**Mini Project 1**

Author:

|  |  |
| --- | --- |
| **Abhirup Ranjan** | **110091866** |

Submitted To:

|  |
| --- |
| **Professor** |
| **Dr. Adel Abusitta** |

**Table of Contents**

[1. Problem Statement 1 2](#_Toc135836254)

[1.1 Python Code 2](#_Toc135836255)

[1.2 Terminal Window Screenshot 3](#_Toc135836256)

[1.3 Output Window Screenshot 4](#_Toc135836257)

[2. Problem Statement 2 4](#_Toc135836258)

[2.1 Python Code 4](#_Toc135836259)

[2.2 Terminal Window Screenshot 6](#_Toc135836260)

[2.3 Output Window Screenshot 6](#_Toc135836261)

[3. Problem Statement 3 7](#_Toc135836262)

[3.1 Python Code 7](#_Toc135836263)

[3.2 Terminal Window Screenshot 8](#_Toc135836264)

[3.3 Output Window Screenshot 8](#_Toc135836265)

# **Problem Statement 1**

Write a program example for fitting a linear regression model using 20 data points with 2 features. The program should plot the model and make predictions for new points.

# **1.1 Python Code**

#PROBLEM: Write a program example for fitting a linear regression model using 20 data points with 2 features.

# The program should plot the model and make predictions for new points.

#NOTES: Took reference from the class material provided in lecture 2 to solve the problem.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Creating 20 sample data with two features

# y = m1x1+m2x2+....c (where y = dependent and x is independent

#2D array of 20 rows and 2 columns

X = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10],

[11,12], [13,14], [15,16],[17,18], [19,20],

[21,22], [23,24], [25,26], [27,28], [29,30],

[31,32], [33,34], [35,36], [37,38], [39,40]]) # Independent variable

y = np.array([12, 14, 16, 18, 20, 22, 24, 26, 28, 30,

32, 34, 36, 38, 40, 42, 44, 46, 48, 50]) # Dependent variable

# Creating a linear regression model for the function

regModel = LinearRegression()

# Fit the linear regression model to the data

regModel.fit(X, y) # independent

# Print the model coefficients

print("Coefficients:", regModel.coef\_)

# Predict the output for new data points marked below

X\_new = np.array([[41, 42], [43, 45], [45, 46]])

y\_pred = regModel.predict(X\_new)

print("Predictions for y:", y\_pred)

# Plotting the data with the linear regression in a 3D plan

fig = plt.figure()

axis = fig.add\_subplot(111, projection='3d') #projection='3d' specifies that we want to create a 3D subplot,

#meaning that we will plot our data in a three-dimensional space.

#The argument 111 in add\_subplot(111) is a shorthand notation for creating a single subplot in a 1x1 grid.

#In this case, it means that we want to create one subplot in the figure.

axis.scatter(X[:,0], X[:,1], y)

# Create a meshgrid to plot the regression plane

xx, yy = np.meshgrid(range(45), range(45))

#np.meshgrid() is a function from the NumPy library that creates a coordinate grid or meshgrid from given vectors.

zz = regModel.coef\_[0] \* xx + regModel.coef\_[1] \* yy + regModel.intercept\_

# Plot the regression plane

axis.plot\_surface(xx, yy, zz, alpha=0.5)

#alpha=0.5 is an optional parameter that sets the transparency or opacity of the surface plot.

# Here, alpha=0.5 means that the surface will be partially transparent,

# allowing some visibility of other elements in the plot.

# Add axis labels and a title to the plot

axis.set\_xlabel('Feature 1')

axis.set\_ylabel('Feature 2')

axis.set\_zlabel('Output')

plt.title('Multiple Linear Regression with Two Features in 3D')

plt.show()

# **1.2 Terminal Window Screenshot**

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 1.1**

# **1.3 Output Window Screenshot**

A picture containing diagram, line, screenshot, design

Description automatically generated

**Figure 1.2**

# **Problem Statement 2**

Write a program example for fitting polynomial regression model using 12 data points with degree = 5. The program should plot the model and make predictions for new points.

# **2.1 Python Code**

#PROBLEM: Write a program example for fitting polynomial regression model using 12 data points with degree = 5.

# The program should plot the model and make predictions for new points.

#NOTES: Took refrence from the class material provided in lecture 2 to solve the problem.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

# 12 sample data points

X = np.array([[5], [7], [9], [11], [13], [15],

[17], [19], [21], [23], [25], [27]]) # Independent variable

y = np.array([1, 2.5, 4.8, 9.1, 10.3, 11.9,

14.4, 17.6, 22.2, 24.5, 28.7, 30]) # Dependent variable

# Below code will Transform the features into degree 5 polynomial feature

polynomial\_transform = PolynomialFeatures(degree=5) # Setting the degree of the polynomial function

X\_poly = polynomial\_transform.fit\_transform(X)

# Creating linear regression object

regObj = LinearRegression()

# Fitting above linear regression using the polynomial features

regObj.fit(X\_poly, y)

# Generate points and plot polynomial curve

x\_plot = np.linspace(5, 31, 100).reshape(-1, 1)

x\_plot\_poly = polynomial\_transform.transform(x\_plot)

y\_plot = regObj.predict(x\_plot\_poly)

# Make the required predictions for the new data points provided below

X\_new = np.array([[29], [31]])

X\_new\_poly = polynomial\_transform.transform(X\_new)

y\_new = regObj.predict(X\_new\_poly)

# plot the predicted value for new data points

print("Predicting the values for new data points:")

for i in range(len(X\_new)):

print("X =", X\_new[i][0], " Predicted y =", y\_new[i])

# Plot all the data points, polynomial curve, and the predicted values

plt.scatter(X, y, color='blue', label='Actual Data')

plt.plot(x\_plot, y\_plot, color='red', label='Polynomial Regression Curve Line')

plt.scatter(X\_new, y\_new, color='green', label='Predicted Points')

plt.xlabel('X (Independent Variable)')

plt.ylabel('y Dependent Variable)')

plt.title('Polynomial Regression for Degree=5')

plt.legend()

plt.show()

# **2.2 Terminal Window Screenshot**

A screen shot of a computer

Description automatically generated with medium confidence

**Figure 2.1**

# **2.3 Output Window Screenshot**

A picture containing text, line, plot, screenshot

Description automatically generated

**Figure 2.2**

# **Problem Statement 3**

Write a program example to fit a k-nearest neighbors (KNN) model with k = 2 using 15 data points. The program should include prediction capabilities and plot the results.

# **3.1 Python Code**

# PROBLEM: Write a program example to fit a k-nearest neighbors (KNN) model with k = 2 using 15 data points.

# The program should include prediction capabilities and plot the results.

#NOTES: Took reference from the class material provided in lecture 2 to solve the problem.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsRegressor

# Generate 15 sample data points

np.random.seed(123)

X = np.sort(5 \* np.random.rand(15, 1), axis=0)

y = np.sin(X).ravel()

y[::5] += 3 \* (0.5 - np.random.rand(3))

# Splitting the data into training and testing sets for both X and y

X\_train, X\_test = X[:5], X[5:]

y\_train, y\_test = y[:5], y[5:]

# Fitting KNN regression model

n\_neighbors = 2 # choosing the number of neighbors here

knn\_model = KNeighborsRegressor(n\_neighbors=n\_neighbors)

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

# Doing predictions by KNN model

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

# Evaluate mean squared error of predictions

mse = np.mean((y\_pred - y\_test) \*\* 2)

print(f"Mean squared error is: {mse:.2f}")

# Plotting result

plt.scatter(X\_test, y\_test, color='red', label='True')

plt.scatter(X\_test, y\_pred, color='blue', label='Predicted')

plt.legend()

plt.title(f"KNN Regression Model for (k={n\_neighbors}, MSE={mse:.2f})")

plt.xlabel('X axis (Independent Variable)')

plt.ylabel('y axis (Dependent Variable)')

plt.show()

# **3.2 Terminal Window Screenshot**

A screen shot of a computer

Description automatically generated with medium confidence

**Figure 3.1**

# **3.3 Output Window Screenshot**

A picture containing text, line, plot, diagram

Description automatically generated

**Figure 3.2**