

KNN Algorithm

STEP 1: Choose the number K of neighbors



STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance



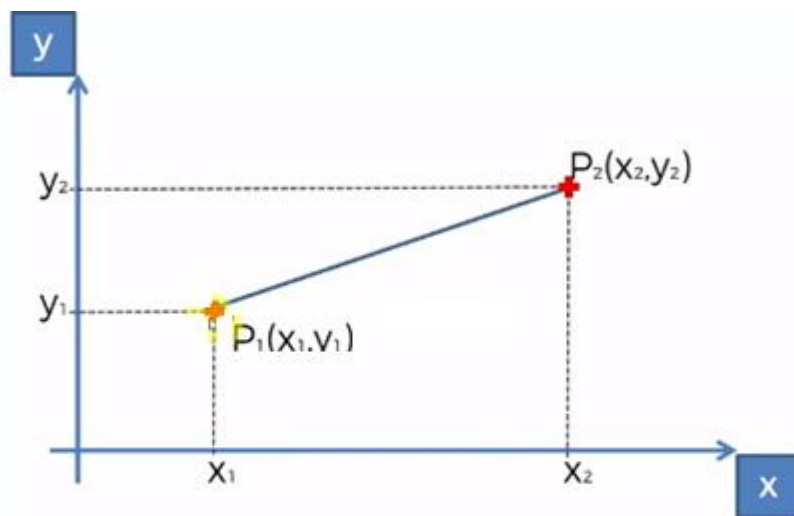
STEP 3: Among these K neighbors, count the number of data points in each category



STEP 4: Assign the new data point to the category where you counted the most neighbors



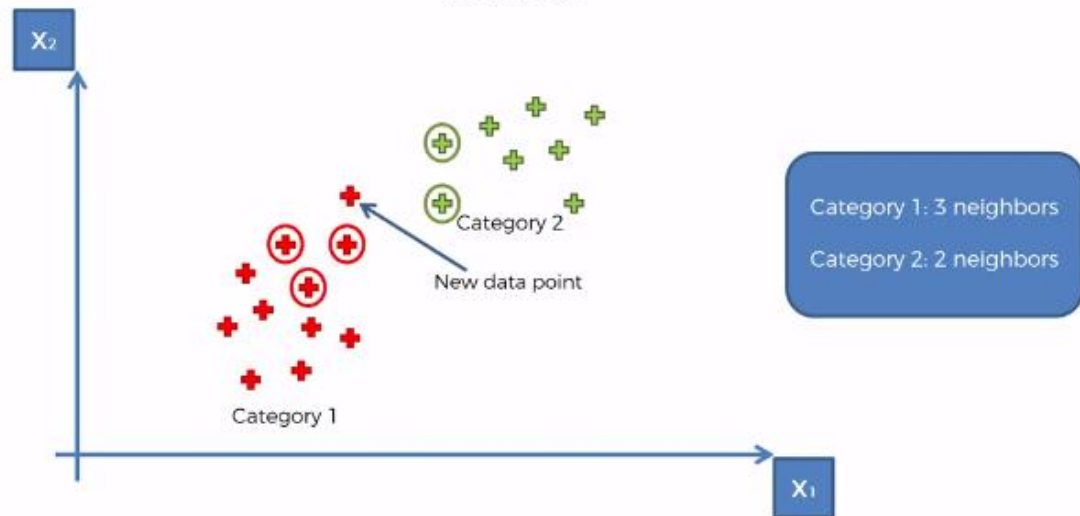
Your Model is Ready



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

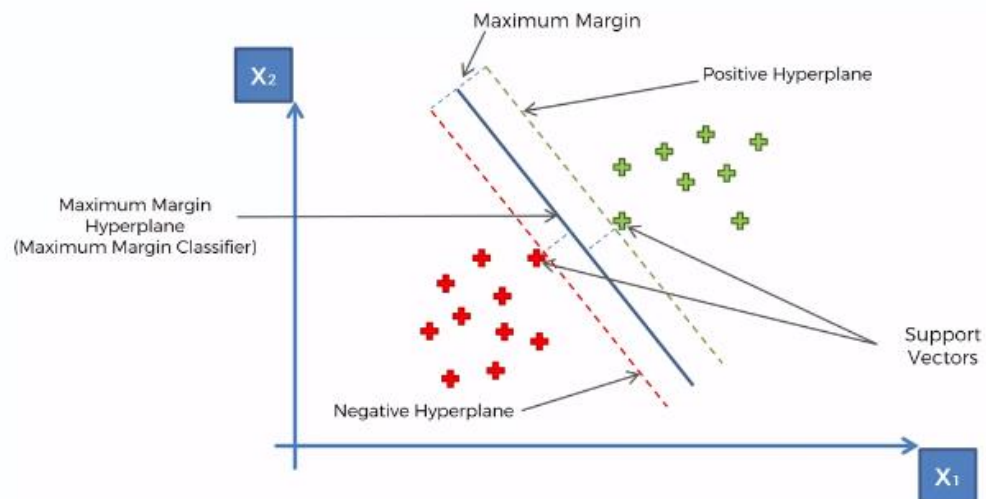
+

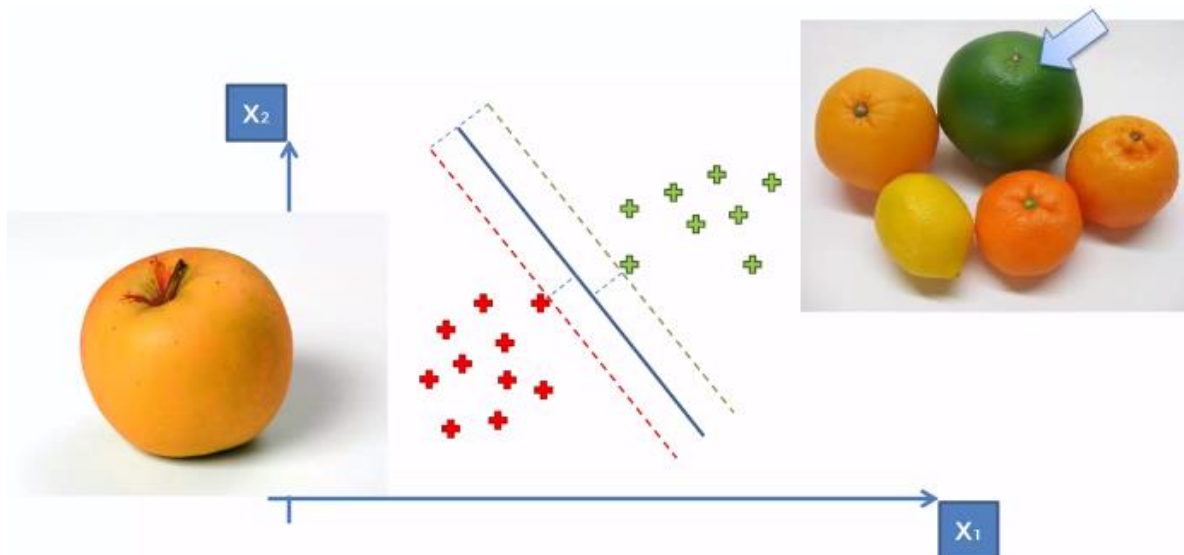
STEP 4: Assign the new data point to the category where you counted the most neighbors



SVM

Hyperplanes

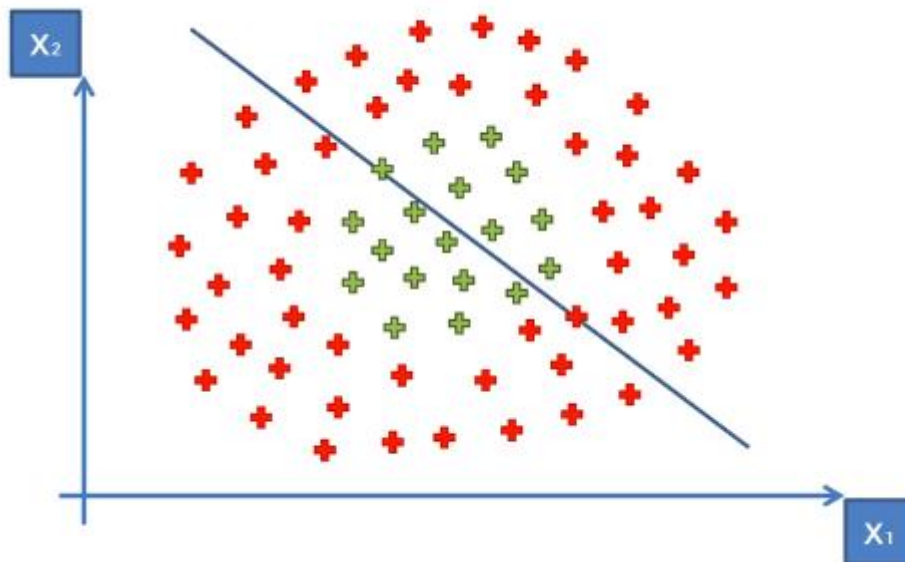




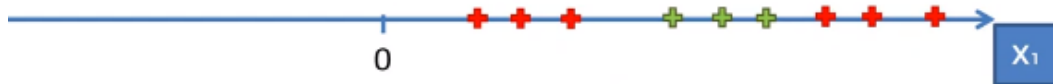
Finding boundaries

Kernel SVM

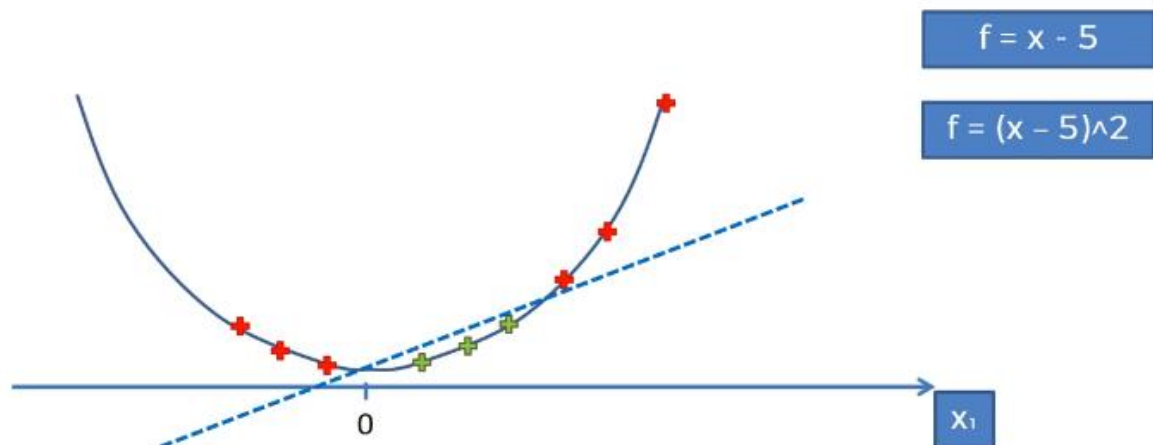
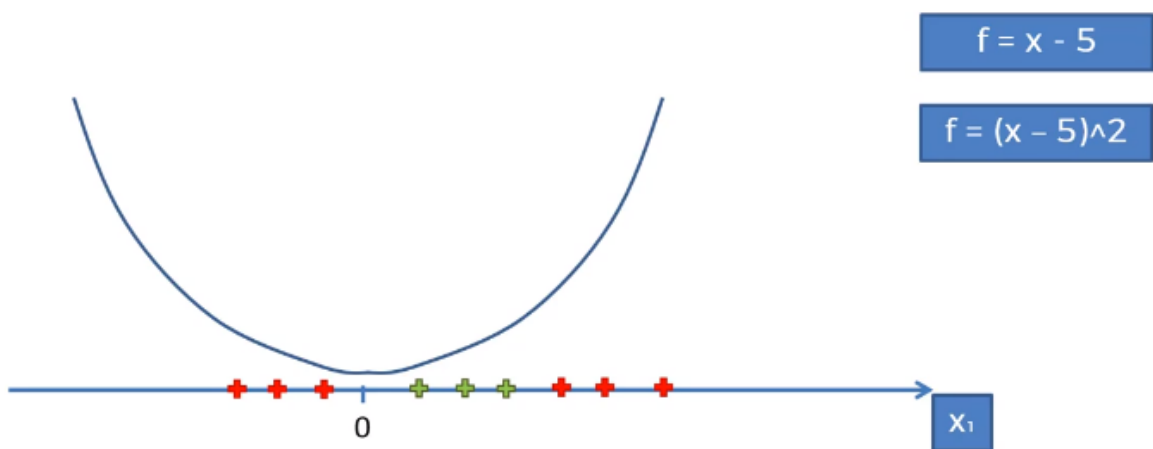
SVM cannot be used in following scenario

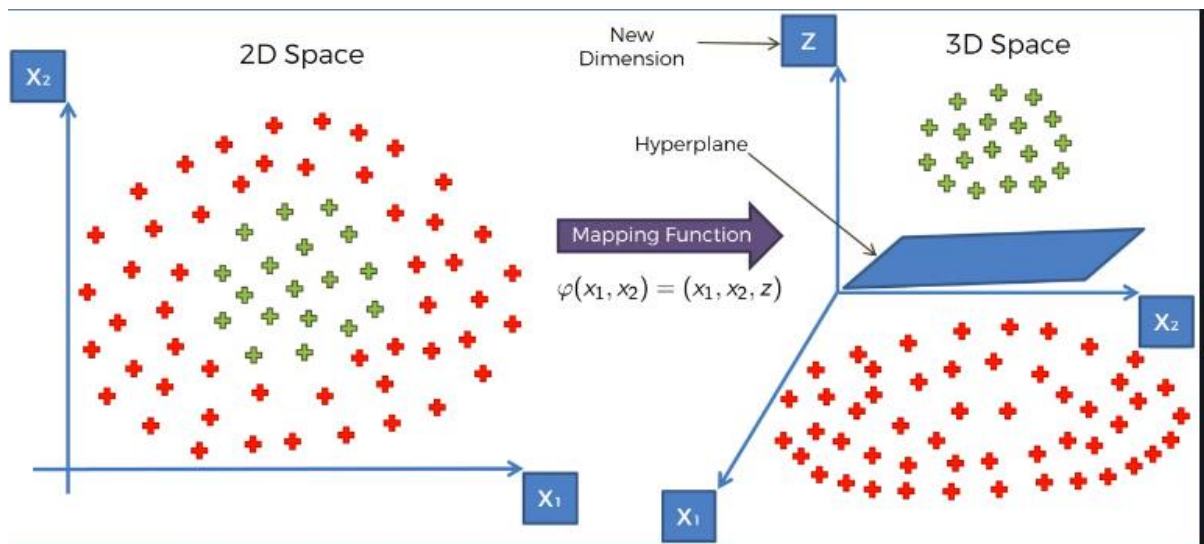


So we convert data from low dimensions to high dimension using formula



This is 1 D data

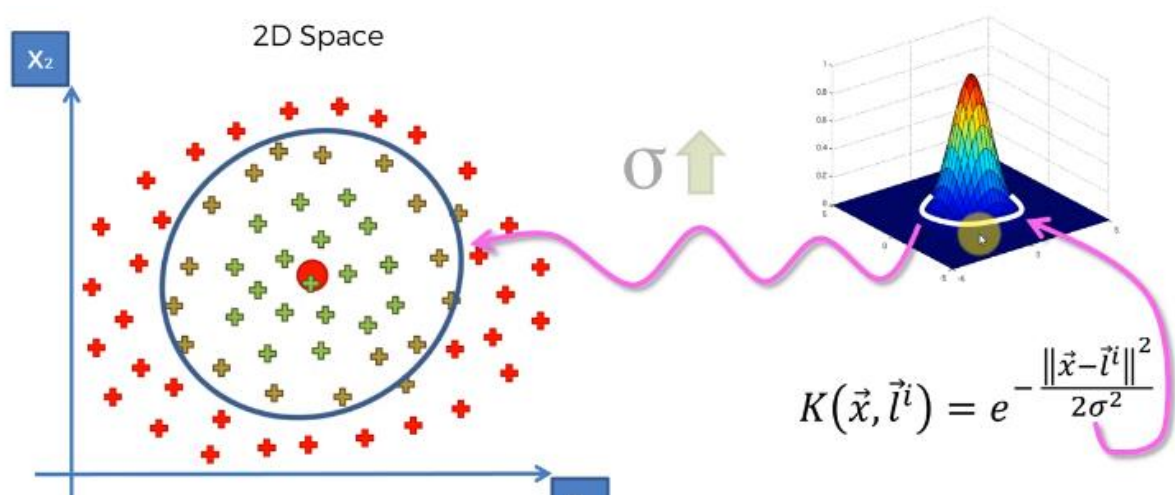




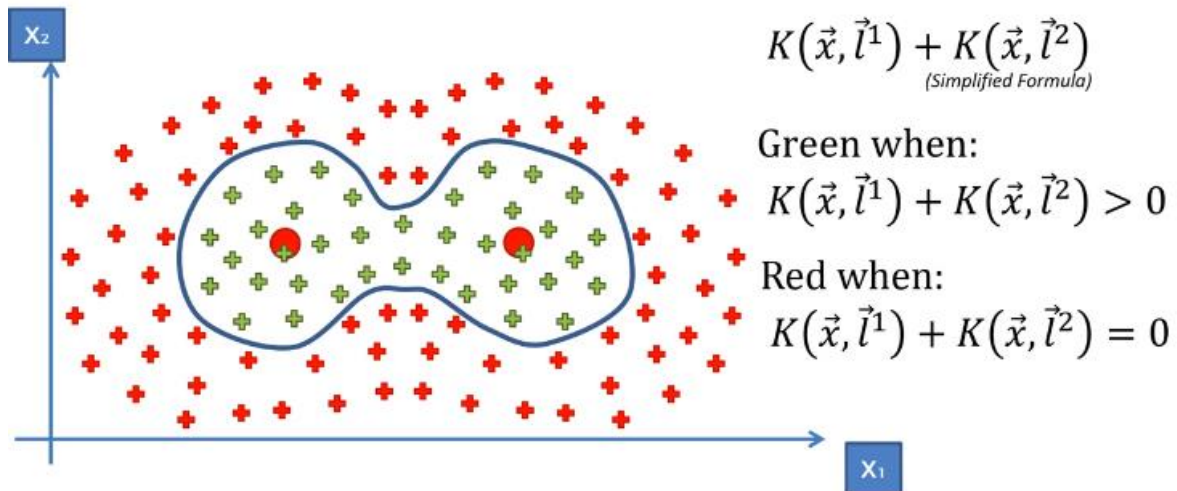
But this is highly compute intensive and hence slow.

So we use kernel function

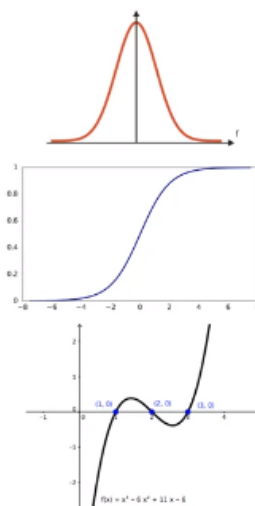
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x} - \vec{l}^i\|^2}{2\sigma^2}}$$



Based on sigma the size of circle is dependent



Types of Kernel Functions



Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x} - \vec{l}^i\|^2}{2\sigma^2}}$$

Sigmoid Kernel

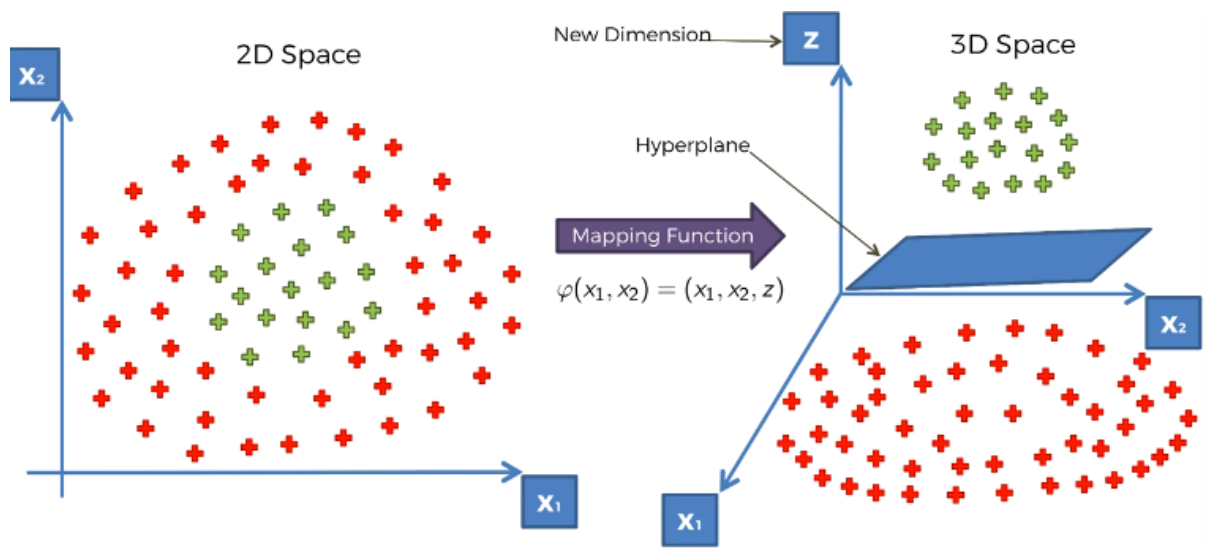
$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$

Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

These are various kernel functions we may use in algorithm

Behind the scene it is calculating higher dimension



Bay's theorem

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Bayes Theorem

Mach1: 30 wrenches / hr

Mach2: 20 wrenches / hr

Out of all produced parts:

We can SEE that 1% are defective

Out of all defective parts:

We can SEE that 50% came from mach1

And 50% came from mach2

Bayes Theorem

Mach1: 30 wrenches / hr

Mach2: 20 wrenches / hr

-> $P(\text{Mach1}) = 30/50 = 0.6$

-> $P(\text{Mach2}) = 20/50 = 0.4$

Out of all produced parts:

We can SEE that 1% are defective

-> $P(\text{Defect}) = 1\%$

Out of all defective parts:

We can SEE that 50% came from mach1

And 50% came from mach2

Question:

**What is the probability that a part
produced by mach2 is defective = ?**

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr

$$\rightarrow P(\text{Mach1}) = 30/50 = 0.6$$

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

Out of all produced parts:
We can SEE that 1% are defective

$$\rightarrow P(\text{Defect}) = 1\%$$

Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2

$$\rightarrow P(\text{Mach1} \mid \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Mach2} \mid \text{Defect}) = 50\%$$

Question:
What is the probability that a part
produced by mach2 is defective = ?

$$\rightarrow P(\text{Defect} \mid \text{Mach2}) = ?$$

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr
Out of all produced parts:
We can SEE that 1% are defective
Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2
Question:
What is the probability that a part
produced by mach2 is defective = ?

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

$$\rightarrow P(\text{Defect}) = 1\%$$

$$\rightarrow P(\text{Mach2} \mid \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Defect} \mid \text{Mach2}) = ?$$

$$P(\text{Defect} \mid \text{Mach2}) = \frac{P(\text{Mach2} \mid \text{Defect}) \cdot P(\text{Defect})}{P(\text{Mach2})}$$

Bayes Theorem

Mach1: 30 wrenches / hr

Mach2: 20 wrenches / hr

Out of all produced parts:

We can SEE that 1% are defective

Out of all defective parts:

We can SEE that 50% came from mach1

And 50% came from mach2

Question:

What is the probability that a part produced by mach2 is defective = ?

$$\rightarrow P(\text{Mach2}) = 20/50 = 0.4$$

$$\rightarrow P(\text{Defect}) = 1\%$$

$$\rightarrow P(\text{Mach2} | \text{Defect}) = 50\%$$

$$\rightarrow P(\text{Defect} | \text{Mach2}) = ?$$

$$P(\text{Defect} | \text{Mach2}) = \frac{0.5 \cdot 0.01}{0.4}$$

$$P(\text{Defect} | \text{Mach2}) = \frac{P(\text{Mach2} | \text{Defect}) \cdot P(\text{Defect})}{P(\text{Mach2})} = 1.25\%$$

Let's look at an example:

- 1000 wrenches
- 400 came from Mach2
- 1% have a defect = 10
- of them 50% came from Mach2 = 5
- % defective parts from Mach2 = $5/400 = 1.25\%$

$$P(\text{Defect} \mid \text{Mach2}) = \frac{P(\text{Mach2} \mid \text{Defect}) \cdot P(\text{Defect})}{P(\text{Mach2})} = 1.25\%$$

Let's look at an example:

- 1000 wrenches
- 400 came from Mach2
- 1% have a defect = 10
- of them 50% came from Mach2 = 5
- % defective parts from Mach2 = $5/400 = 1.25\%$

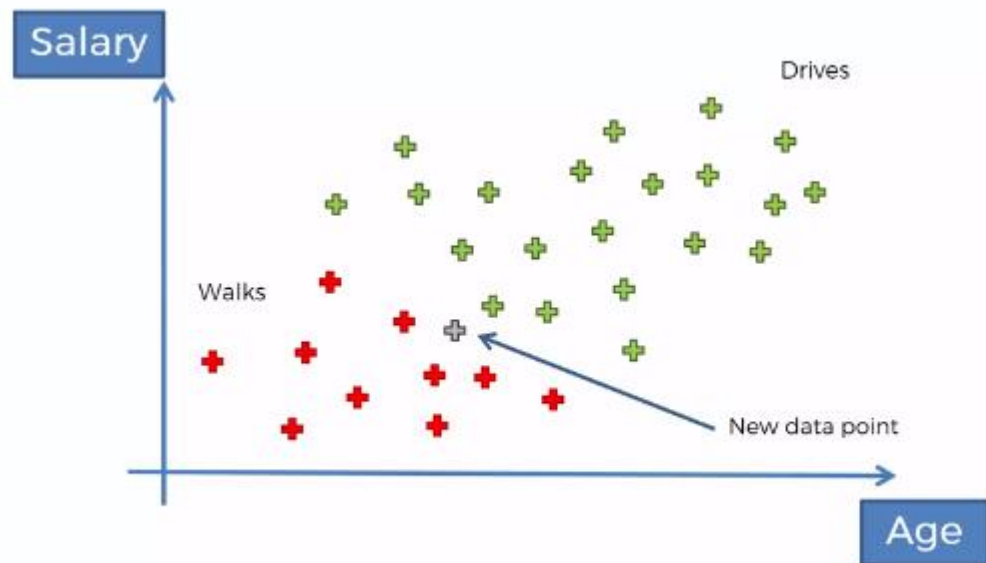
Why we are not counting defective spanner coming from machine 2

Obvious question:

If the items are labeled, why couldn't we just count the number of defective wrenches that came from Mach2 and divide by the total number that came from Mach2?

It may be time consuming when the number of spanners increases

Naïve nbays classification



Step 1 check probability that new person walks

$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

#2 Marginal Likelihood

#4 Posterior Probability #3 Likelihood #1 Prior Probability

$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

#2 Marginal Likelihood

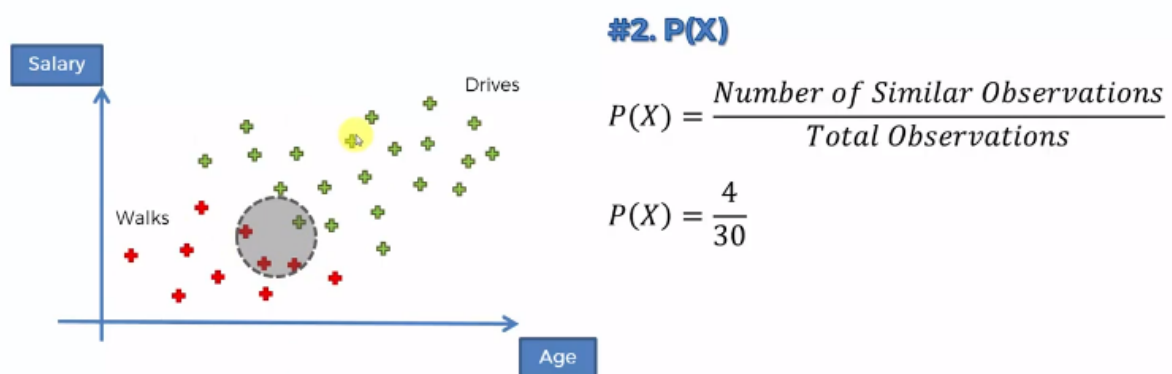
Step 3 : compare both probability

Naïve bays is probability based classifier

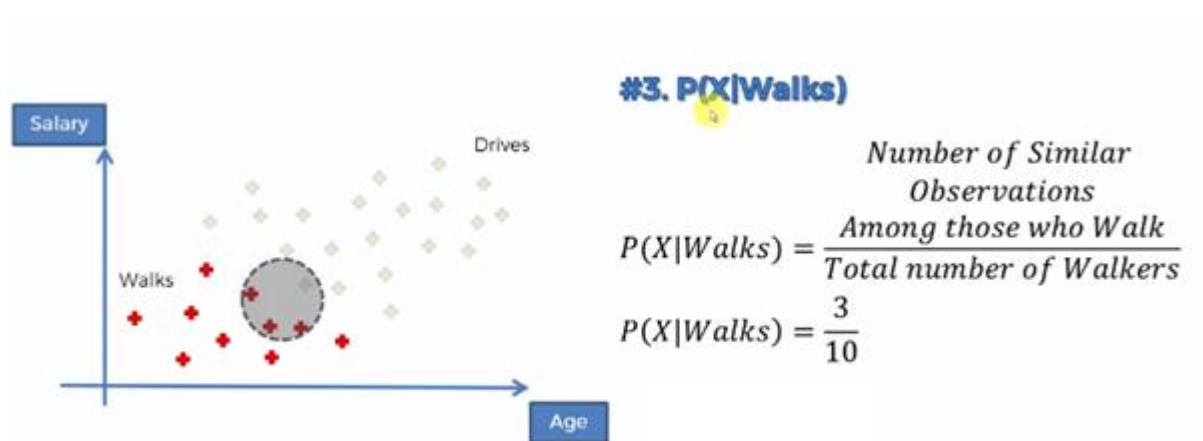
$$P(Walks|X) \text{ v.s. } P(Drives|X)$$

Draw a circle nearby the point you want to add

And check how many points in the circle and calculate



What is likely hood that person walks



w

Diagram illustrating the calculation of the Posterior Probability (#4) using the Likelihood (#3), Prior Probability (#1), and Marginal Likelihood (#2):

$$P(Walks|X) = \frac{\frac{3}{10} * \frac{10}{30}}{\frac{4}{30}} = 0.75$$

The components are labeled as follows:

- #4 Posterior Probability
- #3 Likelihood
- #1 Prior Probability
- #2 Marginal Likelihood

Same way calculate it for drive

#4 Posterior Probability

✓ #3 Likelihood

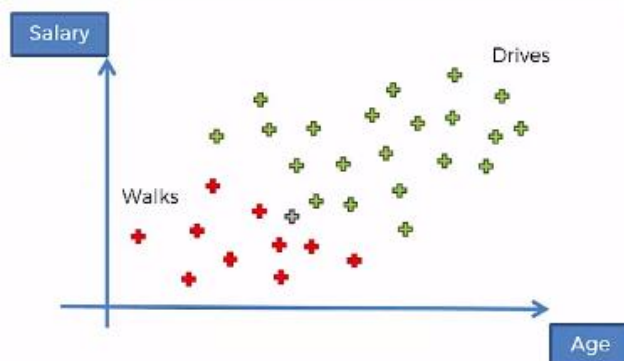
✓ #1 Prior Probability

$$P(\text{Drives}|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

✓ #2 Marginal Likelihood

Person walks probability is 0.75 %

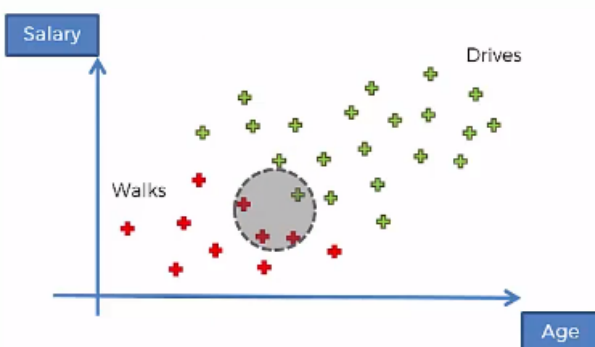
So we will classify the person walks



#1. $P(\text{Drives})$

$$P(\text{Drives}) = \frac{\text{Number of Drivers}}{\text{Total Observations}}$$

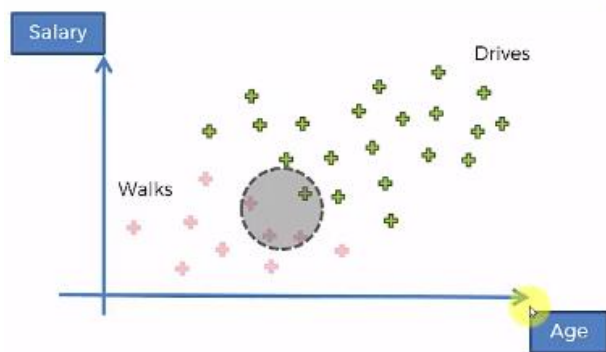
$$P(\text{Drives}) = \frac{20}{30}$$



#2. $P(X)$

$$P(X) = \frac{\text{Number of Similar Observations}}{\text{Total Observations}}$$

$$P(X) = \frac{4}{30}$$



#3. $P(X|Drives)$

$$P(X|Drives) = \frac{\text{Number of Similar Observations Among those who Walk}}{\text{Total number of Walkers}}$$

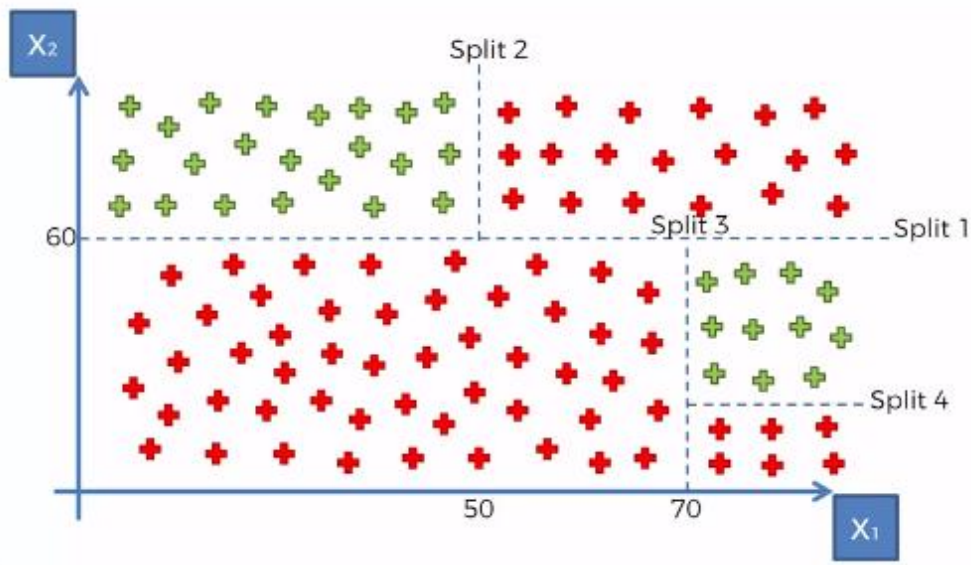
$$P(X|Drives) = \frac{1}{20}$$

So in actual calculation we may compare by using

$$\frac{P(X|Walks) * P(Walks)}{\cancel{P(X)}} \text{ v. s. } \frac{P(X|Drives) * P(Drives)}{\cancel{P(X)}}$$

But can be used only for comparison and not for calculation

Decision tree classification



Random forest classifier

Random Forest Intuition

STEP 1: Pick at random K data points from the Training set.



STEP 2: Build the Decision Tree associated to these K data points.



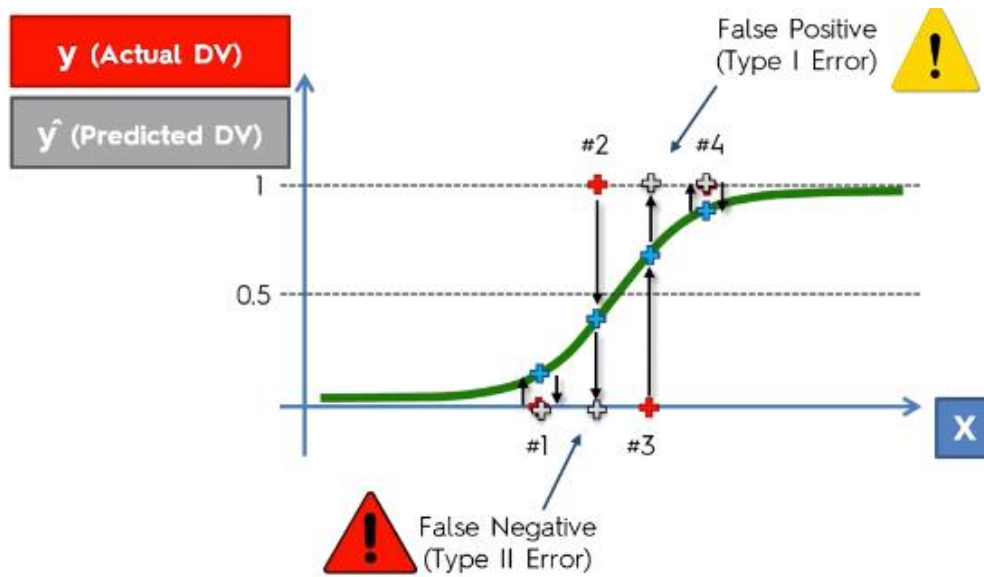
STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2



STEP 4: For a new data point, make each one of your Ntree trees predict the category to which the data points belongs, and assign the new data point to the category that wins the majority vote.

Used in motion sensing games like xbox

Confusion matrix



		\hat{y} (Predicted DV)	
		0	1
y (Actual DV)	0	35	5
	1	10	50

False Positive (Type I Error)



Calculate two rates

1. Accuracy Rate = Correct / Total
AR = 85/100 = 85%

2. Error Rate = Wrong / Total
ER = 15/100 = 15%

False Negative (Type II Error)

Drawback

		\hat{y} (Predicted DV)	
		0	1
y (Actual DV)	0	9,700	150 
	1	50 	100

Scenario 1:





Accuracy Rate = Correct / Total
AR = 9,800/10,000 = 98%

If we say that for every i/p the o/p will be zero we will not use model

So the accuracy for same example will be as follows and if you see accuracy increases

Which is wrong result called a accuracy paradox

so we cannot always predict it on confusion matrix

		\hat{y} (Predicted DV)	
		0	1
y (Actual DV)	0	9,850 	0 
	1	150 	0 

Scenario 1:

Accuracy Rate = Correct / Total
AR = 9,800/10,000 = 98%

Scenario 2:

Accuracy Rate = Correct / Total
AR = 9,850/10,000 = 98.5%

Understanding the Classification report through sklearn

by Muthu KrishnanPosted onJuly 7, 2018

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below. The report is copied from our [previous post](#) related to K-Means on Iris Dataset.

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	50
Iris-versicolor	0.77	0.96	0.86	50
Iris-virginica	0.95	0.72	0.82	50
avg / total	0.91	0.89	0.89	150

The code to generate a report similar to the one above is:

```
• from sklearn.metrics import classification_report
• target_names = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
• print(classification_report(irisdata['Class'],kmeans.labels_,target_names=target_names))
```

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes. There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative:** when a case was negative and predicted negative
2. **TP / True Positive:** when a case was positive and predicted positive
3. **FN / False Negative:** when a case was positive but predicted negative
4. **FP / False Positive:** when a case was negative but predicted positive

Precision – What percent of your predictions were correct?

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.

TP – True Positives
FP – False Positives

Precision – Accuracy of positive predictions.
$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

```
• from sklearn.metrics import precision_score
•
• print("Precision score: {}".format(precision_score(y_true,y_pred)))
```

Recall – What percent of the positive cases did you catch?

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

FN – False Negatives

Recall: Fraction of positives that were correctly identified.
Recall = $TP / (TP + FN)$

```
• from sklearn.metrics import recall_score
•
• print("Recall score: {}".format(recall_score(y_true,y_pred)))
```

F1 score – What percent of positive predictions were correct?

The F_1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F_1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F_1 should be used to compare classifier models, not global accuracy.

$F1 \text{ Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$

```
• from sklearn.metrics import f1_score
•
• print("F1 Score: {}".format(f1_score(y_true,y_pred)))
```

Another measure is you can use accuracy_score also

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
cm = confusion_matrix(y_test, y_pred)
```

```
#evaluate model  
  
print("Confusion Matrix\n",confusion_matrix(y_test, y_pred))  
  
print("classification Report\n",classification_report(y_test, y_pred))  
  
  
print("accuracy score\n",accuracy_score(y_test, y_pred))
```

ROC Curve

Understanding AUC - ROC Curve

In Machine Learning, performance measurement is an essential task. So when it comes to a classification problem, we can count on an AUC - ROC Curve. When we need to check or visualize the performance of the multi - class classification problem, we use AUC (**Area Under The Curve**) ROC (**Receiver Operating Characteristics**) curve. It is one of the most important evaluation metrics for checking any classification model's performance. It is also written as AUROC (**Area Under the Receiver Operating Characteristics**)

This blog aims to answer following questions:

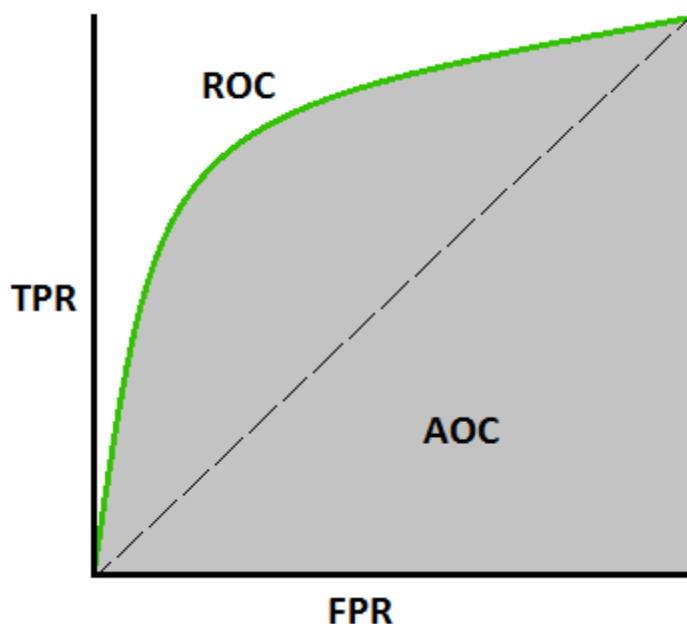
1. What is AUC - ROC Curve?
2. Defining terms used in AUC and ROC Curve.
3. How to speculate the performance of the model?
4. Relation between Sensitivity, Specificity, FPR and Threshold.

5. How to use AUC - ROC curve for multiclass model?

What is AUC – ROC Curve?

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.



AUC - ROC Curve

Defining terms used in AUC and ROC Curve.

TPR (True Positive Rate) / Recall /Sensitivity

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

Specificity

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

FPR

$$\begin{aligned}\text{FPR} &= 1 - \text{Specificity} \\ &= \frac{\text{FP}}{\text{TN} + \text{FP}}\end{aligned}$$

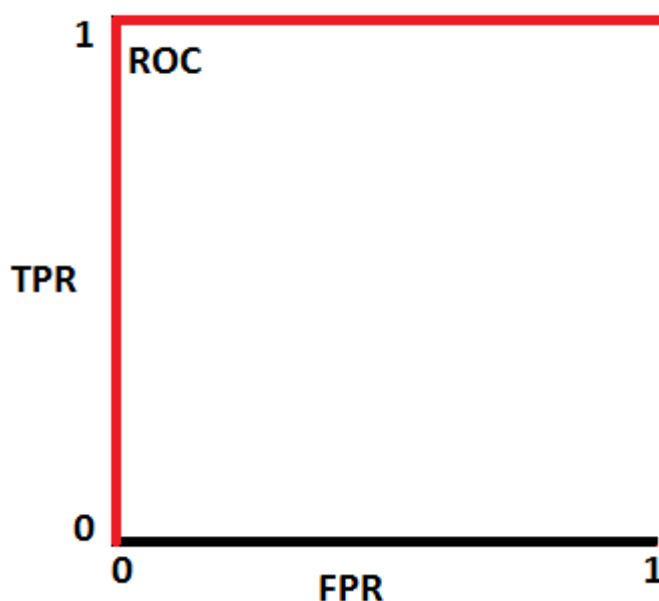
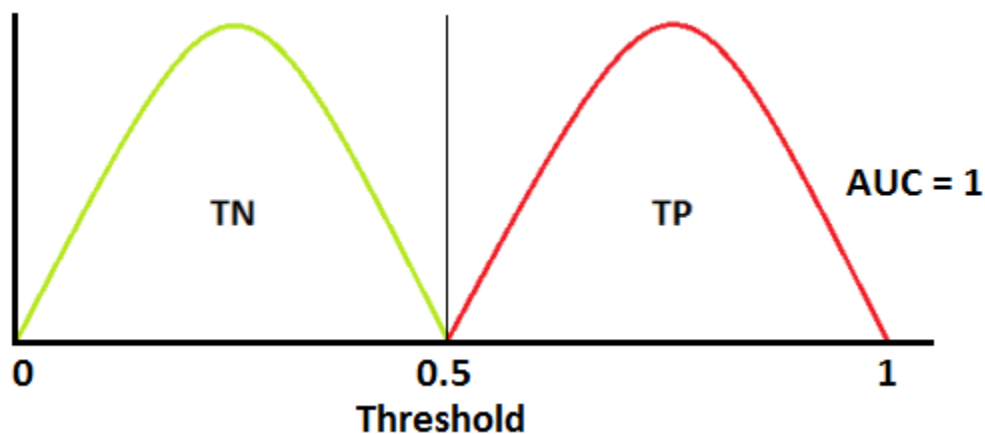
How to speculate the performance of the model?

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.

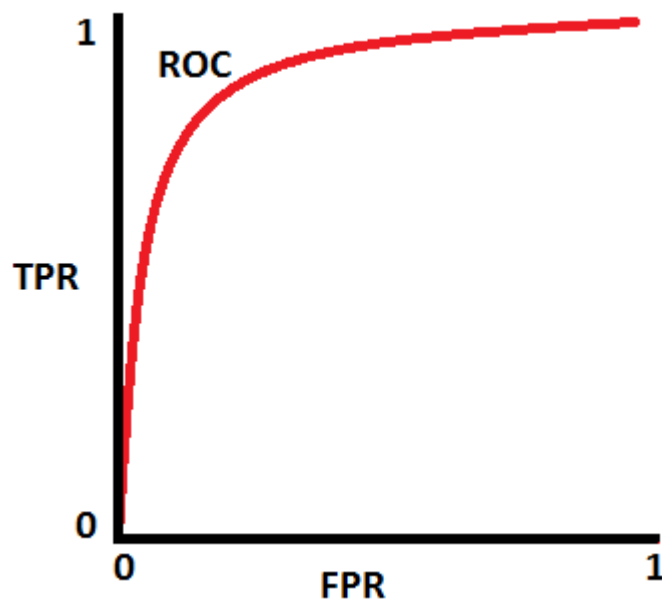
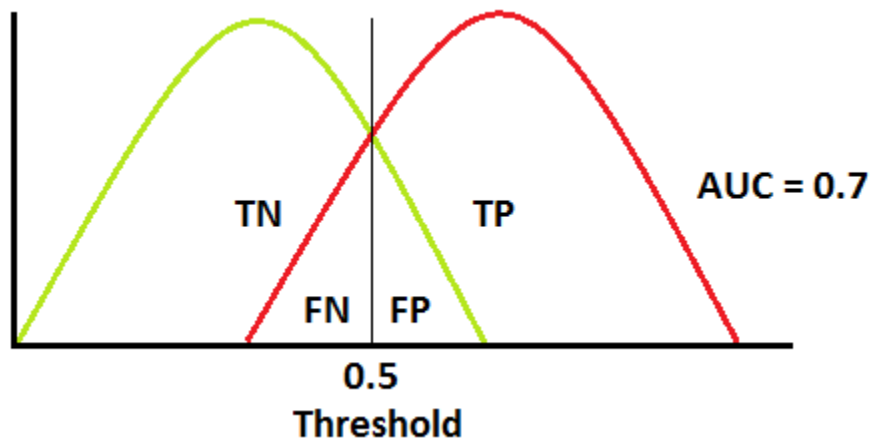
Let's interpret above statements.

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 green distribution curve is of negative class (patients with no disease).

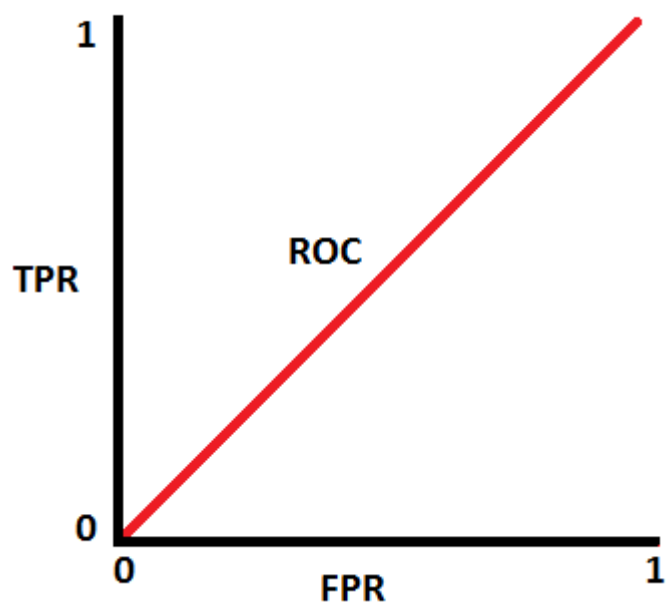
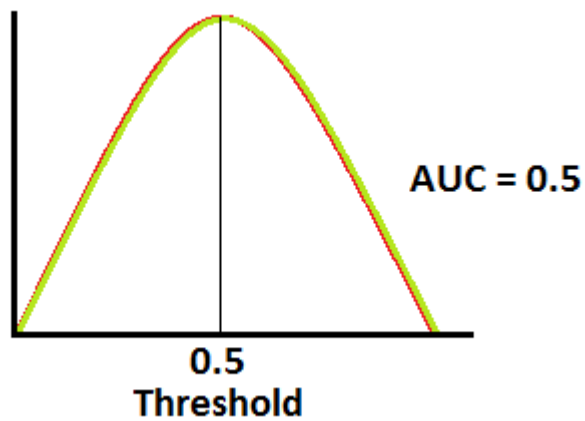


This is an ideal situation. When two curves don't overlap at all means model has an ideal measure of separability. It is perfectly able to distinguish between positive class and negative class.

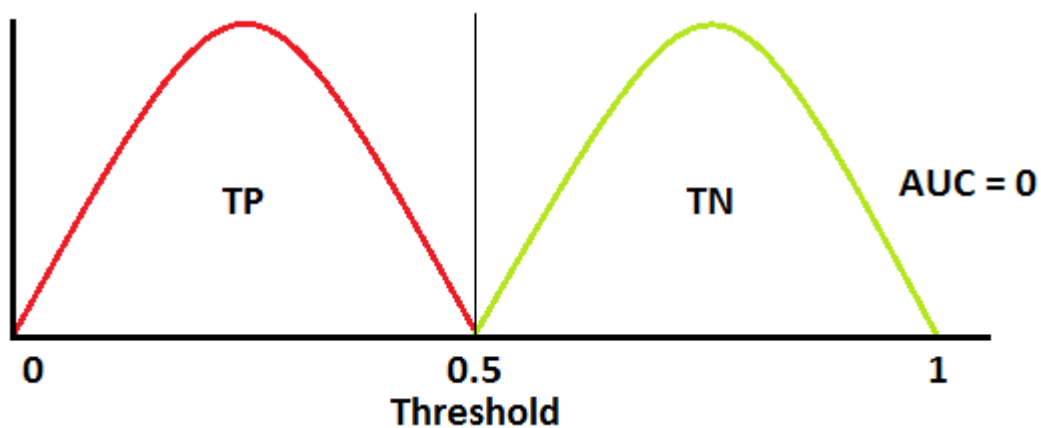


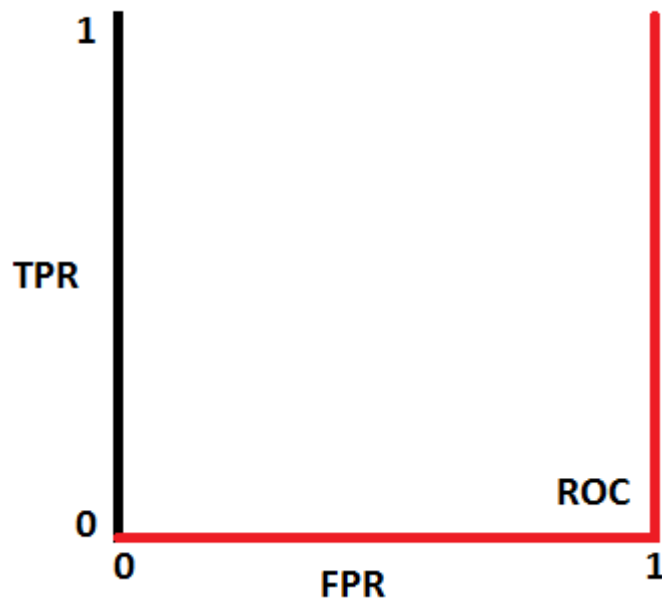
[Image 8 and 9] (Image courtesy: [My Photoshopped Collection](#))

When two distributions overlap, we introduce type 1 and type 2 error. 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.



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.

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 \uparrow , Specificity \downarrow and
Sensitivity \downarrow , Specificity \uparrow

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 $1 - \text{specificity}$. So when we increase TPR, FPR also increases and vice versa.

TPR , FPR  and TPR , FPR 

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

In multi-class model, we can plot N number of AUC ROC Curves for N number classes using 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 a third one of Z classified against Y and X.

Understanding ROC Curves with Python

By Guest Contributor • February 25, 2019 • [2 Comments](#)

In the current age where Data Science / AI is booming, it is important to understand how Machine Learning is used in the industry to solve complex business problems. In order to select which Machine Learning model should be used in production, a selection metric is chosen upon which different machine learning models are scored.

One of the most commonly used metrics nowadays is [AUC-ROC](#) (Area Under Curve - Receiver Operating Characteristics) curve. ROC curves are pretty easy to understand and evaluate once there is a good understanding of confusion matrix and different kinds of errors.

In this article, I will explain the following topics:

- Introduction to confusion matrix and different statistic computed on it
- Definitions of TP, FN, TN, FP

- Type 1 and Type 2 errors
- Statistics computed from Recall, Precision, F-Score
- Introduction to AUC ROC Curve
- Different scenarios with ROC Curve and Model Selection
- Example of ROC Curve with Python

Introduction to Confusion Matrix

In order to showcase the predicted and actual class labels from the Machine Learning models, the confusion matrix is used. Let us take an example of a binary class classification problem.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

The class labeled 1 is the positive class in our example. The class labeled as 0 is the negative class here. As we can see, the Positive and Negative Actual Values are represented as columns, while the Predicted Values are shown as the rows.

Definitions of TP, FP, TN, and FN

Let us understand the terminologies, which we are going to use very often in the understanding of ROC Curves as well:

- TP = True Positive – The model predicted the positive class correctly, to be a positive class.
- FP = False Positive – The model predicted the negative class incorrectly, to be a positive class.

- FN = False Negative – The model predicted the positive class incorrectly, to be the negative class.
- TN = True Negative – The model predicted the negative class correctly, to be the negative class.

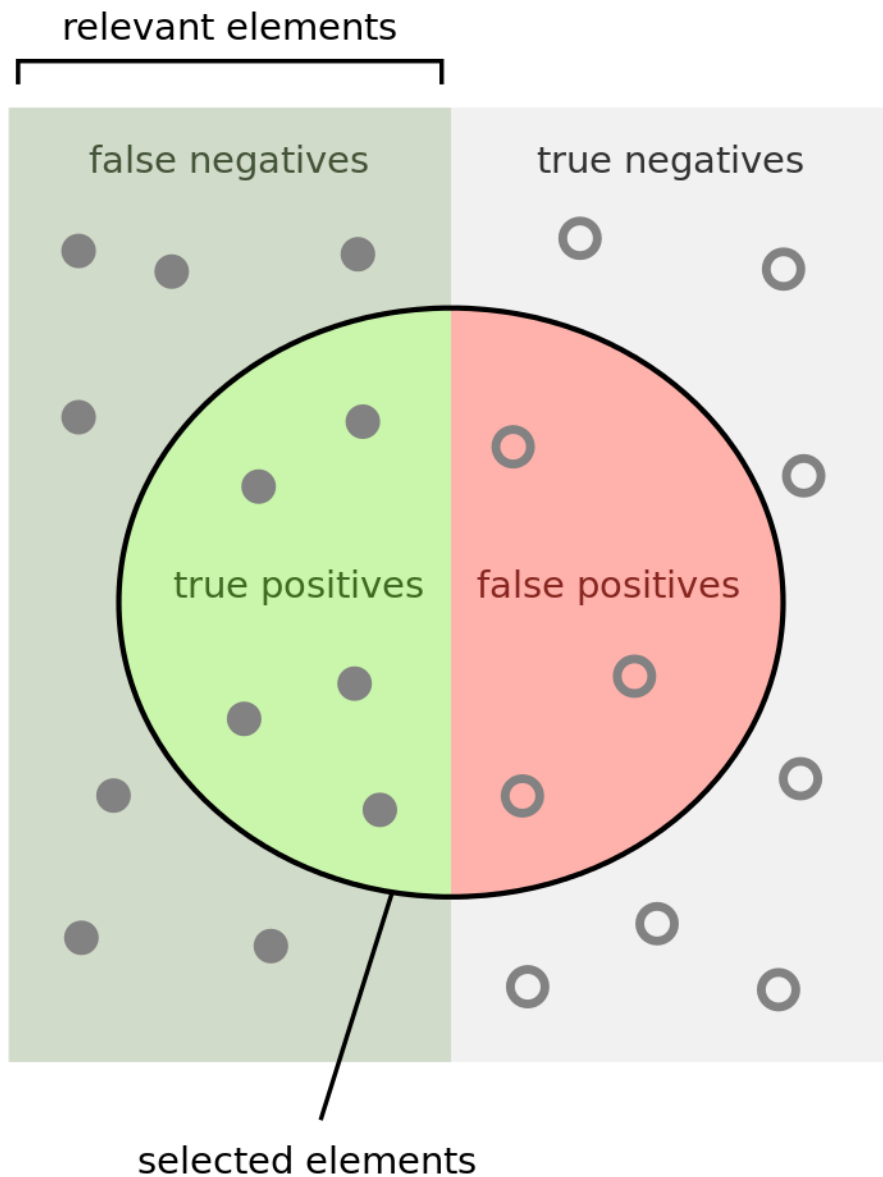
Type 1 and Type 2 Errors

There are two types of errors that can be identified here:

- Type 1 Error: The model predicted the instance to be a Positive class, but it is incorrect. This is False Positive (FP).
- Type 2 Error: The model predicted the instance to be the Negative class, but is it incorrect. This is False Negative (FN).

Statistics Computed from Confusion Matrix

In order to evaluate the model, some basic facts/statistics from the representation of the confusion matrix are calculated.



How many selected
items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant
items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Source: <https://commons.wikimedia.org/wiki/File:Precisionrecall.svg>

Recall: Out of all the positive classes, how many instances were identified correctly.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Precision: Out of all the predicted positive instances, how many were predicted correctly.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

F-Score: From Precision and Recall, F-Measure is computed and used as metrics sometimes. F – Measure is nothing but the harmonic mean of Precision and Recall.

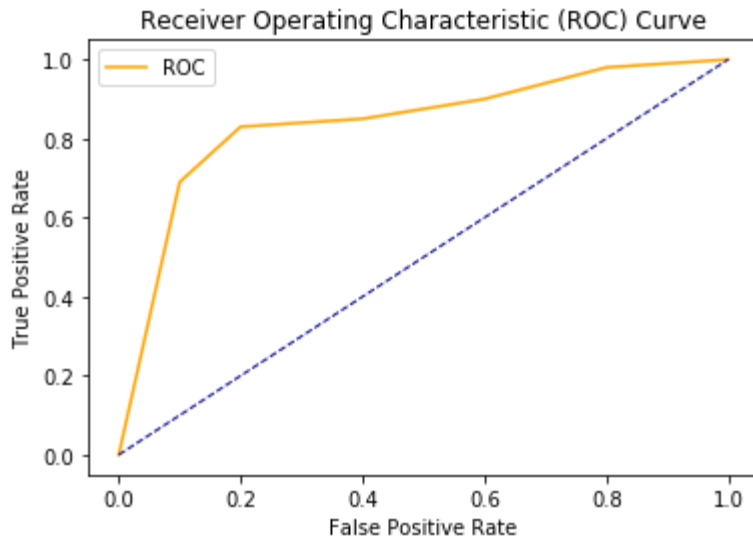
$$\text{F-Score} = (2 * \text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

Introduction to AUC - ROC Curve

(Area Under the curve- Receiver Operating Characteristics)

AUC–ROC curve is the model selection metric for bi–multi class classification problem. ROC is a probability curve for different classes. ROC tells us how good the model is for distinguishing the given classes, in terms of the predicted probability.

A typical ROC curve has False Positive Rate (FPR) on the X-axis and True Positive Rate (TPR) on the Y-axis.

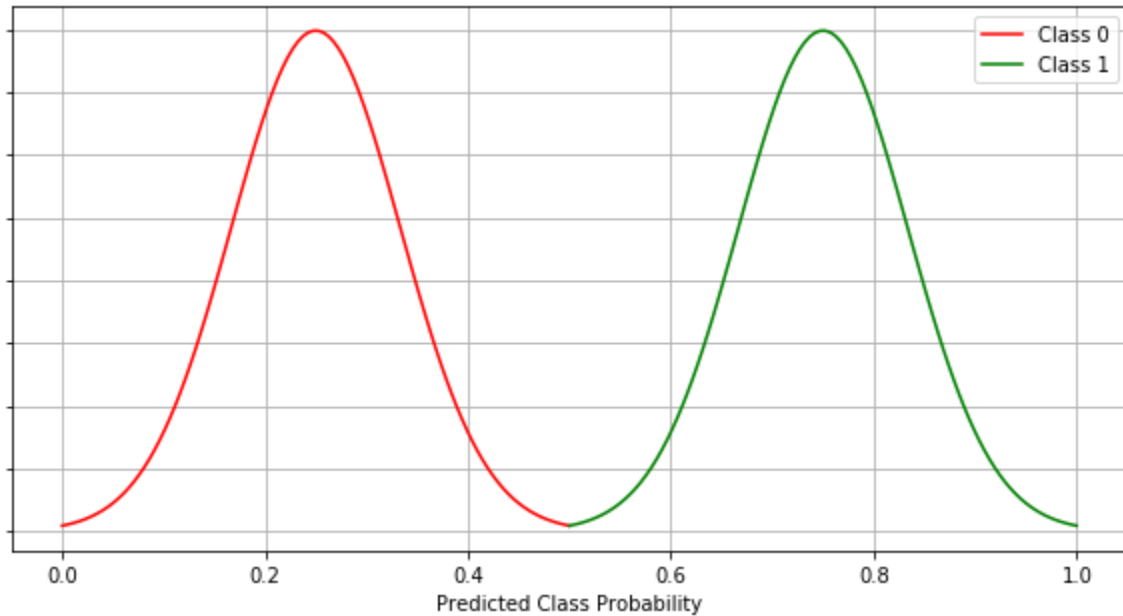


The area covered by the curve is the area between the orange line (ROC) and the axis. This area covered is AUC. The bigger the area covered, the better the machine learning models is at distinguishing the given classes. Ideal value for AUC is 1.

Different Scenarios with ROC Curve and Model Selection

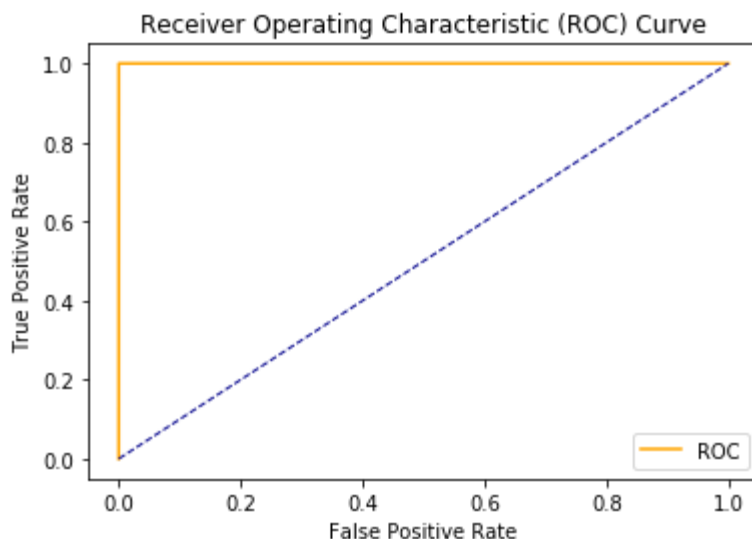
Scenario #1 (Best Case Scenario)

For any classification model, the best scenario is when there is a clear distinction between the two / all the classes.



The graph above shows the Predicted Class Probability for both classes 0 and 1. The threshold is 0.5 which means, if the predicted probability of the class for an instance is less than 0.5, that instance is predicted to be an instance of class 0. If the probability of the class for an instance is equal or greater than 0.5, the instance is classified as the instance of class 1.

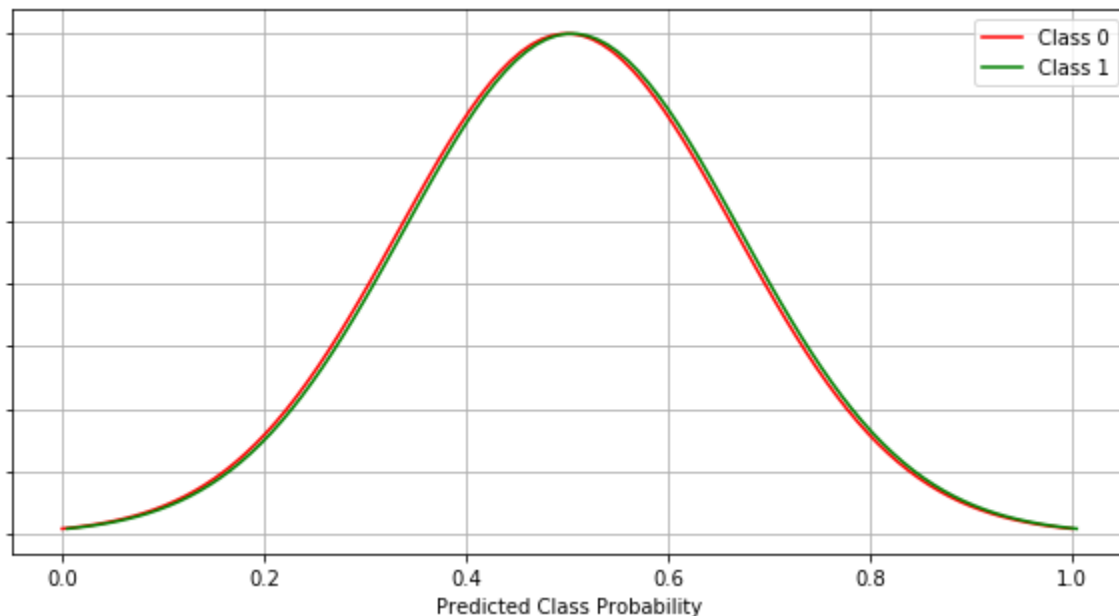
The AUC-ROC curve for this case is as below.



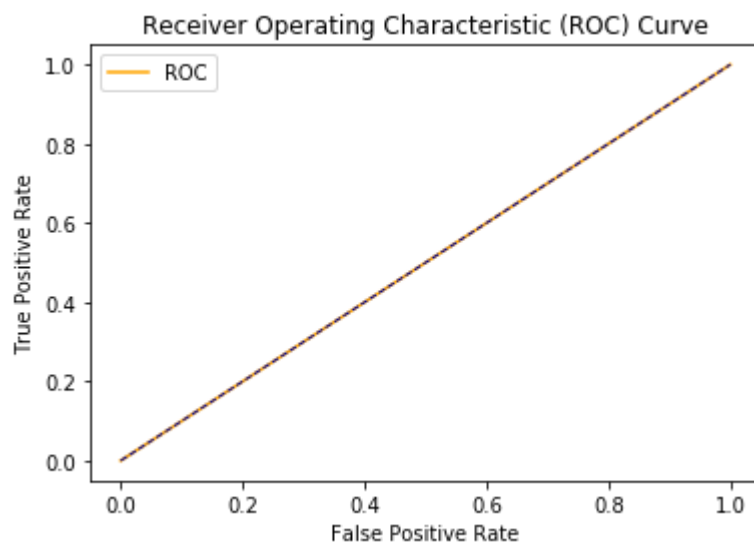
As we can see here, we have a clear distinction between the two classes as a result, we have the AUC of 1. The maximum area between ROC curve and base line is achieved here.

Scenario #2 (Random Guess)

In the event where both the class distribution simply mimic each other, AUC is 0.5. In other words, our model is 50% accurate for instances and their classification. The model has no discrimination capabilities at all in this case.



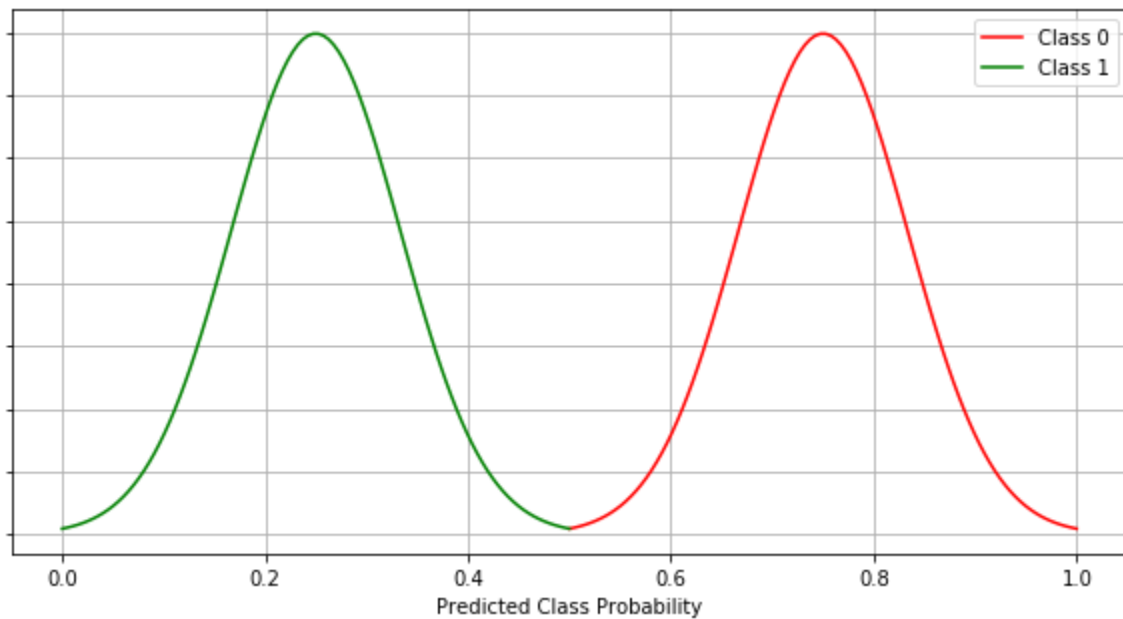
We can see there is no clear discrimination between the two classes.



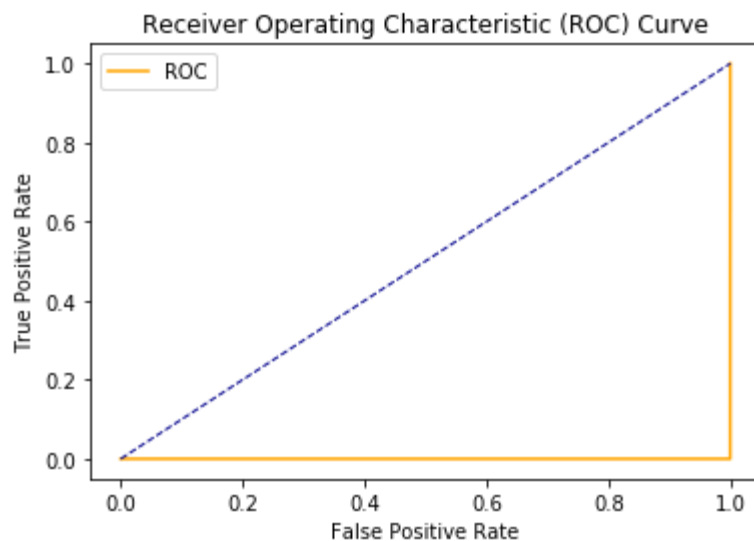
It is evident from the ROC AUC curve diagram, that the area between ROC and the axis is 0.5. This is still not the worst model but it makes a random guess, much like a human would do.

Scenario #3 (Worst Case Scenario)

If the model completely misclassifies the classes, it is the worst case.



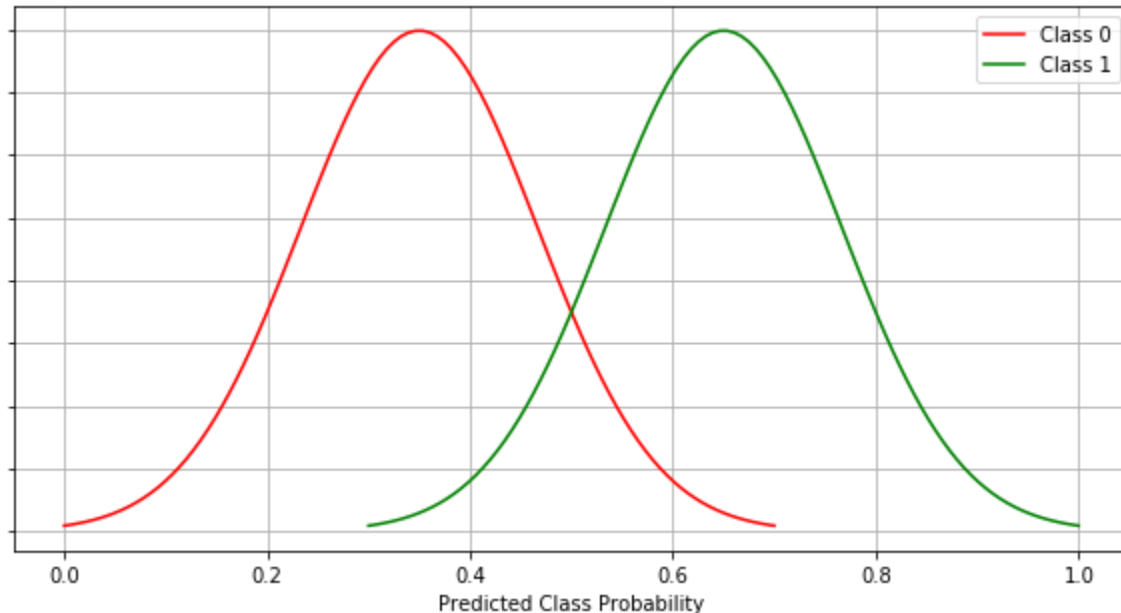
Completely opposite of the best case scenario (scenario #1), in this case, all the instances of class 1 are misclassified as class 0 and all the instances of class 0 are misclassified as class 1.



As a result, we get AUC to be 0, which is the worst case scenario.

Scenario #4 (Industry / Norm Scenario)

In a usual industry scenario, best cases are never observed. We never get a clear distinction between the two classes.



In this case, as observed, we have some overlapping and that introduces Type 1 and Type 2 errors to the model prediction. In this case we get AUC to be somewhere between 0.5 and 1.

Example with Python

Let us see an example of ROC Curves with some data and a classifier in action!

Step 1: Import libraries

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# roc curve and auc score

from sklearn.datasets import make_classification
```

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import roc_curve

from sklearn.metrics import roc_auc_score
```

Step 2: Defining a python function to plot the ROC curves.

```
def plot_roc_curve(fpr, tpr):

    plt.plot(fpr, tpr, color='orange', label='ROC')

    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')

    plt.xlabel('False Positive Rate')

    plt.ylabel('True Positive Rate')

    plt.title('Receiver Operating Characteristic (ROC) Curve')

    plt.legend()

    plt.show()
```

Step 3: Generate sample data.

```
data_X, class_label = make_classification(n_samples=1000, n_classes=2, weights=[1,
1], random_state=1)
```

Step 4: Split the data into train and test sub-datasets.

```
trainX, testX, trainy, testy = train_test_split(data_X, class_label, test_size=0.3
, random_state=1)
```

Step 5: Fit a model on the train data.

```
model = RandomForestClassifier()

model.fit(trainX, trainy)
```

Step 6: Predict probabilities for the test data.

```
probs = model.predict_proba(testX)
```

Step 7: Keep Probabilities of the positive class only.

```
probs = probs[:, 1]
```

Step 8: Compute the AUC Score.

```
auc = roc_auc_score(testy, probs)
```

```
print('AUC: %.2f' % auc)
```

Output:

```
AUC: 0.95
```

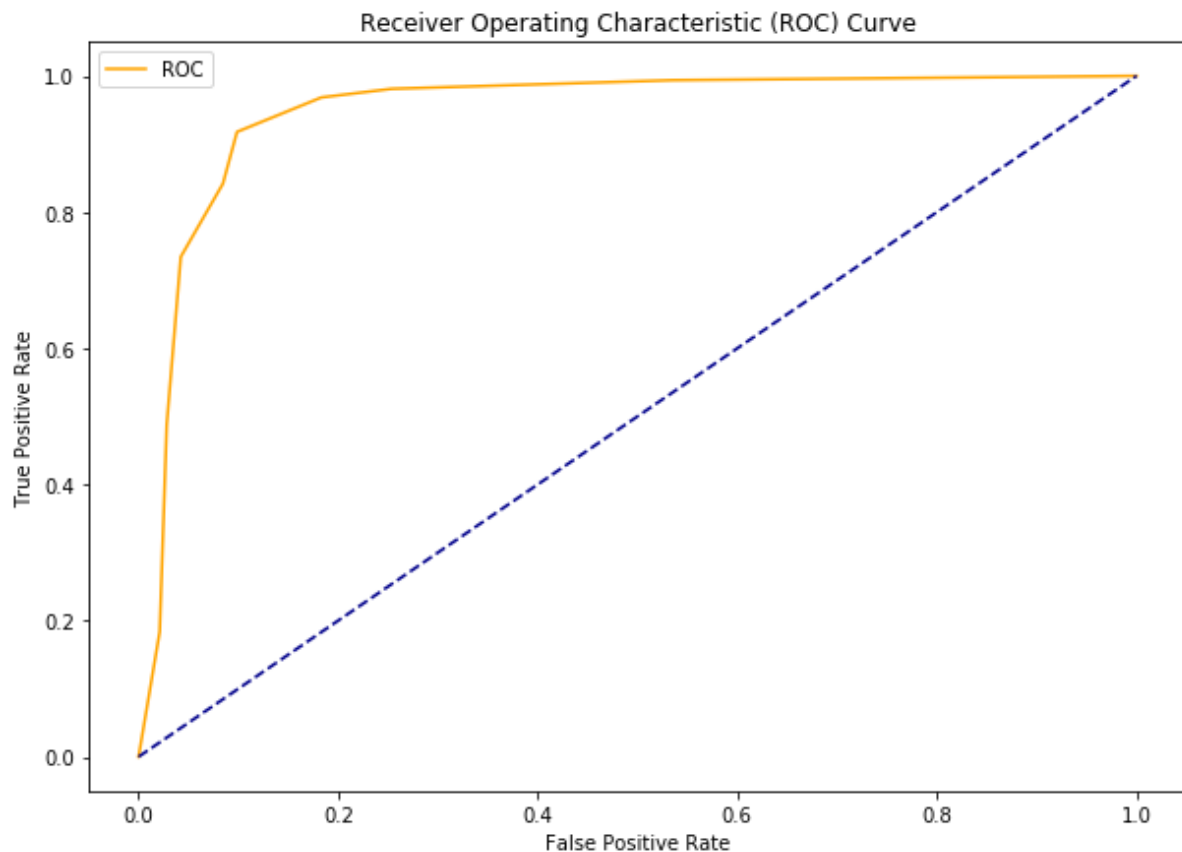
Step 9: Get the ROC Curve.

```
fpr, tpr, thresholds = roc_curve(testy, probs)
```

Step 10: Plot ROC Curve using our defined function

```
plot_roc_curve(fpr, tpr)
```

Output:



Conclusion

AUC-ROC curve is one of the most commonly used metrics to evaluate the performance of machine learning algorithms particularly in the cases where we have imbalanced datasets.

CAP curve is more robust method

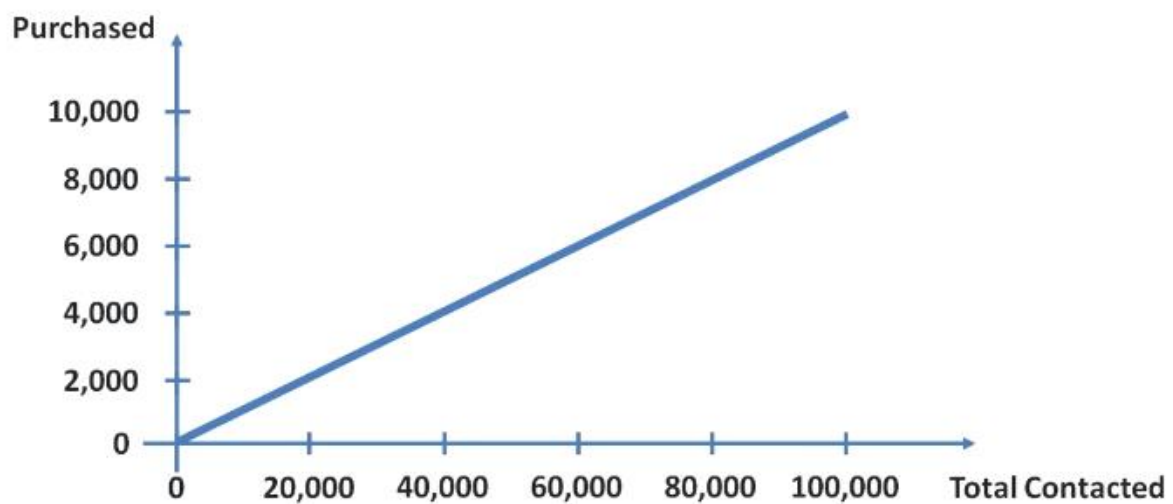
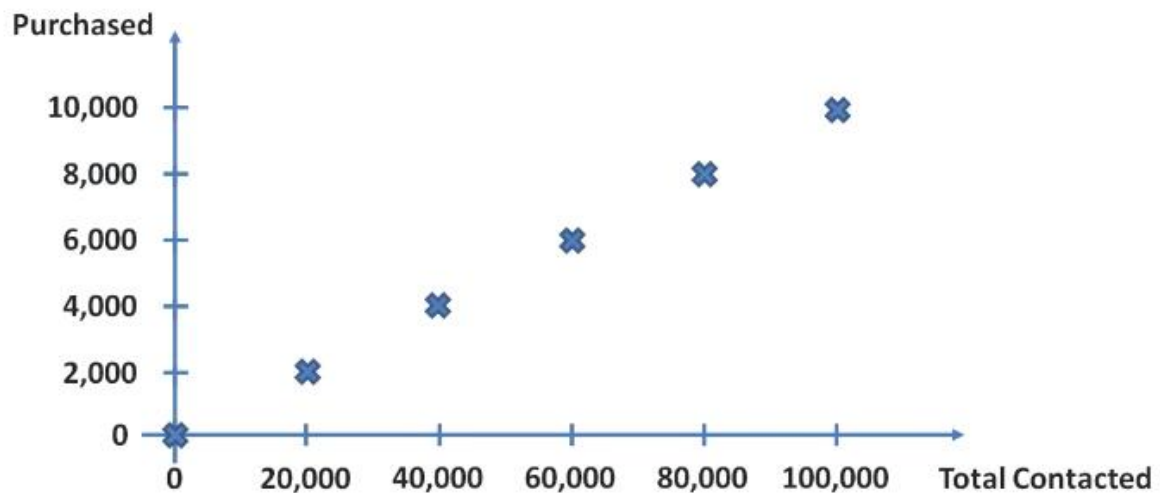
(Cumulative Accuracy Profile)

Suppose you have contacted 100000 people and experience says that around 10% respond and buy product

So 10000 people

If we send it to 0 people then response is 0

So if we randomly select then this is the scenario



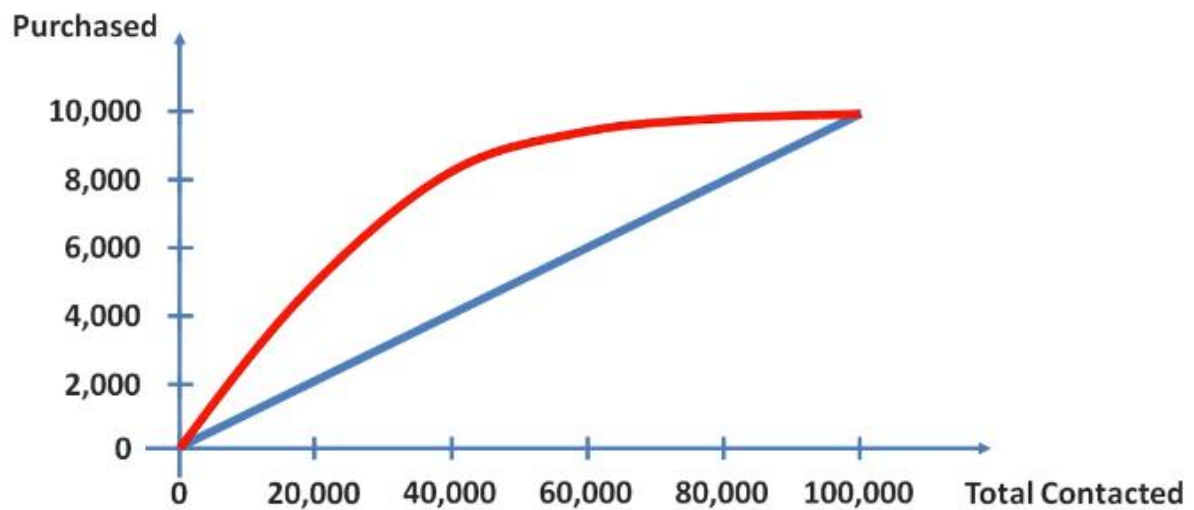
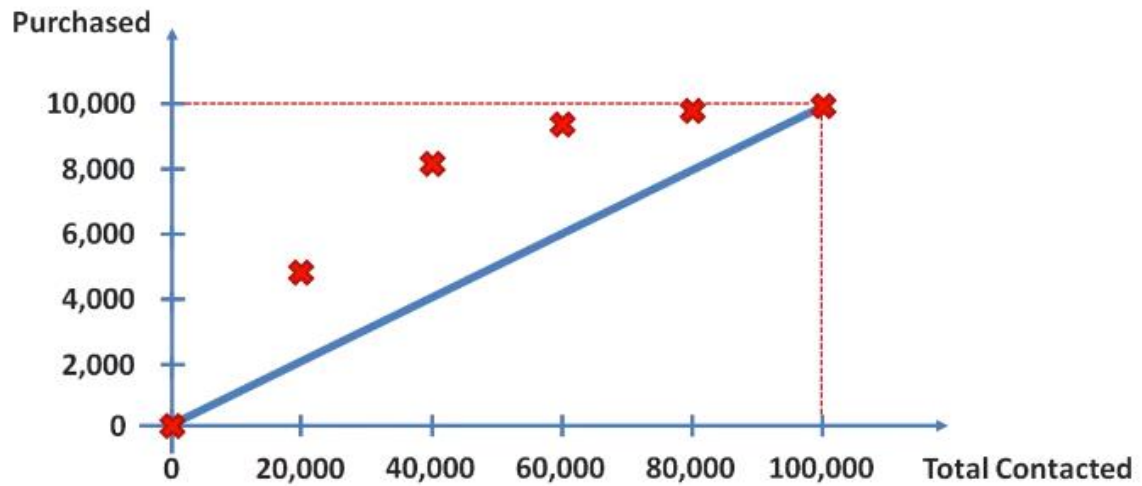
What if we want to improve it? Send the invitation to selected people. So send it only to pick and choose to send and build model for that

We may use classification model

Based on certain criteria based on few observations. May be female who has red color credit card

Male with high salary

So may be performance may be

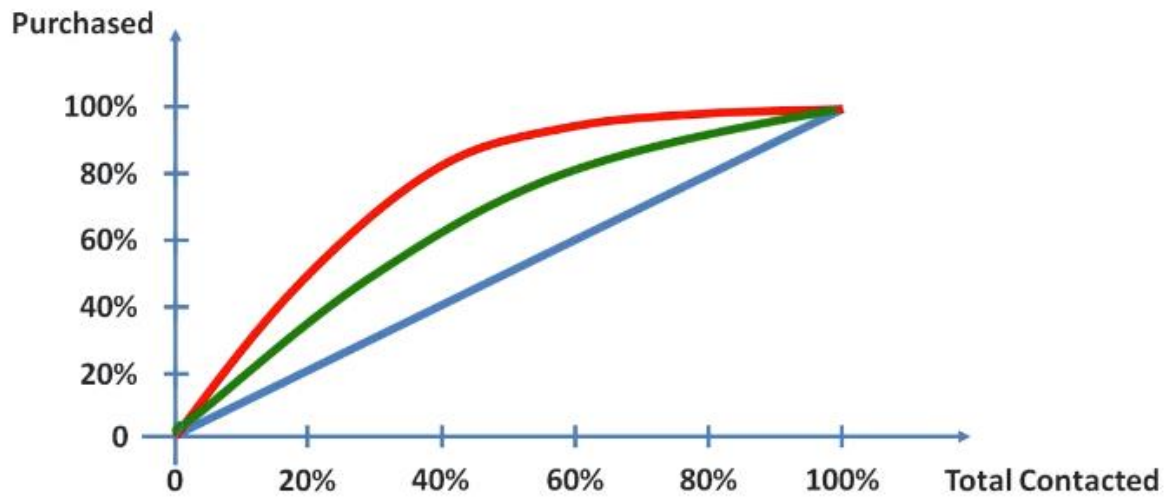


This is cap curve

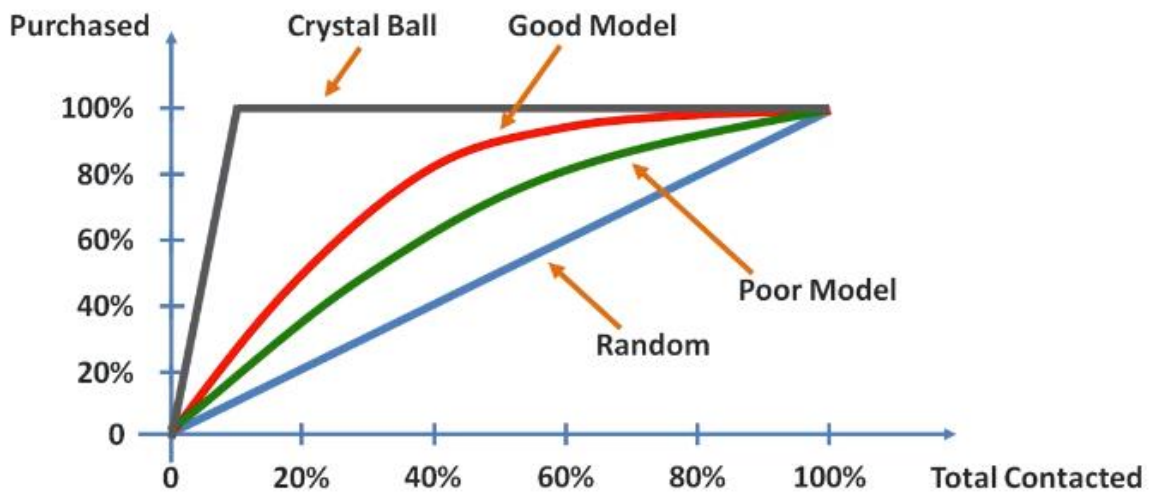
So if area between blue line is more then model is good.

If area is small then not that good.

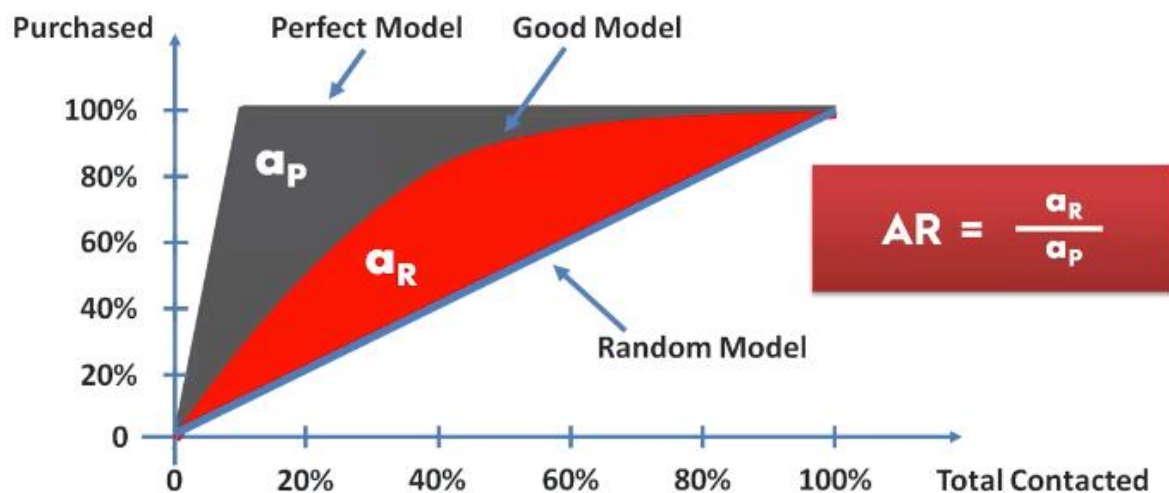
If we change these values to percentages and run another model and we get green line



So you may use this curve

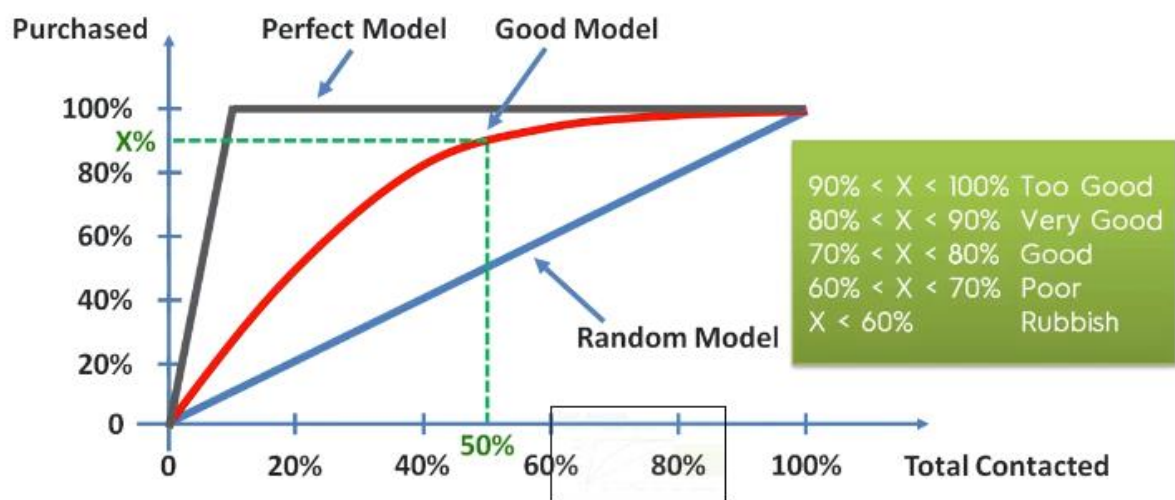


Crystal curve indicates every customer we selected, purchased the product



This is calculation but predicting by visualization may be difficult

So find value of y is x is 50%



When it is in 90% it is better to check is any unwanted independent variable is there and is it overfitting?

Conclusion of Part 3 - Classification

In this Part 3 you learned about 7 classification models. Like for Part 2 - Regression, that's quite a lot so you might be asking yourself the same questions as before:

1. What are the pros and cons of each model ?

Classification

Classification Model	Pros	Cons
Logistic Regression	Probabilistic approach, gives informations about statistical significance of features	The Logistic Regression Assumptions
K-NN	Simple to understand, fast and efficient	Need to choose the number of neighbours k
SVM	Performant, not biased by outliers, not sensitive to overfitting	Not appropriate for non linear problems, not the best choice for large number of features
Kernel SVM	High performance on nonlinear problems, not biased by outliers, not sensitive to overfitting	Not the best choice for large number of features, more complex
Naive Bayes	Efficient, not biased by outliers, works on nonlinear problems, probabilistic approach	Based on the assumption that features have same statistical relevance
Decision Tree Classification	Interpretability, no need for feature scaling, works on both linear / nonlinear problems	Poor results on too small datasets, overfitting can easily occur
Random Forest Classification	Powerful and accurate, good performance on many problems, including non linear	No interpretability, overfitting can easily occur, need to choose the number of trees

2. How can I improve each of these models ?

2. How do I know which model to choose for my problem ?

Same as for regression models, you first need to figure out whether your problem is linear or non linear.

Then:

If your problem is linear, you should go for Logistic Regression or SVM.

If your problem is non linear, you should go for K-NN, Naive Bayes, Decision Tree or Random Forest.

Then which one should you choose in each case ? You will learn that in Use k-Fold Cross Validation.

Then from a business point of view, you would rather use:

- Logistic Regression or Naive Bayes when you want to rank your predictions by their probability.

For example if you want to rank your customers from the highest probability that they buy a certain product, to the lowest probability. Eventually that allows you to target your marketing campaigns.

And of course for this type of business problem, you should use Logistic Regression if your problem is linear, and Naive Bayes if your problem is non linear.

- SVM when you want to predict to which segment your customers belong to. Segments can be any kind of segments, for example some market segments you identified earlier with clustering.

- Decision Tree when you want to have clear interpretation of your model results,

- Random Forest when you are just looking for high performance with less need for interpretation.

3. How can I improve each of these models ?

Using grid search find optimized values of parameter for Parameter Tuning, that will allow you to improve the performance of your models, by tuning them. You probably already noticed that each model is composed of two types of parameters:

- the parameters that are learnt, for example the coefficients in Linear Regression,
- the hyperparameters that the parameter that we choose e.g kernel, random state, criteria epsilon etc.

The hyperparameters are the parameters that are not learnt and that are fixed values inside the model equations. For example, the regularization parameter λ or the penalty parameter C are hyperparameters. So far we used the default value of these hyperparameters, and we haven't searched for their optimal value so that your model reaches even higher performance. Finding their optimal value is exactly what Parameter Tuning is about. So for those of you already interested in improving your model performance and doing some parameter tuning