---

1. Simple Linear Regression using Scikit-learn (Boston Housing Dataset)

We'll predict house prices based on features using a linear regression model.

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

# Load the Boston Housing dataset
boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df['PRICE'] = boston.target

# Selecting a single feature (RM - Average number of rooms per dwelling)
X = df[['RM']]
y = df['PRICE']

# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting
```

```
y_pred = model.predict(X_test)

# Model evaluation
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))

# Plotting the regression line
plt.scatter(X_test, y_test, color="blue", label="Actual Prices")
plt.plot(X_test, y_pred, color="red", linewidth=2, label="Predicted Prices")
plt.xlabel("Number of Rooms (RM)")
plt.ylabel("House Price")
plt.legend()
plt.show()
```

---

2. Classification using K-Nearest Neighbors (KNN) for Spam Detection

We'll classify emails as spam or not using a labeled dataset.

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import pandas as pd

# Load sample dataset (assumes 'label' as target and 'message' as text data)
df = pd.read_csv('spam.csv', encoding='latin-1')

# Preprocessing dataset
df = df[['v1', 'v2']]  # Selecting relevant columns
df.columns = ['label', 'message']
df['label'] = df['label'].map({'ham': 0, 'spam': 1})  # Convert labels to 0 (ham) and 1 (spam)

# Text processing
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(df['message'])
y = df['label']

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

# Training the KNN model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predicting
```

```
y_pred = knn.predict(X_test)


# Model evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
```

---

3. Data Preprocessing Steps in Machine Learning

We'll handle missing data, encode categorical features, scale numerical features, and split the data.

```
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split

# Sample dataset
data = {
    'Age': [25, 30, np.nan, 35, 40],
    'Salary': [50000, 60000, 75000, np.nan, 100000],
    'Country': ['USA', 'UK', 'India', 'UK', 'India'],
    'Purchased': ['Yes', 'No', 'Yes', 'No', 'Yes']
}
df = pd.DataFrame(data)

# Handling missing values
imputer = SimpleImputer(strategy='mean')
df[['Age', 'Salary']] = imputer.fit_transform(df[['Age', 'Salary']])

# Encoding categorical variables
encoder = OneHotEncoder(drop='first', sparse_output=False)
encoded_features = encoder.fit_transform(df[['Country']])
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(['Country']))

# Combining datasets
df = df.drop(['Country'], axis=1)
df = pd.concat([df, encoded_df], axis=1)
```

```
# Converting target variable to numerical
df['Purchased'] = df['Purchased'].map({'No': 0, 'Yes': 1})

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

# Scaling numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print("Processed dataset:\n", pd.DataFrame(X_train).head())
```