Course: CM-072

Class 3

List of topics

Metrics for Regression problems

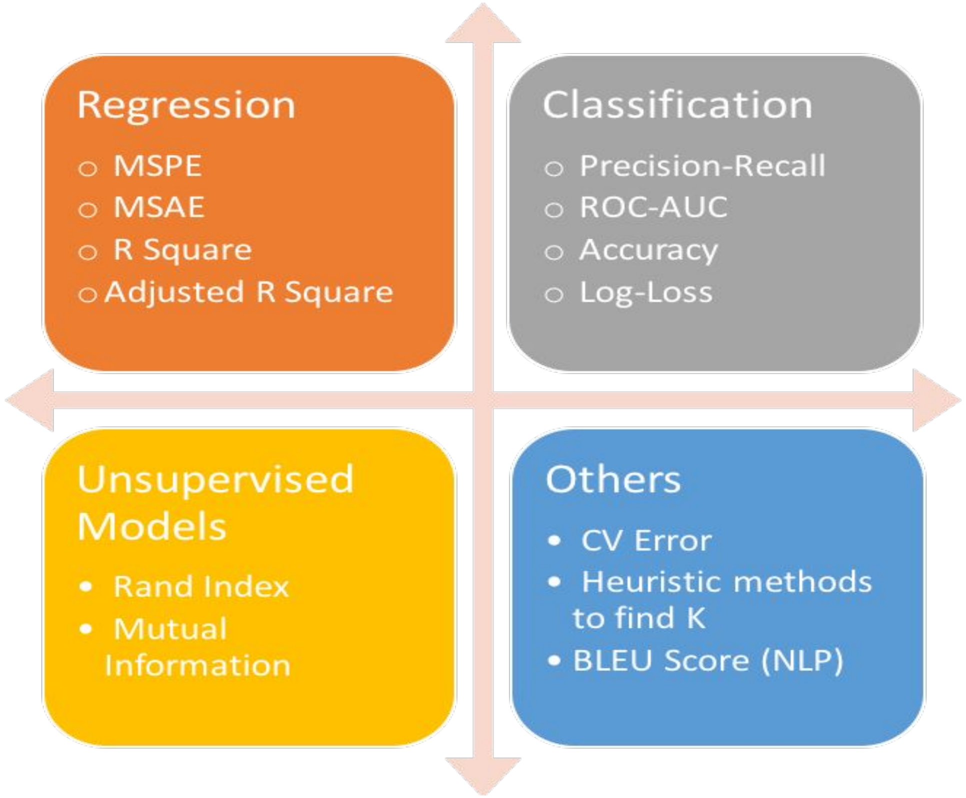
- RMSE
- MAE
- R^2
- Adjusted R Squared

Metrics for Classification problems

- Confusion Matrix
- Accuracy
- Precision
- Recall or Sensitivity
- Specificity
- F1 Score

Metrics for Multi-Classification problems

Most Useful Metrics



Metrics for Regression problems

Regression Metrics

- **RMSE (Root Mean Square Error)**

  It represents the sample standard deviation of the differences between predicted values and observed values (called **residuals**). Mathematically, it is calculated using this formula:

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

Regression Metrics

- **MAE**

  MAE is the average of the absolute difference between the predicted values and observed value. The MAE is a linear score which means that all the individual differences are weighted equally in the average. For example, the difference between 10 and 0 will be twice the difference between 5 and 0. However, same is not true for RMSE. Mathematically, it is calculated using this formula:

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

Regression Metrics

So which one should you choose and why?

Example:

Case 1: Actual Values = [2,4,6,8] , Predicted Values = [4,6,8,10]

Case 2: Actual Values = [2,4,6,8] , Predicted Values = [4,6,8,12]

MAE for case 1 = 2.0, RMSE for case 1 = 2.0

MAE for case 2 = 2.5, RMSE for case 2 = 2.65

From the above example, we can see that RMSE penalizes the last value prediction more heavily than MAE. Generally, RMSE will be higher than or equal to MAE. The only case where it equals MAE is when all the differences are equal or zero (true for case 1 where the difference between actual and predicted is 2 for all observations).

Regression Metrics

So which one should you choose and why?

… However, even after being more complex and biased towards higher deviation, RMSE is still the default metric of many models because loss function defined in terms of **RMSE is smoothly differentiable** and makes it easier to perform mathematical operations!.

However if you want a metric just to compare between two models from interpretation point of view,   MAE is a better choice.

It is important to note that the units of both RMSE & MAE are same as **y** values which is not true for R Square. The range of RMSE & MAE is from 0 to infinity.

Note: Minimizing the squared error over a set of numbers results in finding its mean, and minimizing the absolute error results in finding its median. This is the reason why MAE is robust to outliers whereas RMSE is not.

Regression Metrics

- **R Squared (R^2) and Adjusted R Squared**

  R Squared and Adjusted R Squared are often used for explanatory purposes and explains how well your selected independent variable(s) explain the variability in your dependent variable(s).

  Mathematically R^2 is given by:

$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

  The numerator is MSE (average of the squares of the residuals) and the denominator is the variance in Y values. Higher the MSE, smaller the R^2 and poorer is the model.

Regression Metrics

- **Adjusted R^2**

  Just like R^2 adjusted R^2 also shows how well terms fit a curve or line but adjusts for the number of terms in a model. It is given by below formula:

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

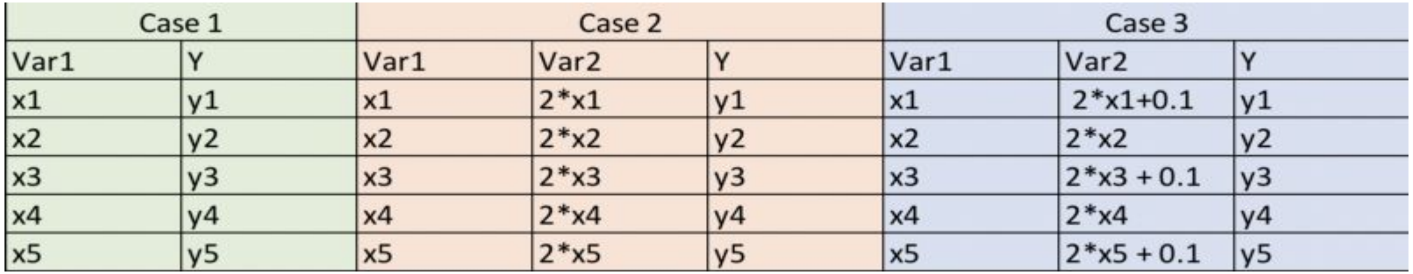
  where **n** is the total number of observations and **k** is the number of predictors. Adjusted R^2 will always be less than or equal to R^2.

Regression Metrics

So which one should you choose and why?

There are some problems with normal R^2 which are solved by Adjusted R^2. An adjusted R^2 will consider the marginal improvement added by an additional term in your model. So it will increase if you add the useful terms and it will decrease if you add less useful predictors. However, R^2 increases with increasing terms even though the model is not actually improving.

Example:



Case 1		Case 2			Case 3		
Var1	Y	Var1	Var2	Y	Var1	Var2	Y
x1	y1	x1	2*x1	y1	x1	2*x1+0.1	y1
x2	y2	x2	2*x2	y2	x2	2*x2	y2
x3	y3	x3	2*x3	y3	x3	2*x3 + 0.1	y3
x4	y4	x4	2*x4	y4	x4	2*x4	y4
x5	y5	x5	2*x5	y5	x5	2*x5 + 0.1	y5

Regression Metrics

So which one should you choose and why?

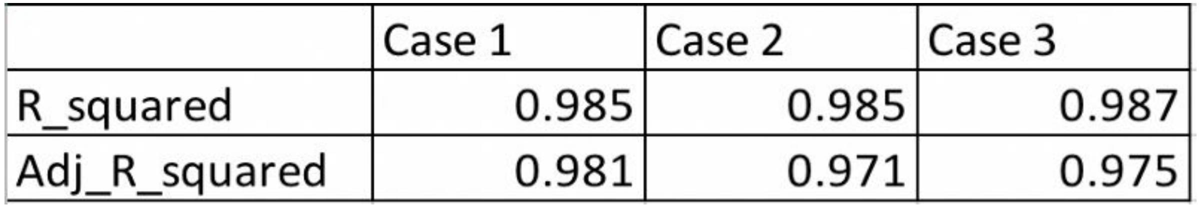
Here, Case 1 is the simple case where we have 5 observations of (x,y). In case 2, we have one more variable which is twice of variable 1 (perfectly correlated with var 1). In Case 3, we have produced a slight disturbance in var2 such that it is no longer perfectly correlated with var1.

So if we fit simple ordinary least squares (OLS) model for each case, logically we are not providing any extra or useful information to case 2 and case 3 with respect to case 1. So the metric value should not improve for these models. However, it is actually not true for R^2 which gives a higher value for model 2 and 3. But the adjusted R^2 takes care of this problem and it is actually decreasing for both cases 2 and 3.

Regression Metrics

So which one should you choose and why?

Predicted values will be same for both model 1 and model 2 and therefore, R^2 will also be same because it depends only on predicted and actual values.



	Case 1	Case 2	Case 3
R_squared	0.985	0.985	0.987
Adj_R_squared	0.981	0.971	0.975

From the above table, we can see that even though we are not adding any additional information from case 1 to case 2, still R^2 is increasing whereas adjusted R^2 is showing the correct trend (penalizing model 2 for more number of variables).

Regression Metrics

Comparison of Adjusted R^2 over RMSE

For the previous example, we will see that RMSE is same for case 1 and case 2 similar to R^2. This is the case where Adjusted R^2 does a better job than RMSE whose scope is limited to comparing predicted values with actual values. Also, the absolute value of RMSE does not actually tell how bad a model is. It can only be used to compare across two models whereas Adjusted R^2 easily does that. For example, if a model has adjusted R^2 equal to 0.05 then it is definitely poor.

However, if you care only about prediction accuracy then RMSE is best. It is computationally simple, easily differentiable and present as default metric for most of the models.

Metrics for Classification problems

Confusion matrix

The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for **Classification problem** where the output can be of two or more types of classes.

By example let's say we are solving a classification problem where we are predicting whether a person is having cancer or not.

Let's give a label of to our target variable:

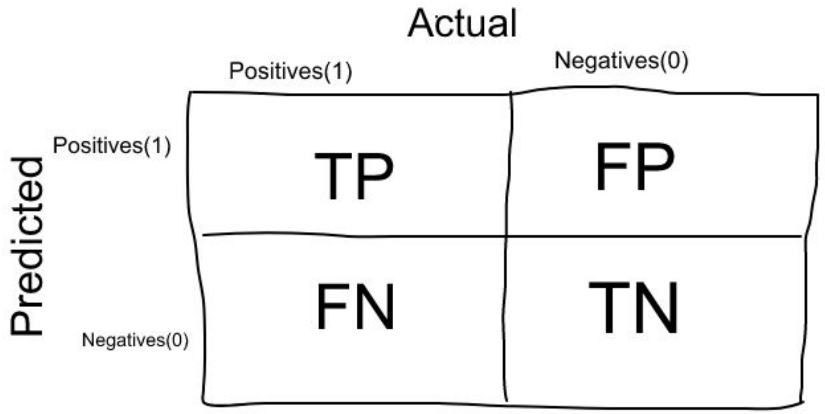
1: When a person is having cancer **0**: When a person is NOT having cancer.

We have identified the problem!.

Confusion matrix

The confusion matrix, is a table with two dimensions (“**Actual**” and “**Predicted**”), and sets of “classes” in both dimensions. Our Actual classifications are columns and Predicted ones are Rows.

The Confusion matrix in itself is not a performance measure as such, but almost all of the performance metrics are based on Confusion Matrix and the numbers inside it.



Terms associated with Confusion matrix

True Positives (TP): True Positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True).

Example: The case where a person is actually having cancer(1) and the model classifying his case as cancer(1) comes under True Positive.

True Negatives (TN): True Negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False).

Example: The case where a person NOT having cancer and the model classifying his case as Not cancer comes under True Negatives.

Terms associated with Confusion matrix

False Positives (FP): False Positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one (1).

Example: A person NOT having cancer and the model classifying his case as cancer comes under False Positives.

False Negatives (FN): False Negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one (0).

Example: A person having cancer and the model classifying his case as No-cancer comes under False Negatives.

When to minimise what?

We know that there will be some error associated with every model that we use for predicting the true class of the target variable. This will result in False Positives and False Negatives.

- **Minimising False Negatives:** Let’s say in our cancer detection problem example, out of 100 people, only 5 people have cancer. In this case, we want to correctly classify all the cancerous patients as even a very BAD model(predicting everyone as NON-cancerous) will give us a 95% **accuracy.** But, in order to capture all cancer cases, we might end up making a classification when the person actually NOT having cancer is classified as cancerous. This might be okay as it is less dangerous than NOT identifying a cancerous patient since we will anyway send the cancer cases for further examination and reports. But missing a cancer patient will be a huge mistake as no further examination will be done on them.

When to minimise what?

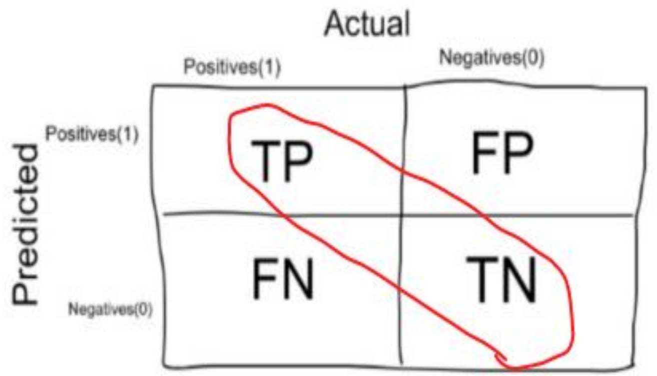
- **Minimising False Positives:**
  For better understanding of False Positives, let’s use a different example where the model classifies whether an email is spam or not.

  Let’s say that you are expecting an important email like hearing back from a recruiter or awaiting an admit letter from a university. Let’s assign a label to the target variable and say,**1:** ‘Email is a spam’ and **0:** ‘Email is not a spam’.

  Suppose the Model classifies that important email that you are desperately waiting for, as Spam(case of False Positive). Now, in this situation, this is pretty bad than classifying a spam email as important or not spam since in that case, we can still go ahead and manually delete it and it’s not a pain if it happens once a while. So in case of Spam email classification, minimising False Positives is more important than False Negatives.

Accuracy

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

In the numerator, are our correct predictions True Positives and True Negatives (marked as red in the figure) and in the denominator, are the kind of all predictions made by the algorithm(right as well as wrong ones).

Accuracy

When to use Accuracy:
Accuracy is a good measure when the target variable classes in the data are nearly balanced.

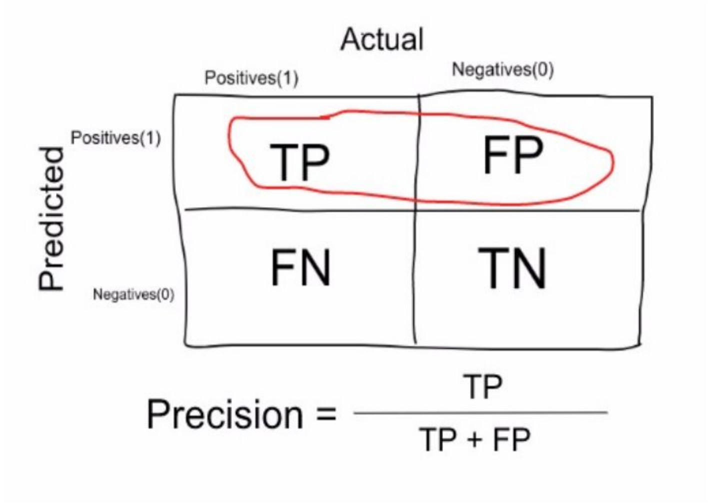
Example:60% classes in our fruits images data are apple and 40% are oranges.
A model which predicts whether a new image is apple or an orange, 97% of times correctly is a very good measure in this example.

When NOT to use Accuracy:
Accuracy should NEVER be used as a measure when the target variable classes in the data are a majority of one class.

Example: In our cancer detection example with 100 people, only 5 people has cancer. Let’s say our model is very bad and predicts every case as No cancer. In doing so, it has classified those 95 non-cancer patients correctly and 5 cancerous patients as Non-cancerous. Now even though the model is terrible at predicting cancer, The accuracy of such a bad model is also 95%.

Precision

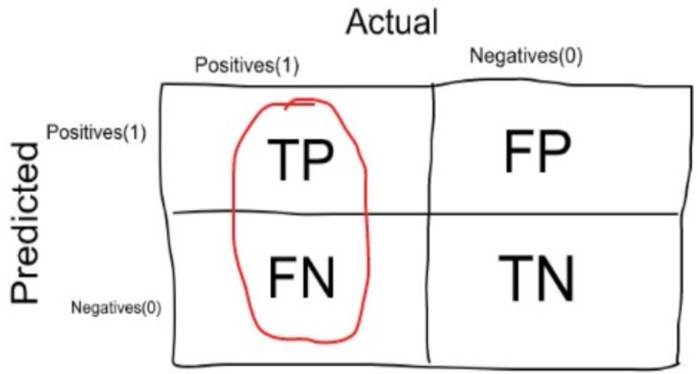
Precision is a measure that tells us what proportion of patients that we diagnosed as having cancer, actually had cancer. The predicted positives (People predicted as cancerous are TP and FP) and the people actually having a cancer are TP.



Example: In our cancer example with 100 people, only 5 people have cancer. Let's say our model is very bad and predicts every case as cancer. Since we are predicting everyone as having cancer, our denominator(True Positives and False Positives) is 100 and the numerator, person having cancer and the model predicting his case as cancer is 5. So in this example, we can say that **precision** of such model is 5%.

Recall or Sensitivity

Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having cancer. The actual positives (people having cancer are TP and FN) and the people diagnosed by the model having a cancer are TP. (FN is included because the person actually had a cancer even though the model predicted otherwise).



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

Example: In our cancer example with 100 people, 5 people actually have cancer. Let's say that the model predicts every case as cancer.
So our denominator(True Positives and False Negatives) is 5 and the numerator, person having cancer and the model predicting his case as cancer is also 5 (Since we predicted 5 cancer cases correctly).
So in this example, we can say that the **Recall** of such model is 100%. And Precision of such a model is 5%.

When to use Precision and when to use Recall?

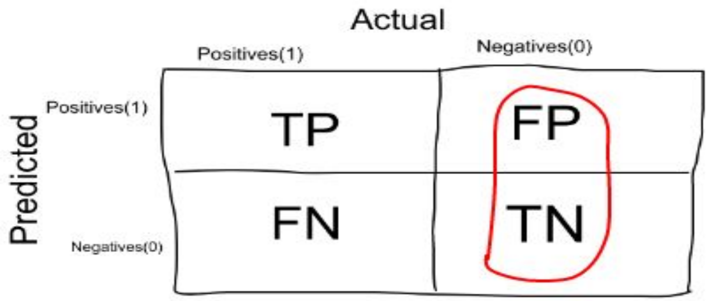
It is clear that **recall** gives us information about a classifier's performance with respect to False Negatives (how many did we miss), while **precision** gives us information about its performance with respect to False Positives(how many did we caught).

- **Precision** is about being precise. So even if we managed to capture only one cancer case, and we captured it correctly, then we are 100% precise.
- **Recall** is not so much about capturing cases correctly but more about capturing all cases that have "cancer" with the answer as "cancer". So if we simply always say every case as "cancer", we have 100% recall.

So basically if we want to focus more on minimising False Negatives, we would want our Recall to be as close to 100% as possible without precision being too bad and if we want to focus on minimising False positives, then our focus should be to make Precision as close to 100% as possible.

Specificity

Specificity is a measure that tells us what proportion of patients that did NOT have cancer, were predicted by the model as non-cancerous. The actual negatives (people actually NOT having cancer are FP and TN) and the people diagnosed by us not having cancer are TN. (FP is included because the person did NOT actually have cancer even though the model predicted otherwise). Specificity is the exact opposite of Recall.



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

Example: In our cancer example with 100 people, 5 people actually have cancer. Let's say that the model predicts every case as cancer.
So our denominator(False positives and True Negatives) is 95 and the numerator, person not having cancer and the model predicting his case as no cancer is 0 (Since we predicted every case as cancer). So in this example, we can that that Specificity of such model is 0%.

F1 Score

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both False Positives and False Negatives into account. F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Is often used in the field of information retrieval for measuring search, document classification, and query classification performance.

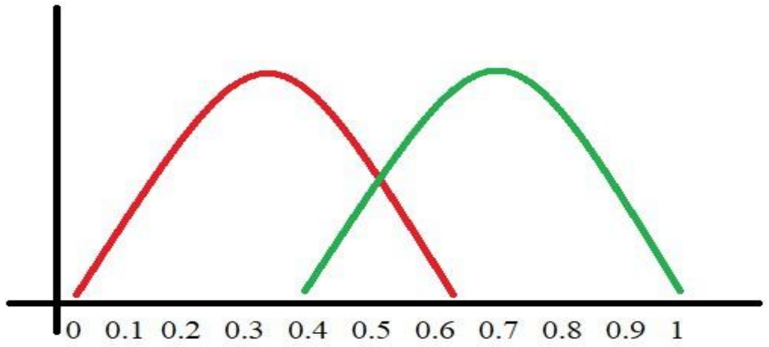
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

F1 score is the harmonic mean of precision and recall.

What is ROC?

ROC (Receiver Operating Characteristic) Curve tells us about how good the model can distinguish between two things (e.g If a patient has a disease or no). Better models can accurately distinguish between the two. Whereas, a poor model will have difficulties in distinguishing between the two.

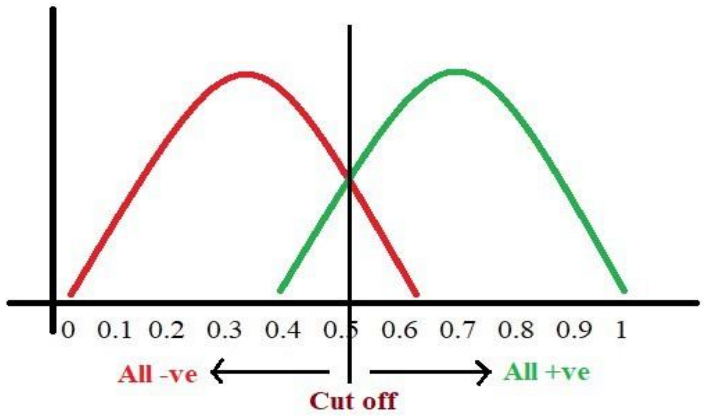
Let's assume we have a model which predicts whether the patient has a particular disease or no. The model predicts probabilities for each patient (in python we use the **predict_proba** function). Using these probabilities, we plot the distribution as shown below:



Here, the red distribution represents all the patients who do not have the disease and the green distribution represents all the patients who have the disease.

ROC Curve

Now we got to pick a value where we need to set the cut off i.e. a **threshold value**, above which we will predict everyone as positive (they have the disease) and below which will predict as negative (they do not have the disease). We will set the threshold at **0.5** as shown below:

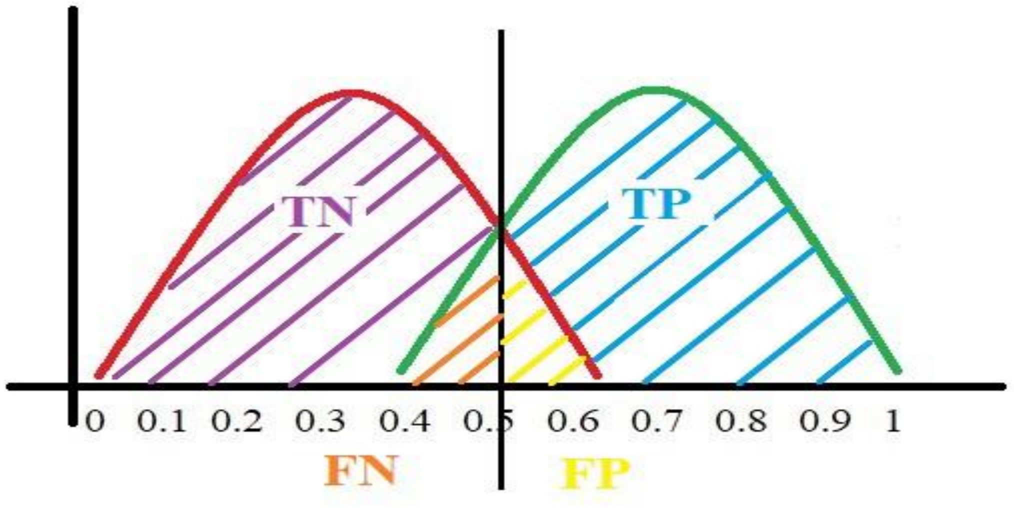


All the positive values above the threshold will be **True Positives** and the negative values above the threshold will be **False Positives** as they are predicted incorrectly as positives.

All the negative values below the threshold will be **True Negatives** and the positive values below the threshold will be **False Negative** as they are predicted incorrectly as negatives.

ROC Curve

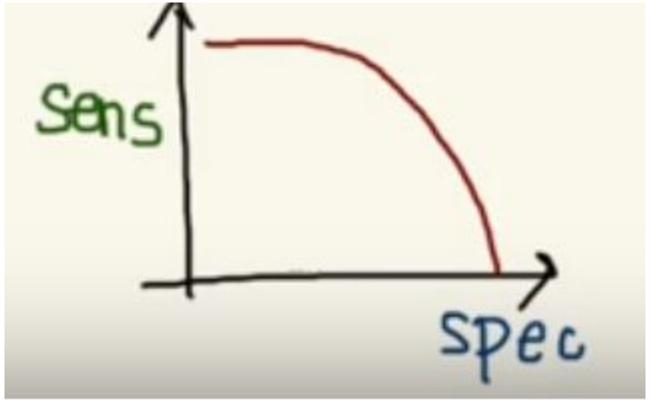
Here, we have got a basic idea of the model predicting correct and incorrect values with respect to the threshold set.



Trade-off between Sensitivity and Specificity

When we decrease the threshold, we get more positive values thus increasing the sensitivity. Meanwhile, this will decrease the specificity.

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

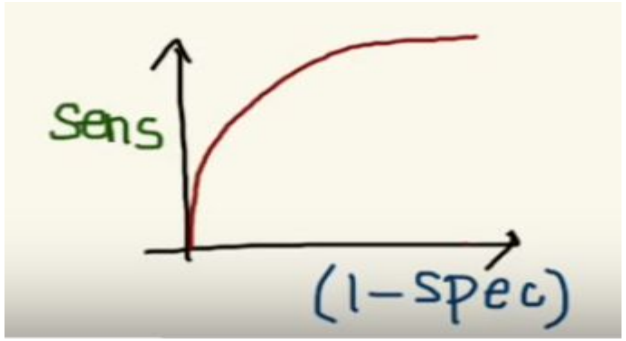


Trade off between Sensitivity & Specificity

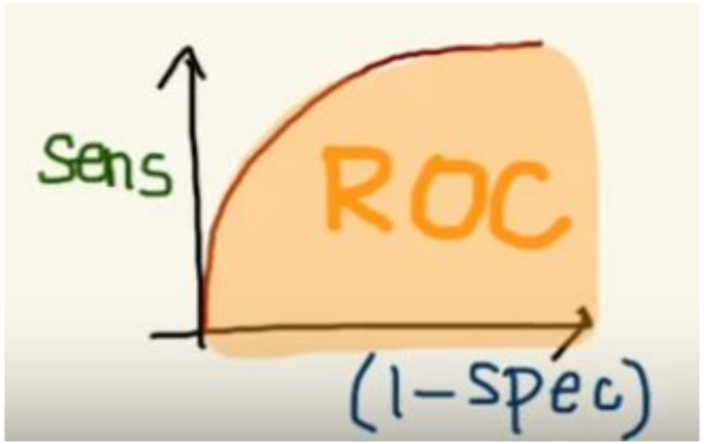
But, this is not how we graph the ROC curve!.

ROC Curve

To plot ROC curve, instead of Specificity we use (1 — Specificity) and the graph will look something like this:

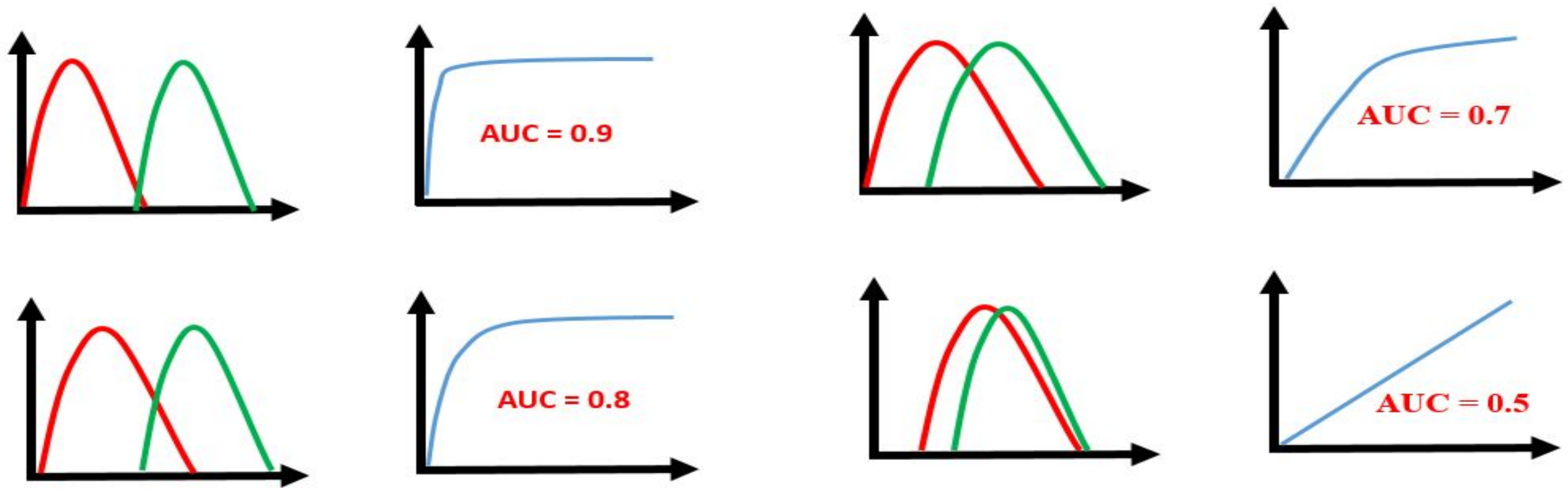


So now, when the sensitivity increases, (1 — specificity) will also increase. This curve is known as the ROC curve.



Area Under the Curve

The AUC is the area under the ROC curve. This score gives us a good idea of how well the model performances.



The first model does quite a good job of distinguishing the positive and the negative values. Therefore, there the AUC score is 0.9 as the area under the ROC curve is large.
Whereas, if we see the last model, predictions are completely overlapping each other and we get the AUC score of 0.5. This means that the model is performing poorly and it is predictions are almost random.

Why do we use (1-Specificity)?

Let’s derive what exactly is (1 — Specificity):

Specificity gives us the True Negative Rate and **(1 — Specificity)** gives us the False Positive Rate. So the sensitivity can be called as the **True Positive Rate** and (1 — Specificity) can be called as the **False Positive Rate**.

So now we are just looking at the positives. As we increase the threshold, we decrease the TPR as well as the FPR and when we decrease the threshold, we are increasing the TPR and FPR.

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

$$1 - Specificity = 1 - \frac{TN}{TN + FP}$$

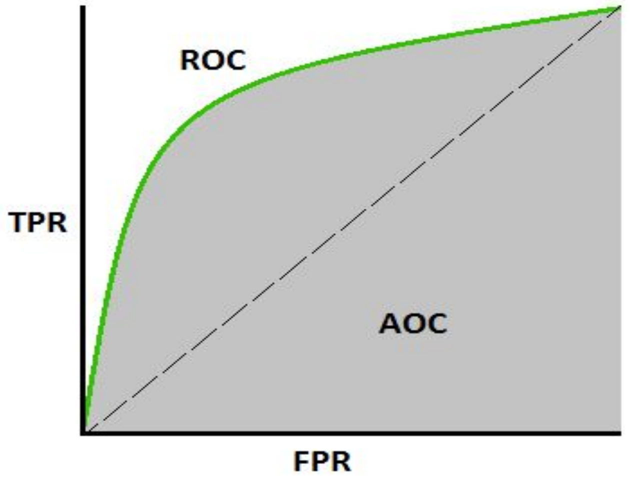
$$1 - Specificity = \frac{TN + FP - TN}{TN + FP}$$

$$1 - Specificity = \frac{FP}{TN + FP}$$

Thus, AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes.

AUC-ROC Curve (general)

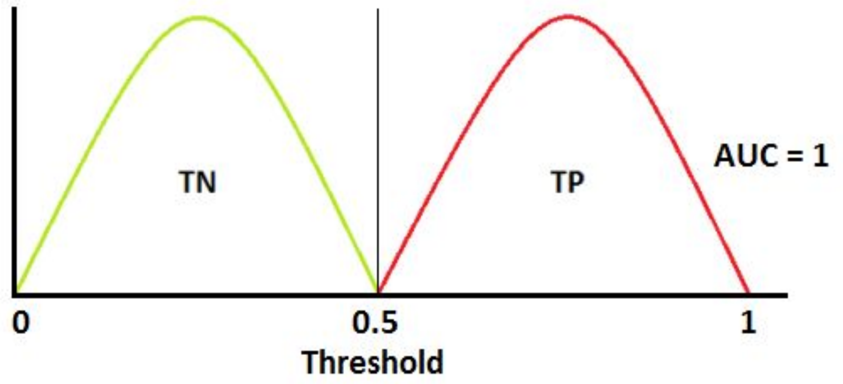
AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting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

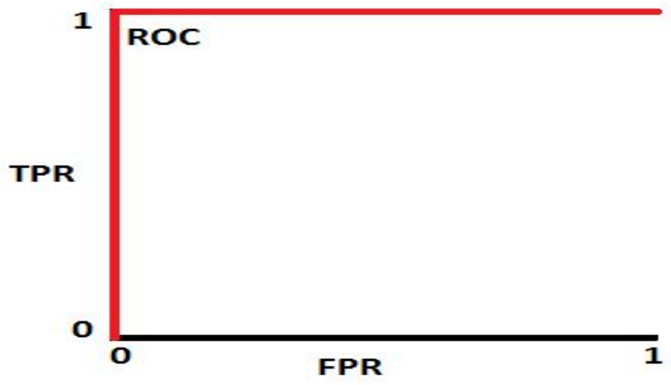


The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.

How to speculate the performance of the model?

An excellent model has AUC near to the 1 which means it has good measure of separability. A poor model has AUC near to the 0 which means it has 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 model has no class separation capacity whatsoever.

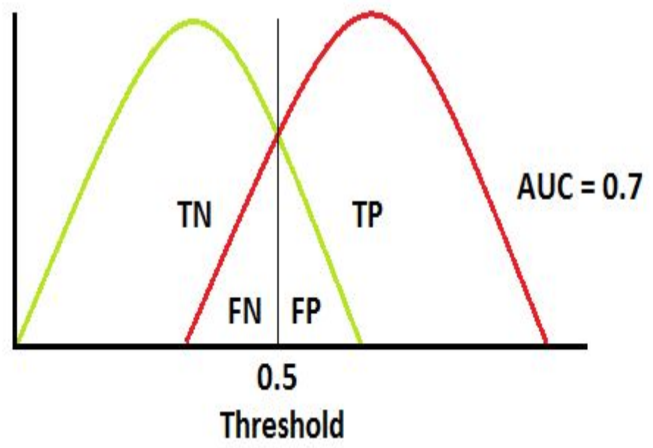


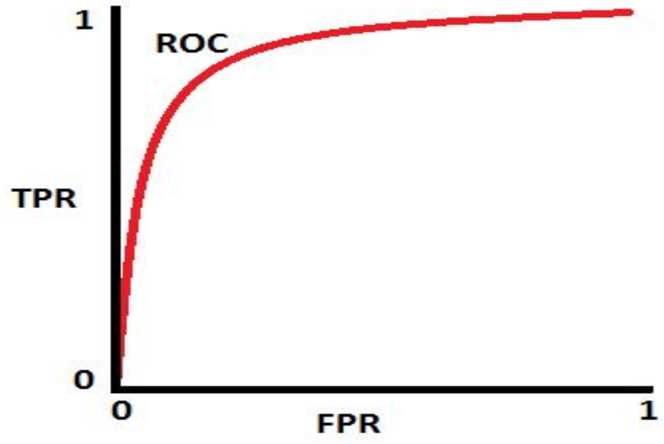


Note: Red distribution curve is of the positive class (patients with disease) and green distribution curve is of negative class(patients with no disease).

How to speculate the performance of the model?

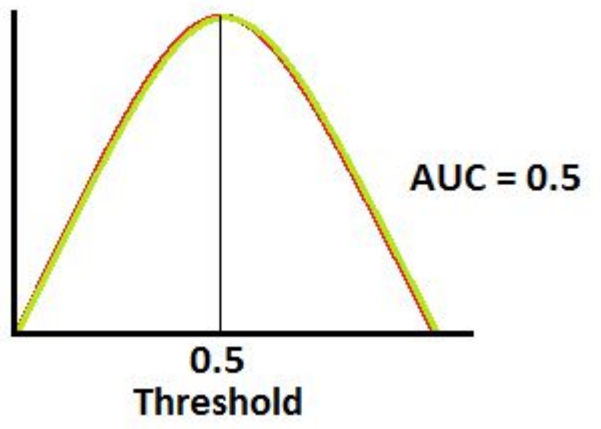
When two curves don’t overlap at all means model has an ideal measure of separability. It is perfectly able to distinguish between positive class and negative class.

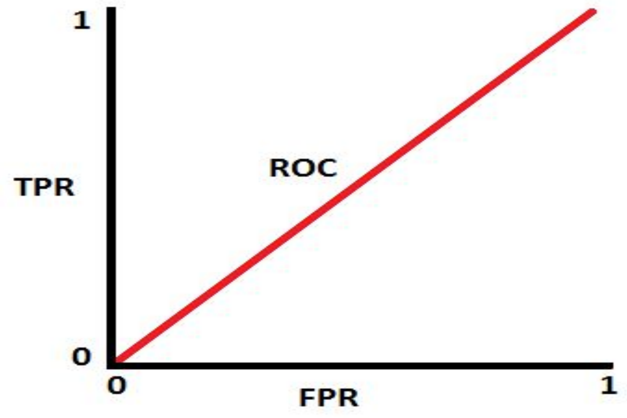




How to speculate the performance of the model?

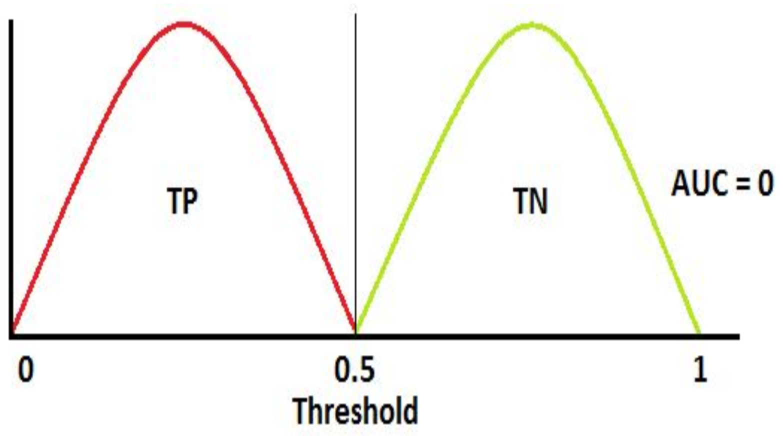
When two distributions overlap, we introduce type 1 and type 2 error (False Positive, False Negative). Depending upon the threshold, we can minimize or maximize them. When AUC is 0.7, it means there is 70% chance that model will be able to distinguish between positive class and negative class.

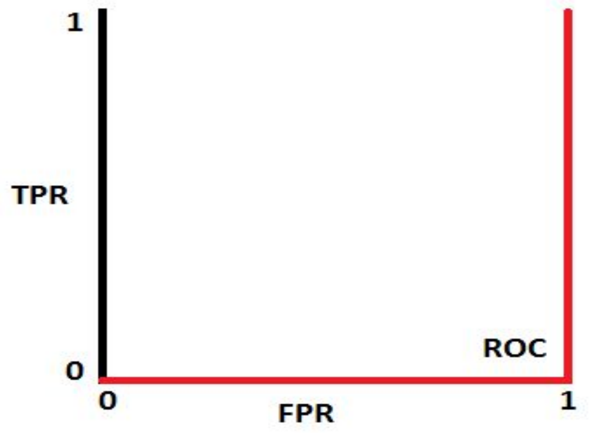




How to speculate the performance of the model?

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





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

Log-Loss

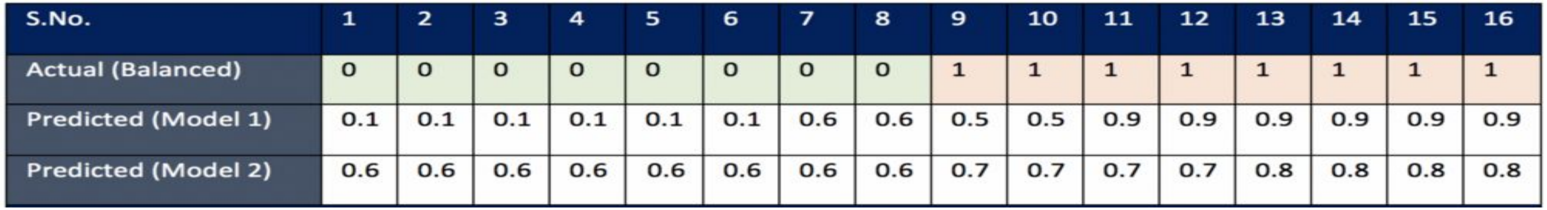
Log-loss is a measurement of accuracy that incorporates the idea of probabilistic confidence given by following expression for binary class:

$$-(y\log(p) + (1-y)\log(1-p))$$

It takes into account the uncertainty of your prediction based on how much it varies from the actual label. In the worst case, let's say you predicted 0.5 for all the observations. So log-loss will become -log(0.5) = 0.69. Hence, we can say that anything above 0.6 is a very poor model considering the actual probabilities.

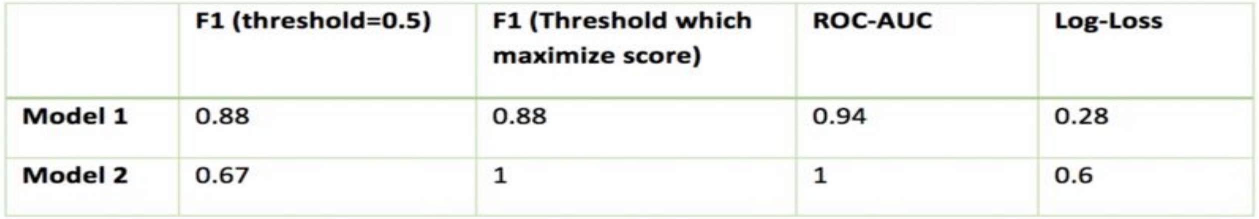
Comparison of Log-loss with ROC and F1

Consider balanced data, it looks like model 1 is doing a better job in predicting the absolute probabilities whereas model 2 is working best in ranking observations according to their true labels.



S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Actual (Balanced)	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Predicted (Model 1)	0.1	0.1	0.1	0.1	0.1	0.1	0.6	0.6	0.5	0.5	0.9	0.9	0.9	0.9	0.9	0.9
Predicted (Model 2)	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.8	0.8	0.8	0.8

Let’s verify with the actual score:



	F1 (threshold=0.5)	F1 (Threshold which maximize score)	ROC-AUC	Log-Loss
Model 1	0.88	0.88	0.94	0.28
Model 2	0.67	1	1	0.6

Comparison of Log-loss with ROC and F1

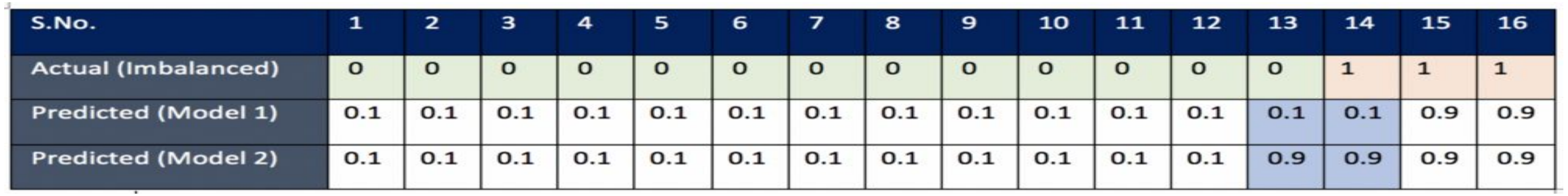
If you consider log-loss, model 2 is worst giving a high value of log-loss because the absolute probabilities have big difference from actual labels. But this is in complete disagreement with F1 and AUC score, according to which model 2 has 100% accuracy. Also, you would like to note that with different thresholds, F1 score is changing, and preferring model 1 over model 2 for default threshold of 0.5.

Inferences drawn from the above example:

- If you care for absolute probabilistic difference, go with log-loss.
- If you care only for the final class prediction and you don’t want to tune threshold, go with AUC score.
- F1 score is sensitive to threshold and you would want to tune it first before comparing the models.

How each of them deals with class imbalance?

The only difference in the two models is their prediction for observation 13 and 14. Model 1 is doing a better job in classifying observation 13 (label 0) whereas Model 2 is doing better in classifying observation 14 (label 1).

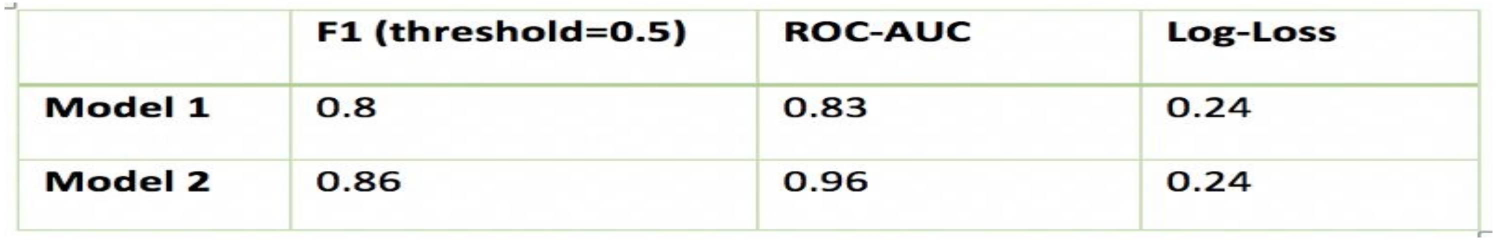


S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Actual (Imbalanced)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Predicted (Model 1)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.9	0.9
Predicted (Model 2)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.9	0.9	0.9	0.9

The goal is to see which model actually captures the difference in classifying the imbalanced class better (class with few observations, here it is label 1).

How each of them deals with class imbalance?

In problems like fraud detection/spam mail detection, where positive labels are few, we would like our model to predict positive classes correctly and hence we will sometime prefer those model who are able to classify these positive labels.

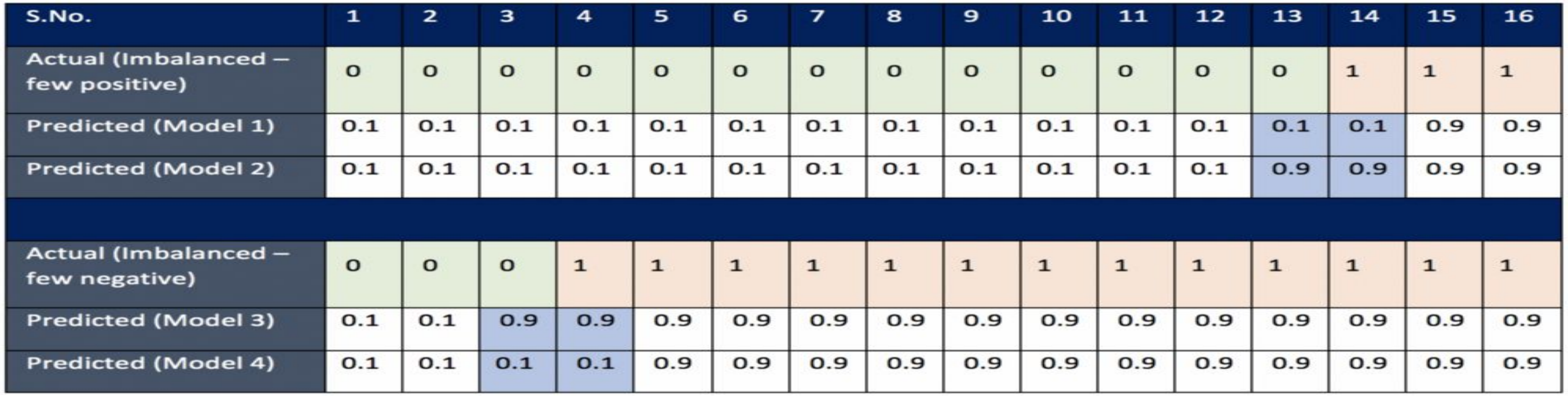


	F1 (threshold=0.5)	ROC-AUC	Log-Loss
Model 1	0.8	0.83	0.24
Model 2	0.86	0.96	0.24

Clearly log-loss is failing in this case because according to log-loss both the models are performing equally. This is because log-loss function is symmetric and does not differentiate between classes .

How each of them deals with class imbalance?

Both F1 score and ROC-AUC score is doing better in preferring model 2 over model 1. So we can use both these methods for class imbalance. But we will have to dig further to see how differently they treat class imbalance.

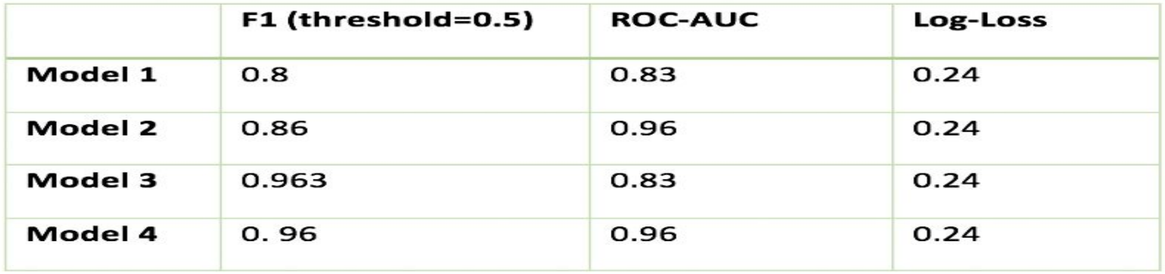


S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Actual (Imbalanced – few positive)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Predicted (Model 1)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.9	0.9
Predicted (Model 2)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.9	0.9	0.9	0.9
Actual (Imbalanced – few negative)	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Predicted (Model 3)	0.1	0.1	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Predicted (Model 4)	0.1	0.1	0.1	0.1	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

In the previous example, we saw that there were few positive labels. In the second example, there were few negative labels.

How each of them deals with class imbalance?

Let's see how F1 score and ROC-AUC differentiate between these two cases.



	F1 (threshold=0.5)	ROC-AUC	Log-Loss
Model 1	0.8	0.83	0.24
Model 2	0.86	0.96	0.24
Model 3	0.963	0.83	0.24
Model 4	0. 96	0.96	0.24

ROC-AUC score handled the case of few negative labels in the same way as it handled the case of few positive labels. An interesting thing to note here is that F1 score is pretty much same for both Model 3 & Model 4 because positive labels are large in number and it cares only for the misclassification of positive labels.

Inferences drawn from above example:

- If you care for a class which is smaller in number independent of the fact whether it is positive or negative, go for ROC-AUC score.

Which metric should you use for multi-classification

We have further three types of non-binary classification:

- **Multi-Class:** classification task with more than two classes such that the input is to be classified into one, and only one of these classes. Example: classify a set of images of fruits into any one of these categories — apples, bananas, and oranges.
- **Multi-labels:** classifying a sample into a set of target labels. Example: tagging a blog into one or more topics like technology, religion, politics etc. Labels are isolated and their relations are not considered important.
- **Hierarchical:** each category can be grouped together with similar categories, creating meta-classes, which in turn can be grouped again until we reach the root level (set containing all data). Examples include text classification and species classification.

We will cover only the first category.

Metric for multiclass-classification

From of article of Marina Sokolova and Guy Lapalme: **A systematic analysis of performance measures for classification tasks:** http://atour.iro.umontreal.ca/rali/sites/default/files/publis/SokolovaLapalme-JIPM09.pdf.

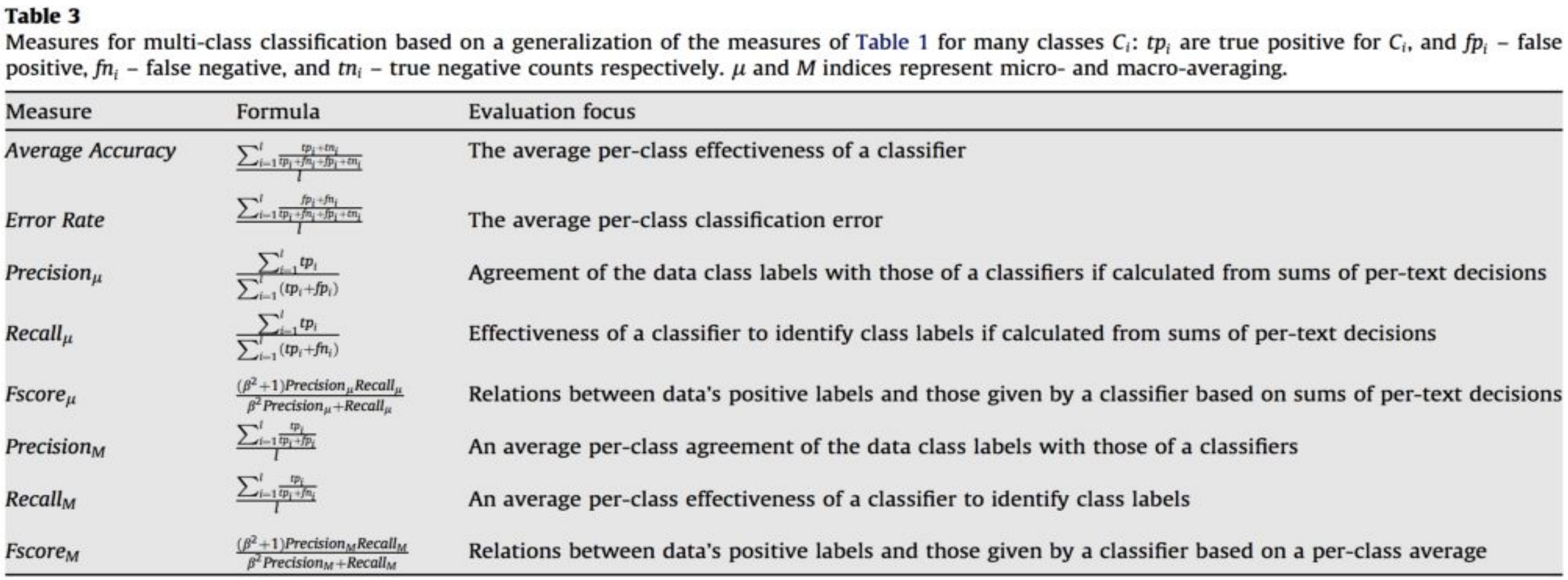


Table 3
Measures for multi-class classification based on a generalization of the measures of Table 1 for many classes C_i: tp_i are true positive for C_i, and fp_i – false positive, fn_i – false negative, and tn_i – true negative counts respectively. μ and M indices represent micro- and macro-averaging.

Measure	Formula	Evaluation focus
Average Accuracy	$\frac{\sum_{i=1}^{l}\frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{l}$	The average per-class effectiveness of a classifier
Error Rate	$\frac{\sum_{i=1}^{l}\frac{fp_i+fn_i}{tp_i+fn_i+fp_i+tn_i}}{l}$	The average per-class classification error
Precision$_\mu$	$\frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i+fp_i)}$	Agreement of the data class labels with those of a classifiers if calculated from sums of per-text decisions
Recall$_\mu$	$\frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i+fn_i)}$	Effectiveness of a classifier to identify class labels if calculated from sums of per-text decisions
Fscore$_\mu$	$\frac{(\beta^2+1)Precision_\mu Recall_\mu}{\beta^2 Precision_\mu+Recall_\mu}$	Relations between data's positive labels and those given by a classifier based on sums of per-text decisions
Precision$_M$	$\frac{\sum_{i=1}^{l}\frac{tp_i}{tp_i+fp_i}}{l}$	An average per-class agreement of the data class labels with those of a classifiers
Recall$_M$	$\frac{\sum_{i=1}^{l}\frac{tp_i}{tp_i+fn_i}}{l}$	An average per-class effectiveness of a classifier to identify class labels
Fscore$_M$	$\frac{(\beta^2+1)Precision_M Recall_M}{\beta^2 Precision_M+Recall_M}$	Relations between data's positive labels and those given by a classifier based on a per-class average

Metric for multiclass-Classification

As you can see in the above table, we have broadly two types of metrics- **micro-average** and **macro-average**. Most commonly used metrics for multi-classes are F1 score, Average Accuracy, Log-loss. There is yet no well-developed ROC-AUC score for multi-class.
Log-loss for multi-class is defined as:

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

Where,

N	*No of Rows in Test set*
M	*No of Fault Delivery Classes*
Y_{ij}	*1 if observation belongs to Class j; else 0*
P_{ij}	*Predicted Probability that observation belong to Class j*

- In Micro-average method, you sum up the individual True Positives, False Positives, and False Negatives of the system for different sets and then apply them to get the statistics.
- In Macro-average, you take the average of the precision and recall of the system on different sets.

Micro-average is preferable if there is a class imbalance problem.

https://sebastianraschka.com/faq/docs/multiclass-metric.html

These notes are based on the articles of the following data science professionals:

- **Alvira Swalin: https://medium.com/@aswalin.**
- **Jocelyn D'Souza: https://medium.com/@djocz.**
- **Mohammed Sunasra: https://medium.com/@MohammedS.**
- **How to handle Imbalanced Classification Problems in machine learning?:https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/.**