Python Code Tasks

# 1. Linear Regression using Scikit-learn

# Task 1: Data Preparation  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Create a simple dataset  
data = {'X': [1, 2, 3, 4, 5], 'Y': [2, 4, 5, 4, 5]}  
df = pd.DataFrame(data)  
  
# Visualize the data  
plt.scatter(df['X'], df['Y'])  
plt.title('Original Data')  
plt.xlabel('X')  
plt.ylabel('Y')  
plt.show()  
  
# Task 2: Train the Linear Regression Model  
X = df[['X']]  
y = df['Y']  
model = LinearRegression()  
model.fit(X, y)  
print("Slope (m):", model.coef\_[0])  
print("Intercept (c):", model.intercept\_)  
  
# Task 3: Predictions and Evaluation  
y\_pred = model.predict(X)  
mse = mean\_squared\_error(y, y\_pred)  
r2 = r2\_score(y, y\_pred)  
print("Mean Squared Error:", mse)  
print("R^2 Score:", r2)  
  
# Task 4: Visualizing the Regression Line  
plt.scatter(X, y, color='blue')  
plt.plot(X, y\_pred, color='red')  
plt.title('Linear Regression Line')  
plt.xlabel('X')  
plt.ylabel('Y')  
plt.show()

# 2. Logistic Regression using Scikit-learn

from sklearn.linear\_model import LogisticRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
  
# Create a dataset  
X = np.array([[1], [2], [3], [4], [5], [6]])  
y = np.array([0, 0, 0, 1, 1, 1])  
  
# Train-test split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
  
# Train logistic regression  
log\_model = LogisticRegression()  
log\_model.fit(X\_train, y\_train)  
  
# Predictions and evaluation  
y\_pred = log\_model.predict(X\_test)  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("Accuracy:", accuracy)  
  
# Visualize  
plt.scatter(X, y, color='black')  
x\_vals = np.linspace(0, 7, 100)  
y\_vals = log\_model.predict\_proba(x\_vals.reshape(-1, 1))[:, 1]  
plt.plot(x\_vals, y\_vals, color='red')  
plt.title('Logistic Regression Curve')  
plt.xlabel('X')  
plt.ylabel('Probability')  
plt.show()

# 3. Using SciPy for Linear Algebra and Signal Processing

import scipy.linalg as la  
from scipy.fft import fft, ifft  
  
# Define a matrix for eigenvalues/eigenvectors  
A = np.array([[4, -2], [1, 1]])  
eigenvalues, eigenvectors = la.eig(A)  
print("Eigenvalues:", eigenvalues)  
print("Eigenvectors:\n", eigenvectors)  
  
# FFT and IFFT  
signal = np.array([1, 2, 3, 4])  
signal\_fft = fft(signal)  
signal\_ifft = ifft(signal\_fft)  
print("FFT:", signal\_fft)  
print("IFFT:", signal\_ifft)