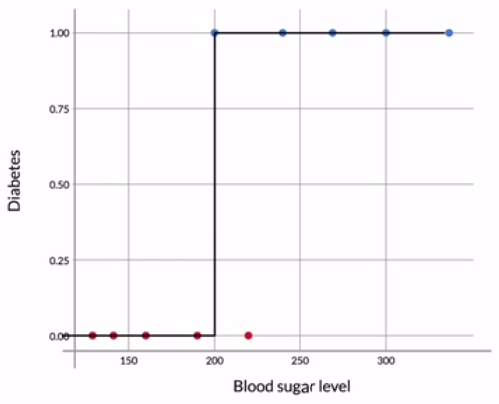
# **Univariate Logistic Regression**

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

**Binary Classification**

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?

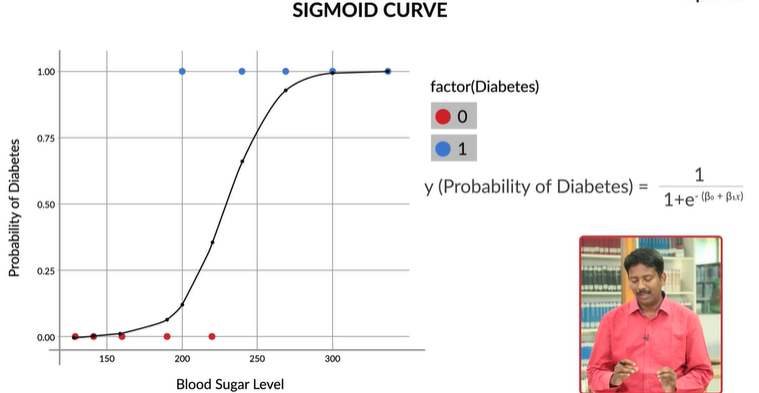


**Diabetes Data**

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.



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**.

**Sigmoid Curve**

This is the sigmoid curve equation: y=P(Diabetes)=1/1+e^−(β0+β1x). Here, let’s say you take β0 = -15 and β1 = 0.065. Now, what will be the probability of diabetes for a patient with sugar level 220?

0.33

**Feedback :**

Natural logarithms use the base **e = 2.71828**, so that given a number **e x**, its natural logarithm is **x**. For example, **e 3. 6888**is equal to 40, so that the natural logarithm of 40 is 3. 6888.

Here, the probability of diabetes for a person with sugar level x is given by P(Diabetes)=1/1+e^−(β0+β1x). Now, taking β0=−15 and β1=0.065, the probability of diabetes for a person with sugar level 220 will be given by P(Diabetes)=1/1+e^−(β0+β1x)

P=1/1+e^−(−15+0.065∗220)=0.33

**Sigmoid Curve**

For the sigmoid curve (β0 = -15 and β1 = 0.065), what will be the probability of diabetes for a patient with sugar level 240?

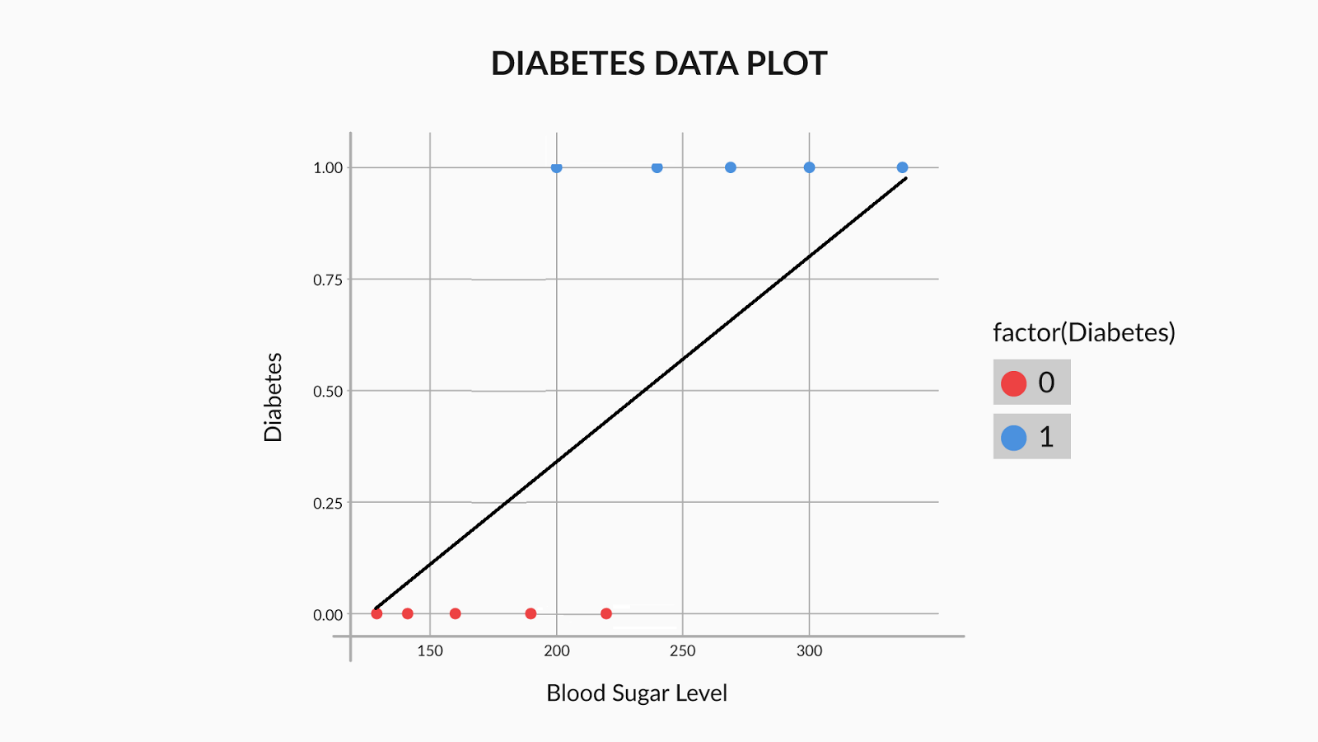
Top of Form

0.645

P=1/1+e^-(-15+0.065\*240) =~ 1/1+2.718^(-0.6) ~= 0.645

The Sigmoid curve actually has the properties 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.

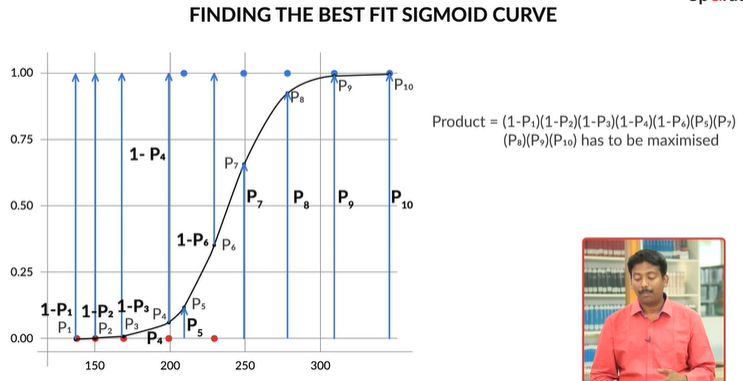


The main problem with a straight line is that it is not steep enough. The sigmoid curve has extremely low values in the start, some intermediate values in the middle and extremely high values towards the end. Also, it will have a boundary region after which the probabilities start increasing from low to high all of a sudden or vice versa. That boundary region is not present on a straight line as on a straight line, the values increase uniformly, there is no sudden increase or decrease.

**Finding the Best Fit Sigmoid Curve - I**

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

Lets see our data points and make then as P1,P2…P10. So we have 10 data points



the best fitting combination of β0 and β1 will be the one which maximises the product:

(1−P1)(1−P2)(1−P3)(1−P4)(1−P6)(P5)(P7)(P8)(P9)(P10)

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

[(1−Pi)(1−Pi)------ for all non-diabetics --------] \* [(Pi)(Pi) -------- 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 |

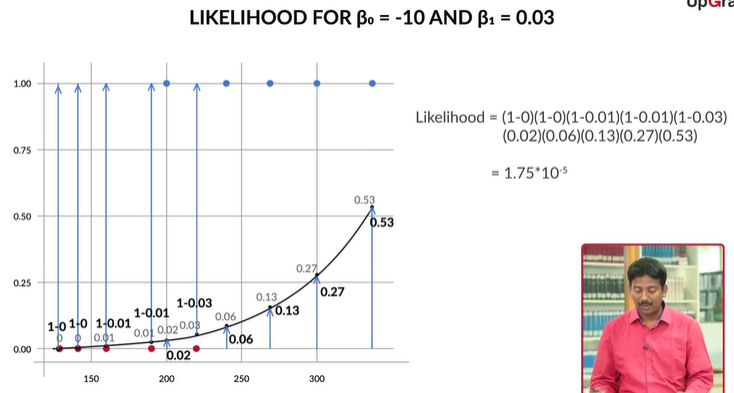
In this case, the likelihood would be equal to (1−P1)(1−P2)(1−P3)(1−P5)(1−P7)(P4)(P6)(P8)(P9)(P10)

**Finding the Best Fit Sigmoid Curve - II**

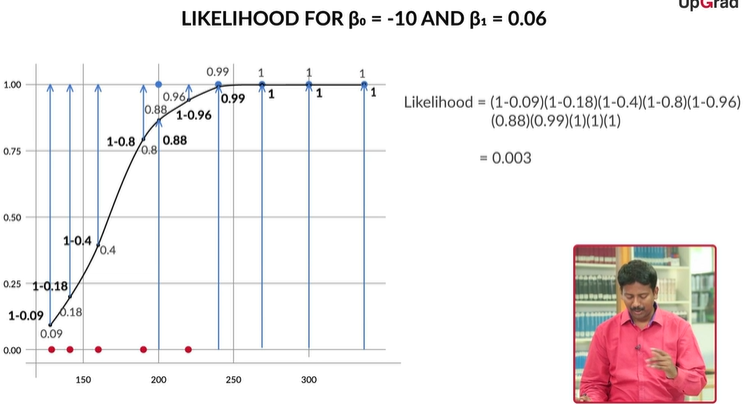
The likelihood function for our data is (1−P1)(1−P2)(1−P3)(1−P4)(1−P6)(P5)(P7)(P8)(P9)(P10) . The best fitting sigmoid curve would be the one which maximises the value of this product.

If you had to find β0 and β1 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  β0 and β1 until you find the combination that minimises the cost function.

For example if we choose β0=-10, β1=0.03



Another example choose β0=-10 and β1=0.06

****

**Maximum Likelihood Cost Function**

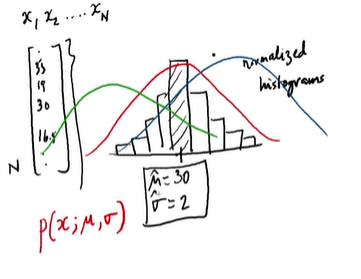
Here we are looking at variables which we are trying to model. The variables are of two types:

* Discrete
* Continuous

Also we have random variable whose values are determined by some probability distribution.

Suppose we are given n observations x1,x2….xn, we could lokk at the histogram of this distribution like below, and then we also normalize it so that we know what is the probability of a range of values can take.

So, we want to define a cost function for which we can say the Gaussian (normal) distribution (red one) is correct for say example μ = 30, σ = 2.



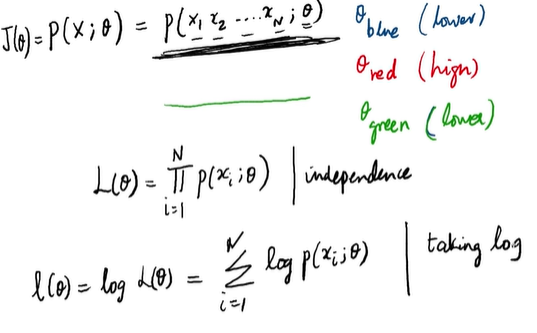
Let’s suppose we have observed n data points from some process. For example here, each data point represents the hair - length of the person.

For these data points, we’ll assume that the data generation process can be adequately described by a Gaussian (normal) distribution. As by visualising the data distribution (via Histogram),  a Gaussian distribution seems plausible because most of the points are accumulated in the middle with few points scattered to the left and the right.

As we know that the Normal Distribution has 2 parameters. ​The mean, μ, and the standard deviation, σ. Different values of these parameters will result in different curves as shown like blue, red or green one.

**Define such cost-function**

**J(theta) = P(X; theta) = P (X1,X2,….Xn, theta)** where the theta is the variable we want to find out.



Suppose we have n samples from independent and identically distributed observations, coming from an unknown probability density function f(x|theta), where theta is unknown.

So, how to arrive at the log-likelihood function? Here are the steps to sum it up:

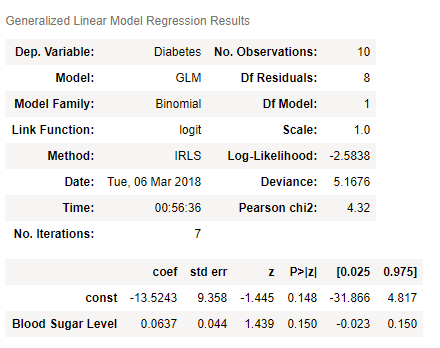
1. Making a Joint Density Function.
2. Finding the Likelihood function, x samples are fixed “parameters” and theta will be the function’s variable.
3. For further simplification, we make it Log-Likelihood function, as it's easier to deal with logs.

On practising machine learning we often use a model to describe the process, that results in the data that are observed. Each model contains its own set of parameters that ultimately defines what the model looks like.

Maximum likelihood estimation is the process of finding parameters for a given statistic which makes the known likelihood distribution a maximum.

**Odds and Log Odds**

The summary of the diabetes patients data set model is given below:



In the summary shown above, 'const' corresponds to β0 and Blood Sugar Level, i.e. 'x1' corresponds to β1. So, β0 = -13.5 and β1 = 0.06.

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

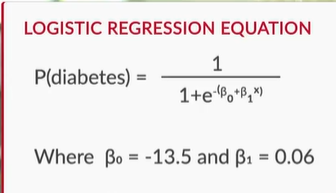
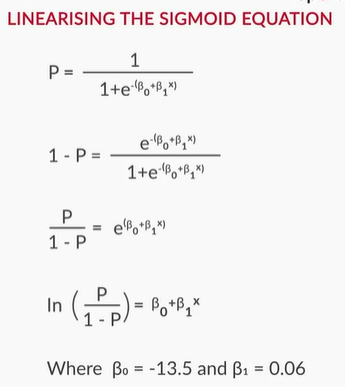
P=1/1+e^−(β0+β1x)

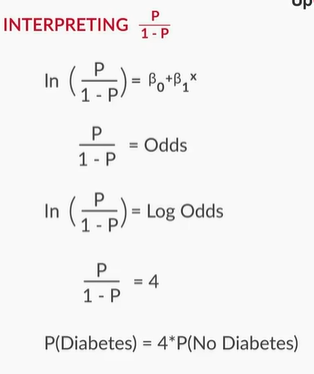
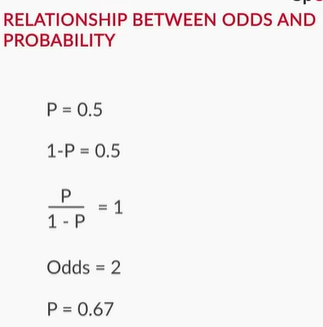
Recall that this equation gives the relationship between P, the probability of diabetes and x, the patient’s blood sugar level.

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

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

**Log Odds**

So, let’s say that the equation for log odds is:

ln(P/1−P)=−13.5+0.06x

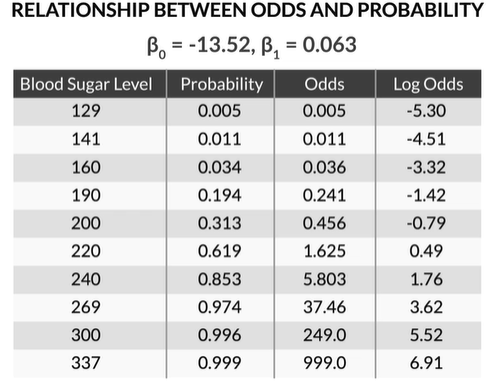
For x = 220, the log odds are equal to -0.3 and for x = 231.5, the log odds are equal to 0.39. For x = 243, the log odds are equal to:

Top of Form

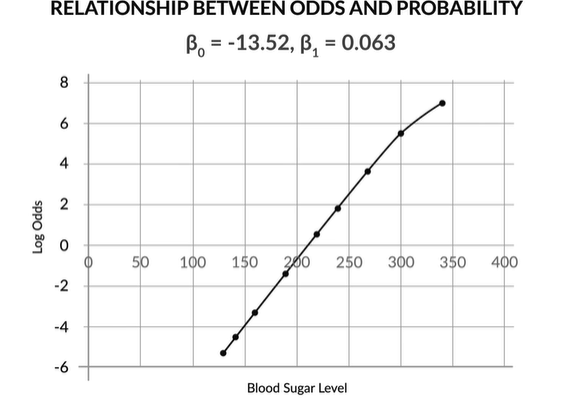
done**Correct**

**Feedback :**For a given value of x, the log odds are equal to -13.5 + 0.06x. Putting in the value of x here, i.e. 243, you get that the log odds = 1.08. However, you can actually directly guess the answer, without any calculations. The last time you increased x by 11.5, i.e. from 220 to 231.5, the log odds increased by 0.69, i.e. from -0.3 to 0.39. Since the relationship between x and log odds is linear, when you increase x by 11.5 again to make it 243, the log odds will increase by 0.69 again to get to 1.08.

Bottom of Form

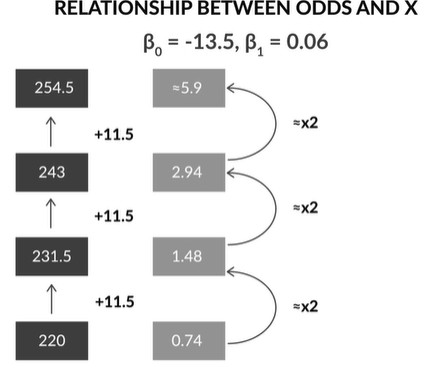


Now, if we plot the log odds v/s blood sugar level we find its actually a linear model



Now, if we consider **odds** in exponential form and change the variable x lets see what happens

P/(1-P) = e^−(β0+β1x)



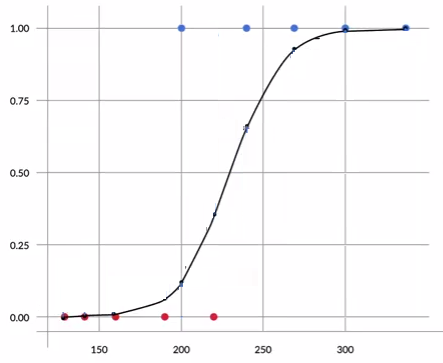
odds are in a **multiplicatively increasing pattern**

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.

# **Other forms of the Logistic Regression Equation**

So, the sigmoid curve a.k.a. the logit equation, is given by this equation - P=1/1+e^−(β0+β1x) It is a good choice for modeling probabilites, as its graph has this shape (extremely low values in the start, extremely high values in the end, intermediate ones in the middle):



**Sigmoid Curve**

However, it's not the only equation that has this form, there is also the probit form of logistic regression: P=ϕ−1(β0+β1x)

Also, there is the cloglog form of logistic regression: P=ln(−ln(1−(β0+β1x))). Both of these are also equations that can be used in logistic regression, as their graphs also give the same trend.

So, for logit, you would use -

links = sm.families.linkslogm = sm.GLM(y\_train,(sm.add\_constant(X\_train)), family=sm.families.Binomial(link=links.logit))logm.fit().summary()

Also, for probit, the code is -

links = sm.families.linkslogm = sm.GLM(y\_train,(sm.add\_constant(X\_train)), family=sm.families.Binomial(link=links.probit))logm.fit().summary()

Lastly, the code for cloglog equation are -

links = sm.families.linkslogm = sm.GLM(y\_train,(sm.add\_constant(X\_train)), family=sm.families.Binomial(link=links.cloglog))logm.fit().summary()

# **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=1/1+e^−(β0+β1X)

The above equation has only one feature variable X, for which the coefficient is β1. 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=1/1+e^−(β0+β1X1+β2X2+β3X3+...+βnXn)

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

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.

**Check ipython notebook ML-1/LogisticRegression/Telecom\_Churn/LogisticRegression-TelecomChurnCaseStudy.ipynb**

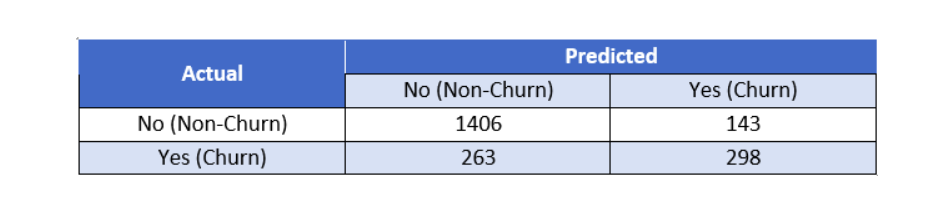
# **Confusion Matrix and Accuracy**

**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:



**Confusion Matrix**

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.

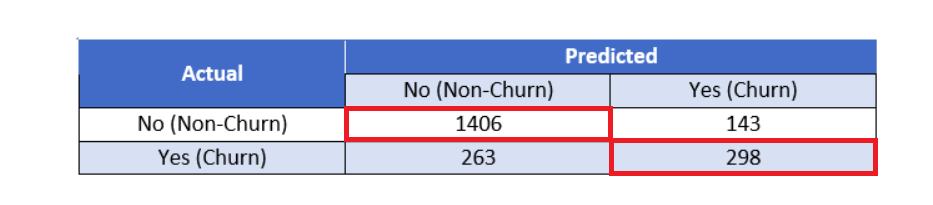
Similarly, the cell at second row, the second column (298) is the number of customers who are actually ‘churn’ and also predicted as ‘churn’.

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 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:



**Correctly Predicted Labels**

Now, accuracy is defined as:

Accuracy=Correctly Predicted Labels/Total Number of Labels

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

Accuracy=1406+298/1406+143+263+298 ~= 80.75

**Calculating Accuracy**

From the confusion matrix you saw in the last question, compute the accuracy of the model.

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | Not Churn | Churn |
| Not Churn | 80 | 30 |
| Churn | 20 | 70 |

**75%**

**Feedback :**

*Correct! The accuracy of a model is given by:*

Accuracy=Correctly precited labelsTotal Number of Labels

*Here, the number of correctly predicted labels are present in the first row, first column and the last row, last column.*

*Hence, you get -*

*Correctly predicted labels = 80 + 70 = 150*

*And the total number of labels is simply the sum of all the numbers present in the confusion matrix. Therefore,*

*Total number of labels = 80 + 30 + 20 + 70 = 200*

*Hence, you get -*

Accuracy=150/200=75%

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
* # 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))

**Confusion Matrix**

Suppose you built a logistic regression model to predict whether a patient has lung cancer or not and you get the following confusion matrix as the output.

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | No | Yes |
| No | 400 | 100 |
| Yes | 50 | 150 |

How many of the patients were wrongly identified as a 'Yes'?

**100**

**Feedback :**

*Look at the table carefully. The value in the first row and the second column will tell you this number. Hence, you get 100 patients which actually didn't have lung cancer but were identified as having lung cancer.*

**Correct**



**Confusion Matrix**

Take a look at the table again.

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | No | Yes |
| No | 400 | 100 |
| Yes | 50 | 150 |

How many of these patients were correctly labelled, i.e. if the patient had lung cancer it was actually predicted as a 'Yes' and if they didn't have lung cancer, it was actually predicted as a 'No'?

550

**Feedback :**

*The sum of values of the numbers in the first row, first column and the last row, last column will give you the answer.*

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | No | Yes |
| No | 400 | 100 |
| Yes | 50 | 150 |

*From the table above, the value in the first row, first column is 400, and the value in the last row, last column is 150. Hence, you get the total correctly predicted labels as 400 + 150 = 550*

**Accuracy Calculation**

From the table you used for the last two questions, what will be the accuracy of the model?

**78.57%**

**Feedback :**

*The accuracy of a model is given by:*

Accuracy=Correctly precited labelsTotal Number of Labels

*The number of correctly predicted labels as you found out from the last question is equal to 550. The total number of labels is (400 + 100 + 50 + 150) = 700. Hence, the accuracy becomes:*

Accuracy=550/700≈78.57%

**Manual Feature Elimination**

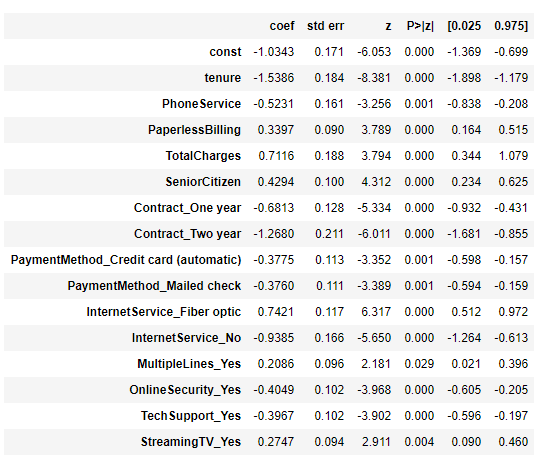
Recall that you had used RFE to select 15 features. But as you saw in the pairwise correlations, there are high values of correlations present between the 15 features, i.e. there is still some multicollinearity among the features. So you definitely need to check the VIFs as well to further eliminate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

VIFi=1/1−Ri2

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

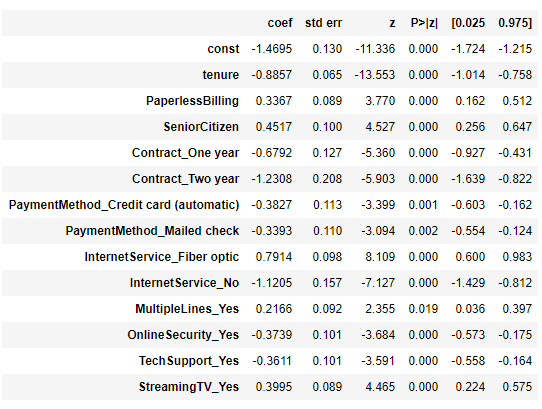
To summarise, you basically performed an iterative manual feature elimination using the VIFs and p-values repeatedly. You also kept on checking the value of accuracy to make sure that dropping a particular feature doesn't affect the accuracy much.

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



**Initial Set of Features (after RFE)**

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



**Final Set of Features after Manual Feature Elimination**

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.

**Multivariate Logistic Regression (Variable Selection)**

Based on the above information, what can you say about the log odds of these two customers?

**PS:***Recall the log odds for univariate logistic regression was given as:*

ln(P/1−P)=β0+β1X

*Hence, for multivariate logistic regression, it would simply become:*

ln(P/1−P)=β0+β1X1+β2X2+β3X3+...+βnXn

**log odds (customer A) > log odds (customer B)**

**Feedback :**

*Recall the log odds are just the linear term present in the logistic regression equation. Hence, here we have 13 variables, so the log odds will be given by:*

ln(P1−P)=β0+β1X1+β2X2+β3X3+...+β13X13

*Now, for the two customers, all beta and all x values are the same, except for*X2*(the variable for paperless billing), which is equal to 1 for customer A and 0 for customer B.*

*Hence, the value will exceed by the coefficient of 'PaperlessBilling' which is 0.3367.*

*Basically, for customer A, this term would be = 0.3367 \* 1*

*And for customer B, this term would be = 0.3367 \* 0*

**Multivariate Logistic Regression (Variable Selection)**

Now, what can you say about the odds of churn for these two customers?

**For customer A, the odds of churning are higher than for customer B**

**Feedback :**

*Recall that in the last question, you were told that log odds for customer A are higher than those for customer B. So, the odds of churning for customer A are also higher than the odds of churning for customer B. This is because, as the number increases, its log increases and vice versa.*

**Multivariate Logistic Regression - Log Odds**

Now, suppose two customers, customer C and customer D, are such that their behaviour is exactly the same, except for the fact that customer C has OnlineSecurity, while customer D does not. What can you say about the odds of churn for these two customers?

Top of Form



**For customer C, the odds of churning are lower than for customer D**

**Feedback :**

*Recall that the log odds for customer C will differ from those for customer D, by a margin of*βOnlineSecurity*. Now since in this case, this coefficient is negative (-0.3739), this means that the log odds of customer C will be 0.3739 less than that of customer D. Since the log odds of customer C are lower, naturally, the actual odds for C would also be lower.*

Bottom of Form

**Log Odds**

Suppose you are working for a media services company like Netflix. They're launching a new show called 'Sacred Games' and you are building a logistic regression model which will predict whether a person will like it or not based on whether consumers have liked/disliked some previous shows. You have the data of five of the previous shows and you're just using the dummy variables for these five shows to build the model. If the variable is 1, it means that the consumer liked the show and if the variable is zero, it means that the consumer didn't like the show. The following table shows the values of the coefficients for these five shows that you got after building the logistic regression model.

|  |  |
| --- | --- |
| **Variable Name** | **Coefficient Value** |
| TrueDetective\_Liked | 0.47 |
| ModernFamily\_Liked | -0.45 |
| Mindhunter\_Liked | 0.39 |
| Friends\_Liked | -0.23 |
| Narcos\_Liked | 0.55 |

 Now, you have the data of three consumers Reetesh, Kshitij, and Shruti for these 5 shows indicating whether or not they liked these shows. This is shown in the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Consumer | TrueDetective\_Liked | ModernFamily\_Liked | Mindhunter\_Liked | Friends\_Liked | Narcos\_Liked |
| Reetesh | 1 | 0 | 0 | 0 | 1 |
| Kshitij | 1 | 1 | 1 | 0 | 1 |
| Shruti | 0 | 1 | 0 | 1 | 1 |

Based on this data, which one of these three consumers is most likely to like to new show 'Sacred Games'?

Top of Form



**Reetesh**

**Feedback :**

*Correct!*

*To find the person who is most likely to like the show, you can use log odds. Recall the log odds is given by:*

ln(P1−P)=β0+β1X1+β2X2+β3X3+...+βnXn

*Here, there are five variables for which the coefficients are given. Hence, the log odds become:*

ln(P1−P)=0.47X1−0.45X2+0.39X3−0.23X4+0.55X5

*As you can see, we have ignored the*β0*since it will be the same for all the three consumers. Now, using the values of the 5 variables given, you get -*

(Log Odds)Reetesh=(0.47×1)−(0.45×0)+(0.39×0)−(0.23×0)+(0.55×1)=1.02

(Log Odds)Kshitij=(0.47×1)−(0.45×1)+(0.39×1)−(0.23×0)+(0.55×1)=0.96

(Log Odds)Shruti=(0.47×0)−(0.45×1)+(0.39×0)−(0.23×1)+(0.55×1)=−0.13

*As you can clearly see, the log odds of Reetesh is the highest, hence, the odds of Reetesh liking the show is the highest and hence, he is most likely to like the new show, Sacred Games.*

Bottom of Form

# **Metrics Beyond Accuracy: Sensitivity & Specificity**

**Check ipython notebook ML-1/LogisticRegression/Telecom\_Churn/LogisticRegression-TelecomChurnCaseStudy.ipynb**

In the previous session, you built a logistic regression model and arrived at the final set of features using RFE (Recursive Feature Elimination) 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.

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:

1. Sensitivity
2. Specificity

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

*Sensitivity = Number of actual Yeses correctly predicted / Total number of actual Yeses*

*i.e.*

*Sensitivity = TP / (TP + FN)*

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 |

* 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.).
* 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).
* Similarly, the third cell contains the actual 'Churns' being predicted as 'Not Churn' which is why we call it **'False Negative'**.
* 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 at in the last row, 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*=692/1287≈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=Number of actual Nos correctly predicted/Total number of actual Nos*

*i.e.*

*Specificity = TN / (TN + FP)*

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:

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

*Specificity=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=3269/3635≈89.931%

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

**Evaluation Metrics**

Among the three metrics that you've learnt about, which one is the highest for the model below?

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | Not Churn | Churn |
| Not Churn | 400 | 100 |
| Churn | 50 | 150 |

Specificity

**Feedback :**

*The formula for the three metrics are given as:*

Accuracy=Correctly Predicted LabelsTotal Number of Labels

Sensitivity=Number of actual Yeses correctly predictedTotal number of actual Yeses=TPTP+FN

Specificity=Number of actual Nos correctly predictedTotal number of actual Nos=TNTN+FP

*Hence, you get:*

Accuracy=400+150/(400+100+50+150)=78.57%

Sensitivity=150/(150+50)=75%

Specificity=400/(400+100)=80%

*As you can clearly see, Specificity (80%) is the highest among the three.*

**Other Metrics**

In the code, you saw Rahim evaluate some other metrics as well. These were:

False Positive Rate=FP/TN+FP

Positive Predictive Value=TP/TP+FP

Negative Predictive Value=TN/TN+FN

* As you can see, the 'False Positive Rate' is basically (1 - Specificity). Check the formula and the values in the code to verify.
* The positive predictive value is**the number of positives correctly predicted by the total number of positives predicted**. This is also known as**'Precision'** which you'll learn more about soon.
* Similarly, the negative predictive value is **the number of negatives correctly predicted by the total number of negatives predicted**. There's no particular term for this as such.

Calculate the given three metrics for the model below and identify which one is the largest among them.

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | Not Churn | Churn |
| Not Churn | 80 | 40 |
| Churn | 30 | 50 |

Negative Predictive Value

**Feedback :**

*Correct! The values that you'll get are:*

False Positive Rate=FP/(TN+FP)=4080+40≈33%

*You could have also used the specificity value you calculated in the last question (~67%) and simply calculated this as 1-Specificity = 1 - 0.67 = 33%*

Positive Predictive Value=TP/(TP+FP)=5050+40=55.55%≈56%

Negative Predictive Value=TN/(TN+FN)=8080+30≈72.72%≈73%

*As you can clearly see, the Negative Predictive Value is the highest of the three.*

**Can someone give examples where we can use Sensitivity and Specificity; and Precision and Recall.**

Taking diabetes diagnosis as an example:

Considering output as either **diabetic**(+ve) or **healthy**(-ve)

* True positive (*TP*): Prediction is +ve and Patient is diabetic,
* True negative (*TN*): Prediction is -ve and Patient is healthy,
* False positive (*FP*): Prediction is +ve and Patient is healthy,
* False negative (*FN*): Prediction is -ve and Patient is diabetic,

**Sensitivity (same as Recall) -** Sensitivity and recall refer how good a test is at detecting the positives. Of all the people who are **diabetic**, how many of those were correctly predicted?

It is given by ***TP/(TP+FN)***

**Specificity** – how good a test is at avoiding false alarms

Of all the people who are **healthy**, how many of those were correctly predicted?

It refers to ***TN/(TN+FP)***

**Precision** – how many of the positively classified were relevant

How many of those who we labeled as diabetic actually diabetic?

**It refers to *TP/(TP+FP)***

**Example 1:** For instance, an AI program for recognizing boys in photographs identifies 8 boys in a picture containing 12 boys and few girls. Of the 8 identified as boys, 5 actually are boys (true positives), while the rest are girls (false positives). The program's **Precision** is 5/8 while its **Recall**is 5/12. So, in this case, **Precision** is "how useful the results are", and **Recall** is "how complete the results are".

**Example 2:**  Suppose in airport security, since testing of passengers is for potential threats to safety, scanners may be set to trigger alarms on low-risk items like belt buckles and keys (low S**pecificity**) in order to increase the probability of identifying dangerous objects and minimize the risk of missing objects that do pose a threat (high S**ensitivity**).

Sensitivity and specificity are mostly screening tests used in Clinical practice to assess the likelihood that a person has a certain medical condition. to evaluate results which measure the presence of disease ( yes or no).

Precision and recall helps you in choosing or classification tasks. Real time examples are : fraud detections, marketing websites to capture emotions of likes/dislikes. Ex: Identifying a passenger as terrorist or innocent.

# **ROC (Receiver Operating Characteristic) 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.

**Excel: ROC+Curve+-+Excel+Demo**

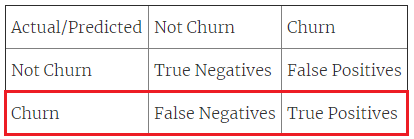
So you saw that the predicted labels depend entirely on the cutoff or the threshold that you have chosen. For low values of threshold, you'd have a higher number of customers predicted as a 1 (Churn). This is because if the threshold is low, it basically means that everything above that threshold would 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).

**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)=True Positives/Total Number of Actual Positives***

Now, recall the labels in the confusion matrix,



**Confusion Matrix Highlighting the Total Number of Actual 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)=True Positives/(True Positives+False Negatives)=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.

**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)=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)=False PositivesTrue Negatives+False Positives=FP/(TN+FP)**

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

***Specificity****=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.

**True Positive Rate**

You have the following table showcasing the actual 'Churn' labels and the predicted probabilities for 5 customers.

|  |  |  |
| --- | --- | --- |
| Customer | Churn | Predicted Churn Probability |
| Thulasi | 1 | 0.52 |
| Aditi | 0 | 0.56 |
| Jaideep | 1 | 0.78 |
| Ashok | 0 | 0.45 |
| Amulya | 0 | 0.22 |

 Calculate the True Positive Rate and False Positive rate for the cutoffs of 0.4 and 0.5. Which of these cutoffs, will give you a better model?

Note: The good model is the one in which TPR is high and FPR is low.

**Cutoff of 0.5**

**Feedback :**

*Now, at the cutoff of 0.4, you get the following values of predicted probabilities:*

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | Churn | Predicted Churn Probability | Predicted Churn Label |
| Thulasi | 1 | 0.52 | 1 |
| Aditi | 0 | 0.56 | 1 |
| Jaideep | 1 | 0.78 | 1 |
| Ashok | 0 | 0.45 | 1 |
| Amulya | 0 | 0.22 | 0 |

*From the above table, you can easily calculate:*

*True Positives = 2, False Positives = 2*

*Also, from the original table, you have:*

*Actual Positives = 2, Actual Negatives = 3*

*Hence, you get:*

TPR=True Positives/Total Actual Positives=2/2=100%

FPR=False Positives/Total Actual Negatives=2/3≈67%

*Performing similar steps for a cutoff of 0.5 will give you -*

TPR=100%

FPR≈33%

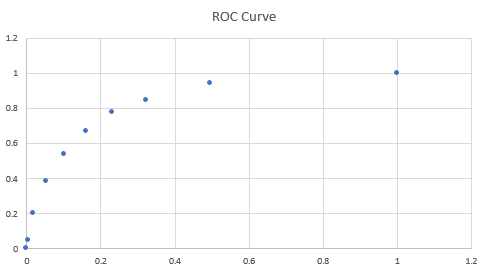
*(Do calculate it yourself to verify)*

*As you can see, with both the cutoffs, the TPR is 100% but for the cutoff 0f 0.5 you have a lower value of FPR. So clearly, a cutoff of 0.5 gives you a better model.*

*Please note that 0.5 just gives the better model among 0.4 and 0.5. It might be possible that there is a cutoff point which gives an even better model.*

So, now that you have understood what these terms are, you'll now learn about **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.

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.

**Changing the Threshold**

You initially chose a threshold of 0.5 wherein a churn probability of greater than 0.5 would result in the customer being identified as 'Churn' and a churn probability of lesser than 0.5 would result in the customer being identified as 'Not Churn'.

Now, suppose you decreased the threshold to a value of 0.3. What will be its effect on the classification?

Top of Form



**More customers would now be classified as 'Churn'.**

**Feedback :**

*Correct! Now since you have decreased the cutoff to 0.3, it would mean that:*

*Customers with churn probability > 0.3 will be identified as 'Churn'.  
Customers with churn probability < 0.3 will be identified as 'Not Churn'.  
Initially, the threshold was 0.5. Look at the customers in the 0.3-0.5 probability range. They were being identified as 'Not Churn' before, but now, are being identified as 'Churn'. Hence, naturally, the number of people being identified as 'Churn' will increase.*

Bottom of Form

**TPR and FPR**

Fill in the blanks:

When the value of TPR increases, the value of FPR \_\_\_\_\_\_.

Top of Form



increases

**Feedback :**

*Correct! This can be clearly seen from the ROC curve as well. When the value of TPR (on the Y-axis) is increasing, the value of FPR (on the X-axis) also increases.*

Bottom of Form

**Area Under the Curve**

You have the following five AUCs (Area under the curve) for ROCs plotted for five different models. Which of these models is the best?

|  |  |
| --- | --- |
| Model | AUC |
| A | 0.54 |
| B | 0.82 |
| C | 0.79 |
| D | 0.66 |
| E | 0.56 |

**B**

**Feedback :**

*Correct!*

*Recall that when the ROC curve is more towards the top left corner of the graph, the model is deemed to be more accurate. Hence, a greater area under the curve would mean the model is more accurate. Among the five models given, B has the highest AUC and hence is the most accurate model. Also, note that the highest value of AUC can be 1.*

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**

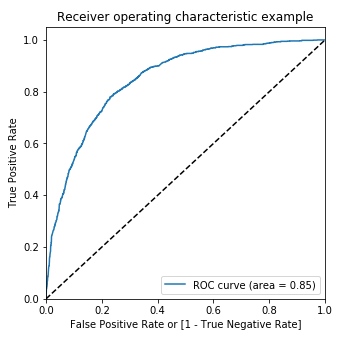
# Calling the function

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.



**ROC Curve**

**The**450**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**

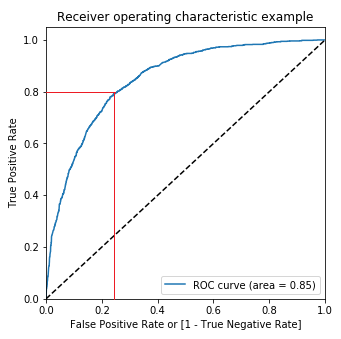
As you saw in the last segment as well, the ROC curve shows thetrade-off between True Positive Rate and False Positive Rate which essentially can also be viewed as a tradeoff between Sensitivity and Specificity. As you can see, on the Y-axis, you have the values of Sensitivity and on the X-axis, you 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 a 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.

**ROC Curve**

Following is the ROC curve that you got.



As you can see, when the 'True Positive Rate' is 0.8, the 'False Positive Rate' is about 0.24. What will be the value of specificity, then?

Top of Form



0.8



0.2



**0.76**

**Feedback :**

*Correct!*

*Recall that the False Postive Rate is nothing but (1 - True Negative Rate) and the True Negative Rate is simply the specificity. Hence,*

*False Positive Rate = 1 - Specificity*

*or, Specificity = 1 - False Postive Rate*

*Here, the False Positive Rate is 0.24. Therefore, Specificity = (1 - 0.24) = 0.76.*

**Correct**

Bottom of Form

 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

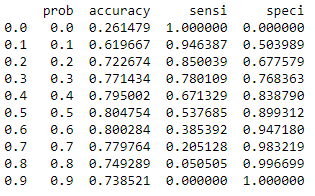
speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])

sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])

cutoff\_df.loc[i] =[ i ,accuracy,sensi,speci]

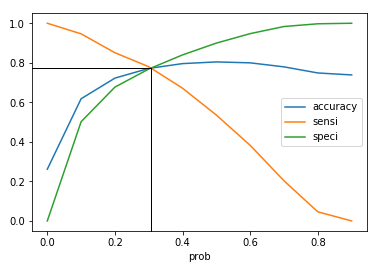
print(cutoff\_df)

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:



**Sensitivity-Specificity Trade-off**

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.



**Accuracy, Sensitivity, and Specificity tradeoff**

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.

**Choosing the Optimal Cut-off**

Suppose you created a dataframe to find out the optimal cut-off point for a model you built. The dataframe looks like the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threshold | Probability | Accuracy | Sensitivity | Specificity |
| 0.0 | 0.0 | 0.21 | 1.00 | 0.00 |
| 0.1 | 0.1 | 0.39 | 0.96 | 0.22 |
| 0.2 | 0.2 | 0.56 | 0.88 | 0.49 |
| 0.3 | 0.3 | 0.59 | 0.81 | 0.53 |
| 0.4 | 0.4 | 0.62 | 0.78 | 0.63 |
| 0.5 | 0.5 | 0.74 | 0.73 | 0.74 |
| 0.6 | 0.6 | 0.81 | 0.64 | 0.79 |
| 0.7 | 0.7 | 0.78 | 0.42 | 0.83 |
| 0.8 | 0.8 | 0.63 | 0.21 | 0.92 |
| 0.9 | 0.9 | 0.56 | 0.03 | 0.98 |

Based on the table above, what will the approximate value of the optimal cut-off be?

0.5

**Feedback :**

*Correct! The optimal cut-off point exists where the values of accuracy, sensitivity, and specificity are fairly decent and almost equal. At the cut-off of 0.5, the metric values are 0.74, 0.73, and 0.74 respectively. This is the optimal value of threshold that you can have.*

**Choosing a model evaluation metric**

As you learnt, there is usually a trade-off between various model evaluation metrics, and you cannot maximise all of them simultaneously. For e.g., if you increase sensitivity (% of correctly predicted churns), the specificity (% of correctly predicted non-churns) will reduce.

Let's say that you are building a telecom churn prediction model with the business objective that your company wants to implement an aggressive customer retention campaign to retain the 'high churn-risk' customers. This is because a competitor has launched extremely low-cost mobile plans, and you want to avoid churn as much as possible by incentivising the customers. Assume that budget is not a constraint.

Which of the following metrics should you choose the maximise?

Top of Form



Accuracy



**Sensitivity**

**Feedback :**

*Yes, high sensitivity implies that your model will correctly identify almost all customers who are likely to churn. It will do that by over-estimating the churn likelihood, i.e. it will misclassify some non-churns as churns, but that is the trade-off you need to choose rather than the opposite case (in which case you may lose some low churn risk customers to the competition).*

**Accuracy of the Model**

Using the threshold of 0.3, what is the approximate accuracy of the model now?

**77%**

**Feedback :**

*Correct!*

*Use the following code to calculate the accuracy:*

metrics.accuracy\_score(y\_train\_pred\_final.Churn, y\_train\_pred\_final.final\_predicted)

*You'll see that you get an accuracy of about 77.14%.*

**Confusion Matrix**

Get the confusion matrix after using the cut-off 0.3. What is the number of 'False Negatives' now?

Top of Form



2793



842



**283**

**Feedback :**

*Correct! When you run the following code to get the confusion matrix,*

confusion2 = metrics.confusion\_matrix(y\_train\_pred\_final.Churn, y\_train\_pred\_final.final\_predicted )

*you'll get the following confusion matrix:*

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | Not Churn | Churn |
| Not Churn | 2793 | 842 |
| Churn | 283 | 1004 |

*Also, recall that the labels in the confusion matrix are:*

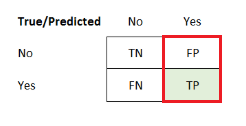
|  |  |  |
| --- | --- | --- |
| Actual/Predicted | Not Churn | Churn |
| Not Churn | True Negatives | False Positives |
| Churn | False Negatives | True Positives |

*You can clearly see that the number of 'False Negatives' is now 283. Also, note that the number of 'False Negatives' has now dropped significantly and the number of 'True Positives' has increased. Thus, choosing a lower cut-off has definitely helped in capturing the 'Churns' better.*

Bottom of Form

# **Precision and Recall**

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



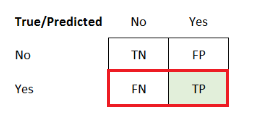
**Precision Elements in a Confusion Matrix**

The formula for precision can be given as:

  Precision=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.



**Recall Elements in a Confusion Matrix**

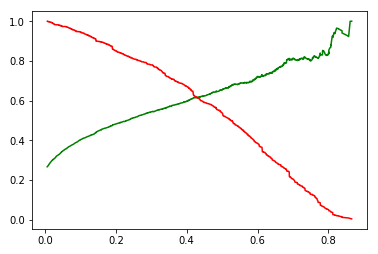
The formula for recall can be given as:

Recall=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.

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:



**Precision-Recall Tradeoff**

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 which changes with cut-offs. And because the predicted values can swing wildly, you get a very jumpy curve.

**Calculating Precision**

Calculate the precision value for the following model.

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | Not Churn | Churn |
| Not Churn | 400 | 100 |
| Churn | 50 | 150 |

Top of Form



**60%**

**Feedback :**

*Correct! The formula for precision is given by:*

Precision=TPTP+FP

*From the matrix given,*

*TP = 150*

*FP = 100*

*Hence, you get,*

Precision=150150+100=60%

Bottom of Form

**F1-score**

There is a measure known as **F1-score**which essentially combines both precision and recall. It is the basically the [harmonic mean](https://en.wikipedia.org/wiki/Harmonic_mean) of precision and recall and its formula is given by:

F=2×precision×recall/precision+recall

The F1-score is useful when you want to look at the performance of precision and recall together.

Calculate the F1-score for the model below:

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | Not Churn | Churn |
| Not Churn | 400 | 100 |
| Churn | 50 | 150 |

**67%**

**Feedback :**

*Correct!*

*From the confusion matrix given,*

*TP = 150*

*FP = 100*

*FN = 50*

*Hence, you get -*

Precision=150/100+150=0.6

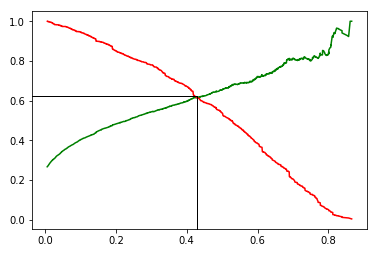
Recall=150/150+50=0.75

*So, the F1-score becomes -*

F=2×0.6×0.75/0.6+0.75≈66.67%≈67%

**Optimal Cut-off**

When using the sensitivity-specificity tradeoff, you found out that the optimal cutoff point was 0.3. Now, when you plotted the precision-recall tradeoff, you got the following curve:



What is the optimal cutoff point according to the curve given above?

 Top of Form



0.24



**0.42**

**Feedback :**

*Yes! The optimal cutoff point is where the values of precision and recall will be equal. This is similar to what you saw in the sensitivity-specificity tradeoff curve as well. So, when precision and recall are both around 0.62, the two curves are intersecting. And at this place, if you extend the line to the X-axis as given, you can see that the threshold value is 0.42.*

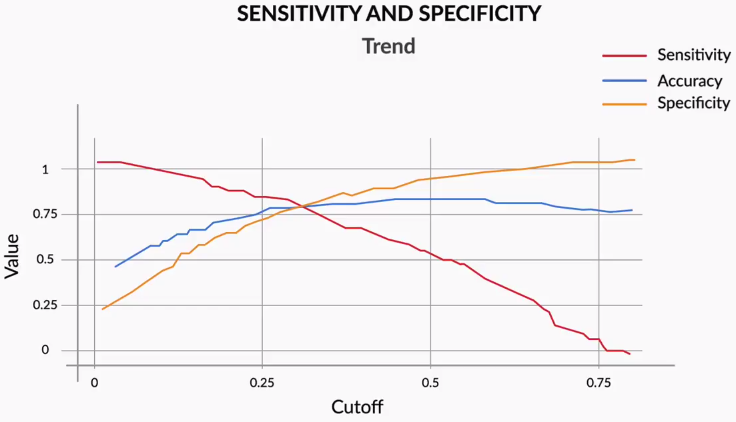
# **Summary**

So to summarise, the steps that you performed throughout 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.



**Sensitivity and Specificity**

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.

**Evaluation Metrics**

Consider the same model given in the last question.

|  |  |  |  |
| --- | --- | --- | --- |
| Patient ID | Heart Disease | Predicted Probability for Heart Disease | Predicted Label |
| 1001 | 0 | 0.34 | 0 |
| 1002 | 1 | 0.58 | 1 |
| 1003 | 1 | 0.79 | 1 |
| 1004 | 0 | 0.68 | 1 |
| 1005 | 0 | 0.21 | 0 |
| 1006 | 0 | 0.04 | 0 |
| 1007 | 1 | 0.48 | 0 |
| 1008 | 1 | 0.64 | 1 |
| 1009 | 0 | 0.61 | 1 |
| 1010 | 1 | 0.86 | 1 |

 Calculate the values of Accuracy, Sensitivity, Specificity, and Precision. Which of these four metrics is the highest for the model?

Top of Form

**Sensitivity**

**Feedback :**

*From the table given above, you can easily find out that -*

*TN = 3*

*FP = 2*

*FN = 1*

*TP = 4*

*Hence, your confusion matrix will look like:*

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | No Heart Disease | Heart Disease |
| No Heart Disease | 3 | 2 |
| Heart Disease | 1 | 4 |

*Hence, you get -*

Accuracy=3+43+2+1+4=70%

Sensitivity=44+1=80%

Specificity=33+2=60%

Precision=44+2≈67%

*As you can clearly see, sensitivity has the highest value.*Bottom of Form

Logistic Regression - Industry Applications - Part I

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.

**Main differences between Logistic Regression v/s Linear Regression**

1. *Dependent/response variable in linear regression is continuous whereas, in logistic regression, it is the discrete type.*
2. *Cost function in linear regression minimise the error term Sum(Actual(Y)-Predicted(Y))^2 but logistic regression uses maximum likelihood method for maximising probabilities.*

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. As the score increases the log odds linearly increases.

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∗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.

**Selecting the right sample** is essential for solving any business problem. There are major errors you should be on the lookout for while selecting the samples, these are for example:

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.

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.

**Nuances of Logistic Regression - Variable Transformation-I**

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**.