The Task:

The purpose of this analysis is to predict the survival of passengers of the Titanic. So, to solve this problem, the nature of the relationship between of each variable with Survived must be understood along with the individual characteristic of each variable. A statistical model is then adopted to further the analysis and arrive at the results and interpretation.

The Dataset:

The dataset contains the following variables:

**PassengerId**

**Survived**

**Pclass**

**Name**

**Sex**

**Age**

**SibSp**

**Parch**

**Ticket**

**Income**

**Fare**

**Cabin**

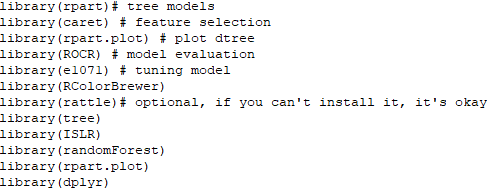
**Embarked**

The statistical model:

Classification is the method of predicting the class of a given input data point. Classification problems are common in machine learning and they fall under the Supervised learning method.

 KNN which stand for K Nearest Neighbor is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points. For example, features such as pointy ears can be used to identify cats and similarly we can identify dogs based on their long ears. After studying the dataset during the training phase, when a new image is given to the model, the KNN algorithm will classify it into either cats or dogs depending on the similarity in their features. So if the new image has pointy ears, it will classify that image as a cat because it is similar to the cat images. In this manner, the KNN algorithm classifies data points based on how similar they are to their neighboring data points.

Setting up the R model by loading the required libraries:



Data:

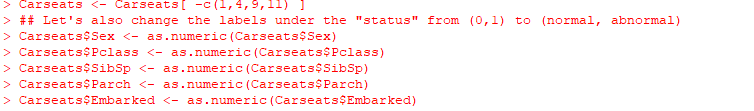
In the next step, the data is read into the R environment from the file.

setwd("C:\\Users\\ADMIN\\Desktop\\R Models\\Decision Tree")

Carseats <- read.csv("Titanic.csv")

Data selection and data type modification:

In the given dataset, the columns which are named as “PassengerId”, “Name”, “Ticket” and “Fare” contains customerid details, names of the passengers, ticket details and fare are of no relevance to our model and is thus excluded. Two of the variables, Pclass and Survived are converted to factors and Survived was in binary form so we converted 0 = “No” and 1 = “Yes”.



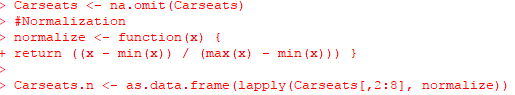
Checking for missing values:

After the data has been cleaned off all the outliers, it is then checked for any missing values in the following manner:

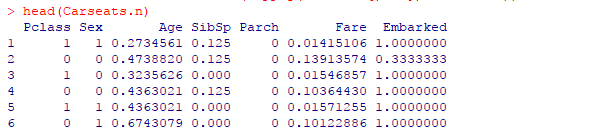
264 missing values were found and has been removed.

Splitting the data:

Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.

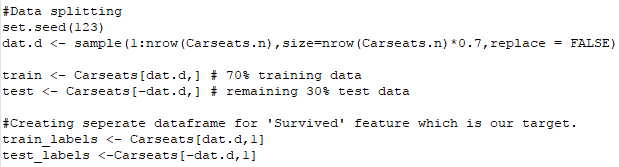


We must always normalize the data set so that the output remains unbiased.



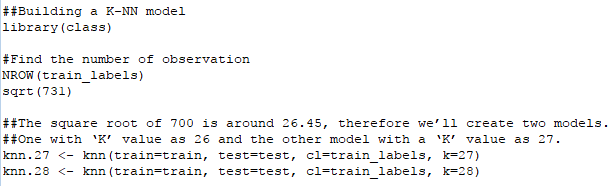
Splitting the data:

The data is then split into two parts. 70 percent of the data is split into ‘development’ and the remaining 30 percent is named ‘validation’. The development part is for training the decision tree model and the validation part is for testing the robustness of the model.



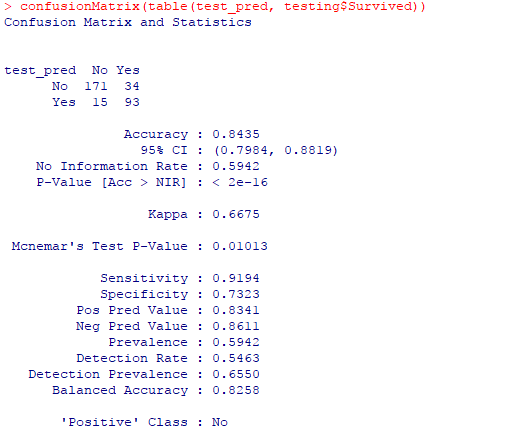
**Building a K-NN model:**

At this stage, we have to build a model by using the training data set. Since we’re using the KNN algorithm to build the model, we must first install the ‘class’ package provided by R. Next, we’re going to calculate the number of observations in the training data set. The reason we’re doing this is that we want to initialize the value of ‘K’ in the KNN model. One of the ways to find the optimal K value is to calculate the square root of the total number of observations in the data set. This square root will give you the ‘K’ value.



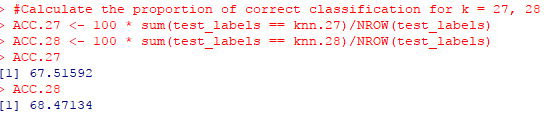
So, we have 731 observations in our training data set. The square root of 731 is around 27.03, therefore we’ll create two models. One with ‘K’ value as 27 and the other model with a ‘K’ value as 28.

A validation of training set has been ran using confusion matrix.



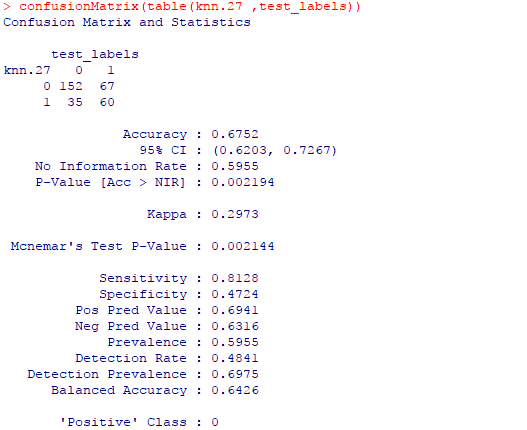
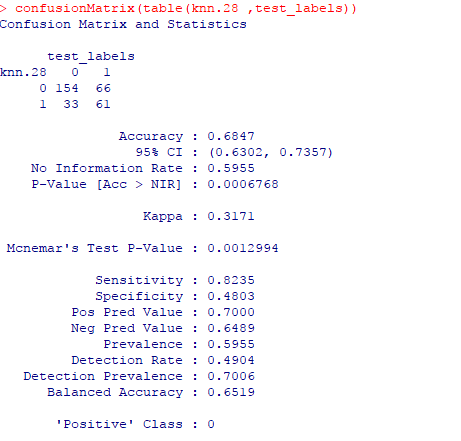
**Model Evaluation**:

After building the model, it is time to calculate the accuracy of the created models.



As shown above, the accuracy for K = 26 is 67.66 and for K = 27 it is 67.33.

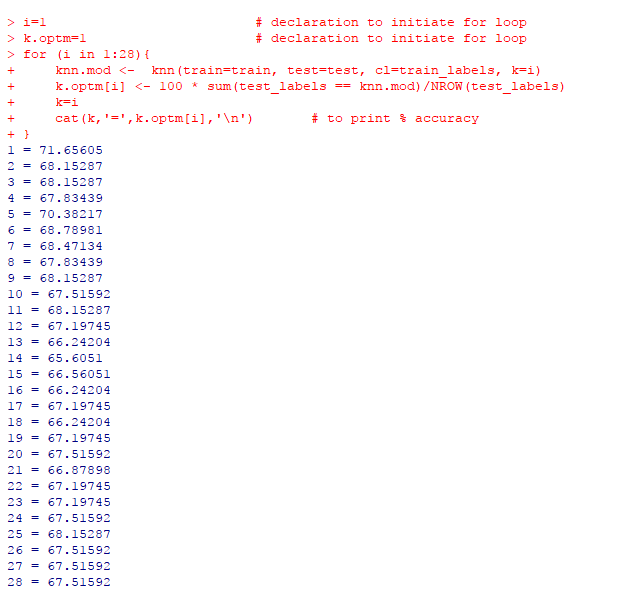
**Validation of K-NN with k = 27 & k = 28:**

**** ****

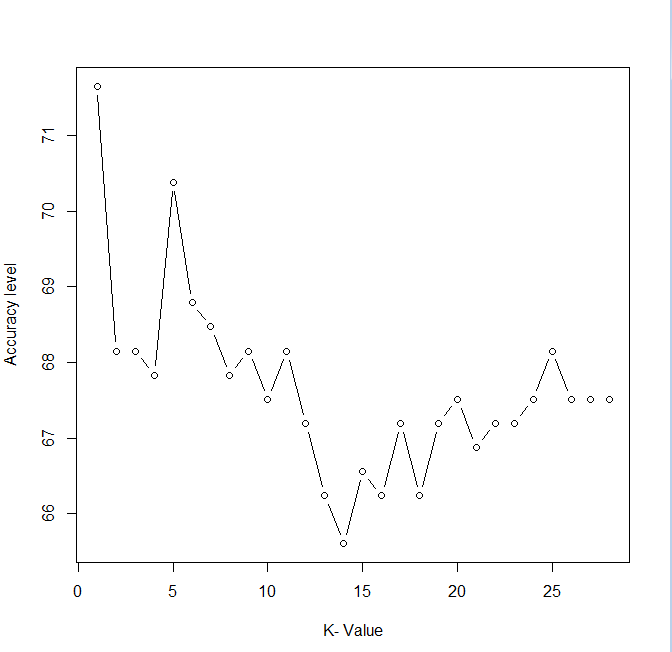
So, from the output, we can see that our model predicts the outcome with an accuracy of 67.52% & 68.47% for k = 27 and k = 28 respectively, which is good since we worked with a small data set. A point to remember is that the more data (optimal data) we feed the machine, the more efficient the model will be.

**Optimizing & tuning of K-NN:**

In order to improve the accuracy of the model, you can use n number of techniques such as the Elbow method and maximum percentage accuracy graph.  In the below code snippet, I’ve created a loop that calculates the accuracy of the KNN model for ‘K’ values ranging from 1 to 28. This way you can check which ‘K’ value will result in the most accurate mode.



From the output you can see that for K = 1, we achieve the maximum accuracy, i.e. 71%. We can also represent this graphically, like so.

  
The above graph shows that for ‘K’ value of 25 we get the maximum accuracy.

**Validation of the final model:**

Checking the confusion matrix to understand how good the model is with a tuned K-NN.

