|  |  |
| --- | --- |
| A blue logo with a black background  Description automatically generated | **AIR UNIVERSITY** |
| **DEPARTMENT OF COMPUTER SCIENCE** |
| **Lab Task 7** |

**Student Name: Hamza Umer Farooq Reg. No: 200789**

**Subject: Data Science Semester: VIII**

**Objective: Linear Regression**

**ASSESSMENT:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attributes** | **Excellent**  **(5)** | **Good**  **(4)** | **Average**  **(3)** | **Satisfactory**  **(2)** | **Unsatisfactory (1)** |
| **Ability to Conduct**  Task |  |  |  |  |  |
| **Ability to assimilate the results** |  |  |  |  |  |
| **Effective use of theorems/postulates/formulas** |  |  |  |  |  |

Total Marks:

Obtained Marks:

**REPORT ASSESSMENT:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attributes** | **Excellent**  **(5)** | **Good**  **(4)** | **Average**  **(3)** | **Satisfactory**  **(2)** | **Unsatisfactory**  **(1)** |
| **Data presentation** |  |  |  |  |  |
| **Experimental results** |  |  |  |  |  |
| **Conclusion** |  |  |  |  |  |

# Lab Task

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

#1 Load the data and show the info and contents:

import pandas as pd

ice = pd.read\_csv('SeaIce.txt', delim\_whitespace = True)

print('Shape:', ice.shape)

ice.head()

#2 Select the numeric features (feature selection).

numeric\_features = ice.select\_dtypes(include=[np.number])

numeric\_features.head()

#3 Find the Basic Statistics (Mean, Variance, S.D)

# Calculate mean

mean = numeric\_features.mean()

# Calculate variance

variance = numeric\_features.var()

# Calculate standard deviation

std\_dev = numeric\_features.std()

print("Mean:\n", mean)

print("\nVariance:\n", variance)

print("\nStandard Deviation:\n", std\_dev)

# Calculate IQR for each numeric feature

Q1 = numeric\_features.quantile(0.25)

Q3 = numeric\_features.quantile(0.75)

IQR = Q3 - Q1

# Define outliers as values below Q1 - 1.5\*IQR or above Q3 + 1.5\*IQR

outliers = (numeric\_features < (Q1 - 1.5 \* IQR)) | (numeric\_features > (Q3 + 1.5 \* IQR))

# Print outliers

print(outliers)

plt.figure(figsize=(15,10))

plt.xticks(rotation='vertical')

plt.boxplot(numeric\_features, vert=False, patch\_artist=True)

plt.title('Box plot for each numeric feature')

plt.show()

# Calculate Q1, Q3, and IQR

Q1 = numeric\_features.quantile(0.25)

Q3 = numeric\_features.quantile(0.75)

IQR = Q3 - Q1

# Define outliers as values below Q1 - 1.5\*IQR or above Q3 + 1.5\*IQR

outliers = (numeric\_features < (Q1 - 1.5 \* IQR)) | (numeric\_features > (Q3 + 1.5 \* IQR))

# Print outliers

print(outliers)

# Remove outliers

numeric\_features = numeric\_features[~outliers.any(axis=1)]

plt.figure(figsize=(15,10))

plt.xticks(rotation='vertical')

plt.boxplot(numeric\_features, vert=False, patch\_artist=True)

plt.title('Box plot for each numeric feature')

plt.show()

numeric\_features.hist(bins=50, figsize=(20,15))

plt.show()

# Function to calculate m and b

def linear\_regression(X, y):

    x\_mean = np.mean(X)

    y\_mean = np.mean(y)

    numerator = np.sum((X - x\_mean) \* (y - y\_mean))

    denominator = np.sum((X - x\_mean) \*\* 2)

    b = numerator / denominator

    a = y\_mean - (b \* x\_mean)

    return a, b

#8 Perform Linear Regression.

# Define the independent variable (X) and the dependent variable (Y)

X = numeric\_features['year'].values

Y = numeric\_features['extent'].values

# Perform linear regression

a, b = linear\_regression(X, Y)

# Print the slope and intercept

print("Slope (b):", b)

print("Intercept (a):", a)

# Function to calculate prediction

def predict(X, a, b):

    return b \* X + a

# Function to calculate RMSE

def rmse(y\_true, y\_pred):

    return np.sqrt(np.mean((y\_true - y\_pred) \*\* 2))

#9 Plot the actual dependent variable and predicted dependent variable.

# Making predictions

predictions = predict(X, a, b)

# Plotting the actual and predicted values

plt.scatter(X, Y, color='blue', label='Data Points')

plt.plot(X, predictions, color='red', label='Regression Line')

plt.xlabel('Year')

plt.ylabel('Extent')

plt.title('Actual vs Predicted Values')

plt.legend()

plt.show()

predictions = predict(X, a, b)

error = rmse(Y, predictions)

print("Predictions:", predictions)

print("RMSE:", error)

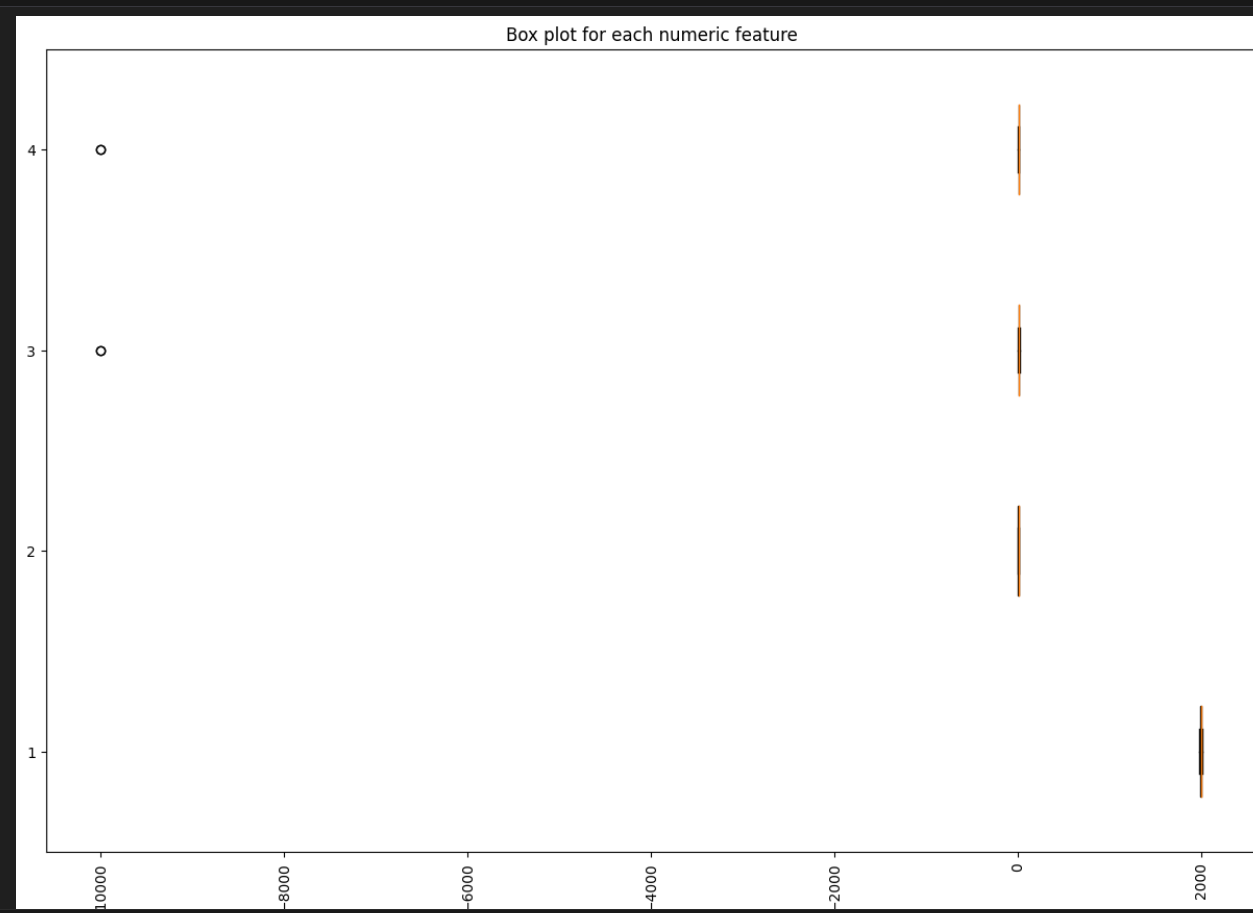
#10 Estimate the extent value for the year 2025

year\_2025 = np.array([2025])

predicted\_extent\_2025 = predict(year\_2025, a, b)

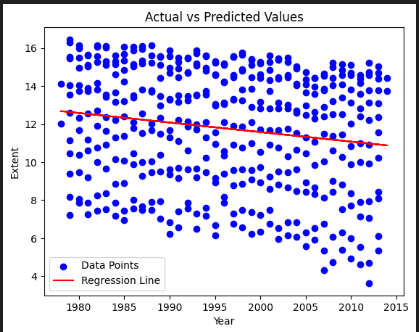
print("Predicted extent for the year 2025:", predicted\_extent\_2025[0])

Outliers before:



Outliers After:  
A white rectangular object with numbers

Description automatically generatedA graph of different sizes and colors

Description automatically generated with medium confidenceA screen shot of a computer code

Description automatically generated