Implementation-of-Simple-Linear-Regression-Model-for-Predicting-the-Marks-Scored

AIM:

To write a program to predict the marks scored by a student using the simple linear regression model.

EQUIPMENT REQUIRED:

Hardware – PCs Anaconda – Python 3.7 Installation / Jupyter notebook

ALGORITHM:

Gather data consisting of two variables. Input- a factor that affects the marks and Output - the marks scored by students Plot the data points on a graph where x-axis represents the input variable and y-axis represents the marks scored Define and initialize the parameters for regression model: slope controls the steepness and intercept represents where the line crsses the y-axis Use the linear equation to predict marks based on the input Predicted Marks = m.(hours studied) + b for each data point calculate the difference between the actual and predicted marks Adjust the values of m and b to reduce the overall error. The gradient descent algorithm helps update these parameters based on the calculated error Once the model parameters are optimized, use the final equation to predict marks for any new input data

PROGRAM:

Program to implement the simple linear regression model for predicting the marks scored.

```
Developed by: SHAHIN J
RegisterNumber: 212223040190
```

```
import numpy as np
import pandas as pd
from sklearn.metrics import mean_absolute_error,mean_squared_error
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

#read csv file
df=pd.read_csv('/content/student_scores (1).csv')

#displaying the content in datafile
print(df.head())
print(df.tail())

# Segregating data to variables
x=df.iloc[:,:-1].values
print(x)
y=df.iloc[:,-1].values
print(y)

print(x.shape)
```

```
print(y.shape)

#splitting train and test data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0)

#import linear regression model and fit the model with the data
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)

#displaying predicted values
y_pred=reg.predict(x_test)
x_pred=reg.predict(x_train)
print(y_pred)
print(x_pred)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

#find mae,mse,rmse
mse = mean_squared_error(y_test,y_pred)
print('MSE = ',mse)
mae = mean_absolute_error(y_test,y_pred)
print('MAE = ',mae)
rmse = np.sqrt(mse)
print('RMSE = ',rmse)

#graph plot
plt.scatter(x_test,y_test,color="blue")
plt.plot(x_test,y_pred,color="silver")
plt.title('Test set(H vs S)')
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.show()

y_pred1=reg.predict(np.array([[13]]))
y_pred1
```

OUTPUT

Head Values

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

Tail Values

```
    Hours  Scores
20    2.7      30
21    4.8      54
22    3.8      35
23    6.9      76
24    7.8      86
```

X Values

```
[[2.5]
 [5.1]
 [3.2]
 [8.5]
 [3.5]
 [1.5]
 [9.2]
 [5.5]
 [8.3]
 [2.7]
 [7.7]
 [5.9]
 [4.5]
 [3.3]
 [1.1]
 [8.9]
 [2.5]
 [1.9]
 [6.1]
 [7.4]
 [2.7]
 [4.8]
 [3.8]
 [6.9]
 [7.8]]
```

Y Values

```
[21 47 27 75 30 20 88 60 81 25 85 62 41 42 17 95 30 24 67 69 30 54 35 76
 86]
```

Shape Values

```
(25, 1)
(25,)
```

Predicted Values

```
[17.04289179 33.51695377 74.21757747 26.73351648 59.68164043 39.33132858
 20.91914167 78.09382734 69.37226512]
[13.16664192 51.92914068 77.12476487 34.48601624 82.93913969 91.66070191
 61.61976537 36.42414117 28.67164142 55.80539056 28.67164142 84.87726463
 26.73351648 49.02195327 88.7535145  46.11476586]
```

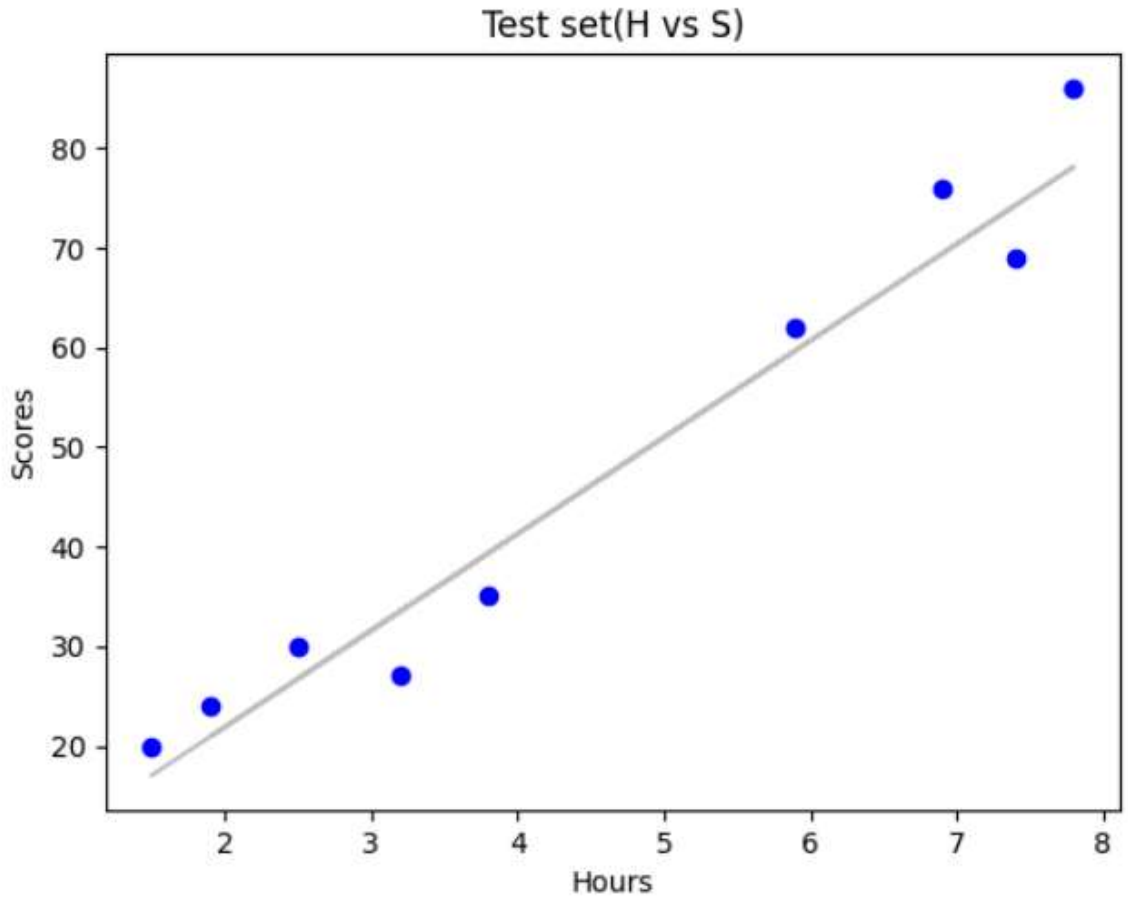
Training and Testing Shapes

```
(16, 1)
(9, 1)
(16,)
(9,)
```

MAE,MSE,RMSE

```
MSE =  25.463280738222593
MAE =  4.691397441397446
RMSE =  5.046115410711748
```

Graph Plot



Array Values

```
array([128.48507574])
```

Result

Thus the program to implement the simple linear regression model for predicting the