Evaluation Metrics

Majorly the given metrics are more than sufficient to evaluate the model at hand. However, as one progresses through different fields of Machine learning, they may encounter new metrics based on the problem at hand.

- Confusion Matrix
- Classification Accuracy.
- Logarithmic loss.
- Area under Curve.
- F1 score.
- Mean Absolute Error.
- Mean Squared Error.

Confusion Matrix:

It creates a *N X N* matrix, where N is the number of classes or categories that are to be predicted. Here we have *N = 2*, so we get a *2 X 2* matrix. Suppose there is a problem for our practice which is a binary classification. Samples of that classification belong to either *Yes* or *No*. So, we build our classifier which will predict the class for a new input sample. After that, we have tested our model with *165* samples, and we get the following result.

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

There are 4 terms you should keep in mind:

1. **True Positives:** It is the case where we predicted Yes and the real output was also yes.
2. **True Negatives:** It is the case where we predicted No and the real output was also No.
3. **False Positives:** It is the case where we predicted Yes but it was actually No.
4. **False Negatives:** It is the case where we predicted No but it was actually Yes.

----------X----------X--------

The above mentioned 4 terms are more important than accuracy in a binary classification problem. Because Often in the real world, accuracy doesn't matter. For example, you are to predict whether an asteroid will hit earth or not. Now If you predict No all the time, then you would have an accuracy of over 99%. Since any random asteroid does not hit the earth majority of times. But in real terms, our prediction is useless, as it does not prepare us to deal with the asteroid.

In such cases, the exact value of False positives and False negatives comes into play and tells us if our model is of any use or not.

Accuracy of the matrix is always calculated by taking average values present in the ***main diagonal i.e.***

$$Accuracy = (TruePositive + TrueNegative)/TotalSampleAccuracy = (100 + 50)/165Accuracy = 0.91$$

Classification Accuracy:

Classification accuracy is the accuracy we generally mean, Whenever we use the term accuracy. We calculate this by calculating the ratio of correct predictions by a total number of input Samples.

$$Accuray = No.of correct predictions/Total number of input samples.$$

It works great if there are an equal number of samples for each class. For example, we have 90% sample of *class A* and 10% sample of *class B* in our training set. Then, our model will predict with the accuracy of 90% by predicting all the training samples belonging to *class A*. If we test the same model with a

test set of 60% from class A and 40% from class B. Then the accuracy will fall, and we will get an accuracy of 60%.

Classification accuracy is good but it gives False Positive sense of achieving high accuracy. The problem arises due to the possibility of mis-classification of minor class samples.

Logarithmic loss.

It is also known as Log loss. Its basic working propaganda is by penalizing the false (False Positive) classification. It usually works well with multi-class classification. Working on Log loss, the classifier should assign a probability for each and every class of all the samples. If there are N samples belong to *M* class, then we calculate the Log loss in this way :

$$LogarithmicLoss = -1 \frac{}{N \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} * \log(p_{ij})}$$

Now the Terms,

- *y_ij* indicates whether sample *i* belongs to class j.
- *p_ij* – the probability of sample *i* belongs to class j.
- Rang of log loss is [0,?). When the log loss is near 0 it indicates high accuracy and when away from zero then, it indicates lower accuracy.

- Let me give you a bonus point, minimizing log loss gives you higher accuracy for the classifier.

Area Under Curve (A U C):

It is one of the widely used metrics and basically used for binary classification. A U C of a classifier is defined as the probability of a classifier that will rank a randomly chosen positive example higher than a negative example. Before going into A U C more, let me make you comfortable with few basic terms.

True positive rate: Also called or termed as sensitivity. True Positive Rate is considered as a portion of positive data points which are correctly considered as positive, with respect to all data points those are positives.

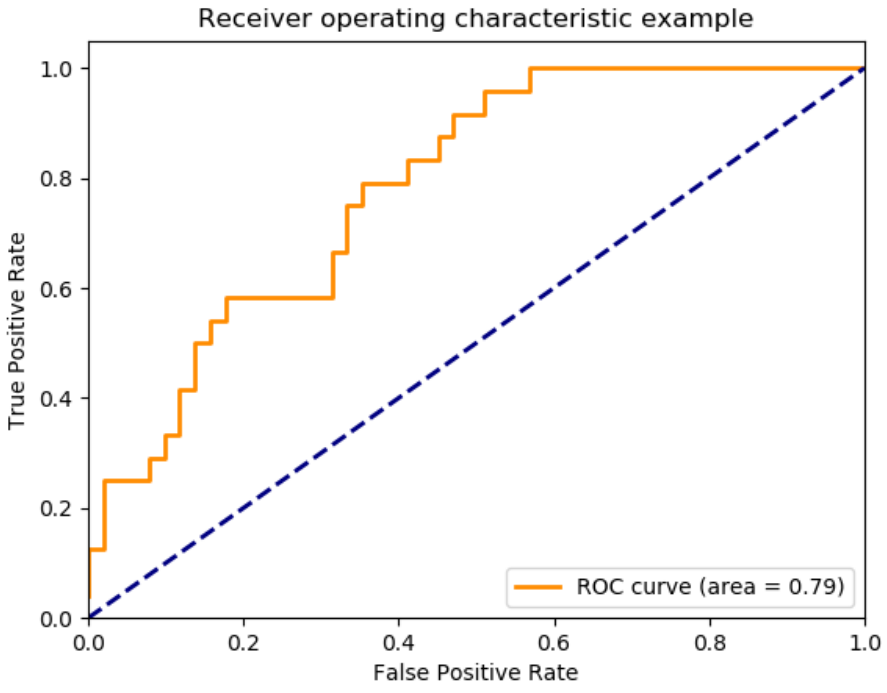
$$TruePositiveRate = TruePositive/(FalseNegative + TruePositive)$$

True Negative Rate: Also called or termed as specificity. False Negative Rate is considered as a portion of negative data points which are correctly considered as negative, with respected to all data points those are negatives.

$$TrueNegativeRate = TrueNegative/(TrueNegative + FalsePsotive)$$

False-positive Rate: False Negative Rate is considered as a portion of negative data points which are mistakenly considered as negative, with respect to all data points those are negatives.

False Positive Rate and True Positive Rate both have values in the range [0, 1]. Now the thing is what is A U C then? So, A U C is a curve plotted between False Positive Rate Vs True Positive Rate at all different data points with a range of [0, 1]. Greater the value of AUCC better the performance of the model.



A U C curve

F1 Score:

It is a harmonic mean between recall and precision. Its range is [0,1]. This metric usually tells us how precise (It correctly classifies how many instances) and robust (does not miss any significant number of instances) our classifier is.

It is used to measure the test's accuracy

Precision:

$$Precision = TruePositives/(TruePosities + FalsePositives)$$

Recall:

$$Recall = TruePositives/(TruePosities + FalsePositives)$$

Lower recall and higher precision give you great accuracy but then it misses a large number of instances. The Better the F1 score the better will be performance. It can be expressed mathematically in this way :

$$F1 = 2 * \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

Mean Absolute Error:

It is the average distance between Predicted and original values. Basically it gives how we have predicted from the actual output. However, there is one limitation i.e. it doesn't give any idea about the direction of the error which is whether we are under-predicting or over-predicting our data. It can be represented mathematically in this way :

$$MeanAbsoluteError = 1\frac{}{N\sum_{j=1}^{N}|y_j - \hat{y}_j|}$$

Mean Squared Error:

It is similar to mean absolute error but the difference is it takes the square of average of between predicted and original values. The main advantage to taking this metric is here, it is easier to calculate the gradient whereas in the case of mean absolute error it takes complicated programming tools to calculate the gradient. By taking the square of errors it pronounces larger errors more than smaller errors, we can focus more on larger errors. It can be expressed mathematically in this way :

$$MeanSquaredError = 1\frac{}{N\sum_{j=1}^{N}(y_j - \hat{y}_j)^2}$$