PART A

(PART A: TO BE REFFERED BY STUDENTS)

Experiment No. 2

A.1 Aim:

To implement Linear Regression.

A.2 Prerequisite:

Python Basic Concepts

A.3 Outcome:

Students will be able to implement Linear Regression.

A.4 Theory:

Machine Learning, being a subset of Artificial Intelligence (AI), has been playing a dominant role in our daily lives. Data science engineers and developers working in various domains are widely using machine learning algorithms to make their tasks simpler and life easier.

What is a Regression Problem?

Majority of the machine learning algorithms fall under the supervised learning category. It is the process where an algorithm is used to predict a result based on the previously entered values and the results generated from them. Suppose we have an input variable 'x' and an output variable 'y' where y is a function of x (y=f{x}). Supervised learning reads the value of entered variable 'x' and the resulting variable 'y' so that it can use those results to later predict a highly accurate output data of 'y' from the entered value of 'x'. A regression problem is when the resulting variable contains a real or a continuous value. It tries to draw the line of best fit from the data gathered from a number of points.

Linear Regression

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear

relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression. If there is a single input variable (x), such linear regression is called simple linear regression. And if there is more than one input variable, such linear regression is called multiple linear regression. The linear regression model gives a sloped straight line describing the relationship within the variables.

To calculate best-fit line linear regression uses a traditional slope-intercept form.

$$y = mx+b \quad \Longrightarrow \quad y = a_0+a_1x$$

y= Dependent Variable.
x= Independent Variable.
a0= intercept of the line.
a1 = Linear regression coefficient.

Need of a Linear regression

As mentioned above, Linear regression estimates the relationship between a dependent variable and an independent variable. Let's understand this with an easy example:

Let's say we want to estimate the salary of an employee based on year of experience. You have the recent company data, which indicates that the relationship between experience and salary. Here year of experience is an independent variable, and the salary of an employee is a dependent variable, as the salary of an employee is dependent on the experience of an employee. Using this insight, we can predict the future salary of the employee based on current & past information.

A regression line can be a Positive Linear Relationship or a Negative Linear Relationship.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

Roll. No. BE-A15	Name: Khan Mohammad TAQI
Class: BE-Comps	Batch: A1
Date of Experiment:	Date of Submission:
Grade:	

B.1 Software Code written by student:

 # To implement Linear Regression.

%matplotlib inline *#for graph open it in google colab*

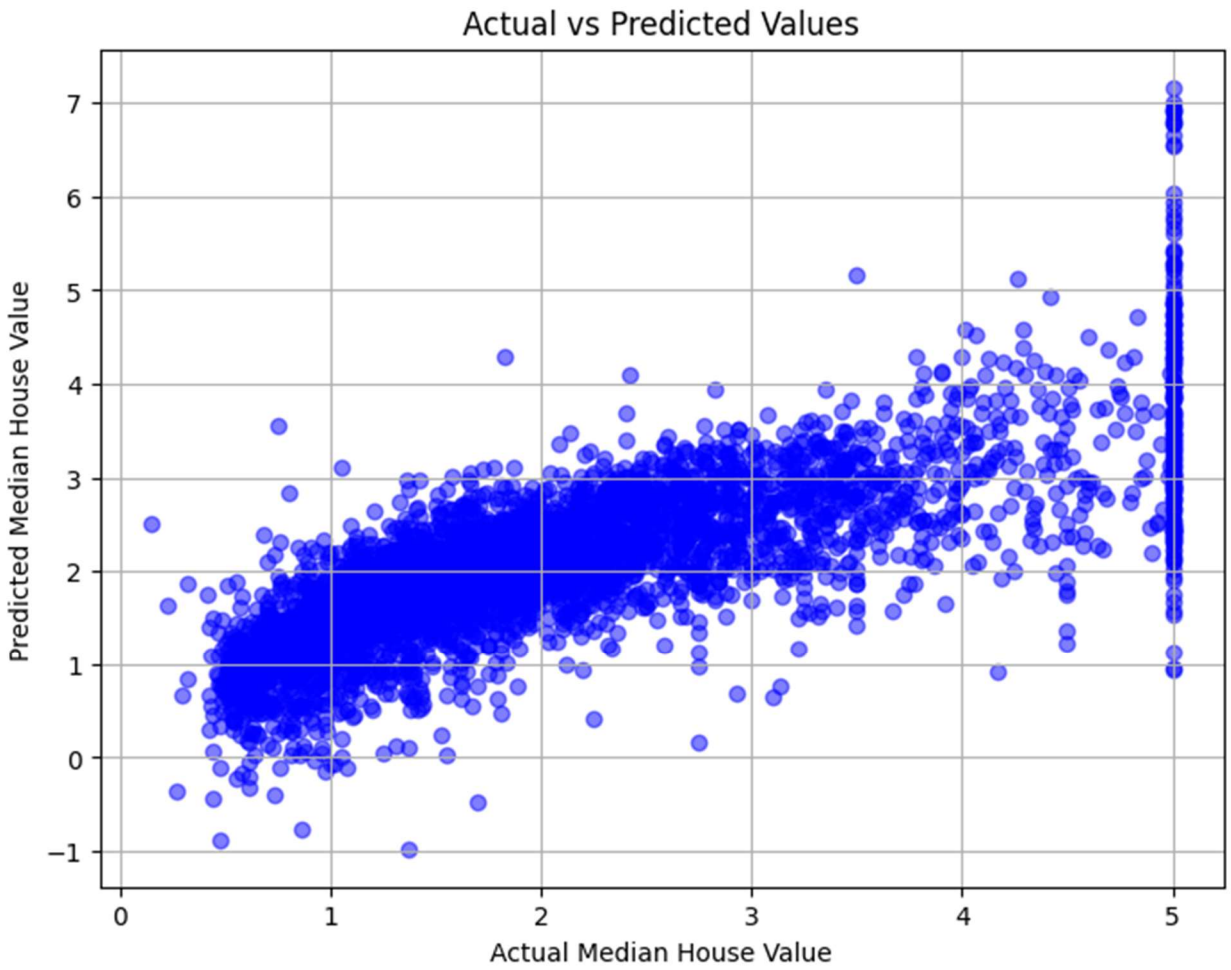
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
df['MedianHouseValue'] = housing.target

print(df.info())
print("Missing values in dataset:\n", df.isnull().sum())

X = df.drop('MedianHouseValue', axis=1)
y = df['MedianHouseValue']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LinearRegression()
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)

print(f"Intercept (a0): {model.intercept_}")
print(f"First Coefficient (a1): {model.coef_[0]}")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred):.2f}")
print(f"R² Score: {r2_score(y_test, y_pred):.2f}")

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred, alpha=0.5, color='blue')
plt.xlabel("Actual Median House Value")
plt.ylabel("Predicted Median House Value")
plt.title("Actual vs Predicted Values")
plt.grid(True)
plt.tight_layout()
plt.show()
```

B.2 Input and Output:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedianHouseValue
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

```
taqi@Taqi:~/experiments/mlmain$ python3 LinearRegression.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   MedInc             20640 non-null  float64
 1   HouseAge           20640 non-null  float64
 2   AveRooms           20640 non-null  float64
 3   AveBedrms          20640 non-null  float64
 4   Population         20640 non-null  float64
 5   AveOccup           20640 non-null  float64
 6   Latitude           20640 non-null  float64
 7   Longitude          20640 non-null  float64
 8   MedianHouseValue   20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
None
Missing values in dataset:
 MedInc               0
HouseAge              0
AveRooms              0
AveBedrms             0
Population            0
AveOccup              0
Latitude              0
Longitude             0
MedianHouseValue      0
dtype: int64
Intercept (a0): 2.0724989589389438
First Coefficient (a1): 0.8262479345502438
Mean Squared Error: 0.53
R² Score: 0.59
```



B.3 Observations and learning:

In this experiment, I successfully implemented the Linear Regression algorithm, a fundamental supervised learning technique used for predictive analysis. I observed that this model works by establishing a linear relationship between a dependent (target) variable and one or more independent (predictor) variables. The core task was to find the "line of best fit" that most accurately represents the data points, which is mathematically described by the slope-intercept formula, y = a0 + a1*x. I noted the distinction between simple linear regression, which involves a single independent variable, and multiple linear regression, which uses several. The example of predicting an employee's salary based on years of experience clearly illustrated how this algorithm can be used to forecast continuous values in real-world scenarios.

B.4 Conclusion:

In conclusion, this experiment fulfilled its aim of implementing a Linear Regression model. Through this process, Ihave gained a practical understanding of how to predict continuous outcomes by modeling the relationships between variables. The experiment reinforces that Linear Regression is a straightforward and powerful statistical tool that serves as a cornerstone of machine learning. Its simplicity and interpretability make it an essential algorithm for data scientists to master for tasks involving forecasting and understanding data relationships.