Model Performance Metrics

Classification

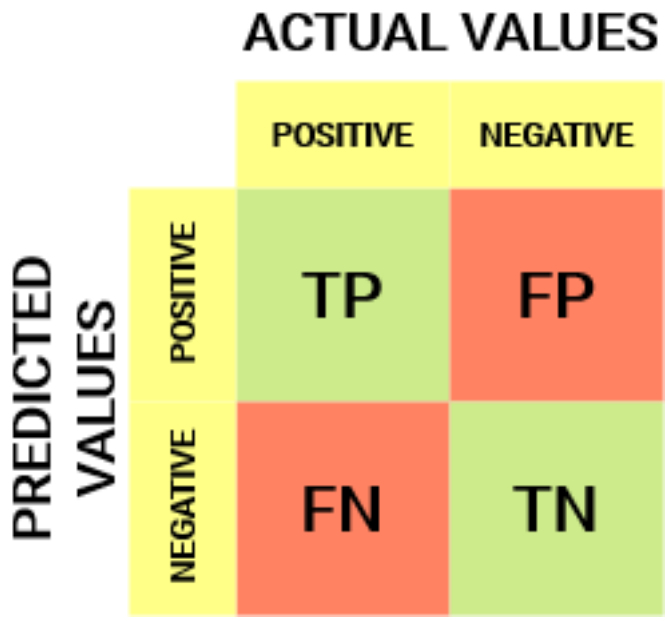
1- Confusion Matrix (not a metric, but important to know!)

What is a Confusion Matrix?

The million-dollar question – what, after all, is a confusion matrix?

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:



Let's decipher the matrix:

- The target variable has two values: **Positive** or **Negative**
- The **columns** represent the **actual values** of the target variable
- The **rows** represent the **predicted values** of the target variable

But wait – what's TP, FP, FN and TN here? That's the crucial part of a confusion matrix. Let's understand each term below.

Understanding True Positive, True Negative, False Positive and False Negative in a Confusion Matrix

True Positive (TP)

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

True Negative (TN)

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

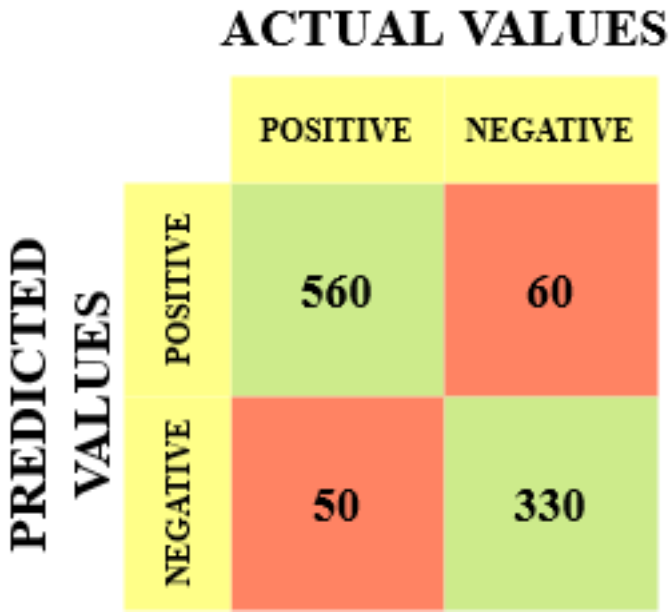
False Positive (FP) – Type 1 error

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the **Type 1 error**

False Negative (FN) – Type 2 error

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the **Type 2 error**

Let me give you an example to better understand this. Suppose we had a classification dataset with 1000 data points. We fit a classifier on it and get the below confusion matrix:



The different values of the Confusion matrix would be as follows:

- True Positive (TP) = 560; meaning 560 positive class data points were correctly classified by the model
- True Negative (TN) = 330; meaning 330 negative class data points were correctly classified by the model
- False Positive (FP) = 60; meaning 60 negative class data points were incorrectly classified as belonging to the positive class by the model
- False Negative (FN) = 50; meaning 50 positive class data points were incorrectly classified as belonging to the negative class by the model

This turned out to be a pretty decent classifier for our dataset considering the relatively larger number of true positive and true negative values.

2- Classification Accuracy

Classification accuracy is perhaps the simplest metrics one can imagine, and is defined as the **number of correct predictions divided by the total number of predictions,** multiplied by 100**.** So in the above example, out of 1100 samples 1030 are predicted correctly, resulting in a classification accuracy of:

Classification accuracy= (90+940)/(1000+100)= 1030/1100= 93.6%

3- Precision

Precision

Precision can be defined as the percentage of correctly predicted positive outcomes out of all the predicted positive outcomes. It can be given as the ratio of true positives (TP) to the sum of true and false positives (TP + FP).

So, **Precision** identifies the proportion of correctly predicted positive outcome. It is more concerned with the positive class than the negative class.

There are many cases in which classification accuracy is not a good indicator of your model performance. One of these scenarios is when your class distribution is imbalanced (one class is more frequent than others). In this case, even if you predict all samples as the most frequent class you would get a high accuracy rate, which does not make sense at all (because your model is not learning anything, and is just predicting everything as the top class). For example in our cat vs non-cat classification above, if the model predicts all samples as non-cat, it would result in a 1000/1100= 90.9%.

Therefore we need to look at class specific performance metrics too. Precision is one of such metrics, which is defined as:

Precision= True_Positive/ (True_Positive+ False_Positive)

The precision of Cat and Non-Cat class in above example can be calculated as:

Precision_cat= #samples correctly predicted cat/#samples predicted as cat

= **90/(90+60) = 60%**

Precision_NonCat= 940/950= 98.9%

As we can see the model has much higher precision in predicting non-cat samples, versus cats. This is not surprising, as model has seen more examples of non-cat images during training, making it better in classifying that class.

4- Recall

Recall is another important metric, which is defined as the fraction of samples from a class which are correctly predicted by the model. More formally:

Recall= True_Positive/ (True_Positive+ False_Negative)

Therefore, for our example above, the recall rate of cat and non-cat classes can be found as:

Recall_cat= 90/100= 90%

Recall_NonCat= 940/1000= 94%

5- F1 Score

Depending on application, you may want to give higher priority to recall or precision. But there are many applications in which both recall and precision are important. Therefore, it is natural to think of a way to combine these two into a single metric. **One popular metric which combines precision and recall is called F1-score**, which is the harmonic mean of precision and recall defined as:

F1-score= 2*Precision*Recall/(Precision+Recall)

So for our classification example with the confusion matrix in Figure 1, the F1-score can be calculated as:

F1_cat= 2*0.6*0.9/(0.6+0.9)= 72%

It is good to mention that there is always a trade-off between precision and recall of a model, if you want to make the precision too high, you would end up seeing a drop in the recall rate, and vice versa.

7- ROC Curve

The **receiver operating characteristic curve** is plot which shows the performance of a binary classifier as function of its cut-off threshold. **It essentially shows the true positive rate (TPR) against the false positive rate (FPR) for various threshold values. Let's explain more.**

Many of the classification models are probabilistic, i.e. they predict the probability of a sample being a cat. They then compare that output probability with some cut-off threshold and if it is larger than the threshold they predict its label as cat, otherwise as non-cat. As an example your model may predict the below probabilities for 4 sample images: **[0.45, 0.6, 0.7, 0.3]**. Then depending on the threshold values below, you will get different labels:

cut-off= 0.5: predicted-labels= [0,1,1,0] (default threshold)
cut-off= 0.2: predicted-labels= [1,1,1,1]
cut-off= 0.8: predicted-labels= [0,0,0,0]

As you can see by varying the threshold values, we will get completely different labels. And as you can imagine each of these scenarios would result in a different precision and recall (as well as TPR, FPR) rates.

ROC curve essentially finds out the TPR and FPR for various threshold values and plots TPR against the FPR

8- AUC

The **area under the curve** (AUC), is an aggregated measure of performance of a binary classifier on all possible threshold values (and therefore it is threshold invariant**)**.

AUC calculates the area under the ROC curve, and therefore it is between 0 and 1. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

On high-level, the higher the AUC of a model the better it is. But sometimes threshold independent measure is not what you want, e.g. you may care about your model recall and require that to be higher than 99% (while it has a reasonable precision or FPR). In that case, you may want to tune your model threshold such that it meets your minimum requirement on those metrics (and you may not care even if you model AUC is not too high).

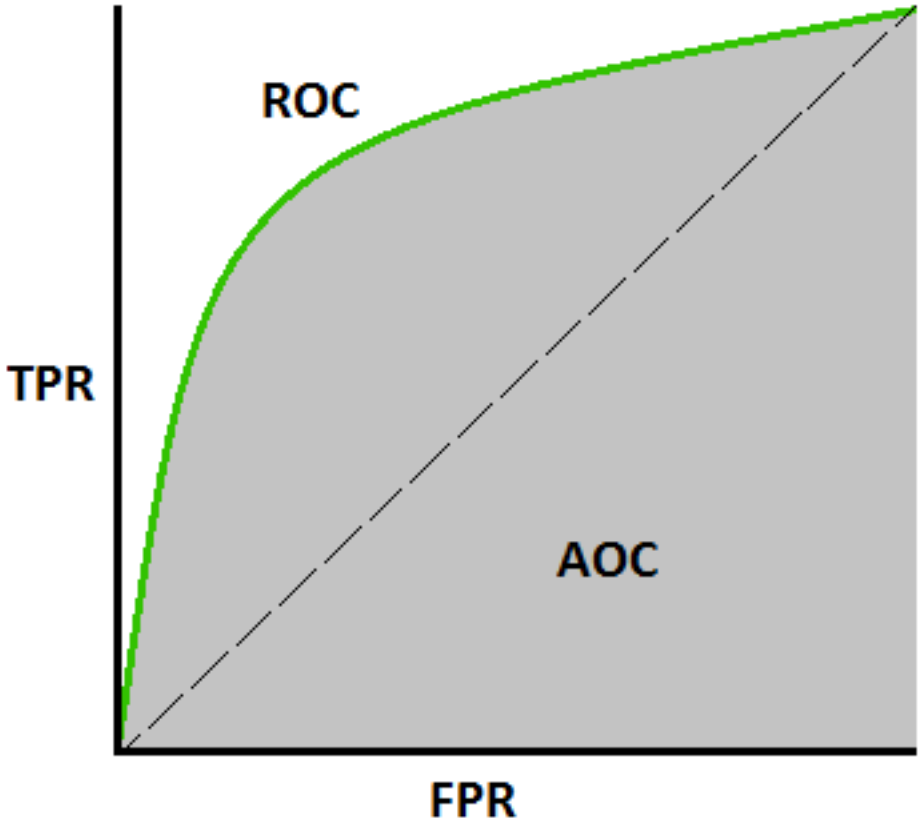
Therefore in order to decide how to evaluate your classification model performance, perhaps you want to have a good understanding of the business/problem requirement and the impact of low recall vs. low precision, and decide what metric to optimize for.

From a practical standpoint, a classification model which outputs probabilities is preferred over a single label output, as it provides the flexibility of tuning the threshold such that it meets your minimum recall/precision requirements. Not all models provide this nice probabilistic outputs though, e.g. SVM does not provide a simple probability as an output (although it provides margin which can be used to tune the decision, but it is not as straightforward and interpretable as having output probabilities).

What is the AUC - ROC Curve?

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.



AUC - ROC Curve [Image 2] (Image courtesy: My Photoshopped Collection)

Defining terms used in AUC and ROC Curve.

TPR (True Positive Rate) / Recall /Sensitivity

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

Image 3

Specificity

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Image 4

FPR

$$FPR = 1 - Specificity$$

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

Image 5

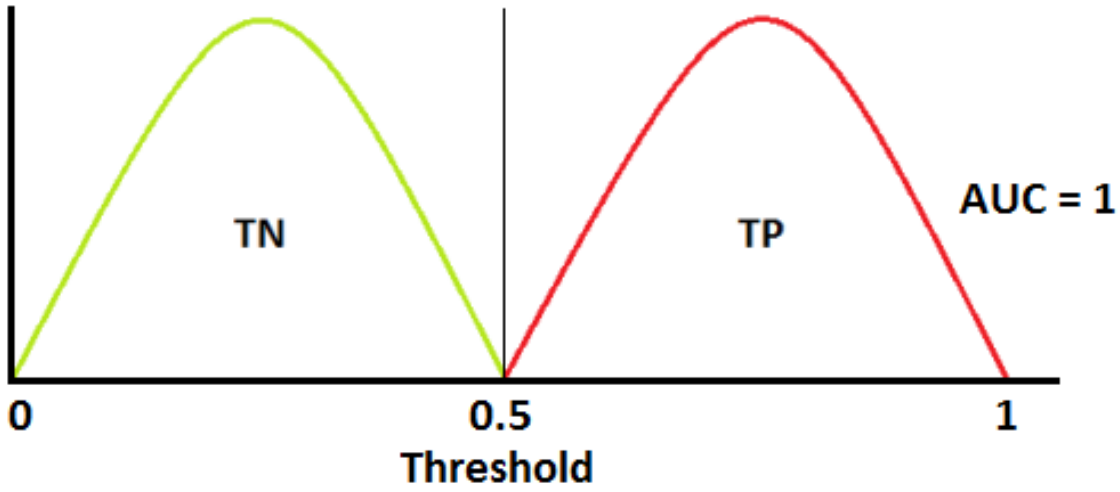
How to speculate about the performance of the model?

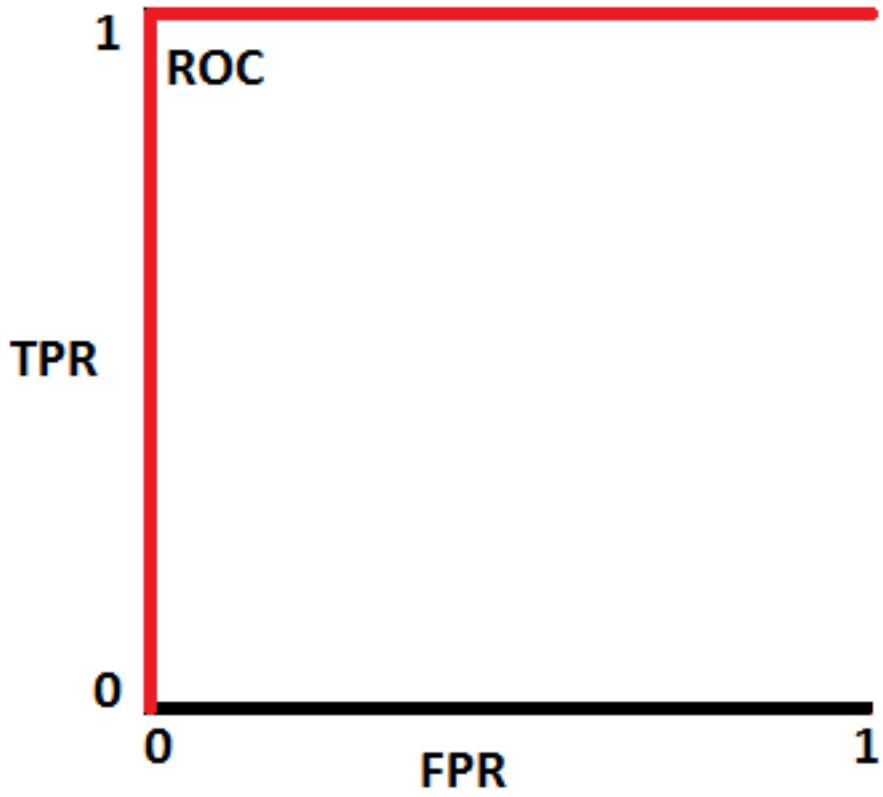
An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.

Let’s interpret the above statements.

As we know, ROC is a curve of probability. So let's plot the distributions of those probabilities:

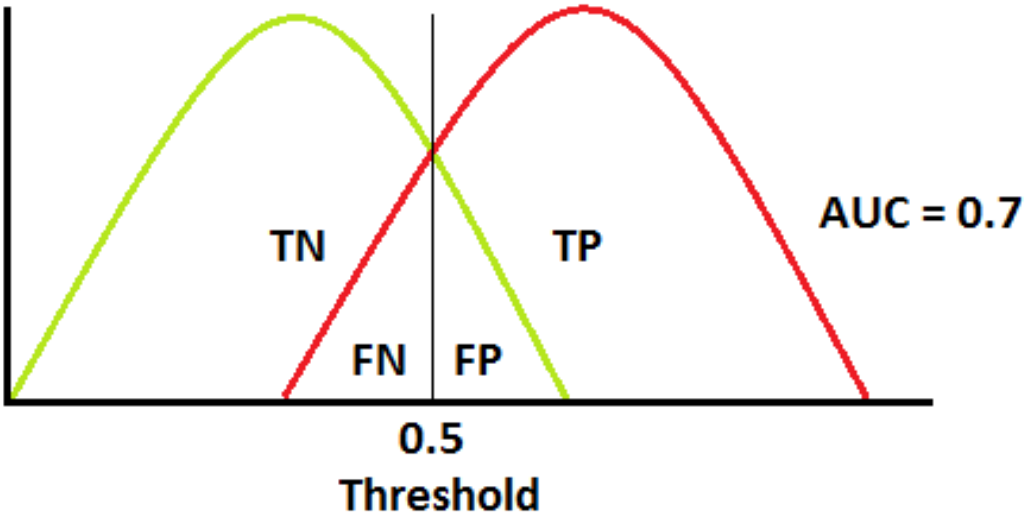
Note: Red distribution curve is of the positive class (patients with disease) and the green distribution curve is of the negative class(patients with no disease).

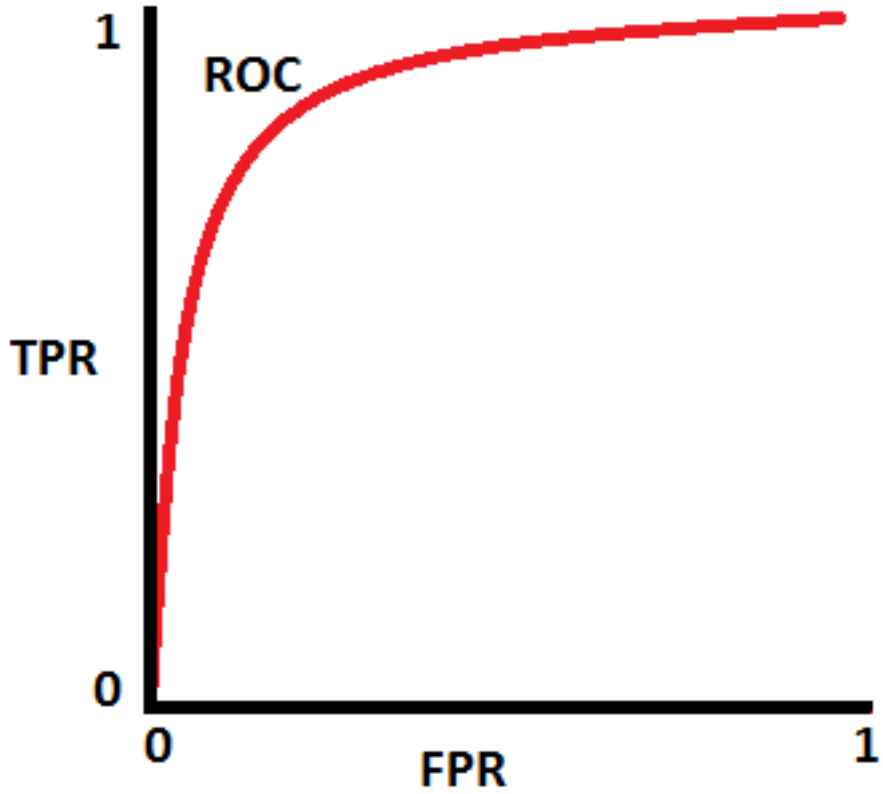




[Image 6 and 7] (Image courtesy: My Photoshopped Collection)

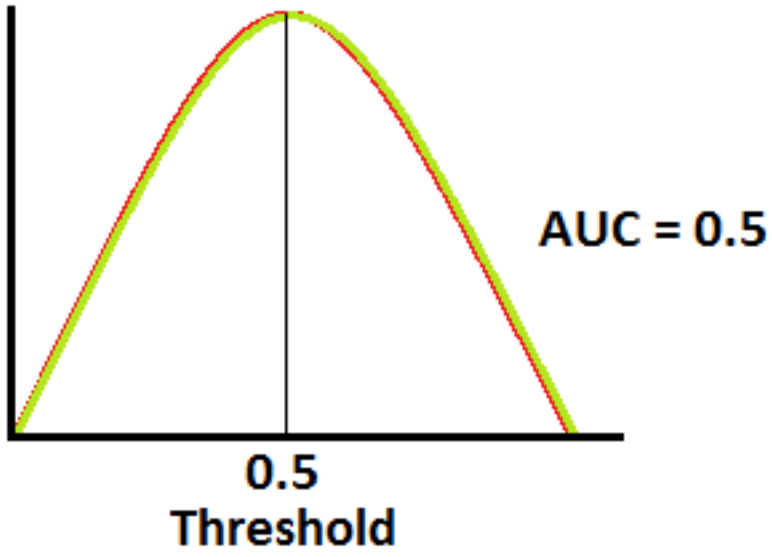
This is an ideal situation. When two curves don't overlap at all means model has an ideal measure of separability. It is perfectly able to distinguish between positive class and negative class.

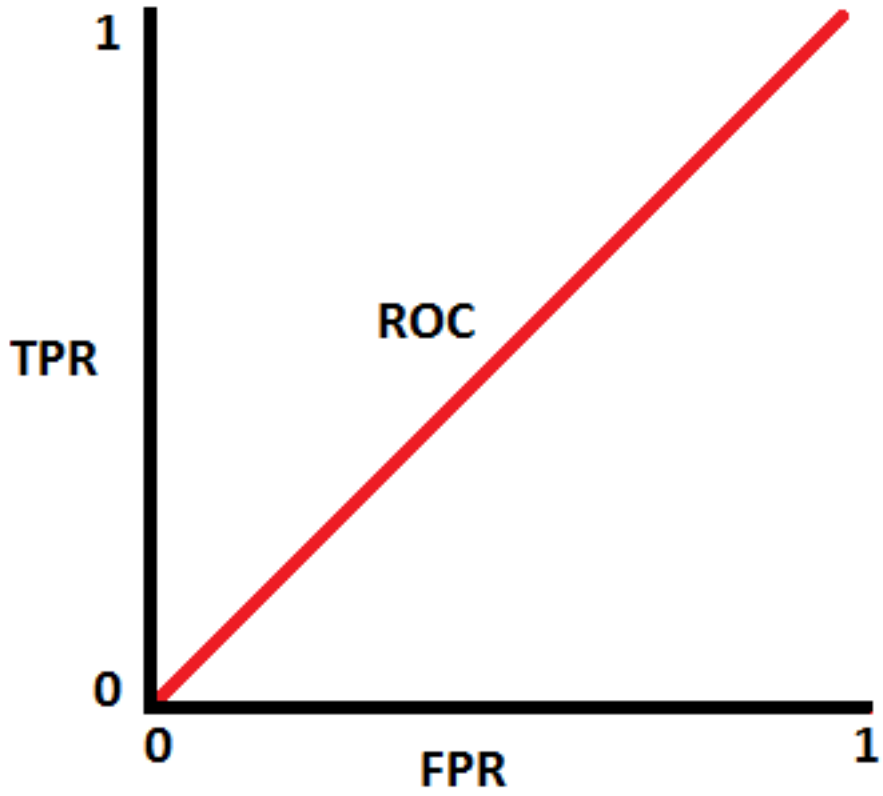




[Image 8 and 9] (Image courtesy: My Photoshopped Collection)

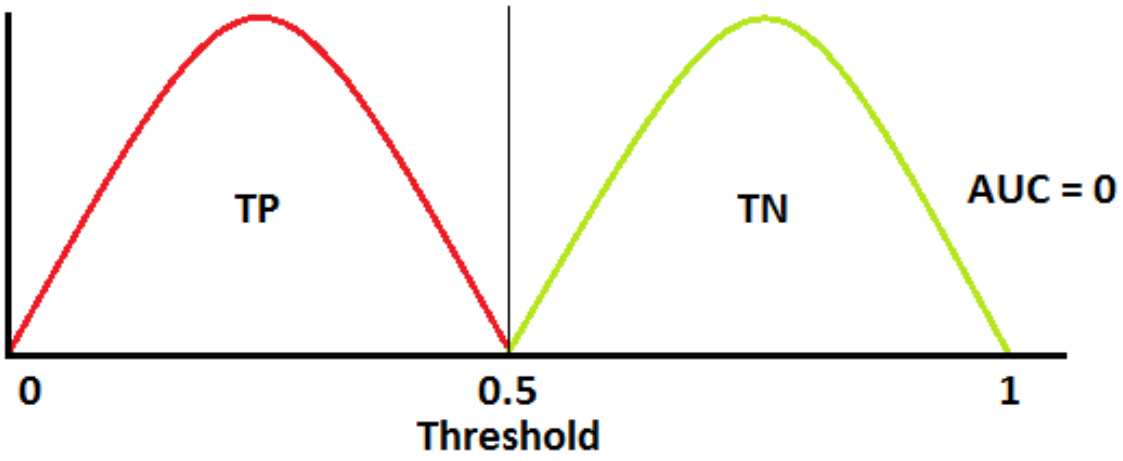
When two distributions overlap, we introduce type 1 and type 2 errors. Depending upon the threshold, we can minimize or maximize them. When AUC is 0.7, it means there is a 70% chance that the model will be able to distinguish between positive class and negative class.

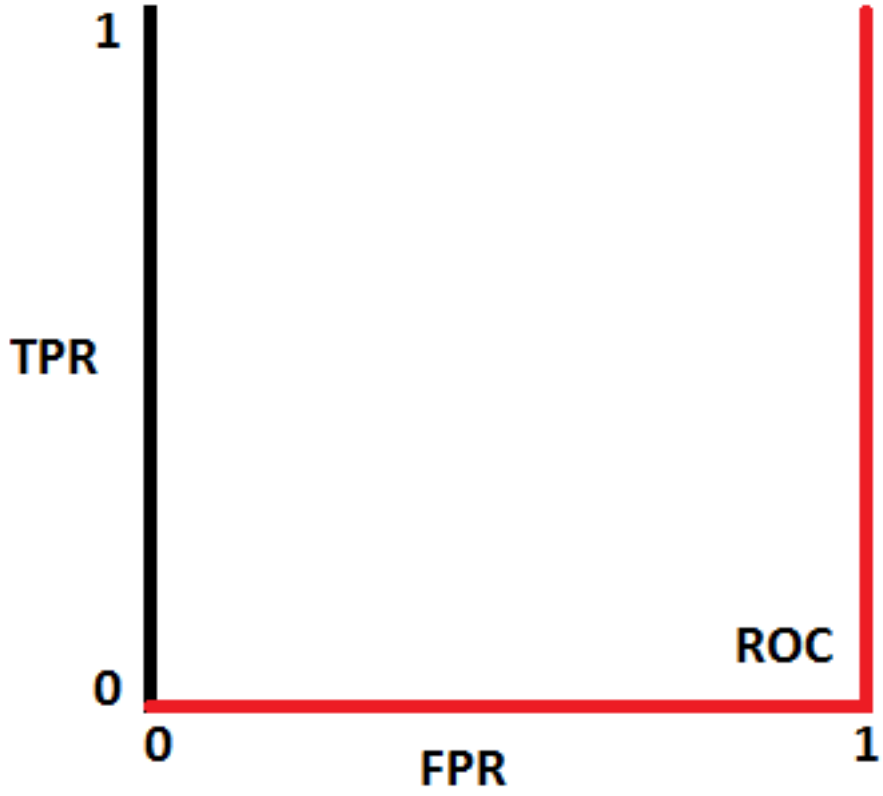




[Image 10 and 11] (Image courtesy: My Photoshopped Collection)

This is the worst situation. When AUC is approximately 0.5, the model has no discrimination capacity to distinguish between positive class and negative class.





[Image 12 and 13] (Image courtesy: My Photoshopped Collection)

When AUC is approximately 0, the model is actually reciprocating the classes. It means the model is predicting a negative class as a positive class and vice versa.

The relation between Sensitivity, Specificity, FPR, and Threshold.

Sensitivity and Specificity are inversely proportional to each other. So when we increase Sensitivity, Specificity decreases, and vice versa.

Sensitivity[↑], Specificity[↓] and Sensitivity[↓], Specificity[↑]

When we decrease the threshold, we get more positive values thus it increases the sensitivity and decreasing the specificity.

Similarly, when we increase the threshold, we get more negative values thus we get higher specificity and lower sensitivity.

As we know FPR is 1 - specificity. So when we increase TPR, FPR also increases and vice versa.

TPR[↑], FPR[↑] and TPR[↓], FPR[↓]

How to use the AUC ROC curve for the multi-class model?

In a multi-class model, we can plot the N number of AUC ROC Curves for N number classes using the One vs ALL methodology. So for example, If you have **three** classes named **X, Y,** and **Z**, you will have one ROC for X classified against Y and Z, another ROC for Y classified against X and Z, and the third one of Z classified against Y and X.

Regression

9- MSE

“Mean squared error” is perhaps the most popular metric used for regression problems. It essentially finds the average squared error between the predicted and actual values.

Let’s assume we have a regression model which predicts the price of houses in Seattle area (show them with $\hat{y}_i$), and let’s say for each house we also have the actual price the house was sold for (denoted with y_i). Then the MSE can be calculated as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Sometimes people use RMSE to have a metric with scale as the target values, which is essentially the square root of MSE.

Looking at house pricing prediction, RMSE essentially shows what is the average deviation in your model predicted house prices from the target values (the prices the houses are sold for).

Root Mean Squared Error

The Root Mean Squared Error, or RMSE, is an extension of the mean squared error. Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

For example, if your target variable has the units “*dollars*,” then the RMSE error score will also have the unit “*dollars*” and not “*squared dollars*” like the MSE.
As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.

The RMSE can be calculated as follows:

- RMSE = sqrt(1 / N * sum for i to N (y_i – yhat_i)^2)

Where *y_i* is the i'th expected value in the dataset, *yhat_i* is the i'th predicted value, and *sqrt()* is the square root function.
We can restate the RMSE in terms of the MSE as:

- RMSE = sqrt(MSE)

Note that the RMSE cannot be calculated as the average of the square root of the mean squared error values. This is a common error made by beginners and is an example of Jensen's inequality.
You may recall that the square root is the inverse of the square operation. MSE uses the square operation to remove the sign of each error value and to punish large errors. The square root reverses this operation, although it ensures that the result remains positive.

10- MAE

Mean absolute error (or mean absolute deviation) is another metric which finds the average absolute distance between the predicted and target values. MAE is define as below:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

MAE is known to be more robust to the outliers than MSE. The main reason being that in MSE by squaring the errors, the outliers (which usually have higher errors than other samples) get more attention and dominance in the final error and impacting the model parameters.

It is also worth mentioning that there is a nice maximum likelihood (MLE) interpretation behind MSE and MAE metrics. If we assume a linear dependence between features and targets, then MSE and MAE correspond to the MLE on the model parameters by assuming Gaussian and Laplace priors on the model errors respectively.