ex 6 q

March 10, 2024

[ ]:
```
EX NO 6 - Performance Analysis on Simple Linear Regression
Date:19-02-2024
HARIHARAN K - URK22AI1048
```

[ ]:
```
Aim
To implement the linear regression model for the given dataset and to␣
 ↪demonstrate the
performance analysis on regression techniques
Description
Linear Regression
Regression searches for relationships among variables. Regression is used to␣
 ↪build a prediction
model to predict the response (y) from the input variables (x) where the␣
 ↪prediction is based on the
previous data.
Linear regression model defines a linear relationship between the output␣
 ↪variable (y) and a
combination of one or more input variables (x)

Simple linear regression
This model has single independent and single dependent variable.
Eg: the experience impact salaries

B 0  = the y-intercept
B 1 = the regression coefficient (slope)
Calculation of B 0 and B 1 :
Formula for B 1

Formula for B 0

Performance Metrics for Regression Problems

Various performance metrics that can be used to evaluate predictions for␣
 ↪regression
problems are mean absolute error, mean squared error and R squared value.
Mean Absolute Error (MAE)
```

```
It is the simplest error metric used in regression problems. It is basically
    the sum of
average of the absolute difference between the predicted and actual values.

 1 |Y- |

=Actual Output Values
 = Predicted Output Values.

mean_absolute_error function of sklearn.metrics is used to compute MAE.
Mean Square Error (MSE)
MSE is like the MAE, but the only difference is that the it squares the
    difference of
actual and predicted output values before summing them all instead of using the
    absolute
value.

 1 (Y- ) 2
```

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

[19]:
```
# Load your dataset, replace 'your_dataset.csv' with your actual dataset
df = pd.read_csv("heatr.csv")
```

[21]:
```
df.dropna(inplace=True)
```

[22]:
```
X1=np.array(df[['biking']])
Y1=df[['smoking']]
```

[23]:
```
m1=LinearRegression(fit_intercept=True)
```

[24]:
```
#a. Calculate the intercept and regression coefficients in y=b0+xb1
m1.fit(X1,Y1)
print("intercept",m1.intercept_)
print("coeffcient",m1.coef_)
```

```
intercept [15.21430386]
coeffcient [[0.00584122]]
```

[25]:
```
#b. Analyse the various performance metrics (Mean Squared Error, Mean Absolute
    Error, Root Mean Squared Error, and R-Squared)

from sklearn import metrics
```

```
MsE=metrics.mean_squared_error(ypred1,Y1)
print("MsE---->",MsE)#Mean Squared Error

print("MaE---->",metrics.mean_absolute_error(ypred1,Y1))#Mean Absolute Error

RMsE=np.sqrt(MsE)
print("RMsE--->",RMsE)#Root Mean Squared Error

print("R^2---->",metrics.r2_score(ypred1,Y1))#R-Squared
```

```
MsE----> 68.56971395661313
MaE----> 7.027674094150692
RMsE---> 8.280683181755785
R^2----> -4362.828376568248
```

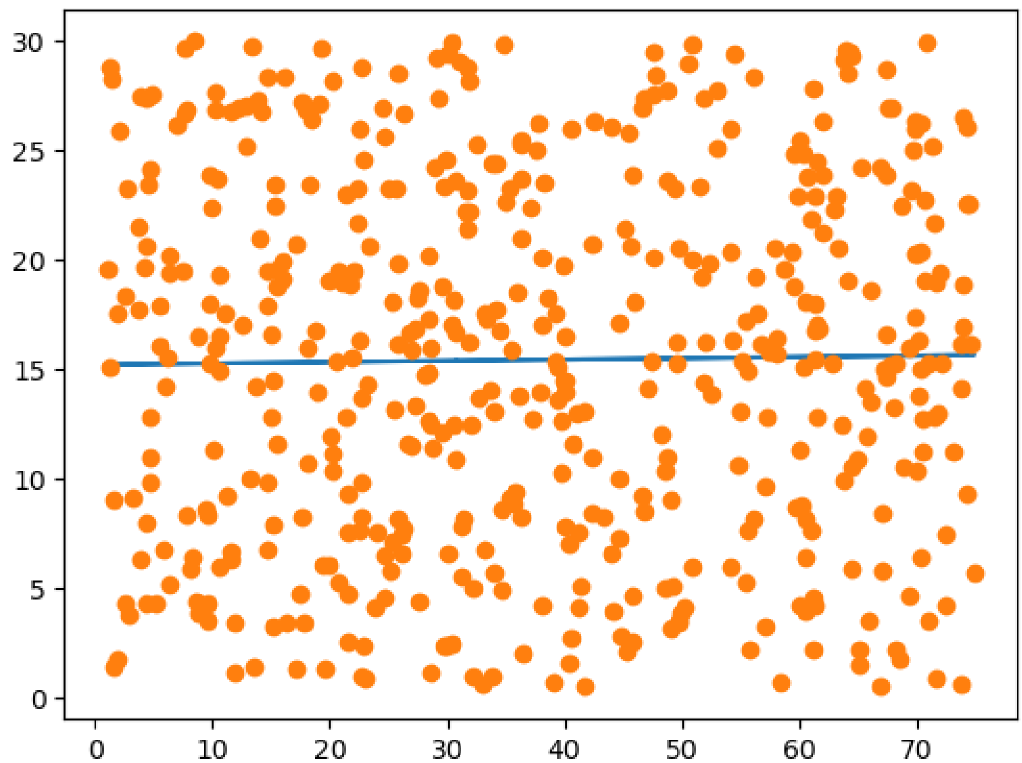
[15]:

```
ypred1=m1.predict(X1)
error1=(Y1-ypred1)**2
print("Error",error1.sum()/400)
print(X1.shape)



plt.plot(X1,ypred1)#regression line
plt.plot(X1,Y1,"o")#plots
plt.show()
```

```
Error smoking    85.369294
dtype: float64
(498, 1)
```



[ ]:
```
Result:
        The Above Program were Created and Executed Successfully
```