PROGRAM : 3

AIM:

Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

Linear Regression

DATASET: Salary_Data.csv

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Importing the dataset
dataset = pd.read_csv('/content/Salary_Data.csv')
```

```
dataset.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

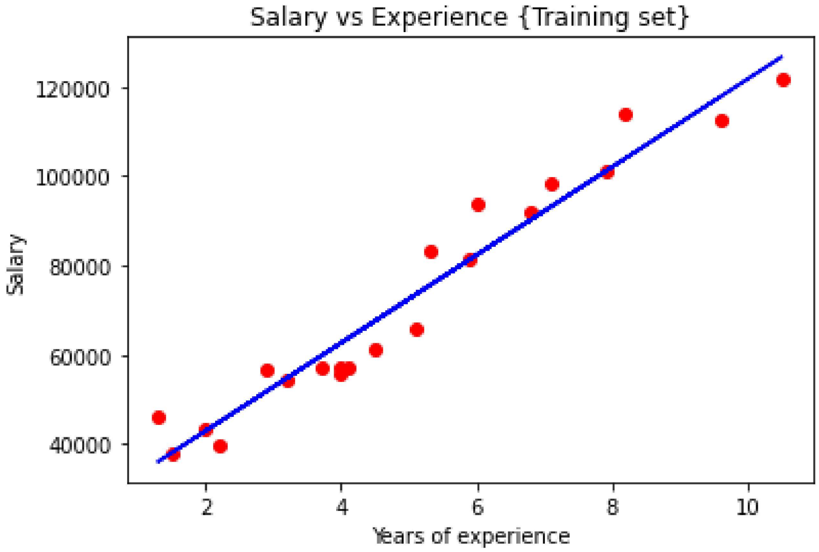
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_sta
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
#LinearRegression() fits a linear model to minimize the residual sum of squares betwe
regressor.fit(X_train, y_train)
```

```
     LinearRegression()
```

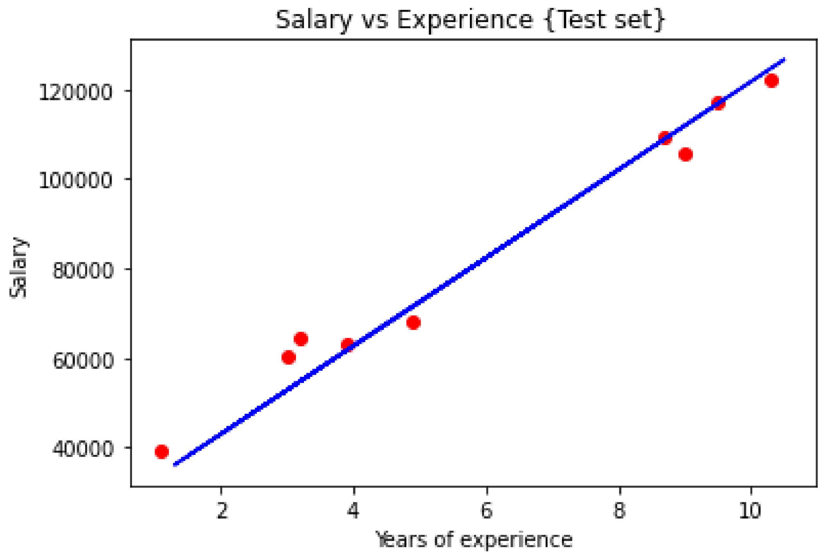
```
#Predicting the Test set results
y_pred = regressor.predict(X_test)
```

```
#By comparing real salaries in y_test to predicted salary y_pred we can find the corr
#Visualizing Training set results
#Plot employees of the company categorized by their number of years of experience by
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience {Training set}')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
```



```
#Next we will plot the test set observation points by keeping the training set regres
#Visualizing Test set results
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')

plt.title('Salary vs Experience {Test set}')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
plt.show()
```



Multiple Regression

DATASET: combined cycle powerplant.csv'

```
#Import dataset
df=pd.read_csv('/content/combined cycle powerplant.csv')
```

```
df.head()
```

	AT	V	AP	RH	PE
0	8.34	40.77	1010.84	90.01	480.48
1	23.64	58.49	1011.40	74.20	445.75
2	29.74	56.90	1007.15	41.91	438.76
3	19.07	49.69	1007.22	76.79	453.09
4	11.80	40.66	1017.13	97.20	464.43

```
#Define x and y(independent and dependent variable)
x=df.drop('PE',axis=1).values #drop all columns except depenedent variable PE
y=df['PE'].values
```

```
#split the dataset in training and testing set
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.3, random_state=45)
```

```
#train the model on the training set
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)#fit the model to linear regression
```

```
     LinearRegression()
```

```
#predict the test set result and see if we are getting the accurate results
y_pred = lr.predict(x_test)
```

```
print(y_pred)
```

```
     [451.12275809 472.67973273 434.23317529 ... 479.20120415 470.80190333
      437.26990979]
```

```
#take the values of first row in x and compare our predicted y values with actual y va
lr.predict([[8.34,40.77,1010.84,90.01]])
```

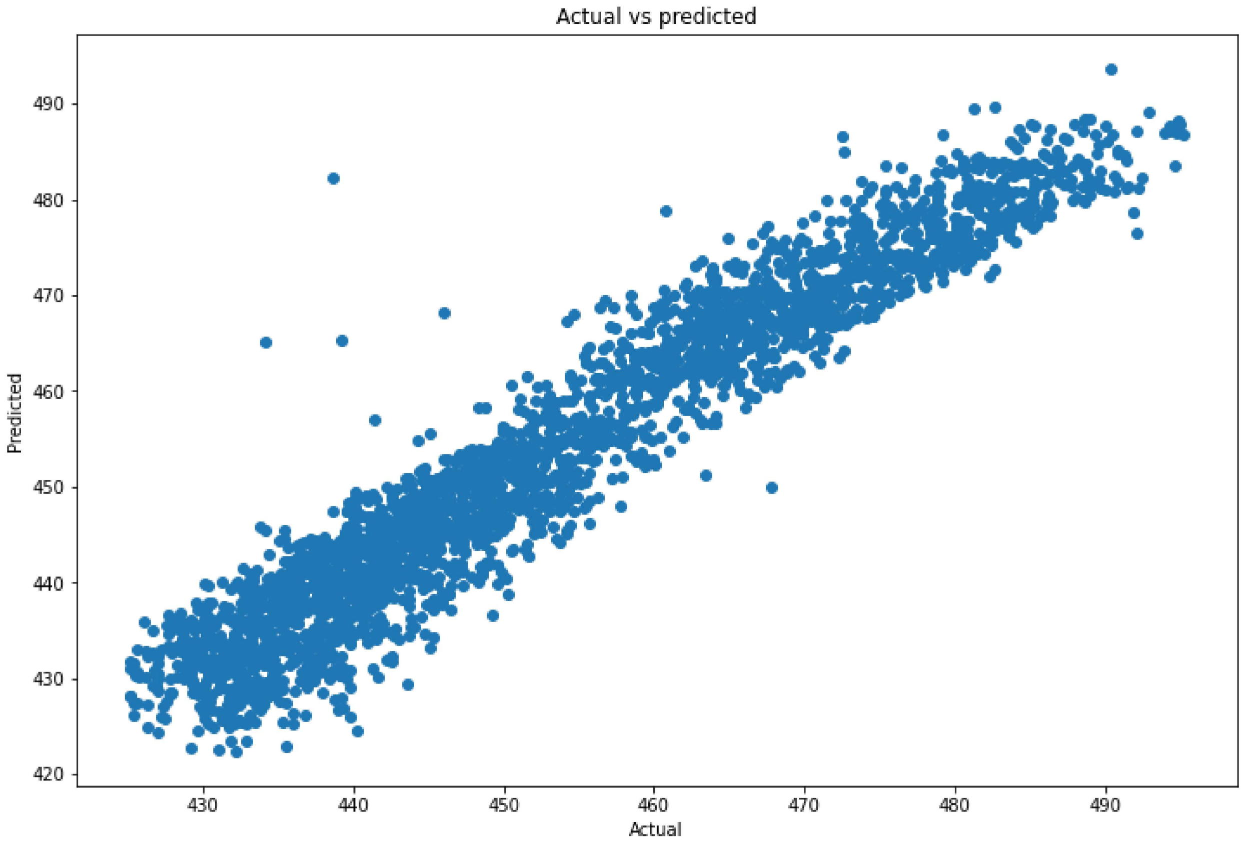
```
     array([477.22694032])
```

```
#inorder to further evaluate the accuracy of our predicted value, evaluate using r2sc
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

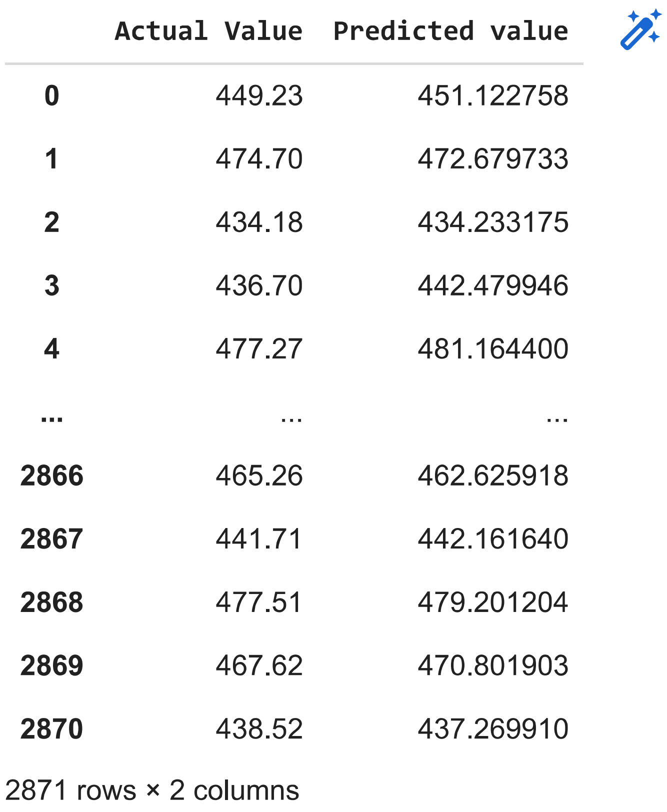
```
     0.9270483924843018
```

```
#then we visualize the predicted results
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
plt.scatter(y_test,y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs predicted')
#here we see that they are pretty close enough
```

```
Text(0.5, 1.0, 'Actual vs predicted')
```



```
#we print the predicted values of our model
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted value':y_pred})
pred_df
```



```
      Actual Value  Predicted value
0           449.23       451.122758
1           474.70       472.679733
2           434.18       434.233175
3           436.70       442.479946
4           477.27       481.164400
...            ...              ...
2866        465.26       462.625918
2867        441.71       442.161640
2868        477.51       479.201204
2869        467.62       470.801903
2870        438.52       437.269910

2871 rows × 2 columns
```

RESULT:

Program is executed successfully and output is obtained.