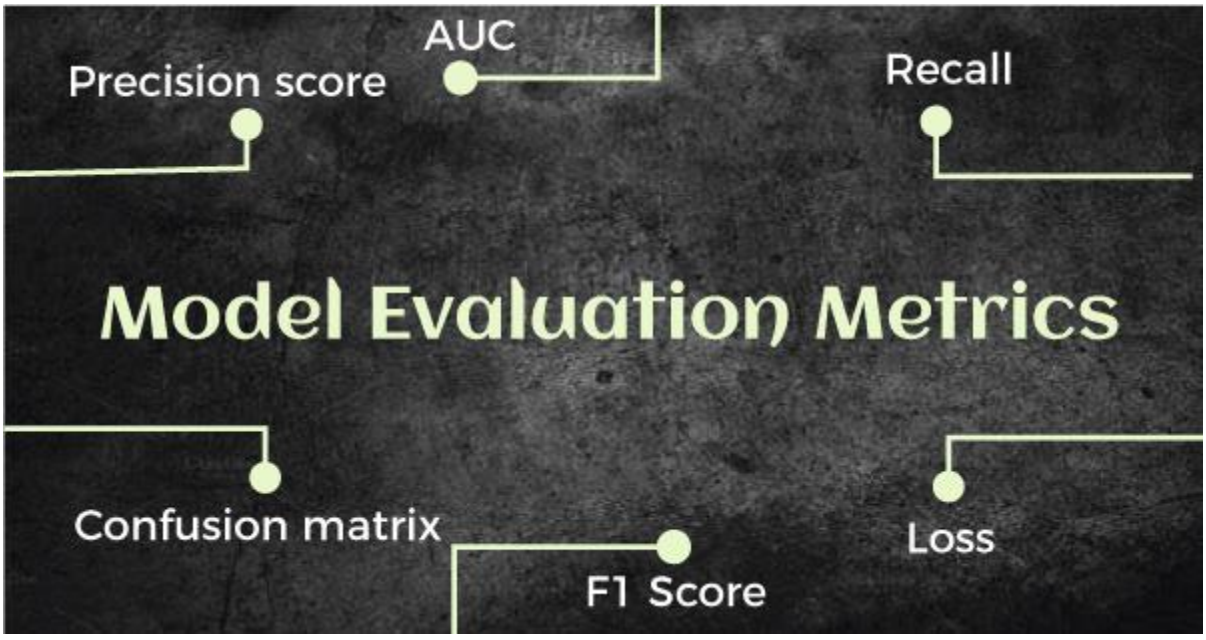
Performance Metrics in Machine Learning

Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. ***To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics.*** These performance metrics help us understand how well our model has performed for the given data. In this way, we can improve the model's performance by tuning the hyper-parameters. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.



In machine learning, each task or problem is divided into **classification** and **Regression**. Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used. Different evaluation metrics are used for both Regression and Classification tasks. In this topic, we will discuss metrics used for classification and regression tasks.

1. Performance Metrics for Classification

In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based

on the training. It predicts class labels as the output, such as *Yes or No, 0 or 1, Spam or Not Spam,* etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

- **Accuracy**
- **Confusion Matrix**
- **Precision**
- **Recall**
- **F-Score**
- **AUC(Area Under the Curve)-ROC**

I. Accuracy

The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.

It can be formulated as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predictions}$$

To implement an accuracy metric, we can compare ground truth and predicted values in a loop, or we can also use the scikit-learn module for this.

Firstly, we need to import the *accuracy_score* function of the scikit-learn library as follows:

```
1. from sklearn.metrics import accuracy_score
2.
3. Here, metrics is a class of sklearn.
4.
```

5. Then we need to pass the ground truth and predicted values in the function to calculate t he accuracy.
6.
7. print(f'Accuracy Score is {accuracy_score(y_test,y_hat)}')

Although it is simple to use and implement, it is suitable only for cases where an equal number of samples belong to each class.

When to Use Accuracy?

It is good to use the Accuracy metric when the target variable classes in data are approximately balanced. For example, if 60% of classes in a fruit image dataset are of Apple, 40% are Mango. In this case, if the model is asked to predict whether the image is of Apple or Mango, it will give a prediction with 97% of accuracy.

When not to use Accuracy?

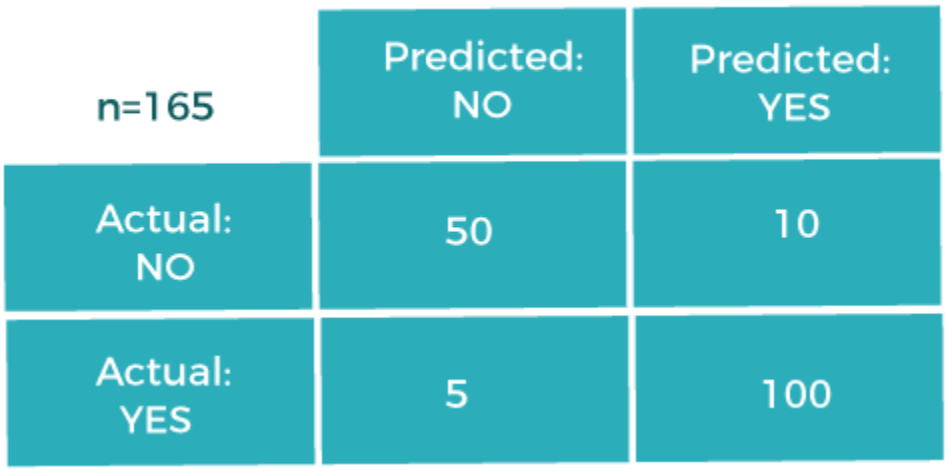
It is recommended not to use the Accuracy measure when the target variable majorly belongs to one class. For example, Suppose there is a model for a disease prediction in which, out of 100 people, only five people have a disease, and 95 people don't have one. In this case, if our model predicts every person with no disease (which means a bad prediction), the Accuracy measure will be 95%, which is not correct.

II. Confusion Matrix

A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known.

The confusion matrix is simple to implement, but the terminologies used in this matrix might be confusing for beginners.

A typical confusion matrix for a binary classifier looks like the below image(However, it can be extended to use for classifiers with more than two classes).



n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

We can determine the following from the above matrix:

- In the matrix, columns are for the prediction values, and rows specify the Actual values. Here Actual and prediction give two possible classes, Yes or No. So, if we are predicting the presence of a disease in a patient, the Prediction column with Yes means, Patient has the disease, and for NO, the Patient doesn't have the disease.
- In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No.
- However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

In general, the table is divided into four terminologies, which are as follows:

1. **True Positive(TP):** In this case, the prediction outcome is true, and it is true in reality, also.
2. True Negative(TN): in this case, the prediction outcome is false, and it is false in reality, also.

3. False Positive(FP): In this case, prediction outcomes are true, but they are false in actuality.
4. False Negative(FN): In this case, predictions are false, and they are true in actuality.

III. Precision

The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive prediction that was actually correct. It can be calculated as the True Positive or predictions that are actually true to the total positive predictions (True Positive and False Positive).

$$Precision = \frac{TP}{(TP + FP)}$$

IV. Recall or Sensitivity

It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positive that was identified incorrectly. It can be calculated as True Positive or predictions that are actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative).

The formula for calculating Recall is given below:

$$\text{Recall} = \frac{TP}{TP+FN}$$

When to use Precision and Recall?

From the above definitions of Precision and Recall, we can say that recall determines the performance of a classifier with respect to a false negative, whereas precision gives information about the performance of a classifier with respect to a false positive.

So, if we want to minimize the false negative, then, Recall should be as near to 100%, and if we want to minimize the false positive, then precision should be close to 100% as possible.

In simple words, *if we maximize precision, it will minimize the FP errors, and if we maximize recall, it will minimize the FN error.*

V. F-Scores

F-score or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, ***the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.***

The formula for calculating the F1 score is given below:

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

When to use F-Score?

As F-score make use of both precision and recall, so it should be used if both of them are important for evaluation, but one (precision or recall) is slightly more important to consider than the other. For example, when False negatives are comparatively more important than false positives, or vice versa.

VI. AUC-ROC

Sometimes we need to visualize the performance of the classification model on charts; then, we can use the AUC-ROC curve. It is one of the popular and important metrics for evaluating the performance of the classification model.

Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve. ***ROC represents a graph to show the performance of a classification model at different threshold levels***. The curve is plotted between two parameters, which are:

- **True Positive Rate**
- **False Positive Rate**

TPR or true Positive rate is a synonym for Recall, hence can be calculated as:

$$TPR = \frac{TP}{TP + FN}$$

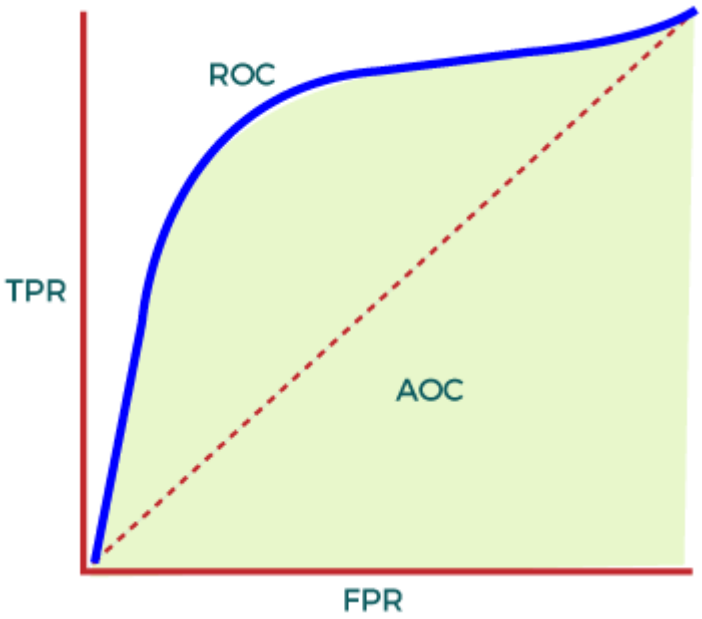
FPR or False Positive Rate can be calculated as:

$$TPR = \frac{FP}{FP + TN}$$

To calculate value at any point in a ROC curve, we can evaluate a logistic regression model multiple times with different classification thresholds, but this would not be much efficient. So, for this, one efficient method is used, which is known as AUC.

AUC: Area Under the ROC curve

AUC is known for **Area Under the ROC curve**. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve, as shown below image:



AUC calculates the performance across all the thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1. It means a model with 100% wrong prediction will have an AUC of 0.0, whereas models with 100% correct predictions will have an AUC of 1.0.

When to Use AUC

AUC should be used to measure how well the predictions are ranked rather than their absolute values. Moreover, it measures the quality of predictions of the model without considering the classification threshold.

When not to use AUC

As AUC is scale-invariant, which is not always desirable, and we need calibrating probability outputs, then AUC is not preferable.

Further, AUC is not a useful metric when there are wide disparities in the cost of false negatives vs. false positives, and it is difficult to minimize one type of classification error.

2. Performance Metrics for Regression

Regression is a supervised learning technique that aims to find the relationships between the dependent and independent variables. A predictive regression model predicts a numeric or discrete value. The metrics used for regression are different from the classification metrics. It means we cannot use the Accuracy metric (explained above) to evaluate a regression model; instead, the performance of a Regression model is reported as errors in the prediction. Following are the popular metrics that are used to evaluate the performance of Regression models.

- **Mean Absolute Error**
- **Mean Squared Error**
- **R2 Score**
- **Adjusted R2**

I. Mean Absolute Error (MAE)

Mean Absolute Error or MAE is one of the simplest metrics, which measures the absolute difference between actual and predicted values, where absolute means taking a number as Positive.

To understand MAE, let's take an example of Linear Regression, where the model draws a best fit line between dependent and independent variables. To measure the MAE or error in prediction, we need to calculate the difference between actual values and predicted values. But in order to find the absolute error for the complete dataset, we need to find the mean absolute of the complete dataset.

The below formula is used to calculate MAE:

$$MAE = 1/N \sum |Y - Y'|$$

Here,

Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.

MAE is much more robust for the outliers. One of the limitations of MAE is that it is not differentiable, so for this, we need to apply different optimizers such as Gradient Descent. However, to overcome this limitation, another metric can be used, which is Mean Squared Error or MSE.

II. Mean Squared Error

Mean Squared error or MSE is one of the most suitable metrics for Regression evaluation. It measures the average of the Squared difference between predicted values and the actual value given by the model.

Since in MSE, errors are squared, therefore it only assumes non-negative values, and it is usually positive and non-zero.

Moreover, due to squared differences, it penalizes small errors also, and hence it leads to over-estimation of how bad the model is.

MSE is a much-preferred metric compared to other regression metrics as it is differentiable and hence optimized better.

The formula for calculating MSE is given below:

$$MSE = 1/N \sum (Y - Y')^2$$

Here,

Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.

III. R Squared Score

R squared error is also known as Coefficient of Determination, which is another popular metric used for Regression model evaluation. The R-squared metric enables us to compare our model

with a constant baseline to determine the performance of the model. To select the constant baseline, we need to take the mean of the data and draw the line at the mean.

The R squared score will always be less than or equal to 1 without concerning if the values are too large or small.

$$R^2 = 1 - \frac{MSE(Model)}{MSE(Baseline)}$$

IV. Adjusted R Squared

Adjusted R squared, as the name suggests, is the improved version of R squared error. R square has a limitation of improvement of a score on increasing the terms, even though the model is not improving, and it may mislead the data scientists.

To overcome the issue of R square, adjusted R squared is used, which will always show a lower value than R^2. It is because it adjusts the values of increasing predictors and only shows improvement if there is a real improvement.

We can calculate the adjusted R squared as follows:

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

Here,

n is the number of observations

k denotes the number of independent variables

and R_a^2 denotes the adjusted R^2

Machine Learning - Performance Metrics

There are various metrics which we can use to evaluate the performance of ML algorithms, classification as well as regression algorithms. We must carefully choose the metrics for evaluating ML performance because −

- How the performance of ML algorithms is measured and compared will be dependent entirely on the metric you choose.
- How you weight the importance of various characteristics in the result will be influenced completely by the metric you choose.

Performance Metrics for Classification Problems

We have discussed classification and its algorithms in the previous chapters. Here, we are going to discuss various performance metrics that can be used to evaluate predictions for classification problems.

Confusion Matrix

It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes. A confusion matrix is nothing but a table with two dimensions viz. “Actual” and “Predicted” and furthermore, both the dimensions have “True Positives (TP)”, “True Negatives (TN)”, “False Positives (FP)”, “False Negatives (FN)” as shown below −

		Actual: 1	Actual: 0
Predicted	1	True Positives (TP)	False Positives (FP)
	0		True Negatives (TN)

Explanation of the terms associated with confusion matrix is as follows −

- **True Positives (TP)** − It is the case when both actual class & predicted class of data point is 1.
- **True Negatives (TN)** − It is the case when both actual class & predicted class of data point is 0.
- **False Positives (FP)** − It is the case when actual class of data point is 0 & predicted class of data point is 1.
- **False Negatives (FN)** − It is the case when actual class of data point is 1 & predicted class of data point is 0.

We can use confusion_matrix function of sklearn.metrics to compute Confusion Matrix of our classification model.

Classification Accuracy

It is most common performance metric for classification algorithms. It may be defined as the number of correct predictions made as a ratio of all predictions made. We can easily calculate it by confusion matrix with the help of following formula −

$$Accuracy=TP+TNTP+FP+FN+TN$$

We can use accuracy_score function of sklearn.metrics to compute accuracy of our classification model.

Classification Report

This report consists of the scores of Precisions, Recall, F1 and Support. They are explained as follows −

Precision

Precision, used in document retrievals, may be defined as the number of correct documents returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

Precision=TPTP+FP

Recall or Sensitivity

Recall may be defined as the number of positives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

Recall=TPTP+FN

Specificity

Specificity, in contrast to recall, may be defined as the number of negatives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

Specificity=TNTN+FP

Support

Support may be defined as the number of samples of the true response that lies in each class of target values.

F1 Score

This score will give us the harmonic mean of precision and recall. Mathematically, F1 score is the weighted average of the precision and recall. The best value of F1 would be 1 and worst would be 0. We can calculate F1 score with the help of following formula −

$$\boldsymbol{F1} = \mathbf{2} * (\boldsymbol{precision} * \boldsymbol{recall}) / (\boldsymbol{precision} + \boldsymbol{recall})$$

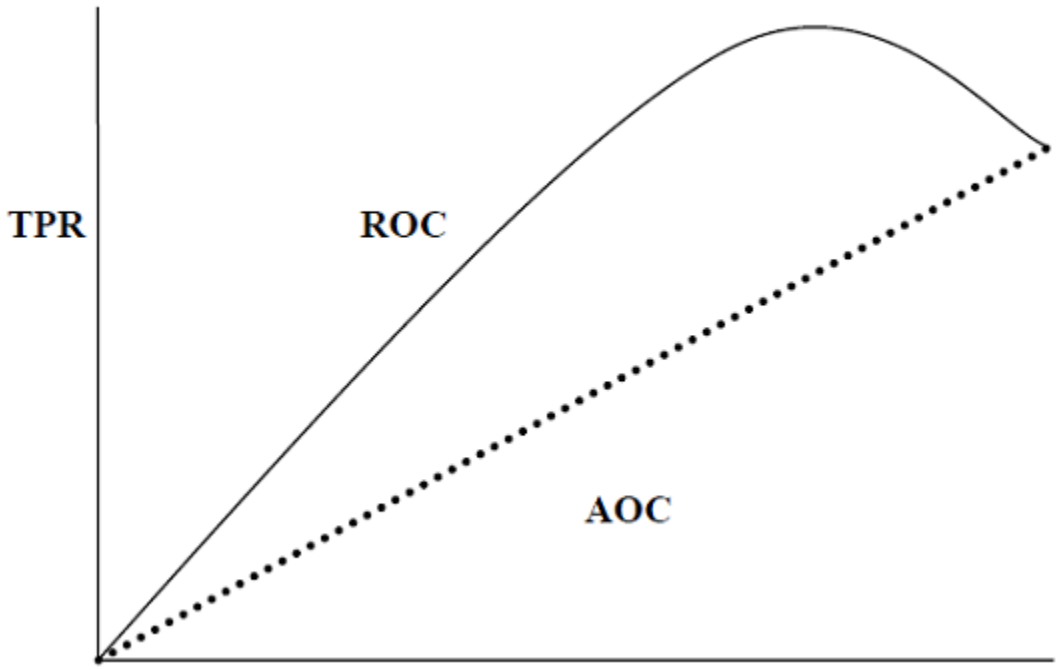
F1 score is having equal relative contribution of precision and recall.

We can use classification_report function of sklearn.metrics to get the classification report of our classification model.

AUC (Area Under ROC curve)

AUC (Area Under Curve)-ROC (Receiver Operating Characteristic) is a performance metric, based on varying threshold values, for classification problems. As name suggests, ROC is a probability curve and AUC measure the separability. In simple words, AUC-ROC metric will tell us about the capability of model in distinguishing the classes. Higher the AUC, better the model.

Mathematically, it can be created by plotting TPR (True Positive Rate) i.e. Sensitivity or recall vs FPR (False Positive Rate) i.e. 1-Specificity, at various threshold values. Following is the graph showing ROC, AUC having TPR at y-axis and FPR at x-axis −



We can use roc_auc_score function of sklearn.metrics to compute AUC-ROC.

LOGLOSS (Logarithmic Loss)

It is also called Logistic regression loss or cross-entropy loss. It basically defined on probability estimates and measures the performance of a classification model where the input is a probability value between 0 and 1. It can be understood more clearly by differentiating it with accuracy. As we know that accuracy is the count of predictions (predicted value = actual value) in our model whereas Log Loss is the amount of uncertainty of our prediction based on how much it varies from

the actual label. With the help of Log Loss value, we can have more accurate view of the performance of our model. We can use log_loss function of sklearn.metrics to compute Log Loss.

Example

The following is a simple recipe in Python which will give us an insight about how we can use the above explained performance metrics on binary classification model −

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import log_loss
X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]
results = confusion_matrix(X_actual, Y_predic)
print ('Confusion Matrix :')
print(results)
print ('Accuracy Score is',accuracy_score(X_actual, Y_predic))
print ('Classification Report : ')
print (classification_report(X_actual, Y_predic))
print('AUC-ROC:',roc_auc_score(X_actual, Y_predic))
print('LOGLOSS Value is',log_loss(X_actual, Y_predic))
```

Output

```
Confusion Matrix :
[
  [3 3]
  [1 3]
]
Accuracy Score is 0.6
Classification Report :
```

	precision	recall	f1-score	support
0	0.75	0.50	0.60	6
1	0.50	0.75	0.60	4
micro avg	0.60	0.60	0.60	10
macro avg	0.62	0.62	0.60	10
weighted avg	0.65	0.60	0.60	10

AUC-ROC: 0.625

LOGLOSS Value is 13.815750437193334

Performance Metrics for Regression Problems

We have discussed regression and its algorithms in previous chapters. Here, we are going to discuss various performance metrics that can be used to evaluate predictions for regression problems.

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. In simple words, with MAE, we can get an idea of how wrong the predictions were. MAE does not indicate the direction of the model i.e. no indication about underperformance or overperformance of the model. The following is the formula to calculate MAE −

$$MAE=1n\sum|Y-Y^|$$

Here, Y=Actual Output Values

And $Y^$

= Predicted Output Values.

We can use mean_absolute_error function of sklearn.metrics 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. The difference can be noticed in the following equation −

$$MSE=1n\sum(Y-Y^{\wedge})$$

Here, Y=Actual Output Values

And $Y^{\wedge}$

= Predicted Output Values.

We can use mean_squared_error function of sklearn.metrics to compute MSE.

R Squared (R^2)

R Squared metric is generally used for explanatory purpose and provides an indication of the goodness or fit of a set of predicted output values to the actual output values. The following formula will help us understanding it −

$$R2=1-1n\sum ni=1(Yi-Yi^{\wedge})21n\sum ni=1(Yi-Yi)2^{-}$$

In the above equation, numerator is MSE and the denominator is the variance in Y values.

We can use r2_score function of sklearn.metrics to compute R squared value.

Example

The following is a simple recipe in Python which will give us an insight about how we can use the above explained performance metrics on regression model −

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
X_actual = [5, -1, 2, 10]
```

```
Y_predic = [3.5, -0.9, 2, 9.9]
print ('R Squared =',r2_score(X_actual, Y_predic))
print ('MAE =',mean_absolute_error(X_actual, Y_predic))
print ('MSE =',mean_squared_error(X_actual, Y_predic))
```

Output

```
R Squared = 0.9656060606060606
MAE = 0.42499999999999993
MSE = 0.5674999999999999
```