

		Predicted classes	
		Negative 0	Positive 1
Actual classes	Negative 0	TN	FP
	Positive 1	FN	TP

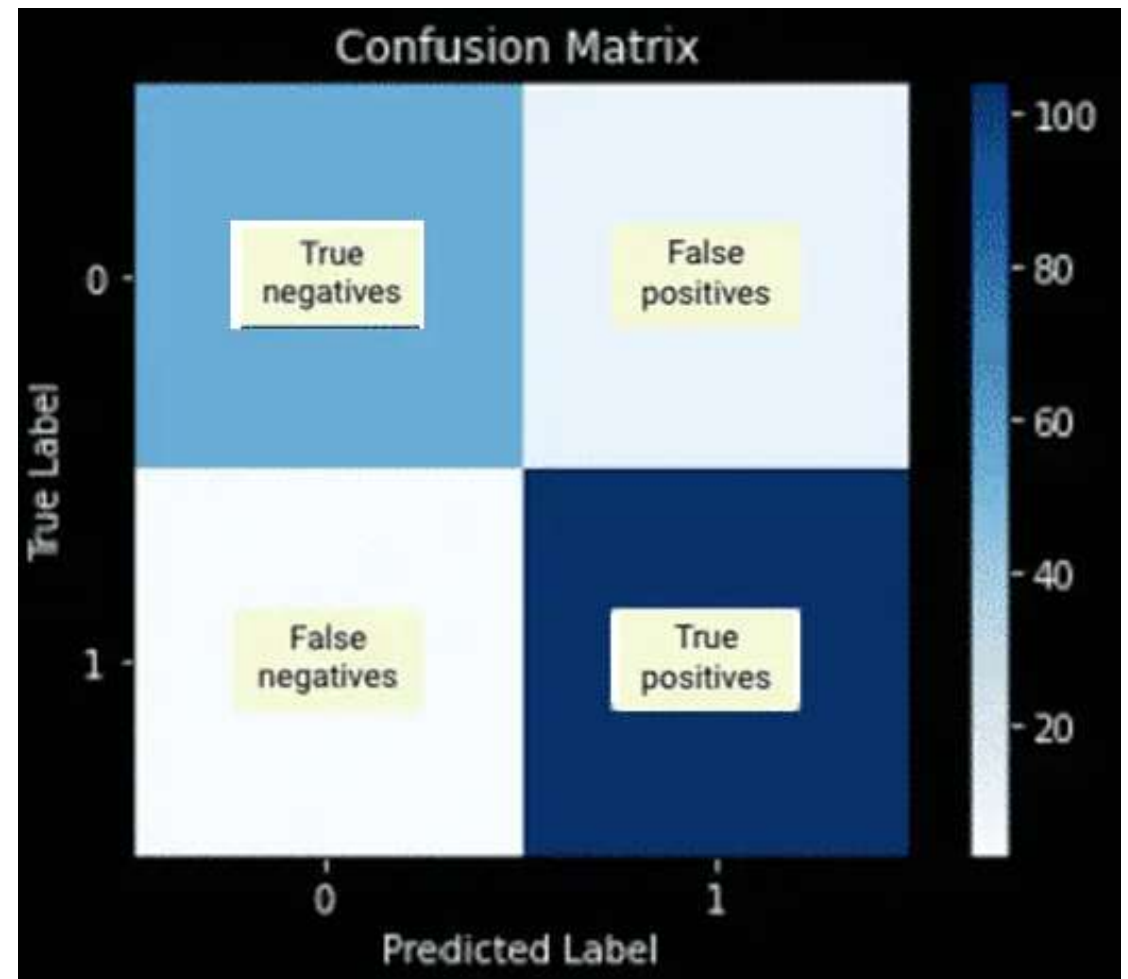
# Confusion Matrix

- A Confusion matrix is an  $N \times N$  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.

A good model is one which has ***high TP and TN rates***, while ***low FP and FN rates***.

If you have an ***imbalanced dataset*** to work with, it's always better to use ***confusion matrix*** as your evaluation criteria for your machine learning model.

- This is a binary classification problem .
- Some of the commonly used terms are:
  - **True positives (TP)**
    - Predicted positive and are actually positive.
  - **False positives (FP)**
    - Predicted positive and are actually negative.
  - **True negatives (TN)**
    - Predicted negative and are actually negative.
  - **False negatives (FN)**
    - Predicted negative and are actually positive.



`cm[0][0] = TN`  
`cm[0][1] = FP`  
`cm[1][0] = FN`  
`cm[1][1] = TP`

**True Negative (Top-Left Quadrant)**  
**False Positive (Top-Right Quadrant)**  
**False Negative (Bottom-Left Quadrant)**  
**True Positive (Bottom-Right Quadrant)**

- A **confusion matrix** is a tabular summary of the number of **correct and incorrect predictions** made by a classifier.
- It is used to measure the performance of a classification model.
- It can be used to evaluate the performance of a classification model through the calculation of performance metrics like
  - a. Accuracy
  - b. Precision
  - c. Recall (TPR, Sensitivity)
  - d. F1-Score
  - e. FPR (Type I Error)
  - f. FNR (Type II Error)
- Confusion matrices are widely used because they give a better idea of a model's performance than classification accuracy does.

# Accuracy

- **Accuracy** simply measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions.
- The **accuracy metric** is ***not suited*** for **imbalanced classes**. **Accuracy** has its own *disadvantages*, for **imbalanced data**, when the model predicts that each point belongs to the majority class label, the accuracy will be high. But, the model is not accurate.
- **Accuracy** is a valid choice of ***evaluation for classification problems*** which are ***well balanced*** and ***not skewed or there is no class imbalance***.

		Predicted	
		Negative (N) -	Positive (P) +
Actual	Negative -	True Negatives (TN)	False Positives (FP) Type I error
	Positive +	False Negatives (FN) Type II error	True Positives (TP)

ACTUAL VALUES





Positive (CAT)

Negative (DOG)

PREDICTED VALUES

Positive (CAT)

Negative (DOG)

	<b>TRUE POSITIVE</b> 6 YOU ARE A CAT		<b>FALSE NEGATIVE</b> 1 YOU ARE A DOG TYPE II ERROR
	<b>FALSE POSITIVE</b> 2 YOU ARE A CAT TYPE I ERROR		<b>TRUE NEGATIVE</b> 11 YOU ARE NOT A CAT

		PREDICTED VALUES	
		Positive (1)	Negative (0)
ACTUAL VALUES	Positive (1)	<div>6</div> <div>TRUE POSITIVE</div>	<div>1</div> <div>FALSE NEGATIVE</div>
	Negative (0)	<div>2</div> <div>FALSE POSITIVE</div>	<div>11</div> <div>TRUE NEGATIVE</div>

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{6 + 11}{6 + 11 + 2 + 1} = 85\%$$

Accuracy

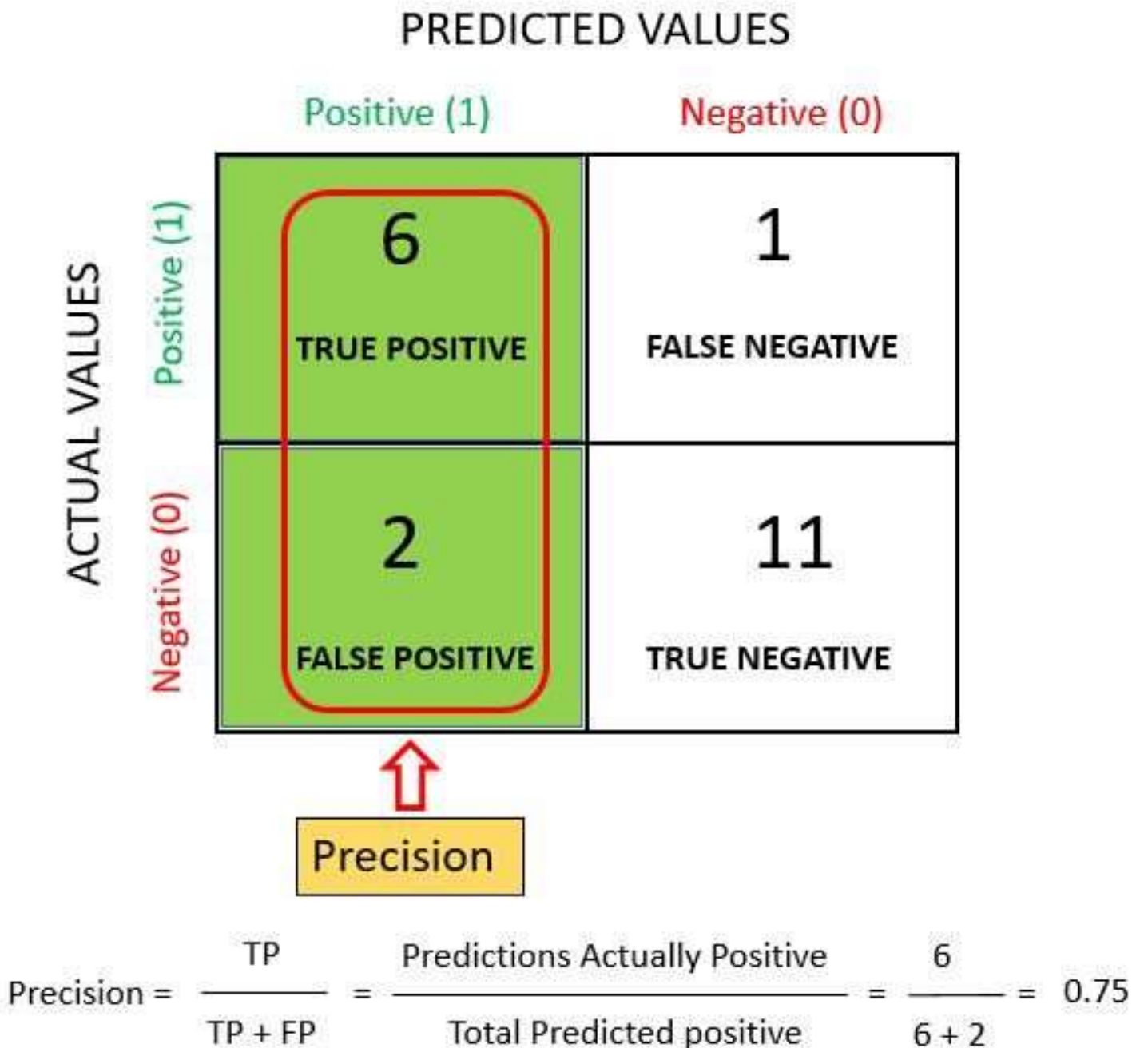


# Precision

- It is a measure of **correctness** that is achieved in **true prediction**. In simple words, it tells us how many predictions are ***actually positive*** out of all the ***total positive predicted***.
- Precision is defined as the ratio of the total number of *correctly classified positive classes* divided by the *total number of predicted positive classes*. Or, out of all the predictive positive classes, how much we predicted correctly. **Precision should be high(ideally 1).**
- “***Precision*** is a useful metric in cases where ***False Positive*** is a higher concern than ***False Negatives***”

**Ex 1:- In Spam Detection** : Need to focus on **precision**  
Suppose **mail is not a spam** but model is **predicted as spam** : FP (False Positive). We always try to reduce FP.

**Ex 2:- Precision** is important in *music or video recommendation systems, e-commerce websites, etc.* Wrong **results** could lead to customer churn and be *harmful to the business*.



# Recall

- It is a measure of **actual observations** which are predicted **correctly**, i.e. how many observations of positive class are actually predicted as positive. It is also known as **Sensitivity**. **Recall** is a valid choice of evaluation metric when we want to capture ***as many positives*** as possible.
- Recall is defined as the ratio of the total number of *correctly classified positive classes* divide by the *total number of positive classes*. Or, out of all the positive classes, how much we have predicted correctly. **Recall should be high(ideally 1)**.
- “***Recall is a useful metric in cases where False Negative trumps False Positive***”

		PREDICTED VALUES	
		Positive (1)	Negative (0)
ACTUAL VALUES	Positive (1)	<div>6</div> <div>TRUE POSITIVE</div>	<div>1</div> <div>FALSE NEGATIVE</div>
	Negative (0)	<div>2</div> <div>FALSE POSITIVE</div>	<div>11</div> <div>TRUE NEGATIVE</div>

← Recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Predictions Actually Positive}}{\text{Total Actual positive}} = \frac{6}{6 + 1} = 0.85$$

- **Ex 1:-** suppose person having cancer (or) not? **He is suffering from cancer** but model predicted as **not suffering from cancer**
- **Ex 2:-** *Recall is important in **medical cases** where it doesn't matter whether we raise a false alarm but the **actual positive cases should not go undetected!***
- *Recall would be a better metric because we don't want to **accidentally discharge an infected person** and let them mix with the healthy population thereby **spreading contagious virus**. Now you can understand why accuracy was a bad metric for our model.*
- **Trick to remember : Precision has Predictive Results in the denominator.**

# F-Measure/F1 Score

- The **F1 score** is a number between **0 and 1** and is the ***harmonic mean of precision and recall***. We use harmonic mean because it is not sensitive to extremely large values, unlike simple averages.
- **F1 score** sort of maintains a **balance** between the ***precision and recall*** for your classifier. If your ***precision is low***, the ***F1 is low*** and if the ***recall is low*** again your ***F1 score is low***.
- There will be cases where there is no clear distinction between whether *Precision is more important or Recall*. We combine them!
- In practice, when we try to increase the precision of our model, the recall goes down and vice-versa. The F1-score captures both the trends in a single value.

$$\text{F1-Score} = 2 * \frac{(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} = 2 * \frac{(0.85 * 0.75)}{(0.85 + 0.75)} = 0.79$$

- F1 score is a harmonic mean of Precision and Recall.
- As compared to Arithmetic Mean, Harmonic Mean punishes the extreme values more. F
- 1-score should be high(ideally 1).

# When to use Accuracy / Precision / Recall / F1-Score?

- **Accuracy** is used when the **True Positives and True Negatives** are more important. **Accuracy** is a better metric for **Balanced Data**.
- Whenever **False Positive** is much more important use **Precision**.
- Whenever **False Negative** is much more important use **Recall**.
- **F1-Score** is used when the **False Negatives and False Positives** are important. **F1-Score** is a better metric for **Imbalanced Data**.



# Confusion Matrix in Python

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test,y_pred)
```

```
Cm
```

## OUTPUT

```
array([[ 57,  7],  
2      [  5, 102]])
```

# Run the classification report

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.92	0.89	0.90	64
1	0.94	0.95	0.94	107
accuracy			0.93	171
macro avg	0.93	0.92	0.92	171
weighted avg	0.93	0.93	0.93	171

Metric	What it is	Sklearn's Metric Method
Accuracy	$(\text{true positive} + \text{true negative}) / \text{total predictions}$	<code>metrics.precision_score(true, pred)</code>
Precision	$\text{true positive} / (\text{true positive} + \text{false positive})$	<code>metrics.precision_score(true, pred)</code>
Recall (sensitivity)	$\text{true positive} / (\text{true positive} + \text{false negative})$	<code>metrics.recall_score(true, pred)</code>
F1-Score	$2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$	<code>metrics.f1_score(true, pred)</code>
Specificity	$\text{true negative} / (\text{true negative} + \text{false positive})$	<code>metrics.recall_score(true, pred, pos_label=0)</code>

**`confusion_matrix(y_true, y_pred, *, labels=None, sample_weight=None, normalize=None)`**

- `y_true`: Ground truth (correct) target values. array-like of shape (n\_samples,)
- `y_pred`: Estimated targets as returned by a classifier. array-like of shape (n\_samples,)
- `labels`: List of labels to index the matrix. This may be used to reorder or select a subset of labels. If None is given, those that appear at least once in `y_true` or `y_pred` are used in sorted order. array-like of shape (n\_classes), default=None
- `sample_weight`: Sample weights. array-like of shape (n\_samples,), default=None
- `normalize` : Normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. If None, confusion matrix will not be normalized. {'true', 'pred', 'all'}, default=None

AUC – ROC Curve

# AUC-ROC:

- 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. e.g., the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.
- ROC is an acronym for Receiver Operating Characteristics.
- AUC is an acronym for Area Under Curve.

# Threshold

- The classification threshold in machine learning is a **boundary** or a **cut-off point** used to assign a specific predicted class for each object.
- Any machine learning algorithm for classification gives output in the probability format, i.e probability of an instance belonging to a particular class.
- In order to assign a class to an instance for binary classification, we compare the probability value to the threshold, i.e if the value is greater than or less than the threshold.
- For probability  $\geq$  threshold, class = 1
- probability  $<$  threshold, class = 0



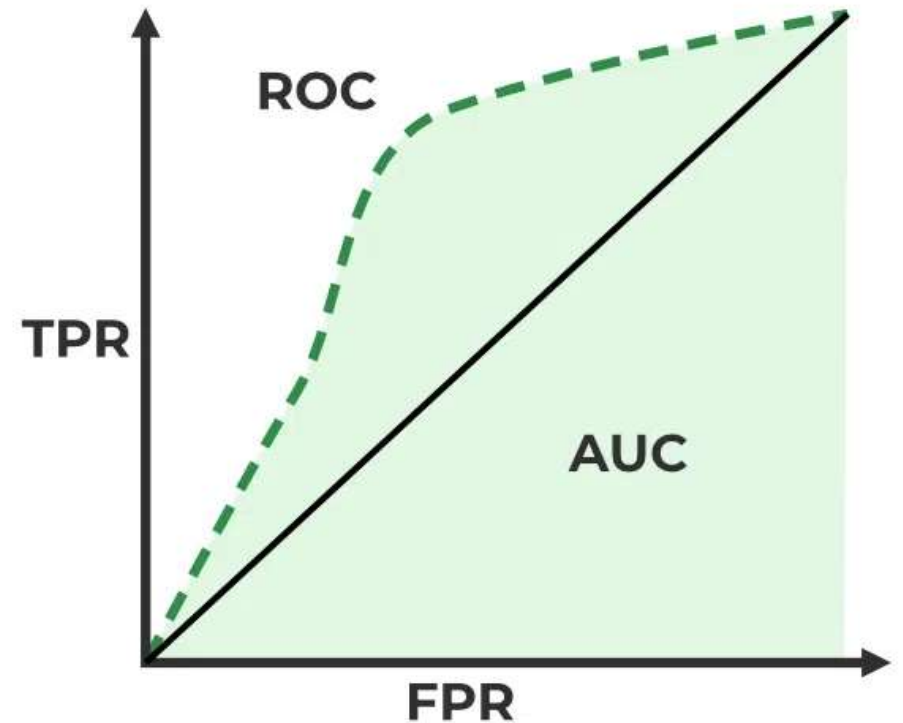
- x-axis represents the **False Positive Rate (FPR)**
- y-axis represents the **True Positive Rate (TPR)**.

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

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

$$\text{FPR} = 1 - \text{Specificity}$$

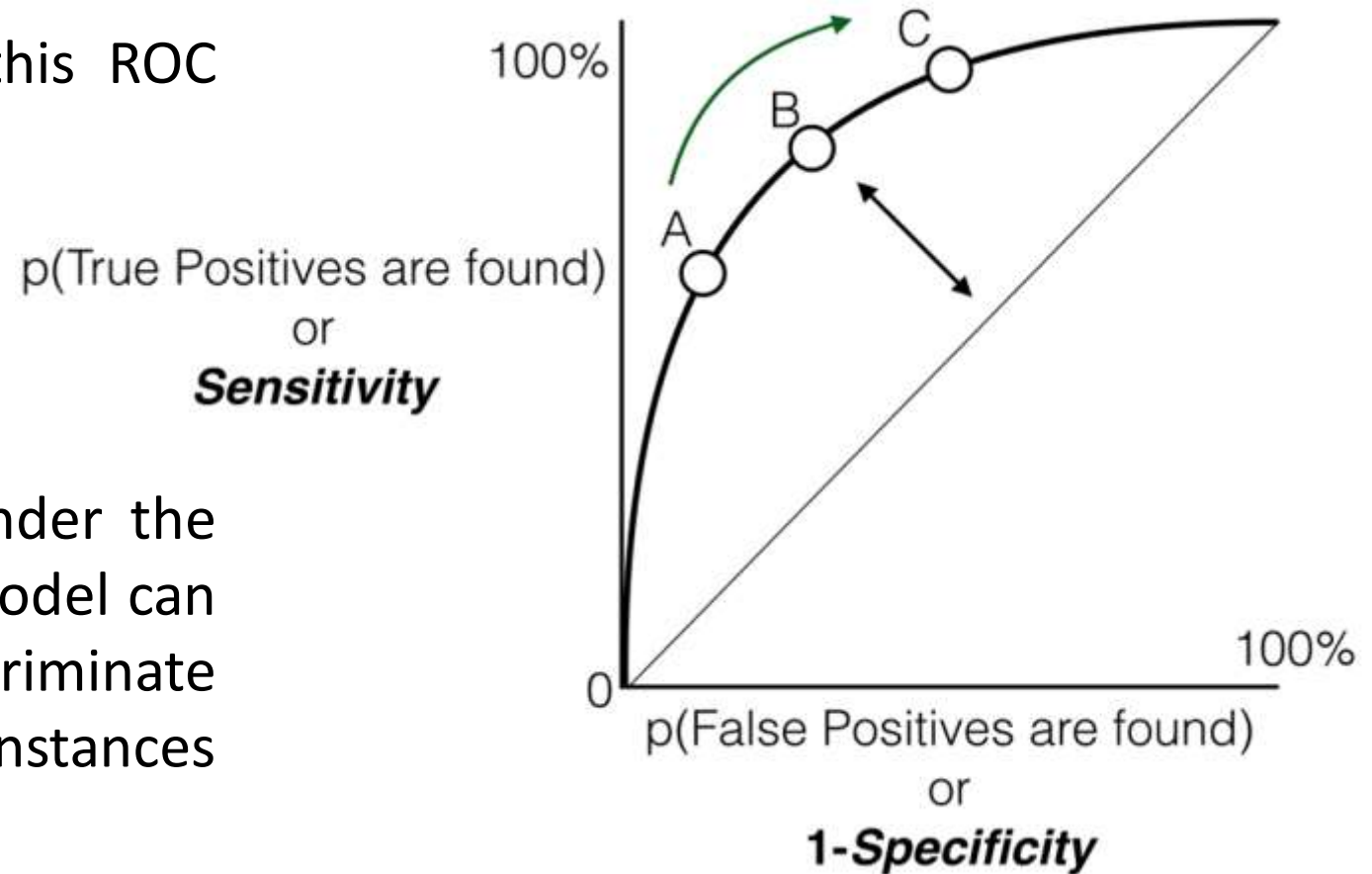
$$= \frac{\text{FP}}{\text{TN} + \text{FP}}$$



Sensitivity and Specificity are inversely proportional to each other.  
TPR and FPR are proportional to each other.

# ROC Curve

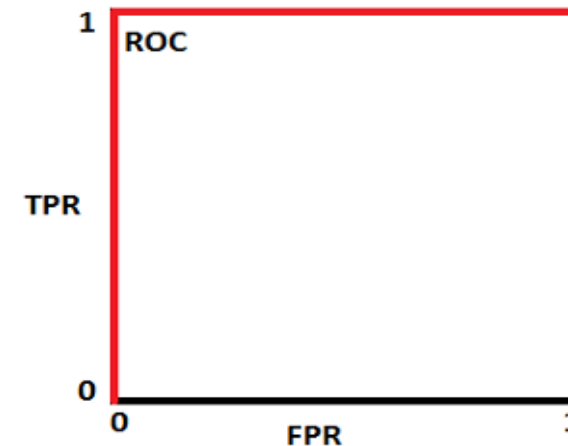
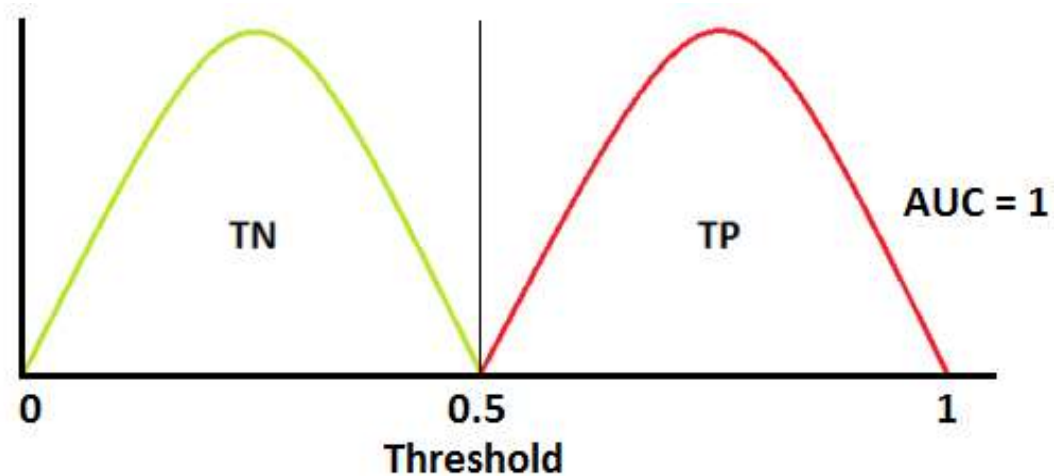
- ROC is the plot between TPR and FPR across all possible thresholds
  - AUC is the entire area beneath this ROC curve.
- 
- The **ROC AUC score** is the area under the ROC curve. It sums up how well a model can produce relative scores to discriminate between positive or negative instances across all classification thresholds.
- 
- The ROC AUC score **ranges from 0 to 1**



- AUC is desirable for the following two reasons:
- **AUC is scale-invariant:** It measures how well predictions are ranked, rather than their absolute values.
- **AUC is classification-threshold-invariant:** It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

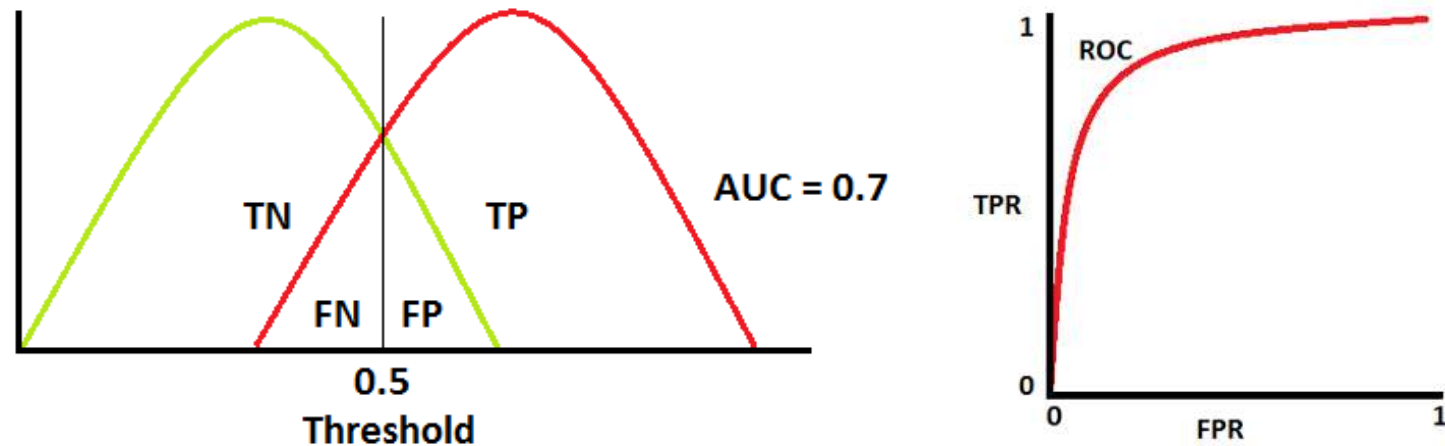
# ROC Curve

- The red distribution shows True Positive and Green distribution shows True Negative.
- If there is no overlap,  $AUC = 1$  and the model has ideal measure of separability.



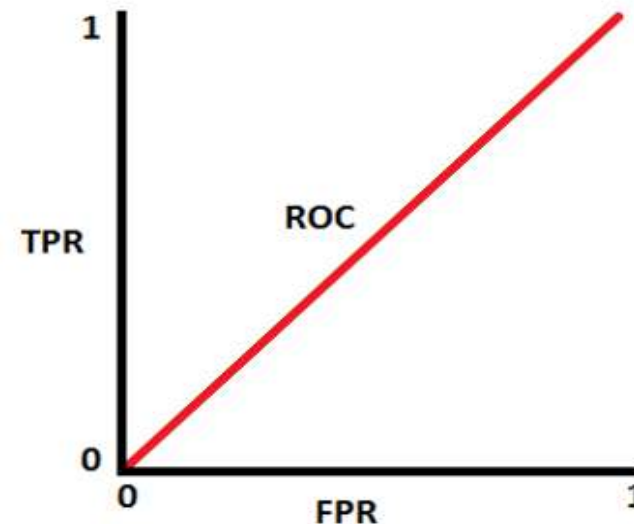
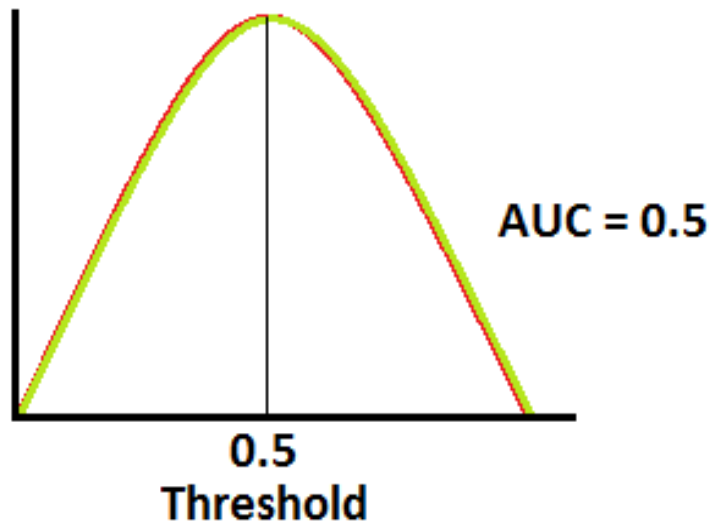
# ROC Curve

- Figure shows an overlap between the two distributions.
- The overlap results in Type 1 and Type 2 errors.
- We can minimize and maximize them using the threshold.
- The AUC – 0.7 means that there are 70% chances that the model will be able to discriminate between negative and positive classes



# ROC Curve

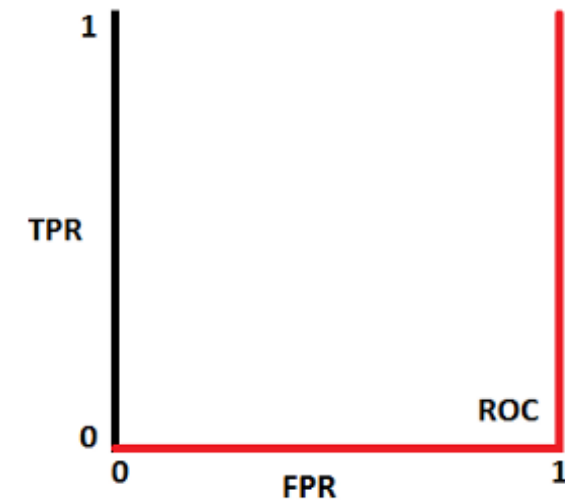
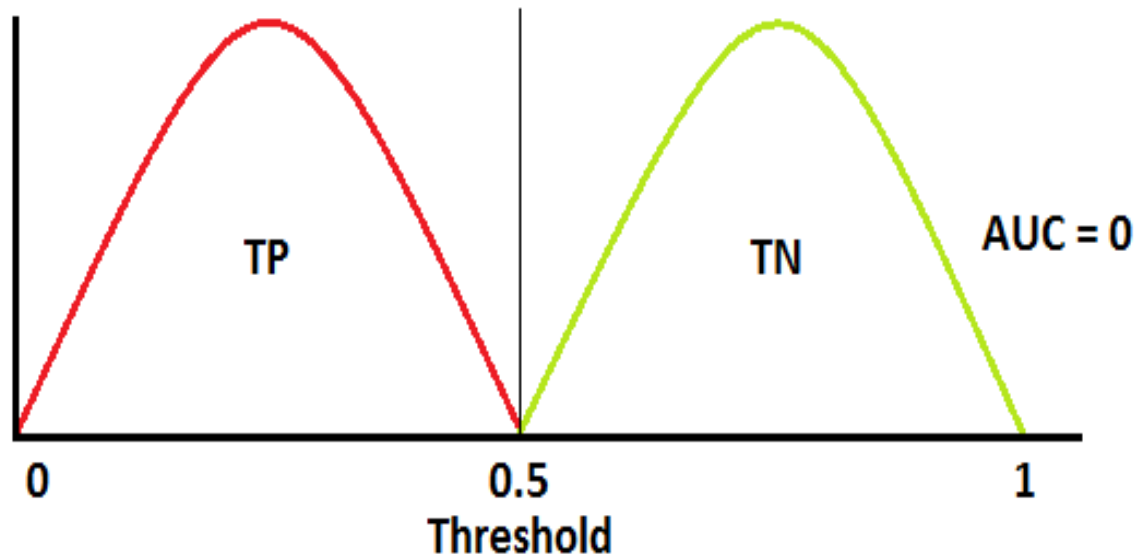
- Figure demonstrate the two distributions that have completely overlapped.
- In this case the distribution will not be able to differentiate between negative and positive classes.
- AUC is 0.5 in this case.
- This is the worst situation. When AUC is approximately 0.5, the model has no discrimination capacity to distinguish between positive class and negative class.



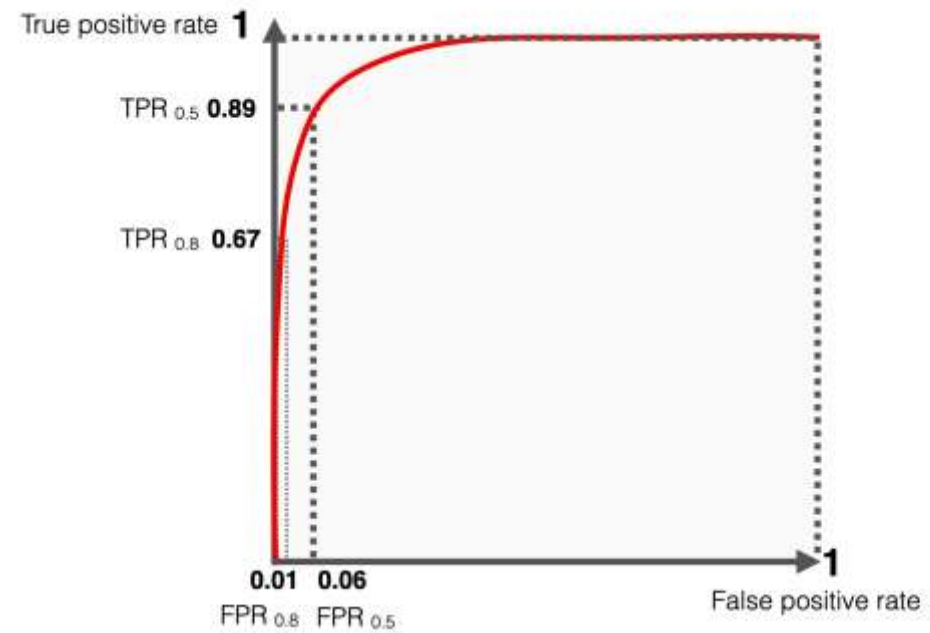
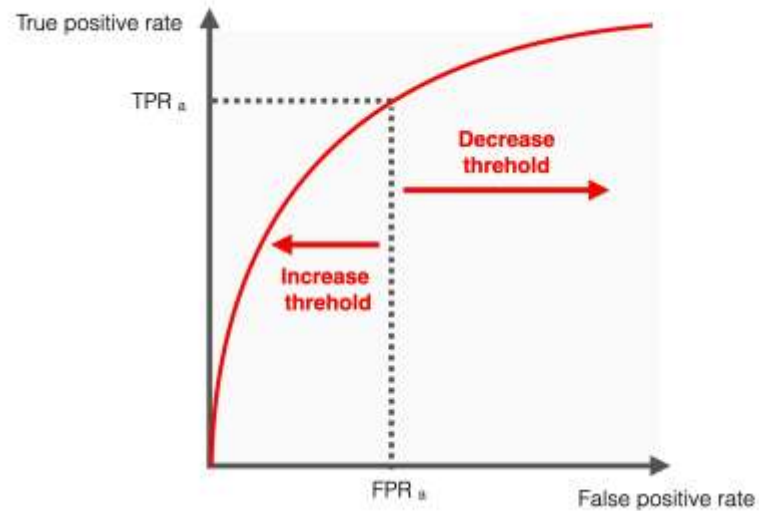
# ROC Curve

Figure demonstrates the case where both the distributions have swapped. In the case AUC is 0.

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.



When you increase the threshold, you move left on the curve. If you decrease the threshold, you move to the right





# Multi-Class Confusion Matrix

- A multi-class confusion matrix is different from a binary confusion matrix. Let's explore how this is different:
- **Diagonal elements:** values along the diagonal represent the number of instances where the model correctly predicted the class. They are equivalent to True Positives (TP) in the binary case, but for each class.
- **Off-diagonal elements:** all values that aren't on the diagonal represent the number of instances where the model incorrectly predicted the class. They correspond to False Positives (FP) and False Negatives (FN) in the binary case, but for each combination of classes.
- In a multi-class confusion matrix, the sum of all diagonal elements gives the total number of correct predictions, and the sum of all off-diagonal elements gives the total number of incorrect predictions.

# Thanks

Samatrix Consulting Pvt Ltd