Name: Ujwal Sahu

Branch: BVoc (AIDS)

Division: B

EXPERIMENT NO 6

Title: To understand and implement Simple Linear Regression & Multiple Linear Regression model for predicting values based on given Datasets.

Tools: Anaconda (Jupyter Notebook).

Theory:

Linear regression is one of the most fundamental and widely used predictive modeling techniques in machine learning. It is used to establish a relationship between an independent variable (predictor) and a dependent variable (outcome). Linear regression can be classified into two types:

Simple Linear Regression (SLR) – Involves a single independent variable.

Multiple Linear Regression (MLR) – Involves two or more independent variables.

Simple Linear Regression is a method that models the relationship between a dependent variable Y and a single independent variable X using a straight-line equation: $Y = mX + c$

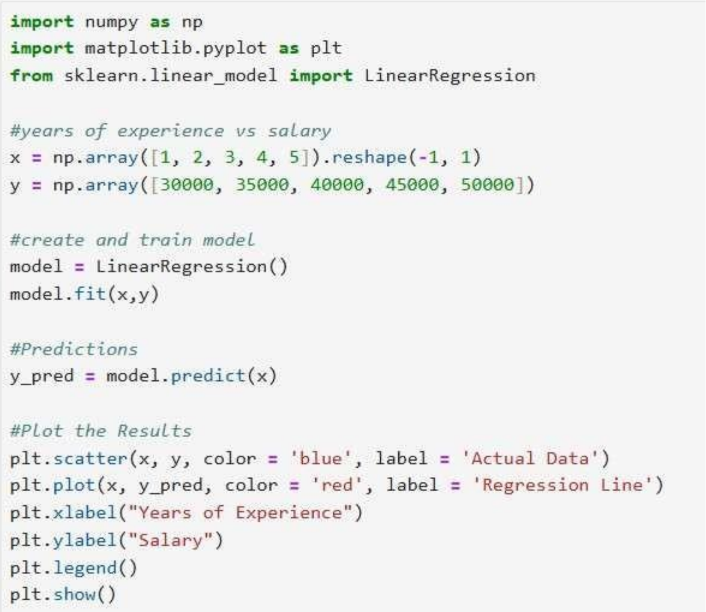
Code:

1] **Simple Linear Regression.**

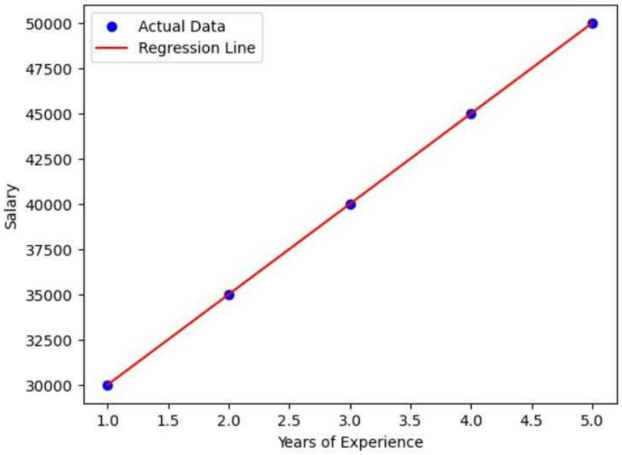
```
import numpy as np import

matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression


x = np.array([1, 2, 3, 4, 5]).reshape(-1, 1) y =

np.array([30000, 35000, 40000, 45000, 50000])


model = LinearRegression()

model.fit(x,y)
```

```
y_pred = model.predict(x)


plt.scatter(x, y, color = 'blue', label = 'Actual Data')
plt.plot(x, y_pred, color = 'red', label = 'Regression Line')
plt.xlabel("Years of Experience") plt.ylabel("Salary")
plt.legend() plt.show()
```

Output:



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

#years of experience vs salary
x = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([30000, 35000, 40000, 45000, 50000])

#create and train model
model = LinearRegression()
model.fit(x,y)

#Predictions
y_pred = model.predict(x)

#Plot the Results
plt.scatter(x, y, color = 'blue', label = 'Actual Data')
plt.plot(x, y_pred, color = 'red', label = 'Regression Line')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.legend()
plt.show()
```

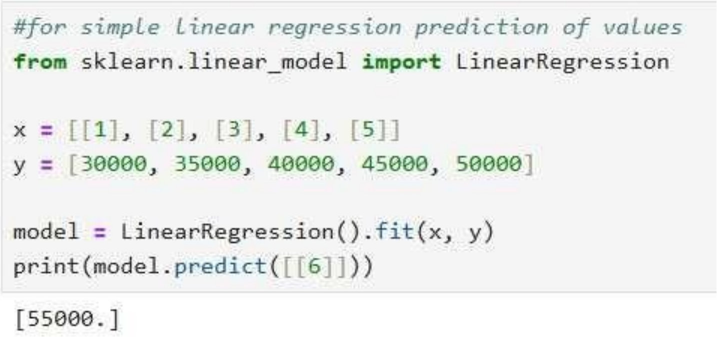


2] **Simple Linear Regression for predicting values.**

```
from sklearn.linear_model import LinearRegression
```

```
x = [[1], [2], [3], [4], [5]] y = [30000,
35000, 40000, 45000, 50000]

model = LinearRegression().fit(x, y)
print(model.predict([[6]]))
```

Output:



```
#for simple linear regression prediction of values
from sklearn.linear_model import LinearRegression

x = [[1], [2], [3], [4], [5]]
y = [30000, 35000, 40000, 45000, 50000]

model = LinearRegression().fit(x, y)
print(model.predict([[6]]))
```

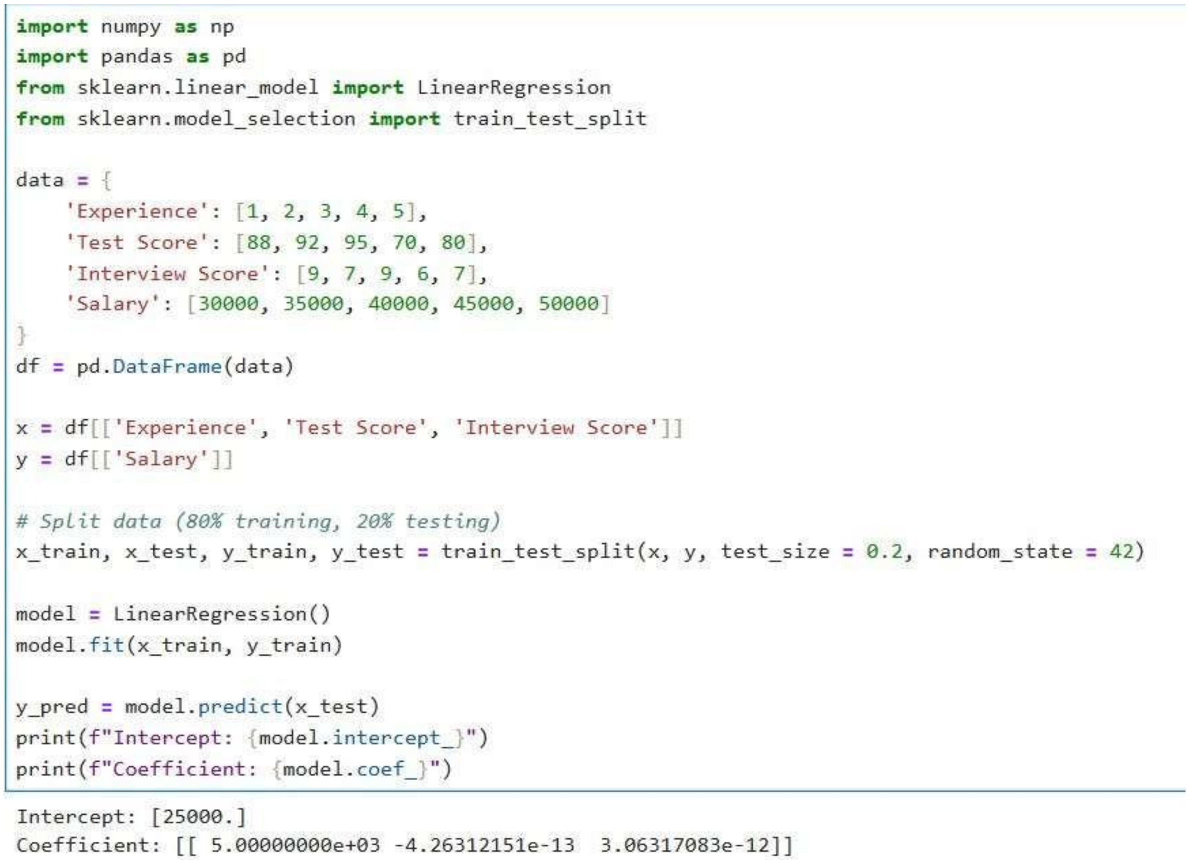
```
[55000.]
```

3] **Multiple Linear Regression.**

```
import numpy as np import
pandas as pd
from sklearn.linear_model import LinearRegression from
sklearn.model_selection import train_test_split


data = {
    'Experience': [1, 2, 3, 4, 5],
    'Test Score': [88, 92, 95, 70, 80],
    'Interview Score': [9, 7, 9, 6, 7],
    'Salary': [30000, 35000, 40000, 45000, 50000]
}
df = pd.DataFrame(data)

x = df[['Experience', 'Test Score', 'Interview Score']] y
= df[['Salary']]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
```

```
model = LinearRegression() model.fit(x_train,
y_train)

y_pred    =    model.predict(x_test)
print(f"Intercept:  {model.intercept_}")
print(f"Coefficient: {model.coef_}")
```

Output:



```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

data = {
    'Experience': [1, 2, 3, 4, 5],
    'Test Score': [88, 92, 95, 70, 80],
    'Interview Score': [9, 7, 9, 6, 7],
    'Salary': [30000, 35000, 40000, 45000, 50000]
}
df = pd.DataFrame(data)

x = df[['Experience', 'Test Score', 'Interview Score']]
y = df[['Salary']]

# Split data (80% training, 20% testing)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)

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

y_pred = model.predict(x_test)
print(f"Intercept: {model.intercept_}")
print(f"Coefficient: {model.coef_}")
```

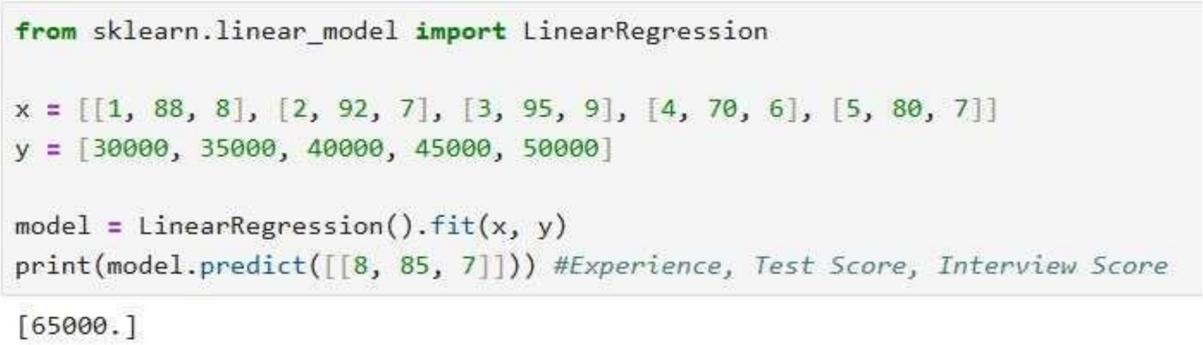
```
Intercept: [25000.]
Coefficient: [[ 5.00000000e+03 -4.26312151e-13  3.06317083e-12]]
```

4] **Multiple Linear Regression for predicting values.**

```
from sklearn.linear_model import LinearRegression
```

```
x = [[1, 88, 8], [2, 92, 7], [3, 95, 9], [4, 70, 6], [5, 80, 7]]
y = [30000, 35000, 40000, 45000, 50000]

model = LinearRegression().fit(x, y)
print(model.predict([[8, 85, 7]])) #Experience, Test Score, Interview Score
```

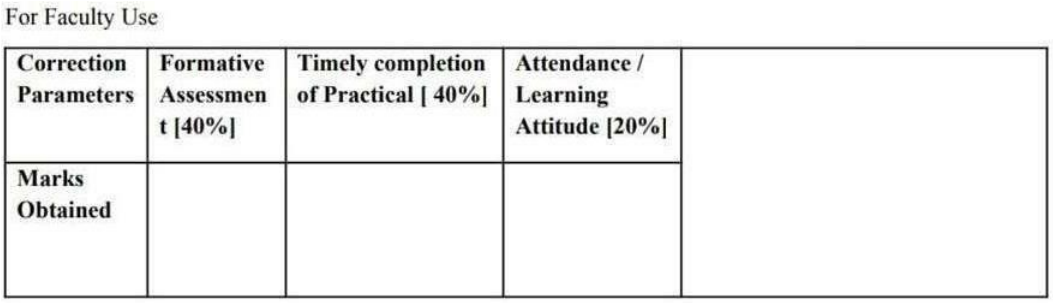
OUTPUT:



```
from sklearn.linear_model import LinearRegression

x = [[1, 88, 8], [2, 92, 7], [3, 95, 9], [4, 70, 6], [5, 80, 7]]
y = [30000, 35000, 40000, 45000, 50000]

model = LinearRegression().fit(x, y)
print(model.predict([[8, 85, 7]])) #Experience, Test Score, Interview Score
```

```
[65000.]
```

Conclusion: Simple Linear Regression is effective for modeling a relationship between two variables but fails when there are multiple influencing factors. Multiple Linear Regression provides a more robust model when multiple independent variables impact the dependent variable. Model Evaluation Metrics such as R-squared and RMSE help assess the accuracy of the predictions. Assumptions of linear regression (e.g., linearity, normality, no multicollinearity) should be verified before applying the model.



For Faculty Use

Correction Parameters	**Formative Assessmen t [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				