6CS4-22: Machine Learning Lab **(Date : 03/04/2020)**

Experiment No. : 02

Instructor: Mr. Katkar Atish R. *Name:* Anjali Malav*, Roll number:* 17EEJCS002

AIM

Implementation of simple & multiple linear regression using Python

INTRODUCTION TO LINEAR REGRESSION

The term "linearity" in algebra refers to a linear relationship between two or more variables. If the relationship is drawn in a two-dimensional space (between two variables), a straight line is found. Linear regression is used for finding linear relationship between target and one or more predictors. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). This regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

There are two types of linear regression-

— Simple linear regression

— Multiple linear regression

Simple linear regression is useful for finding relationship between two continuous variables. One is predictor or independent variable and other is response or dependent variable. It looks for statistical relationship but not deterministic relationship. Relationship between two variables is said to be deterministic if one variable can be accurately expressed by the other. For example, using temperature in degree Celsius it is possible to accurately predict Fahrenheit. Statistical relationship is not accurate in determining relationship between two variables. For example, relationship between height and weight. The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (all data points) are as small as possible. Error is the distance between the point to the regression line.The equation for plotting a line using simple linear regression :

$$y = mx + c$$

Where b is the intercept and m is the slope of the line. The linear regression algorithm gives the most optimal value for the intercept and the slope (in two dimensions). The y and x variables remain the same, since they are the data features and cannot be changed. There can be multiple straight lines depending upon the values of intercept and slope.

Multiple linear regression use the concept of simple linear regression but in multiple regression more then two variable exist. This is called multiple linear regression. For instance, consider a scenario where you have to predict the price of the house based upon its area, number of bedrooms, the average income of the people in the area, the age of the house, and so on. In this case, the dependent variable(target variable) is dependent upon several independent variables. A regression model involving multiple variables can be represented as :

$$y = b_0 + m_1 b_1 + m_2 b_2 + m_3 b_3 + \ldots\ldots m_n b_n$$

This is the equation of a hyperplane, where $b_1, b_2, b_3 \ldots\ldots b_n$ are independent variable and slopes for different planes are given as $m_1, m_2, m_3 \ldots\ldots m_n$. A linear regression model in two dimensions is a straight line, in three dimensions it is a plane, and in more than three dimensions, a hyperplane.

PERFORMANCE MEASURE

Performance measure is required to evaluate the performance of the algorithm. To measure the performance of algorithm we want the total number of features vectors (n), actual(y_j) and predicted values($\widehat{y}$). For regression algorithms, three evaluation metrics are commonly used :

Mean Absolute Error (MAE)

MAE is the mean of the absolute value of the errors. It is calculated as :

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |y_j - \widehat{y}|$$

Mean Squared Error (MSE)

MSE is the mean of the squared errors and is calculated as :

$$MSE = \frac{1}{n}\sum_{j=1}^{n} (y_j - \widehat{y})^2$$

Root Mean Squared Error (RMSE)

RMSE is the square root of the mean of the squared errors :

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n} (y_j - \widehat{y})^2}$$

DATASET DESCRIPTION

In this experimental study we have used two different datasets (weather dataset & winequality dataset) for implementation, one for implementing simple linear regression and another for implementing multiple linear regression :
The dataset identified for simple linear regression experimental study is Weather dataset. The dataset has 32 columns and 119041 rows. In 32 columns data of min temprature , maximum temprature, humidity, mean temprature etc. is given.
The dataset identified for multiple linear regression experimental study is Winequality dataset. The dataset has 13 columns and 1600 rows. In 13 columns data of fixed acidity, volatile acidity, citric acid, residual sugar, chlorides etc. is given.

IMPLEMENTATION CODE FOR SIMPLE LINEAR REGRESSION

```
#PROGRAMME FOR SIMPLE LINEAR REGRESSION

# Packages imported from different libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
get_ipython().run_line_magic('matplotlib', 'inline')

# Loading CSV dataset using pandas
dataset = pd.read_csv('/home/asus/Desktop/machine learing/Weather.csv')

# Display the shape of dataset
dataset.shape

# Show dataset
dataset.describe()

# Ploting two attributes of dataset
dataset.plot(x='MinTemp', y='MaxTemp', style='o')
plt.title('MinTemp vs MaxTemp')
plt.xlabel('MinTemp')
plt.ylabel('MaxTemp')
plt.show()

# Plot graph for MaxTemp
plt.figure(figsize=(15,10))
plt.tight_layout()
seabornInstance.distplot(dataset['MaxTemp'])

# Assigning MinTemp values to X & MaxTemp values to Y
X = dataset['MinTemp'].values.reshape(-1,1)
y = dataset['MaxTemp'].values.reshape(-1,1)

# Splitting train & test data in 8:2 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=0)

# importing LinearRegression model in regressor
regressor = LinearRegression()
regressor.fit(X_train, y_train) #training the algorithm

#To retrieve the intercept
print(regressor.intercept_)

#For retrieving the slope
```

```
print(regressor.coef_)

# Finding predicted values for X_test data
y_pred = regressor.predict(X_test)

# Comparing the actual output values for X_test with the predicted values
df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
df

# Ploting actual & predicted values for intial 25 values
df1 = df.head(25)
df1.plot(kind='bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()

#  Ploting straight line with the test data
plt.scatter(X_test, y_test,  color='gray')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()

# Printing different errors
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
    y_pred)))
```

OUTPUT FOR SIMPLE LINEAR REGRESSION

1. **Output from dataset.describe() :**

	STA	WindGustSpd	MaxTemp	MinTemp	MeanTemp	YR	MO	DA	DR	SPD	...
count	119040.000000	532.000000	119040.000000	119040.000000	119040.000000	119040.000000	119040.000000	119040.000000	533.000000	532.000000	...
mean	29659.435795	37.774534	27.045111	17.789511	22.411631	43.805284	6.726016	15.797530	26.998124	20.396617	...
std	20953.209402	10.297808	8.717817	8.334572	8.297982	1.136718	3.425561	8.794541	15.221732	5.560371	...
min	10001.000000	18.520000	-33.333333	-38.333333	-35.555556	40.000000	1.000000	1.000000	2.000000	10.000000	...
25%	11801.000000	29.632000	25.555556	15.000000	20.555556	43.000000	4.000000	8.000000	11.000000	16.000000	...
50%	22508.000000	37.040000	29.444444	21.111111	25.555556	44.000000	7.000000	16.000000	32.000000	20.000000	...
75%	33501.000000	43.059000	31.666667	23.333333	27.222222	45.000000	10.000000	23.000000	34.000000	23.250000	...
max	82506.000000	75.932000	50.000000	34.444444	40.000000	45.000000	12.000000	31.000000	78.000000	41.000000	...

FIGURE 1: Output from data.describe()

2. **Output from plt.show() :**

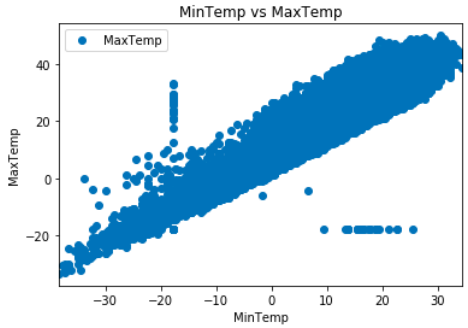

FIGURE 2: Plotting between minimum and maximum temperature

3. **Output from ploting of MaxTemp :**

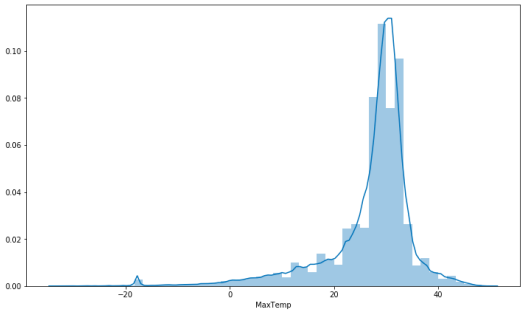

FIGURE 3: Plotting maximum temperature data

4. **Plotting graph for actual and predicted data :**

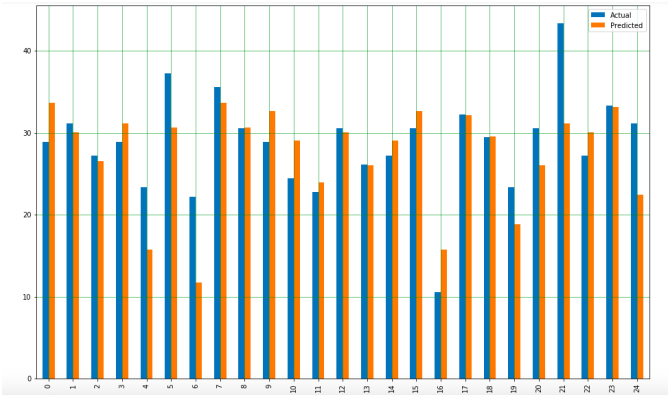

FIGURE 4: Graph showing difference between actual & predicted data

5. **Line plotted for simple linear regression**

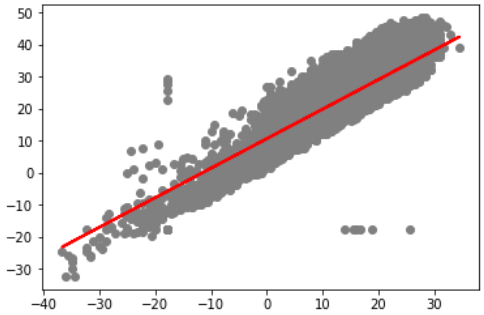

FIGURE 5: Line plotted by simple linear regression

```
------------------------- PERFORMANCE MEASURES--------------------------

Anjali_Malav/Machine_Learning \$
Mean Absolute Error: 3.19932917837853
Mean Squared Error: 17.631568097568447
Root Mean Squared Error: 4.198996082109204
```

IMPLEMENTATION CODE FOR MULTIPLE LINEAR REGRESSION

```
# PROGRAMME FOR MULTIPLE LINEAR REGRESSION

# Packages imported from different libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
get_ipython().run_line_magic('matplotlib', 'inline')

# Loading CSV dataset using pandas
dataset = pd.read_csv('/home/asus/Desktop/machine learing/winequality.csv')

# Display the shape of dataset
dataset.shape
```

```
18
19 # Show dataset
20 dataset.describe()
21
22 # Function to check whether any data is NULL
23 dataset.isnull().any()
24
25 # Filling NULL values to 0
26 dataset = dataset.fillna(method='ffill')
27
28 # Assigning values to X & y
29 X = dataset[['fixed acidity', 'volatile acidity', 'citric acid', 'residual
       sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density
       ', 'pH', 'sulphates','alcohol']].values
30 y = dataset['quality'].values
31
32 # Plotting graph for quality
33 plt.figure(figsize=(15,10))
34 plt.tight_layout()
35 seabornInstance.distplot(dataset['quality'])
36
37 # Splitting training & testing data in 8:2 ratio
38 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
       random_state=0)
39
40 # Importing model and training it for train data
41 regressor = LinearRegression()
42 regressor.fit(X_train, y_train)
43
44 # Genrating dataframe using pandas
45 coeff_df = pd.DataFrame(regressor.coef_, dataset.columns[0:11], columns=['
       Coefficient'])
46 coeff_df
47
48 # Saving predicted data for test dataset
49 y_pred = regressor.predict(X_test)
50
51 # Generating list for training & testing data
52 df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
53 df1 = df.head(25)
54 df1
55
56 # Graph showing difference between actual & predicted data
57 df1.plot(kind='bar',figsize=(10,8))
58 plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
59 plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
60 plt.show()
61
62 # Performance measure
63 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
64 print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
65 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
       y_pred)))
```

OUTPUT FOR SIMPLE LINEAR REGRESSION

1. **Output from dataset.describe() :**

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000

FIGURE 6: Output from data.describe()

2. **Graph for quality :**
   This function is used to plot datapoints between minTemp and MaxTemp attributes.

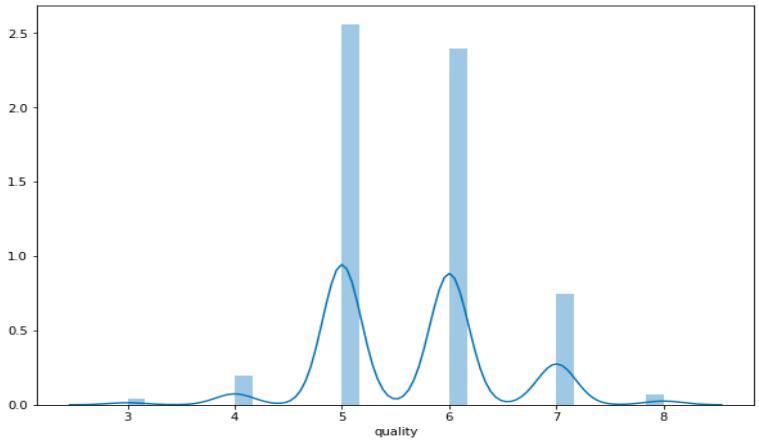

FIGURE 7: Plotting graph for quality

3. **List of coefficient for different attributes :**

	Coefficient
fixed acidity	0.041284
volatile acidity	-1.149528
citric acid	-0.177927
residual sugar	0.027870
chlorides	-1.873407
free sulfur dioxide	0.002684
total sulfur dioxide	-0.002777
density	-31.516666
pH	-0.254486
sulphates	0.924040
alcohol	0.267797

FIGURE 8: coefficient of different attributes

4. **Plotting graph for actual and predicted data :**

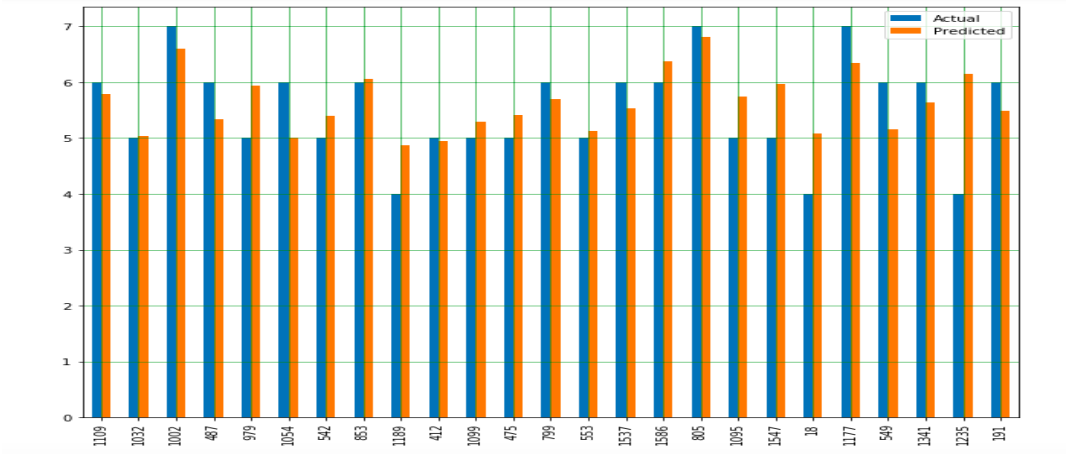

FIGURE 9: Graph showing difference between actual & predicted data

```
------------------------- PERFORMANCE MEASURES--------------------------

Anjali_Malav/Machine_Learning \$
Mean Absolute Error: 0.4696330928661103
Mean Squared Error: 0.38447119782012373
Root Mean Squared Error: 0.6200574149384263
```

CONCLUSION

Simple linear regression gave 4.19 root mean squared error, which is more than 10% of the mean value of the percent of all the temperature i.e. 22.41. This means that our algorithm was not very accurate but can still make reasonably good predictions.
Multiple linear regression gave 0.62 root mean squared error, which is slightly greater than 10% of the mean value which is 5.63. This means that our algorithm was not very accurate but can still make reasonably good predictions.