

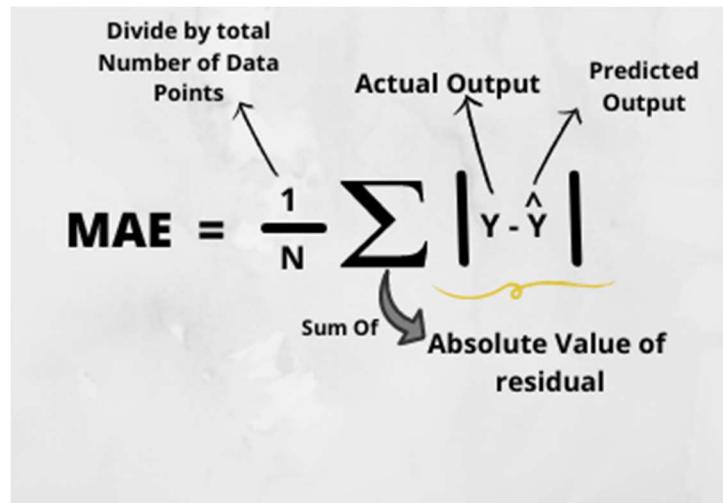
## **DSS Lab Experiment Number: 7**

AIM: To implement Linear Regression Model Evaluation

**Software Required:** Python

### **1) Mean Absolute Error(MAE)**

MAE is a very simple metric which calculates the absolute difference between actual and predicted values. To better understand, let's take an example you have input data and output data and use Linear Regression, which draws a best-fit line. Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset. so, sum all the errors and divide them by a total number of observations And this is MAE. And we aim to get a minimum MAE because this is a loss.



#### **Advantages of MAE**

- The MAE you get is in the same unit as the output variable.
- It is most Robust to outliers.

#### **Disadvantages of MAE**

- The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.

Code:

```
from sklearn.metrics import mean_absolute_error  
print("MAE",mean_absolute_error(y_test,y_pred))
```

## 2) Mean Squared Error(MSE)

MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value. So, above we are finding the absolute difference and here we are finding the squared difference.

What actually the MSE represents? It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n} \sum \underbrace{\left( y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

### Advantages of MSE

The graph of MSE is differentiable, so you can easily use it as a loss function.

### Disadvantages of MSE

- The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.
- If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

Code:

```
from sklearn.metrics import mean_squared_error  
print("MSE",mean_squared_error(y_test,y_pred))
```

### 3) Root Mean Squared Error(RMSE)

As RMSE is clear by the name itself, that it is a simple square root of mean squared error.

$$\text{RMSE} = \sqrt{\text{MSE}}$$
$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

#### Advantages of RMSE

- The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

#### Disadvantages of RMSE

- It is not that robust to outliers as compared to MAE.

for performing RMSE we have to NumPy NumPy square root function over MSE.

```
code  
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
```

Most of the time people use RMSE as an evaluation metric and mostly when you are working with deep learning techniques the most preferred metric is RMSE.

### 4) Root Mean Squared Log Error(RMSLE)

Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale.

To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE.

To perform RMSLE we have to use the NumPy log function over RMSE.

```
print("RMSE",np.log(np.sqrt(mean_squared_error(y_test,y_pred))))
```

It is a very simple metric that is used by most of the datasets hosted for Machine Learning competitions.

## 5) R Squared (R2)

R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context.

So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how much regression line is better than a mean line.

Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.

$$\mathbf{R^2 \text{ Squared} = 1 - \frac{SS_r}{SS_m}}$$

**SS<sub>r</sub> = Squared sum error of regression line**

**SS<sub>m</sub> = Squared sum error of mean line**

Now, how will you interpret the R2 score? suppose If the R2 score is zero then the above regression line by mean line is equal means 1 so 1-1 is zero. So, in this case, both lines are overlapping means model performance is worst, It is not capable to take advantage of the output column. Now the second case is when the R2 score is 1, it means when the division term is zero and it will happen when the regression line does not make any mistake, it is perfect. In the real world, it is not possible. So we can conclude that as our regression line moves towards perfection, R2 score move

towards one. And the model performance improves. The normal case is when the R2 score is between zero and one like 0.8 which means your model is capable to explain 80 per cent of the variance of data.

Code:

```
from sklearn.metrics import r2_score  
r2 = r2_score(y_test,y_pred)  
print(r2)
```

## 6) Adjusted R Squared

The disadvantage of the R2 score is while adding new features in data the R2 score starts increasing or remains constant but it never decreases because It assumes that while adding more data variance of data increases. But the problem is when we add an irrelevant feature in the dataset then at that time R2 sometimes starts increasing which is incorrect.

Hence, To control this situation Adjusted R Squared came into existence.

$$R_a^2 = 1 - \left[ \left( \frac{n-1}{n-k-1} \right) \times (1 - R^2) \right]$$

where:

n = number of observations  
k = number of independent variables  
 $R_a^2$  = adjusted  $R^2$

Now as K increases by adding some features so the denominator will decrease, n-1 will remain constant. R2 score will remain constant or will increase slightly so the complete answer will increase and when we subtract this from one then the resultant score will decrease. so this is the case when we add an irrelevant feature in the dataset.

And if we add a relevant feature then the R2 score will increase and 1-R2 will decrease heavily and the denominator will also decrease so the complete term decreases, and on subtracting from one the score increases.

Code:

```
n=40  
k=2  
adj_r2_score = 1 - ((1-r2)*(n-1)/(n-k-1))  
print(adj_r2_score)
```

Hence, this metric becomes one of the most important metrics to use during the evaluation of the model.

### **Mean Absolute Error Code**

```
from sklearn.metrics import mean_absolute_error  
print("MAE",mean_absolute_error(y_test,y_pred))
```

### **Mean Squared Error(MSE)**

```
from sklearn.metrics import mean_squared_error  
print("MSE",mean_squared_error(y_test,y_pred))
```

### **Root Mean Squared Error(RMSE)**

```
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
```

### **R Squared (R2)**

```
from sklearn.metrics import r2_score r2 = r2_score(y_test,y_pred) print(r2)
```

### **Adjusted R Squared**

Conclusion:

In this way we have evaluated performance of linear regression model.