**“rough notes - just for reference”**

**1. Introduction to Logistic Regression**

When data scientists may come across a new classification problem, the first algorithm that may come across their mind is **Logistic Regression**. It is a supervised learning classification algorithm which is used to predict observations to a discrete set of classes. Practically, it is used to classify observations into different categories. Hence, its output is discrete in nature. **Logistic Regression** is also called **Logit Regression**. It is one of the most simple, straightforward and versatile classification algorithms which is used to solve classification problems.

**2. Logistic Regression intuition**

In statistics, the **Logistic Regression model** is a widely used statistical model which is primarily used for classification purposes. It means that given a set of observations, Logistic Regression algorithm helps us to classify these observations into two or more discrete classes. So, the target variable is discrete in nature.

The Logistic Regression algorithm works as follows -

**Implement linear equation**

Logistic Regression algorithm works by implementing a linear equation with independent or explanatory variables to predict a response value. For example, we consider the example of number of hours studied and probability of passing the exam. Here, number of hours studied is the explanatory variable and it is denoted by x1. Probability of passing the exam is the response or target variable and it is denoted by z.

If we have one explanatory variable (x1) and one response variable (z), then the linear equation would be given mathematically with the following equation-

y = β0 + β1x1

Here, the coefficients β0 and β1 are the parameters of the model.

If there are multiple explanatory variables, then the above equation can be extended to

z = β0 + β1x1+ β2x2+……..+ βnxn

Here, the coefficients β0, β1, β2 and βn are the parameters of the model.

So, the predicted response value is given by the above equations and is denoted by z.

**Sigmoid Function**

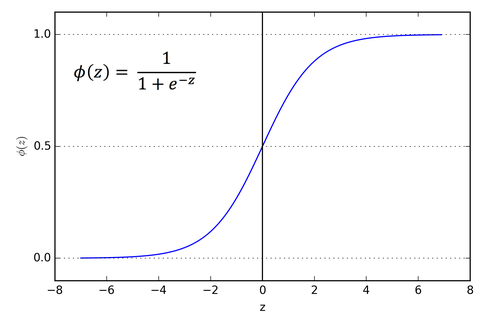
This predicted response value, denoted by z is then converted into a probability value that lie between 0 and 1. We use the sigmoid function in order to map predicted values to probability values. This sigmoid function then maps any real value into a probability value between 0 and 1.

In machine learning, sigmoid function is used to map predictions to probabilities. The sigmoid function has an S shaped curve. It is also called sigmoid curve.

A Sigmoid function is a special case of the Logistic function. It is given by the following mathematical formula.

Graphically, we can represent sigmoid function with the following graph.

Sigmoid Function



**Decision boundary**

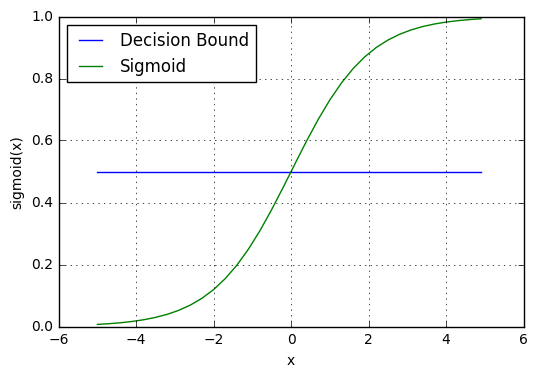
The sigmoid function returns a probability value between 0 and 1. This probability value is then mapped to a discrete class which is either “0” or “1”. In order to map this probability value to a discrete class (pass/fail, yes/no, true/false), we select a threshold value. This threshold value is called Decision boundary. Above this threshold value, we will map the probability values into class 1 and below which we will map values into class 0.

Mathematically, it can be expressed as follows:-

p ≥ 0.5 => class = 1

p < 0.5 => class = 0

Generally, the decision boundary is set to 0.5. So, if the probability value is 0.8 (> 0.5), we will map this observation to class 1. Similarly, if the probability value is 0.2 (< 0.5), we will map this observation to class 0. This is represented in the graph below-



**Making predictions**

Now, we know about sigmoid function and decision boundary in logistic regression. We can use our knowledge of sigmoid function and decision boundary to write a prediction function. A prediction function in logistic regression returns the probability of the observation being positive, Yes or True. We call this as class 1 and it is denoted by P(class = 1). If the probability inches closer to one, then we will be more confident about our model that the observation is in class 1, otherwise it is in class 0.

**3. Assumptions of Logistic Regression**

The Logistic Regression model requires several key assumptions. These are as follows:-

1. Logistic Regression model requires the dependent variable to be **binary**(0/1/2+), multinomial or ordinal in nature.
2. It requires the observations to be independent of each other. So, the observations should not come from repeated measurements.
3. Logistic Regression algorithm requires little or no multicollinearity among the independent variables. It means that the independent variables should not be too highly correlated with each other.
4. Logistic Regression model assumes linearity of independent variables.
5. The success of Logistic Regression model depends on the sample sizes. Typically, it requires a large sample size to achieve the high accuracy.

**4. Types of Logistic Regression**

Logistic Regression model can be classified into three groups based on the target variable categories. These three groups are described below:-

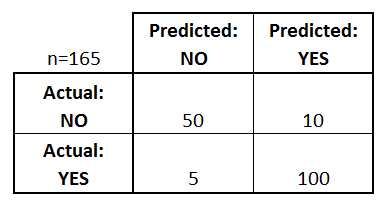
* **Binary Logistic Regression -** In Binary Logistic Regression, the target variable has two possible categories. The common examples of categories are yes or no, good or bad, true or false, spam or no spam and pass or fail.
* **Multinomial Logistic Regression -** In Multinomial Logistic Regression, the target variable has three or more categories which are not in any particular order. So, there are three or more **nominal** categories. The examples include the type of categories of fruits - apple, mango, orange and banana.0/1/2+
* **Ordinal Logistic Regression -** In Ordinal Logistic Regression, the target variable has three or more ordinal categories. So, there is intrinsic order involved with the categories. For example, the student performance can be categorized as poor, average, good and excellent.

—-----------------

Precision, Recall, Accuracy etc etc ✍ Confusion matrix

# Describing the Performance of a Logistic model

A **confusion matrix** is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. Let us look at some of the important terms of confusion matrix.



confusion matrix whether employees will leave a company or not

### The Confusion Matrix tells us the following:

* There are two possible predicted classes: “yes” and “no”. If we were predicting that employees would leave an organisation, for example, “yes” would mean they will, and “no” would mean they won’t leave the organisation.
* The classifier made a total of 165 predictions (e.g., 165 employees were being studied).
* Out of those 165 cases, the classifier predicted “yes” 110 times, and “no” 55 times.
* In reality, 105 employees in the sample leave the organisation, and 60 do not.

### Basic terms related to Confusion matrix:

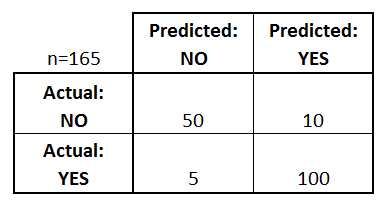
### 

* **True positives (TP)**: These are cases in which we predicted yes (employees will leave the organisation), and employees actually leave i.e 100
* **True negatives (TN)**: We predicted no(employees will not leave the organisation) and they don’t leave i.e 50
* **False positives (FP)**: We predicted yes they will leave, but they don’t leave. (Also known as a “**Type I error**.”) i.e 10
* **False negatives (FN)**: We predicted no they will not leave, but they actually leave (Also known as a “**Type II error**.”) i.e 5

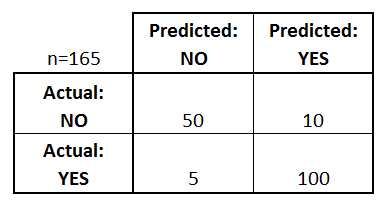
### Evaluating a Classification Model

* **Accuracy** : (TP+TN)/Total(FP+FN+TP+TN) . Describes overall, how often the classifier correct. i.e 100+50/165 Measures of Accuracy

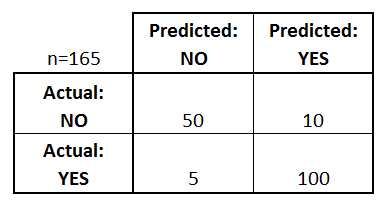
Sensitivity and specificity are statistical measures of the performance of a binary classification test:



* **Sensitivity/Recall** = TP/(TP + FN). When it’s actually yes, how often does it predict yes? i.e 100/(100+5)



* **Sensitivity/Recall** = TP/(TP + FN). When it’s actually yes, how often does it predict yes? i.e 100/(100+5)
* **Precision** = TP/(TP+FP). When it predicts yes, how often is it correct?100/(10+100)



**Specificity** = TN/(TN + FP).When it’s actually no, how often does it predict no?? i.e 50/(50+10)

* [**F1 Score**](https://en.wikipedia.org/wiki/F1_score) = 2\*((precision\*recall)/(precision+recall)).

It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.