Logistic Regression

Module 1 : Univariate Logistic Regression

Module Introduction: Logistic Regression:-

Welcome to the module on '**Logistic Regression**'. In the last module on linear regression, you learnt how a linear regression model can be used to make predictions for continuous variables.

In this module
Now, in this module, you will learn logistic regression, which is a **classification model**, i.e. it will help you make predictions in cases where the output is a categorical variable.

Since logistic regression is the most easily interpretable of all classification models, it is very commonly used in various industries such as banking, healthcare, etc.

You can download the data sets used in this module from the files given below.

Introduction: Univariate Logistic Regression:-

Welcome to the session on '**Univariate Logistic Regression**'.

In this session

In this session, you will learn a few basic concepts related to logistic regression. Broadly speaking, the topics that will be covered in this session are:

- Binary classification
- Sigmoid function
- Likelihood function
- Building a logistic regression model in Python
- Odds and log odds

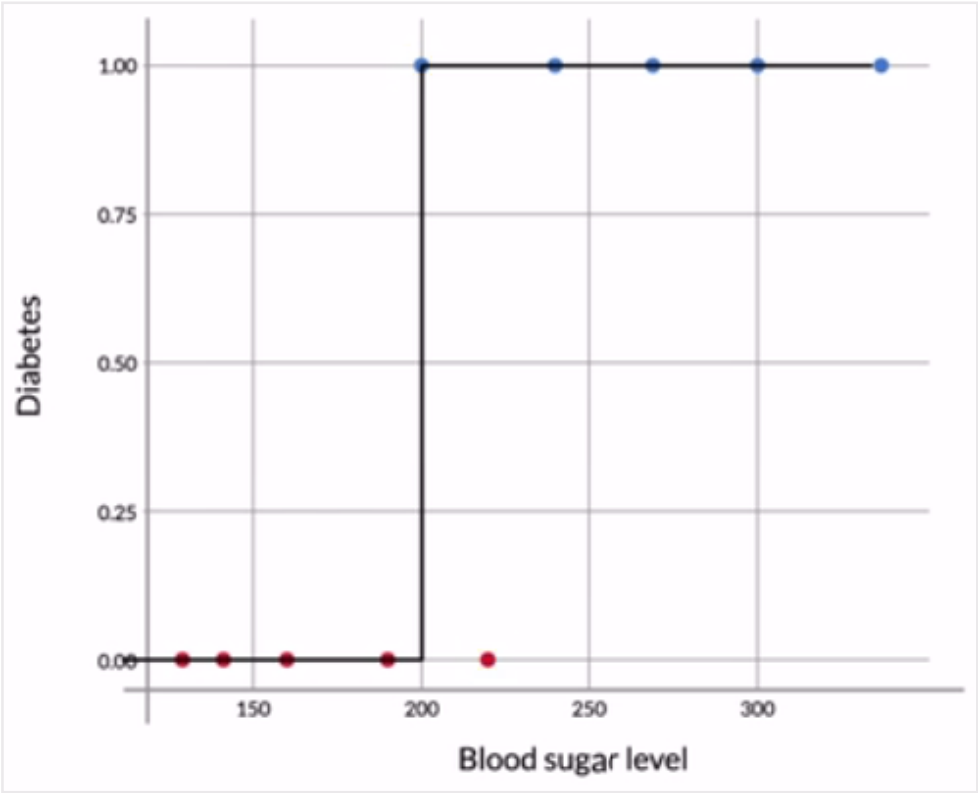
You will learn about all these concepts through a univariate logistic regression example. Also, if these terms sound a little alien to you right now, you don't need to worry. By the time you are done with this module, you will be well-versed in the terms of odds and log odds!

Binary Classification:-

The most common use of logistic regression models is in **binary classification** problems. So, let's hear from Prof. Dinesh what a binary classification problem is.

So, now you know what a binary classification problem is. In the next video, Prof. Dinesh explains it with a specific example. Let's see what that is.

Now, recall the graph of the diabetes example. Suppose there is another person, with a blood sugar level of 195, and you do not know whether that person has diabetes or not. What would you do then? Would you classify him/her as a diabetic or as a non-diabetic?



Now, based on the boundary, you may be tempted to declare this person a diabetic, but can you really do that? This person's sugar level (195 mg/dL) is very close to the threshold (200 mg/dL), below which people are declared as non-diabetic. It is, therefore, quite possible that this person was just a non-diabetic with a slightly high blood sugar level. After all, the data does have people with slightly high sugar levels (220 mg/dL), who are not diabetics.

Sigmoid Curve:-

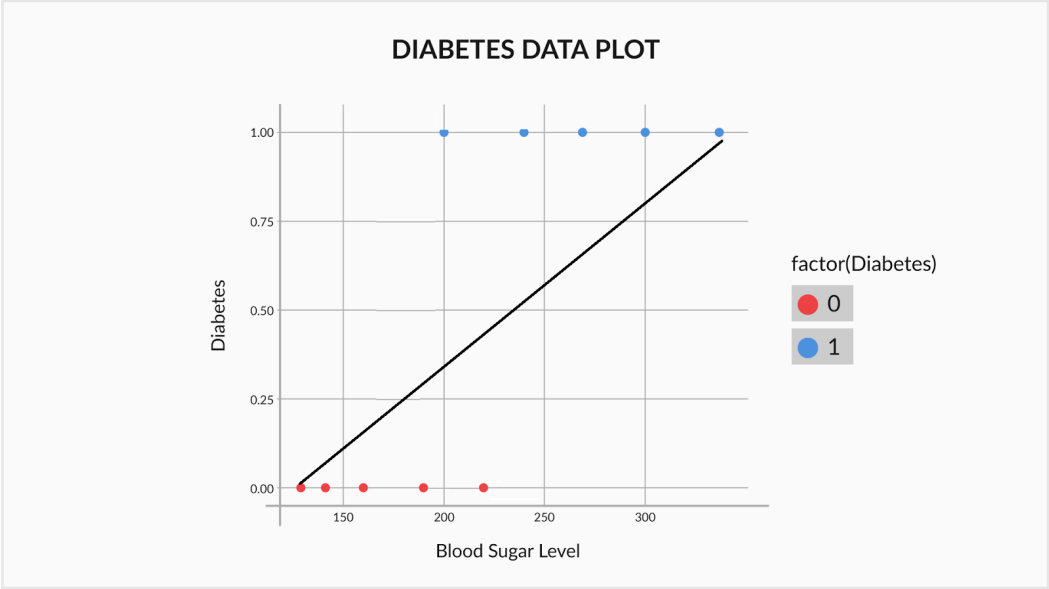
In the last section, you saw what a binary classification problem is, and then you saw an example of a binary classification problem, where a model is trying to predict whether a person has diabetes or not based on his/her blood sugar level. You saw how using a **simple boundary decision method** would not work in this case.

Now, let's hear from Prof. Dinesh on how the primitive binary classification model you saw earlier can be modified to make it more useful.

So, to recap, since the **sigmoid curve** has all the properties you would want — extremely low values in the start, extremely high values in the end, and intermediate values in the middle — it's a good choice for modelling the value of the **probability of diabetes**.

So now we have verified, with actual values, that the sigmoid curve actually has the properties we discussed earlier, i.e. extremely low values in the start, extremely high values in the end, and intermediate values in the middle.

However, you may be wondering — why can't you just fit a straight line here? This would also have the same properties — low values in the start, high ones towards the end, and intermediate ones in the middle.



The main problem with a straight line is that it is not steep enough. In the sigmoid curve, as you can see, you have low values for a lot of points, then the values rise all of a sudden, after which you have a lot of high values. In a straight line though, the values rise from low to high very uniformly, and hence, the "boundary" region, the one where the probabilities transition from high to low is not present.

You now have a good idea of what exactly a sigmoid curve is. In the next segment, you will learn how to find the best fit sigmoid curve.

Finding the Best Fit Sigmoid Curve - I:-

So, in the previous lecture, you saw what a sigmoid function is and why it is a good choice for modelling the probability of a class. Now, in this section, you will learn how you can find the **best fit sigmoid curve**. In other words, you will

learn how to find the combination of
β0 and β1 which fits the data best.

So, by varying the values of β0 and β1, you get different sigmoid curves. Now, based on some function that you have to minimise or maximise, you will get the best fit sigmoid curve.

Before you move on to that, here's the interactive app used by Prof. Dinesh in the video. You can use it and see for yourself how the curve changes when the values of β0 and β1 are changed.

In the next video, you will learn how to find the best fit sigmoid curve by choosing appropriate values of β0 and β1.

So, the best fitting combination of β_0 and β_1 will be the one which maximises the product:

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

This product is called the **likelihood function**. It is the product of:

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

So, say that for the ten points in our example, the labels are a little different, somewhat like this:

Point no.	1	2	3	4	5	6	7	8	9	10
Diabetes	no	no	no	yes	no	yes	no	yes	yes	yes

In this case, the likelihood would be equal to

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

Finding the Best Fit Sigmoid Curve - II:-

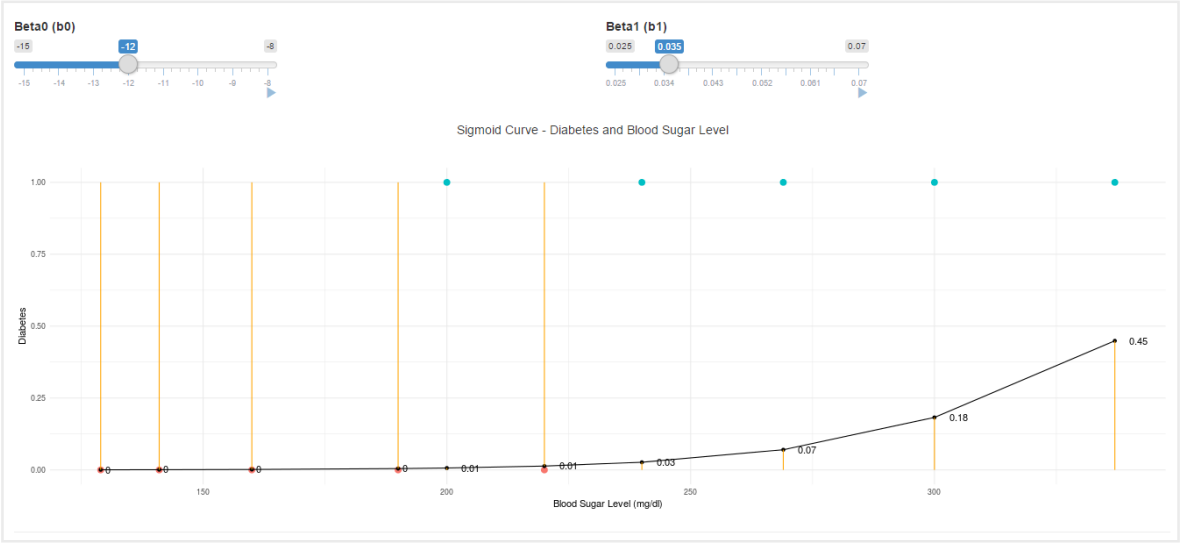
In the previous lecture, you understood what a likelihood function is. To recap, the likelihood function for our data is $(1-P_1)(1-P_2)(1-P_3)(1-P_4)(1-P_6)(P_5)(P_7)(P_8)(P_9)(P_{10})$. The best-fitting sigmoid curve would be the one which maximises the value of this product.

So, let's hear from Prof. Dinesh on how this best-fitting sigmoid curve can be found.
If you had to find $\beta_0$and β_1 for the best-fitting sigmoid curve, you would have to try a lot of combinations, unless you arrive at the one which maximises the likelihood. This is similar to linear regression, where you vary β_0 and β_1 until you find the combination that minimises the cost function.

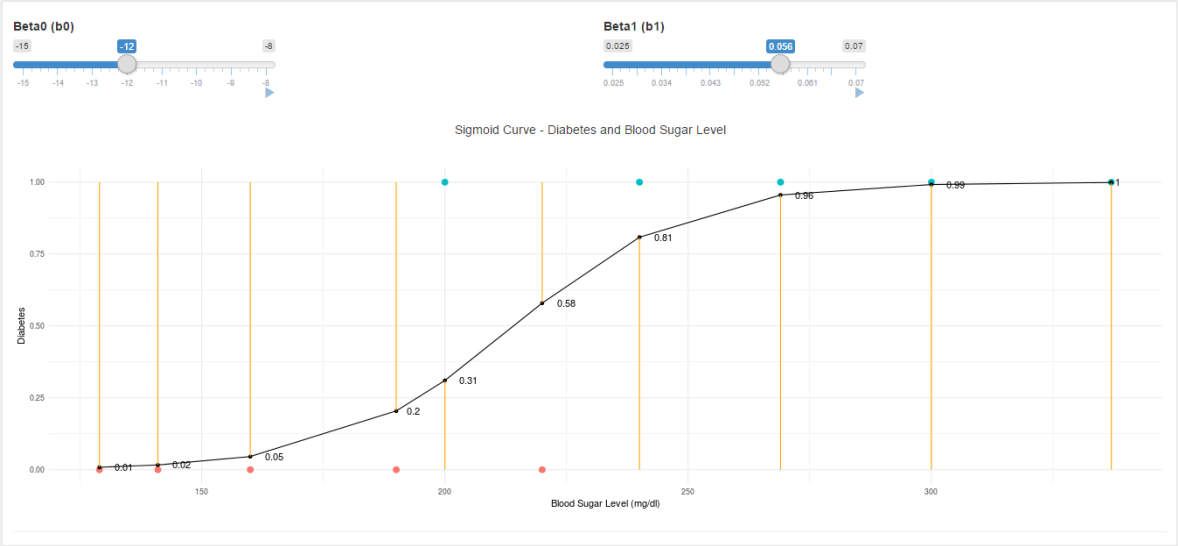
Correction: In the above video, the value of the likelihood at the time 1:15 is 2.12×10^{-5} instead of 1.75×10^{-5}

In the interactive app given below, you can try a few combinations yourself and see how the likelihood varies with betas.

So, just by looking at the curve here, you can get a general idea of the curve's fit. Just look at the yellow bars for each of the 10 points. A curve that has a lot of big yellow bars is a good curve. For example, this curve is not a good fit:



This curve, though, is a better fit -



Clearly, this curve is a better fit. It has many big yellow bars, and even the small ones are reasonably large. Just by looking at this curve, you can tell that it will have a high likelihood value.

Odds and Log Odds:-

In the previous segment, you saw that by trying different values of

β0 and β1, you can manipulate the shape of the sigmoid curve. At some combination of β0 and β1, the 'likelihood' (length of yellow bars) will be maximised.

Logistic Regression - Optimisation Methods (Optional)

The question is - how do you find the optimal values of β_0 and β_1 such that the likelihood function is maximized? The optimisation methods used to do that are an optional part of the course (maximum likelihood estimation, or MLE). You can study the optimisation techniques in detail in a separate optional session here. The following concepts are covered in the optional session:

- Introduction to maximum likelihood estimation (MLE)
- MLE for continuous (normal) and discrete probability distributions (Bernoulli distribution and logistic regression)
- Optimising MLE cost functions using gradient descent
- Alternate optimisation methods: Newton-Raphson method

You may study the optional session either along with this module or sometime later; this will not interrupt your flow of study for this module (there is no deadline for the optional session). There are no prerequisites for the optional session apart from the previous module.

Logistic Regression in Python

Let's now look at how logistic regression is implemented in python.

In python, logistic regression can be implemented using libraries such as SKLearn and statsmodels, though looking at the coefficients and the model summary is easier using statsmodels.

You can find the optimum values of β_0 and β_1 using the python code given below. Please download and run the code and observe the values of the coefficients (you may also want to visualise them on the interactive app on the previous page, though note that β_0 can only take integer values in the app, so you can use -13 or -14 approximately).

Please note that you will study a detailed Python code for logistic regression in the next module. This Python code has been run so as to find the optimum values of β_0 and β_1 so that we can first proceed with the very important concept of **Odds** and **Log Odds**.

The summary of the model is given below:

Generalized Linear Model Regression Results

Dep. Variable:	Diabetes	**No. Observations:**	10
Model:	GLM	**Df Residuals:**	8
Model Family:	Binomial	**Df Model:**	1
Link Function:	logit	**Scale:**	1.0
Method:	IRLS	**Log-Likelihood:**	-2.5838
Date:	Tue, 06 Mar 2018	**Deviance:**	5.1676
Time:	00:56:36	**Pearson chi2:**	4.32
No. Iterations:	7		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-13.5243	9.358	-1.445	0.148	-31.866	4.817
Blood Sugar Level	0.0637	0.044	1.439	0.150	-0.023	0.150

In the summary shown above, 'const' corresponds to β_0 and Blood Sugar Level, i.e. 'x1' corresponds to β_1 . So, β_0 = -13.5 and β_1 = 0.06.

Odds and Log Odds

So far, you've seen this equation for logistic regression:

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

Recall that this equation gives the relationship between P, the probability of diabetes and x, the patient's blood sugar level.

While the equation is correct, it is not very intuitive. In other words, the relationship between P and x is so complex that it is difficult to understand what kind of trend exists between the two. If you increase x by regular intervals of, say, 11.5, how will that affect the probability? Will it also increase by some regular interval? If not, what will happen?

So, clearly, the relationship between P and x is too complex to see any apparent trends. However, if you convert the equation to a slightly different form, you can achieve a much more intuitive relationship. In the next video, let's hear from Prof. Dinesh on how that can be done.

[*Note: By default, for this course, if the base of the logarithm is not specified, take it as e. So,* $log(x) = log_e(x)$.]

So, now, instead of probability, you have **odds** and **log odds**. Clearly, the relationship between them and x is much more **intuitive** and easy to understand.

For example, if you increase x by regular intervals of, say, 11.5, how will that affect the log odds?

Please note that, in the video above, at **3:05**, instead of 2.94, it should have been **2.96**

So, the relationship between x and probability is not intuitive, while that between x and **odds/log odds** is. This has important implications. Suppose you are discussing sugar levels and the probability they correspond to. While talking about 4 patients with sugar levels of 180, 200, 220 and 240, you will not be able to intuitively understand the relationship between their probabilities (10%, 28%, 58%, 83%). However, if you are talking about the log odds of these 4 patients, you know that their log odds are in a **linearly increasing pattern** (-2.18, -0.92, 0.34, 1.60) and that the odds are in a **multiplicatively increasing pattern** (0.11, 0.40, 1.40, 4.95, increasing by a factor of 3.55).

Hence, many times, it makes more sense to present a logistic regression model's results in terms of log odds or odds than to talk in terms of probability.

This happens especially a lot in industries like finance, banking, etc.

That's the end of this session on univariate logistic regression. You studied logistic regression, specifically, the sigmoid function, which has this equation:

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

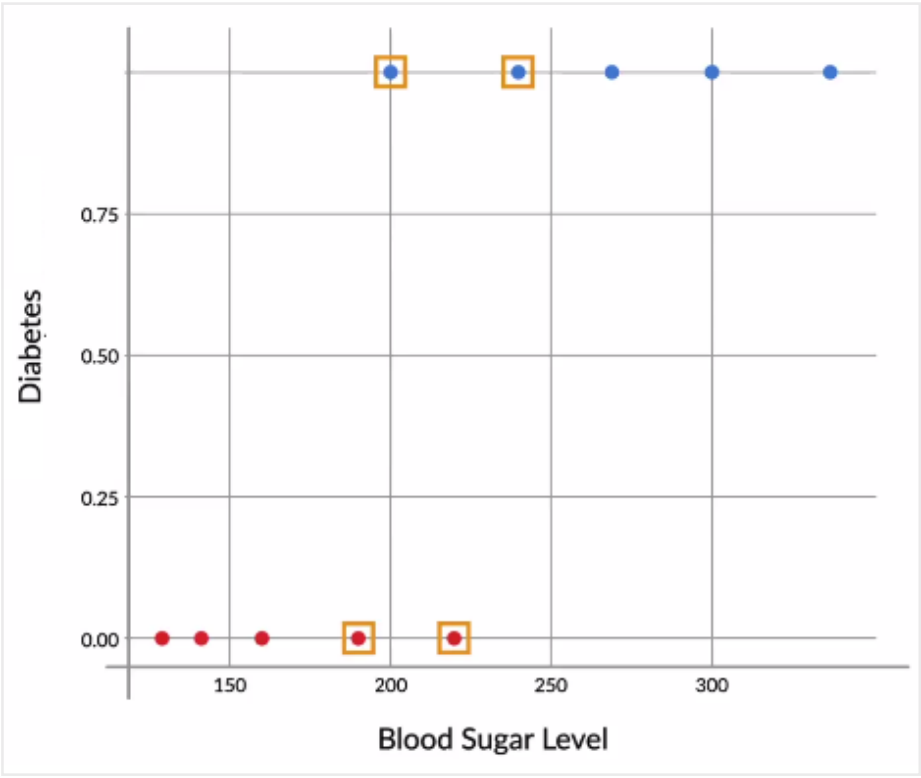
However, this is not the only form of equation for logistic regression. If you wish to learn about what the other forms are, you can go through them here. For this course, you do not need to know about the other forms, as we will not be discussing them anywhere.

Summary:-

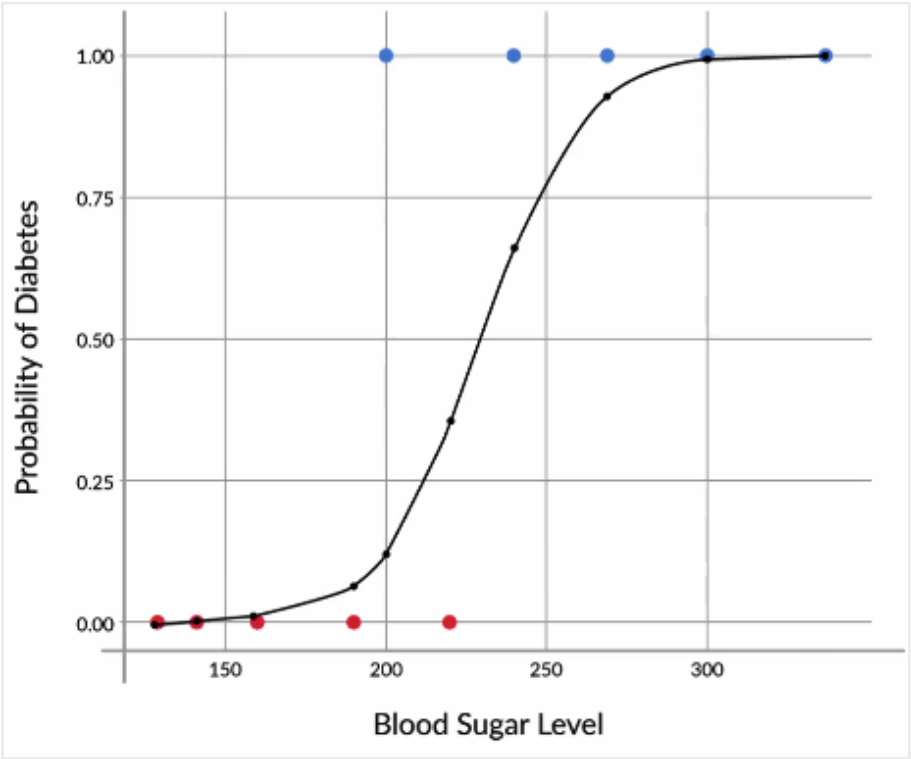
You first learnt what a **binary classification** is. Basically, it is a classification problem in which the target variable has only 2 possible values.

You then went through the **diabetes example** in detail, wherein you tried 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 the cutoff because, especially in the middle, the patients could belong to any class — diabetic or non-diabetic.



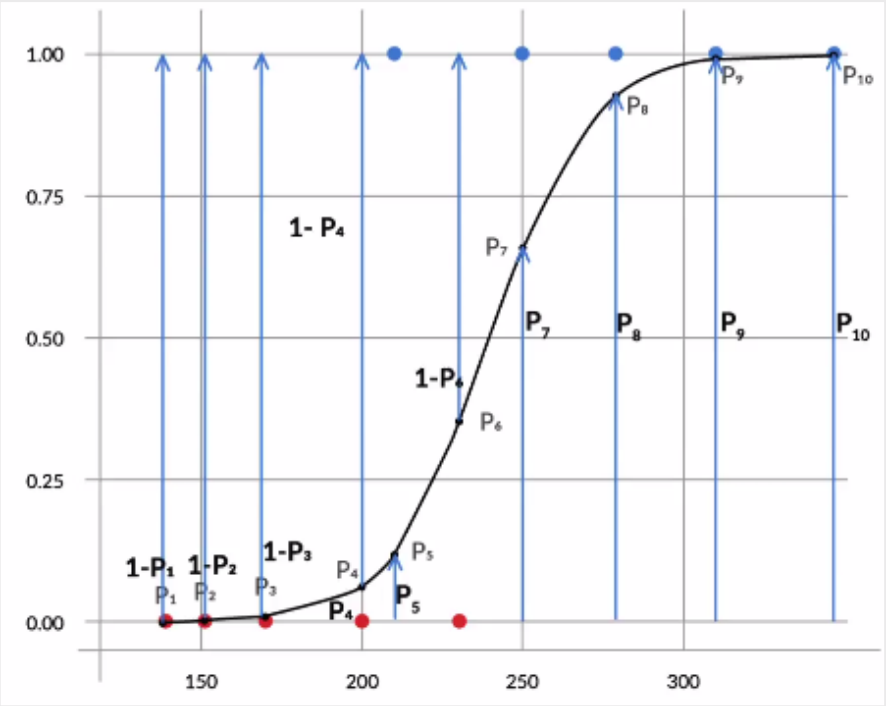
Hence, you learnt that it is better to talk in terms of **probability**. One such curve which can model the probability of diabetes very well is the **sigmoid curve**.



Its equation is given by the following expression:

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

Then, 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, the likelihood is given by the expression:



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

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 the probability of diabetes, is called **logistic regression.**

After this, you saw a simpler way of interpreting the equation for logistic regression. You saw that the following linearised equation is much easier to interpret:

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

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 likelier 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).

You also 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 are approximately doubled, i.e. they increase by a multiplicative factor of about 2.

Module 2 : Multivariate Logistic Regression - Model Building

Introduction:-

Welcome to the session on **'Multivariate Logistic Regression (Model Building)'**.

Just like when you're building a model using linear regression, one independent variable might not be enough to capture all the uncertainties of the target variable in logistic regression as well. So in order to make good and accurate predictions, you need multiple variables and that is what we'll study in this session.

Before starting with multivariate logistic regression, the first question that arises is, "Do you need any extensions while moving from univariate to multivariate logistic regression?" Recall the equation used in the case of

univariate logistic regression was:

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

The above equation has only one feature variable X, for which the coefficient is β_1. Now, if you have multiple features, say n, you can simply extend this equation with 'n' feature variables and 'n' corresponding coefficients such that the equation now becomes:

$$P = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_3+...+\beta_n X_n)}}$$

Recall this extension is similar to what you did while moving from simple to multiple linear regression.

In this session

In this session, you will learn how to:

- Build a multivariate logistic regression model in Python
- Conduct feature selection for logistic regression using:
  - Automated methods: RFE -Recursive Feature Elimination
  - Manual methods: VIF and p-value check

We will use the 'Telecom Churn' dataset in this session to build a model using multivariate logistic regression. This will involve all the familiar steps such as:

- Data cleaning and preparation
- Preprocessing steps
- Test-train split
- Feature scaling
- Model Building using RFE, p-values and VIFs

Apart from the familiar old steps, you'll also be introduced to something known as a confusion matrix and you'll also learn how the accuracy is measured for a logistic regression model.

Multivariate Logistic Regression - Telecom Churn Example:-

Let's now look at the process of building a logistic regression model in Python.

You will be looking at the **telecom churn prediction** example. You will use 21 variables related to customer behaviour (such as the monthly bill, internet usage etc.) to predict whether a particular customer will switch to another telecom provider or not (i.e. churn or not).

Problem Statment

You have a telecom firm which has collected data of all its customers. The main types of attributes are:

- Demographics (age, gender etc.)
- Services availed (internet packs purchased, special offers taken etc.)
- Expenses (amount of recharge done per month etc.)

Based on all this past information, you want to build a model which will predict whether a particular customer will churn or not, i.e. whether they will switch to a different service provider or not. So the variable of interest, i.e. the target variable here is 'Churn' which will tell us whether or not a particular customer has churned. It is a binary variable - 1 means that the customer has churned and 0 means the customer has not churned.

You can download the datasets here:

So, here's what the data frame churn_data looks like:

	customerID	tenure	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	1	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	34	Yes	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	2	Yes	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	45	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	2	Yes	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

Also, here's the data frame customer_data:

	customerID	gender	SeniorCitizen	Partner	Dependents
0	7590-VHVEG	Female	0	Yes	No
1	5575-GNVDE	Male	0	No	No
2	3668-QPYBK	Male	0	No	No
3	7795-CFOCW	Male	0	No	No
4	9237-HQITU	Female	0	No	No

Lastly, here's the data frame internet_data:

	customerID	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies
0	7590-VHVEG	No phone service	DSL	No	Yes	No	No	No	No
1	5575-GNVDE	No	DSL	Yes	No	Yes	No	No	No
2	3668-QPYBK	No	DSL	Yes	Yes	No	No	No	No
3	7795-CFOCW	No phone service	DSL	Yes	No	Yes	Yes	No	No
4	9237-HQITU	No	Fiber optic	No	No	No	No	No	No

Now, as you can clearly see, the first 5 customer IDs are exactly the same for each of these data frames. Hence, using the column customer ID, you can collate or merge the data into a single data frame. We'll start with that in the next segment.

Coming Up
In the next segment, you will start with reading and inspecting the dataframes and then move on to preparing that data for model building.

Data Cleaning and Preparation - I:-

Before you jump into the actual model building, you first need to clean and prepare your data. As you might have seen in the last segment, all the useful information is present in three dataframes with 'Customer ID' being the common column. So as the first step, you need to merge these three data files so that you have all the useful data combined into a single master dataframe.

Now that you have the master dataframe in place, and you have also performed a binary mapping for few of the categorical variables, the next step would be to create dummy variables for features with multiple levels. The dummy variable creation process is similar to what you did in linear regression as well.

Note: At 3.22 -"convert_objects" function has been used to convert the column from objects to numeric. This function is deprecated in the newer version of Pandas. Instead, pd.to_numeric can be used.

So the process of dummy variable creation was quite familiar, except this time, you manually dropped one of the columns for many dummy variables. For example, for the column 'MultipleLines', you dropped the level 'MultipleLines_No phone service' manually instead of simply using 'drop_first = True' which would've dropped the first level present in the 'MultipleLines' column. The reason we did this is that if you check the variables 'MultipleLines' using the following command, you can see that it has the following three levels:

```
telecom['MultipleLines'].astype('category').value_counts()
```

```
No                  3390
Yes                 2971
No phone service     682
Name: MultipleLines, dtype: int64
```

Now, out of these levels, it is best that you drop ‘No phone service’ since it isn’t of any use because it is anyway being indicated by the variable ‘PhoneService’ already present in the dataframe.

To simply put it, the variable **‘PhoneService’** already tells you whether the phone services are availed or not by a particular customer. In fact, if you check the value counts of the variable 'PhoneService', following is the output that you get:

```
telecom['PhoneService'].astype('category').value_counts()
```

```
Yes    6361
No      682
Name: PhoneService, dtype: int64
```

You can see that the level 'No' appears **682 times** which is **exactly equal to** the count of the level 'No phone service' in 'MultipleLines'.

You can see that the dummy variable for this level, i.e. 'MultipleLines_No phone service' is clearly redundant since it doesn't contain any extra information and hence, to drop it is the best option at this point. You can verify it similarly for all the other categorical variables for which one of the levels was manually dropped.

Coming Up
You will go on with the data cleaning and preparation steps in the next segment.

Data Cleaning and Preparation - II:-

You’ve merged your dataframes and handled the categorical variables present in them. But you still need to check the data for any outliers or missing values and treat them accordingly. Let's get this done as well.

You saw that one of the columns, i.e. 'TotalCharges' had 11 missing values. Since this isn't a big number compared to the number of rows present in a dataset, we decided to drop them since we won't lose much data.

Now that you have completely prepared your data, you can start with the preprocessing steps. As you might remember from the previous module, you

first need to split the data into train and test sets and then rescale the features. So let's start with that.

> Recall that, for continuous variables, Rahim scaled the variables to standardise the three continuous variables — tenure, monthly charges and total charges. Recall that scaling basically reduces the values in a column to within a certain range — in this case, we have converted the values to the Z-scores.
>
> For example, let's say that, for a particular customer, tenure = 72. After standardising, the value of scaled tenure becomes:
>
> $$\frac{72-32.4}{24.6} = 1.61$$
>
> because for the variable tenure, mean(μ) = 32.4 and standard deviation(σ) = 24.6.
>
> The variables had these ranges before standardisation:
>
> - Tenure = 1 to 72
> - Monthly charges = 18.25 to 118.80
> - Total charges = 18.8 to 8685

After standardisation, the ranges of the variables changed to:

- Tenure = -1.28 to +1.61
- Monthly charges = -1.55 to +1.79
- Total charges = -0.99 to 2.83

Clearly, none of the variables will have a disproportionate effect on the model's results now.

Churn Rate and Class Imbalance

Another thing to note here was the Churn Rate which Rahim talked about at the end of the video. You saw that the data has almost 27% churn rate. Checking the churn rate is important since you usually want your data to have a balance between the 0s and 1s (in this case churn and not-churn).

The reason for having a balance is simple. Let's do a simple thought experiment - if you had a data with, say, 95% not-churn (0) and just 5% churn (1), then even if you predict everything as 0, you would still get a model which is 95% accurate (though it is, of course, a bad model). This problem is called **class-imbalance** and you'll learn to solve such cases later.

Fortunately, in this case, we have about 27% churn rate. This is neither exactly 'balanced' (which a 50-50 ratio would be called) nor heavily imbalanced. So we'll not have to do any special treatment for this dataset.

Coming Up
Now that everything's in place, we can start building our model from the next segment.

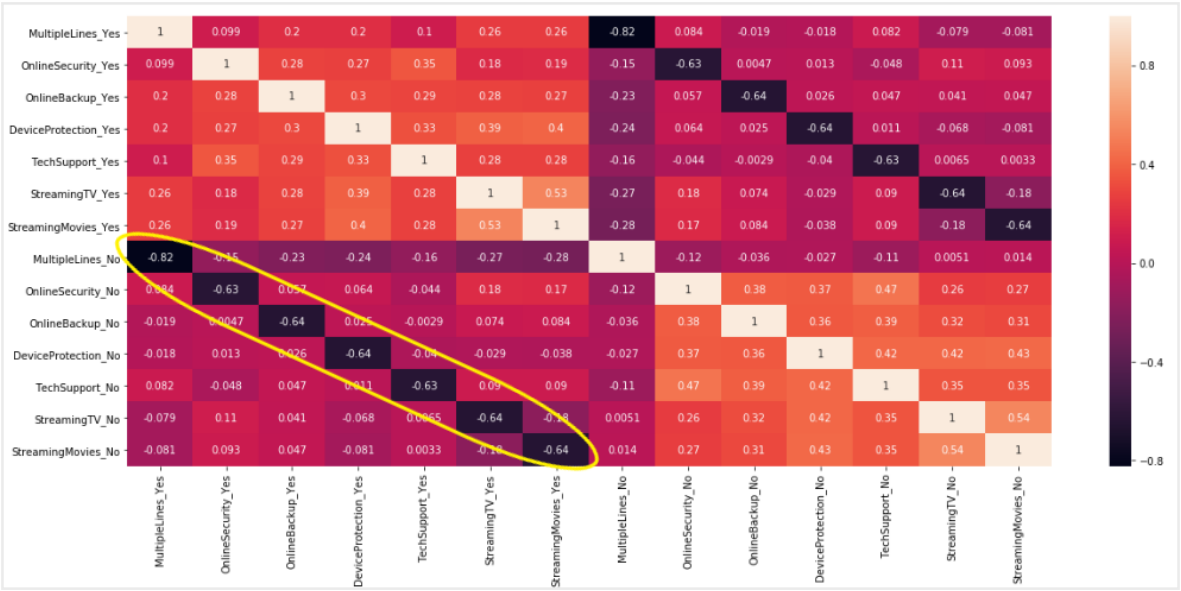
Building your First Model:-

Let's now proceed to model building. Recall that the first step in model building is to check the correlations between features to get an idea about how the different independent variables are correlated. In general, the process of feature selection is almost exactly analogous to linear regression.

Looking at the correlations certainly did help, as you identified a lot of features beforehand which wouldn't have been useful for model building. Recall that Rahim dropped the following features after looking at the correlations from the heatmap:

- MultipleLines_No
- OnlineSecurity_No
- OnlineBackup_No
- DeviceProtection_No
- TechSupport_No
- StreamingTV_No
- StreamingMovies_No

If you look at the correlations between these dummy variables with their complimentary dummy variables,
i.e. **'MultipleLines_No'** with **'MultipleLines_Yes'** or **'OnlineSecurity_No'** with **'OnlineSecurity_Yes'**, you'll find out they're highly correlated. Have a look at the heat map below:



If you check the highlighted portion, you'll see that there are high correlations

among the pairs of dummy variables which were created for the same column. For example, **'StreamingTV_No'** has a correlation of **-0.64** with **'StreamingTV_Yes'**. So it is better than we drop one of these variables from each pair as they won't add much value to the model. The choice of which of these pair of variables you desire to drop is completely up to you; we've chosen to drop all the 'Nos' because the 'Yeses' are generally more interpretable and easy-to-work-with variables.

Now that you have completed all the pre-processing steps, inspected the correlation values and have eliminated a few variables, it's time to build our first model.

So you finally built your first multivariate logistic regression model using all the features present in the dataset. This is the summary output for different variables that you got:

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.9382	1.546	-2.547	0.011	-6.969	-0.908
tenure	-1.5172	0.189	-8.015	0.000	-1.888	-1.146
PhoneService	0.9507	0.789	1.205	0.228	-0.595	2.497
PaperlessBilling	0.3254	0.090	3.614	0.000	0.149	0.502
MonthlyCharges	-2.1806	1.160	-1.880	0.060	-4.454	0.092
TotalCharges	0.7332	0.198	3.705	0.000	0.345	1.121
SeniorCitizen	0.3984	0.102	3.924	0.000	0.199	0.597
Partner	0.0374	0.094	0.399	0.690	-0.146	0.221
Dependents	-0.1430	0.107	-1.332	0.183	-0.353	0.067
Contract_One year	-0.6578	0.129	-5.106	0.000	-0.910	-0.405
Contract_Two year	-1.2455	0.212	-5.874	0.000	-1.661	-0.830
PaymentMethod_Credit card (automatic)	-0.2577	0.137	-1.883	0.060	-0.526	0.011
PaymentMethod_Electronic check	0.1615	0.113	1.434	0.152	-0.059	0.382
PaymentMethod_Mailed check	-0.2536	0.137	-1.845	0.065	-0.523	0.016
gender_Male	-0.0346	0.078	-0.442	0.658	-0.188	0.119
InternetService_Fiber optic	2.5124	0.967	2.599	0.009	0.618	4.407
InternetService_No	-2.7792	0.982	-2.831	0.005	-4.703	-0.855
MultipleLines_Yes	0.5623	0.214	2.628	0.009	0.143	0.982
OnlineSecurity_Yes	-0.0245	0.216	-0.113	0.910	-0.448	0.399
OnlineBackup_Yes	0.1740	0.212	0.822	0.411	-0.241	0.589
DeviceProtection_Yes	0.3229	0.215	1.501	0.133	-0.099	0.744
TechSupport_Yes	-0.0305	0.216	-0.141	0.888	-0.455	0.394
StreamingTV_Yes	0.9598	0.396	2.423	0.015	0.183	1.736
StreamingMovies_Yes	0.8484	0.396	2.143	0.032	0.072	1.624

In this table, our key focus area is just the different **coefficients** and their respective **p-values**. As you can see, there are many variables whose p-values are high, implying that that variable is statistically insignificant. So we need to eliminate some of the variables in order to build a better model.

We'll first eliminate a few features using Recursive Feature Elimination (RFE), and once we have reached a small set of variables to work with, we can then

use manual feature elimination (i.e. manually eliminating features based on observing the p-values and VIFs).

Coming Up
In the next segment, you'll use RFE to build another model and also look at some metrics using which you can check the goodness of fit.

Feature Elimination using RFE:-

You built your first model in the previous segment. Based on the summary statistics, you inferred that many of the variables might be insignificant and hence, you need to do some feature elimination. Since the number of features is huge, let's first start off with an automated feature selection technique (RFE) and then move to manual feature elimination (using p-values and VIFs) - this is exactly the same process that you did in linear regression.

So let's start off with the automatic feature selection technique - RFE.

Let's summarise the steps you just performed one by one. First, you imported the logistic regression library from sklearn and created a logistic regression object using:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

Then you run an RFE on the dataset using the same command as you did in linear regression. In this case, we choose to select 15 features first (15 is, of course, an arbitrary number).

```
from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15)             # running RFE with 15 variables as output
rfe = rfe.fit(X_train, y_train)
```

RFE selected 15 features for you and following is the output you got:

```
[('tenure', True, 1),
 ('PhoneService', True, 1),
 ('PaperlessBilling', True, 1),
 ('MonthlyCharges', False, 6),
 ('TotalCharges', True, 1),
 ('SeniorCitizen', True, 1),
 ('Partner', False, 8),
 ('Dependents', False, 4),
 ('Contract_One year', True, 1),
 ('Contract_Two year', True, 1),
 ('PaymentMethod_Credit card (automatic)', True, 1),
 ('PaymentMethod_Electronic check', False, 3),
 ('PaymentMethod_Mailed check', True, 1),
 ('gender_Male', False, 9),
 ('InternetService_Fiber optic', True, 1),
 ('InternetService_No', True, 1),
 ('MultipleLines_Yes', True, 1),
 ('OnlineSecurity_Yes', True, 1),
 ('OnlineBackup_Yes', False, 2),
 ('DeviceProtection_Yes', False, 7),
 ('TechSupport_Yes', True, 1),
 ('StreamingTV_Yes', True, 1),
 ('StreamingMovies_Yes', False, 5)]
```

You can see that RFE has eliminated certain features such as 'MonthlyCharges', 'Partner', 'Dependents', etc.

We decided to go ahead with this model but since we are also interested in the statistics, we take the columns selected by RFE and use them to build a model using statsmodels using:

```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
```

Here, you use the **GLM (Generalized Linear Models)** method of the library statsmodels. **'Binomial()'** in the 'family' argument tells statsmodels that it needs to fit **a logit curve to a binomial data** (i.e. in which the target will have just two classes, here 'Churn' and 'Non-Churn').

Now, recall that the logistic regression curve gives you the **probabilities of churning and not churning**. You can get these probabilities by simply using the **'predict'** function as shown in the notebook.

Since the logistic curve gives you just the probabilities and not the actual classification of **'Churn'** and **'Non-Churn'**, you need to find a **threshold probability** to classify customers as 'churn' and 'non-churn'. Here, we choose 0.5 as an arbitrary cutoff wherein if the probability of a particular customer churning is less than 0.5, you'd classify it as **'Non-Churn'** and if it's greater than 0.5, you'd classify it as **'Churn'**. **The choice of 0.5 is completely**

arbitrary at this stage and you'll learn how to find the optimal cutoff in 'Model Evaluation', but for now, we'll move forward with 0.5 as the cutoff.

Coming Up
In the next segment, you will learn how to calculate the accuracy of the fitted logistic regression curve.

Confusion Matrix and Accuracy:-

You chose a cutoff of 0.5 in order to classify the customers into 'Churn' and 'Non-Churn'. Now, since you're classifying the customers into two classes, you'll obviously have some errors. The classes of errors that would be there are:

- 'Churn' customers being (incorrectly) classified as 'Non-Churn'
- 'Non-Churn' customers being (incorrectly) classified as 'Churn'

To capture these errors, and to evaluate how well the model is, you'll use something known as the **'Confusion Matrix'**. A typical confusion matrix would look like the following:

Actual	Predicted	
	No (Non-Churn)	Yes (Churn)
No (Non-Churn)	1406	143
Yes (Churn)	263	298

This table shows a comparison of the predicted and actual labels. The actual labels are along the vertical axis, while the predicted labels are along the horizontal axis. Thus, the second row and first column (263) is the number of customers who have actually ‘churned’ but the model has predicted them as non-churn.

Similarly, the cell at the second row, the second column (298) is the number of customers who are actually ‘churn’ and also predicted as ‘churn’.

Note that this is an example table and not what you'll get in Python for the model you've built so far. It is just used as an example to illustrate the concept.

Now, the simplest model evaluation metric for classification models is **accuracy** - it is the percentage of correctly predicted labels. So what would the correctly predicted labels be? They would be:

- 'Churn' customers being actually identified as churn
- 'Non-churn' customers being actually identified as non-churn.

As you can see from the table above, the correctly predicted labels are

contained in the first row and first column, and the last row and last column as can be seen highlighted in the table below:

Actual	Predicted	
	No (Non-Churn)	Yes (Churn)
No (Non-Churn)	1406	143
Yes (Churn)	263	298

Now, accuracy is defined as:

$$Accuracy = \frac{Correctly\ Predicted\ Labels}{Total\ Number\ of\ Labels}$$

Hence, using the table, we can say that the accuracy for this table would be:

$$Accuracy = \frac{1406+298}{1406+143+263+298} \approx 80.75$$

Now that you know about confusion matrix and accuracy, let's see how good is your model built so far based on the accuracy. But first, answer a couple of questions.

So using the confusion matrix, you got an accuracy of about 80.8% which seems to be a good number to begin with. The steps you need to calculate accuracy are:

- Create the confusion matrix
- Calculate the accuracy by applying the 'accuracy_score' function to the above matrix

The code used to do this was:

```
# Create confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.Churn,
y_train_pred_final.predicted)

# Calculate accuracy
print(metrics.accuracy_score(y_train_pred_final.Churn,
y_train_pred_final.predicted))
```

Coming Up

So far you have only selected features based on RFE. Further elimination of features using the p-values and VIFs manually is yet to be done. You'll do that in the next section.

Manual Feature Elimination:-

Recall that you had used RFE to select 15 features. But as you saw in the pairwise correlations, there are high values of correlations present between the 15 features, i.e. there is still some multicollinearity among the features. So you definitely need to check the VIFs as well to further eliminate the redundant variables. Recall that VIF calculates how well one independent variable is explained by all the other independent variables combined. And its formula is given as:

$$VIF_i = \frac{1}{1-{R_i}^2}$$

where 'i' refers to the ith variable which is being represented as a combination of rest of the independent variables.

Let's see Rahim talk about eliminating the insignificant variables based on the VIFs, and the p-values.

To summarise, you basically performed an iterative manual feature elimination using the VIFs and p-values repeatedly. You also kept on checking the value of accuracy to make sure that dropping a particular feature doesn't affect the accuracy much.

This was the set of 15 features that RFE had selected which we began with:

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.0343	0.171	-6.053	0.000	-1.369	-0.699
tenure	-1.5386	0.184	-8.381	0.000	-1.898	-1.179
PhoneService	-0.5231	0.161	-3.256	0.001	-0.838	-0.208
PaperlessBilling	0.3397	0.090	3.789	0.000	0.164	0.515
TotalCharges	0.7116	0.188	3.794	0.000	0.344	1.079
SeniorCitizen	0.4294	0.100	4.312	0.000	0.234	0.625
Contract_One year	-0.6813	0.128	-5.334	0.000	-0.932	-0.431
Contract_Two year	-1.2680	0.211	-6.011	0.000	-1.681	-0.855
PaymentMethod_Credit card (automatic)	-0.3775	0.113	-3.352	0.001	-0.598	-0.157
PaymentMethod_Mailed check	-0.3760	0.111	-3.389	0.001	-0.594	-0.159
InternetService_Fiber optic	0.7421	0.117	6.317	0.000	0.512	0.972
InternetService_No	-0.9385	0.166	-5.650	0.000	-1.264	-0.613
MultipleLines_Yes	0.2086	0.096	2.181	0.029	0.021	0.396
OnlineSecurity_Yes	-0.4049	0.102	-3.968	0.000	-0.605	-0.205
TechSupport_Yes	-0.3967	0.102	-3.902	0.000	-0.596	-0.197
StreamingTV_Yes	0.2747	0.094	2.911	0.004	0.090	0.460

And this is the final set of features which you arrived at after eliminating features manually:

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.4695	0.130	-11.336	0.000	-1.724	-1.215
tenure	-0.8857	0.065	-13.553	0.000	-1.014	-0.758
PaperlessBilling	0.3367	0.089	3.770	0.000	0.162	0.512
SeniorCitizen	0.4517	0.100	4.527	0.000	0.256	0.647
Contract_One year	-0.6792	0.127	-5.360	0.000	-0.927	-0.431
Contract_Two year	-1.2308	0.208	-5.903	0.000	-1.639	-0.822
PaymentMethod_Credit card (automatic)	-0.3827	0.113	-3.399	0.001	-0.603	-0.162
PaymentMethod_Mailed check	-0.3393	0.110	-3.094	0.002	-0.554	-0.124
InternetService_Fiber optic	0.7914	0.098	8.109	0.000	0.600	0.983
InternetService_No	-1.1205	0.157	-7.127	0.000	-1.429	-0.812
MultipleLines_Yes	0.2166	0.092	2.355	0.019	0.036	0.397
OnlineSecurity_Yes	-0.3739	0.101	-3.684	0.000	-0.573	-0.175
TechSupport_Yes	-0.3611	0.101	-3.591	0.000	-0.558	-0.164
StreamingTV_Yes	0.3995	0.089	4.465	0.000	0.224	0.575

As you can see, we had dropped the features **'PhoneService'** and **'TotalCharges'** as a part of manual feature elimination.

Interpreting the Model
Refer to the above image, i.e. the final summary statistics after completing manual feature elimination. Now suppose you are a data analyst working for the telecom company, and you want to compare two customers, customer A and customer B. For both of them, the value of the variables tenure, PhoneService, Contract_One year, etc. are all the same, except for the variable **PaperlessBilling**, which is equal to **1 for customer A** and **0 for customer B**.

In other words, customer A and customer B have the exact same behaviour as far as these variables are concerned, except that customer A opts for paperless billing, and customer B does not. Now use this information to answer the following questions.

Now that we have a final model, we can begin with model evaluation and making predictions. We'll start doing that in the next session.

Coming Up
For now, let's summarise your learnings in this session in the next segment. We'll start with model evaluation separately in the next session.

Summary:-

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

$$P = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_3+...+\beta_n X_n)}}$$

The example used for building the multivariate model in Python was the **telecom churn example**. Basically, you learnt how Python can be used to decide the probability of a customer churning based on the value of 21 predictor variables such as monthly charges, paperless billing, etc.

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

1. Missing value imputation
2. Outlier treatment
3. Dummy variable creation for categorical variables
4. Test-train split of the data
5. Standardisation of the scales of continuous variables

After all of this was done, a logistic regression model was built in Python using the function **GLM()** under statsmodel library. This model contained all the variables, some of which had insignificant coefficients. Hence, some of these variables were removed first based on an automated approach, i.e. RFE and then a manual approach based on VIF and p-value.

Apart from this, you also learnt about confusion matrix and accuracy and saw how accuracy was calculated for a logistic regression model.

Module 3 : Multivariate Logistic Regression - Model Evaluation

Introduction:-

Welcome to the session on '**Multivariate Logistic Regression (Model Evaluation)**'.

In this session

In this session, you will first learn about a few more metrics beyond accuracy that are essential to evaluate the performance of a logistic regression model. Then based on these metrics, you'll learn how to find out the optimal scenario where the model will perform the best. The metrics that you'll learn about are:

- Accuracy
- Sensitivity, specificity and the ROC curve
- Precision and Recall

Finally, once you've chosen the optimal scenario based on the evaluation metrics, you'll finally go on and make predictions on the test dataset and see how your model performs there as well.

Metrics Beyond Accuracy: Sensitivity & Specificity:-

In the previous session, you built a logistic regression model and arrived at the final set of features using RFE and manual feature elimination. You got an accuracy of about **80.475%** for the model. But the question now is - Is accuracy enough to assess the goodness of the model? As you'll see, the answer is a big **NO!**

To understand why accuracy is often not the best metric, consider this business problem -

"Let's say that increasing 'churn' is the most serious issue in the telecom company, and the company desperately wants to retain customers. To do that, the marketing head decides to roll out discounts and offers to all customers who are likely to churn - ideally, not a single 'churn' customer should be missed. Hence, it is important that the model identifies almost all the 'churn' customers correctly. It is fine if it incorrectly predicts some of the 'non-churn' customers as 'churn' since in that case, the worst that will happen is that the company will offer discounts to those customers who would anyway stay."

Let's take a look at the confusion matrix we got for our final model again - the actual labels are along the column while the predicted labels are along the rows (for e.g. 595 customers are actually 'churn' but predicted as 'not-churn'):

Actual/Predicted	Not Churn	Churn
Not Churn	3269	366
Churn	595	692

From the table above, you can see that there are **595 + 692 = 1287** actual 'churn' customers, so ideally the model should predict all of them as 'churn' (i.e. corresponding to the business problem above). But out of these 1287, the current model only predicts 692 as 'churn'. Thus, only 692 out of 1287, or **only about 53% of 'churn' customers**, **will be predicted by the model as 'churn'**. This is very risky - the company won't be able to roll out

offers to the rest 47% ‘churn’ customers and they could switch to a competitor!

So although the accuracy is about 80%, the model only predicts 53% of churn cases correctly.

In essence, what’s happening here is that you care more about one class (class='churn') than the other. This is a very common situation in classification problems - you almost always care more about one class than the other. On the other hand, the accuracy tells you model's performance on both classes combined - which is fine, but not the most important metric.

Consider another example - suppose you're building a logistic regression model for cancer patients. Based on certain features, you need to predict whether the patient has cancer or not. In this case, if you incorrectly predict many diseased patients as 'Not having cancer', it can be very risky. In such cases, it is better that instead of looking at the overall accuracy, you care about predicting the 1's (the diseased) correctly.

Similarly, if you're building a model to determine whether you should block (where blocking is a 1 and not blocking is a 0) a customer's transactions or not based on his past transaction behaviour in order to identify frauds, you'd care more about getting the 0's right. This is because you might not want to wrongly block a good customer's transactions as it might lead to a very bad customer experience.

Hence, it is very crucial that you consider the **overall business problem** you are trying to solve to decide the metric you want to maximise or minimise.

This brings us to two of the most commonly used metrics to evaluate a classification model:

- Sensitivity
- Specificity

Let's understand these metrics one by one. **Sensitivity** is defined as:

$$Sensitivity = \frac{Number\ of\ actual\ Yeses\ correctly\ predicted}{Total\ number\ of\ actual\ Yeses}$$

Here, 'yes' means 'churn' and 'no' means 'non-churn'. Let's look at the confusion matrix again.

Actual/Predicted	Not Churn	Churn
Not Churn	3269	366
Churn	595	692

The different elements in this matrix can be labelled as follows:

Actual/Predicted	Not Churn	Churn
Not Churn	True Negatives	False Positives
Churn	False Negatives	True Positives

1. The first cell contains the actual 'Not Churns' being predicted as 'Not-Churn' and hence, is labelled **'True Negatives'** (Negative implying that the class is '0', here, Not-Churn.).
2. The second cell contains the actual 'Not Churns' being predicted as 'Churn' and hence, is labelled **'False Positive'** (because it is predicted as 'Churn' (Positive) but in actuality, it's not a Churn).
3. Similarly, the third cell contains the actual 'Churns' being predicted as 'Not Churn' which is why we call it **'False Negative'**.
4. And finally, the fourth cell contains the actual 'Churns' being predicted as 'Churn' and so, it's labelled as **'True Positives'**.

Now, to find out the sensitivity, you first need the number of actual Yeses correctly predicted. This number can be found in the last row and the last column of the matrix (which is denoted as true positives). This number if **692.** Now, you need the total number of actual Yeses. This number will be the sum of the numbers present in the last row, i.e. the actual number of churns (this will include the actual churns being wrongly identified as not-churns, and the actual churns being correctly identified as churns). Hence, you get **(595 + 692) = 1287**.

Now, when you replace these values in the sensitivity formula, you get:

$$Sensitivity = \frac{692}{1287} \approx 53.768\%$$

Thus, you can clearly see that although you had a high accuracy (~80.475%), your sensitivity turned out to be quite low (~53.768%)

Now, similarly, **specificity** is defined as:

$$Specificity = \frac{Number\ of\ actual\ Nos\ correctly\ predicted}{Total\ number\ of\ actual\ Nos}$$

As you can now infer, this value will be given by the value **True Negatives (3269)** divided by the actual number of negatives, i.e. **True Negatives + False Positives (3269 + 366 = 3635)**. Hence, by replacing these values in the formula, you get specificity as:

$$Specificity = \frac{3269}{3635} \approx 89.931\%$$

Coming Up
In the next section, you will learn how to evaluate these metrics using Python.

Sensitivity and Specificity in Python:-

In the last segment, you learnt the importance of having evaluation metrics other than accuracy. Thus, you were introduced to two new metrics - **sensitivity** and **specificity**. You learnt the theory of sensitivity and specificity and how to calculate them using a confusion matrix. Now, let's learn how to calculate these metrics in Python as well.

As you saw in the code, you can access the different elements in the matrix using the following indexing -

TP = confusion[**1**,**1**] # true positive
TN = confusion[**0**,**0**] # true negatives
FP = confusion[**0**,**1**] # false positives
FN = confusion[**1**,**0**] # false negatives

And now, let's rewrite the formulas of sensitivity and specificity using the labels of the confusion matrix.

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives}$$

Coming Up
So your model seems to have **high accuracy (~80.475%)** and **high specificity (~89.931%)**, but **low sensitivity (~53.768%)** and since you're interested in identifying the customers which might churn, you clearly need to deal with this. You'll learn how to do this in the next section.

Understanding ROC Curve:-

So far you have learned about some evaluation metrics and saw why they're important to evaluate a logistic regression model. Now, recall that the sensitivity that you got **(~53.768%)** was quite low and clearly needs to be dealt with. But what was the cause of such a low sensitivity in the first place?

If you remember, when you assigned 0s and 1s to the customers after building the model, you arbitrarily chose a cut-off of **0.5** wherein if the probability of churning for a customer is greater than 0.5, you classified it as a 'Churn' and if the probability of churning for a customer is less than 0.5, you classified it as a 'Non-churn'.

Now, this cut-off was chosen at random and there was no particular logic behind it. So it might not be the ideal cut-off point for classification which is why we might be getting such a low sensitivity. So how do you find the ideal cutoff point? Let's start by watching the following video. For a more intuitive understanding, this part has been demonstrated in Excel. You can download the excel file from below and follow along with the lecture.

So you saw that the predicted labels depend entirely on the cutoff or the threshold that you have chosen. For low values of threshold, you'd have a higher number of customers predicted as a 1 (Churn). This is because if the threshold is low, it basically means that everything above that threshold would be one and everything below that threshold would be zero. So naturally, a lower cutoff would mean a higher number of customers being identified as 'Churn'. Similarly, for high values of threshold, you'd have a higher number of customer predicted as a 0 (Not-Churn) and a lower number of customers predicted as a 1 (Churn).

Now, let's move forward with our discussion on how to choose an optimal threshold point. For that, you'd first need a few basic terminologies (some of which you have seen in earlier sections.). So let's hear what these terminologies are.

So you learned about the following two terminologies -

True Positive Rate (TPR)
This value gives you the number of positives correctly predicted divided by the total number of positives. Its formula as shown in the video is:

$$True\ Positive\ Rate\ (TPR) = \frac{True\ Positives}{Total\ Number\ of\ Actual\ Positives}$$

Now, recall the labels in the confusion matrix,

Actual/Predicted	Not Churn	Churn
Not Churn	True Negatives	False Positives
Churn	False Negatives	True Positives

As you can see, the highlighted portion shows the row containing the total number of actual positives.

Therefore, the denominator term, i.e. in the formula for TPR is nothing but -

$$Total\ Number\ of\ Actual\ Positives = True\ Positives + False\ Negatives$$

So, the formula for True Positive Rate (TPR) becomes -

$$True\ Positive\ Rate\ (TPR) = \frac{True\ Positives}{True\ Positives + False\ Negatives} = \frac{TP}{TP+FN}$$

As you might remember, the above formula is nothing but the formula for **sensitivity**. Hence, the term True Positive Rate that you just learnt about is nothing but sensitivity.

The second term which you saw was -

False Positive Rate (FPR)

This term gives you the number of false positives (0s predicted as 1s) divided by the total number of negatives. The formula was -

$$False\ Positive\ Rate\ (FPR) = \frac{False\ Positives}{Total\ Number\ of\ Actual\ Negatives}$$

Again, using the confusion matrix, you can easily see that the denominator here is nothing but the first row. Hence, it can be written as -

$$Total\ Number\ of\ Actual\ Negatives = True\ Negatives + False\ Positives$$

Therefore, the formula now becomes -

$$False\ Positive\ Rate\ (FPR) = \frac{False\ Positives}{True\ Negatives + False\ Positives} = \frac{FP}{TN+FP}$$

Again, if you recall the formula for specificity, it is given by -

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

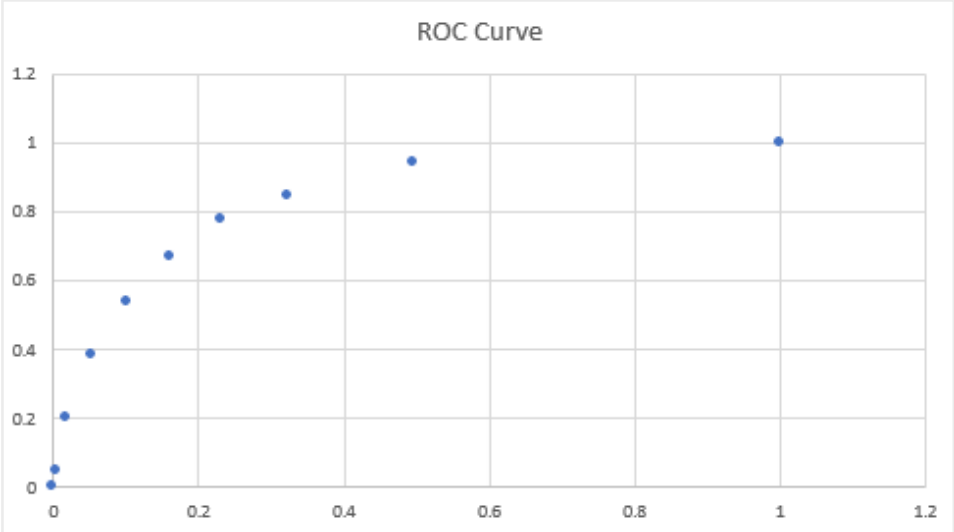
Hence, you can see that the formula for False Positive Rate (FPR) is nothing but (**1 - Specificity**). You can easily verify it yourself.

So, now that you have understood what these terms are, you'll now learn about **Receiver Operating Characteristic (ROC) Curves** which show the **tradeoff between the True Positive Rate (TPR) and the False Positive Rate (FPR)**. And as was established from the formulas above, TPR and FPR are nothing but sensitivity and (1 - specificity), so it can also be looked at as a tradeoff between sensitivity and specificity.

In the next video, you will see how the cutoff varies as you change the values of TPR and FPR for each row. You will also plot the values of TPR against FPR to understand your model better.

So you can clearly see that there is **a tradeoff between the True Positive Rate and the False Positive Rate, or simply, a tradeoff between sensitivity and**

specificity. When you plot the true positive rate against the false positive rate, you get a graph which shows the trade-off between them and this curve is known as the ROC curve. The following image shows the ROC curve that you plotted in Excel.



As you can see, for higher values of TPR, you will also have higher values of FPR, which might not be good. So it's all about finding a balance between these two metrics and that's what the ROC curve helps you find. You also learnt that a good ROC curve is the one which touches the upper-left corner of the graph; so higher the area under the curve of an ROC curve, the better is your model.

You'll learn more on ROC curves in the coming segments but first, try out some questions.

Coming Up
Now that you're familiar with the ROC curve, in the next segment, you will learn how to plot an ROC curve in Python.

ROC Curve in Python:-

Now that you have learnt the theory of ROC curve, let's plot an ROC curve in Python for our telecom churn case study

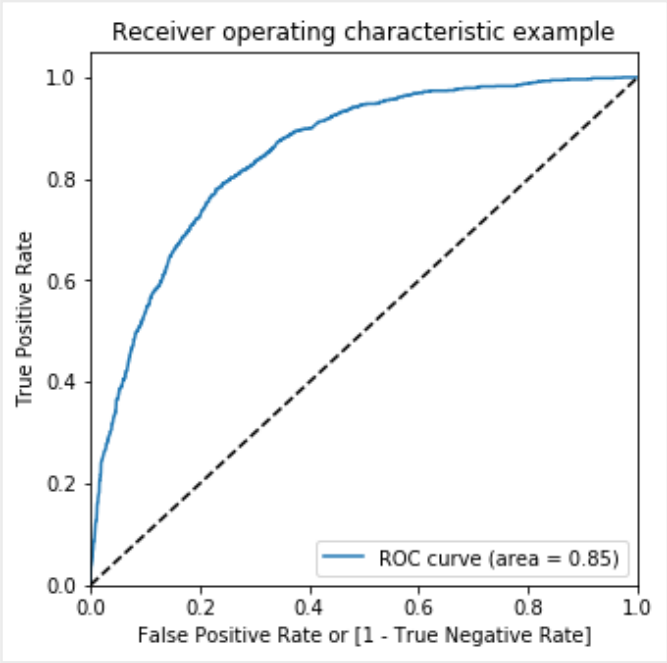
Let's first take a look at the ROC curve code that you just saw:

```
# Defining the function to plot the ROC curve
def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()

    return None

# Calling the function
draw_roc(y_train_pred_final.Churn, y_train_pred_final.Churn_Prob)
```

Notice that in the last line you're giving the actual Churn values and the respective Churn Probabilities to the curve.

Interpreting the ROC Curve

Following is the ROC curve that you got. Note that it is the same curve you got in Excel as well but that was using scatter plot to represent the discrete points and here we are using a continuous line.



The 45 degree **Diagonal**
For a completely random model, the ROC curve will pass through the 45-degree line that has been shown in the graph above and in the best case it passes through the upper left corner of the graph. So the least area that an ROC curve can have is 0.5, and the highest area it can have is 1.

The Sensitivity vs Specificity Trade-off
As you saw in the last segment as well, the ROC curve shows the trade-off between True Positive Rate and False Positive Rate which essentially can also be viewed as a tradeoff between Sensitivity and Specificity. As you can see, on the Y-axis, you have the values of Sensitivity and on the X-axis, you have the value of (1 - Specificity). Notice that in the curve when Sensitivity is increasing, (1 - Specificity), And since, (1 - Specificity) is increasing, it simply means that Specificity is decreasing.

Area Under the Curve
By determining the area under the curve (AUC) of an ROC curve, you can determine how good the model is. If the ROC curve is more towards the upper-left corner of the graph, it means that the model is very good and if it is more towards the 45-degree diagonal, it means that the model is almost completely random. So, the larger the AUC, the better will be your model which is something you saw in the last segment as well.

Coming Up
Now that you've seen that there is clearly a trade-off between sensitivity and specificity, you'll learn how to decide the optimal threshold in the next segment.

Additional Reading
The following link provides a very interesting insight into the ROC curve.

Finding the Optimal Threshold:-

In the last segment, you saw that the ROC curve essentially shows you a trade-off between the sensitivity and specificity. But how do you find the optimal threshold in order to get a decent accuracy, sensitivity, as well as specificity? Let's hear what Rahim has to say.

So, first Rahim calculated the values of accuracy, sensitivity, and specificity at different cut-off values and stored them in a dataframe using the code below:

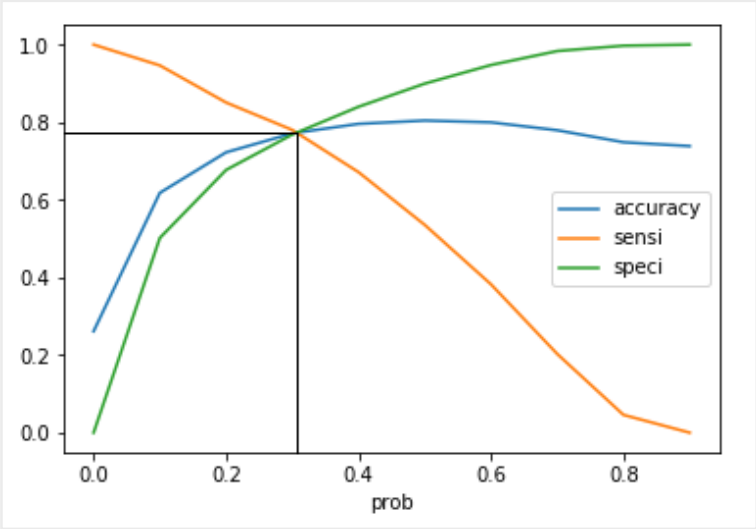
```
# Now let's calculate accuracy sensitivity and specificity for various probability cutoffs.
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
from sklearn.metrics import confusion_matrix

# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = metrics.confusion_matrix(y_train_pred_final.Churn, y_train_pred_final[i] )
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

The key takeaway from this code is the accuracy, sensitivity, and specificity values which have been calculated using the appropriate elements in the confusion matrix. The code outputted the following dataframe:

	prob	accuracy	sensi	speci
0.0	0.0	0.261479	1.000000	0.000000
0.1	0.1	0.619667	0.946387	0.503989
0.2	0.2	0.722674	0.850039	0.677579
0.3	0.3	0.771434	0.780109	0.768363
0.4	0.4	0.795002	0.671329	0.838790
0.5	0.5	0.804754	0.537685	0.899312
0.6	0.6	0.800284	0.385392	0.947180
0.7	0.7	0.779764	0.205128	0.983219
0.8	0.8	0.749289	0.050505	0.996699
0.9	0.9	0.738521	0.000000	1.000000

As you can see, when the probability thresholds are very low, the sensitivity is

very high and specificity is very low. Similarly, for larger probability thresholds, the sensitivity values are very low but the specificity values are very high. And at about 0.3, the three metrics seem to be almost equal with decent values and hence, we choose 0.3 as the optimal cut-off point. The following graph also showcases that at about 0.3, the three metrics intersect.



As you can see, at about a threshold of 0.3, the curves of accuracy, sensitivity and specificity intersect, and they all take a value of around 77-78%.

Now, as Rahim mentioned, you could've chosen any other cut-off point as well based on which of these metrics you want to be high. If you want to capture the 'Churns' better, you could have let go of a little accuracy and would've chosen an even lower cut-off and vice-versa. It is completely dependent on the situation you're in. In this case, we just chose the 'Optimal' cut-off point to give you a fair idea of how the thresholds should be chosen.

Coming Up
In the next segment, you'll perform an exercise using the concepts you've just learnt.

Model Evaluation Metrics - Exercise:-

These questions are **non-graded**.

In the last segment, you saw that the optimal cut-off for the model turned out to be 0.3. Now, in order to assess the model, you need to re-run the predictions on the train set using the cut-off of 0.3. Recall when you did this using the cut-off of 0.5, you used the following code:

```
# Creating new column 'predicted' with 1 if Churn_Prob > 0.5 else 0
y_train_pred_final['predicted'] = y_train_pred_final.Churn_Prob.map(lambda x: 1 if x > 0.5 else 0)
```

Now, using the same code with just the cut-off value changed, calculate all the metrics (accuracy, sensitivity, specificity) and answer the questions below.

If you want the solution to these questions, the code has been provided in the Logistic Regression Jupyter Notebook that you downloaded. It is present below the plotted trade-off curve, although it is recommended that you write the code and attempt these questions on your own before looking at the given code.

Coming Up
In the next segment, you'll learn a few more evaluation metrics that are used in the industry.

Precision and Recall:-

So far you learnt about sensitivity and specificity. You learnt how these metrics are defined, why they are important, and how to calculate them. Now, apart from sensitivity and specificity, there are two more metrics that are widely used in the industry which you should know about. They're known as **'Precision'** and **'Recall'**. Now, these metrics are very similar to sensitivity and specificity; it's just that knowing the exact terminologies can be helpful as both of these pairs of metrics are often used in the industry. So let's first hear Rahim introduce these metrics.

Note: At 5:12, professor said at lower threshold precision is high and recall is low ,it should be low precision and high recall.

Let's go through the definitions of precision and recall once again:

- **Precision:** Probability that a predicted 'Yes' is actually a 'Yes'.

True/Predicted	No	Yes
No	TN	FP
Yes	FN	TP

The formula for precision can be given as:

$$Precision = \frac{TP}{TP+FP}$$

Remember that 'Precision' is the same as the 'Positive Predictive Value' that you learnt about earlier. From now on, we will call it precision.

- **Recall:** Probability that an actual 'Yes' case is predicted correctly.

True/Predicted	No	Yes
No	TN	FP
Yes	FN	TP

The formula for recall can be given as:

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

Remember that 'Recall' is exactly the same as sensitivity. Don't get confused between these.

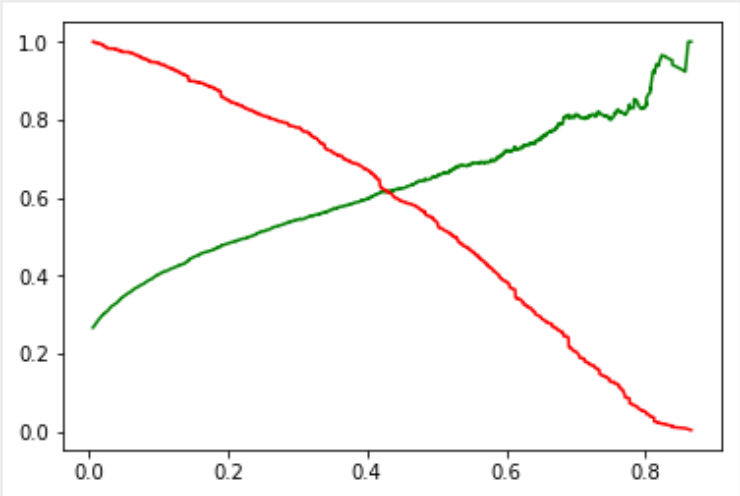
You might be wondering, if these are almost the same, then why even study them separately? The main reason behind this is that in the industry, some businesses follow the 'Sensitivity-Specificity' view and some other businesses follow the 'Precision-Recall' view and hence, will be helpful for you if you know both these standard pairs of metrics.

Now, let's check the precision and recall in code as well.

As Rahim said, whatever view you select might give you different interpretations for the same model. It is completely up to you which view you choose to take while building a logistic regression model.

Now, recall that Rahim had told that similar to sensitivity and specificity, there is a trade-off between precision and recall as well.

So similar to the sensitivity-specificity tradeoff, you learnt that there is a tradeoff between precision and recall as well. Following is the tradeoff curve that you plotted:



As you can see, the curve is similar to what you got for sensitivity and specificity. Except now, the curve for precision is quite jumpy towards the end. This is because the denominator of precision, i.e. (TP + FP) is not constant as these are the predicted values of 1s. And because the predicted values can swing wildly, you get a very jumpy curve.

Coming Up
Now that you have evaluated the model using the train dataset, you can finally move to making predictions on the test dataset in the next segment.

Making Predictions:-

So the model evaluation on the train set is complete and the model seems to be doing a decent job. You saw two views of the evaluation metrics - one was the sensitivity-specificity view, and the other was the precision-recall view. You can choose any of the metrics you like; it is completely up to you.

In this session, we will go forward with the sensitivity-specificity view of things and make predictions based on the 0.3 cut-off that we decided earlier. Note that now we are making predictions on the test set, so we have to choose one threshold that we determined during the training phase. Now, you can choose either the sensitivity-specificity view (where the cut-off came to be 0.3) or the precision-recall view (where the cut-off came to be 0.42) when making predictions. Here in the demonstration, we have chosen the sensitivity-specificity view and hence are going with the 0.3 cut-off determined during the training phase. In the notebook, we have taken the cut-off as 0.42 to show the precision-recall view. You are free to adjust the cut-off as per your choice even though both the approaches would yield slightly different results

Note: At 2:45, Rahim mistakenly says, "On the main model we had accuracy about 79%." Please note that he said this by mistake. It should be 77% as you had calculated in the 'Model Evaluation Metrics - Exercise' as well.

The metrics seem to hold on the test dataset as well. So, it looks like you have created a decent model for the churn dataset as the metrics are decent for both the training and test datasets.

You can also take the cutoff you got from the precision-recall tradeoff curve and try making predictions based on that on your own.

Coming Up

In the next segment, let's summarise all your learnings in the last two sessions.

Summary:-

Let's hear Rahim summarise the last two sessions for you.

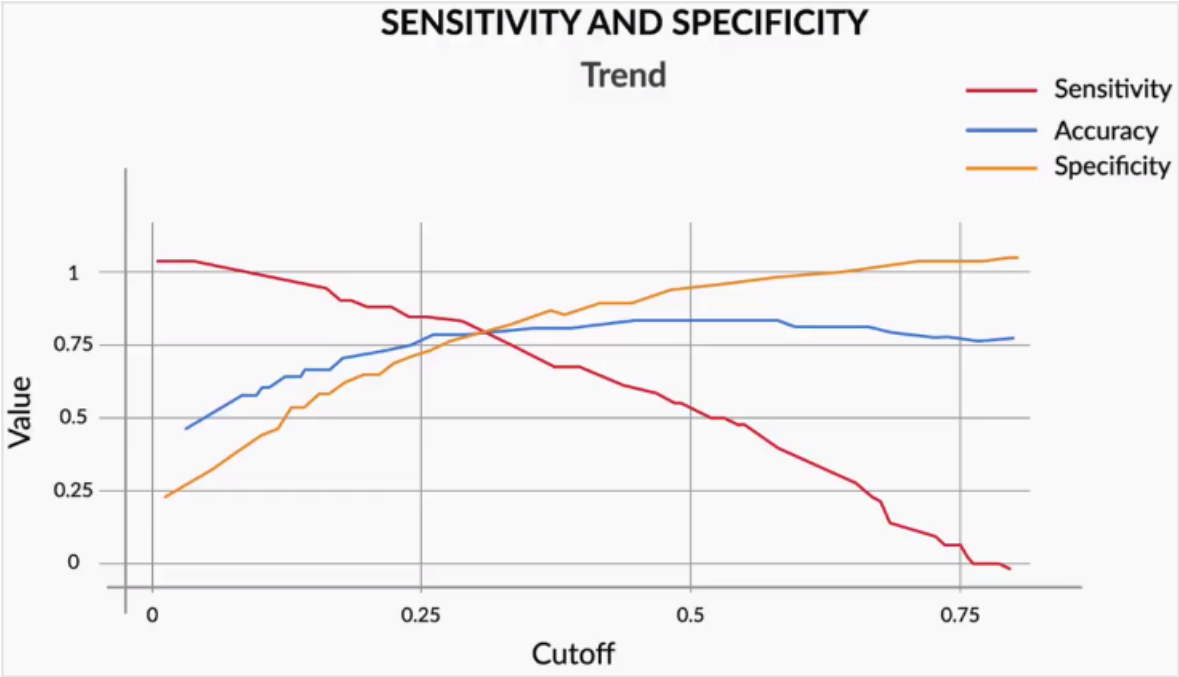
So to summarise, the steps that you performed throughout the model building and model evaluation were:

1. Data cleaning and preparation
    - Combining three dataframes
    - Handling categorical variables
        - Mapping categorical variables to integers
        - Dummy variable creation
    - Handling missing values
2. Test-train split and scaling
3. Model Building
    - Feature elimination based on correlations
    - Feature selection using RFE (Coarse Tuning)
    - Manual feature elimination (using p-values and VIFs)
4. Model Evaluation
    - Accuracy
    - Sensitivity and Specificity
    - Optimal cut-off using ROC curve
    - Precision and Recall
5. Predictions on the test set

So first, classes were assigned to all the customers in the test data set. For this, a probability **cutoff** of 0.5 was used. The model thus made, was very accurate (Accuracy = ~80%), but it had a very low **sensitivity** (~53%). Thus, a different cutoff was tried out, i.e. 0.3, which resulted in a model with slightly lower accuracy (~77%), but a much better sensitivity (~78%). Hence, you learnt 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.

Also, recall that the sensitivity of a model is the proportion of yeses (or positives) correctly predicted by it as yeses (or positives). Also, the **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 by 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. But it totally depends on the business context. Sometimes you might want a higher sensitivity, sometimes you might want a higher specificity.

You also saw another view of things which was the Precision and Recall view. Those were very much related to sensitivity and specificity. Precision essentially means of the 'Yeses' predicted, how many were actually yeses. Recall on the other hand is that same as sensitivity, i.e. out of the total actual yeses, how many did you correctly predict.

Module 4 : Logistic Regression - Industry Applications - Part I

Introduction:-

Welcome to the session on "Industry Applications of Logistic Regression". In the previous sessions, you learnt the process of building a logistic regression

model in Python and evaluating its performance using a few key metrics.

In this session

You will learn how to use the concepts you learnt earlier in actual business settings. Broadly speaking, the agenda for the session is as follows:

1. Types of logistic regression
2. Nuances of logistic regression
    1. Sample selection
    2. Segmentation
    3. Variable transformation

Getting familiar with Logistic Regression:-

So far, you have understood how logistic regression works and how performance measures can be evaluated. Now that you've acquired the **theoretical knowledge** of this technique, let's move on to its **application** since it is equally important. So, let’s understand the various applications of logistic regression in different business scenarios across multiple industries from our industry expert Hindol Basu.

In general, logistic regression by definition tries to predict what state a particular individual or system will be in the future. You learnt about the two **types of logistic regression**:

1. **Binary logit**
2. **Multinomial logit**

Binary logit involves two levels of the dependent variable. For example, the telecom churn example you learnt in earlier sessions is a binary logistic regression problem, as it classifies customers into two levels, churns and non-churns. **Multinomial logit**, however, involves more than 2 levels of dependent variables, such as whether a customer will purchase product A, product B or not purchase anything.

So, the rule of thumb for deciding whether the problem is a binary classification problem or multinomial classification problem is that you should first understand the dependent variable.

Let’s see some more examples and try to understand the importance of logistic regression in detail.

To summarise, logistic regression is a widely used technique in various types of industries. This is because of two **main reasons**:

1. It is very easy to **understand** and offers an **intuitive explanation** of the variables
2. The output (i.e. the probabilities) has a linear relationship with the log of odds, which can be very useful for explaining results to managers

Also, recall that Hindol mentioned something called model scores. In an earlier session though, you learnt that a logistic regression model gives log odds as output. So, to understand what scores are, let's go back to the telecom churn example from earlier sessions:

customerID	Probability	Odds	Log Odds	Score
8773-HHUOZ	0.084	0.092	-2.389	331
8865-TNMNX	0.257	0.346	-1.062	369
9867-JCZSP	0.297	0.422	-0.862	375
9420-LOJKX	0.435	0.770	-0.261	392
6234-RAAPL	0.439	0.783	-0.245	393
7760-OYPDY	0.443	0.795	-0.229	393
8012-SOUDQ	0.446	0.805	-0.217	394
3413-BMNZE	0.461	0.855	-0.156	395
6575-SUVOI	0.688	2.205	0.791	423
6388-TABGU	0.753	3.049	1.115	432

You must have noticed the column called score. Basically, it's a different way of reporting your findings. Earlier, you saw that log odds make more sense as the output instead of probabilities because of their linear relationship with the variables. However, log odds have weird values, such as -0.245, -0.156 etc., which is not a very elegant form of output.

Hence, instead of reporting the log odds as output, you can report scores.

Score is calculated using the following expression:

$$Score = 400 + (20 * \frac{log(odds)}{log(2)})$$

This **expression** is decided based on business understanding. You could come up with your own expression for the score, one that converts log odds into a more presentable form.

So, in this lecture, you understood the uses and importance of logistic regression. Next, we'll go through some nuances of logistic regression.

Nuances of Logistic Regression - Sample Selection:-

In earlier sessions, you learnt how to build a logistic regression model in Python, and how to evaluate it. However, even before you start building a model, you have to decide what kind of data would be appropriate for building it.

Let's listen to Hindol on what nuances you should keep in mind while doing so:

So, to summarise, 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:

1. **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.
2. The sample should be **representative of the population** on which the model will be applied in the future.
3. For **rare events samples**, the sample should be balanced before it is used for modelling.

So, these were the nuances of sample selection. Are there any other nuances you need to be aware of? Yes. In the next segment, you will learn about the various nuances of segmentation.

Nuances of Logistic Regression - Segmentation:-

You learnt the nuances of sample selection in the last lecture. However, suppose the model you built, after selecting the data with all due considerations, has a low accuracy. Here, you know that the model is not performing well on the chosen sample data set. Your task is to make a model which gives a decent model performance.

So, assuming that you cannot take another sample, what can you do to increase the model's performance? There are various ways of handling such problems in industries, and one of them is the segmentation of the population. Let's learn how to perform a population segmentation in detail from Hindol Basu**.**

Hence, it is very helpful to perform segmentation of the population before building a model.

Let's talk about the ICICI example again.

For students and salaried people, different variables may be important. While students' defaulting and not defaulting will depend on factors such as program enrolled for, the prestige of the university attended, parents' income, etc., the probability of salaried people will depend on factors such as 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.

Use the dataset given below, to answer the question that follows-

So with that, we've looked at the various aspects to take care of while selecting data (samples) for building a model. In the next lecture, you will learn which aspects you should take care of when you talk about variables.

Additional reading

- Guide to building better predictive models using segmentation

Nuances of Logistic Regression - Variable Transformation-I:-

In the last lecture, you learnt about segmentation, which is primarily done for increasing the predictive power of a model. However, so far you've only seen topics such as sample selection and segmentation, which talk about the data that is used in the **model building process**.

Now, the next steps, as you may recall from the last session, are dummy variable creation, standardising scales of continuous variables, etc. These processes are generally referred to as variable transformation. Can other types of variable transformations be performed before building a logistic regression model?

Let's hear from Hindol about that.

From earlier sessions, 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, using the techniques mentioned by Hindol in the video above.

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**.

Nuances of Logistic Regression - Variable Transformation-II:-

So, creating dummy variables is one way of transforming variables. Let's now move on to another technique commonly used for transforming variables — **Weight of evidence (WOE) analysis**.

So, to summarise, you learnt **three important** things in this lecture:

1. Calculating **woe values** for fine binning and coarse binning
2. The **importance** of woe for fine binning and coarse binning
3. The **usage** of woe transformation

WOE can be calculated using the following equation:

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

Or, it can be expressed as:

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

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 the WOE values again.

As mentioned in the lecture, there are two main advantages of WOE:

1. 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.
2. 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.

Let's also understand the positives and negatives of woe transformation from Hindol.

So, basically, the pros and cons of a WOE transformation are similar to dummy variables.

1. Pros: The model becomes more stable because small changes in the continuous variables will not impact the input so much.
2. 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 be 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.

Comprehension 1: WOE and Information Value Analysis

You are required to **download the data set** from below for answering the questions that follow:

In the attached file, there are three sheets. The first sheet contains three variables (Tenure, Second Contract and Churn) from the telecom data. The second sheet contains the distribution of the binned tenure variable. The third sheet contains the distribution of goods and bad information of the contract variable.

With this information, let’s try out the following questions:

Nuances of Logistic Regression - Variable Transformation-III:-

Note: You can access the solution to the **Comprehension 1** questions from below:

In previous sessions, you learnt about the different ways of transforming woe transformation, the importance of information variables, and various other advantages and disadvantages of woe transformation.

You learnt how these methods can be used to treat **continuous variables**. In the next video. let's explore it a little more.

This is how you can work on continuous variables.

In the next lecture, you will learn about some advanced transformation techniques such as spline transformation, interaction variables, mathematical transformation and principal component transformation. These transformations are hardly used in the model exercise. However, let's see what the benefits and importance of these transformations are in the next lecture.

Comprehension 2: Missing Value -WOE

You saw that NA values can be treated with WOE values. However, you can replace the NA bucket with a bucket which shows similar woe values.

Let's try to practice this with some examples. For this exercise, you are supposed to **download the data set** from below which is the subset data of the loan file:

In this file, there are two sheets. The first one contains the loan data set with only two variables — **Employment length** and **Default**. The second file contains the bucket distributions of employment length.

With this information, attempt the questions given below.

Nuances of Logistic Regression - Variable Transformation-IV (Optional):-

Note: You can access the solution to the **Comprehension 2 questions** from below:

Please note that the content on this page is purely optional and you will not be graded on this content, nor is it required for the logistic regression case study. ***You can skip through this page if needed and directly move to the next page.***

However, if you wish to go through the content on the page, you are welcome to do so. You should keep in mind, though, that the videos on this page assume that the viewer has knowledge of some advanced concepts.

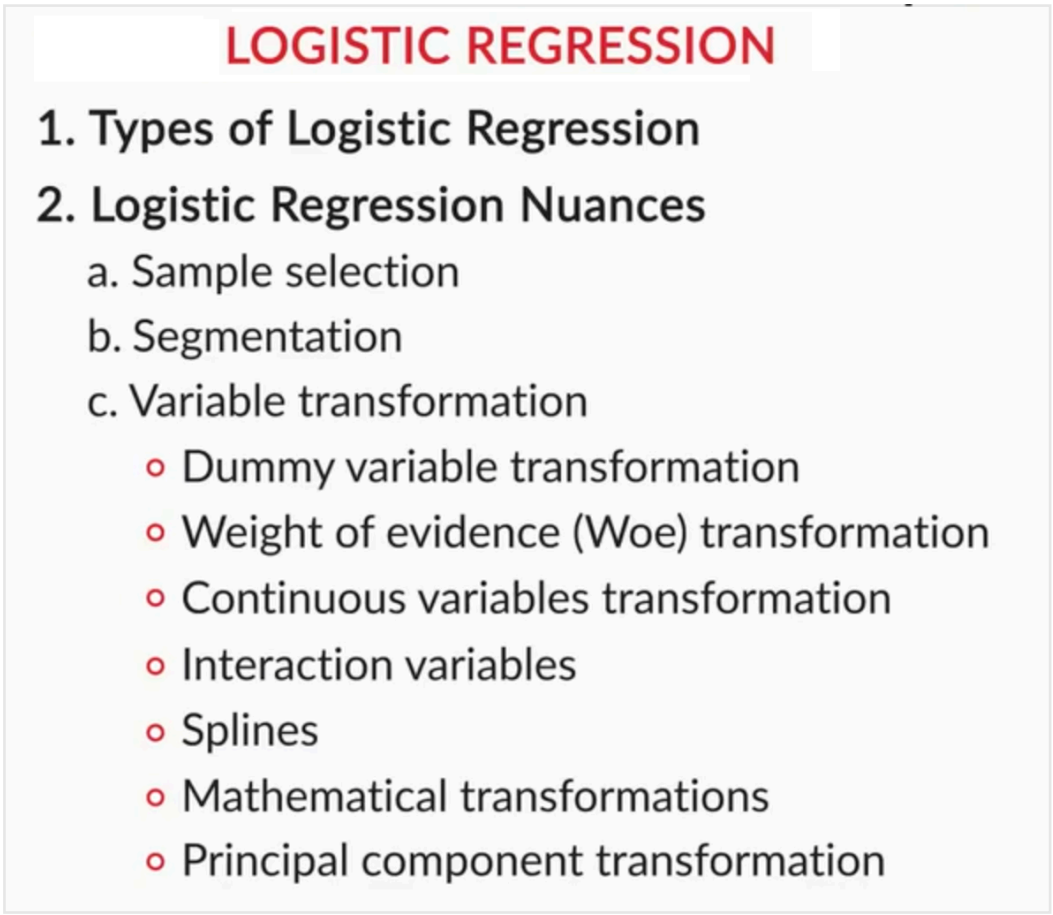
The page can be accessed from this link.

Summary:-

Note: You can access the solution of the graded questions from the link below:

In this session, you learnt the industry relevance of logistic regression and its applicability. Apart from its applicability, you learnt new techniques such as sample selection, segmentation and variable transformation in order to improve the model performance.

Here is the **flowchart**, which will help you understand what we have covered so far in this session.



Module 5 : Logistic Regression - Industry Applications - Part II

Commonly Faced Challenges in Implementation of Logistic Regression:-

Let's go through some more challenges faced regularly by data analysts while building a logistic regression model.

Let's look back at the credit card example from earlier sessions.

Let's look at 2 cases, where the variable's value is missing (equal to NA):

1. **Utilisation is missing:** As mentioned earlier, if this variable is missing

for a particular customer, that could very well be because the bank did not find that customer worthy enough for a credit card. Hence, these missing values are not missing at random, and it would be unfair to just replace them with the mean or the median. As mentioned earlier, it would be wiser to perform a WOE analysis and then replace these values.

2. **Age is missing:** Consider why the variable age is missing for some customers. Here, it may actually make more sense to just replace the missing value with the mean or the median, instead of wasting time on WOE analysis. This is because it is very likely that the variable age is just missing because of a system error or a manual error, and there is no clear pattern behind the missing values.

Also, recall that Hindol mentioned that there are two more methods for treating missing values. Those were not taught in this course because of their complexity. However, if you wish to, you can go through both of them here:

1. Markov Chain Monte Carlo - SAS Support
2. Expectation Maximisation - Oxford University Statistics

Model Evaluation (A Second Look):-

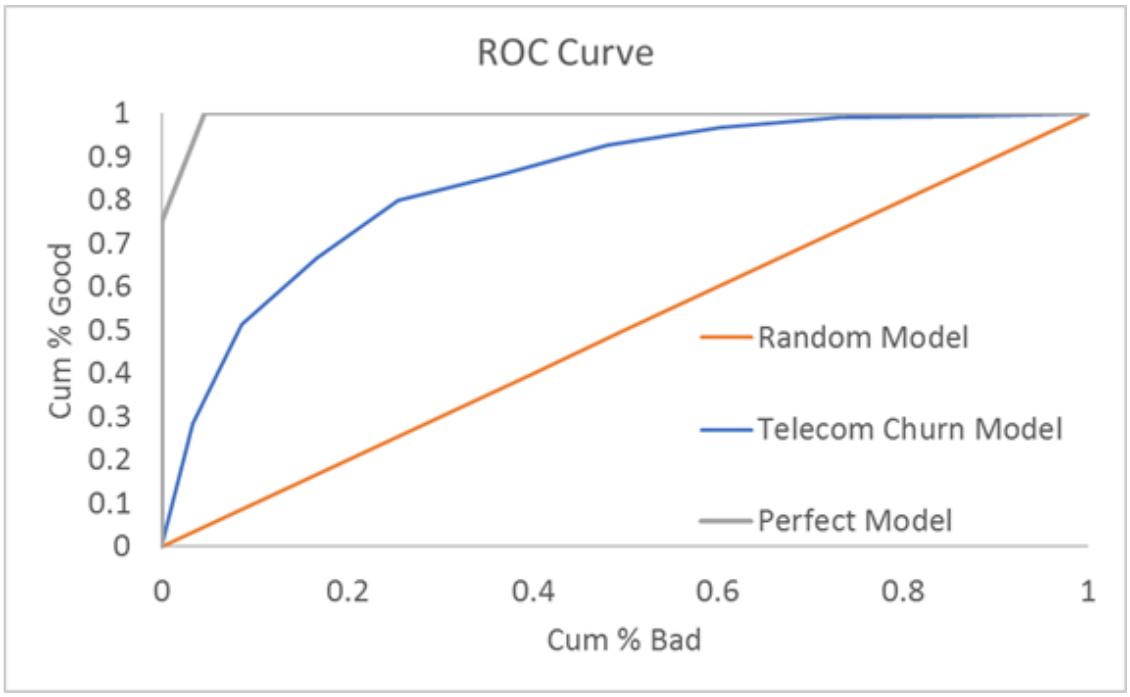
In a previous session on model evaluation, you learnt about various model evaluation measures, such as accuracy, sensitivity, specificity, KS statistic, etc. Now, let's look at some more measures commonly used for model evaluation.

Note: At 02:29 our SME's says "%Goods in X-axis and %Bad in Y-axis", please note that it should be %Bad in X-axis(FPR) and %Good in Y-axis(TPR).

Hindol also talked about some measures other than sensitivity and specificity. If you want to go deep and learn about them, you can access them through the following links:

- KS Statistic
- Rank Ordering

The following shows the ROC curve for 3 cases, i.e. the random model (orange line), the ideal model (grey line), and the real model(blue line), in our case, the telecom churn model.



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, which is nothing but a fancy word and is given by -

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

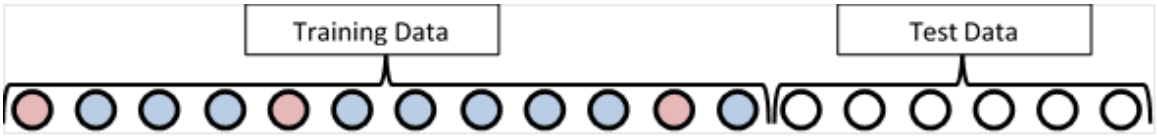
will be high for a good model.

Model Validation and Importance of Stability:-

So, you've seen how a model is evaluated, using various parameters such as accuracy, sensitivity, KS statistic, gini coefficient, etc. However, so far you've only tested models on data that was formed after splitting one data set into training data and testing data. Is that enough? Let's see.

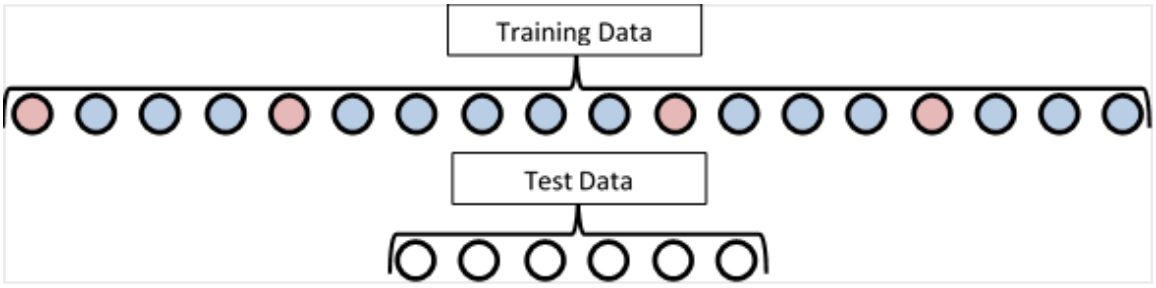
Let's again look at the telecom churn example from before. The data used to build the model was from 2014.

You split the original data into two parts, i.e. training and testing data. However, these two parts were both built with data from 2014.

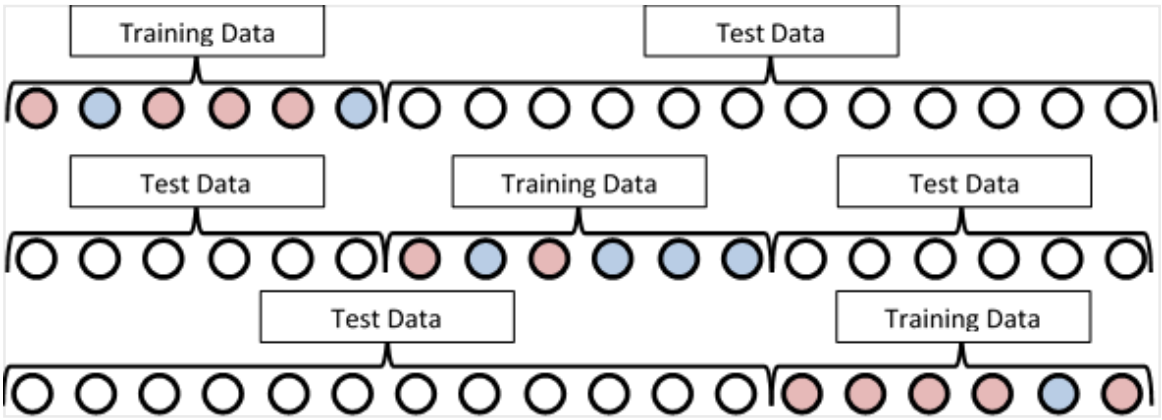


This is called in-sample validation. Testing your model on this test data may not be enough though, as the test data here is too similar to training data.

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.



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:



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 the module on Model Selection.

*Note for **0:43 to 2:07:** While explaining how WOE is linked to stability and PSI, Hindol used IV values mistakenly. He meant to use WOE values.*

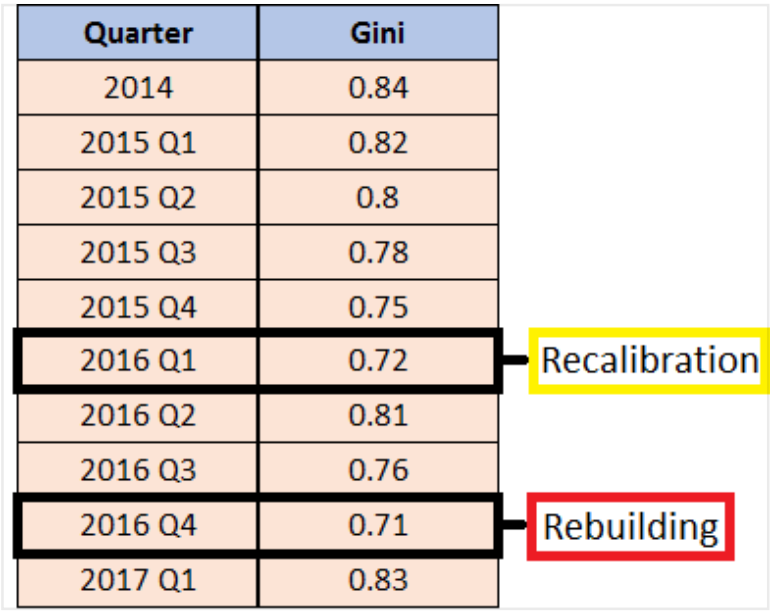
Obviously, a good model will be stable. A model is considered stable if it has:

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

Tracking of Model Performance Over Time:-

Now, suppose you conclude, based on the previously mentioned criteria, that the model is not stable. What will you do then? Let's hear from Hindol on that.

Let's go back to the telecom churn example. If you recall, the model was built using **data from 2014**. Now suppose, you are tracking its performance over time, and that ends up giving you the following results:

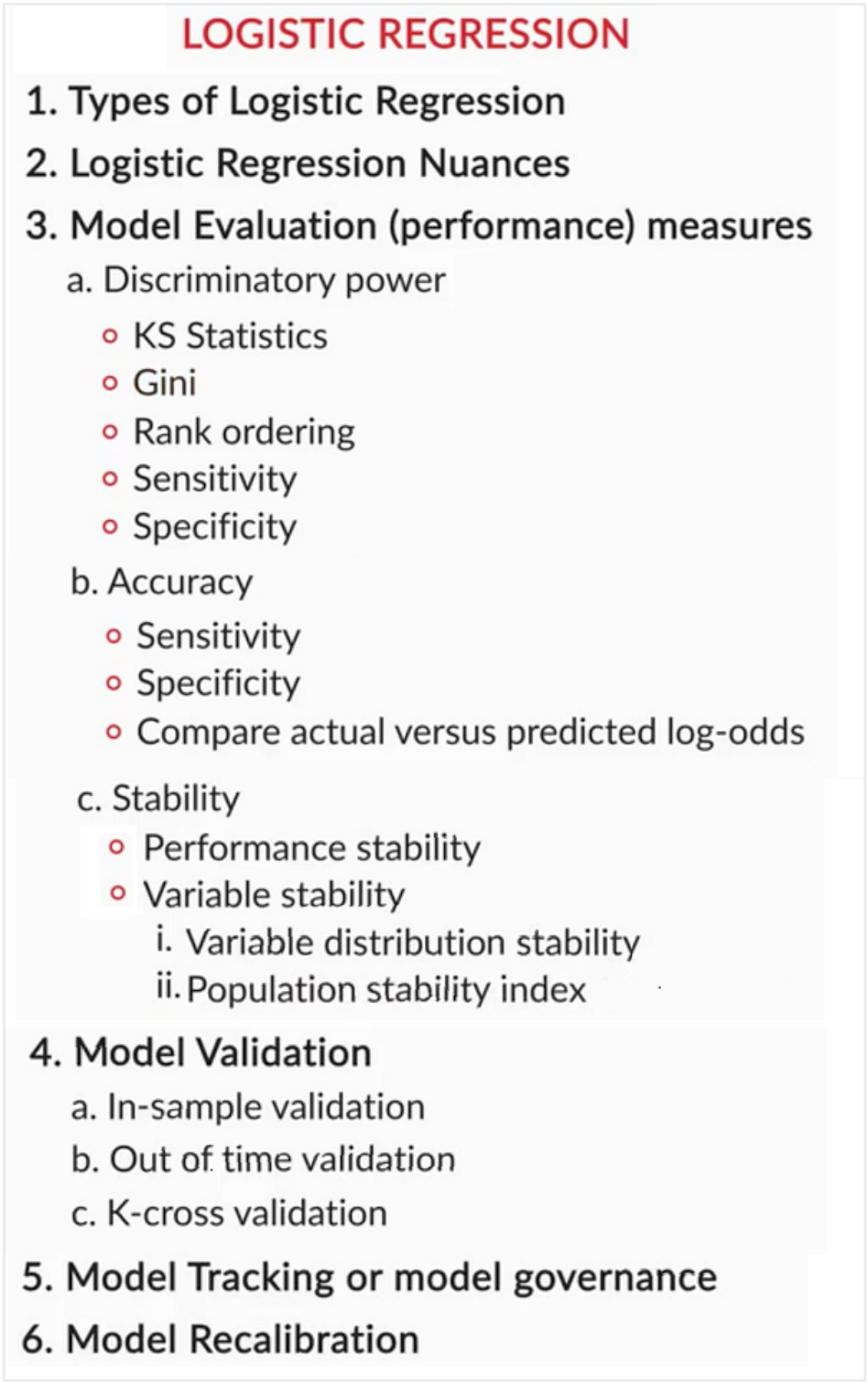


So, the **first time**, when the model's Gini dropped to **0.72**, you avoided building a new model. Basically, you just **recalibrated**, i.e. updated the coefficients of the variables. That resulted in a slight increase of Gini. However, the next time Gini dropped to a low value, i.e. **0.71**, we just rebuilt the model, i.e. got new sample data, performed data prep, etc. and built the entire model.

Summary:-

In this session, you learnt about various model evaluation measures, such as accuracy, sensitivity, specificity, KS statistic, etc as shown in figure 1. Then, you

learnt how to validate your model on in-sample data and out-time data (unseen data). Also, for stability, you learnt and understood stability by two metrics — Variable Distributive Stability and Predictive Pattern Stability. Then, you learnt about model tracking and model recalibrating steps in detail.



You can also download the lecture notes for the logistic regression module below.
There's also an additional module on **'Logistic Regression Interview Questions'**. Feel free to visit it anytime on this link.

Subjective Questions - I:-

The following questions related to logistic regression are asked quite frequently in interviews.

Q1. What is a logistic function? What is the range of values of a logistic function?

The logistic function is as defined below:

$$f(z) = \frac{1}{(1+e^{-z})}$$

The values of a logistic function will range from 0 to 1. The values of Z will vary from $-\infty$ to $+\infty$.

Q2. Why is logistic regression very popular/widely used?

Logistic regression is famous because it can convert the values of logits (log-odds), which can range from $-\infty$ to $+\infty$ to a range between 0 and 1. As logistic functions output the probability of occurrence of an event, it can be applied to many real-life scenarios. It is for this reason that the logistic regression model is very popular. Another reason why logistic fairs in comparison to linear regression is that it is able to handle the categorical variables.

Q3. What is the formula for the logistic regression function?

In general, the formula for logistic regression is given by the following expression:

$$f(z) = \frac{1}{(1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_k X_k)})}$$

Q4. How can the probability of a logistic regression model be expressed as a conditional probability?

The conditional probability can be given as:

$$P(Discrete\ value\ of\ target\ variable|X_1, X_2, X_3 \ldots . X_k)$$

It is the probability of the target variable to take up a discrete value (either 0 or 1 in case of binary classification problems) when the values of independent variables are given. For example, the probability an employee will attrite (target variable) given his attributes such as his age, salary, KRA's, etc.

Q5. What are odds?

It is the ratio of the probability of an event occurring to the probability of the event not occurring. For example, let's assume that the probability of winning a lottery is 0.01. Then, the probability of not winning is 1 - 0.01 = 0.99.

Now, as per the definition,

The odds of winning the lottery = (Probability of winning)/(Probability of not winning)
The odds of winning the lottery = 0.01/0.99

Hence, the odds of winning the lottery is 1 to 99, and the odds of not winning the lottery is 99 to 1

Q6. Why can't linear regression be used in place of logistic regression for binary classification?

The reasons why linear regressions cannot be used in case of binary classification are as follows:

Distribution of error terms: The distribution of data in the case of linear and logistic regression is different. Linear regression assumes that error terms are normally distributed. In the case of binary classification, this assumption does not hold true.

Model output: In linear regression, the output is continuous. In the case of binary classification, an output of a continuous value does not make sense. For binary classification problems, linear regression may predict values that can go beyond 0 and 1. If we want the output in the form of probabilities, which can be mapped to two different classes, then its range should be restricted to 0 and 1. As the logistic regression model can output probabilities with logistic/sigmoid function, it is preferred over linear regression.

Variance of Residual errors: Linear regression assumes that the variance of random errors is constant. This assumption is also violated in the case of logistic regression

Q7. What is the likelihood function?

The likelihood function is the joint probability of observing the data. For example, let's assume that a coin is tossed 100 times and you want to know the probability of getting 60 heads from the tosses. This example follows the binomial distribution formula.

p = Probability of heads from a single coin toss

n = 100 (the number of coin tosses)

x = 60 (the number of heads – success)

n - x = 40 (the number of tails)

Pr (X=60 | n = 100, p)

The likelihood function is the probability that the number of heads received is 60 in a trail of 100 coin tosses, where the probability of heads received in each coin toss is p. Here the coin toss result follows a binomial distribution.

This can be reframed as follows:

Pr(X=60|n=100, p) = $c \times p^{60} \times (1-p)^{100-60}$

c = constant

p = unknown parameter

The likelihood function gives the probability of observing the results using unknown parameters.

Q8. What are the outputs of the logistic model and the logistic function?

The logistic model outputs the logits, i.e. log odds; and the logistic function outputs the probabilities.

$$Logistic\ model = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_n X_n$$

The output of the same will be logits.

$$Logistic\ function = f(z) = \frac{1}{(1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_3+\ldots+\beta_n X_n)})}$$

The output, in this case, will be the probabilities

Q9. How to interpret the results of a logistic regression model? Or, what are the meanings of the different betas in a logistic regression model?

β_0 is the baseline in a logistic regression model. It is the log odds for an instance when all the attributes $(X_1, X_2, X_3, \ldots, X_n)$ are zero. In practical scenarios, the probability of all the attributes being zero is very low. In another interpretation, β_0 is the log odds for an instance when none of the attributes is taken into consideration.

All the other Betas are the values by which the log odds change by a unit change in a particular attribute by keeping all other attributes fixed or unchanged (control variables).

Q10. What is odds ratio?

Odds ratio is the ratio of odds between two groups. For example, let's assume that you are trying to ascertain the effectiveness of a medicine. You administered this medicine to the 'intervention' group and a placebo to the 'control' group.

$$Odds\ Ratio\ (OR) = \frac{Odds\ of\ the\ Intervention\ Group}{Odds\ of\ the\ Control\ Group}$$

Interpretation

- If odds ratio = 1, then there is no difference between the intervention group and the control group.
- If the odds ratio is greater than 1, then the odds of the intervention group is greater than the odds of the control group.
- If the odds ratio is less than 1, then the odds of the intervention group is smaller than the odds of the control group.

Q11. What is the formula for calculating odds ratio?

The formula can be given as:

$$OR_{X_1,X_0} = e^{\sum_{i=1}^{k} \beta_i (X_{1i} - X_{0i})}$$

In the formula above, X1 and Xo stand for two different groups for which the odds ratio needs to be calculated. X1*i* stands for the instance '*i*' in group X1. Xo*i* stands for the instance '*i*' in group Xo.βo stands for the coefficient of the logistic regression model. Note that the baseline is not included in this formula.

Subjective Questions - II:-

The following questions are related to the maximum likelihood estimator that are asked frequently in interviews.

Q1. What is the Maximum Likelihood Estimator (MLE)?

The MLE chooses those sets of unknown parameters (estimator) that maximise the likelihood function. The method to find the MLE is to use calculus and setting the derivative of the logistic function with respect to an unknown parameter to zero, and solving it will give the MLE. For a binomial model, this will be easy, but for a logistic model, the calculations are complex. Computer programs are used for deriving MLE for logistic models.

(*Here's another approach to answering the question.*)

MLE is a statistical approach to estimate the parameters of a mathematical model. MLE and ordinary square estimation give the same results for linear regression if the dependent variable is assumed to be normally distributed. MLE does not assume anything about independent variables.

Q2. What are the different methods of MLE and when is each method preferred?

In the case of logistic regression, there are two approaches to MLE. They are conditional and unconditional methods. Conditional and unconditional methods are algorithms that use different likelihood functions. The unconditional formula employs the joint probability of positives (for example, churn) and negatives (for example, non-churn). The conditional formula is the ratio of the probability of observed data to the probability of all possible configurations.

The unconditional method is preferred if the number of parameters is lower compared to the number of instances. If the number of parameters is high compared to the number of instances, then conditional MLE is to be preferred. Statisticians suggest that conditional MLE is to be used when in doubt. Conditional MLE will always provide unbiased results.

Q3. What are the advantages and disadvantages of conditional and unconditional methods of MLE?

Conditional methods do not estimate unwanted parameters. Unconditional methods estimate the values of unwanted parameters also. Unconditional formulas can directly be developed with joint probabilities. This cannot be done with conditional probability. If the number of parameters is high relative to the number of instances, then the unconditional method will give biased results. Conditional results will be unbiased in such cases.

Q4. What is the output of a standard MLE program?

The output of a standard MLE program is as follows:

Maximised likelihood value: This is the numerical value obtained by replacing the unknown parameter values in the likelihood function with the MLE parameter estimator.

Estimated variance-covariance matrix: The diagonal of this matrix consists of the estimated variances of the ML estimates. The off-diagonal consists of the covariances of the pairs of the ML estimates

Q5. Why can't we use Mean Square Error (MSE) as a cost function for logistic regression?

In logistic regression, we use the sigmoid function and perform a non-linear transformation to obtain the probabilities. Squaring this non-linear transformation will lead to non-convexity with local minimums. Finding the global minimum in such cases using gradient descent is not possible. Due to this reason, MSE is not suitable for logistic regression. Cross-entropy or log loss is used as a cost function for logistic regression. In the cost function for logistic regression, the confident wrong predictions are penalised heavily. The confident right predictions are rewarded less. By optimising this cost function, convergence is achieved.

Subjective Questions - III:-

The following interview questions are based on the evaluation metrics used in logistic regression.

Q1. What is accuracy?

Accuracy is the number of correct predictions out of all predictions made.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Number\ of\ Predictions}$$

Q2. Why is accuracy not a good measure for classification problems?

Accuracy is not a good measure for classification problems because it gives equal importance to both true positives and true negatives. However, this may not be the case in most business problems. For example, in the case of cancer prediction, declaring cancer as benign is more serious than wrongly informing the patient that he is suffering from cancer. Accuracy gives equal importance to both cases and cannot differentiate between them.

Q3. What is the importance of a baseline in a classification problem?

Most classification problems deal with imbalanced datasets. Examples include telecom churn, employee attrition, cancer prediction, fraud detection, online advertisement targeting, and so on. In all these problems, the number of the positive classes will be very low when compared to the negative classes. In some cases, it is common to have positive classes that are less than 1% of the total sample. In such cases, an accuracy of 99% may sound very good but, in reality, it may not be.

Here, the negatives are 99%, and hence, the baseline will remain the same. If the algorithms predict all the instances as negative, then also the accuracy will be 99%. In this case, all the positives will be predicted wrongly, which is very important for any business. Even though all the positives are predicted wrongly, an accuracy of 99% is achieved. So, the baseline is very important, and the algorithm needs to be evaluated relative to the baseline.

Q4. What are false positives and false negatives?

False positives are those cases in which the negatives are wrongly predicted as positives. For example, predicting that a customer will churn when, in fact, he is not churning.

False negatives are those cases in which the positives are wrongly predicted as negatives. For example, predicting that a customer will not churn when, in fact, he churns.

Q5. What are the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR)?

TPR refers to the ratio of correctly classified positives to the total number of positive instances in the data.

$$TPR = \frac{TP}{TP+FN}$$

TNR refers to the ratio of correctly classified negatives to the total number of negative instances in the data.

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

FPR refers to the ratio of negatives in the data which are incorrectly classified as positives to the total number of negative instances in the data.

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

FNR refers to the ratio of positive instances in the data which are incorrectly classified as negatives to the total number of positive instances in the data.

$$FNR = \frac{FN}{TP+FN}$$

Q6. What are sensitivity and specificity?

Specificity is the same as true negative rate, or it is equal to 1 – false positive rate. It tells you out of all the actual '0' labels, how many were correctly predicted.

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

Sensitivity is the true positive rate. It tells you out of all the actual '1' labels, how many were correctly predicted.

$$Sensitivity = \frac{TP}{TP+FN}$$

Q7. What are precision and recall?

Precision is the proportion of true positives out of predicted positives. To put it in another way, it is the accuracy of the prediction. It is also known as the 'positive predictive value'.

$$Precision = \frac{TP}{TP+FP}$$

Recall is the same as the true positive rate (TPR) or the sensitivity.

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

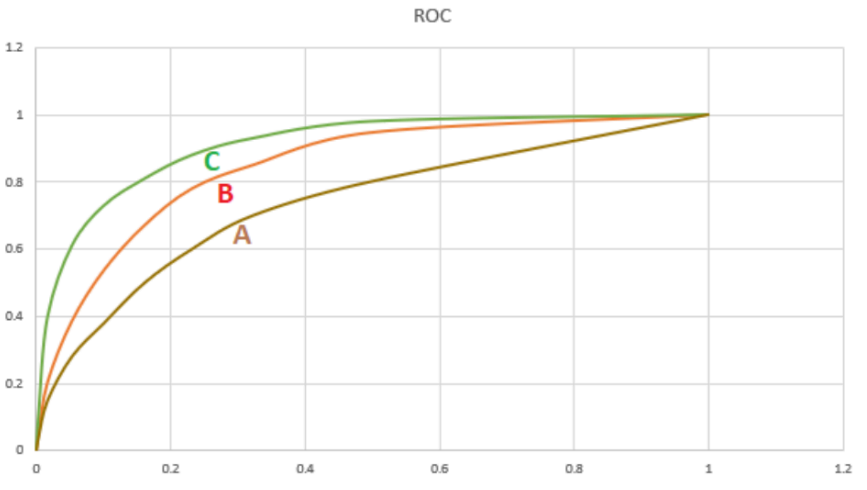
Q8. What is F-measure?

It is the harmonic mean of precision and recall. In some cases, there will be a trade-off between the precision and the recall. In such cases, the F-measure will drop. It will be high when both the precision and the recall are high. Depending on the business case at hand and the goal of data analytics, an appropriate metric should be selected.

$$F-measure = 2 \times \frac{Precision \times Recall}{Precision+Recall}$$

Q9. Explain the use of ROC curves and the AUC of an ROC Curve.

An ROC (Receiver Operating Characteristic) curve illustrates the performance of a binary classification model. It is basically a TPR versus FPR (true positive rate versus false positive rate) curve for all the threshold values ranging from 0 to 1. In an ROC curve, each point in the ROC space will be associated with a different confusion matrix. A diagonal line from the bottom-left to the top-right on the ROC graph represents random guessing. The Area Under the Curve (AUC) signifies how good the classifier model is. If the value for AUC is high (near 1), then the model is working satisfactorily, whereas if the value is low (around 0.5), then the model is not working properly and just guessing randomly. From the image below, curve C (green) is the best ROC curve among the three and curve A (brown) is the worst ROC curve among the three.



Q10. How to choose a cutoff point in case of a logistic regression model?

The cutoff point depends on the business objective. Depending on the goals of your business, the cutoff point needs to be selected. For example, let's consider loan defaults. If the business objective is to reduce the loss, then the specificity needs to be high. If the aim is to increase the profits, then it is an entirely different matter. It may not be the case that profits will increase by avoiding giving loans to all predicted default cases. But it may be the case that the business has to disburse loans to default cases that are slightly less risky to increase the profits. In such a case, a different cutoff point, which maximises profit, will be required. In most of the instances, businesses will operate around many constraints. The cutoff point that satisfies the business objective will not be the same with and without limitations. The cutoff point needs to be selected considering all these points. If the business context doesn't matter much and you want to create a balanced model, then you use an ROC curve to see the tradeoff between sensitivity and specificity and accordingly choose an optimal cutoff point where both these values along with accuracy are decent.

Module 6 : Multiclassification using Logistic Regression