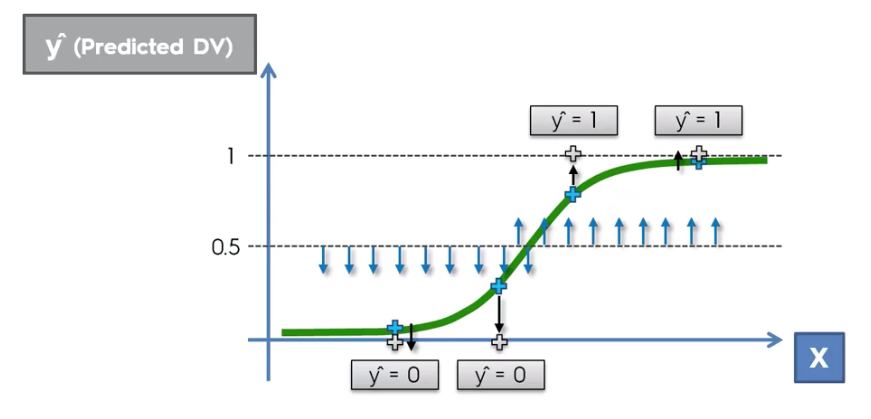
Chapter 3 : part 8

**Evaluating Classification Models Performance**

False Positives, False Negatives, Confusion Matrix, Accuracy Paradox, CAP Curve, CAP Curve Analysis

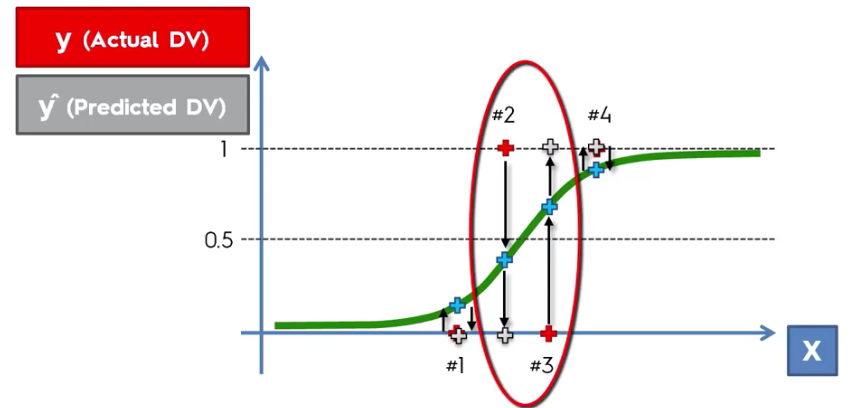
**3.8.1 False Positives - False Negatives**



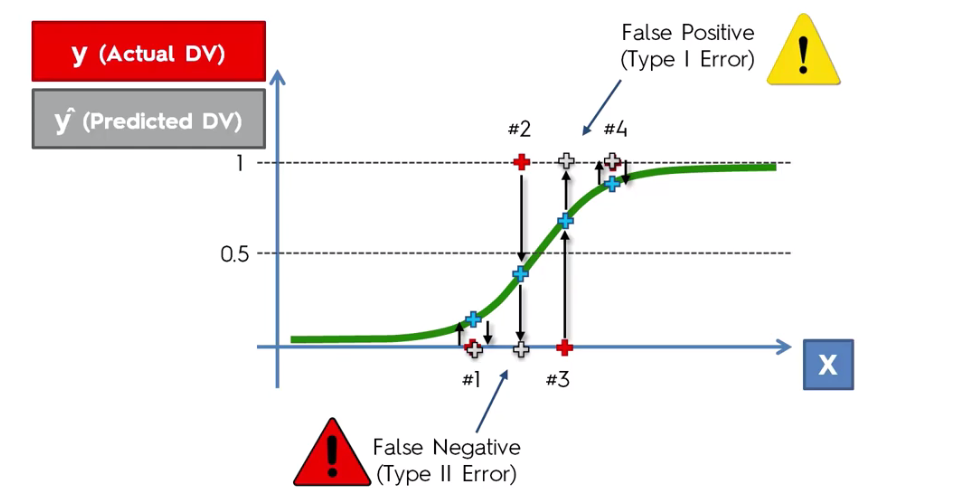
* In our Logistic Regression Function, in terms of the predicted value for the dependent variable (***y***), we agreed that anything *below* the ***50%*** line will be *projected* *downwards* onto the ***0*** horizontal line (***y=0***) and anything *above* the ***50%*** line will be *projected* *upwards* onto the ***100*** percent horizontal line (***y=1***) and that allowed us to turn probabilities into actual predictions. Either ***Yes*** or ***No***.
* Now, let's just pick out *four* *values* that we really know they *exist* in our *data* *set*. And we use them to create this Logistic Regression. And let's do the same thing with them.
* Here the actual values of these four observation is Red colored, we then project those in our Logistic curve.
* Now according to ***50% horizontal line***, #2 and #3 observations falls in the wrong "Horizontal line". So these two are the error of the model (therefore the logistic regression made two mistakes). The 1st observation #1 and 4th observation #4 are correct predications.

|  |  |
| --- | --- |
|  |  |

* False Positive (Type I error): The first kind of error is the mistaken rejection of a null hypothesis (mistaken rejection of an actually true null hypothesis) as the result of a test procedure. This kind of error is called a type I error (false positive) and is sometimes called an error of the first kind.
* It means that we said we predicted a positive outcome but it was false
* In terms of the courtroom example, a type I error corresponds to *convicting* an *innocent* *defendant*.
* The #3 point is predicted as "it will happen", but in reality this "actually not happened". This is kind of "Warning". ***Predicted*** as ***Positive*** but in ***reality*** they are ***Negative***, hence False-Positive.
* False Negative (Type II error): The second kind of error is the mistaken acceptance of the null hypothesis (mistaken acceptance of an actually false null hypothesis) as the result of a test procedure. This sort of error is called a type II error (false negative) and is also referred to as an error of the second kind.
* We predicted that there won't be an effect but the effect actually did occur
* In terms of the Courtroom example, a type II error corresponds to *acquitting* a *criminal*.
* The #2 point is predicted as "it won't happen", but in reality this "actually happened". This is kind of "fatal Error". ***Predicted*** as ***Negative*** but in ***reality*** they are ***Positive***, hence False-Negative.

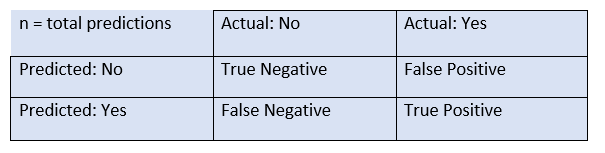


* False negative is a bit worse: Some people think, type 1 as less dangerous than type 2. False negative is a bit worse in my view, because once you say something's *not going to happen* but it *actually does happen* then you can't even be Prepared for it.



**3.8.2 Confusion Matrix**

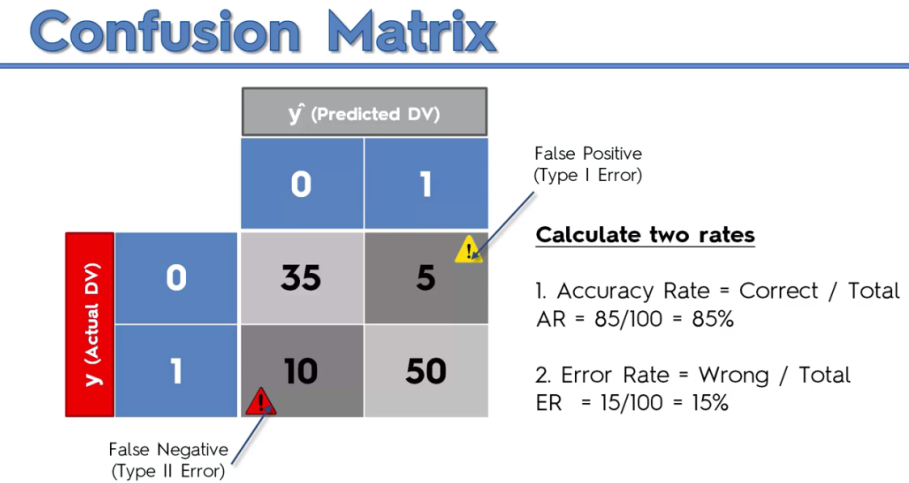
* CONFUSION MATRIX: The *confusion* *matrix* is a matrix used to determine the *performance* of the *classification* *models* for a given set of ***test-data***. It can *only be determined* if the true values for test data are known.
* Since it shows the errors in the model performance in the form of a matrix, hence also known as an ERROR MATRIX. Some features of Confusion matrix are given below:
* For the 2 prediction classes of classifiers, the matrix is of ***2\*2 table***, for 3 classes, it is ***3\*3 table***, and so on.
* The matrix is divided into 2D, that are predicted values and actual values along with the total number of predictions.
* Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations. It looks like the below table:



* The above table has the following cases:
* True Negative: Model has given ***prediction*** ***No***, and the real or actual value was also ***No***.
* True Positive: The model has ***predicted*** ***yes***, and the actual value was also ***true***.
* False Negative: The model has ***predicted*** ***no***, but the actual value was ***Yes***, it is also called as ***Type-II error***.
* False Positive: The model has ***predicted*** ***Yes***, but the actual value was ***No***. It is also called a ***Type-I error***.
* Need for Confusion Matrix in Machine learning:
* It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.
* It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error.
* With the help of the ***confusion*** ***matrix***, we can calculate the ***different*** ***parameters*** for the model, such as ***accuracy***, ***precision***, etc.
* Consider the following example: Suppose we are trying to create a model that ***can*** ***predict*** the result for the ***disease*** that is either a ***person*** ***has*** that disease or ***not***. So, the confusion matrix for this is given as:

|  |  |
| --- | --- |
| Confusion Matrix in Machine Learning |  True Negative: 65   True Positive: 24   False Negative: 8   False Positive: 3 |

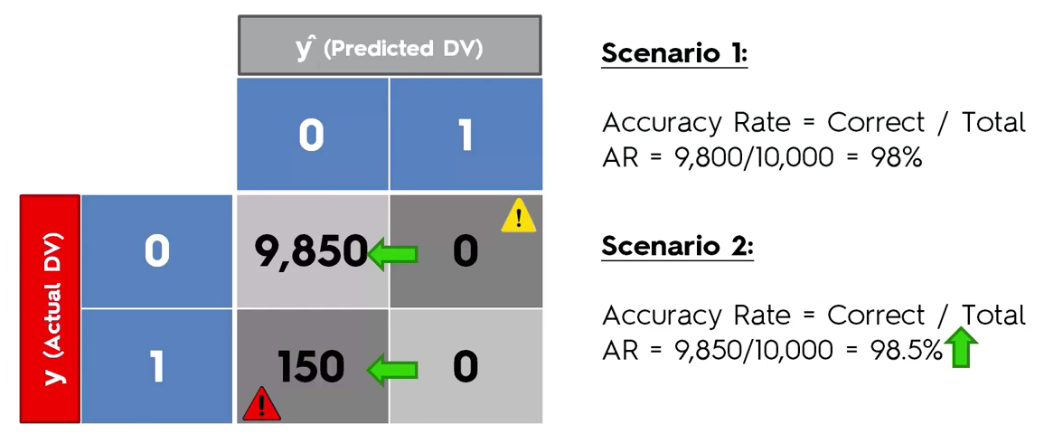
* The table is given for the two-class classifier, which has two predictions "Yes" and "NO." Here, *Yes* defines that patient has the *disease*, and *No* defines that patient does *not has that disease*.
* The classifier has made a total of 100 predictions. Out of 100 predictions, 89 are true predictions, and 11 are incorrect predictions.
* The ***model*** has given prediction "***yes***" for ***32*** times, and "***No***" for ***68*** times. Whereas the *actual* "*Yes*" was *27*, and actual "*No*" was *73* times.
* Classification Accuracy/ Accuracy Rate: It is one of the important parameters to determine the *accuracy of the classification* problems. It defines ***how often the model predicts the correct output***. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of Total predictions.
* In our example
* Misclassification rate/ Error Rate: It is also termed as Error rate, and it defines ***how often the model gives the wrong predictions***. The value of Error Rate can be calculated as the number of incorrect predictions to Total predictions.
* In our example



**3.8.3 Accuracy Paradox**

|  |  |
| --- | --- |
| * Here a confusion matrix of 10000 records in it. This model has made 150 TYPE-I errors and 50 TYPE-II errors. Now let's calculate the accuracy rate: * Now we abandon the model: We're going to tell the model to stop making predictions, which is going to abandon them all completely. And we're going to say that from now on our prediction is always ***0*** (i.e predict ***0*** only, not ***1***). We're always going to predict that the event is not going to occur. |  |

* So basically what will happen to the confusion matrix is these records will move from the right column to the left column and our new confusion matrix will look like this:



* The accuracy rises when we abandon our model. Now it is 98.5%. And as you can see what we did is we just completely stopped using a model but the accuracy rate went up.
* Now you're not applying any kind of logic into your decision making process. And your accuracy rate is going up so it's misleading you into a wrong conclusion that You Should Stop Using Models and this effect is called the ***Accuracy*** ***Paradox***.

**3.8.4 CAP Curve Analysis**

Lets say a store sells clothes and your store has a total of 100000 customers (horizontal axis). Whenever you send an offer like an e-mail to all 100000 customers approximately 10% of them respond and purchase the product. So I'm going to place 10000 (10% percent of the total) on the vertical axis.

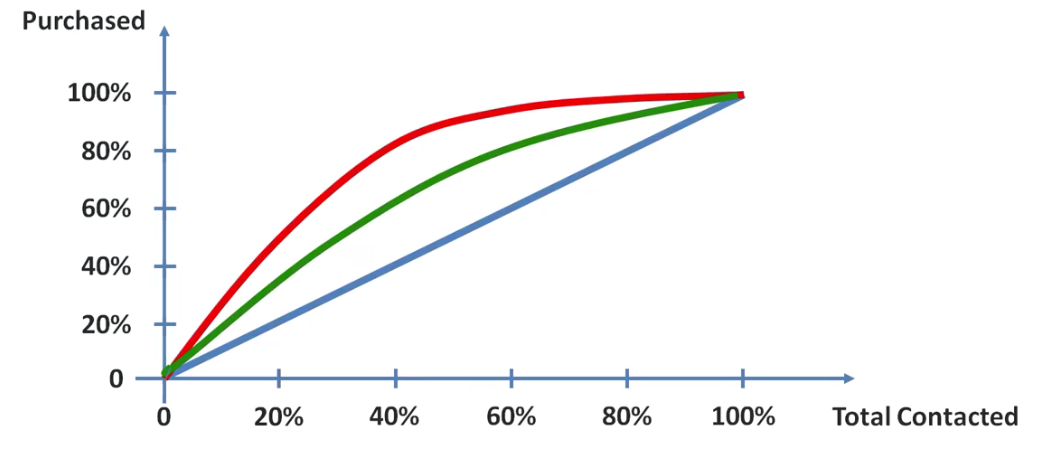
* We now draw an average line (for random values) for our *customer response rate*.

|  |  |
| --- | --- |
|  |  |

* So above represents the line for average response rate. Now the question is can we somehow improve this experience, can we get more customers to respond to offers when we send out our letters. Can we somehow ***target our customers more appropriately*** so to get a *better response rate* instead of sending out these offers *randomly*.
* Well to start off with let's build a model just like we did in the previous section. It will actually predict whether or not they will purchase the product. Because purchasing product is also a binary variable (yes or no). So we can build a logistic model.
* We can take a group of customers who purchased from our store, before we send out the offer. For example: A *male* or *female*. Which *country* were they. What *age* they were. Were they browsing on *mobile* or *computer*.
* So that we can take them into account and put them into a logistic regression and get a model. Which will help us to predict the likelihood of certain types of customers purchasing from the store (based on their characteristics).
* And once we've built this model how about we apply it to select customers we will send the offer.

|  |  |
| --- | --- |
|  |  |

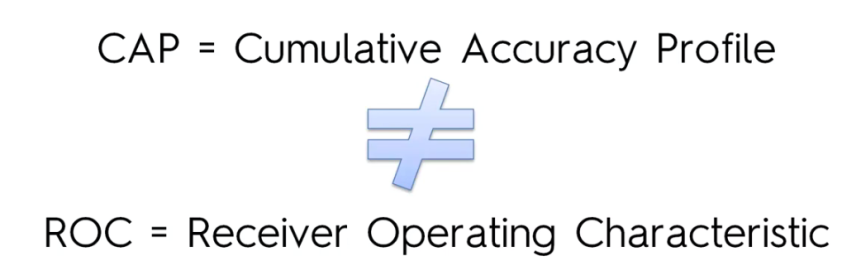
* Lets say Red curve represents the model. This Red line here is called the cumulative accuracy profile of your model. The better your model the ***larger*** will be the ***area*** between ***Red*** and ***Blue*** line.
* If your model is worse, this red line will be closer to the blue line (random line). And this is how the ***CAP-curve*** is normally represented.
* Let consider another model represented by Green curve. Now let's say we had *less access to independent variables* or we didn't see that *there's a multicollinearity* effect in a model or something else that went wrong. And that making this model worse.
* By plotting the CAP CURVES (Green and Red) you'll be able to compare models to each other and understand how much gain this is also sometimes called the Gain Chart.
* How much gain you get in each of these models compared to the random scenario or how much again you get additional gain you get from switching from one model to the next or from the Green one to the Red one for instance. This is how you're improving your hit ratio and therefore you're improving your return on investment.



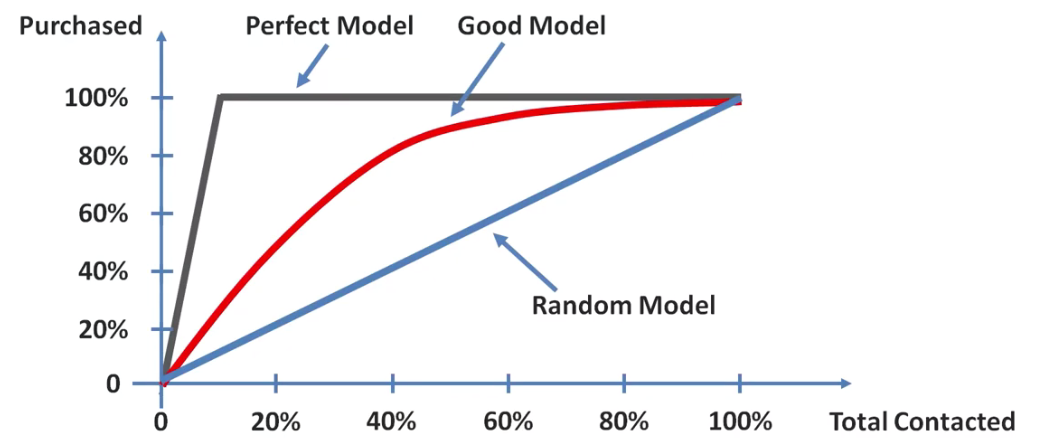
* Hence, the read model is better. The green line is a poor model (it's always is better than random but it's still not as good as the red one).

|  |  |
| --- | --- |
| * Ideal Line: And there's one more line, this line is the ***ideal*** ***line***. Something like, you had a crystal ball if you could predict exactly who is going to purchase and contact those people. This is what it would look like (the Gray line). * We know that only 10% of our customers ever purchase. Notice that the ***Red-Dotted vertical lin***e indicates exactly ***10%***. |  |

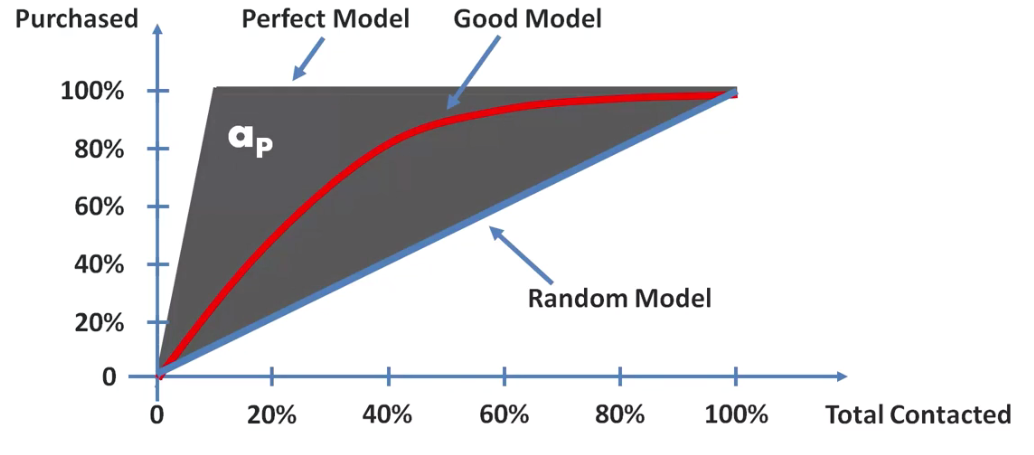
* CAP curve and ROC curve are not the same: Note that CAP means *Cumulative Accuracy Profile*. There is a ROC-curve which is *Receiver Operating Characteristic*.



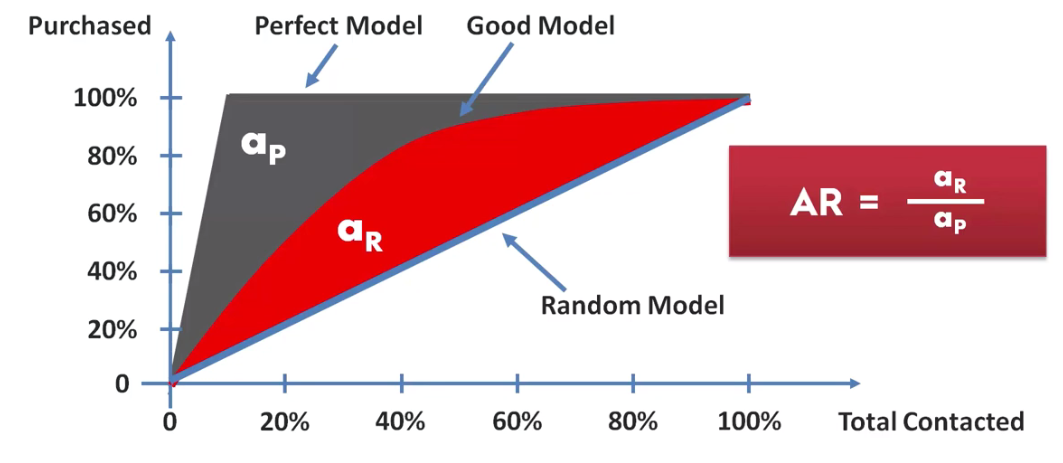
* How to Analyze: So now we have, three lines. One the CAP-Curve Red line), the Blue line is the random line, the Gray line which is the *perfect* *model*.



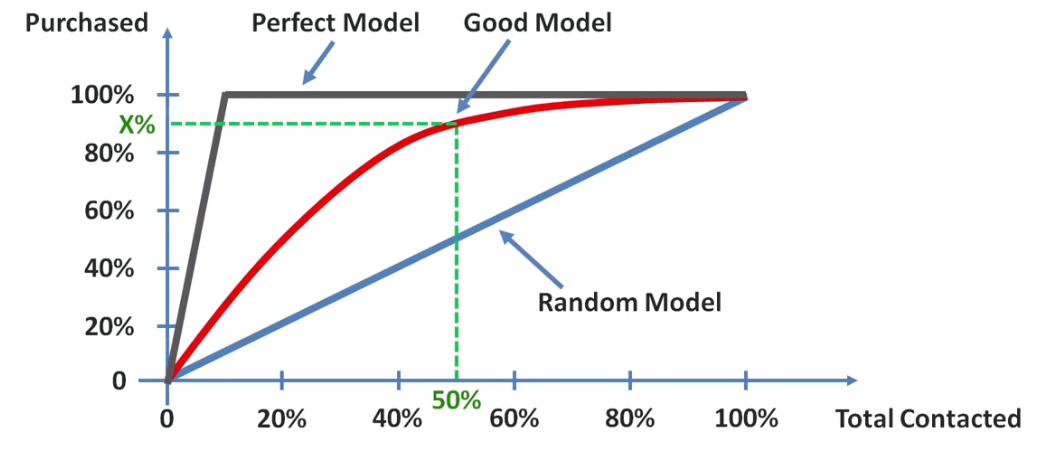
* The closer your *Red* line to the *Gray* line the *better* you model, the closer to the *Blue* line the *worse*. So how can we quantify this effect? Well there is a standard approach to Calculate The Accuracy Ratio and to calculate accuracy ratios: you take the ***area under the perfect model*** or the ***perfect line*** which is color in Gray here and it's called . Notice the following figure



* And then you need to take that ***area under the Red line*** which is colored in Red here which is and then you need to divide one by the other. This ratio is between 0 and 1.



* The closer this ratio is to 1 the better, the further it is away from 1 and close to 0 the worse your model is.
* Other way: Now let's get rid of areas and instead of looking at the area what you can do is look at the 50% line on the horizontal axis (X-axis) and look where it crosses your model-curve and then project that point on the vertical axis (Y-axis).



* And there are some rules for the X% on the Y-axis projected value:

***X < 60%*** : Rubbish model

***60% < X < 70%*** : Poor model

***70% < X < 80%*** : Good model

***80% < X < 90%*** : Very Good model

***90% < X < 100%*** : That's odd !! This model is so good !! Need to check that there is OVERFITTING.

* OVERFITTING is bad: So Remember the OVERFITTING. It could Ruin your model.