**Survival of Titanic Passengers – ML Classification problem with the accuracy of 84%**

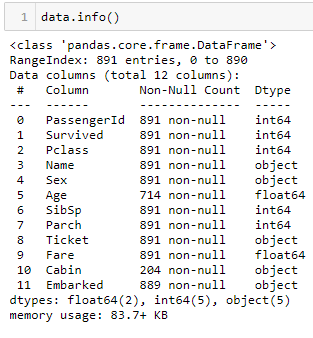
**Problem Definition:**

 The Titanic Problem is based on the sinking of the ‘Unsinkable’ ship Titanic in early 1912. It gives you information about multiple people like their ages, sexes, sibling counts, embarkment points, and whether or not they survived the disaster. Based on these features, you have to predict if an arbitrary passenger on Titanic would survive the sinking or not.

**Data Analysis:**

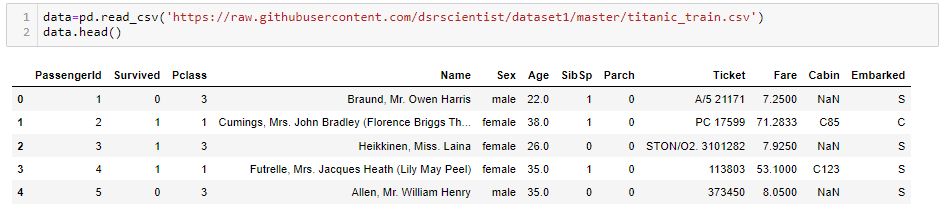


Data set contains 891 data points and 12 features.

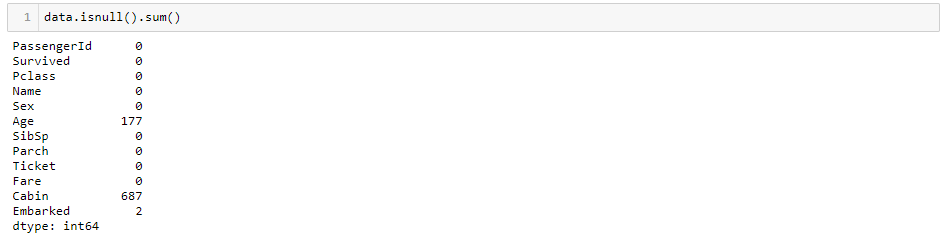


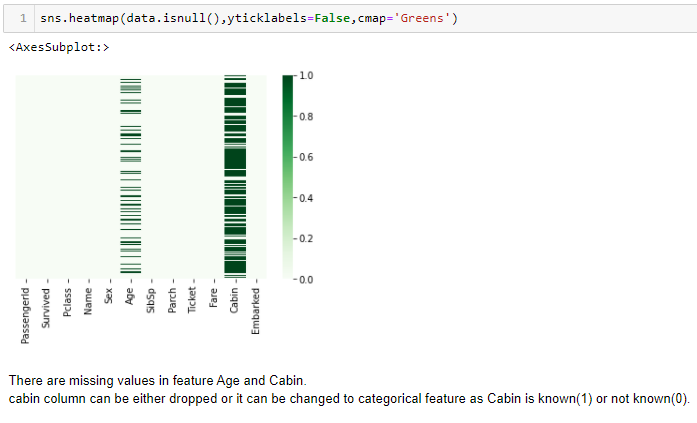
**The training-set has 891 examples and 11 features + the target variable (survived)**. 2 of the features are floats, 5 are integers and 5 are objects. Below I have listed the features with a short description:

survival: Survival   
PassengerId: Unique Id of a passenger.   
pclass: Ticket class   
sex: Sex   
Age: Age in years   
sibsp: # of siblings / spouses aboard the Titanic   
parch: # of parents / children aboard the Titanic   
ticket: Ticket number   
fare: Passenger fare   
cabin: Cabin number   
embarked: Port of Embarkation



Above we can see that **38% out of the training-set survived the Titanic**. We can also see that the passenger ages range from 0.4 to 80. On top of that we can already detect some features, that contain missing values as shown below.



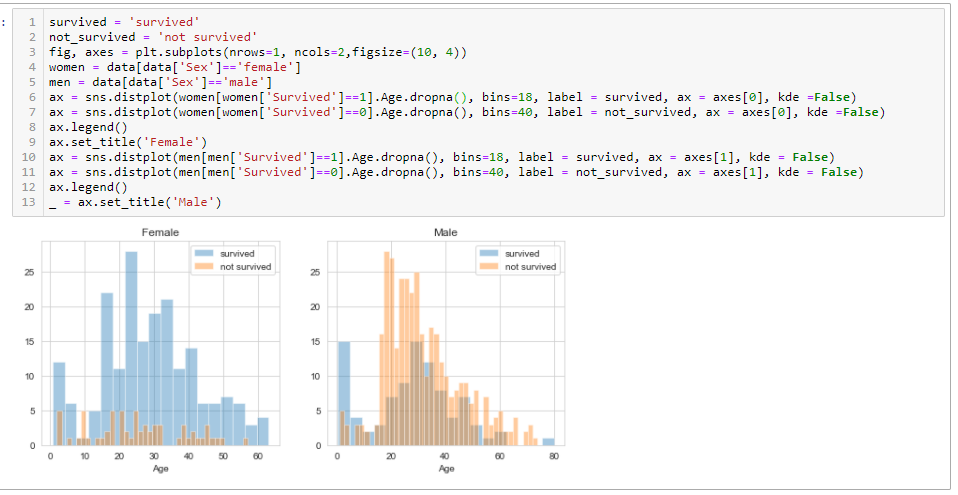


**Age and Survived:**

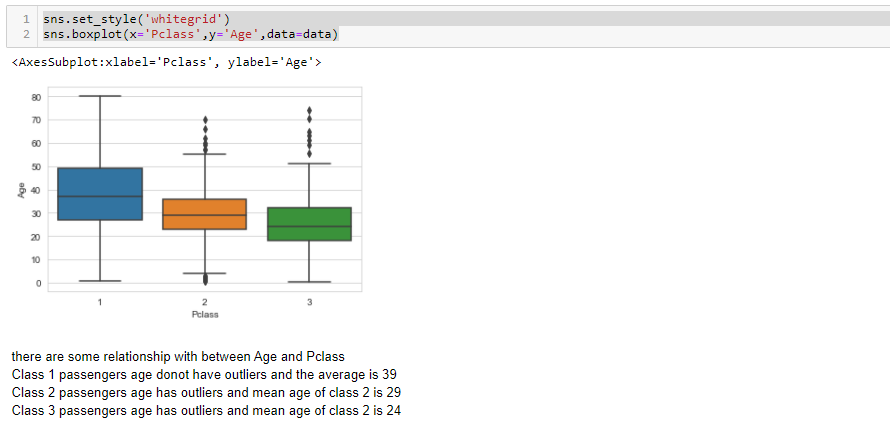
You can see that men have a high probability of survival when they are between 18 and 30 years old, which is also a little bit true for women but not fully. For women the survival chances are higher between 14 and 40.

For men the probability of survival is very low between the age of 5 and 18, but that isn’t true for women. Another thing to note is that infants also have a little bit higher probability of survival.

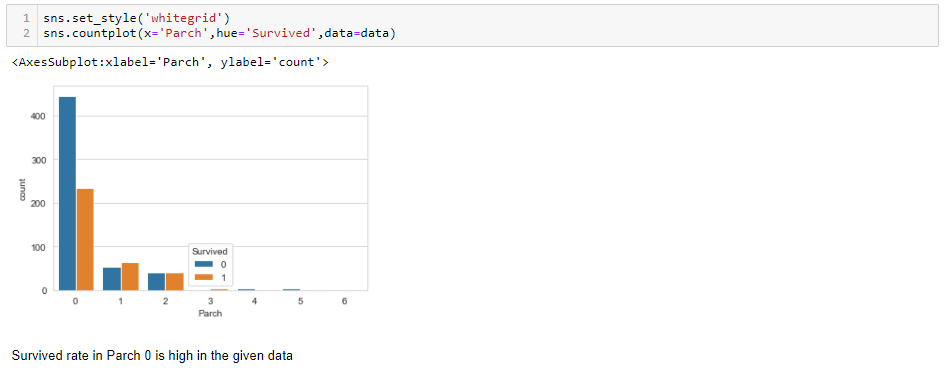
Since there seem to be **certain ages, which have increased odds of survival** and because I want every feature to be roughly on the same scale, I will create age groups later on.



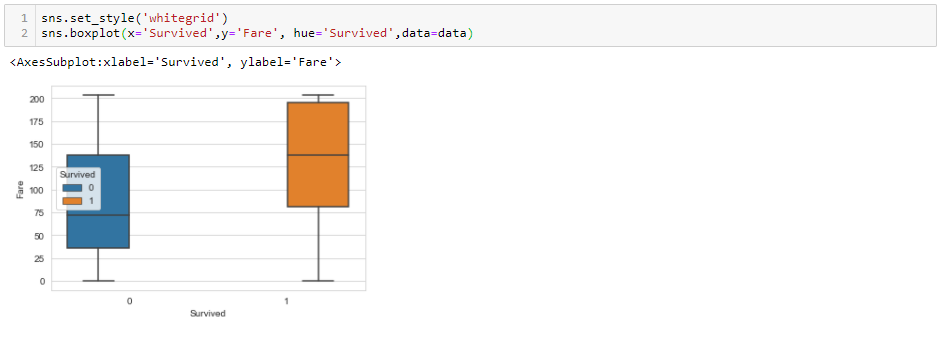
According to me there are some relationship between Pclass and the age as shown below.



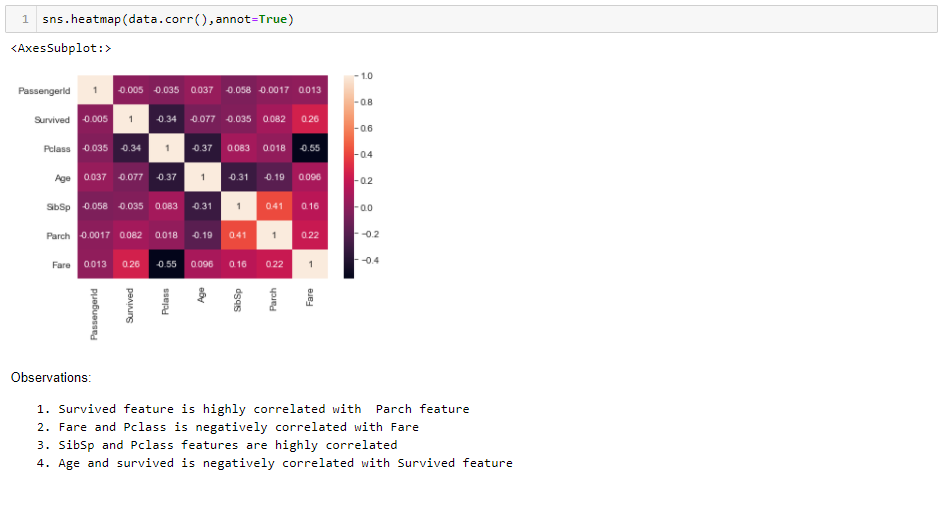
In the given data set parch 0 is having a high rate of survival



In the given data set, survival rate is low when the fare is high.

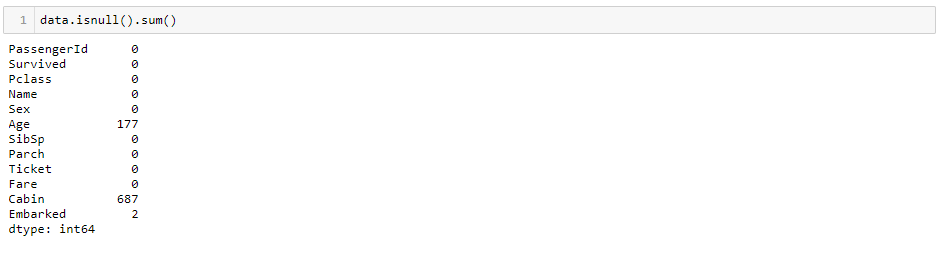


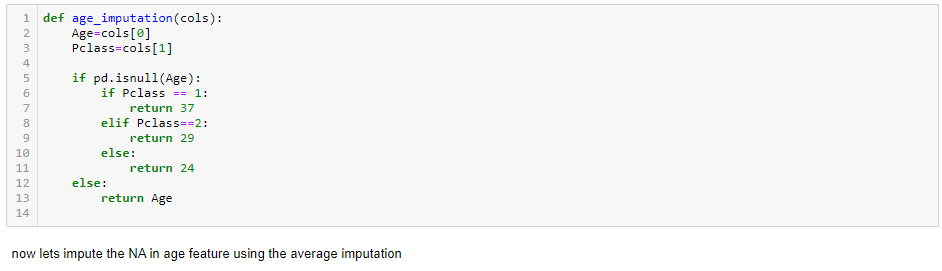
Below chart explains the correlation between features



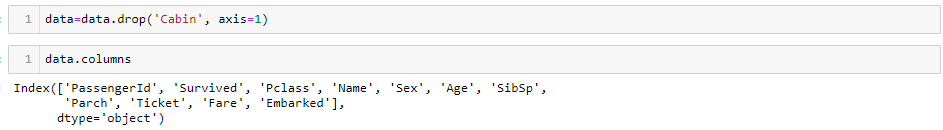
# Data Pre-processing:

Imputing Age feature to replace the null values with Pclass relationship as shown below.

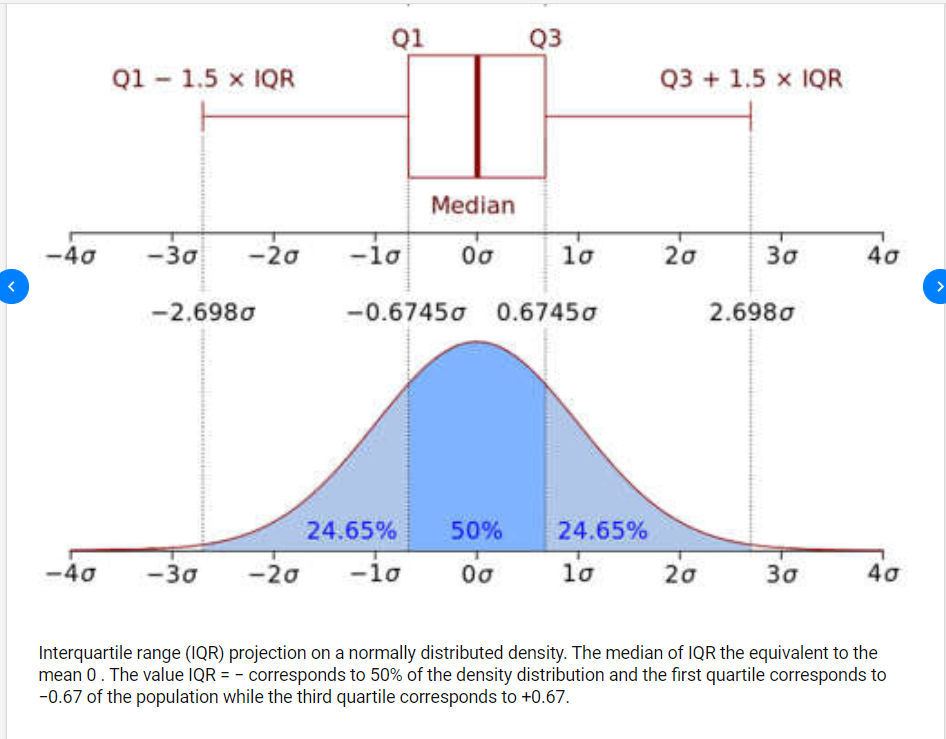


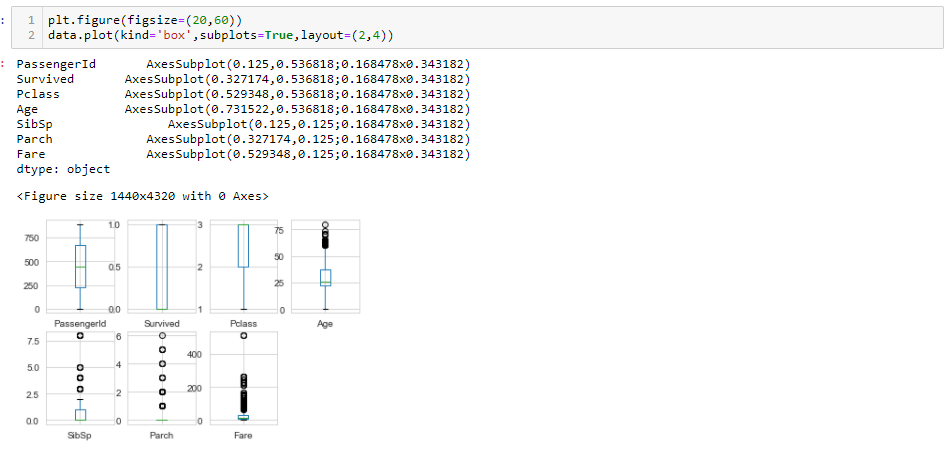


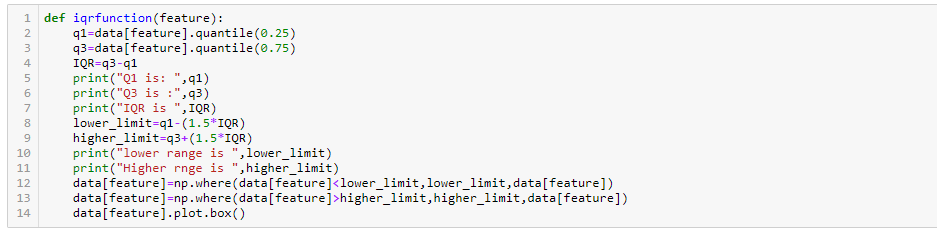
I had dropped Cabin feature as almost 78% of data points are containing null values.

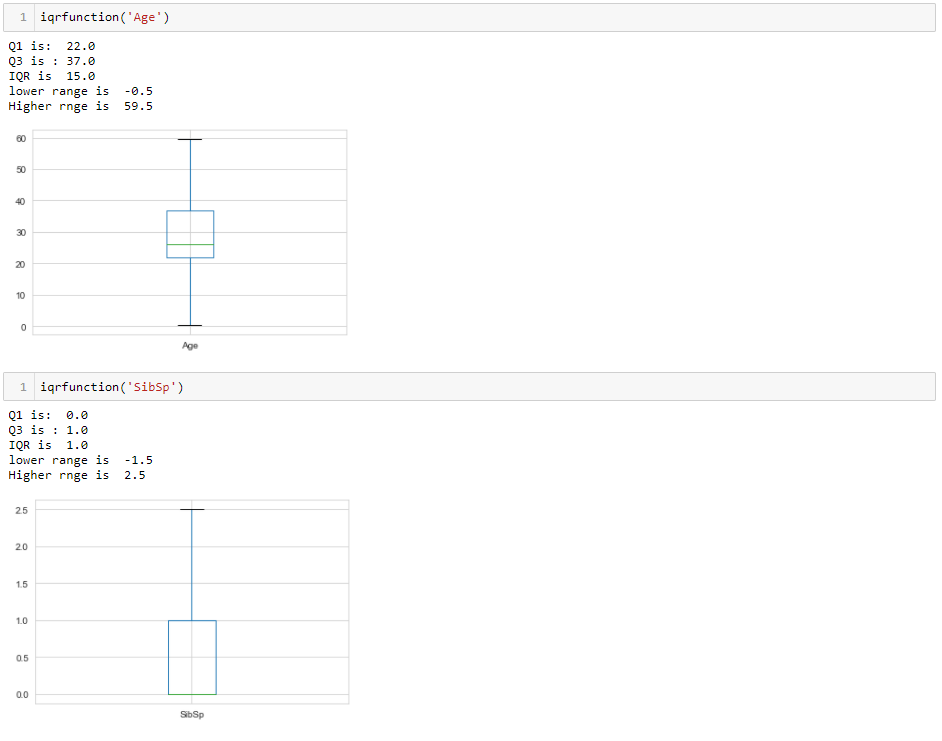


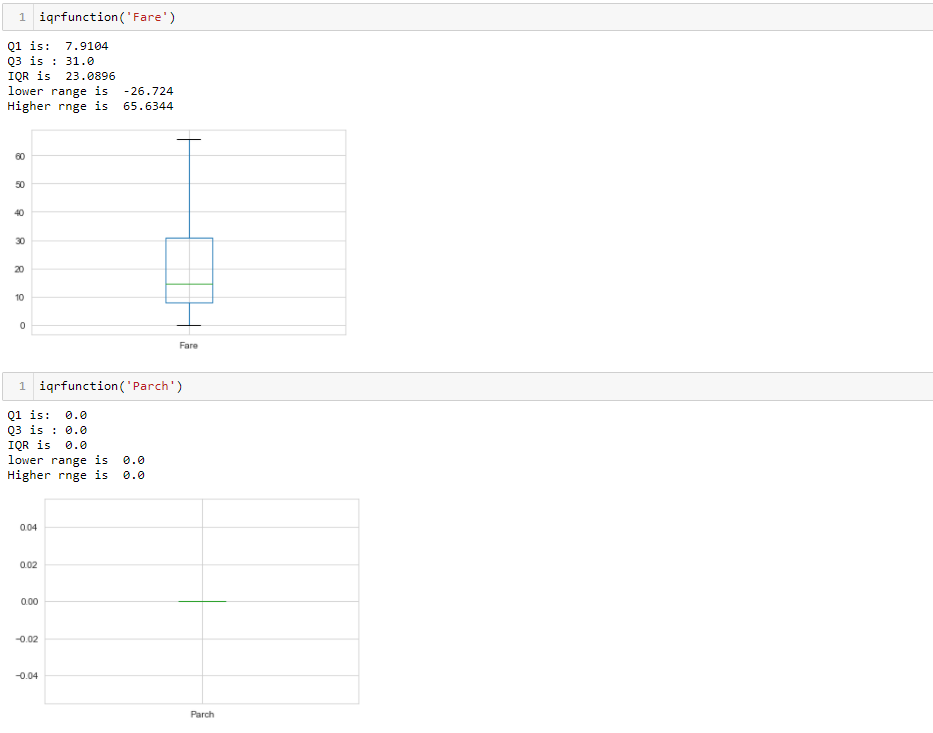
Fixing outliers using IQR technique.

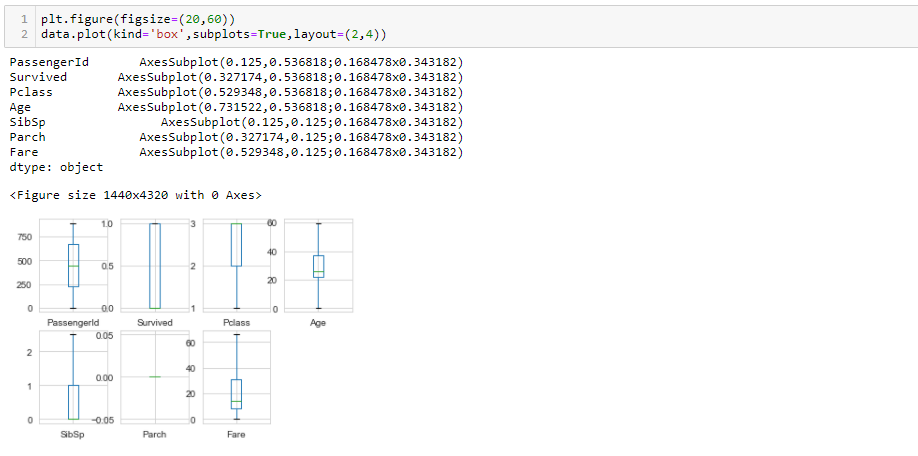


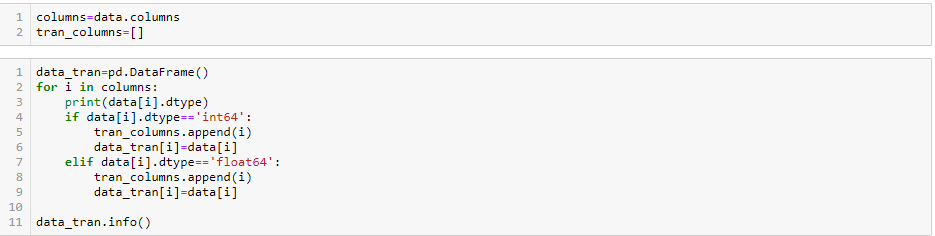






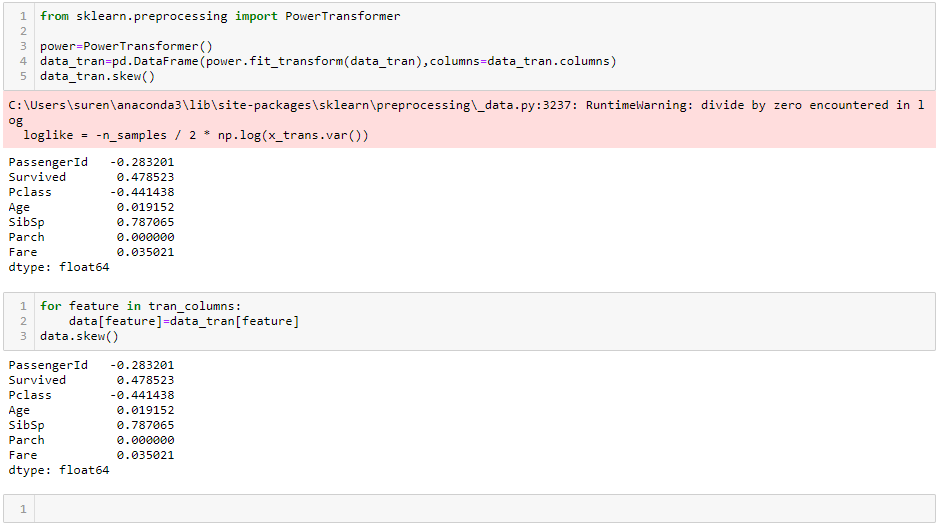






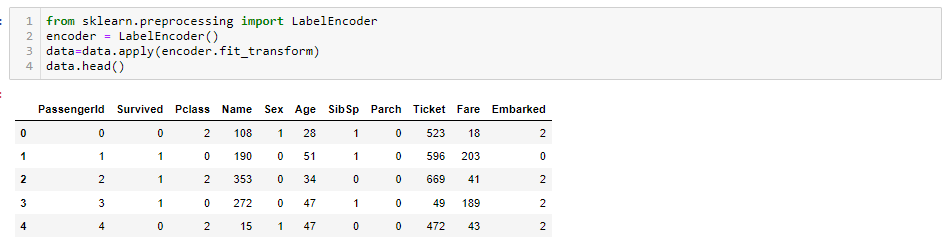
**Standardization:**

I had used power transformer transformation technique to remove the skewness in features to build a better performing model



**Encoding categorical features:**

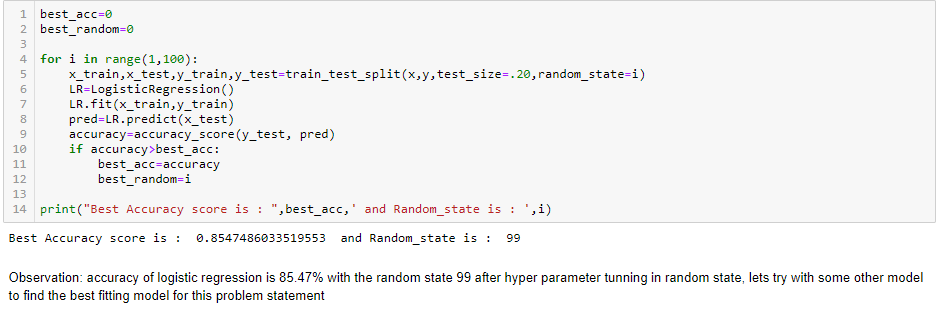
I had used Label encoder to encode the categorical Variables as shown below.





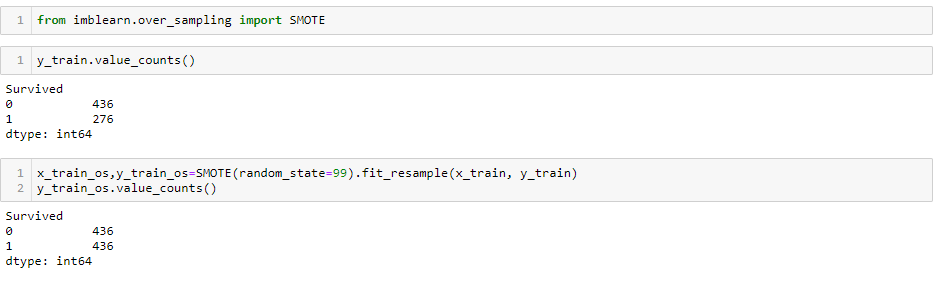
**Train test Split:**

I had tried to find the best random state using logistic regression as shown below. I got accuracy of 85% at random state 99.



**Balancing dataset:**

Given data set is imbalanced data set as shown below. I had used SMOTE over sampling technique to balance the data set.



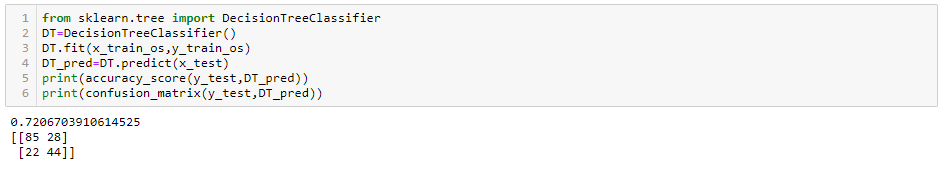
**Scaling the data:**

I had used Standard scaler since the data are highly different between features.



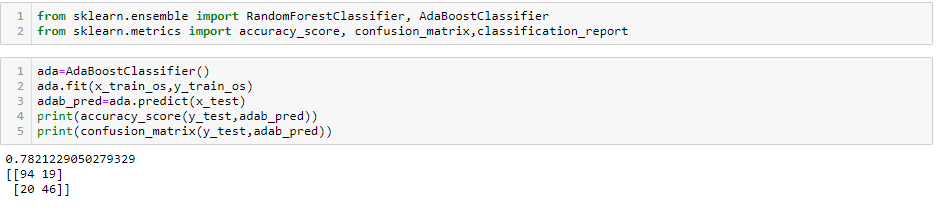
**Decision tree classifier:**

Decision tree classifier gives me the accuracy of 72% as shown below



**ADA Boost classifier:**

ADA Boost classifier gives me the accuracy of 78% as shown below



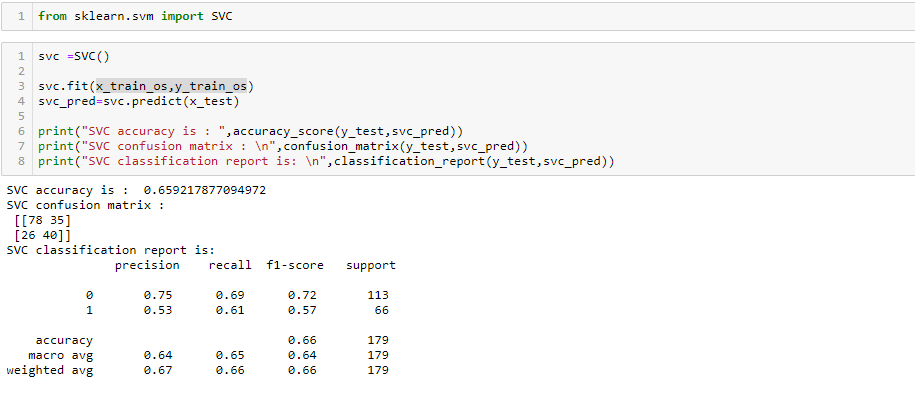
**Random Forest classifier:**

Random Forest classifier gives me the accuracy of 83% as shown below



**Support vector classifier:**

Support vector classifier gives me the accuracy of 65% as shown below



I had got the cross validation score for each model as shown below. Out of which Random forest classifier gives me the best accuracy of 83%.



# Random Forest

## What is Random Forest ?

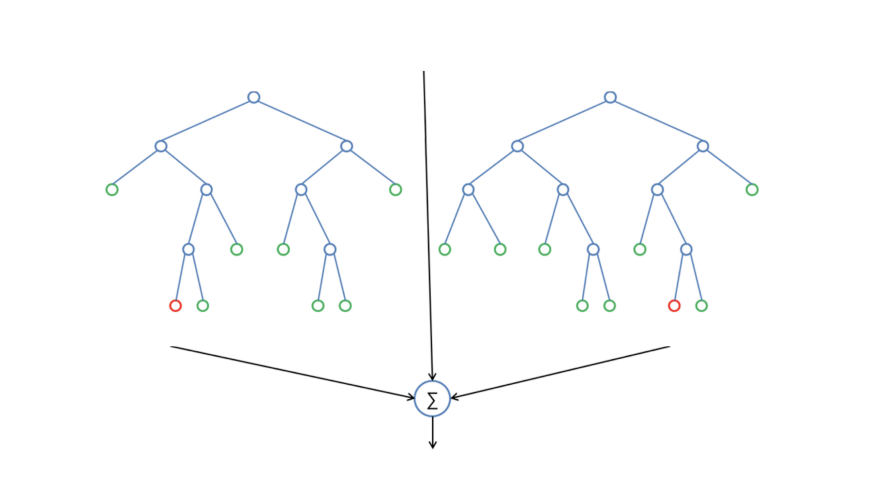
Random Forest is a supervised learning algorithm. Like you can already see from it’s name, it creates a forest and makes it somehow random. The „forest“ it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. With a few exceptions a random-forest classifier has all the hyperparameters of a decision-tree classifier and also all the hyperparameters of a bagging classifier, to control the ensemble itself.

The random-forest algorithm brings extra randomness into the model, when it is growing the trees. Instead of searching for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process creates a wide diversity, which generally results in a better model. Therefore when you are growing a tree in random forest, only a random subset of the features is considered for splitting a node. You can even make trees more random, by using random thresholds on top of it, for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Below you can see how a random forest would look like with two trees:



**HYPER PARAMETER TUNNING:**

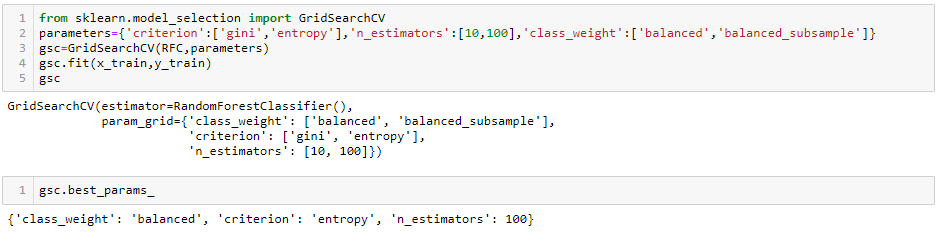
I had used Grid search CV for hyper parameter tunning to find the best parameter for Random forest classifier for the given data set.

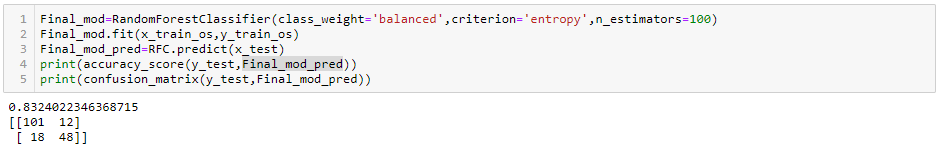
Best parameters from grid search CV are,

N\_estimator = 100

Criterion = Entropy

Class\_weight = balanced

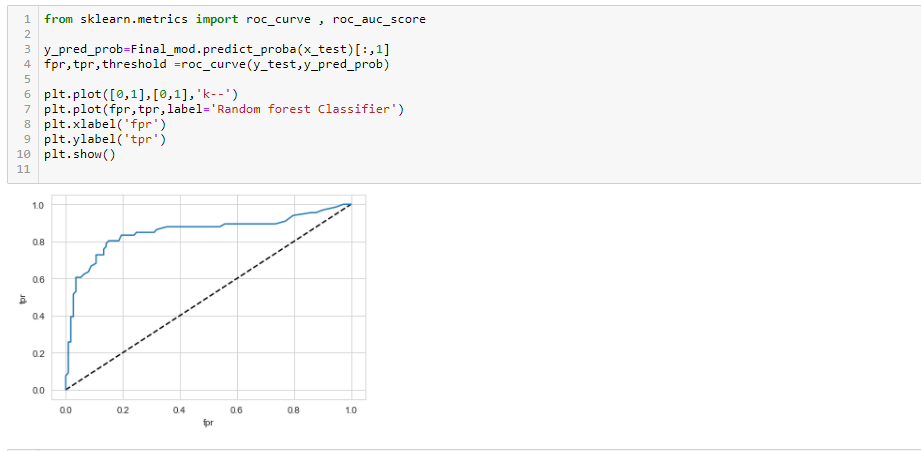




Final model gives me the best accuracy of 83% s shown above.

**AUC ROC CURVE:**

Another way to evaluate and compare your binary classifier is provided by the ROC AUC Curve. This curve plots the true positive rate (also called recall) against the false positive rate (ratio of incorrectly classified negative instances), instead of plotting the precision versus the recall.



The dotted line in the middle represents a purely random classifier (e.g a coin flip) and therefore your classifier should be as far away from it as possible. Our Random Forest model seems to do a good job.

Of course we also have a trade-off here, because the classifier produces more false positives, the higher the true positive rate is.

## ROC AUC Score:

The ROC AUC Score is the corresponding score to the ROC AUC Curve. It is simply computed by measuring the area under the curve, which is called AUC.

A classifiers that is 100% correct, would have a ROC AUC Score of 1 and a completely random classifier would have a score of 0.5.

# Summary

We started with the data exploration where we got a feeling for the dataset, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data preprocessing part, we computed missing values, converted features into numeric ones, grouped values into categories and created a few new features. Afterwards we started training 8 different machine learning models, picked one of them (random forest) and applied cross validation on it. Then we discussed how random forest works, took a look at the importance it assigns to the different features and tuned it’s performace through optimizing it’s hyper parameter values. Lastly, we looked at it’s confusion matrix and computed the models precision, and recall.