***LAB ASSIGNMENT 2***

***NAME ZAIN ZAHOOR***

***REG NO FA21-BSE-166***

***ARTIFICIAL INTELLIGENCE***

Combine code off all three questions

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Data

xval = np.array([301.8267195898913, 206.48068590297993, 141.3506905750422, 181.45188429363267,

41.78127527790235, 36.74157682849997, 49.77129275587662, -21.54843159015938,

75.4361571381737, -3.8882674060444335, -9.46856287330992, -71.3194584463108,

42.1215633120422, 49.088012341286515, 86.97664315575287, 66.69982327495062,

122.96273379072505, 105.46219673639621, 139.43161572804308, 281.7900813768265,

229.67424006279856])

yval = np.array([-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Display the data

print("xval:", xval)

print("yval:", yval)

# Function to plot data

def plotData(xval, yval, title="Data Plot"):

plt.scatter(xval, yval, color='blue', label='Data Points')

plt.title(title)

plt.xlabel('x')

plt.ylabel('y')

plt.grid(True)

plt.legend()

plt.show()

# Function to fit a linear model

def fitLinearModel(xval, yval):

xval\_reshaped = xval.reshape(-1, 1) # Reshape xval for LinearRegression

model = LinearRegression()

model.fit(xval\_reshaped, yval)

y\_pred = model.predict(xval\_reshaped)

a = model.coef\_[0] # Slope

b = model.intercept\_ # Intercept

plt.plot(xval, y\_pred, color='red', label=f'Linear Fit: y = {a:.2f}x + {b:.2f}')

return a, b, y\_pred

# Function to fit polynomial models of different degrees

def fitPolynomialModel(xval, yval, degree):

coeffs = np.polyfit(xval, yval, degree) # Fit polynomial of given degree

poly = np.poly1d(coeffs)

y\_pred\_poly = poly(xval)

label = f'Poly Degree {degree}: ' + " + ".join([f"{coef:.2f}x^{deg}" for deg, coef in enumerate(coeffs[::-1])])

plt.plot(xval, y\_pred\_poly, label=label)

return coeffs, y\_pred\_poly

# Function to calculate Mean Squared Error (MSE)

def calculateMSE(yval, y\_pred):

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

return mse

# Plot the data

plotData(xval, yval)

# Fit linear model and plot it

a\_linear, b\_linear, y\_pred\_linear = fitLinearModel(xval, yval)

# Fit polynomial models of degree 2, 4, 8, and 16

degrees = [4, 8, 16]

for degree in degrees:

coeffs, y\_pred\_poly = fitPolynomialModel(xval, yval, degree)

mse\_poly = calculateMSE(yval, y\_pred\_poly)

print(f"Degree {degree} Polynomial MSE: {mse\_poly}")

# Calculate MSE for the linear model

mse\_linear = calculateMSE(yval, y\_pred\_linear)

print(f"Linear Model MSE: {mse\_linear}")

# Show the final plot with all models

plt.legend()

plt.show()

# Print coefficients for linear model

print(f"Linear Model coefficients: a = {a\_linear:.2f}, b = {b\_linear:.2f}")