Lecture Notes

Logistic Regression

In the last module, you learnt **Linear Regression**, which is a supervised regression model. In other words, linear regression allows you to make predictions from labelled data, if the target (output) variable is numeric.

Hence, in this module, you moved to the next step, i.e., **Logistic Regression**. Logistic Regression is a supervised classification model. It allows you to make predictions from labelled data, if the target (output) variable is categorical.

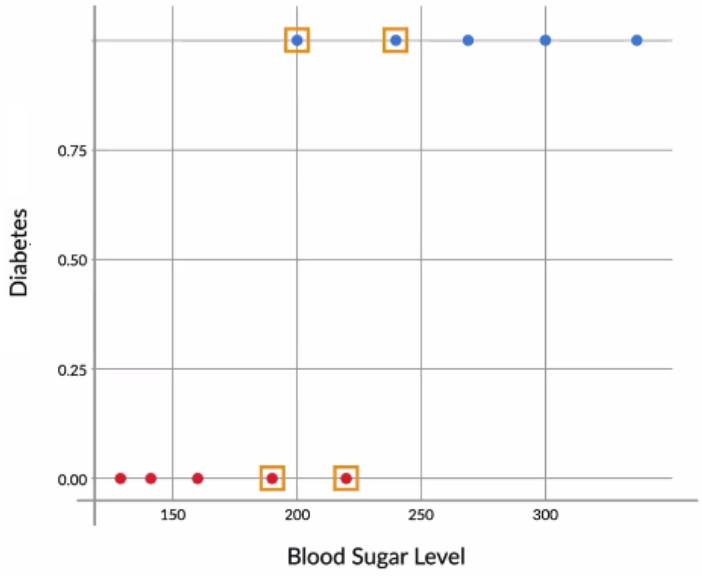
Binary Classification

You first learnt what a **binary classification** is. Basically, it is a classification problem in which the target variable has only 2 possible values, or in other words, two classes. Some examples of binary classification are –

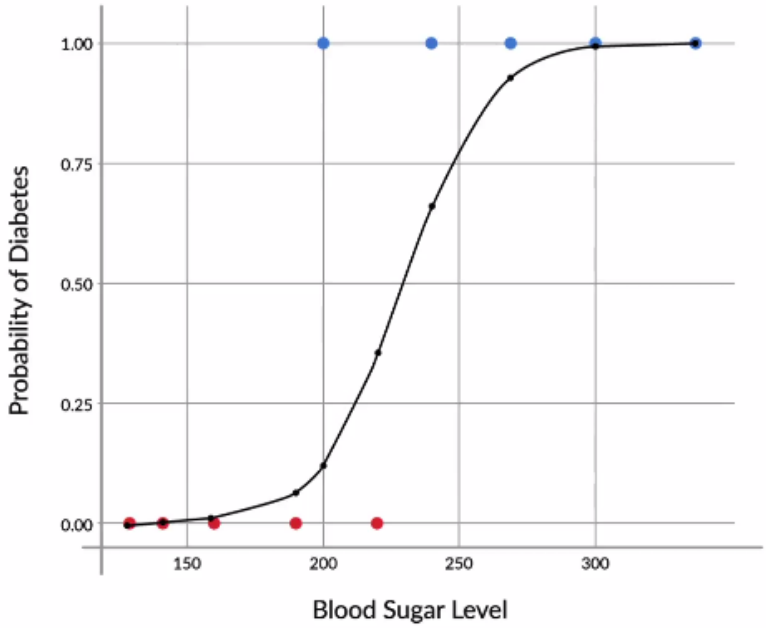
1. A bank wants to predict, based on some variables, whether a particular customer will default on a loan or not
2. A factory manager wants to predict, based on some variables, whether a particular machine will break down in the next month or not
3. Google's backend wants to predict, based on some variables, whether an incoming email is spam or not

You then saw an example which was discussed in detail, which is the **diabetes example**. Basically, in this example, you try to predict whether a person has diabetes or not, based on that person's blood sugar level.

You saw why a simple boundary decision approach does not work very well for this example. It would be too risky to decide the class blatantly on the basis of cutoff, as especially in the middle, the patients could basically belong to any class, diabetic or non-diabetic.



Hence, you learnt it is better, actually to talk in terms of **probability**. One such curve which can model the probability of diabetes very well, is the **sigmoid curve**.

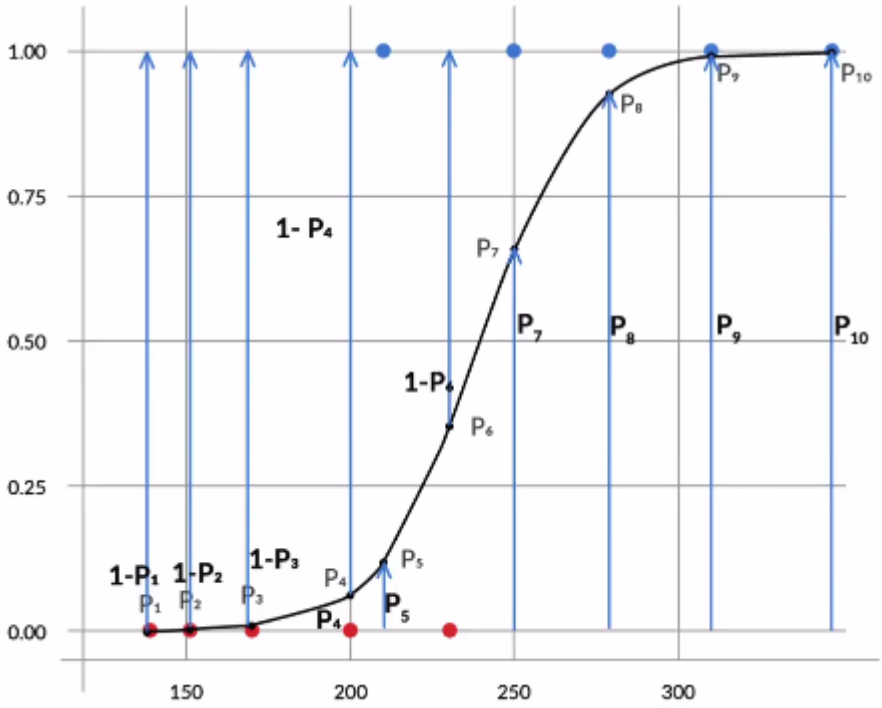


Its equation is given by the following expression -

$$P(\text{Diabetes}) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

Likelihood

The next step, just like linear regression, would be to find the **best fit curve**. Hence, you learnt that in order to find the best fit sigmoid curve, you need to vary β_0 and β_1 until you get the combination of beta values that maximises the **likelihood**. For the diabetes example, likelihood is given by the expression -



Likelihood = $(1-P_1)(1-P_2)(1-P_3)(1-P_4)(P_5)(1-P_6)(P_7)(P_8)(P_9)(P_{10})$

Generally, it is the product of -

[(1-P_i)(1-P_i) ------ for all non-diabetics --------] * [(P_i)(P_i) -------- for all diabetics -------]

This process, where you vary the betas, until you find the best fit curve for probability of diabetes, is called **logistic regression**.

Odds and Log Odds

Then, you saw a simpler way of interpreting the equation for logistic regression. You saw that the following **linearized** equation is much easier to interpret -

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x$$

The left hand side of this equation is what is called **log odds**. Basically, the **odds** of having diabetes (P/1-P), indicate how much more likely a person is to have diabetes than to not have it. For example, a person for whom the odds of having diabetes are equal to 3, is 3 times more likely to have diabetes than to not have it. In other words, P(Diabetes) = 3*P(No diabetes).

Also, you saw how odds vary with variation in x. Basically, with every linear increase in x, the increase in odds is **multiplicative**. For example, in the diabetes case, after every increase of 11.5 in the value of x, the odds get approximately doubled, i.e., increase by a multiplicative factor of around 2.

Multivariate Logistic Regression (Telecom Churn Example)

In this session, you learnt how to build a multivariate logistic regression model in R. The equation for **multivariate logistic regression** is basically just an extension of the univariate equation -

$$P = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\cdots)}}$$

The example used for building the multivariate model in R, was the **Telecom Churn Example**. Basically, you learnt how R can be used to decide the probability of a customer churning, based on the value of 21 predictor variables, like monthly charges, paperless billing, etc.

Multivariate Logistic Regression (Model Building)

First, the data was imported, which was present in 3 separate csv files. After creating a merged master dataset, one that contains all 21 variables, data preparation was done, which involved the following steps -

1. Missing value imputation (not covered in video)
2. Outlier treatment (not covered in video)
3. Standardising scales of continuous variables
4. Dummy variable creation for categorical variables

After all of this was done, a logistic regression model was built in R using the function **glm()**. This model contained all the variables, some of which had insignificant coefficients and for many of them, the coefficients were NA.

Multivariate Logistic Regression (Variable Selection)

Hence, some of these variables were removed using these two algorithms -

1. Stepwise variable selection based on AIC [using stepAIC()]
2. Backward variable selection based on VIF and p value

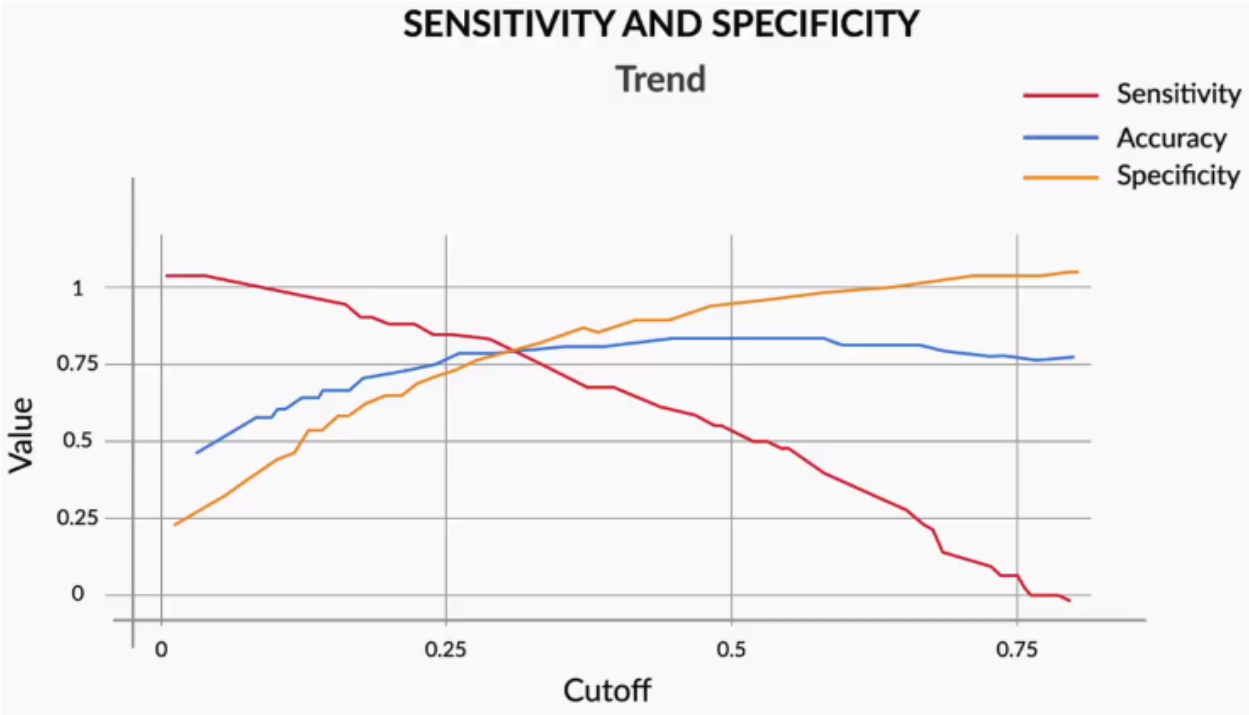
After this whole process was completed, you ended up with 10 variables, all of which were significant. Also, VIF for any variable was not very high. Thus, this model was declared as the final model.

Accuracy, Sensitivity and Specificity

Then, you learnt how the model built in earlier sessions can be evaluated, i.e., checked to see how good its performance is.

So first, classes were assigned to all 2110 customers in the test data set. For this, a probability cutoff of 0.5 was used. The model thus made, was very accurate (Accuracy = 81%), but it had a very low sensitivity (53%). Thus, a different cutoff was tried out, i.e., 0.4, which resulted in a model with slightly lower accuracy (79%), but a much better sensitivity (65%). Hence, you were told that you should not just blindly use 0.5 as the cutoff for probability, every time you make a model. Business understanding must be applied. Here, that means playing around with the cutoff, until you get the most useful model.

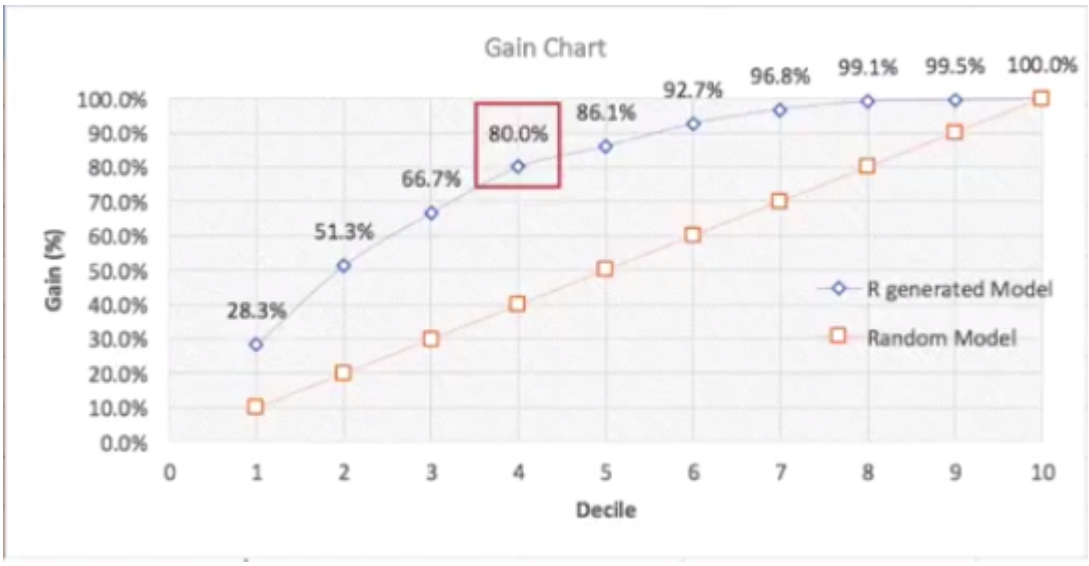
Sensitivity of a model is the proportion of yeses (or positives) correctly predicted by it as yeses (or positives). Also, specificity is equal to the proportion of nos (or negatives) correctly predicted by the model as nos (or negatives). For any given model, if the sensitivity increases on changing the cutoff, its specificity goes down.



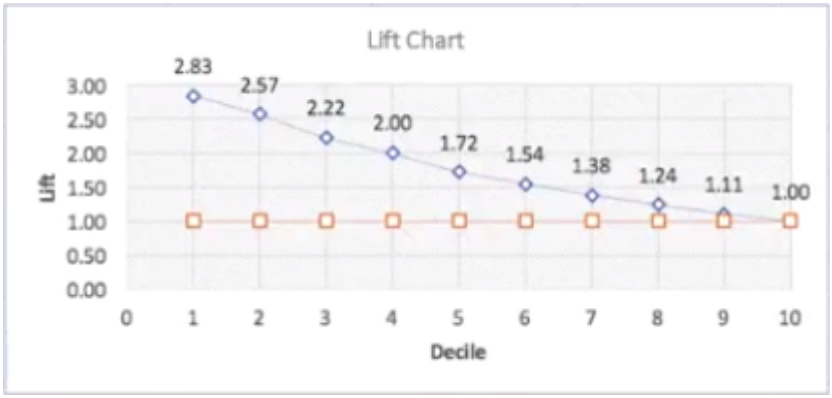
High values of both cannot be achieved in a single model. Hence, you have to choose which one you would want to be higher. The safest option, though, is the one in which you just take the cutoff that equalises accuracy, sensitivity and specificity.

Gain and Lift Charts

Next, you saw what gain and lift charts are.



If you say that the gain for a given model, is 78% by the 4th decile, what it basically means, is that if you sort all customers according to probability, then among the top 40% customers of this sorted list, you would find 78% of all customers that were likely to churn.



The lift just tells you the factor by which your model is outperforming a random model. For example, if your model's lift is equal to 2.1 by the 3rd decile, it means that your model's gain by the end of the 3rd decile is 2.1 times that of a random model's gain at the end of 3 deciles. In other words, the model catches 2.1 times more churns than a random model would have caught.

KS statistic

To end it all, you learnt about KS statistic.

Decile	Observations	Churn	Cum- Churn	% Cum-Churn	Non- Churn	Cum-Non-Churn	%Cum-Non-Churn	(%Cum-Churn) - (%Cum-Non-Churn)
1	211	159	159	28.3%	52	52	3.4%	25.0%
2	211	129	288	51.3%	82	134	8.7%	42.7%
3	211	86	374	66.7%	125	259	16.7%	49.9%
4	211	75	449	80.0%	136	395	25.5%	54.5%
5	211	34	483	86.1%	177	572	36.9%	49.2%
6	211	37	520	92.7%	174	746	48.2%	44.5%
7	211	23	543	96.8%	188	934	60.3%	36.5%
8	211	13	556	99.1%	198	1132	73.1%	26.0%
9	211	2	558	99.5%	209	1341	86.6%	12.9%
10	211	3	561	100.0%	208	1549	100.0%	0.0%
Total	2110	561			1549			

The highest value of the term (%cum-churn - %cum-non-churn) is called the KS statistic.

A high KS statistic means that not only does your model have all churns at the top, it has has all non-churns at the bottom. For a good model, KS statistic would be more than 40% and would lie in the top few deciles (1st to 4th).

Sample Selection for logistic Regression

Selecting the right sample is essential for solving any business problem. As discussed in the lecture, there are major errors you should be on the lookout for, while selecting a sample. These include:

- **Cyclical or seasonal fluctuations** in the business that need to be taken care of while building the samples. E.g. Diwali sales, economic ups and downs etc.
- The sample should be **representative of the population** on which the model will be applied in the future.
- For **rare events samples**, the sample should be balanced before they are used for modelling.

Segmentation

It is very helpful to perform **segmentation** of population before building a model.

Let's talk about the ICICI example,For students and salaried people, different variables may be important. While students' defaulting and not defaulting will depend on factors like program enrolled for, the prestige of the university attended, parents' income etc., the probability of salaried people will depend on factors like marital status, income etc. So, the predictive pattern across these two segments is very different, and hence, it would make more sense to make different child models for both of them, than to make one parent model.

A **segmentation** that divides your population into male and female may not be that effective, as the predictive pattern would not be that different for these two segments.

Variable Transformation

There are several approaches for trasforming independent variables. Few of them are as:

- Dummy variables transformation
- Weight of evidence (Woe) transformation
- Continuous variables transformation
- Interaction variables
- Splines
- Mathematical transformation
- Principal component transformation

Dummy Variable Transformation

You already know that **categorical variables** have to be transformed into **dummies**. Also, you were told that numeric variables have to be standardised, so that they all have the same scale. However, you could also convert **numeric variables** into **dummy** variables

There are some **pros and cons of transforming variables** to dummies. Creating dummies for categorical variables is very straightforward. You can directly create **n-1 new variables** from an existing categorical variable if it has **n levels**. But **for continuous variables**, you would be required to do some kind of EDA analysis for binning the variables.

The **major advantage** offered by dummies especially for continuous variables is that they make the **model stable**. In other words, small variations in the variables would not have a very big impact on a model that was made using dummies, but they would still have a sizeable impact on a model built using continuous variables as is.

On the other side, there are some major **disadvantages** that exist. E.g. if you change the continuous variable to dummies, all the data will be **compressed** into **very few categories** and that might result in **data clumping**.

Woe Transformation

Three are **three important** things about **WOE**:

- Calculating woe values for fine binning and coarse binning
- Importance of woe for fine binning and coarse binning
- Usage of woe transformation

There are two main advantages of **WOE**:

WOE reflects group identity, this means it captures the general trend of distribution of good and bad customers. E.g. the difference between customers with 30% credit card utilisation and 45% credit card utilisation is not the same as

the difference between customers with 45% credit card utilisation and customers with 60% credit card utilisation. This is captured by transforming the variable credit card utilisation using WOE.

Secondly, WOE helps you in treating missing values logically for both types of variables - categorical and continuous. E.g. in the credit card case, if you replace the continuous variable credit card utilisation with WOE values, you would replace all categories mentioned above (0%-45%, 45% - 60%, etc.) with certain specific values, and that would include the category "missing" as well, which would also be replaced with a WOE value.

WOE can be calculated using the following **equation**:

$$WOE = \ln\left(\frac{good\ in\ the\ bucket}{Total\ \ Good}\right) - \ln\left(\frac{bad\ in\ the\ bucket}{Total\ \ bad}\right)$$

Figure 1: Weight of Evidence

Or it can also be express by as follows:

$$WOE = ln(\frac{Percentage\ of\ Good}{Percentage\ of\ Bad})$$

Once you've calculated woe values, it is also important to note that they should follow an increasing or **decreasing trend** across bins. If the trend is not **monotonic**, then you would need to compress the buckets/ bins(coarse buckets) of that variable and then **calculate woe values again**.

Woe transformation does have **pros and cons** as well, which are quite similar to dummy variables.

- **Pros**: Model becomes more stable because small changes in the continuous variables will not impact the input so much.
- **Cons**: You may end up doing some score clumping.

This is because when you are using WOE values in your model, you are doing something similar to creating dummy variables - you are replacing a range of values with an indicative variable. It is just that instead of replacing it with a simple 1 or 0, which was not thought out at all, you are replacing it with a well thought out WOE value. Hence, the chances of undesired score clumping will be a lot less here.

Let's now move on to **IV (Information Value),** which is a very important concept. So, **information value** can be calculated using the following expression:

$$IV = WOE * \frac{(Good\ in\ the\ bucket}{Total\ Good} - \frac{Bad\ in\ the\ Bucket}{Total\ Bad})$$

Or it can expressed as :

$$IV = WOE * (Percentage\ of\ good\ in\ the\ bucket - Percentage\ of\ bad\ in\ the\ bucket)$$

It is an important indicator of **predictive power**.

Mainly, it helps you understand how the binning of variables should be done. The binning should be done such that the WOE trend across bins is monotonic - either increasing all the time or decreasing all the time. But one more thing that needs to be taken care of, is that IV (information value) should be high.

As there are so many other transformations, we can perform but they are not as important as discussed above .

Model Evaluation (A Second Look)

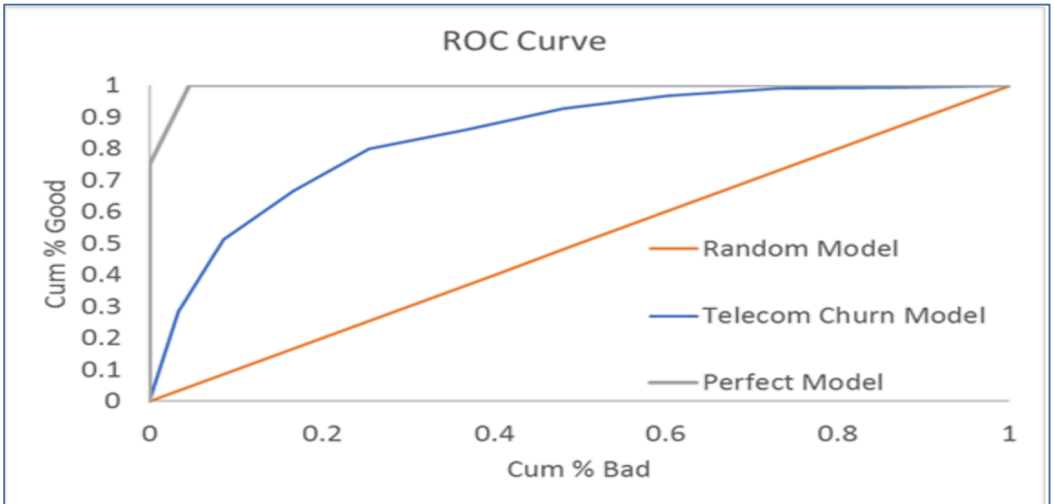
ROC Curve:

As discussed in lectures, **ROC** can be plotted between % of bad and % of good, or in simple language, % of event happen and % of non-event happen. Recall the telecom business problem, We calculated the KS table in the lecture which can be represented as:

Decile	Observations	Churn	Cum- Churn	% Cum-Churn	Non- Churn	Cum-Non-Churn	%Cum-Non-Churn	(%Cum-Churn) - (%Cum-Non-Churn)
1	211	159	159	28.3%	52	52	3.4%	25.0%
2	211	129	288	51.3%	82	134	8.7%	42.7%
3	211	86	374	66.7%	125	259	16.7%	49.9%
4	211	75	449	80.0%	136	395	25.5%	54.5%
5	211	34	483	86.1%	177	572	36.9%	49.2%
6	211	37	520	92.7%	174	746	48.2%	44.5%
7	211	23	543	96.8%	188	934	60.3%	36.5%
8	211	13	556	99.1%	198	1132	73.1%	26.0%
9	211	2	558	99.5%	209	1341	86.6%	12.9%
10	211	3	561	100.0%	208	1549	100.0%	0.0%
Total	2110	561			1549			

KS Statistic Calculation Table

Roc curve can be **visualised** as :



ROC Curve for Telecom Churn

The above image also contains the ROC curve for **the random model (red line)** and the **perfect model (grey line)**.

Clearly, the perfect model is pretty much a right triangle, whereas the random model is a straight line. Basically, a model that rises steeply is a good model.

Another way of saying that is, it will have a higher area under the curve. So, the Gini coefficient is given by:

$$Gini = Area\,Under\,ROC\,Curve$$

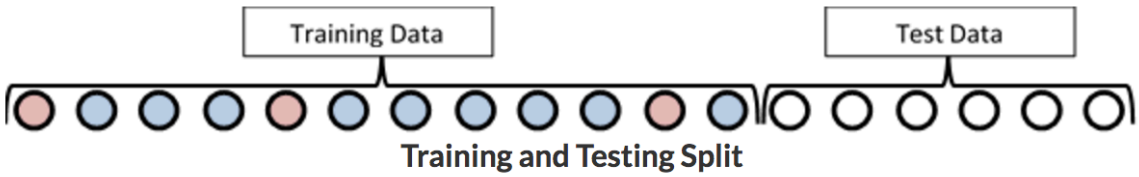
Gini will be high for a good model.

Model Validation

Model can be validated on:

- In-sample validation
- Out-time validation
- K-fold cross validation

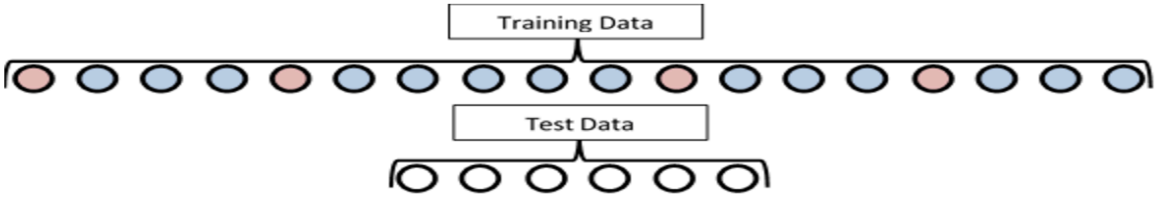
Recall Telecom business problem, The data used to build the model was from 2014. You split the original data into two parts, training and test data. However, these two parts were both with data from 2014.



Training and Testing Split

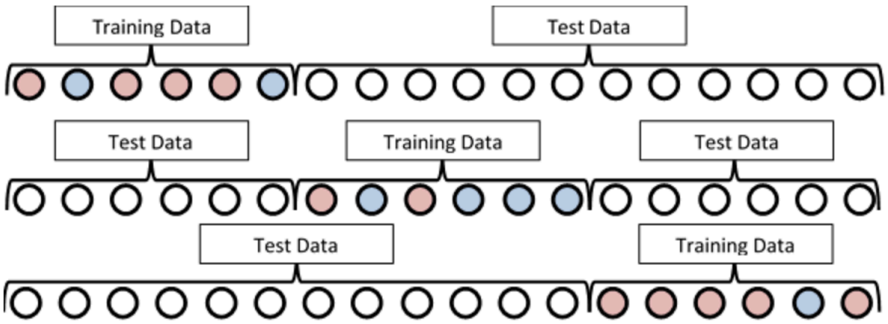
This is called **in-sample validation**. Testing your model on this test data may not be enough though, as test data is too similar to training data.

So, it makes sense to actually test the model on data that is from some other time, like 2016. This is called, **out of time validation.**



Out of Time Validation

Another way to do the same thing is to use K-fold cross validation. Basically, the evaluation of the sample is done for k-iterations. E.g. here's a representation of how 3 fold cross validation works:



3 Fold Cross Validation

Basically, there are 3 iterations in which evaluation is done. In the first iteration, 1/3rd of the data is selected as training data and the remaining 2/3rd of it is selected as testing data. In the next iteration, a different 1/3rd of the data is selected as the training data set and then the model is built and evaluated. Similarly, the third iteration is completed.

Such an approach is necessary if the data you have for model building is very small, i.e., has very few data points.

If these three methods of validation are still unclear to you, you need not worry as of now. They will be covered at length **in Course 4 (Predictive Analytics II)**

Model Stability

Obviously, a good model will be stable. A model is considered stable if it has:

- **Performance Stability** - Results of in-sample validation approximately match those of out-of-time validation
- **Variable Stability** - Sample used for model building hasn't changed too much and has the same general characteristics

Again, if stability is still a little cloudy, you need not worry. It will also be covered at length in Course 4 (Predictive Analytics II)