
df['PRICE'] = housing.target
print(df.head())

X = df.drop('PRICE', axis=1)
y = df['PRICE']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
poly = PolynomialFeatures(degree=2)

X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
model = LinearRegression()
model.fit(X_train_poly, y_train)

y_pred = model.predict(X_test_poly)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")

plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices (Polynomial Regression)")
plt.show()
```

Q2)

```
import pandas as pd
```

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

url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/00267/data_banknote_authentication.txt"
column_names = ["variance", "skewness", "curtosis", "entropy", "class"]
df = pd.read_csv(url, header=None, names=column_names)

X = df.drop("class", axis=1)
y = df["class"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)
y_pred = dt_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

SLIP 12

Q1)

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Q2)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error

data = pd.read_csv("Salary_positions.csv")
X = data[['Position_Level']].values
y = data['Salary'].values

linear_model = LinearRegression()
linear_model.fit(X, y)
poly_features = PolynomialFeatures(degree=4)
X_poly = poly_features.fit_transform(X)
poly_model = LinearRegression()
poly_model.fit(X_poly, y)

y_pred_linear = linear_model.predict(X)
mse_linear = mean_squared_error(y, y_pred_linear)
y_pred_poly = poly_model.predict(X_poly)
mse_poly = mean_squared_error(y, y_pred_poly)

print(f"Linear Regression MSE: {mse_linear:.2f}")
print(f"Polynomial Regression (Degree 4) MSE: {mse_poly:.2f}")

plt.scatter(X, y, color="blue", label="Actual Salary Data")
```

```
plt.plot(X, y_pred_linear, color="red", label="Simple Linear Regression")
plt.plot(X, y_pred_poly, color="green", label="Polynomial Regression (Degree 4)")
plt.xlabel("Level")
plt.ylabel("Salary")
plt.legend()
plt.show()

level_11 = np.array([[11]])
level_12 = np.array([[12]])

salary_11_linear = linear_model.predict(level_11)[0]
salary_12_linear = linear_model.predict(level_12)[0]

salary_11_poly = poly_model.predict(poly_features.transform(level_11))[0]
salary_12_poly = poly_model.predict(poly_features.transform(level_12))[0]

print(f"\nPredicted Salary for Level 11 (Linear): {salary_11_linear:.2f}")
print(f"Predicted Salary for Level 11 (Polynomial): {salary_11_poly:.2f}")

print(f"Predicted Salary for Level 12 (Linear): {salary_12_linear:.2f}")
print(f"Predicted Salary for Level 12 (Polynomial): {salary_12_poly:.2f}")
```

SLIP 14

Q2)

```
import pandas as pd
```

```
import numpy as np

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', np.nan],
    'Age': [24, np.nan, 22, 32, 29, 27],
    'City': ['New York', 'Los Angeles', 'Chicago', np.nan, 'Houston', 'Phoenix'],
    'Salary': [70000, 80000, np.nan, 54000, 62000, 67000]
}

df = pd.DataFrame(data)
print("Original Dataset with Null Values:")
print(df)

print("\nNull Values in Each Column:")
print(df.isnull().sum())

df_cleaned = df.dropna()
print("\nDataset after Removing Rows with Null Values:")
print(df_cleaned)
```

SLIP 15

Q1)

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import fetch_california_housing
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score

data = fetch_california_housing()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target
average_price = y.mean()

y_binary = (y > average_price).astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y_binary,test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = Sequential()

model.add(Dense(16, input_dim=X_train_scaled.shape[1], activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X_train_scaled, y_train, epochs=50, batch_size=32, verbose=1,
    validation_split=0.2)
```

```
loss, accuracy = model.evaluate(X_test_scaled, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

y_pred = (model.predict(X_test_scaled) > 0.5).astype("int32")

accuracy_test = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Set: {accuracy_test:.4f}")
```

Q2)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = {
    'Size': [1500, 2000, 2500, 1800, 2200, 1700],
    'Bedrooms': [3, 4, 3, 2, 4, 3],
    'Age': [20, 15, 10, 30, 8, 12],
    'Price': [300000, 400000, 450000, 350000, 500000, 320000]
}

df = pd.DataFrame(data)
X = df[['Size', 'Bedrooms', 'Age']]
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

```
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Evaluation:")
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")
print("\nCoefficients:", model.coef_)
print("Intercept:", model.intercept_)

new_house = pd.DataFrame({'Size': [2000], 'Bedrooms': [3], 'Age': [10]})
predicted_price = model.predict(new_house)
print(f"\nPredicted Price for new house (Size=2000, Bedrooms=3, Age=10):
${predicted_price[0]:,.2f}")
```

SLIP 16

Q2)

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
df['MedHouseVal'] = housing.target

X = df[['AveRooms']]
y = df['MedHouseVal']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model = LinearRegression()
model.fit(X_train_poly, y_train)
y_pred = model.predict(X_test_poly)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Polynomial Regression Model Evaluation:")
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
plt.scatter(X, y, color='blue', label='Data Points')
X_fit = pd.DataFrame(np.linspace(X.min(), X.max(), 100), columns=['AveRooms'])
X_fit_poly = poly.transform(X_fit)
y_fit = model.predict(X_fit_poly)
plt.plot(X_fit, y_fit, color='red', label='Polynomial Fit (Degree 2)')
plt.xlabel("Average number of rooms per household (AveRooms)")
plt.ylabel("Median House Value (MedHouseVal)")
plt.title("Polynomial Regression Fit for California Housing Data")
plt.legend()
plt.show()
```

SLIP 19

Q1)

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Q2)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('weather.csv')
label_encoder = LabelEncoder()

df['Outlook'] = label_encoder.fit_transform(df['Outlook'])
df['Temperature'] = label_encoder.fit_transform(df['Temperature'])
df['Humidity'] = label_encoder.fit_transform(df['Humidity'])
df['Wind'] = label_encoder.fit_transform(df['Wind'])
```

```
df['PlayTennis'] = label_encoder.fit_transform(df['PlayTennis'])

X = df.drop('PlayTennis', axis=1)
y = df['PlayTennis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)
y_pred = naive_bayes_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy of Naive Bayes Model: {accuracy * 100:.2f}%")
print("Confusion Matrix:")
print(conf_matrix)
```

SLIP 23

Q1)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

```
from sklearn.metrics import mean_squared_error

df = pd.read_csv('Salary_positions.csv')

X = df['Position Level'].values.reshape(-1, 1)
y = df['Salary'].values

simple_linear_reg = LinearRegression()
simple_linear_reg.fit(X, y)

poly = PolynomialFeatures(degree=4)
X_poly = poly.fit_transform(X)
poly_linear_reg = LinearRegression()
poly_linear_reg.fit(X_poly, y)

y_pred_simple = simple_linear_reg.predict(X)
y_pred_poly = poly_linear_reg.predict(X_poly)
mse_simple = mean_squared_error(y, y_pred_simple)
mse_poly = mean_squared_error(y, y_pred_poly)
print(f"Mean Squared Error for Simple Linear Regression: {mse_simple}")
print(f"Mean Squared Error for Polynomial Linear Regression: {mse_poly}")

level_11 = np.array([[11]])
level_12 = np.array([[12]])

salary_11_simple = simple_linear_reg.predict(level_11)
salary_12_simple = simple_linear_reg.predict(level_12)
```

```
salary_11_poly = poly_linear_reg.predict(poly.transform(level_11))
salary_12_poly = poly_linear_reg.predict(poly.transform(level_12))
print(f"Predicted Salary for Level 11 (Simple Linear Regression):
{salary_11_simple[0]}")
print(f"Predicted Salary for Level 12 (Simple Linear Regression):
{salary_12_simple[0]}")
print(f"Predicted Salary for Level 11 (Polynomial Linear Regression):
{salary_11_poly[0]}")
print(f"Predicted Salary for Level 12 (Polynomial Linear Regression):
{salary_12_poly[0]}")

plt.scatter(X, y, color='blue')
plt.plot(X, y_pred_simple, color='red')
plt.title('Simple Linear Regression')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()

plt.scatter(X, y, color='blue')
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.plot(X_grid, poly_linear_reg.predict(poly.transform(X_grid)), color='red')
plt.title('Polynomial Linear Regression')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()
```

Q2)

```
import pandas as pd
import numpy as np

data = {
    'EmployeeID': [1, 2, 3, 4, 5],
    'Name': ['Alice', 'Bob', np.nan, 'David', 'Eva'],
    'Age': [25, np.nan, 29, 40, 35],
    'Department': ['HR', 'Finance', 'IT', np.nan, 'Marketing'],
    'Salary': [50000, 60000, 55000, 65000, np.nan]
}
df = pd.DataFrame(data)
df.to_csv("your_dataset.csv", index=False)
print("Sample CSV file 'your_dataset.csv' created with dummy data and null values.")
```

SLIP 24

Q2)

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

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

df = pd.read_csv('UniversalBank.csv')
df = df.drop(['ID', 'ZIP Code'], axis=1)

label_encoder = LabelEncoder()
df['Education'] = label_encoder.fit_transform(df['Education'])

X = df.drop('PersonalLoan', axis=1)
y = df['PersonalLoan']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train_scaled, y_train)
y_pred = svm.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

SLIP 28

Q1)

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score

newsgroups = fetch_20newsgroups(subset='all')
X, y = newsgroups.data, newsgroups.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

vectorizer = TfidfVectorizer(stop_words='english', max_df=0.5)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

model = MultinomialNB()
model.fit(X_train_tfidf, y_train)
y_pred = model.predict(X_test_tfidf)
print("Classification Report:\n", classification_report(y_test, y_pred,
target_names=newsgroups.target_names))
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")

def categorize_news(text):
    text_tfidf = vectorizer.transform([text])
    category_index = model.predict(text_tfidf)[0]
    return newsgroups.target_names[category_index]

sample_text = "The government has announced new tax reforms for the next fiscal year."
category = categorize_news(sample_text)
print(f"The category of the given text is: {category}")
```

Q2)

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = iris.target

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

```
svm_kernels = {
        'linear': SVC(kernel='linear'),
        'poly': SVC(kernel='poly', degree=3),
        'rbf': SVC(kernel='rbf')
        }
accuracy_results = {}
for kernel, model in svm_kernels.items():
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)
  accuracy_results[kernel] = accuracy
for kernel, accuracy in accuracy_results.items():
  print(f'Accuracy of SVM with {kernel} kernel: {accuracy:.4f}')
```