1. What is a classification problem?

Classification can be performed on structured or unstructured data.

Classification is a technique where we categorize data into a given number of classes.

The main goal of a classification problem is to identify the category/class to which a new data will fall under.

Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

The output variables are often called labels or categories. The mapping function predicts the class or category for a given observation.

A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

A classification model attempts to draw some conclusion from observed values.

For example, an email of text can be classified as belonging to one of two classes: “spam“ and “not spam“. For example, when filtering emails “spam” or “not spam”, when looking at transaction data, “fraudulent”, or “authorized”. In short Classification either predicts categorical class labels or classifies data (construct a model) based on the training set and the values (class labels) in classifying attributes and uses it in classifying new data. There are a number of classification models. Classification models include logistic regression, decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs-rest, and Naive Bayes.

A classification problem requires that examples be classified into one of two or more classes.

A classification can have real-valued or discrete input variables.

Binary Classification: Classification task with two possible outcomes. Eg: Gender classification (Male / Female)

A problem with two classes is often called a two-class or binary classification problem.

Multi class classification: Classification with more than two classes.

In multi class classification each sample is assigned to one and only one target label. Eg: An animal can be cat or dog but not both at the same time.

A problem with more than two classes is often called a multi-class classification problem.

Multi label classification: Classification task where each sample is mapped to a set of target labels (more than one class). Eg: A news article can be about sports, a person, and location at the same time.

A problem where an example is assigned multiple classes is called a multi-label classification problem.

It is common for classification models to predict a continuous value as the probability of a given example belonging to each output class. The probabilities can be interpreted as the likelihood or confidence of a given example belonging to each class. A predicted probability can be converted into a class value by selecting the class label that has the highest probability.

Classification vs Regression

Classification :Dependent variable Categirical and Independent varaiable Categorical/Contineous.

Classification is the task of predicting a discrete class label.

A classification algorithm may predict a continuous value, but the continuous value is in the form of a probability for a class label.

Classification predictions can be evaluated using accuracy, whereas regression predictions cannot.

Regression :Dependent variable Continuous and Independent varaiable Categorrical/Contineous.

Regression is the task of predicting a continuous quantity.

A regression algorithm may predict a discrete value, but the discrete value in the form of an integer quantity.

Regression predictions can be evaluated using root mean squared error, whereas classification predictions cannot.

Some algorithms can be used for both classification and regression with small modifications, such as decision trees and artificial neural networks. Some algorithms cannot, or cannot easily be used for both problem types, such as linear regression for regression predictive modeling and logistic regression for classification predictive modeling.

Fundamentally, classification is about predicting a label and regression is about predicting a quantity.

Convert Between Classification and Regression Problems

In some cases, it is possible to convert a regression problem to a classification problem. For example, the quantity to be predicted could be converted into discrete buckets.

For example, amounts in a continuous range between $0 and 100$ could be converted into 2 buckets:

Class 0: $0 to 49$ Class 1: $50 to 100$ This is often called discretization and the resulting output variable is a classification where the labels have an ordered relationship (called ordinal).

In some cases, a classification problem can be converted to a regression problem. For example, a label can be converted into a continuous range.

Some algorithms do this already by predicting a probability for each class that in turn could be scaled to a specific range:

quantity = min + probability * range 1 quantity = min + probability * range Alternately, class values can be ordered and mapped to a continuous range:

$0 to 49$ for Class 1 $50 to 100$ for Class 2 If the class labels in the classification problem do not have a natural ordinal relationship, the conversion from classification to regression may result in surprising or poor performance as the model may learn a false or non-existent mapping from inputs to the continuous output range.

2. What is Logistic Regression?

Logistic Regression

Logistic Regression is a classification algorithm.

It is used to predict a binary outcome (1 / 0, Yes / No, True / False)

It predicts the probability of occurrence of a binary outcome using a logit function.

we are using log of odds as dependent variable.

In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

logistic regression it will generate the probabilites and these probabilities will be the probability of event. Event means defaulter and non-defaulter when it will happen.

sigmoid function is also called as logit function what it does it will tries to convert all probablities values between 0 and 1.

Linear regression we have error as residual = actual -predicted

Logistic regrssion we dont have errorr here we will have mis classsification.i.e 1 predict as 0 and 0 predict as 1.

logistic regression uses a maximum likelihood estimation procedure rather than the least squares estimation procedure that is used in linear regression.

The logit distribution contains the estimated probabilities to lie between 0 and 1.

Logistic Regression is used when the dependent variable(target) is categorical.

We use the activation function (sigmoid) to convert the outcome into categorical value. There are many examples where we can use logistic regression for example, it can be used for fraud detection, spam detection, cancer detection, etc. Despite its name, it’s not a regression problem algorithm where you want to predict a continuous outcome. Instead, logistic regression is the go-to technique of binary classification.

Difference between Linear Regression vs Logistic Regression

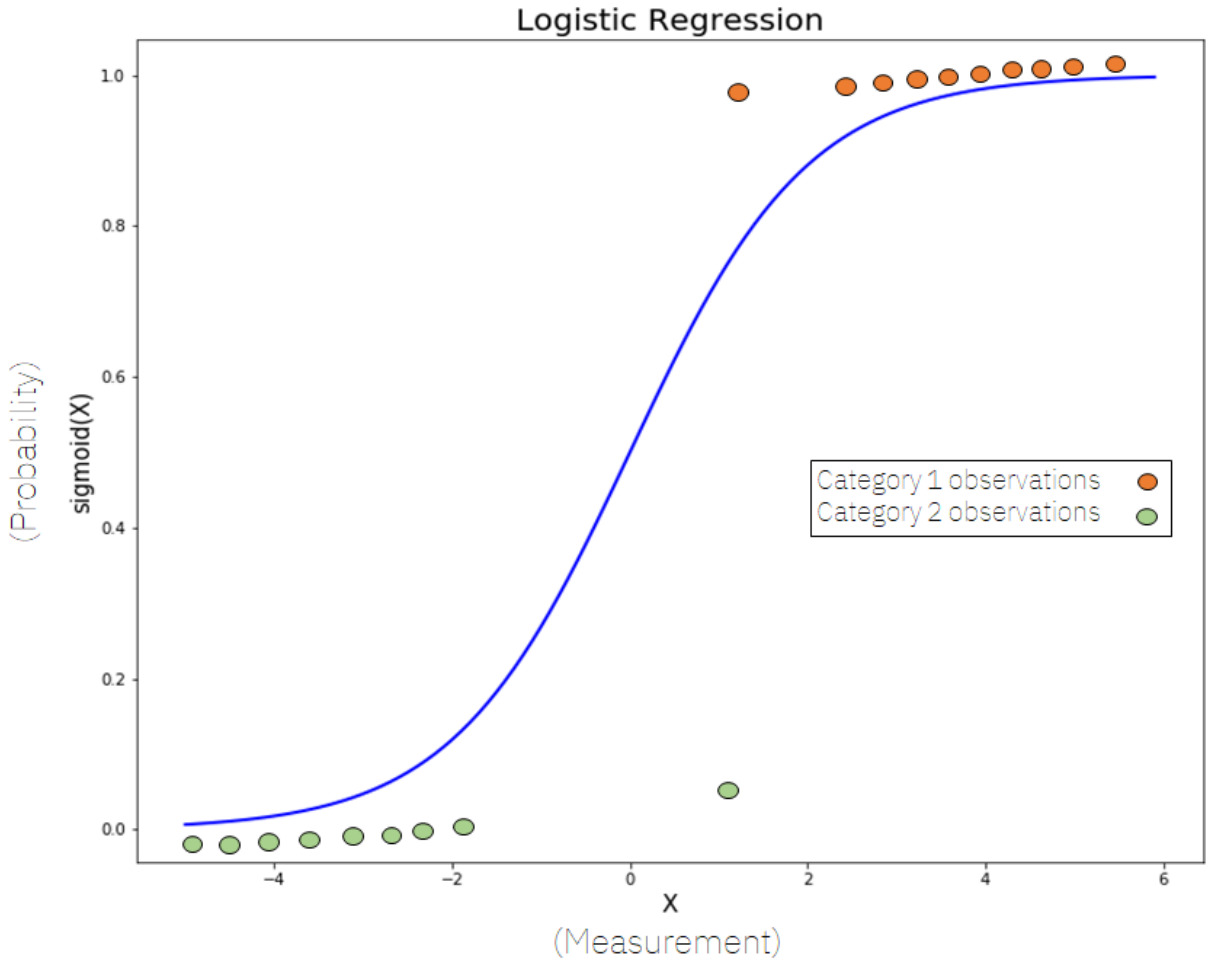
Linear Regression is used when our dependent variable is continuous in nature for example weight, height, numbers, etc.

Logistic Regression is used when the dependent variable is binary or limited for example: yes and no, true and false, 1 or 2 or 3 or 4, etc.

Linear regression uses the ordinary least square method to minimize the error and arrives at the best possible solution

Logistic regression achieves the best outcomes by using the maximum likelihood method.

Logistic Regression models are classification models; specifically binary classification models (they can only be used to distinguish between 2 different categories — like if a person is obese or not given its weight, or if a house is big or small given its size). This means that our data has two kinds of observations (Category 1 and Category 2 observations) like we can observe in the figure.



as we can see, the Y-axis goes from 0 to 1. This is because the sigmoid function always takes ,as maximum and minimum these two values, and this fits very well our goal of classifying samples in two different categories. By computing the sigmoid function of X (that is a weighted sum of the input features, just like in Linear Regression), we get a probability (between 0 and 1 obviously) of an observation belonging to one of the two categories. The formula for the sigmoid function is the following:

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

Types of Logistic Regression

Types of Logistic Regression

1. Binary Logistic Regression The categorical response has only two 2 possible outcomes. Example: Spam or Not
2. Multinomial Logistic Regression Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)
3. Ordinal Logistic Regression Three or more categories with ordering. Example: Movie rating from 1 to 5

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function
It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1,

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

Learning the Logistic Regression Model

The coefficients (Beta values b) of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation.

Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data (more on this when we talk about preparing your data).

The best coefficients would result in a model that would predict a value very close to 1 (e.g. male) for the default class and a value very close to 0 (e.g. female) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that minimize the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class).

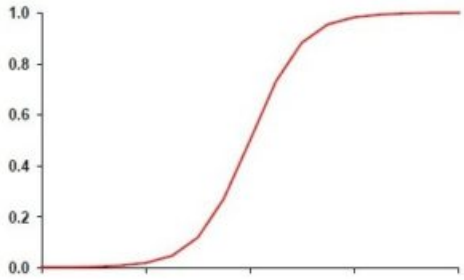
We are not going to go into the math of maximum likelihood. It is enough to say that a minimization algorithm is used to optimize the best values for the coefficients for your training data. This is often implemented in practice using efficient numerical optimization algorithm (like the Quasi-newton method).

When you are learning logistic, you can implement it yourself from scratch using the much simpler gradient descent algorithm.

What is Logistic Regression?

Logistic Regression is used to solve the classification problems, so it's called as Classification Algorithm that models the probability of output class.

- It is a classification problem where your target element is categorical
- Unlike in Linear Regression, in Logistic regression the output required is represented in discrete values like binary 0 and
- It estimates relationship between a dependent variable (target) and one or more independent variable (predictors) where dependent variable is categorical/nominal.



Sigmoid Function:

- It is the logistic expression especially used in Logistic Regression.
- The sigmoid function converts any line into a curve which has discrete values like binary 0 and.
- In this session let's see how a continuous linear regression can be manipulated and converted into Classifies Logistic.

$$p = \frac{1}{1 + e^{-\hat{y}}}$$

Where,

P represents Probability of Output class Y represents predicted output

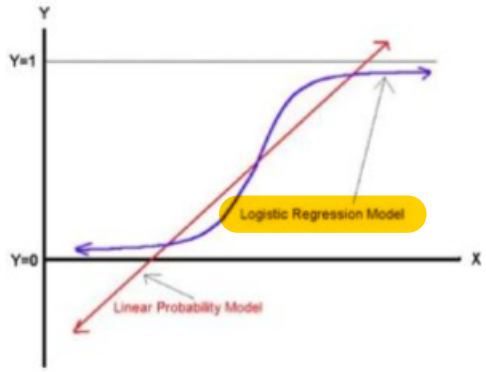
Learning Logistic Regression Model:

Consider a scenario where we need to classify whether a patient has diabetes or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.3 and the threshold value is 0.6, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Comparing Linear Probability Model and Logistic Regression Model:

As Linear Regression is unbounded, it's not useful to solve classification problems. So this is where Logistic Regression comes into picture.



We know that the below expression is the Linear equation used in Linear Regression.

$$\hat{y} = mx + c$$

x is the independent variable

Now let's see when what happens when we related both the algorithm equations.

$$\hat{y} = mx + c$$

$$\downarrow$$

$$p = \frac{1}{1 + e^{-\hat{y}}}$$

When we substitute the linear equation ?□ in the probability equation we get the below result.

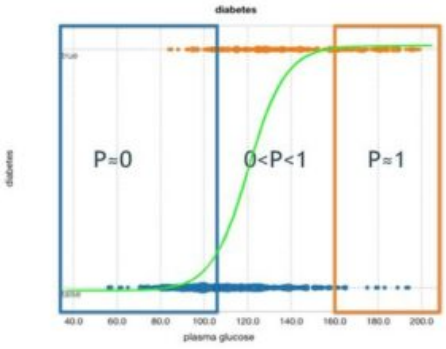
$$p = \frac{1}{1 + e^{-(mx+c)}}$$

$$\downarrow$$

$$ln\left(\frac{p}{1-p}\right) = mx + c$$

From the above equation we can see that the value of P lies in between 0 and 1. So the graphical representation

From the above equation we can see that the value of P lies in between 0 and 1. So the graphical representation of the same will be as below.



Types of Logistic Regression:

- Binary Logistic Regression
- Multinomial Logistic Regression
- Ordinal Logistic Regression

For the model to be a cent percent accurate one, we need to calculate and find out few parameters of the algorithm in order to check how accurate our Binary Logistic Regression model is.

The key parameters we calculate and check are dependent of the topic called CONFUSION MATRIX.

3.Why is the Sigmoid function used for Logistic Regression?

In llnear regression we will have output as contineous values.

if we used same Linear regression equation y =mx+c to find out classification problem.

Logistic regression we will have output as 1 and 0 means we are prediction class in logistic regression

but if you Linear regression eqn to solved classification problem it will not going to work out.

as we need to find out class 1 and 0 so we used here probability .

as we know probabilty have range betn o to 1 so that we can compress the linear regrssion line

so we will use logit or odd or sigmoid function.

sigmoid function is also called as logit function what it does it will tries to convert all probablities values between 0 and 1.

The probability that an event will occur divide by total event.Probabilities always range between 0 and 1.

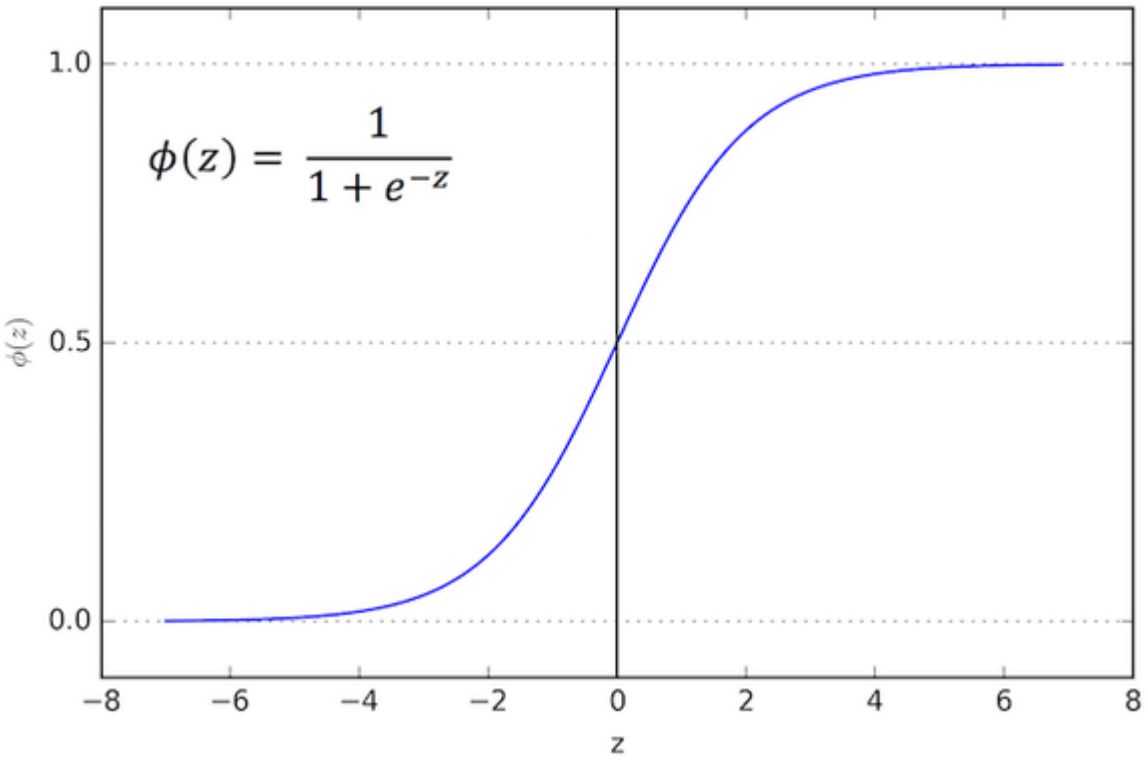
The odds are defined as the probability that the event will occur divided by the probability that the event will not occur.

Odds are defined as the ration of the probabilty of success divide by probabilty of the failure odds ration hat it does whatever data you have it will try to convert into probabilities of two types.tries to convert into 0 nand 1 once we got 0 and 1 then we use logit function on that we get prediction.

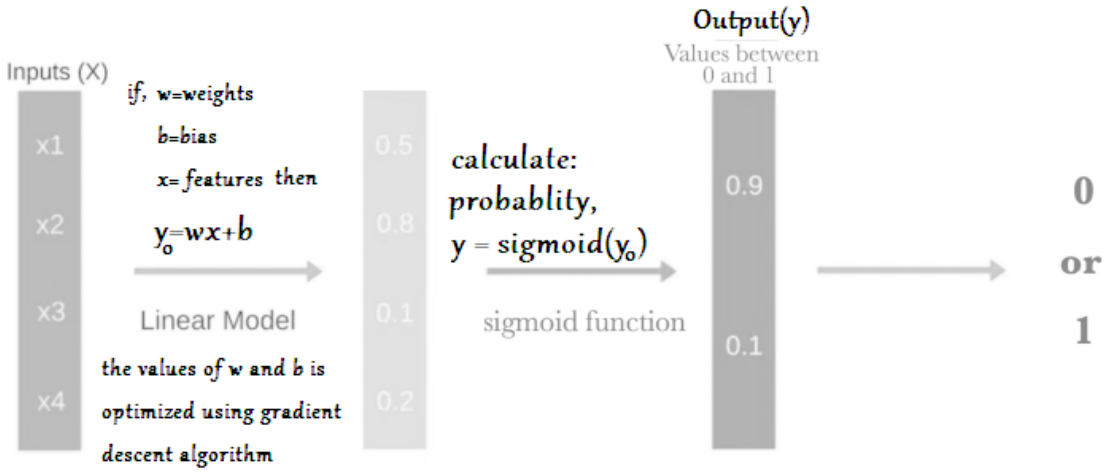
Odd : probabilty of occurs is divide by probability of non occurance. range between zero to infinity.

$$odds = \frac{P}{1-P}$$

Logistic function (sigmoid function) and its graph:



Steps involved in logistic regression can be understood from the image below:



Linear regression is suitable for predicting output that is continuous value, such as predicting the price of a property. Its prediction output can be any real number, range from negative infinity to infinity. The regression line is a straight line.

Whereas logistic regression is for classification problems, which predicts a probability range between 0 to 1.

For example, predict whether a customer will make a purchase or not. The regression line is a sigmoid curve.

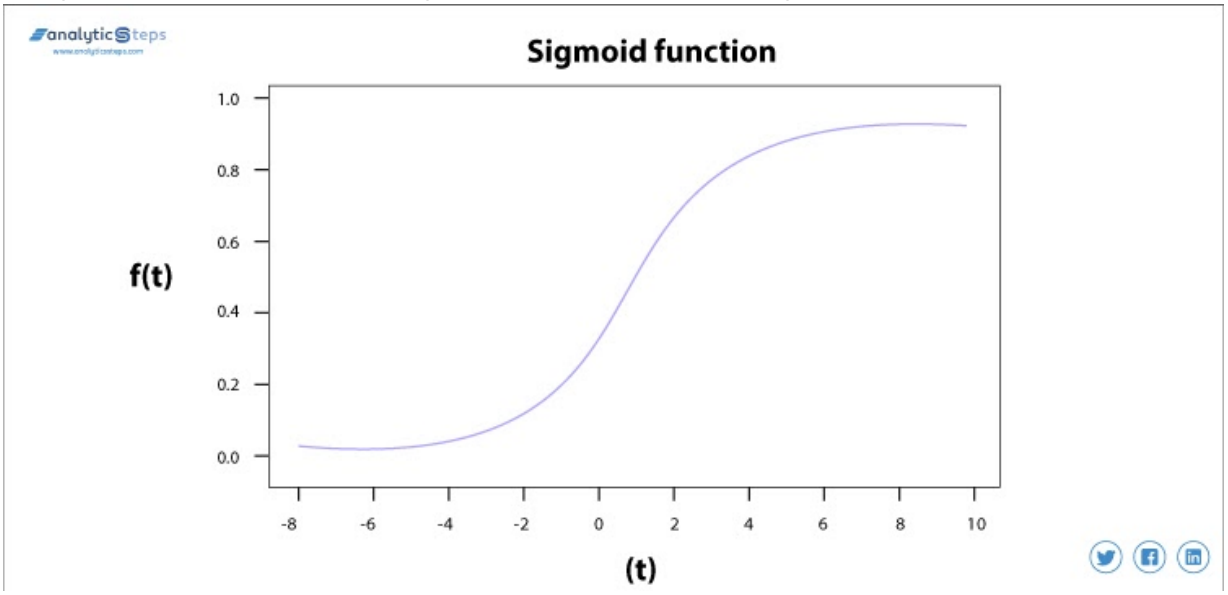
What is the Sigmoid Function?

It is a mathematical function having a characteristic that can take any real value and map it to between 0 to 1 shaped like the letter “S”. The sigmoid function also called a logistic function.

Y = 1 / 1+e -z

Outlining Sigmoid Function curve mapping between the values 0 to 1.

This is because the sigmoid function always takes ,as maximum and minimum these two values, and this fits very well our goal of classifying samples in two different categories. By computing the sigmoid function of X (that is a weighted sum of the input features, just like in Linear Regression), we get a probability (between 0 and 1 obviously) of an observation belonging to one of the two categories. The formula for the sigmoid function is the following:



So, if the value of z goes to positive infinity then the predicted value of y will become 1 and if it goes to negative infinity then the predicted value of y will become 0. And if the outcome of the sigmoid function is more than 0.5 then we classify that label as class 1 or positive class and if it is less than 0.5 than we can classify it to negative class or label as class 0.

logistic regression is that the sigmoid function outputs the conditional probabilities of the prediction, the class probabilities. How does it work? Let’s start with the so-called “odds ratio” p / (1 - p), which describes the ratio between the probability that a certain, positive, event occurs and the probability that it doesn’t occur – where positive refers to the “event that we want to predict”, i.e., p(y=1 | x).

WHY LINEAR REGRESSION IS NOT SUITABLE FOR CLASSIFICATION

Problem #1: Predicted value is continuous, not probabilistic In a binary classification problem, what we are interested in is the probability of an outcome occurring.

Probability is ranged between 0 and 1, where the probability of something certain to happen is 1, and 0 is something unlikely to happen. But in linear regression, we are predicting an absolute number, which can range outside 0 and 1.

2>sensitive to imbalance data when using linear regression for classification

===

Now that we know what the logistic function is, let's see how it is used in logistic regression.

Logistic Regression Predicts Probabilities (Technical Interlude) Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modeling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

P(sex=male|height)

Written another way, we are modeling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

P(X) = P(Y=1|X)

We're predicting probabilities? I thought logistic regression was a classification algorithm?

Note that the probability prediction must be transformed into a binary values (0 or 1) in order to actually make a probability prediction. More on this later when we talk about making predictions.

Logistic regression is a linear method, but the predictions are transformed using the logistic function. The impact of this is that we can no longer understand the predictions as a linear combination of the inputs as we can with linear regression, for example, continuing on from above, the model can be stated as:

p(X) = e^(b0 + b1*X) / (1 + e^(b0 + b1*X))

I don't want to dive into the math too much, but we can turn around the above equation as follows (remember we can remove the e from one side by adding a natural logarithm (ln) to the other):

ln(p(X) / 1 – p(X)) = b0 + b1 * X

This is useful because we can see that the calculation of the output on the right is linear again (just like linear regression), and the input on the left is a log of the probability of the default class.

This ratio on the left is called the odds of the default class (it's historical that we use odds, for example, odds are used in horse racing rather than probabilities). Odds are calculated as a ratio of the probability of the event divided by the probability of not the event, e.g. 0.8/(1-0.8) which has the odds of 4. So we could instead write:

ln(odds) = b0 + b1 * X

Because the odds are log transformed, we call this left hand side the log-odds or the probit. It is possible to use other types of functions for the transform (which is out of scope_, but as such it is common to refer to the transform that relates the linear regression equation to the probabilities as the link function, e.g. the probit link function.

We can move the exponent back to the right and write it as:

odds = e^(b0 + b1 * X)

All of this helps us understand that indeed the model is still a linear combination of the inputs, but that this linear combination relates to the log-odds of the default class.

At last, here are some points about Logistic regression to ponder upon:

Does NOT assume a linear relationship between the dependent variable and the independent variables, but it does assume linear relationship between the logit of the explanatory variables and the response. Independent variables can be even the power terms or some other nonlinear transformations of the original independent variables. The dependent variable does NOT need to be normally distributed, but it typically assumes a distribution from an exponential family (e.g. binomial, Poisson, multinomial, normal,…); binary logistic regression assume binomial distribution of the response. The homogeneity of variance does NOT need to be satisfied. Errors need to be independent but NOT normally distributed. It uses maximum likelihood estimation (MLE) rather than ordinary least squares (OLS) to estimate the parameters, and thus relies on large-sample approximations.

4.What is the cost function?

Logistic Regression is used when the variable you want to predict can only take discrete values i.e. it is used to solve classification problem.

Sigmoid activation In order to map predicted values to probabilities, we use the sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

Math

S(z)=1/1+e^−z

s(z) = output between 0 and 1 (probability estimate) z = input to the function (your algorithm's prediction e.g. mx + b) e = base of natural log

Decision boundary Our current prediction function returns a probability score between 0 and 1. In order to map this to a discrete class (true/false, cat/dog), we select a threshold value or tipping point above which we will classify values into class 1 and below which we classify values into class 2.

p≥0.5,class=1p<0.5,class=0 For example, if our threshold was .5 and our prediction function returned .7, we would classify this observation as positive. If our prediction was .2 we would classify the observation as negative. For logistic regression with multiple classes we could select the class with the highest predicted probability.

The cost function return value that representing how well your model perform. It's like a function that gives you the amount of error rate.

To find the optimal model that has minimum error rate (cost function) we use gradient descent.

Cost function is used to calculate the error value in the model. The error is basically the difference between predicted output and the actual output. In Logistic Regression, the Cost function is represented is defined as:

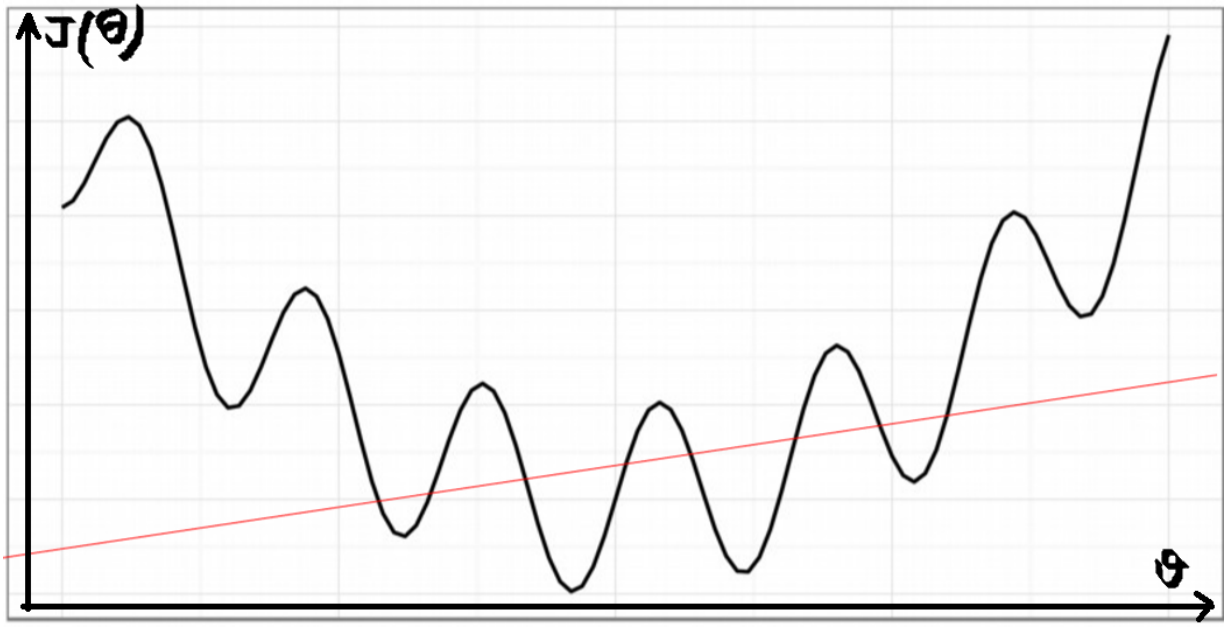
$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if y} = 1 \\ -\log(1 - h_\theta(x)) & \text{if y} = 0 \end{cases}$$

This is the cost the algorithm pays if it predicts a value hθ(x) while the actual cost label turns out to be y.

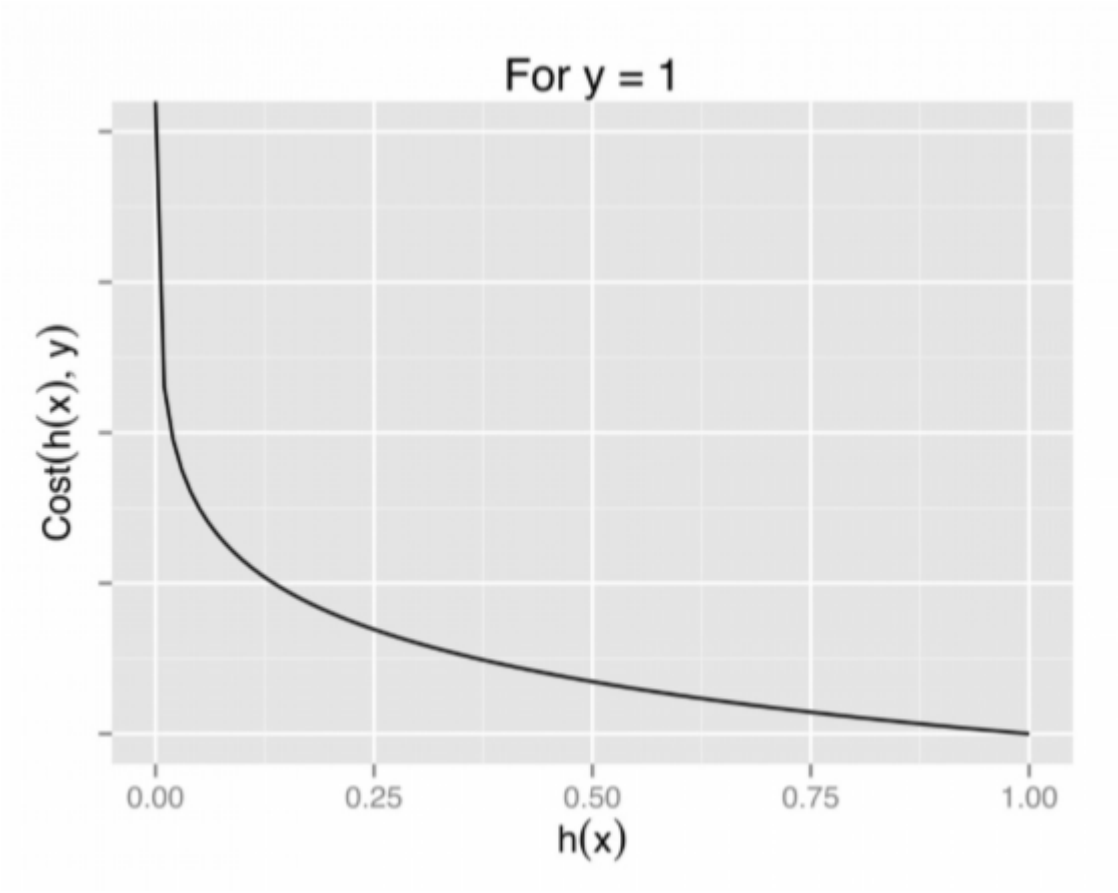
Below graph shows how the cost function looks like for Logistic Regression:

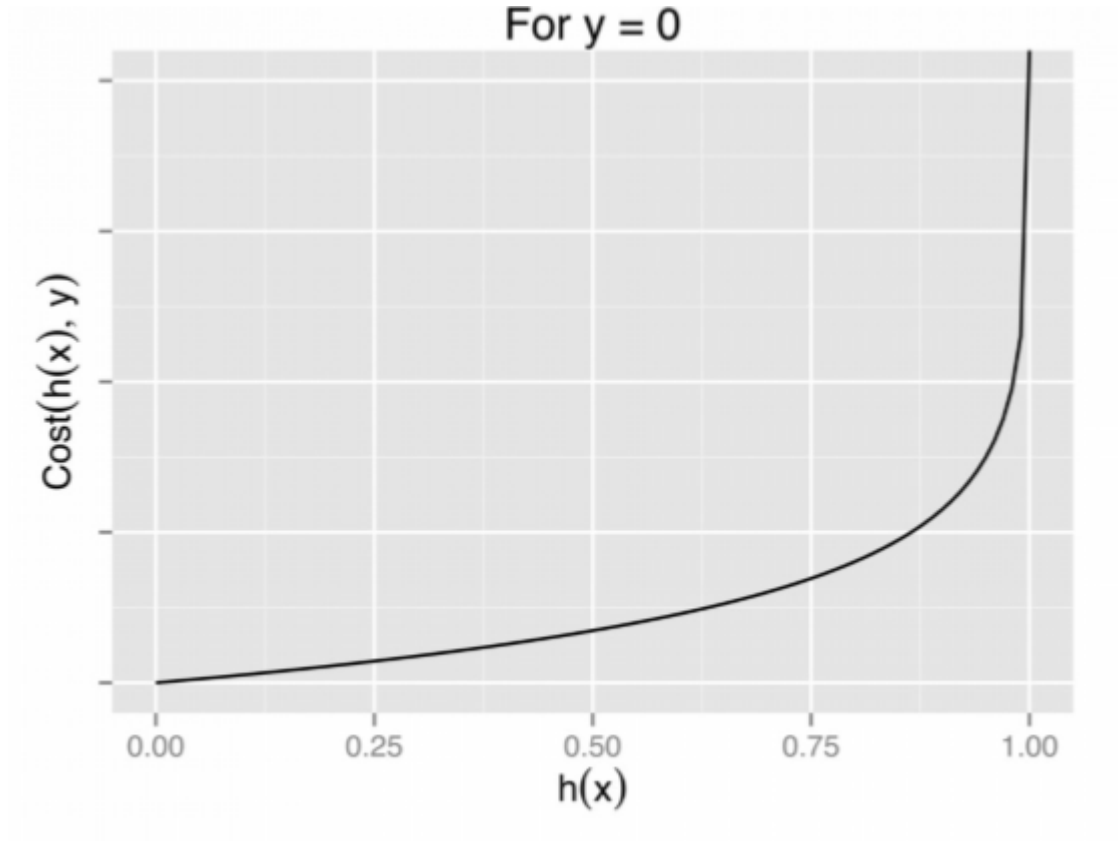
Cost Function We learnt about the cost function J(θ) in the Linear regression, the cost function represents optimization objective i.e. we create a cost function and minimize it so that we can develop an accurate model with minimum error.

The Cost function of Linear regression If we try to use the cost function of the linear regression in 'Logistic Regression' then it would be of no use as it would end up being a non-convex function with many local minimums, in which it would be very difficult to minimize the cost value and find the global minimum.



$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





The above two functions can be compressed into a single function i.e.

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

To fit parameter θ, J(θ) has to be minimized and for that Gradient Descent is required.

Gradient Descent

how do we reduce the cost value. Well, this can be done by using Gradient Descent. The main goal of Gradient descent is to minimize the cost value. i.e. min J(θ).

Now to minimize our cost function we need to run the gradient descent function on each parameter i.e.

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Objective: To minimize the cost function we have to run the gradient descent function on each parameter

Want $\min_\theta J(\theta)$:

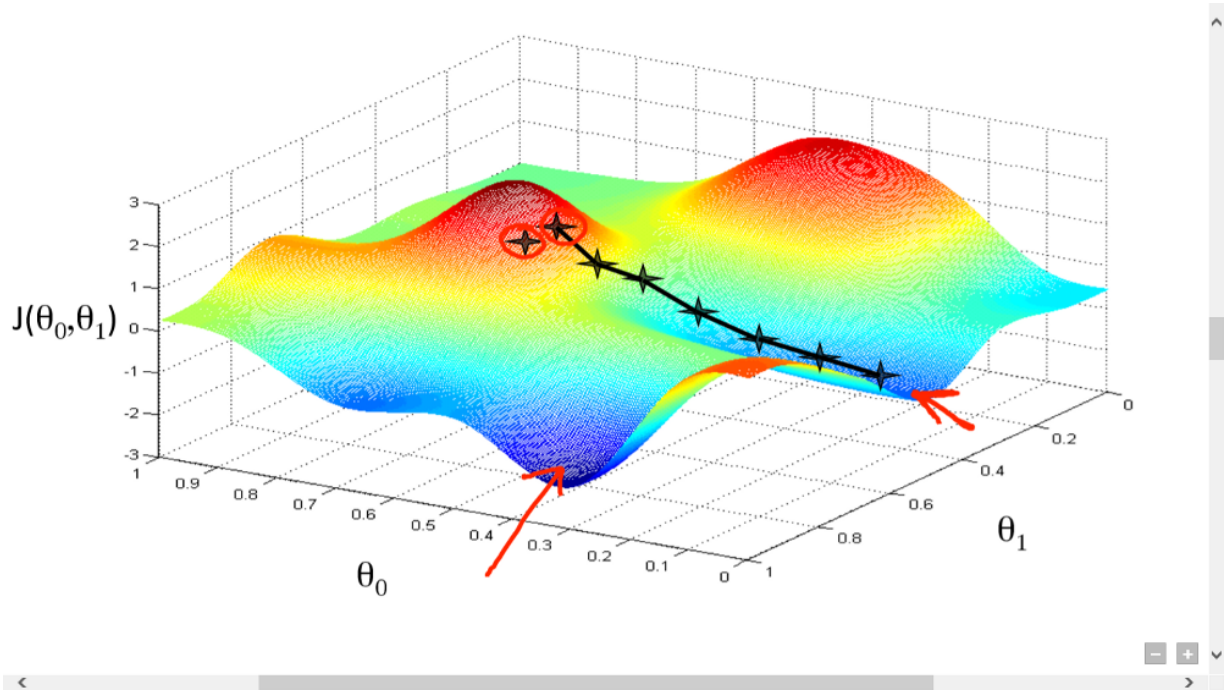
Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

Gradient descent has an analogy in which we have to imagine ourselves at the top of a mountain valley and left stranded and blindfolded, our objective is to reach the bottom of the hill. Feeling the slope of the terrain around you is what everyone would do. Well, this action is analogous to calculating the gradient descent, and taking a step is analogous to one iteration of the update to the parameters.



1. Gradient Descent is an iterative process to compute weight coefficients w, which minimise the cost function $J(w)$ i.e. sum of squared errors between actual (y) & predicted labels $(h(w^T x))$ i.e.
$J(w) = \frac{1}{2m} \sum^m (y - h(w^T x))^2$, where $h(z)$ is hypothesis function, m is number of samples. In case of logistic regression, $h(z) = \frac{1}{1+e^{-z}}$

2. For Gradient Descent to converge to global minimum, it has to be a convex function. But, above $J(w)$ will be a concave function.

3. So, we remodel cost function for logistic regression as
$J(w) = \frac{1}{2m} \sum^m y \log(h(w^T x)) + (1 - y) \log(1 - h(w^T x))$, so that it will be a convex function satisfying all constraints.

Cost Function of Logistic regression

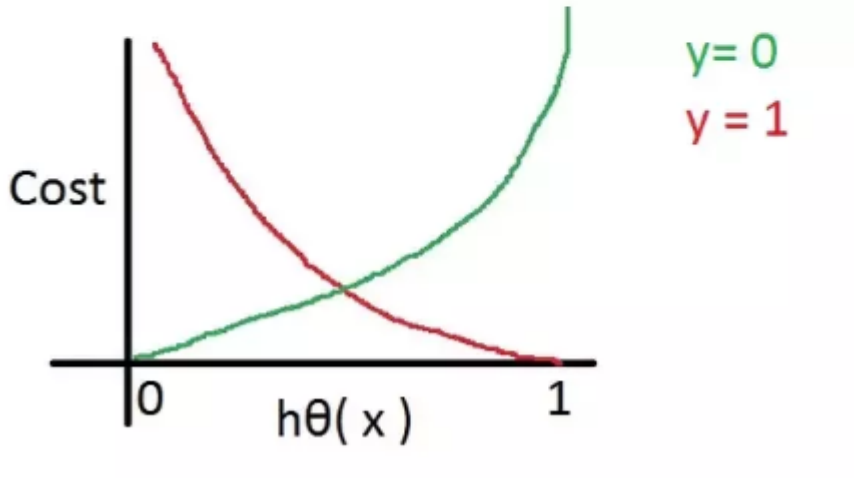
Logistic regression finds an estimate which minimizes the inverse logistic cost function.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i)) \quad (2)$$

Where $h_\theta(x)$is defined as follows,

$$h_\theta(x) = \frac{1}{1+e^{-\theta(x)}} \quad (3)$$

In order to understand the above cost function in a better way, please see the below diagram.



y = actual label (It takes 0 for negative class and 1 is positive class)

$h_\theta(x)$ = Predicted probabilities by logistic regression.

If an actual label of a particular data point $(y(x_i))$ is zero , then the cost of logistic function will be given by green graph. If an actual label of a particular data point $(y(x_i))$is one , then the cost of logistic function will be given by red graph.

It shows that, If an actual label of a particular data point $(y(x_i))$ is zero and the predicted probability of x_i is one, then the cost of the logistic function will be very high. Similarly, If an actual label of a particular data point $(y(x_i))$and the predicted probability of x_i are same, then the cost of a logistic function will be zero. So, we need to find an estimate $(\hat{\beta})$ in such a way that the cost function will have to be minimum.

The above graph shows that the logistic cost function is convex cost function, so we don't need to worry about local minimum. But, it is not possible to find a global minimum point using closed form solution as linear regression $(\hat{\beta} = (X^T X)^{-1} X^T y)$ because the sigmoid function is non-linear.

Gradient Descent Algorithm

We can use many optimization algorithms like gradient descent, Conjugate gradient, BFGS to find the global minimum point, which is nothing but estimate of β. Among those algorithms, the most popular one is gradient descent algorithm. It doesn't find an estimate in a single step like regression. It moves the estimate value towards optimum iteratively. We need to choose the suitable step size for gradient algorithm, which decide the number of iteration require for our estimate to converge optimal point. It is important tuning parameter for an algorithm. Too small step size will make the algorithm convergence slower. Similarly, too large step size may skip the optimal point. Some of the advanced optimization algorithm like BFGS finds the optimal step size itself.

A typical usage of logistic regression is classification, assigning a set of variables to a class from a predefined set of classes. Applying an input to the model's parameters results in a model selecting the input's class.

You want to train your model to classify with as little error es possible. The cost function quantifies the error, as it compares the model's response with the correct answer. Gradient descent is a learning algorithm, that given the value of the cost/loss function finds new values for the parameters that result in a smaller error. Descent comes from minimizing the error. The gradient part refers to how the algorithm updates the model's parameters at each training step. The update is based on the gradient of the cost function w.r.t to each parameter. The gradient is the first partial derivative of the loss function w.r.t. to a parameter.

Let's assume your data is composed of examples each consisting of 2 independent variables (e.g. x1, x2) and a class label y. Your logistic regression model provides a parameter for each indep. variable (e.g. theta1, theta2).

Your model's response h(X, Theta) is a function of x and the parameters.

Your loss function is a distance measure between h and the correct answer y. If you tried out all different parameter combinations for a single data sample, you would get a surface, where the height at each point tells you how wrong your model is if it uses that set of parameters. You need to select the lowest point on that surface. An exhaustive search for the lowest point is not feasible. We therefore opt for an algorithm like learning descent that will start at a random location on this unexplored surface and explore it by taking iterative steps in the direction with largest descent. It calculate the slope along each axis on this surface and goes in the direction of where the loss decreases.

ML | Cost function in Logistic Regression

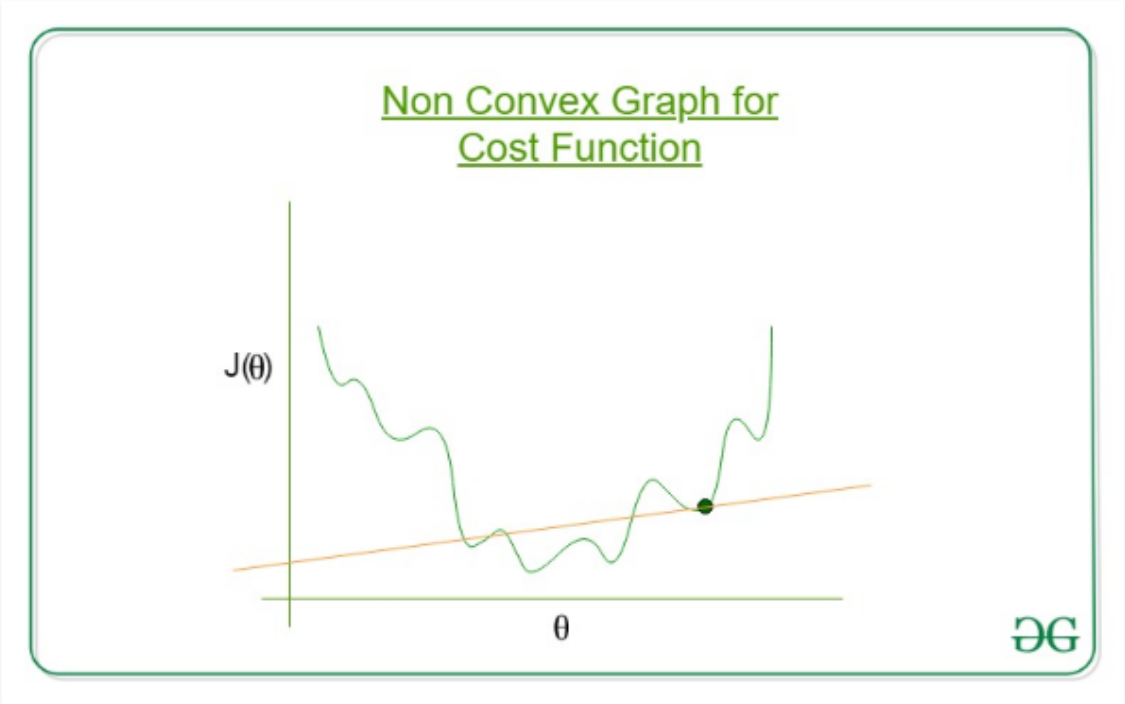
In the case of Linear Regression, the Cost function is –

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}[h_\Theta(x^{(i)}) - y^{(i)}]^2$$

But for Logistic Regression,

$$h_\Theta(x) = g(\Theta^T x)$$

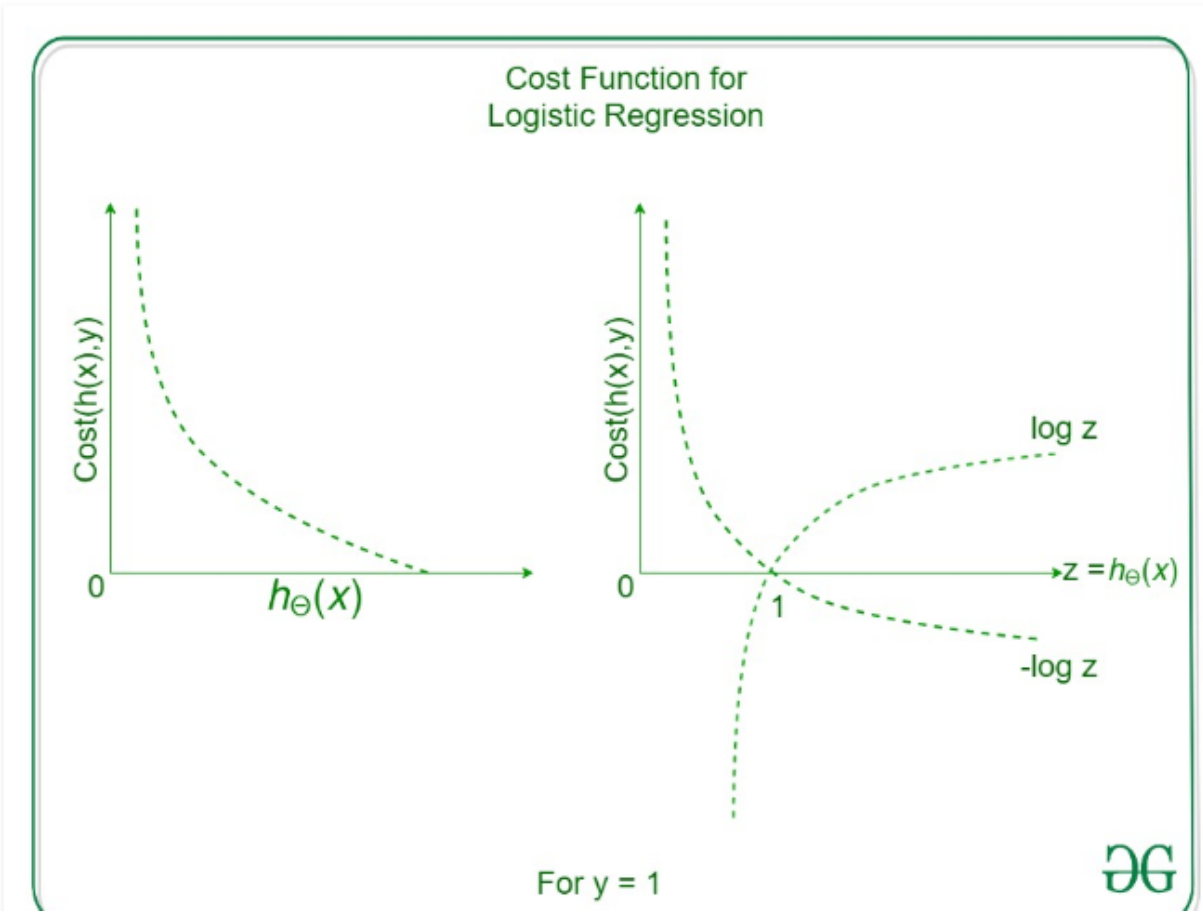
It will result in a non-convex cost function. But this results in cost function with local optima's which is a very big problen for Gradient Descent to compute the global optima.



So, for Logistic Regression the cost function is

$$\text{Cost}(h_\Theta(x), y) = \begin{cases} -log(h_\Theta(x)) & if \quad y = 1 \\ -log(1 - h_\Theta(x)) & if \quad y = 0 \end{cases}$$

If y = 1



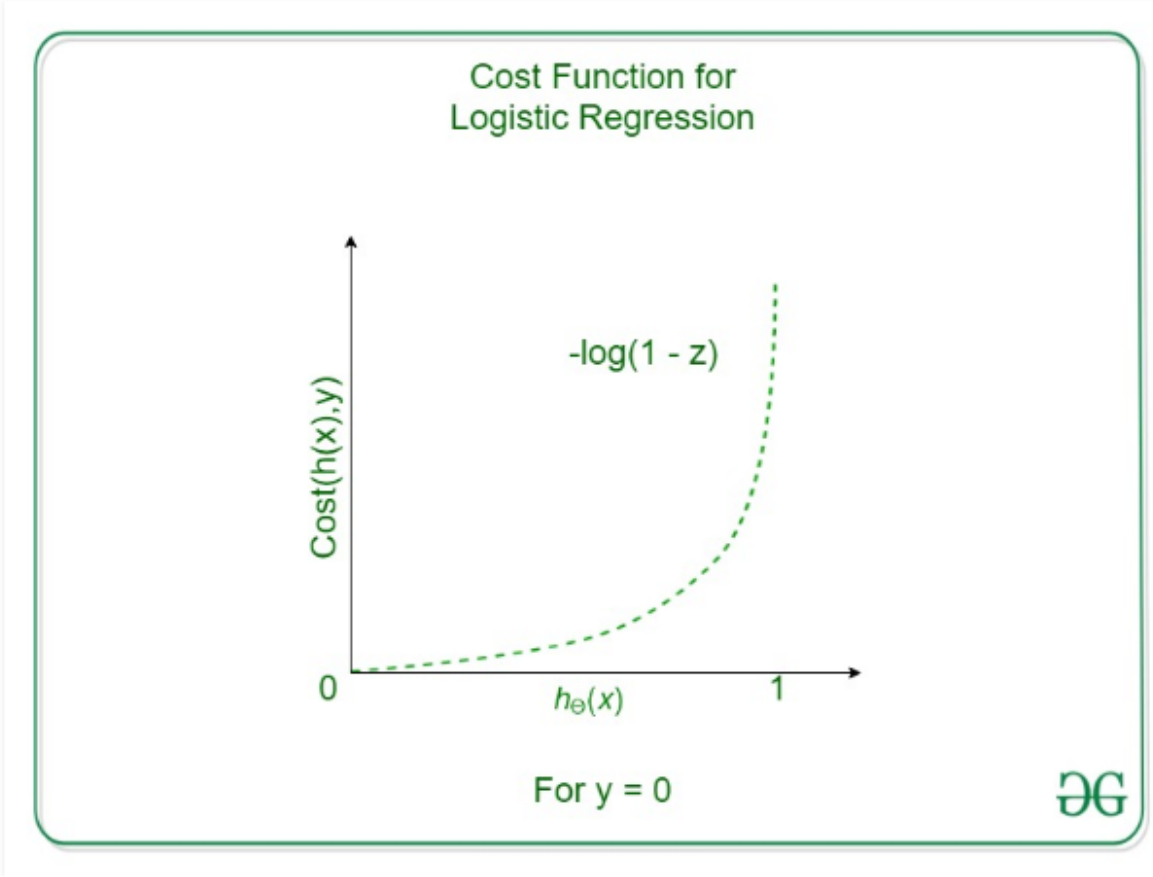
Cost = 0 if y = 1, $h_\theta(x) = 1$

But as,

$h_\theta(x)$ -> 0

Cost -> Infinity

If y = 0



So,

$$\text{Cost}(h_\Theta(x), y) = \begin{cases} 0 & if \quad h_\Theta(x) = y \\ \infty & if \quad y = 0 \quad and \quad h_\Theta(x) \to 1 \\ \infty & if \quad y = 1 \quad and \quad h_\Theta(x) \to 0 \end{cases}$$

$$\text{Cost}(h_\Theta(x), y) = -y log(h_\Theta(x)) - (1 - y) log(1 - h_\Theta(x))$$

$$J(\Theta) = \frac{-1}{m} \sum_{i=1}^{m} Cost(h_\Theta(x), y)$$

To fit parameter θ, J(θ) has to be minimized and for that Gradient Descent is required.

Gradient Descent – Looks similar to that of Linear Regression but the difference lies in the hypothesis $h_\theta(x)$

$$\Theta_j := \Theta_j - \alpha \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

=========

1. The *cost function* of logistic regression is NOT the same as sigmoid function. **Cost function** (or loss function) refers to a function that denotes how much error there is and thereby lending itself as a means of estimation since we want to minimize it. Estimation becomes optimization. However, sigmoid function was adopted NOT to compute the error but rather calibrate the somewhat unreasonable results within $(-\infty, \infty)$ to another interval $[0, 1]$ which makes sense for probabilities. (Because you're modeling a conditional probability $\mathbb{P}(y = 1_n | X, \beta) = \exp(X\beta) / \{1_n + \exp(X\beta)\}$.
2. The logarithm of sigmoid function is NOT CONVEX. Sigmoid function is log-concave, which is technically not very different but still.

====

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function (commonly called as loss/cost functions in machine learning and deep learning). To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. If, instead, one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent. Gradient descent is also known as steepest descent.

Before going through the variants of gradient descent let’s first understand how gradient descent algorithm works. For this let’s take an example of logistic regression model. For convenience, let’s suppose that our model has only two parameters: one weight w and bias b. For this the algorithm can be written as:

- Initialize our w and b with random guesses.(e.g. say w=1, b=1).

- Select a value for learning rate α. learning rate is a factor that determines how big or small the step should be on each iterations. If the learning rate is too small, it would take long time to converge and thus will be computationally too expensive. In the same way if the value of learning rate is too large, it will overshoot the minimum and fail to converge.

- The data should be normalized to a suitable scale if is highly varying. It will decrease the chances of error. There are various techniques of normalization. To read about normalization you can visit this page.

- Suppose our cost function/ loss function (for brief about loss/cost functions visit here.) which is to be minimized be $J(w,b)$. On each iteration, we take partial derivative of cost function $J(w,b)$ with respect to the parameters (w,b) which are called gradients of loss function w.r.t weights and bias:

$$\frac{\partial}{\partial w} J(w,b) = \nabla_w J$$

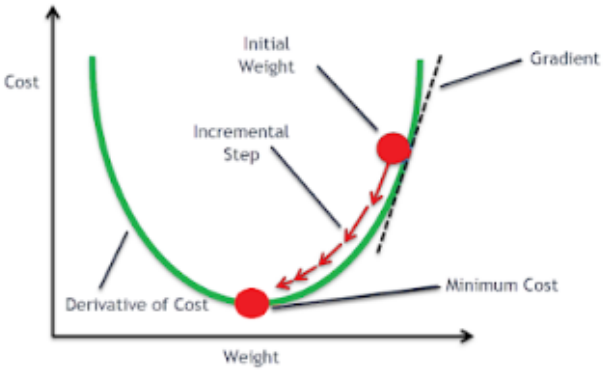
$$\frac{\partial}{\partial b} J(w,b) = \nabla_b J$$

- We then update our previous weight w and bias b as shown below:

$$w = w - \alpha \nabla_w J$$

$$b = b - \alpha \nabla_b J$$

- We continue this process until the cost function converges.



Now let's discuss in brief about the variants of gradient descent algorithms.

• Batch Gradient Descent

It is also simply called gradient descent. In this algorithm, we consider all of our examples (whole data set) on each iteration when we update our parameters.

Update equation:

$$w = w - \alpha \nabla_w J(w)$$

Advantages:

i. It has unbiased estimate of gradient i.e. lower chances of error.

ii. We can use a fixed learning rate during training for our entire example.

Disadvantages:

i. It takes long time to converge if we have large data set.

ii. It is computationally expensive.

• Mini-batch Gradient Descent

In this algorithm, instead of going through entire examples (whole data set), we perform gradient descent algorithm taking several mini-batches. So even we have large number of training examples, it is processed in batches of certain example (batch size). We divide our data set into several mini batches say n batches with certain batch size.

The batch size can be tuned by us. Generally it is chosen as power of 2, examples 32, 64, 128 etc. It is done because some hardware such as GPUs achieves better runtime with batch size as power of 2.

Update equation:

$$w = w - \alpha \nabla_w J(x^{[i:i+b]}, y^{[i:i+b]}; w)$$

Advantages

i. It is faster than the batch gradient descent.

ii. Random selection of examples will help to avoid examples that are very similar and do not contribute much to the learning.

Disadvantages

i. Mini-batch requires the configuration of an additional "mini-batch size" hyperparameter for the learning algorithm.

ii. Error information must be accumulated across mini-batches of training examples like batch gradient descent.

- Stochastic Gradient Descent

Stochastic gradient descent, often abbreviated SGD, is a variation of the gradient descent algorithm that calculates the error and updates the model for each example in the training dataset. In other words, can say it is a mini batch gradient descent with batch size 1 and has batches equal to the number of training examples.

Update equation:

$$w = w - \alpha \nabla_w J(x^i, y^i; w)$$

Advantages:

i. It is simple to understand and implement, especially for beginners.

ii. This increase the model updates frequency that can result in faster learning on some problems.

Disadvantages:

i. Updating the model frequently can be computationally too much expensive in comparison to the above variants.

ii. The frequent updates may result in a noisy gradient signal, that may cause the model parameters and turn the model error to jump around.

======

Decision Boundary

To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.

Say, if predicted_value ≥ 0.5, then classify email as spam else as not spam.

Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary.

Cost Function

$$\text{Cost}(h_\Theta(x), Y(\text{actual})) = -\log(h_\Theta(x)) \text{ if } y=1$$
$$-\log(1 - h_\Theta(x)) \text{ if } y=0$$

Figure 4: Cost Function of Logistic Regression

Why cost function which has been used for linear can not be used for logistic?

Linear regression uses mean squared error as its cost function. If this is used for logistic regression, then it will be a non-convex function of parameters (theta). Gradient descent will converge into global minimum only if the function is convex.

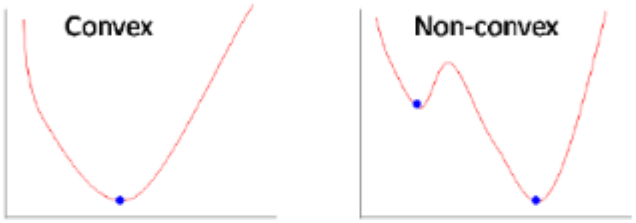
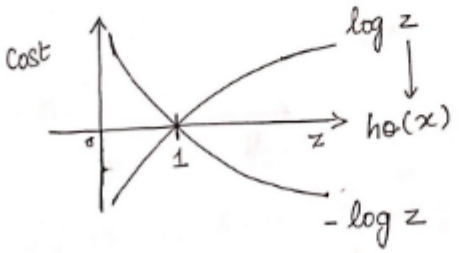


Figure 5: Convex and non-convex cost function

Cost function explanation



$$\text{Cost}\,(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

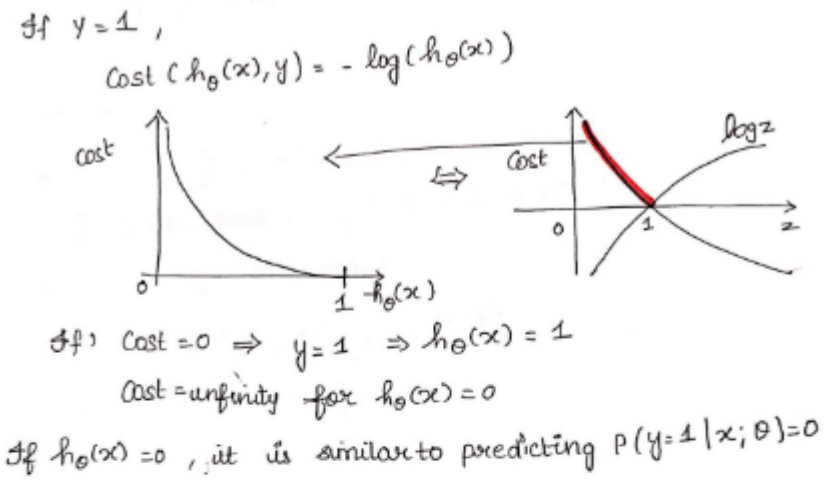
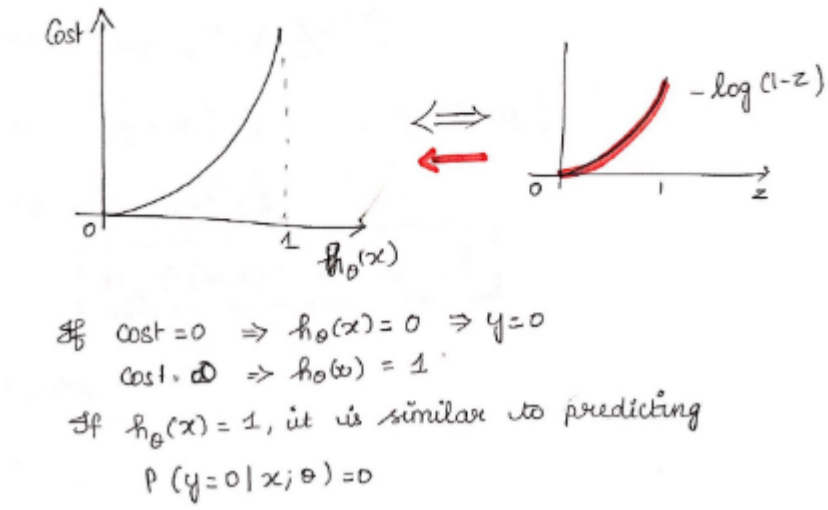


Figure 6: Cost Function part 1



Simplified cost function

$\text{Cost}(h_\Theta(x), y) = -y \log(h_\Theta(x)) - (1-y) \log(1- h_\Theta(x))$

If y = 1, (1-y) term will become zero, therefore $-\log(h_\Theta(x))$ alone will be present

If y = 0, (y) term will become zero, therefore $-\log(1- h_\Theta(x))$ alone will be present

Figure 8: Simplified Cost Function

Why this cost function?

Let us consider,

* $\hat{y} = P(y=1|x)$

  $\hat{y}$ is the probability that $y=1$, given x

* $1-\hat{y} = P(y=0|x)$

* $P(y|x) = \hat{y}^{y} \cdot (1-\hat{y})^{(1-y)}$

If $y=1 \Rightarrow P(y|x) = \hat{y}$

Figure 9: Maximum Likelihood Explanation part-1

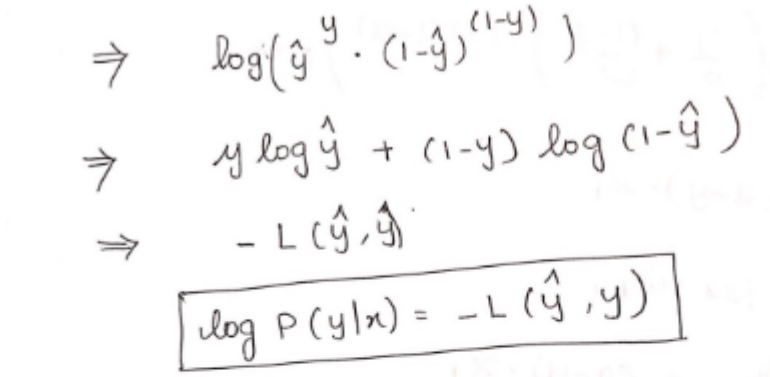


Figure 10: Maximum Likelihood Explanation part-2

This negative function is because when we train, we need to maximize the probability by minimizing loss function. Decreasing the cost will increase the maximum likelihood assuming that samples are drawn from an identically independent distribution.

Deriving the formula for Gradient Descent Algorithm

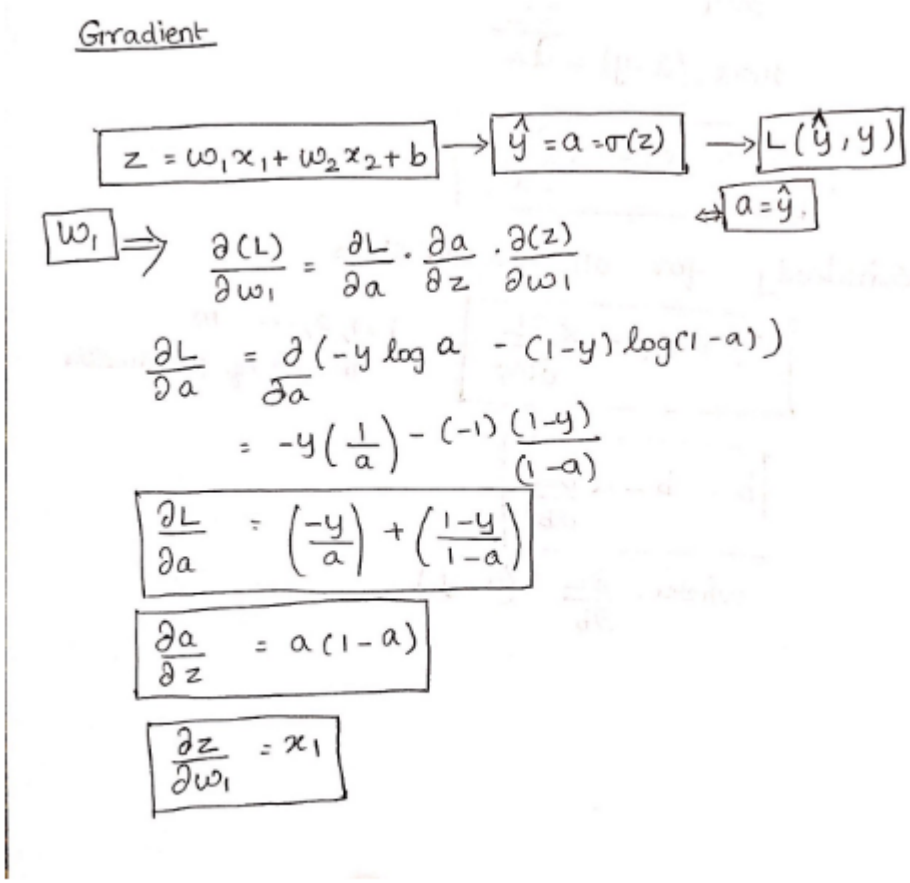


Figure 11: Gradient Descent Algorithm part 1

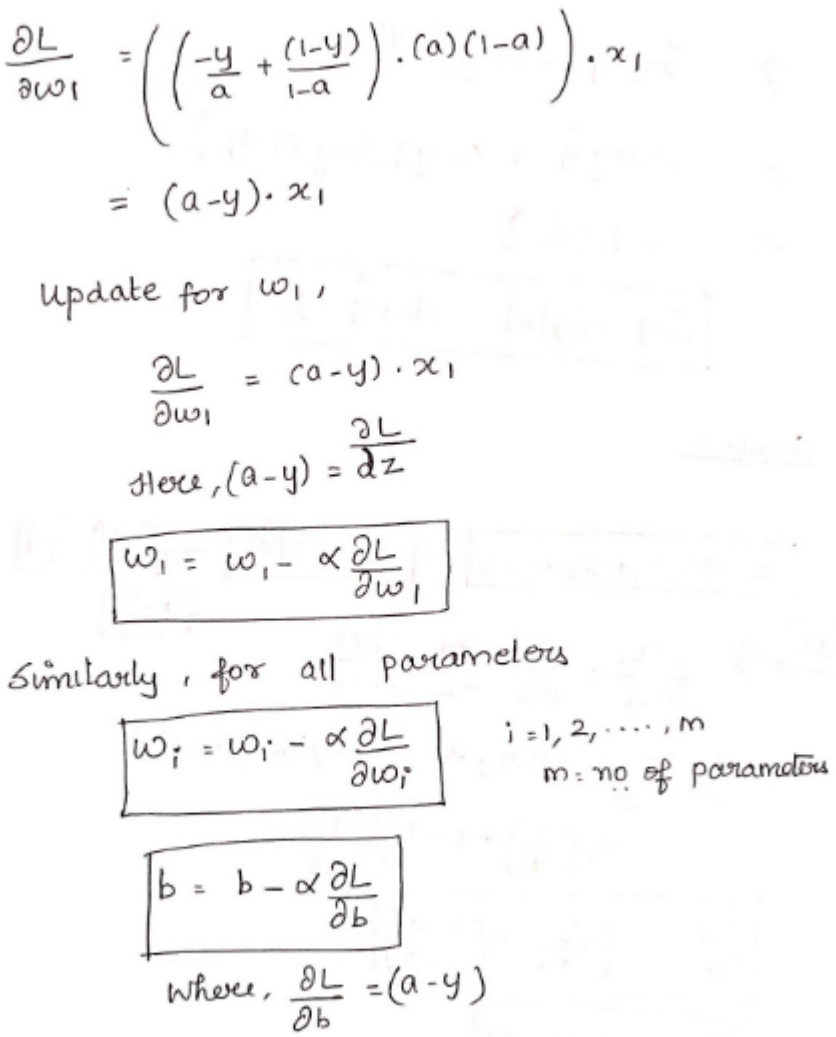


Figure 12: Gradient Descent part 2

5. What is multiple Logistic Function?

In multi class classification each sample is assigned to one and only one target label. Eg: An animal can be cat or dog but not both at the same time.

A problem with more than two classes is often called a multi-class classification problem. we used softmax function

multi classification used softmax function

At the center of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

logit(P(Y=1|x1,…,xk))=β0+β1×1+⋯+βnxn

With real constants β0,β1,…,βn. The logit model can be estimated via maximum likelihood estimation using numerical methods as we will do in Python. Multiple logistic regression can be determined by a stepwise procedure using the step function.
This function selects models to minimize AIC, not according to p-values

Multiple logistic regression is sensitive to the presence of multicollinearity and this makes using stepwise regression less recommended unless you carefully study multicollinearity between the regressor variables.

When the outcome is binary then you would do binary (simple or multiple) logistic regression. Simple means you have only one predictor. Multiple means you have more predictors. Stepwise is an (unrecommended) method for multiple logistic regression to select some predictors among all. Understanding the output and interpreting the results is a complicated task.

Multiple logistic regression finds the equation that best predicts the value of the Y variable for the values of the X variables.

ln[Y/(1−Y)]=a+b1X1+b2X2+b3X3...

You find the slopes (b1, b2, etc.) and intercept (a) of the best-fitting equation in a multiple logistic regression using the maximum-likelihood method, rather than the least-squares method used for multiple linear regression. Maximum likelihood is a computer-intensive technique; the basic idea is that it finds the values of the parameters under which you would be most likely to get the observed results.

You might want to have a measure of how well the equation fits the data, similar to the R2 of multiple linear regression. However, statisticians do not agree on the best measure of fit for multiple logistic regression. Some use deviance, D, for which smaller numbers represent better fit, and some use one of several pseudo-R2 values, for which larger numbers represent better fit.

Multiclass classification: classification task with more than two classes. Each sample can only be labelled as one class.

For example, classification using features extracted from a set of images of fruit, where each image may either be of an orange, an apple, or a pear. Each image is one sample and is labelled as one of the 3 possible classes. Multiclass classification makes the assumption that each sample is assigned to one and only one label - one sample cannot, for example, be both a pear and an apple.

6. What is multilabel Logistic Regression?

Multi label classification: Classification task where each sample is mapped to a set of target labels (more than one class). Eg: A news article can be about sports, a person, and location at the same time.

A problem where an example is assigned multiple classes is called a multi-label classification problem.

Multinomial logistic regression is used to model nominal outcome variables, in which the log odds of the outcomes are modeled as a linear combination of the predictor variables.

Multinomial logistic regression is a simple extension of binary logistic regression that allows for more than two categories of the dependent or outcome variable. Like binary logistic regression, multinomial logistic regression uses maximum likelihood estimation to evaluate the probability of categorical membership.

Multinomial logistic regression is used when you have a categorical dependent variable with two or more unordered levels (i.e. two or more discrete outcomes).

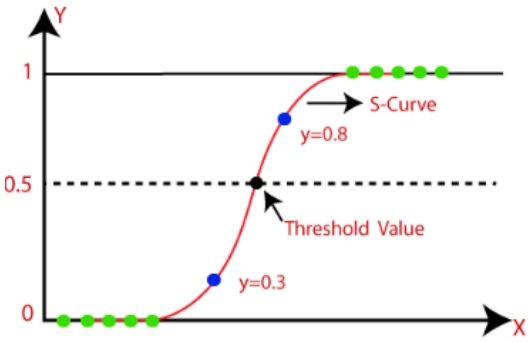
7. How does the Logistic Regression Algorithm learn?

Learning Logistic Regression Model:

Consider a scenario where we need to classify whether a patient has diabetes or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.3 and the threshold value is 0.6, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1.**
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems.**
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Type of Logistic Regression: On the basis of the categories, Logistic Regression can be classified into three types:

Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc. Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep" Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

8. Explain the confusion matrix.

Confusion matrixis used to summarize, describe or evaluate the performance of a Binary classification task or model. A confusion matrix is a performance measurement technique for Machine learning classification. The Key concept of confusion matrix is that it **calculates the no. of correct & incorrect predictions** which is further summarized with the no. of count values and breakdown into each classes.

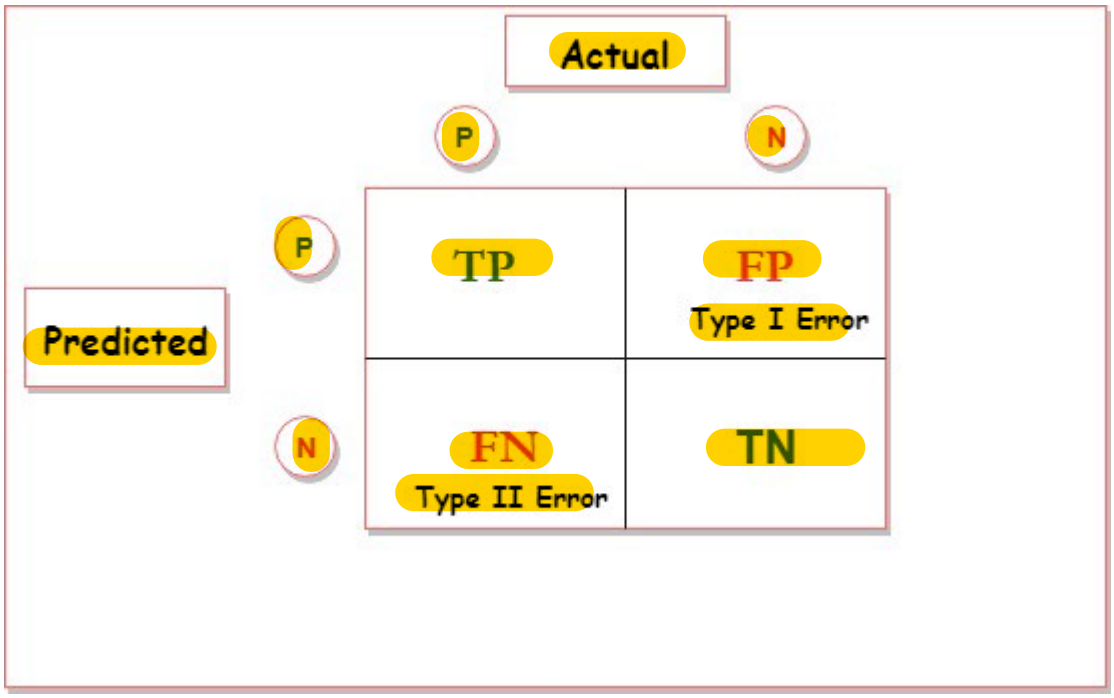
Here, it comes with 2 rows & 2 columns. Consisting of, True Positives, True Negatives False Positives, False Negatives.

Confusion Matrix is a useful machine learning method which allows you to measure Recall, Precision, Accuracy, and AUC-ROC curve. Below given is an example to know the terms True Positive, True Negative, False Negative, and True Negative.

What is Confusion Matrix and why you need it?

Well, it is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.



You can compute the **accuracy test** from the confusion matrix:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Here we have kept the predictions as rows and actual values as columns Diving little deeper in each of the terms:

• Positive (P) : Actual is positive (for example: is an apple).

• Negative (N): Actual is not positive (for example: is not an apple).

True positive: correctly classified or detected.

True Positive (TP): Actual is positive, and is predicted to be positive.

False positive (Type 1 Error): incorrectly classified or detected. It represents the type I error.

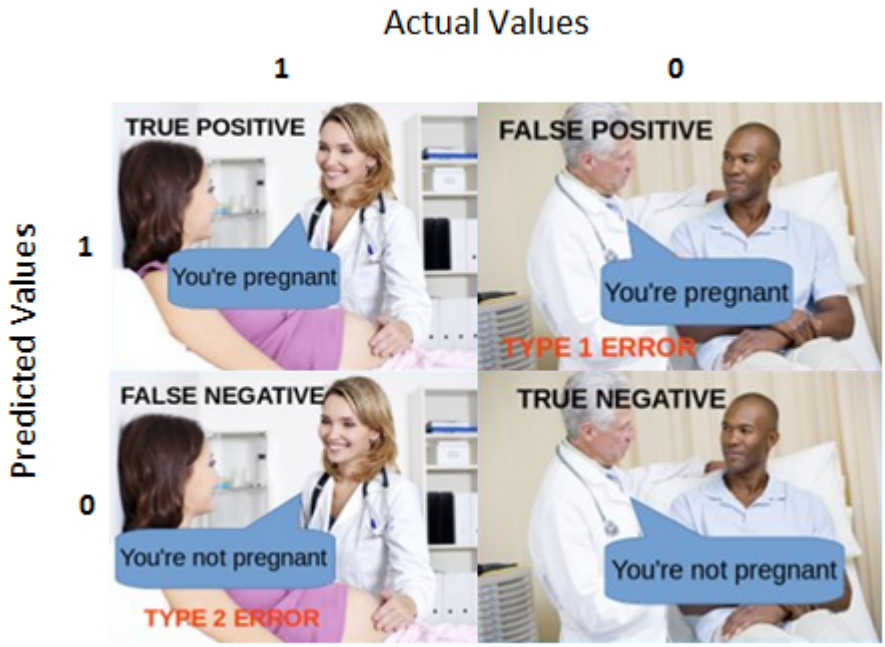
False Positive (FP): Actual is negative, but is predicted positive.

False negative (Type 2 Error): incorrectly rejected. It represents the type II error.

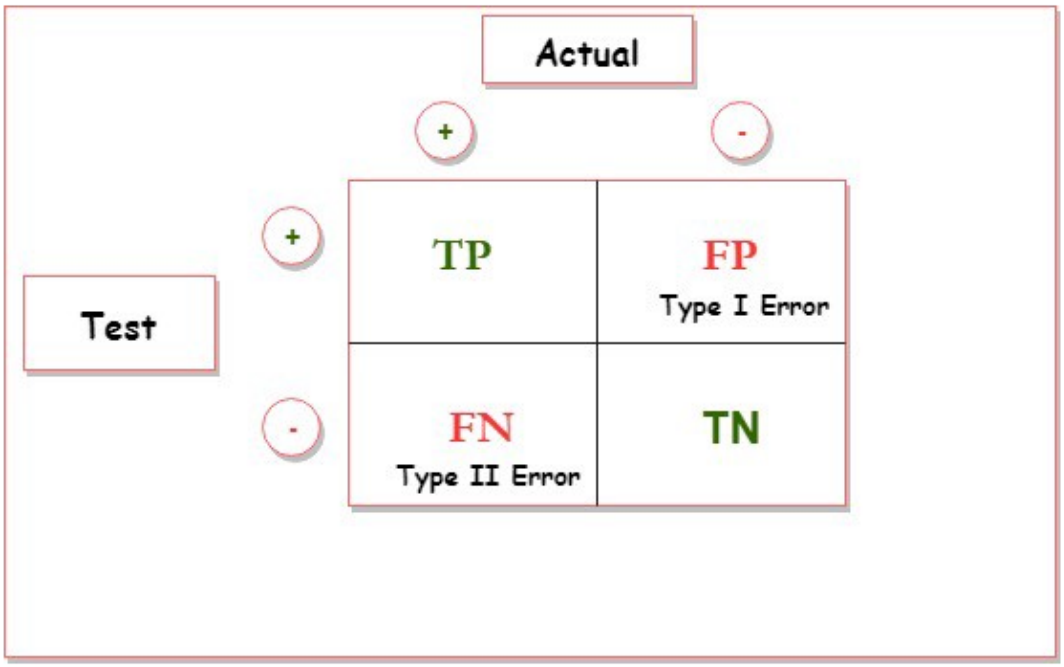
False Negative (FN): Actual is positive, but is predicted negative.

True negative: correctly rejected.

True Negative (TN): Actual is negative, and is predicted to be negative.



Let's understand it with an example of HIV Test:



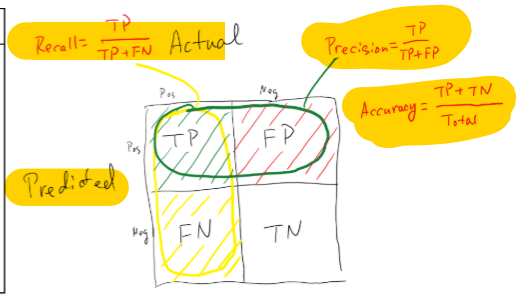
Seeing the above diagram we can say

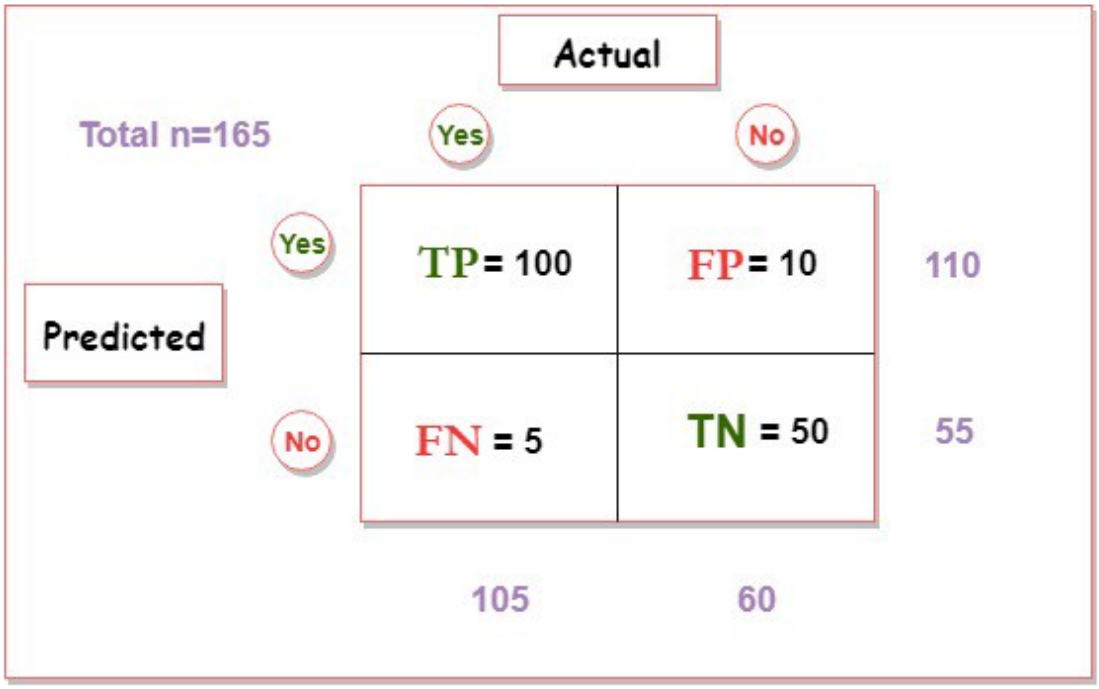
- **True Positives (TP):** We **tested** for Positive (Will Have) & they **actual** have the disease.
- **True Negatives (TN):** We **tested** for Negative (Will Not Have) & they **actual** don't have the disease.
- **False Positives (FP):** We **tested** for Positive (Will Have) & they **actual** don't have the disease. (Also known as a "Type I error.")
- **False Negatives (FN):** We **tested** for Negative (Will Not Have) & they **actual** have the disease. (Also known as a "Type II error.")

How to Calculate Confusion Matrix for a 2-class classification problem?

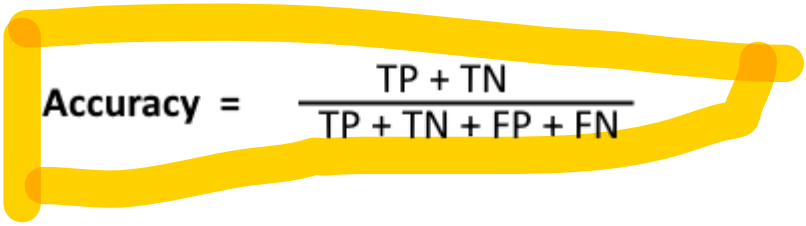
Let's understand confusion matrix through math.

y	y pred	output for threshold 0.6	Recall	Precision	Accuracy
0	0.5	0			
1	0.9	1			
0	0.7	1			
1	0.7	1	**1/2**	**2/3**	**4/7**
1	0.3	0			
0	0.4	0			
1	0.5	0			

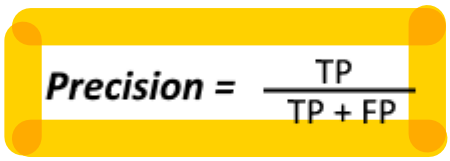




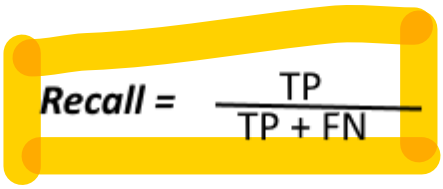
- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total = (10+5)/165 = 0.09
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  - TP/actual yes = 100/105 = 0.95
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  - FP/actual no = 10/60 = 0.17
- **Specificity:** When it's actually no, how often does it predict no?
  - TN/actual no = 50/60 = 0.83
  - equivalent to 1 minus False Positive Rate
- **Precision:** When it predicts yes, how often is it correct?
  - TP/predicted yes = 100/110 = 0.91
- **Prevalence:** How often does the yes condition actually occur in our sample?
  - actual yes/total = 105/165 = 0.64



Accuracy



Precision



Recall

High recall, low precision: Indicates that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.

Low recall, high precision: Indicates that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP).

F-score

It is the Harmonic mean of the two values which we have i.e. Precision and Recall.

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

F-Score

It considers both the Precision and Recall of the procedure to compute the score.

Higher the F-score, the better will be the predictive power of the classification procedure.

A score of 1 means the classification procedure is perfect. Lowest possible F-score is 0.

You can compute the **accuracy test** from the confusion matrix:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall

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

Out of all the positive classes, how much we predicted correctly. It should be high as possible.

Precision

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

Out of all the positive classes we have predicted correctly, how many are actually positive.

and **Accuracy** will be

Out of all the classes, how much we predicted correctly, which will be, in this case 4/7. It should be high as possible.

F-measure

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

It is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.

How to Calculate a Confusion Matrix

Here, is step by step process for calculating a confusion Matrix in data mining

- Step 1) First, you need to test dataset with its expected outcome values.
- Step 2) Predict all the rows in the test dataset.
- Step 3) Calculate the expected predictions and outcomes:

1. The total of correct predictions of each class.
2. The total of incorrect predictions of each class.

After that, these numbers are organized in the below-given methods:

- Every row of the matrix links to a predicted class.
- Every column of the matrix corresponds with an actual class.
- The total counts of correct and incorrect classification are entered into the table.
- The sum of correct predictions for a class go into the predicted column and expected row for that class value.
- The sum of incorrect predictions for a class goes into the expected row for that class value and the predicted column for that specific class value.

Other Important Terms using a Confusion matrix

- **Positive Predictive Value(PVV):** This is very much near to precision. One significant difference between the two-term is that PVV considers prevalence. In the situation where the classes are perfectly balanced, the positive predictive value is the same as precision.

- **Null Error Rate:** This term is used to define how many times your prediction would be wrong if you can predict the majority class. You can consider it as a baseline metric to compare your classifier.

- **F Score:** F1 score is a weighted average score of the true positive (recall) and precision.

- **Roc Curve:** Roc curve shows the true positive rates against the false positive rate at various cut points. It also demonstrates a trade-off between sensitivity (recall and specificity or the true negative rate).
- **Precision:** The precision metric shows the accuracy of the positive class. It measures how likely the prediction of the positive class is correct.

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

The maximum score is 1 when the classifier perfectly classifies all the positive values. Precision alone is not very helpful because it ignores the negative class. The metric is usually paired with Recall metric. Recall is also called sensitivity or true positive rate.

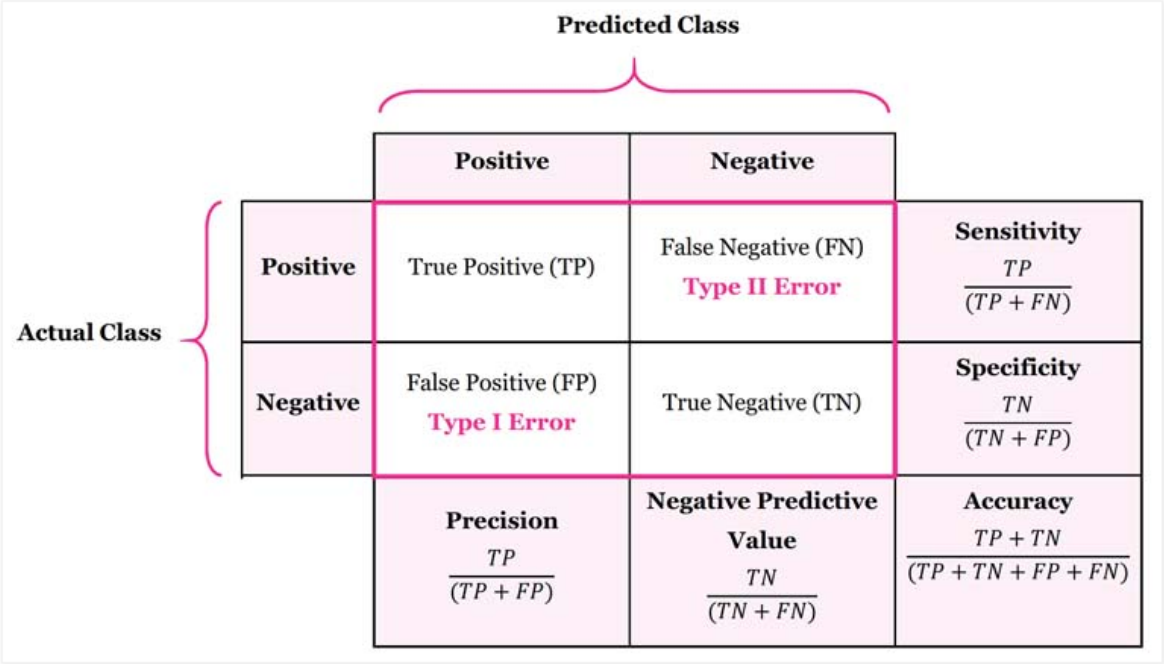
- **Sensitivity:** Sensitivity computes the ratio of positive classes correctly detected. This metric gives how good the model is to recognize a positive class.

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

Why you need Confusion matrix?

Here are pros/benefits of using a confusion matrix.

- It shows how any classification model is confused when it makes predictions.
- Confusion matrix not only gives you insight into the errors being made by your classifier but also types of errors that are being made.
- This breakdown helps you to overcomes the limitation of using classification accuracy alone.
- Every column of the confusion matrix represents the instances of that predicted class.
- Each row of the confusion matrix represents the instances of the actual class.
- It provides insight not only the errors which are made by a classifier but also errors that are being made.



		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Table 1. Confusion matrix with advanced classification metrics

9. What is Accuracy?

We can find the confusion matrix with the help of *confusion_matrix()* function of sklearn. With the help of the following script, we can find the confusion matrix of above built binary classifier −

```
from sklearn.metrics import confusion_matrix
```

Output

```
[[ 73  7]
 [ 4 144]]
```

Accuracy

It may be defined as the number of correct predictions made by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

For above built binary classifier, TP + TN = 73+144 = 217 and TP+FP+FN+TN = 73+7+4+144=228.

Hence, Accuracy = 217/228 = 0.951754385965 which is same as we have calculated after creating our binary classifier.

Precision

Precision, used in document retrievals, may be defined as the number of correct documents returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

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

For the above built binary classifier, TP = 73 and TP+FP = 73+7 = 80.

Hence, Precision = 73/80 = 0.915

Recall or Sensitivity

Recall may be defined as the number of positives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

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

For above built binary classifier, TP = 73 and TP+FN = 73+4 = 77.

Hence, Precision = 73/77 = 0.94805

Specificity

Specificity, in contrast to recall, may be defined as the number of negatives returned by our ML model. We can easily calculate it by confusion matrix with the help of following formula −

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

For the above built binary classifier, TN = 144 and TN+FP = 144+7 = 151.

Hence, Precision = 144/151 = 0.95364

Accuracy is the proportion of the total number of predictions that are correct.

		Predicted Class: Spam	Predicted Class: Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

Table 6. Accuracy in confusion matrix

Accuracy = (45+30)/(45+20+5+30) = 75%

The 75% of examples are correctly classified by the classifier.

Precision is ratio of total number of correctly classified positive examples and the total number of predicted positive examples. It shows correctness achieved in positive prediction.

		Predicted Class: Spam	Predicted Class: Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

Table 5. Precision in confusion matrix

Precision = 45/(45+5)= 90%

The 90% of examples are classified as spam are actually spam.

Specificity is also know as *True Negative Rate*. It is measure of negative examples labeled as negative by classifier. There should be high specificity. For instance, proportion of emails which are non-spam among all non-spam emails.

		Predicted Class: Spam	Predicted Class: Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

Table 4. Specificity in confusion matrix

$specificity = 30/(30+5) = 85.71\%$.

The 85.71% non-spam emails are accurately classified and excluded from all spam emails.

Sensitivity is also referred as *True Positive Rate* or *Recall*. It is measure of positive examples labeled as positive by classifier. It should be higher. For instance, proportion of emails which are spam among all spam emails.

		Predicted Class: Spam	Predicted Class: Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

Table 3. Sensitivity in confusion matrix

$Sensitivity = 45/(45+20) = 69.23\%$.

The 69.23% spam emails are correctly classified and excluded from all non-spam emails.

F1 score is a weighted average of the recall (sensitivity) and precision. F1 score might be good choice when you seek to balance between Precision and Recall.

$$\text{F1 Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

It helps to compute recall and precision in one equation so that the problem to distinguish the models with low recall and high precision or vice versa could be solved.

Accuracy and Components of Confusion Matrix

After the confusion matrix is created and we determine all the components values, it becomes quite easy for us to calculate the accuracy. So, let us have a look at the components to understand this better.

- **Classification Accuracy**

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

From the above formula, the sum of TP (True Positive) and the TN (True Negative) are the correct predicted results. Hence in order to calculate the accuracy in percentage, we divide with all the other components. However, there are some problems in the accuracy and we cannot completely depend on it.

Let us consider that our dataset is completely imbalanced. In this Scenario, 98% accuracy can be good or bad based on the problem statement. Hence we have some more key terms which will help us to be sure about the accuracy we calculate. The terms are as given below:

- **TPR (True Positive Rate) or Sensitivity:**

True Positive rate which is also known as Sensitivity measures the percentage of the True Positive with respect to the Total Actual Positives which is indicated by (TP+ FN)

	Predicted Class 1 Value EG: 1	**Predicted Class 2 Value EG:0**	**Total**
Actual Class 1 Value EG: 1	***TP (True Positive)***	***FN (False Negative)***	***Total Actual Positives***
Actual Class 2 Value EG: 0	FP (False Positive)	TN (True Negative)	Total Actual Negatives

TPR= True Positive/ (True Positive + False Negative

- **TNR (True Negative Rate) or Specificity:**

True Negative Rate or Specificity measures the proportion of actual negatives with respect to the Total Negatives

	Predicted Class 1 Value EG: 1	Predicted Class 2 Value EG:0	Total
Actual Class 1 Value EG: 1	TP (True Positive)	FN (False Negative)	Total Actual Positives
Actual Class 2 Value EG: 0	***FP (False Positive)***	***TN (True Negative)***	***Total Actual Negatives***

TNR= True Negative/ (True Negative+ False Positive)

- **False Positive Rate(FPR):**

False Positive Rate is the percentage of Predicted False Positive (FP) to the Total No of Predicted Positive Results (TP + FP).

	Predicted Class 1 Value EG: 1	Predicted Class 2 Value EG:0
Actual Class 1 Value EG: 1	***TP (True Positive)***	FN (False Negative)
Actual Class 2 Value EG: 0	***FP (False Positive)***	TN (True Negative)
	Sum of Total Predicted Positive	Sum of Total Predicted Negative

FPR= False Positive/ (True Positive + False Positive)

- **False Negative Rate (FNR):**

False Negative Rate is the percentage of Predicted False Negative (FP) to the Total No of Predicted Negative Results (TN + FN).

	Predicted Class 1 Value EG: 1	Predicted Class 2 Value EG:0
Actual Class 1 Value EG: 1	TP (True Positive)	***FN (False Negative)***
Actual Class 2 Value EG: 0	FP (False Positive)	***TN (True Negative)***
	Sum of Total Predicted Positive	***Sum of Total Predicted Negative***

FNR= False Negative/ (False Negative + True Negative)

Precision, Recall, and F-Measure

- **Recall:**

A recall is similar to the True Positive Rate and it is the ratio of the Total number of correctly predicted positive values(TP) to all the Positive Values.

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

- **Precision:**

The Precision basically indicates all the points the model predicted to be positive and what percentage of them are actually Positive.

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

Precision and Recall are metrics results which focus on the positive class as shown from the above formulas.

10. Why there is a need for other metrics if 'accuracy' is already present?

AIC value: This is a measure of the quality of the model and is like Adjusted R-squared in that it accounts for the number of variables used compared to the number of observations. A low AIC is desirable.

Performance of Logistic Regression Model (Performance Metrics):

To evaluate the performance of a logistic regression model, we must consider few metrics. Irrespective of tool (SAS, R, Python) we would work on, always look for:

1. **AIC (Akaike Information Criteria)** — The analogous metric of adjusted R^2 in logistic regression is AIC. AIC is the measure of fit which penalizes model for the number of model coefficients. Therefore, we always prefer model with minimum AIC value.

2. **Null Deviance and Residual Deviance** — Null Deviance indicates the response predicted by a model with nothing but an intercept. Lower the value, better the model. Residual deviance indicates the response predicted by a model on adding independent variables. Lower the value, better the model.

3. **Confusion Matrix:** It is nothing but a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model and avoid over-fitting. This is how it looks like:

Actual		Predicted	
		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Predictive Model: Evaluation

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn}$$

predictive result / classification		actual result / classification: yes	actual result / classification: no
	yes	tp (true positive)	fp (false positive) ← Type 1 error
	no	fn (false negative)	tn (true negative)

$$\text{Precision} = \frac{tp}{tp+fp} \qquad \text{Recall} = \frac{tp}{tp+fn}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad \text{True Negative Rate} = \frac{tn}{tn+fp}$$

Confusion Matrix and ROC Curve

		Predicted Class: No	Predicted Class: Yes
Observed Class	No	TN	FP
	Yes	FN	TP

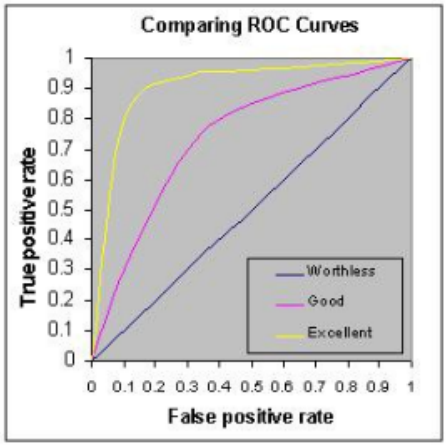
TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	= (TN+TP)/(TN+FP+FN+TP)
Precision	= TP/(FP+TP)
Sensitivity	= TP/(TP+FN)
Specificity	= TN/(TN+FP)

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

4. **ROC Curve:** Receiver Operating Characteristic (ROC) summarizes the model's performance by evaluating the trade-offs between true positive rate (sensitivity) and false positive rate (1- specificity). For plotting ROC, it is advisable to assume p > 0.5 since we are more concerned about success rate. ROC summarizes the predictive power for all possible values of p > 0.5. The area under curve (AUC), referred to as index of accuracy (A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model. Below is a sample ROC curve. The ROC of a perfect predictive model has TP equals 1 and FP equals 0. This curve will touch the top left corner of the graph.



For model performance, we can also consider likelihood function. It is called so, because it selects the coefficient values which maximizes the likelihood of explaining the observed data. It indicates goodness of fit as its value approaches one, and a poor fit of the data as its value approaches zero.

=======

How can you evaluate Logistic Regression model fit and accuracy?

In Linear Regression, we check adjusted R^2, F Statistics, MAE, and RMSE to evaluate model fit and accuracy. But, Logistic Regression employs all different sets of metrics. Here, we deal with probabilities and categorical values. Following are the evaluation metrics used for Logistic Regression:

1. Akaike Information Criteria (AIC)

You can look at AIC as counterpart of adjusted r square in multiple regression. It's an important indicator of model fit. It follows the rule: Smaller the better. AIC penalizes increasing number of coefficients in the model. In other words, adding more variables to the model wouldn't let AIC increase. It helps to avoid overfitting.

Looking at the AIC metric of one model wouldn't really help. It is more useful in comparing models (model selection). So, build 2 or 3 Logistic Regression models and compare their AIC. The model with the lowest AIC will be relatively better.

2. Null Deviance and Residual Deviance

Deviance of an observation is computed as -2 times log likelihood of that observation. The importance of deviance can be further understood using its types: Null and Residual Deviance. Null deviance is calculated from the model with no features, i.e.,only intercept. The null model predicts class via a constant probability.

Residual deviance is calculated from the model having all the features.On comarison with Linear Regression, think of residual deviance as residual sum of square (RSS) and null deviance as total sum of squares (TSS). The larger the difference between null and residual deviance, better the model.

Also, you can use these metrics to compared multiple models: whichever model has a lower null deviance, means that the model explains deviance pretty well, and is a better model. Also, lower the residual deviance, better the model. Practically, AIC is always given preference above deviance to evaluate model fit.

3. Confusion Matrix

Confusion matrix is the most crucial metric commonly used to evaluate classification models. It's quite confusing but make sure you understand it by heart. If you still don't understand anything, ask me in comments. The skeleton of a confusion matrix looks like this:

	1 (Predicted)	0 (Predicted)
1 (Actual)	True Positive	False Negative
0 (Actual)	False Positive	True Negative

As you can see, the confusion matrix avoids "confusion" by measuring the actual and predicted values in a tabular format. In table above, Positive class = 1 and Negative class = 0. Following are the metrics we can derive from a confusion matrix:

Accuracy – It determines the overall predicted accuracy of the model. It is calculated as `Accuracy = (True Positives + True Negatives)/(True Positives + True Negatives + False Positives + False Negatives)`

True Positive Rate (TPR) – It indicates how many positive values, out of all the positive values, have been correctly predicted. The formula to calculate the true positive rate is `(TP/TP + FN)`. Also, `TPR = 1 - False Negative Rate`. It is also known as Sensitivity or Recall.

False Positive Rate (FPR) – It indicates how many negative values, out of all the negative values, have been incorrectly predicted. The formula to calculate the false positive rate is `(FP/FP + TN)`. Also, `FPR = 1 - True Negative Rate`.

True Negative Rate (TNR) – It indicates how many negative values, out of all the negative values, have been correctly predicted. The formula to calculate the true negative rate is `(TN/TN + FP)`. It is also known as Specificity.

False Negative Rate (FNR) – It indicates how many positive values, out of all the positive values, have been incorrectly predicted. The formula to calculate false negative rate is `(FN/FN + TP)`.

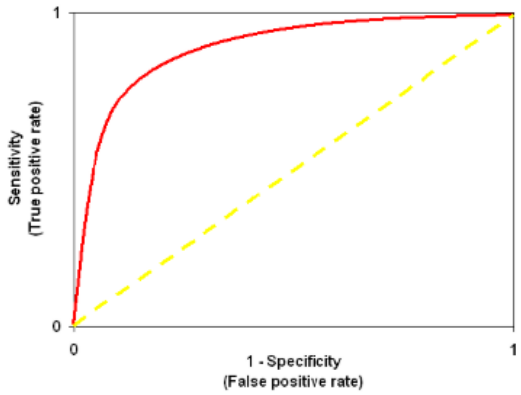
Precision: It indicates how many values, out of all the predicted positive values, are actually positive. It is formulated as:`(TP / TP + FP)`.

F Score: F score is the harmonic mean of precision and recall. It lies between 0 and 1. Higher the value, better the model. It is formulated as `2((precision*recall) / (precision+recall))`.

4. Receiver Operator Characteristic (ROC)

ROC determines the accuracy of a classification model at a user defined threshold value. It determines the model's accuracy using Area Under Curve (AUC). The area under the curve (AUC), also referred to as index of accuracy (A) or concordant index, represents the performance of the ROC curve. Higher the area, better the model. ROC is plotted between True Positive Rate (Y axis) and False Positive Rate (X Axis). In this plot, our aim is to push the red curve (shown below) toward 1 (left corner) and maximize the area under curve. Higher the curve, better the model. The yellow line represents the ROC curve at 0.5 threshold. At this point, sensitivity = specificity.

r



==========

Predict function after Logistic regression gives you probability as the output. Hence, you need to convert that probability into a binary or multinomaial variable.

Then you can compare this variable with your target variable.

Validation techniques: 1.Classification Table = sum(diagonal)/Total sum

2.Hosmer - Lemeshow (HL test)

3.ROC curve / AUC value

4.Concordance

Misclassification Error

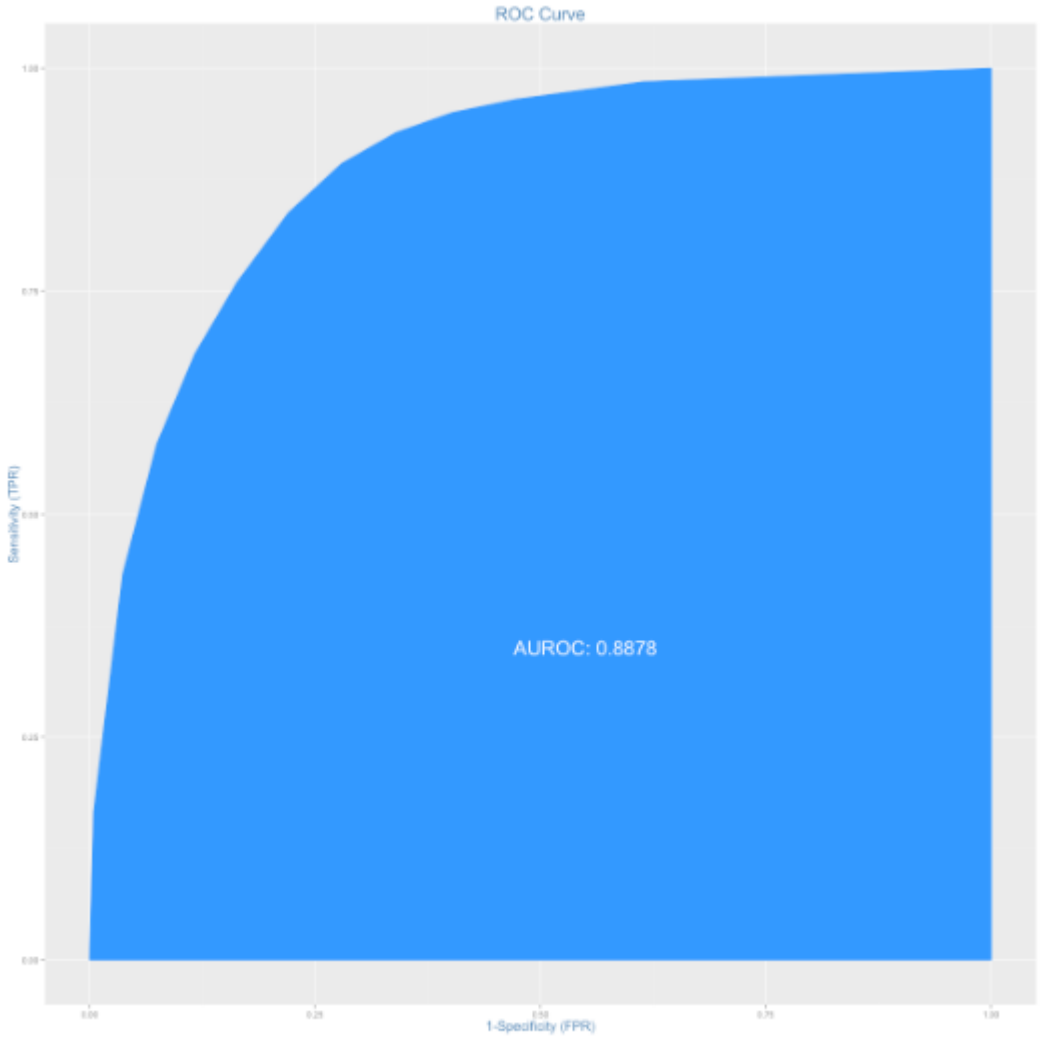
Misclassification error is the percentage mismatch of predcited vs actuals, irrespective of 1's or 0's. The lower the misclassification error, the better is your model.

```
misClassError(testData$ABOVE50K, predicted, threshold = optCutOff)
#=> 0.0899
```

ROC

Receiver Operating Characteristics Curve traces the percentage of true positives accurately predicted by a given logit model as the prediction probability cutoff is lowered from 1 to 0. For a good model, as the cutoff is lowered, it should mark more of actual 1's as positives and lesser of actual 0's as 1's. So for a good model, the curve should rise steeply, indicating that the TPR (Y-Axis) increases faster than the FPR (X-Axis) as the cutoff score decreases. Greater the area under the ROC curve, better the predictive ability of the model.

```
plotROC(testData$ABOVE50K, predicted)
```



The above model has area under ROC curve 88.78%, which is pretty good.

Concordance

Ideally, the model-calculated-probability-scores of all actual Positive's, (aka Ones) should be greater than the model-calculated-probability-scores of ALL the Negatives (aka Zeroes). Such a model is said to be perfectly concordant and a highly reliable one. This phenomenon can be measured by Concordance and Discordance.

In simpler words, of all combinations of 1-0 pairs (actuals), *Concordance* is the percentage of pairs, whose scores of actual positive's are greater than the scores of actual negative's. For a perfect model, this will be 100%. So, the higher the concordance, the better is the quality of model.

```
Concordance(testData$ABOVE50K, predicted)
#> 0.8915
```

The above model with a concordance of 89.2% is indeed a good quality model.

Specificity and Sensitivity

Sensitivity (or True Positive Rate) is the percentage of 1's (actuals) correctly predicted by the model, while, specificity is the percentage of 0's (actuals) correctly predicted. Specificity can also be calculated as 1 − *False Positive Rate*.

$$Sensitivity = \frac{\#\ Actual\ 1's\ and\ Predicted\ as\ 1's}{\#\ of\ Actual\ 1's}$$

$$Specificity = \frac{\#\ Actual\ 0's\ and\ Predicted\ as\ 0's}{\#\ of\ Actual\ 0's}$$

```
sensitivity(testData$ABOVE50K, predicted, threshold = optCutOff)
#> 0.3089
specificity(testData$ABOVE50K, predicted, threshold = optCutOff)
#> 0.9836
```

The above numbers are calculated on the validation sample that was not used for training the model. So, a truth detection rate of 31% on test data is good.

Confusion Matrix

```
confusionMatrix(testData$ABOVE50K, predicted, threshold = optCutOff)
# The columns are actuals, while rows are predicteds.
#>       0    1
#> 0 18918 1626
#> 1   314  727
```

11. What are Recall and Precision? How do they differ?

Recall or Sensitivity

The mathematical formula is:

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

Or, as the name suggests, it is a measure of: from the total number of positive results how many positives were correctly predicted by the model.

It shows how relevant the model is, in terms of positive results only.

Let's suppose in the previous model, the model gave 50 correct predictions(TP) but failed to identify 200 cancer patients(FN). Recall in that case will be:

Recall= $\frac{50}{(50+200)}$ = 0.2 (The model was able to recall only 20% of the cancer patients)

Precision

Precision is a measure of amongst all the positive predictions, how many of them were actually positive. Mathematically,

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

Let's suppose in the previous example, the model identified 50 people as cancer patients(TP) but also raised a false alarm for 100 patients(FP). Hence,

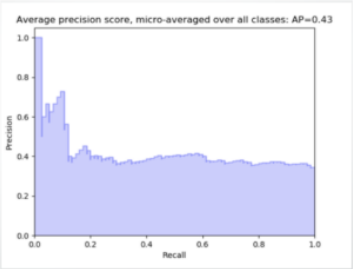
Precision= $\frac{50}{(50+100)}$ =0.33 (The model only has a precision of 33%)

But we have a problem!!

As evident from the previous example, the model had a very high Accuracy but performed poorly in terms of Precision and Recall. So, necessarily *Accuracy* is not the metric to use for evaluating the model in this case

Imagine a scenario, where the requirement was that the model recalled all the defaulters who did not pay back the loan. Suppose there were 10 such defaulters and to recall those 10 defaulters, and the model gave you 20 results out of which only the 10 are the actual defaulters. Now, the recall of the model is 100%, but the precision goes down to 50%.

A Trade-off?



As observed from the graph, with an increase in the Recall, there is a drop in Precision of the model.

As observed from the graph, with an increase in the Recall, there is a drop in Precision of the model.

So the question is - what to go for? Precision or Recall?

Well, the answer is: it depends on the business requirement.

For example, **if you are predicting cancer, you need a 100 % recall. But suppose you are predicting whether a person is innocent or not, you need 100% precision.**

Can we maximise both at the same time? No

So, there is a need for a better metric then?

Yes. And it's called an *F1 Score*

12. How does the tradeoff between recall and precision work?

What is precision recall tradeoff ?

Posted on May 10, 2019 by InterviewBuddy

Tradeoff means increasing one parameter would lead to decreasing of other. Let us explain this in context to binary classification and first define what is precision and recall. Let us call one class as positive and other as negative. Then,

1. TP represents the true positives, which is the number of positive predictions which are actually positive.
2. FP represents the false positives, which is the number of negative predictions incorrectly classified as a positive ,i.e. they were identified as positives though they were from a negative class.
3. TN represents the true negatives, which is the number of negative predictions correctly classified as negative.
4. FN represents the false negatives which is the number of positive instances incorrectly identified into the negative class, i.e. they were identified as negative but in reality they were from the positive class.

Precision is the fraction of correct positives among the total predicted positives. It is also called the accuracy of positive predictions.

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

Recall is the fraction of correct positives among the total positives in the dataset. It is indicating how many total positives of the actual dataset were covered(classified correctly) while doing prediction.

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

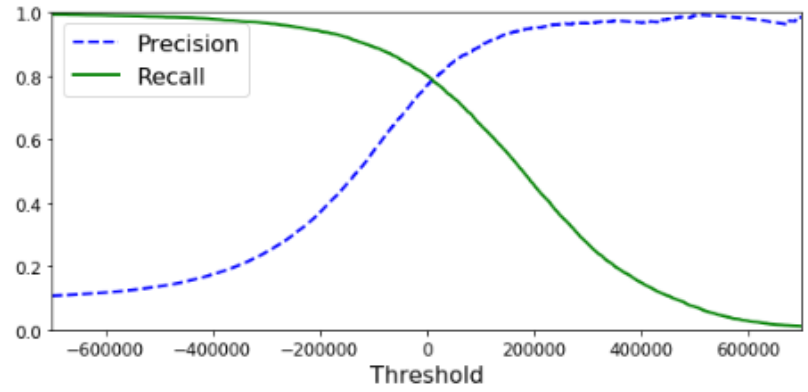


Fig 1. Precision Recall Tradeoff

In an ideal scenario where there is a perfectly separable data, both precision and recall can get maximum value of 1.0. But in most of the practical situations, there is noise in the dataset and the dataset is not perfectly separable. There might be some points of positive class closer to the negative class and vice versa. In such cases, shifting the decision boundary can either increase the precision or recall but not both. Increasing one parameter leads to decreasing of the other. In other words, binary classifier will miss classify some points always. Miss classification means classifying data point from negative class as positive and from positive class as negative. This miss rate is either compromising precision or recall score.

precision-recall tradeoff occur due to increasing one of the parameter(precision or recall) while keeping the model same. This is possible, for instance, by changing the threshold of the classifier. For eg, fig 1 is a plot for the binary classifier showing how precision and recall can vary based on threshold. This threshold decides the decision boundary. When threshold is 0, both precision and recall have the same value of around 0.8. When threshold is increased to around 200000, precision reaches close to 0.95 but recall decreases drastically to around 0.4. When threshold is decreased to -200000, recall increases to 0.95 but precision decreases to 0.4. Note that increasing or decreasing threshold is similar to shifting the decision boundary.

https://www.machinelearningaptitude.com/topics/machine-learning/what-is-precision-recall-tradeoff/ (https://www.machinelearningaptitude.com/topics/machine-learning/what-is-precision-recall-tradeoff/)

13. Explain the F1 score.

F1 Score

From the previous examples, it is clear that we need a metric that considers both Precision and Recall for evaluating a model. One such metric is the F1 score.

F1 score is defined as the harmonic mean of Precision and Recall.

The mathematical formula is: F1 score= $\frac{2*((Precision*Recall))}{(Precision+Recall))}$

14. What is specificity?

Specificity or True Negative Rate

This represents how specific is the model while predicting the True Negatives. Mathematically,

Specificity= $\frac{TN}{(TN+FP)}$ Or, it can be said that it quantifies the total number of negatives predicted by the model with respect to the total number of actual negative or non favorable outcomes.

Similarly, False Positive rate can be defined as: (1- specificity) Or, $\frac{FP}{(TN+FP)}$

15. Explain the significance of ROC.

Area under ROC may be good for comparing overall performance of two competing models. Of course, if the area is very high or very low then it is easier to say model is very good or very bad. But in most cases, if your model is reasonable, the area will be somewhere in the middle.

To cut a long story short, you may report that area to give some idea of performance. But as you are seeing, it will suffer from the same kind of problems. So better is to -

a) either fix %true +ve beforehand and choose a model & report false +ve rate there... OR

b) decide beforehand how much false +ve is tolerable and choose the model & report true +ve rate there

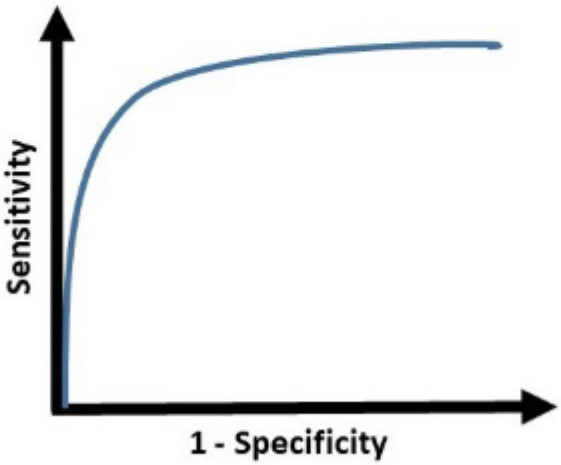
16. What is AUC, and when is it used?

Let's learn about AUC ROC Curve!

Jocelyn D'Souza Follow
Mar 15, 2018 · 5 min read

In this post, I will go through the AUC ROC curve and explain how it evaluates your model's performance. Highly suggest you go through the Confusion Matrix post before you go ahead.

All set? Let's explore it! :D

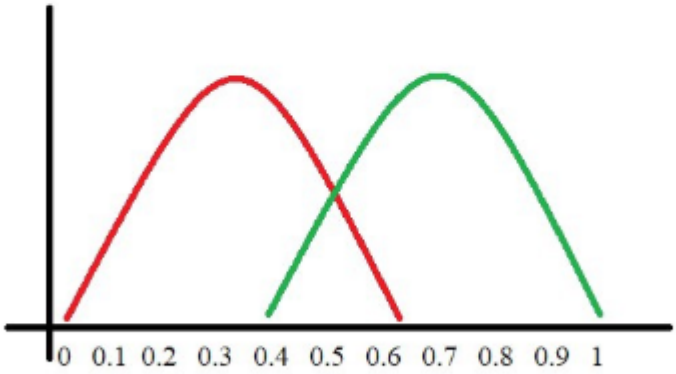


AUC ROC is one of the most important evaluation metrics for any classification model's performance.

What is ROC?

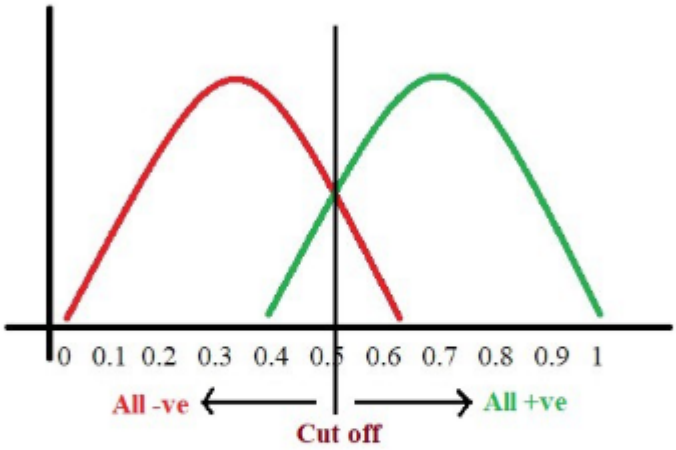
ROC (*Receiver Operating Characteristic*) Curve tells us about how good the model can distinguish between two things (*e.g If a patient has a disease or no*). Better models can accurately distinguish between the two. Whereas, a poor model will have difficulties in distinguishing between the two.

Let's assume we have a model which predicts whether the patient has a particular disease or no. The model predicts probabilities for each patient (*in python we use the "predict_proba" function*). Using these probabilities, we plot the distribution as shown below:



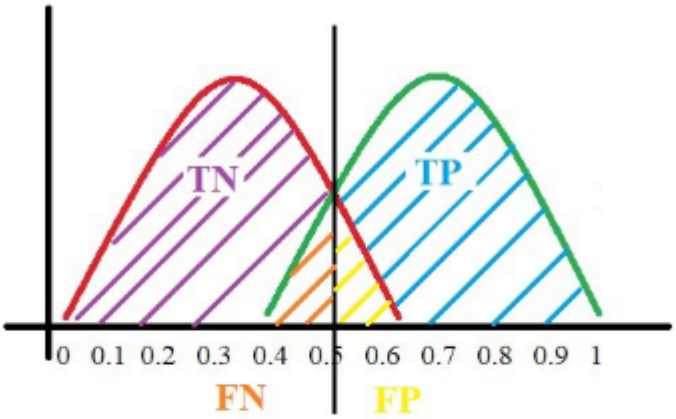
Here, the red distribution represents all the patients who do not have the disease and the green distribution represents all the patients who have the disease.

Now we got to pick a value where we need to set the cut off i.e. a threshold value, above which we will predict everyone as positive (*they have the disease*) and below which will predict as negative (*they do not have the disease*). We will set the threshold at "0.5" as shown below:



All the positive values above the threshold will be "**True Positives**" and the negative values above the threshold will be "**False Positives**" as they are predicted incorrectly as positives.

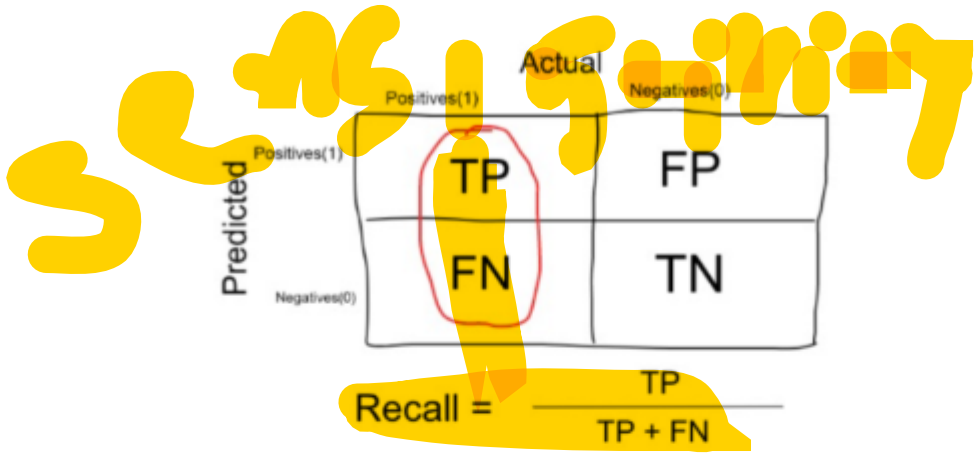
All the negative values below the threshold will be "**True Negatives**" and the positive values below the threshold will be "**False Negative**" as they are predicted incorrectly as negatives.



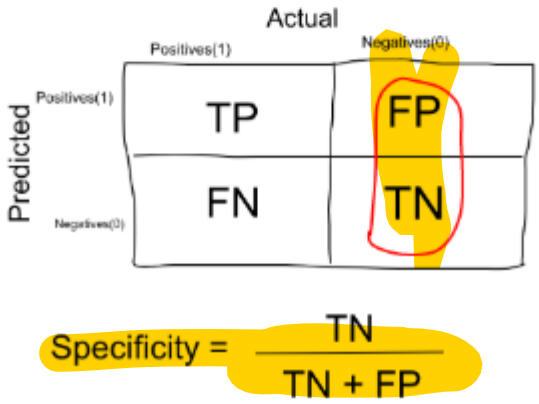
Here, we have got a basic idea of the model predicting correct and incorrect values with respect to the threshold set. Before we move on, let's go through two important terms: **Sensitivity and Specificity.**

What is Sensitivity and Specificity?

In simple terms, the proportion of patients that were identified correctly to have the disease (*i.e. True Positive*) upon the total number of patients who actually have the disease is called as Sensitivity or Recall.



Similarly, the proportion of patients that were identified correctly to not have the disease (*i.e. True Negative*) upon the total number of patients who do not have the disease is called as Specificity.



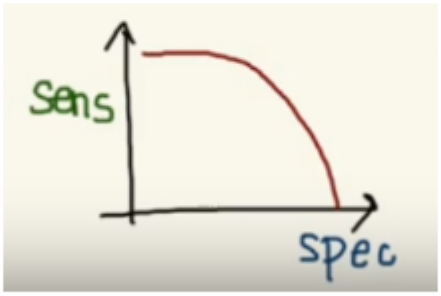
Trade-off between Sensitivity and Specificity

When we decrease the threshold, we get more positive values thus increasing the sensitivity. Meanwhile, this will decrease the specificity.

Similarly, when we increase the threshold, we get more negative values thus increasing the specificity and decreasing sensitivity.

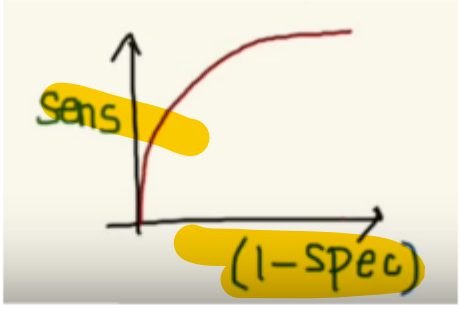
As **Sensitivity ↓ Specificity ↑**

As **Specificity ↓ Sensitivity ↑**

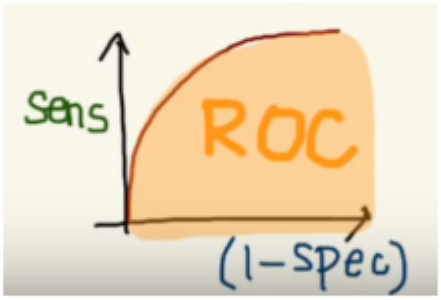


Trade off between Sensitivity & Specificity

But, this is not how we graph the ROC curve. To plot ROC curve, instead of Specificity we use (1 — Specificity) and the graph will look something like this:



So now, when the sensitivity increases, (1 — specificity) will also increase. This curve is known as the ROC curve.

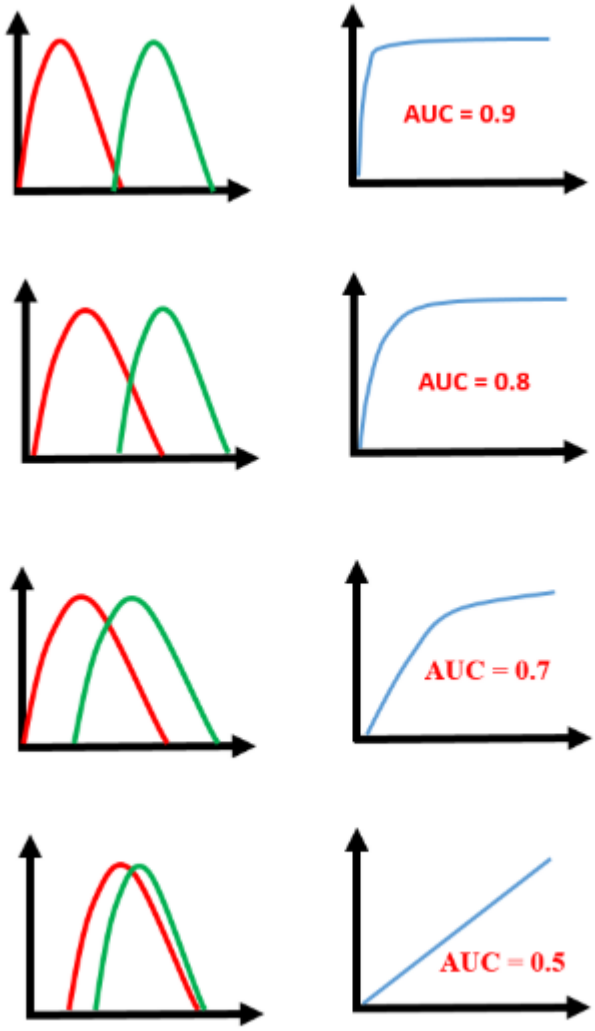


I know you might be wondering why do we use (1 — specificity)? Don't worry I'll come back to it soon. :)

Area Under the Curve

The AUC is the area under the ROC curve. This score gives us a good idea of how well the model performances.

Let's take a few examples



As we see, the first model does quite a good job of distinguishing the

positive and the negative values. Therefore, there the AUC score is 0.9 as the area under the ROC curve is large.

Whereas, if we see the last model, predictions are completely overlapping each other and we get the AUC score of 0.5. This means that the model is performing poorly and it is predictions are almost random.

Why do we use (1 — Specificity)?

Let's derive what exactly is (1 — Specificity):

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

$$1 - Specificity = 1 - \frac{TN}{TN + FP}$$

$$1 - Specificity = \frac{TN + FP - TN}{TN + FP}$$

$$1 - Specificity = \frac{FP}{TN + FP}$$

As we see above, Specificity gives us the True Negative Rate and (1 — Specificity) gives us the False Positive Rate.

So the sensitivity can be called as the "***True Positive Rate***" and (1 — Specificity) can be called as the "***False Positive Rate***".

So now we are just looking at the positives. As we increase the threshold, we decrease the *TPR* as well as the *FPR* and when we decrease the threshold, we are increasing the *TPR* and *FPR*.

Thus, AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes.

. . .

I hope I've given you some understanding on what exactly is the AUC ROC curve and how it evaluates your model's performance. To get a better understanding, suggesting you to watch this video.

17. Explain the steps for Heroku deployment.

Steps before cloud deployment:

We need to change our code a bit so that it works unhindered on the cloud, as well.

- Add a file called ‘gitignore’ inside the project folder. This folder contains the list of the files which we don’t want to include in the git repository. My gitignore file looks like:

  .idea

As I am using PyCharm as an IDE, and it’s provided by the Intellij Idea community, it automatically adds the .idea folder containing some metadata. We need not include them in our cloud app.

- Add a file called ‘Procfile’ inside the ‘reviewScrapper’ folder. This folder contains the command to run the flask application once deployed on the server:

  web: gunicorn app:app

Here, the keyword ‘web’ specifies that the application is a web application. And the part ‘app:app’ instructs the program to look for a flask application called ‘app’ inside the ‘app.py’ file. Gunicorn is a Web Server Gateway Interface (WSGI) HTTP server for Python.

- Open a command prompt window and navigate to your ‘reviewScrapper’ folder. Enter the command ‘pip freeze > requirements.txt’. This command generates the ‘requirements.txt’ file

The requirements.txt helps the Heroku cloud app to install all the dependencies before starting the webserver.

18. What difficulties did you face while deploying to Heroku?

In [ ]:

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

Logistic Regression is one of the most popular ways to fit models for categorical data, especially for binary response data in Data Modeling. It is the most important (and probably most used) member of a class of models called generalized linear models. Unlike linear regression, logistic regression can directly predict probabilities (values that are restricted to the (0,1) interval); furthermore, those probabilities are well-calibrated when compared to the probabilities predicted by some other classifiers, such as Naive Bayes. Logistic regression preserves the marginal probabilities of the training data. The coefficients of the model also provide some hint of the relative importance of each input variable.

Logistic Regression is used when the dependent variable (target) is categorical.

For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Logistic regression is generally used where the dependent variable is Binary or Dichotomous. That means the dependent variable can take only two possible values such as “Yes or No”, “Default or No Default”, “Living or Dead”, “Responder or Non Responder”, “Yes or No” etc. Independent factors or variables can be categorical or numerical variables.

Logistic Regression Assumptions:

· Binary logistic regression requires the dependent variable to be binary.

· For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.

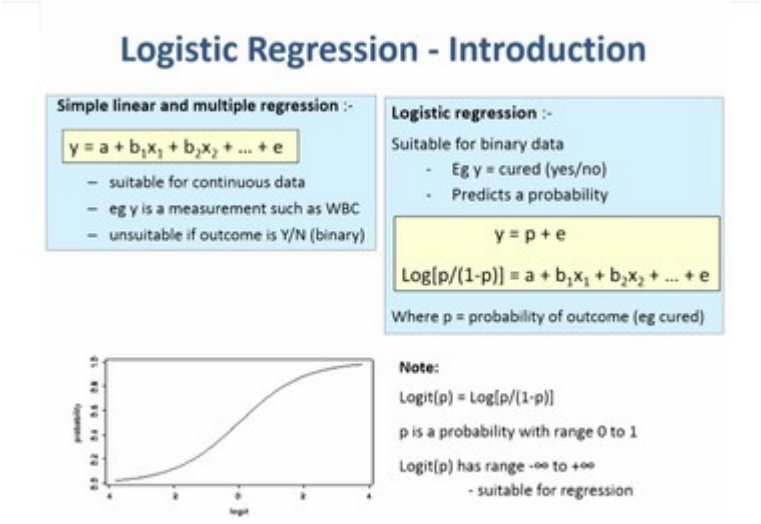
· Only the meaningful variables should be included.

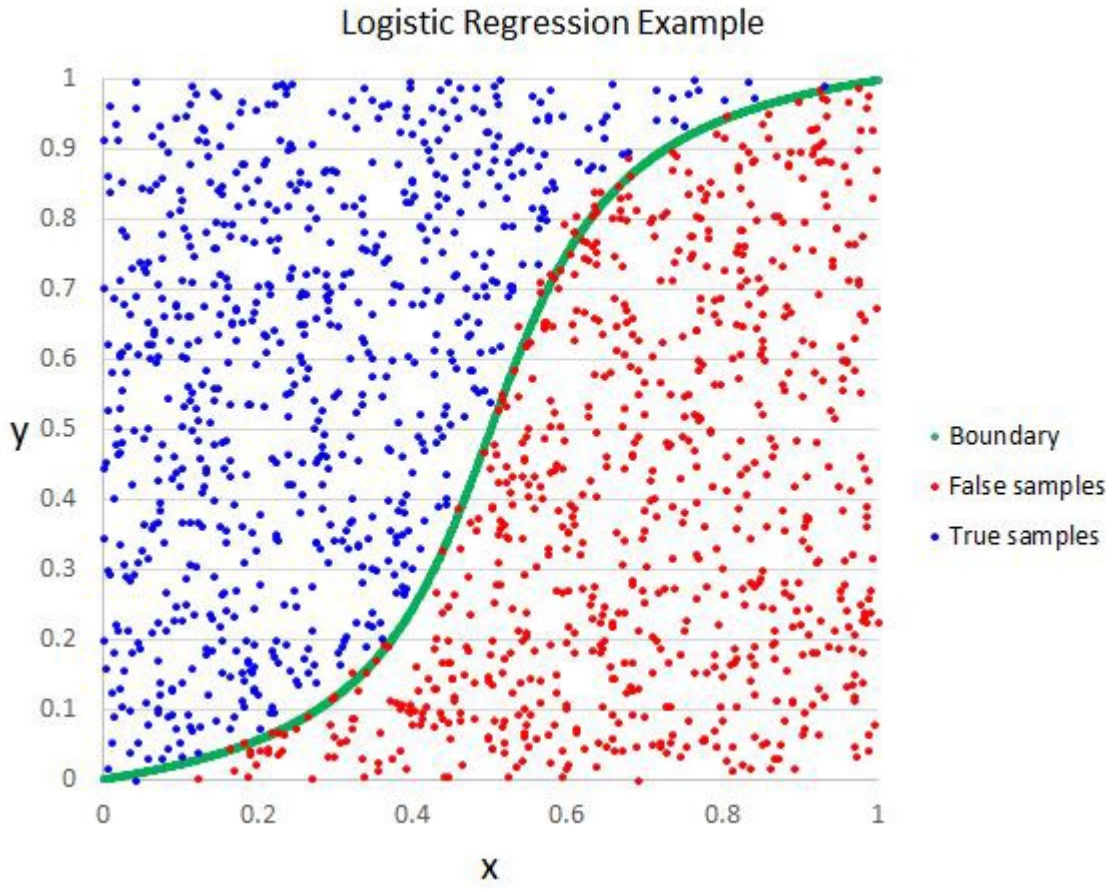
· The independent variables should be independent of each other. That is, the model should have little or no multi-collinearity.

· The independent variables are linearly related to the log odds.

· Logistic regression requires quite large sample sizes.

Even though logistic (logit) regression is frequently used for binary variables (2 classes), it can be used for categorical dependent variables with more than 2 classes. In this case it's called Multinomial Logistic Regression.





Types of Logistic Regression:

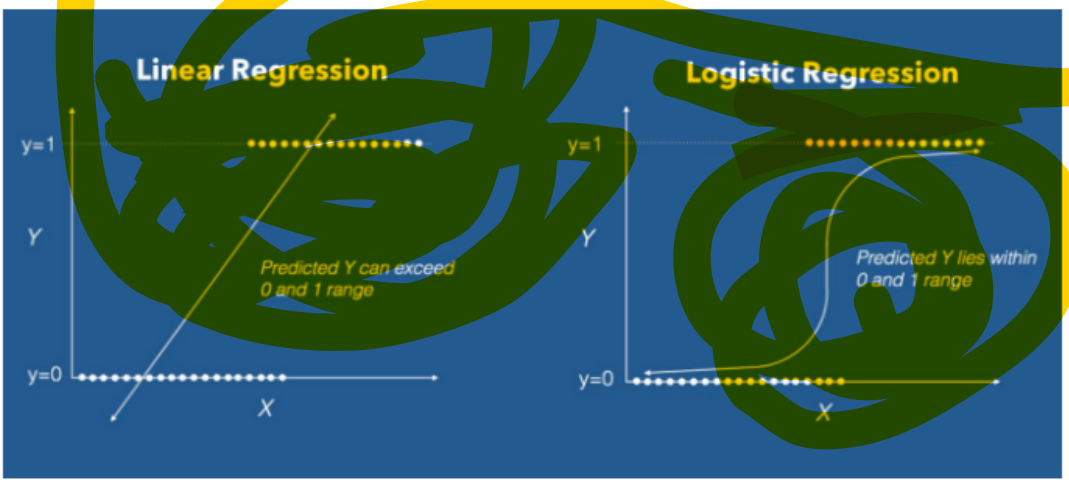
1. **Binary Logistic Regression:** The categorical response has only two 2 possible outcomes. E.g.: Spam or Not

2. **Multinomial Logistic Regression:** Three or more categories without ordering. E.g.: Predicting which food is preferred more (Veg, Non-Veg, Vegan)

3. **Ordinal Logistic Regression:** Three or more categories with ordering. E.g.: Movie rating from 1 to 5

Linear versus Logistic Regression

Linear Regression	Logistic Regression
Target is an interval variable.	Target is a discrete (binary or ordinal) variable.
Input variables have any measurement level.	Input variables have any measurement level.
Predicted values are the mean of the target variable at the given values of the input variables.	Predicted values are the probability of a particular level(s) of the target variable at the given values of the input variables.



Logistic Regression Equation:

The underlying algorithm of Maximum Likelihood Estimation (MLE) determines the regression coefficient for the model that accurately predicts the probability of the binary dependent variable. The algorithm stops when the convergence criterion is met or maximum number of iterations are reached. Since the probability of any event lies between 0 and 1 (or 0% to 100%), when we plot the probability of dependent variable by independent factors, it will demonstrate an 'S' shape curve.

Logit Transformation is defined as follows-

Logit = Log (p/1-p) = log (probability of event happening/ probability of event not happening) = log (Odds)

Logistic Regression is part of a larger class of algorithms known as Generalized Linear Model (GLM). The fundamental equation of generalized linear model is:

$g(E(y)) = \alpha + \beta x1 + \gamma x2$

Here, g() is the link function, E(y) is the expectation of target variable and α

$+ \beta x1 + \gamma x2$ is the linear predictor (α, β, γ to be predicted). The role of link function is to 'link' the expectation of y to linear predictor.

Key Points :

1. GLM does not assume a linear relationship between dependent and independent variables. However, it assumes a linear relationship between link function and independent variables in logit model.
2. The dependent variable need not to be normally distributed.
3. It does not uses OLS (Ordinary Least Square) for parameter estimation. Instead, it uses maximum likelihood estimation (MLE).
4. Errors need to be independent but not normally distributed.

To understand, consider the following example:

We are provided a sample of 1000 customers. We need to predict the probability whether a customer will buy (y) a particular magazine or not. As we've a categorical outcome variable, we'll use logistic regression.

To start with logistic regression, first write the simple linear regression equation with dependent variable enclosed in a link function:

$g(y) = \beta o + \beta(Age)$ —— (a)

For understanding, consider '*Age*' as independent variable.

In logistic regression, we are only concerned about the probability of outcome dependent variable (success or failure). As described above, g() is the link function. This function is established using two things: Probability of Success(p) and Probability of Failure(1-p). p should meet following criteria:

1. It must always be positive (since p >= 0)
2. It must always be less than equals to 1 (since p <= 1)

Now, simply satisfy these 2 conditions and get to the core of logistic regression. To establish link function, we denote g() with 'p' initially and eventually end up deriving this function.

Since probability must always be positive, we'll put the linear equation in exponential form. For any value of slope and dependent variable, exponent of this equation will never be negative.

$$p = \exp(\beta o + \beta(Age)) = e^{(\beta o + \beta(Age))} \text{ ———— (b)}$$

To make the probability less than 1, divide p by a number greater than p. This can simply be done by:

$$p = \exp(\beta o + \beta(Age)) / \exp(\beta o + \beta(Age)) + 1 = e^{(\beta o + \beta(Age))} / e^{(\beta o + \beta(Age))} + 1 \text{ ———— ©}$$

Using (a), (b) and ©, we can redefine the probability as:

$$p = e^{y} / 1 + e^{y} \text{ —— (d)}$$

where p is the probability of success. *This (d) is the Logit Function*

If p is the probability of success, 1-p will be the probability of failure which can be written as:

q = 1 — p = 1 — (e^y/ 1 + e^y) — — (e)

where q is the probability of failure

On dividing, (d) / (e), we get,

p/(1-p) = e^y

After taking *log* on both side, we get,

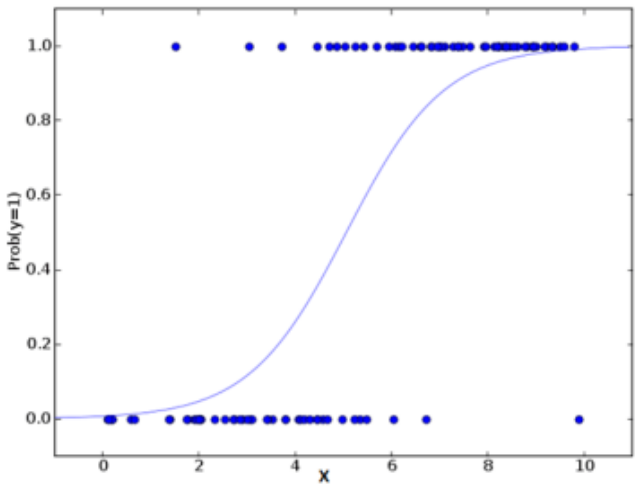
log(p/(1-p)) = y

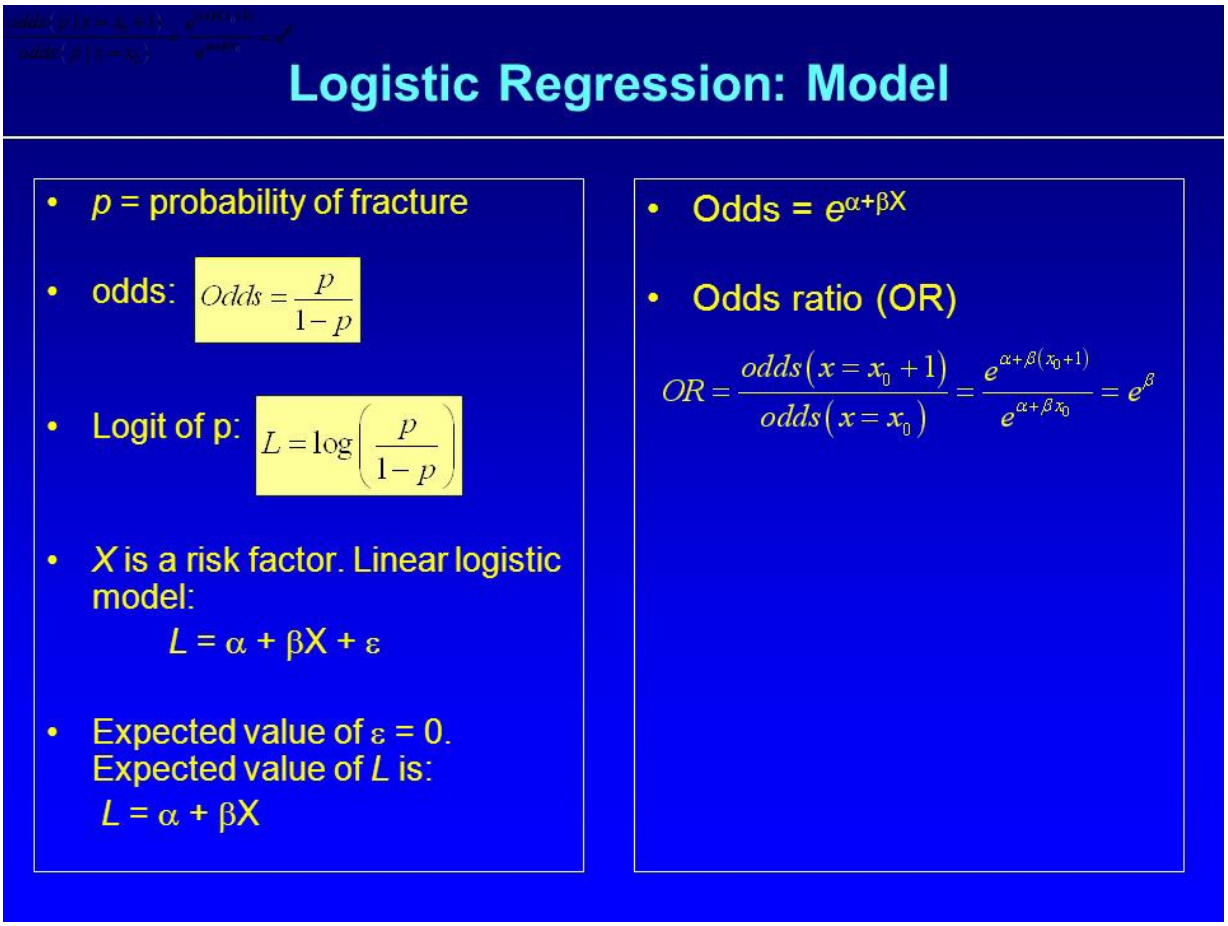
log(p/1-p) is the link function. Logarithmic transformation on the outcome variable allows us to model a non-linear association in a linear way.

After substituting value of y, we'll get:

log(p/(1-p)) = βo + β(Age)

This is the equation used in Logistic Regression. Here (p/1-p) is the odd ratio. Whenever the log of odd ratio is found to be positive, the probability of of success is always more than 50%. A typical logistic model plot is shown below. It shows probability never goes below 0 and above 1.

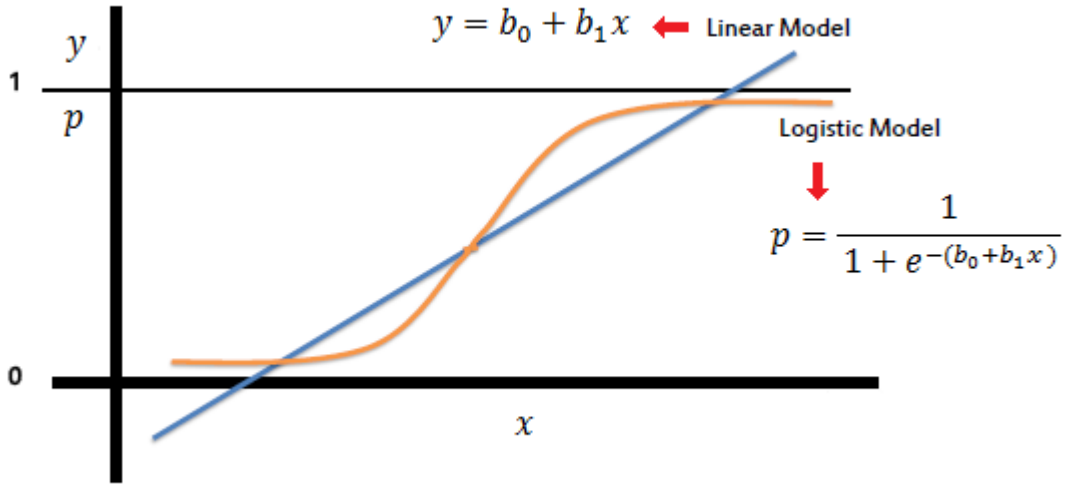



Logistic Regression: Model
• p = probability of fracture
• odds: $Odds = \frac{p}{1-p}$
• Logit of p: $L = \log\left(\frac{p}{1-p}\right)$
• X is a risk factor. Linear logistic model:
L = α + βX + ε
• Expected value of ε = 0. Expected value of L is:
L = α + βX
• Odds = $e^{\alpha+\beta X}$
• Odds ratio (OR)
$OR = \frac{odds(x = x_0 + 1)}{odds(x = x_0)} = \frac{e^{\alpha+\beta(x_0+1)}}{e^{\alpha+\beta x_0}} = e^{\beta}$

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g. predicting probabilities
  outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the "odds" of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.



In the logistic regression the constant ($b0$) moves the curve left and right and the slope ($b1$) defines the steepness of the curve. Logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses **maximum likelihood estimation (MLE)** to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until **LL** (Log Likelihood) does not change significantly.

Performance of Logistic Regression Model (Performance Metrics):

To evaluate the performance of a logistic regression model, we must consider few metrics. Irrespective of tool (SAS, R, Python) we would work on, always look for:

1. **AIC (Akaike Information Criteria)** — The analogous metric of adjusted R^2 in logistic regression is AIC. AIC is the measure of fit which penalizes model for the number of model coefficients. Therefore, we always prefer model with minimum AIC value.

2. **Null Deviance and Residual Deviance** — Null Deviance indicates the response predicted by a model with nothing but an intercept. Lower the value, better the model. Residual deviance indicates the response predicted by a model on adding independent variables. Lower the value, better the model.

3. **Confusion Matrix:** It is nothing but a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model and avoid over-fitting. This is how it looks like:

Actual		Predicted	
		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Predictive Model: Evaluation

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

predictive result / classification		actual result / classification: yes	actual result / classification: no
	yes	tp (true positive)	fp (false positive) ← Type 1 error
	no	fn (false negative)	tn (true negative)

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad \text{True Negative Rate} = \frac{tn}{tn + fp}$$

Confusion Matrix and ROC Curve

		Predicted Class: No	Predicted Class: Yes
Observed Class	No	TN	FP
	Yes	FN	TP

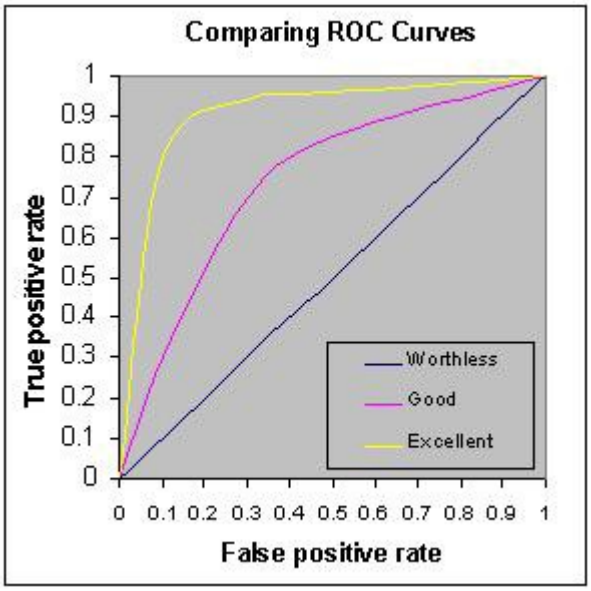
TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	= (TN+TP)/(TN+FP+FN+TP)
Precision	= TP/(FP+TP)
Sensitivity	= TP/(TP+FN)
Specificity	= TN/(TN+FP)

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

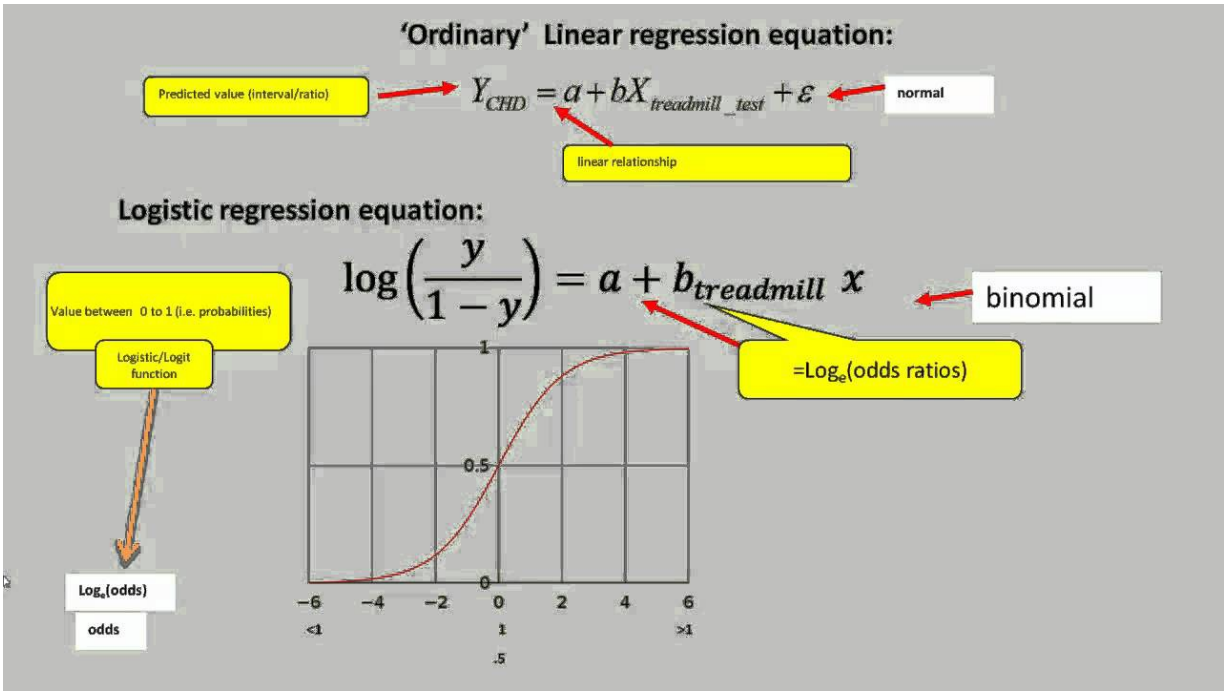
4. **ROC Curve:** Receiver Operating Characteristic (ROC) summarizes the model's performance by evaluating the trade-offs between true positive rate (sensitivity) and false positive rate (1- specificity). For plotting ROC, it is advisable to assume p > 0.5 since we are more concerned about success rate. ROC summarizes the predictive power for all possible values of p > 0.5. The area under curve (AUC), referred to as index of accuracy (A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model. Below is a sample ROC curve. The ROC of a perfect predictive model has TP equals 1 and FP equals 0. This curve will touch the top left corner of the graph.

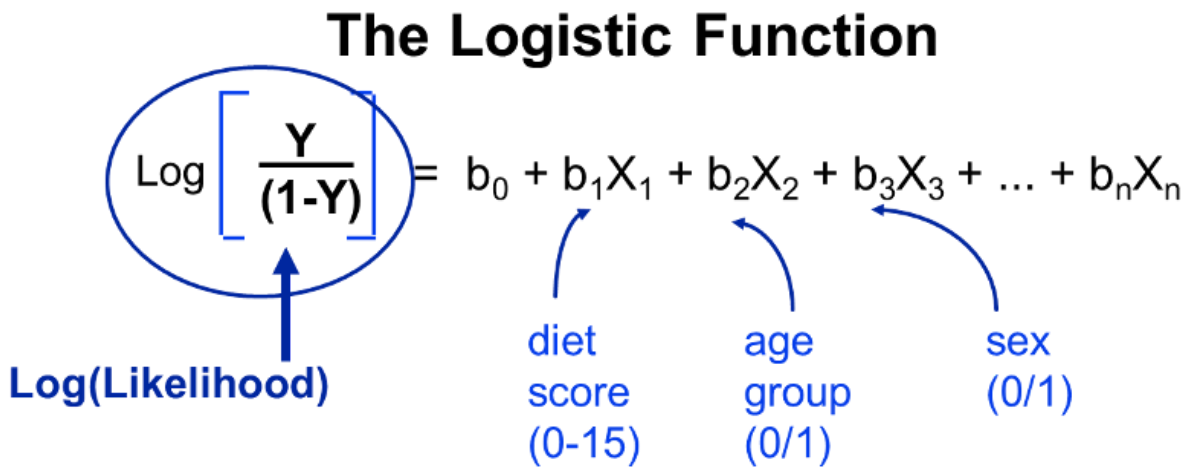


For model performance, we can also consider likelihood function. It is called so, because it selects the coefficient values which maximizes the likelihood of explaining the observed data. It indicates goodness of fit as its value approaches one, and a poor fit of the data as its value approaches zero.

Summary :

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. We can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a **logit** function.





· It is widely used for **classification problems**

· Logistic regression doesn't require linear relationship between dependent and independent variables. It can handle various types of relationships because it applies a non-linear log transformation to the predicted odds ratio

· To avoid over fitting and under fitting, we should include all significant variables. A good approach to ensure this practice is to use a step wise method to estimate the logistic regression

· It requires **large sample sizes** because maximum likelihood estimates are less powerful at low sample sizes than ordinary least square

· The independent variables should not be correlated with each other i.e. **no multi collinearity**. However, we have the options to include interaction effects of categorical variables in the analysis and in the model.

· If the values of dependent variable is ordinal, then it is called as **Ordinal logistic regression**

· If dependent variable is multi class then it is known as **Multinomial Logistic regression.**

What is Logistic Regression?

Like many other machine learning techniques, it is borrowed from the field of statistics and despite its name, it is not an algorithm for regression problems, where you want to predict a continuous outcome. Instead, Logistic Regression is the go-to method for binary classification. It gives you a discrete binary outcome between 0 and 1. To say it in simpler words, it's outcome is either one thing or another.

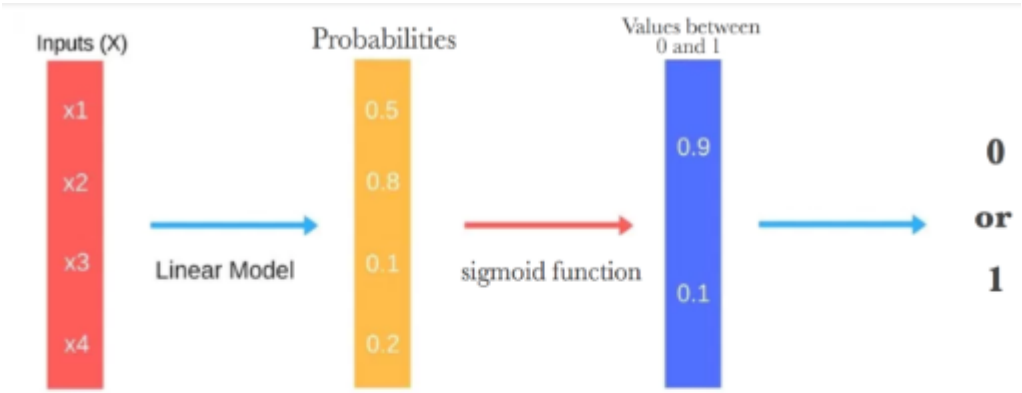
A simple example of a Logistic Regression problem would be an algorithm used for cancer detection that takes screening picture as an input and should tell if a patient has cancer (1) or not (0).

How it works

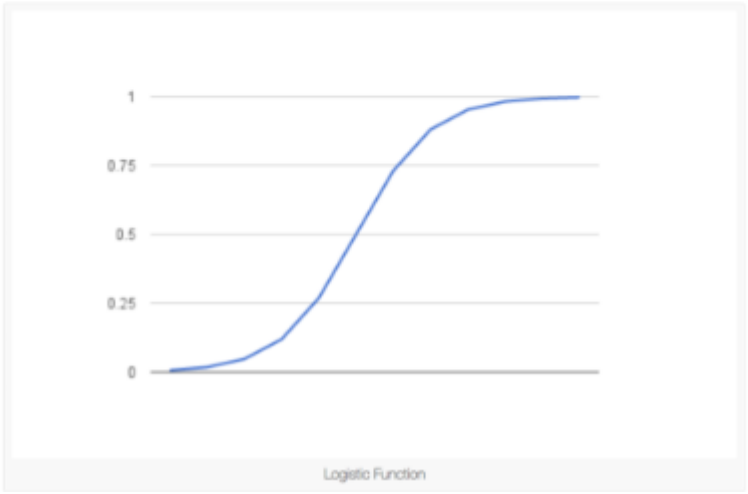
Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.

These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. The Sigmoid-Function is an S-shaped curve that can take any real-valued number and map it into a value between the range of 0 and 1, but never exactly at those limits. This values between 0 and 1 will then be transformed into either 0 or 1 using a threshold classifier.

The picture below illustrates the steps that logistic regression goes through to give you your desired output.



low you can see how the logistic function (sigmoid function) looks like:



We want to maximize the likelihood that a random data point gets classified correctly, which is called Maximum Likelihood Estimation. Maximum Likelihood Estimation is a general approach to estimating parameters in statistical models. You can maximize the likelihood using different methods like an optimization algorithm. Newton's Method is such an algorithm and can be used to find maximum (or minimum) of many different functions, including the likelihood function. Instead of Newton's Method, you could also use Gradient Descent.

Logistic VS. Linear Regression

You may be asking yourself what the difference between logistic and linear regression is. Logistic regression gives you a discrete outcome but linear regression gives a continuous outcome. A good example of a continuous outcome would be a model that predicts the value of a house. That value will always be different based on parameters like it's size or location. A discrete outcome will always be one thing (you have cancer) or another (you have no cancer).

Advantages / Disadvantages

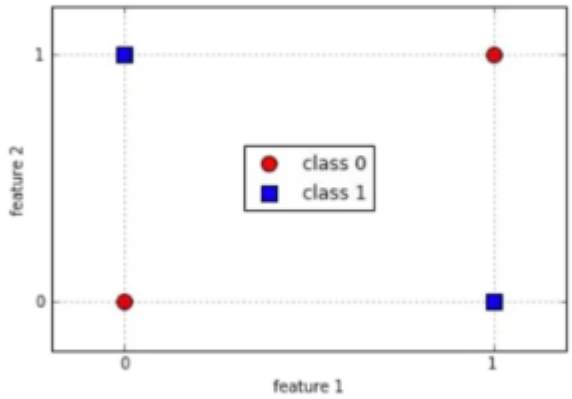
It is a widely used technique because it is very efficient, does not require too many computational resources, it's highly interpretable, it doesn't require input features to be scaled, it doesn't require any tuning, it's easy to regularize, and it outputs well-calibrated predicted probabilities.

Like linear regression, logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. Therefore Feature Engineering plays an important role in regards to the performance of Logistic and also Linear Regression. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train. I typically start with a Logistic Regression model as a benchmark and try using more complex algorithms from there on.

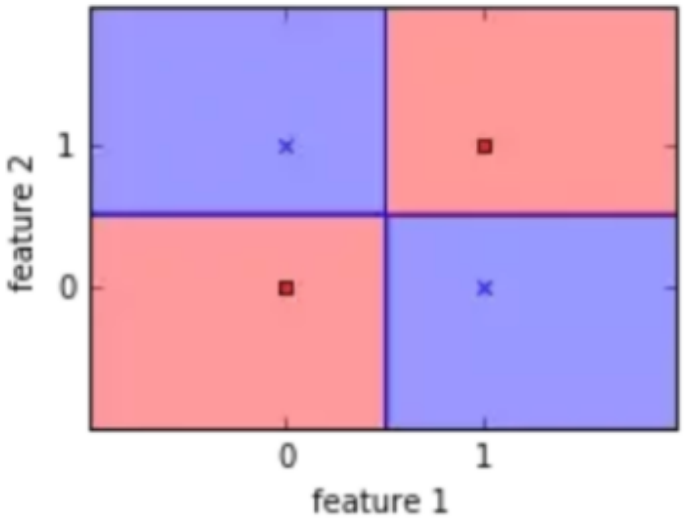
Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is also a good baseline that you can use to measure the performance of other more

complex Algorithms.

A disadvantage of it is that we can't solve non-linear problems with logistic regression since it's decision surface is linear. Just take a look at the example below that has 2 binary features from 2 examples.



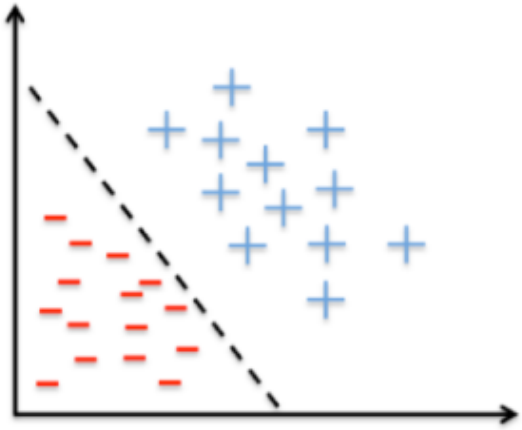
It is clearly visible that we can't draw a line that separates these 2 classes without a huge error. To use a simple decision tree would be a much better choice.



Logistic Regression is also not one of the most powerful algorithms out there and can be easily outperformed by more complex ones. Another disadvantage is its high reliance on a proper presentation of your data. This means that logistic regression is not a useful tool unless you have already identified all the important independent variables. Since its outcome is discrete, Logistic Regression can only predict a categorical outcome. It is also an Algorithm that is known for its vulnerability to overfitting.

When to use it

Like I already mentioned, Logistic Regression separates your input into two „regions" by a linear boundary, one for each class. Therefore it is required that your data is linearly separable, like the data points in the image below:



In other words: You should think about using logistic regression when your Y variable takes on only two values (e.g when you are facing a classification problem). Note that you could also use Logistic Regression for multiclass classification, which will be discussed in the next section.

Multiclass Classification

Out there are algorithms that can deal by themselves with predicting multiple classes, like Random Forest classifiers or the Naive Bayes Classifier. There are also algorithms that can't do that, like Logistic Regression, but with some tricks, you can predict multiple classes with it too.

Let's discuss the most common of these "tricks" at the example of the MNIST Dataset, which contains handwritten images of digits, ranging from 0 to 9. This is a classification task where our Algorithm should tell us which number is on an image.

1) one-versus-all (OvA)

With this strategy, you train 10 binary classifiers, one for each number. This simply means training one classifier to detect 0s, one to detect 1s, one to detect 2s and so on. When you then want to classify an image, you just look at which classifier has the best decision score

2) one-versus-one (OvO)

Here you train a binary classifier for every pair of digits. This means training a classifier that can distinguish between 0s and 1s, one that can distinguish between 0s and 2s, one that can distinguish between 1s and 2s etc. If there are N classes, you would need to train NxN(N-1)/2 classifiers, which are 45 in the case of the MNIST dataset.

When you then want to classify images, you need to run each of these 45 classifiers and choose the best performing one. This strategy has one big advantage over the others and this is, that you only need to train it on a part of the training set for the 2 classes it distinguishes between.

Algorithms like Support Vector Machine Classifiers don't scale well at large datasets, which is why in this case using a binary classification algorithm like Logistic Regression with the OvO strategy would do better, because it is faster to train a lot of classifiers on a small dataset than training just one at a large dataset.

At most algorithms, sklearn recognizes when you use a binary classifier for a multiclass classification task and automatically uses the OvA strategy. There is an exception: When you try to use a Support Vector Machine classifier, it automatically runs the OvO strategy.

Other Classification Algorithms

Other common classification algorithms are Naive Bayes, Decision Trees, Random Forests, Support Vector Machines, k-nearest neighbor and many others. We will also discuss them in future blog posts but don't feel overwhelmed by the amount of Machine Learning algorithms that are out there. Note that it is better to know 4 or 5 algorithms really well and to concentrate your energy at feature-engineering, but this is also a topic for future posts.

Summary

In this post, you have learned what Logistic Regression is and how it works. You now have a solid understanding of its advantages and disadvantages and know when you can use it. Also, you have discovered ways to use Logistic Regression to do multiclass classification with sklearn and why it is a good baseline to compare other Machine Learning algorithms with.

In [ ]:
```
https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-er

    https://www.analyticsvidhya.com/blog/2017/03/imbalanced-data-classificati
```