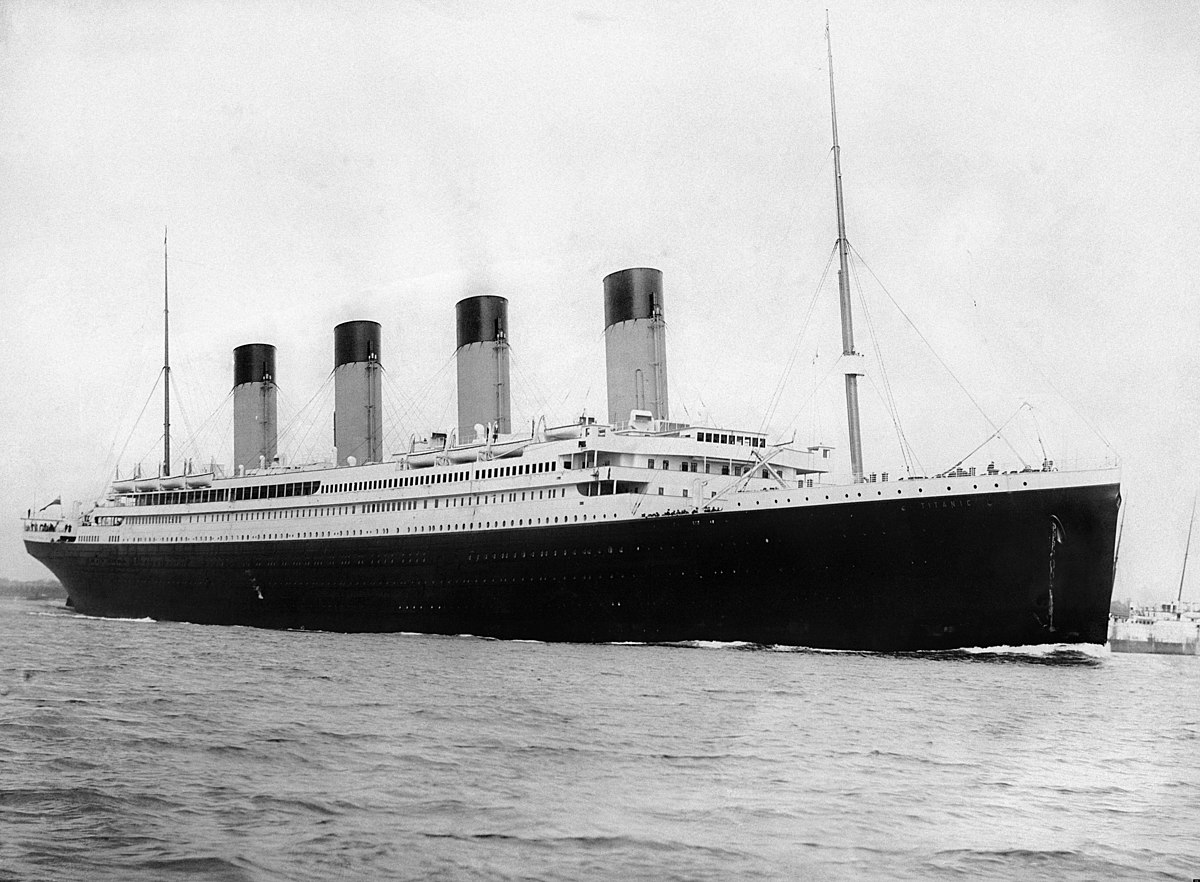
A study of Titanic Dataset Using Machine Learning

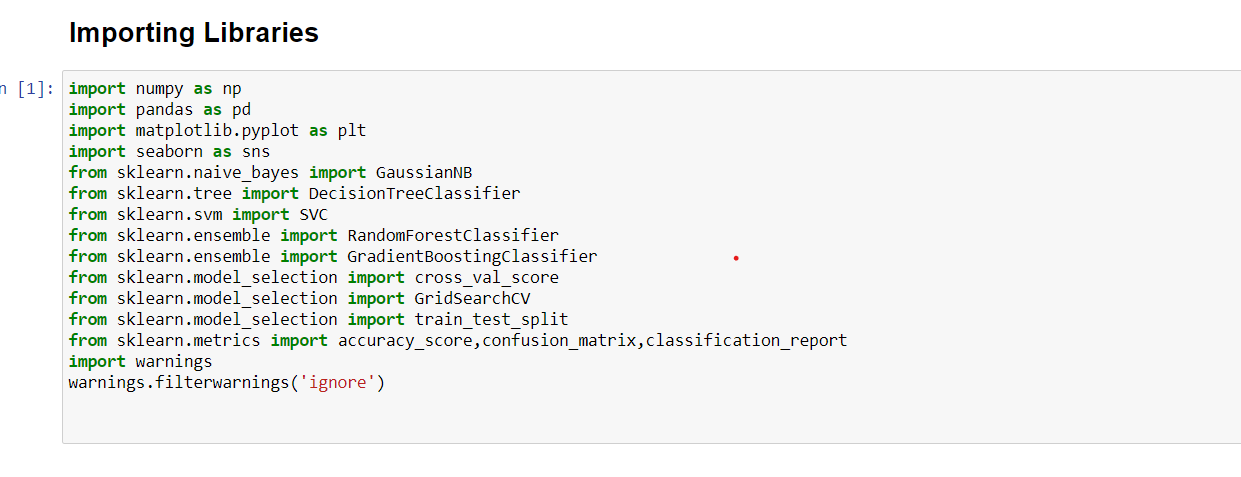
**Problem Definition**:

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

Our dataset contains the following variables

|  |  |  |
| --- | --- | --- |
| **Variables** |  |  |
| PassengerId | ID of the passenger |  |
| Survived | Survival of the passenger | 0=No 1=Yes |
| Pclass | Ticket class | 1=1st 2=2nd 3=3rd |
| Name | Name of the passengers |  |
| Sex | Passenger’s sex | F=Female M=Male |
| Age | Age of the passengers |  |
| SibSp | Number of siblings or spouses aboard the Titanic |  |
| Parch | Number of parents or children aboard the Titanic |  |
| Ticket | Ticket number |  |
| Fare | Passenger fare |  |
| Cabin | Cabin number |  |
| Embarked | Port of embarkation | C=Cherbourg Q=Queenstown  S=Southampton |

First, we will import the required libraries.



Then we will load the dataset using the code df=pd.read\_csv("Titanic\_dataset.csv").

**Data Analysis**:

This section discusses the data analysis in python Machine Learning in detail-

We can load the data directly from the GitHub.Here we are using pandas to load the data.Observe the following code,we wre using this code to load the data.



When you run the code, you can observe that the dataset loads and is ready to be analyzed. Here, we have downloaded the Titanic\_dataset.csv file and moved it into our working directory and loaded it using the local file name.

**Summarizing the dataset**

Summarizing the data can be done in many ways as follows −

* Check dimensions of the dataset
* List the entire data
* View the statistical summary of all attributes

Dimension of the dataset :-

You can use the following command to check how many instances (rows) and attributes (columns) the data contains with the shape property.

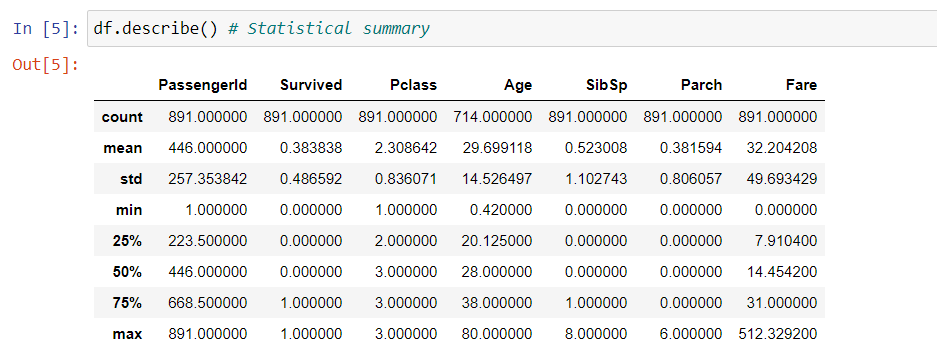


Then, for the code that we have discussed, we can see 891 instances and 12 attributes –



View the Statistical Summary :-

You can view the statistical summary of each attribute, which includes the count, unique, top and freq, by using the following command.



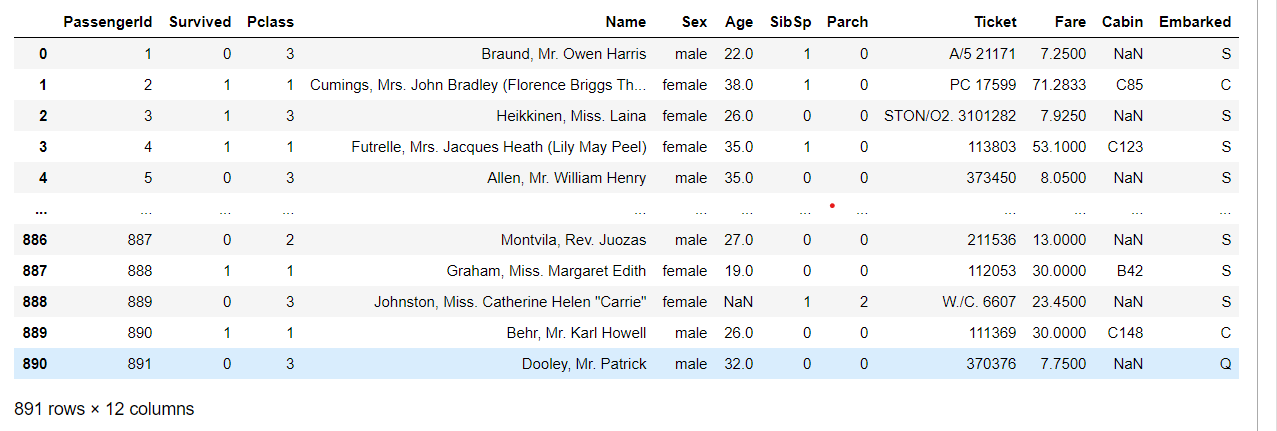
This command gives you output shown above that shows the statistical summary of each attribute .

List the Entire Data :-

You can view the entire data and understand its summary using the following command

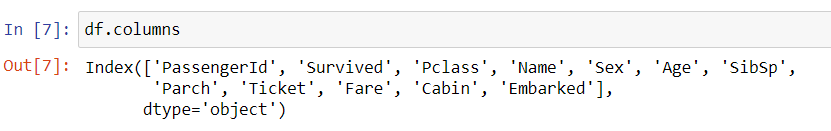


And the output will be the following column.



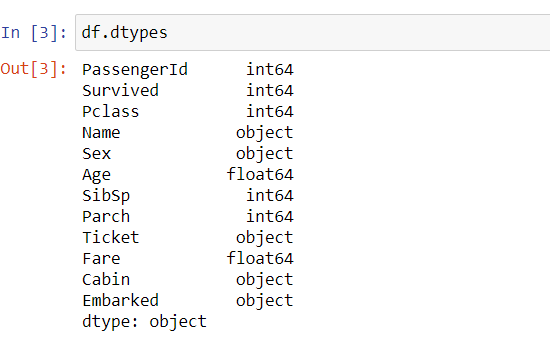
We will get the column list of the dataset by using the command df.columns

The output is given below-



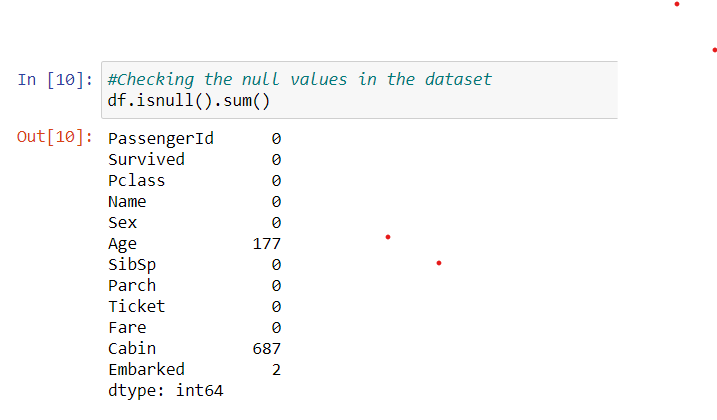
Before cleaning the data we have to check the datatypes of the each variable using the command df.dtypes

The output is given below-



In the first step we will investigate the Titanic dataset. This dataset contains 12 columns among which “Survived” is the target variable. We have to predict this variable using the other 11 feature variables. From the dataset clearly, we can observe that Nan values are present in some columns. Missing values irritate our algorithm, so it is important task to clean up the data.

Now we will check the null values in the dataset



The dataset contains missing values in the columns Age and cabin.we will replace these null values at the time of data cleaning.

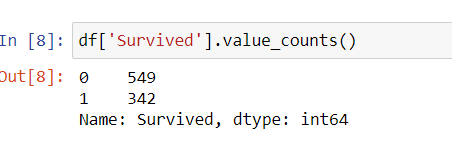
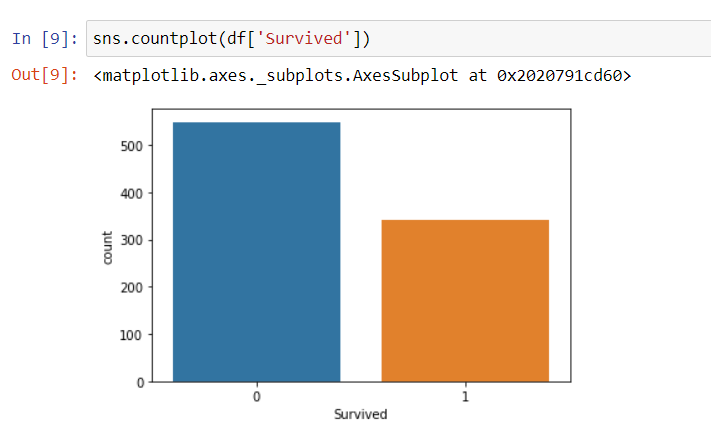
**EDA Concluding Remarks:**

Data Visualization :-

We can visualize the data using two types of plots as shown −

* Univariate plots to understand each attribute.
* Bivariate plots to understand the relationship between two variables.
* Multivariate plots to understand the relationships between attributes.

Let’s have a look at the **Univariate analysis** of the dataset

****

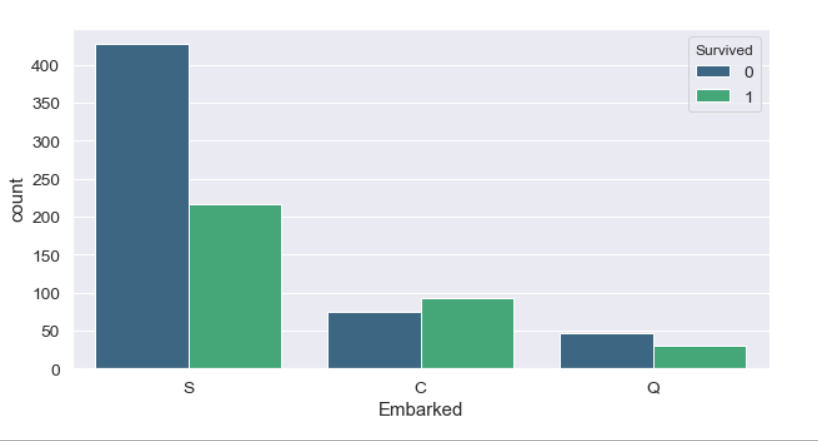
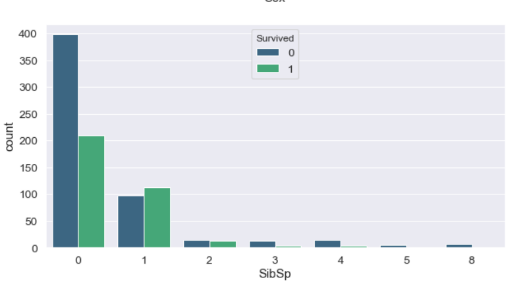
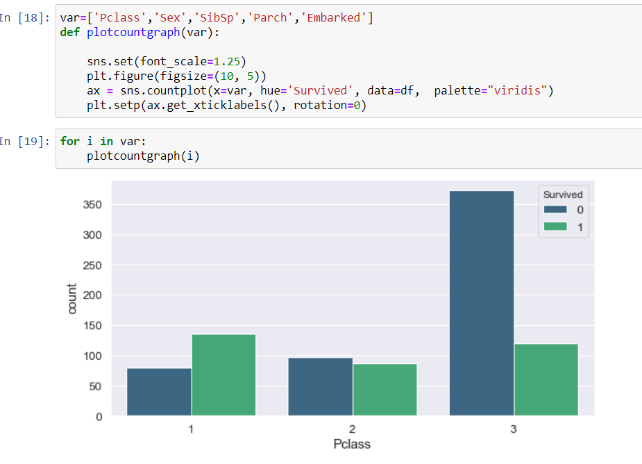
From this count plot, the blue colour bar denotes the number of passengers not survived and orange colour denotes the number of passengers survived. The number of passengers not survived is 549 and the number of passengers survived is 342.

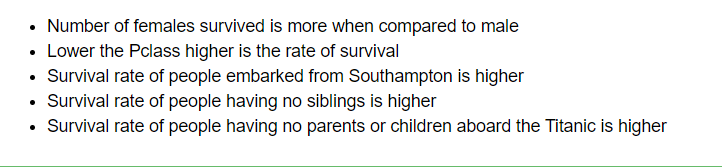
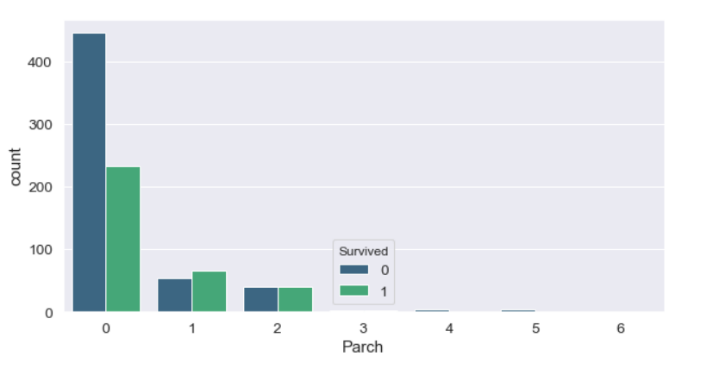
Like the above count plot we made the visualization of object type variables.



From these count plot we can infer that Male passengers are more in number than the female passengers. And it is clear from the count plot of column embarked that the number of passengers who board the ship from the port Southampton is more in number compared to Cherbourg and Queenstown.

Let’s look at the conclusions we made after the **bivariate analysis**



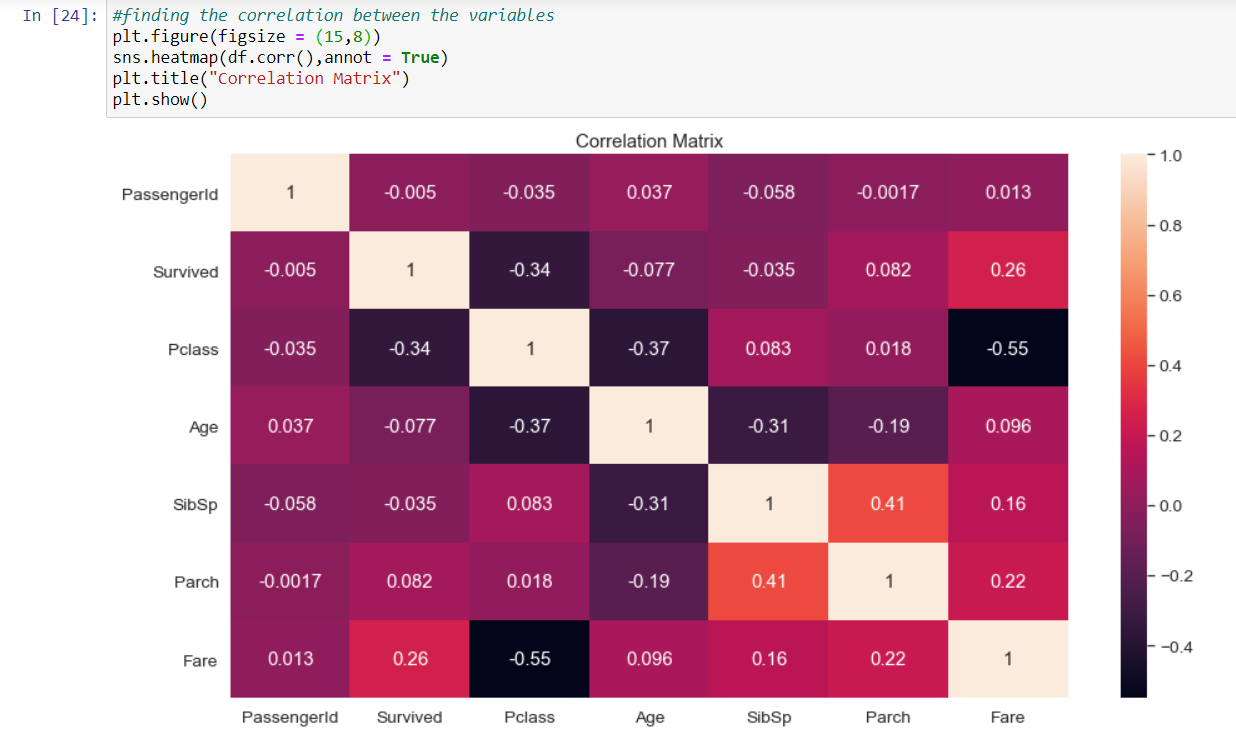


Observations from bivariate analysis:

**Multivariate Analysis**

Multivariate plots help us to understand the interactions between the variables.

To check the correlation between the variables we plot correlation matrix using heatmap.

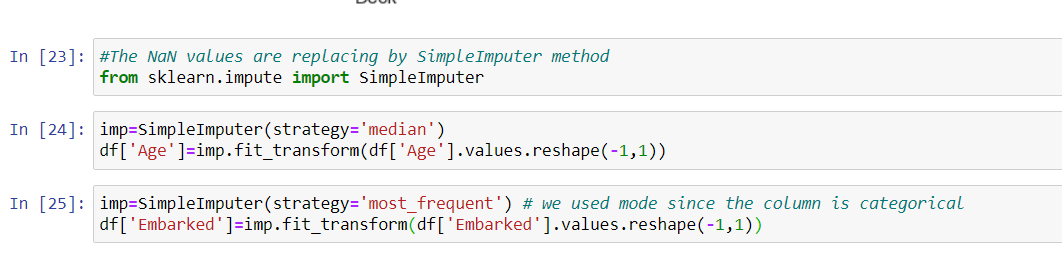


Fare have strong correlation with survival. So we can infer that people's survival is depends on the amount they paid for tickets.

**Data Preprocessing:**

Replacing NaN values using SimpleImputer:

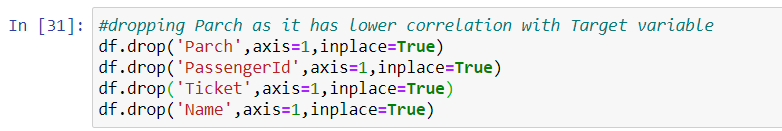
Now we re going to replace the NaN values in the columns of the dataset using the function SimpleImputer. For the categorical type variables we will replace null values using mode. and for numerical, we will replace null values using median or mean



Now we will drop the column Cabin. Since we create new column “Deck” from “Cabin”..

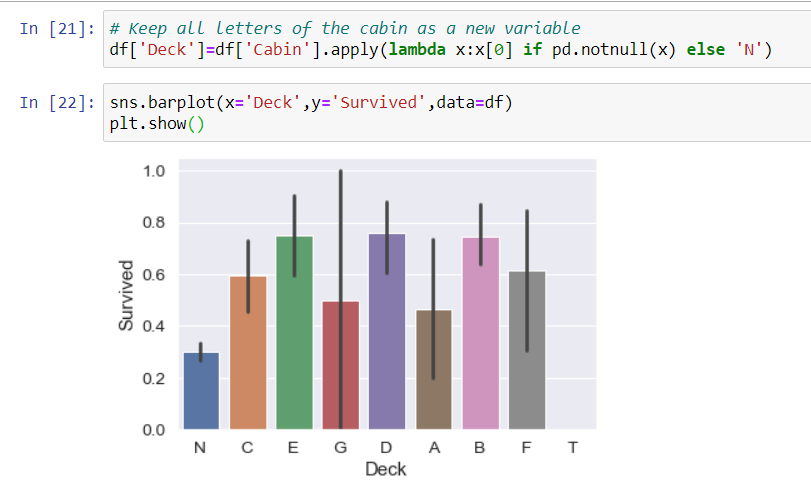


Also we are dropping columns which are not so important for further analysis and the column which have weak correlation with the target variable



**Creating new variables from the existing variables**

Now we will extract a new variable Deck from the variable Cabin using the command shown below.

 \* From the plot we can infer that the passengers belongs to the Deck E, D,B survived more compared to other decks.

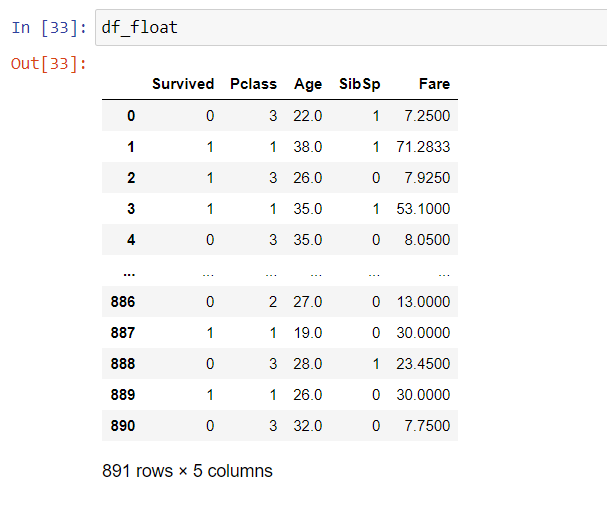
Now we will set up bins of ages and create column of Age group to sort ages into logical category. And we plot a bar graph of Age group against the target variable Survived.



\* From the plot it is clear that survival decreases with age.

Now we will extract the dataframe of numerical columns from original dataframe for ease using

df\_float=df.select\_dtypes(include=[np.number])

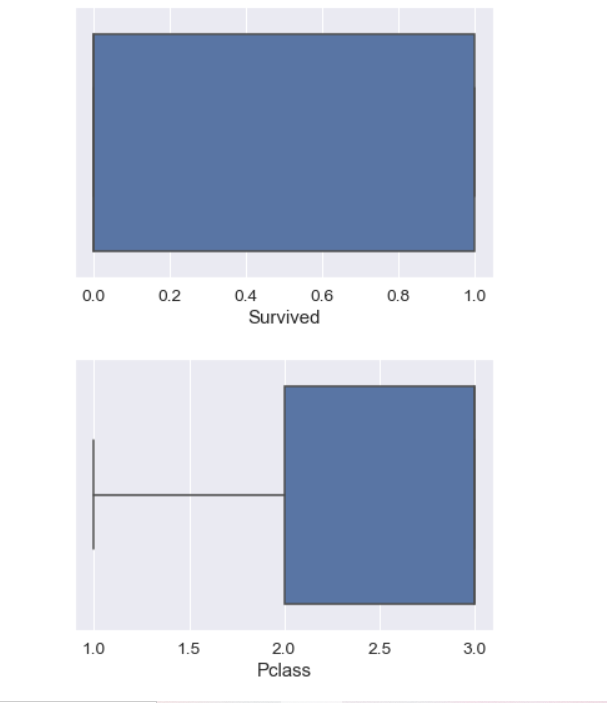
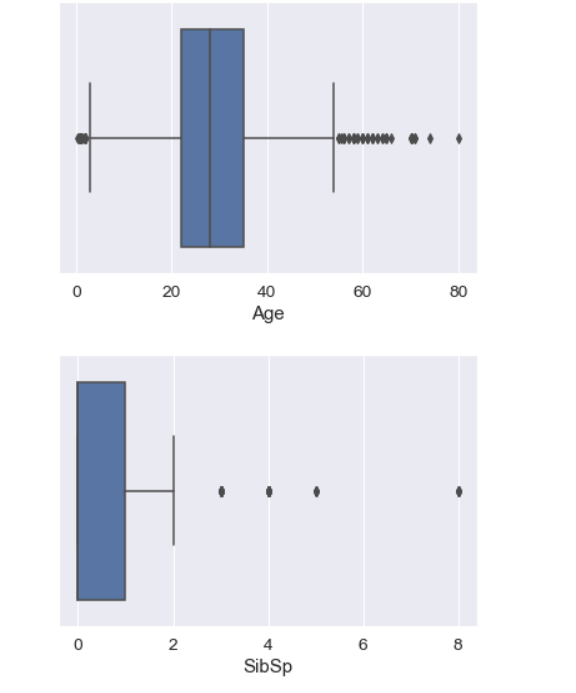
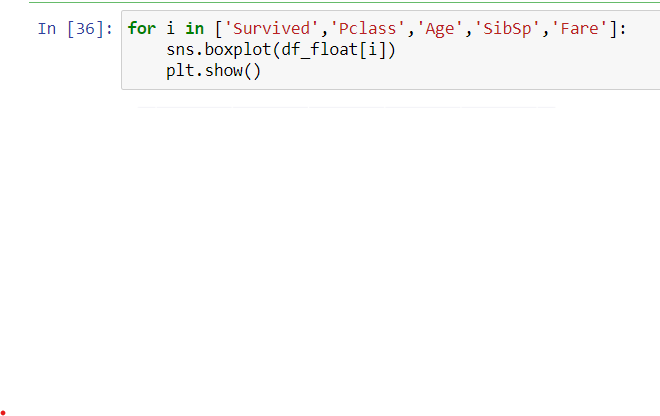


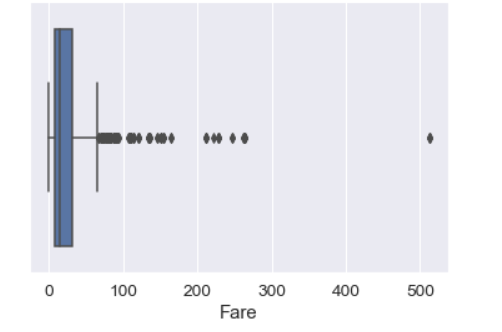
Checking Outliers:

Outlier is an observation that appears far away and diverges from an overall pattern in a sample. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results. So Detection and removal of Outliers is an important process in the Machine Learning.

Here in this dataset to check the outliers in the dataset, for that we will visualize the boxplot of dataframe. Here we are not checking the outliers of categorical columns, since outliers are not present in the categorical columns.

Here in this dataset we have plotted boxplot of columns using the command shown below-

.

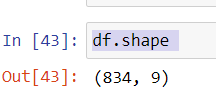


From these boxplots it is clear that outliers are present in the columns Fare, Age, SibSp.

Now we have to remove these outliers otherwise it will disturb our algorithms.

Removing Outliers:

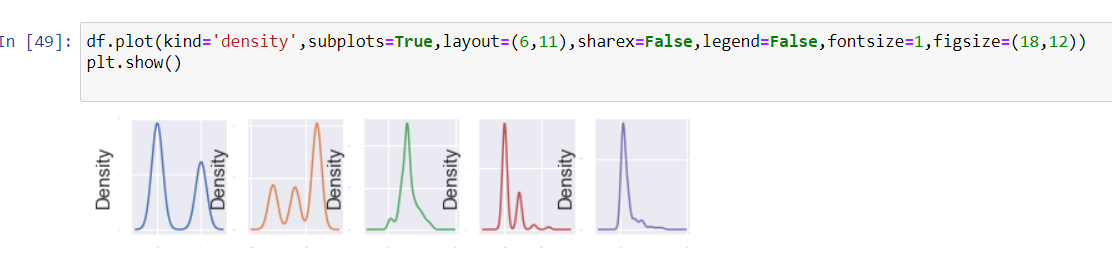
Removal of Outliers are very important step in Machine Learning. Here we will remove outliers using **z score test,** we will use the code  and setting the threshold value as 3, we will remove outliers. After removing outliers we have to check the shape of the dataframe .

The size of the new dataframe is (834,9).

We can calculate the loss of data after the removal of outliers. Here our data loss is around 6 %.

Distribution of data along columns:

Here we will plot the data distribution of different variables to check whether it is normally distributed or right or left skewed.



We can see skewness in the 5th distribution that is “Fare”, it is right skewed.

Now we will remove this skewness.

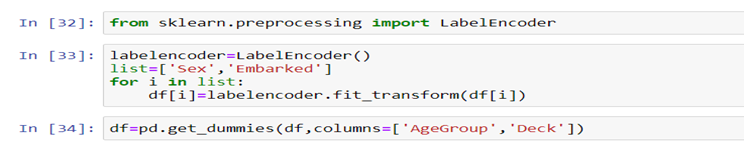
Converting Categorical columns into Numerical:

Most algorithms cannot do anything with strings, so the variables are often recoded before modelling. Label Encoding maps non-numerical values to numbers. For example, for Sex , 0 and len(sex)-1, which is, 1.

This leads to another problem. Many algorithms assume that there is a logical sequence within a column. However, this is not always expressed by the numerical ratio. Therefore it is needed to one hot encoding the variables afterwards. The column Sex then becomes two columns Sex\_1 and Sex\_2, in which it is binary coded whether someone was male or female. So the algorithm can usually process the information better.

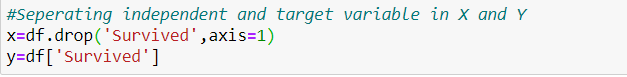
Here in this dataset we are going to convert categorical columns into numerical using LabelEncoder and pd.get\_dummies ()

The function LabelEncoder is using for categorical variables with up to 3 classes.if the variable contains more than 3 variables we will apply OneHotEncoder or pd.get\_dummies etc.



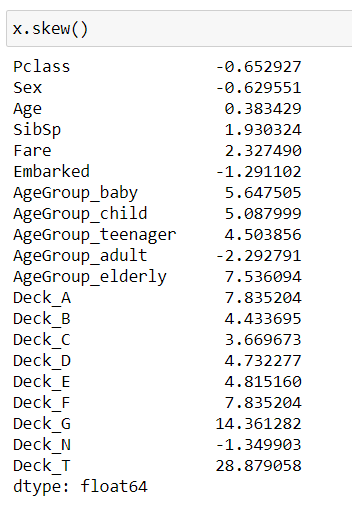
Separating the Feature variables and Target variables:

Before building a machine learning model, data is always split into two different parts that are called Training and Testing. For the training purpose of the model, we only expose the training data and never allow testing data to be exposed. Once the model gets trained using that data, we make use of the model to compute the predictions over the testing data, which is stored in a single variable known as y\_pred. We can store it in a different variable as well. We will first define the independent variable and dependent variable x and y, respectively. Now we will split our data. Use the below code to the same.



Checking Skewness:

We will check the skewness of the columns by using x.skew().Here we are checking only the skewness of independent variables, since we can’t do any transformation to our target variable.



Keeping the threshold value as ±0.5, we will find out the skewness.

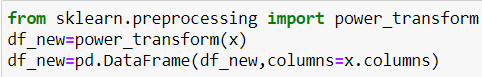
Here skewness is present in the continuous variable Fare only. The other variables whose values are greater than ±0.5 are Categorical variables. We don’t consider them as skewed according to the value,since skewness is not present in the categorical variables.

Removing Skewness:

We will remove the skewness by applying the Power Transform function. if the skewness remains in the data it will disturb our algorithm. When removing skewness, transformations are attempting to make the dataset follow the Gaussian distribution. The reason is simply that if the dataset can be transformed to be statistically close enough to a Gaussian dataset, then the largest set of tools possible are available to them to use. So it is necessary to remove the skewness in the data.

There are many methods to remove the skewness of the data. Here we are using powe transform, since the values are positive and negative.

We removed skewness of the variable Fare using the command shown below



Scaling the dataset:

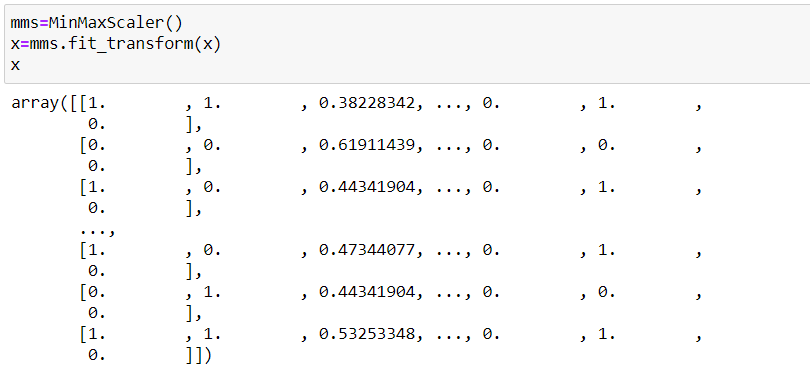
Feature scaling in machine learning is one of the most important steps during the preprocessing of data before creating a machine learning model. This can make a difference between a weak machine learning model and a strong one.

Feature scaling is essential for machine learning algorithms that calculates distances between data. If not scaled the feature with higher value range will start dominating when calculating distances. And leads taking long time to train a model, A model can be so big and that it’s can’t fit well into the working memory of the training device etc.

Here in this dataset we will apply MinMaxScaler to normalise the dataset. We will use the code

to import the MinMaxScaler.

After fitting MinMaxScaler into the dataframe. Our x will transform and shown below.



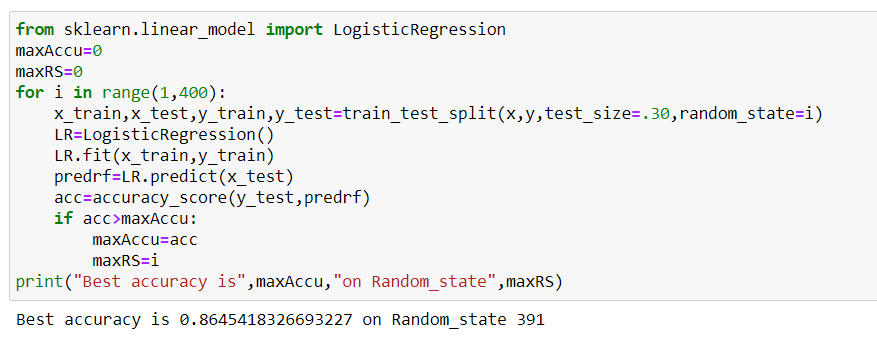
**Here in this dataset we are using Classification models, since our target variable Survived contains two classes/labels 0 and 1. And this is a Binary classification.**

**Building Machine Learning Models:**

Finding Best Random State:

First, we will find out the best random state, at that random state accuracy score will be maximum. random\_state number splits the test and training datasets with a random manner. In addition to that, it is important to remember that random\_state value can have significant effect on the quality of your model (by quality I essentially mean accuracy to predict). For instance, If you take a certain dataset and train a regression model with it, without specifying the random\_state value, there is the potential that every time, you will get a different accuracy result for your trained model on the test data. So, it is important to find the best random\_state value to provide you with the most accurate model. And then, that number will be used to reproduce your model in another occasion such as another research experiment. To do so, it is possible to split and train the model in a for-loop by assigning random numbers to random\_state parameter:

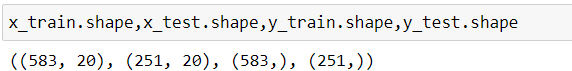
Here in this dataset we use the below command to find out the best random state-



Our best random state is 391 and the accuracy score is 86.45%

Train Test Split:

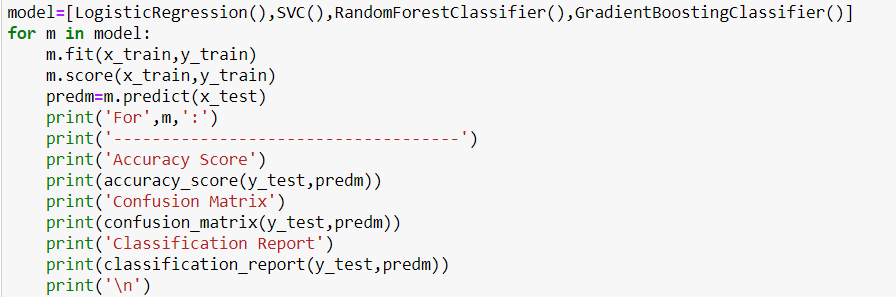
In this step we will split the train and test set and check the dimension of the training and testing set.



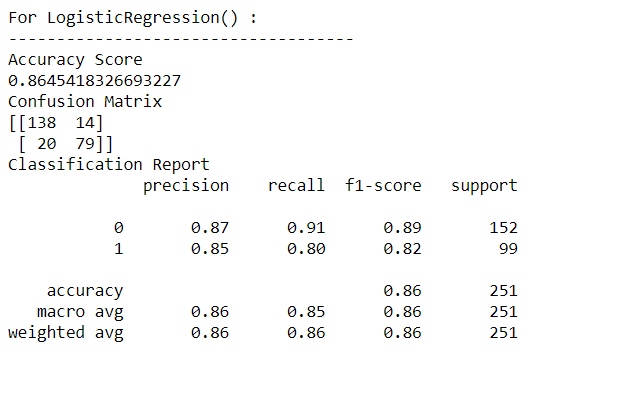
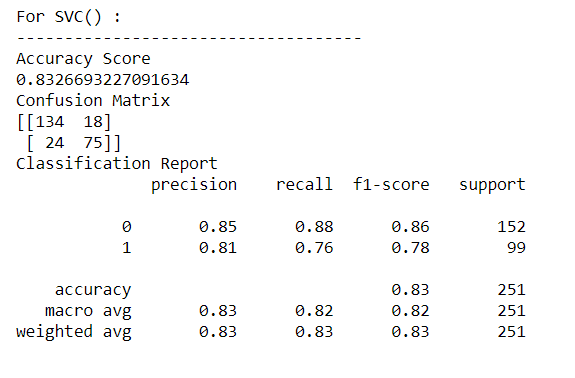


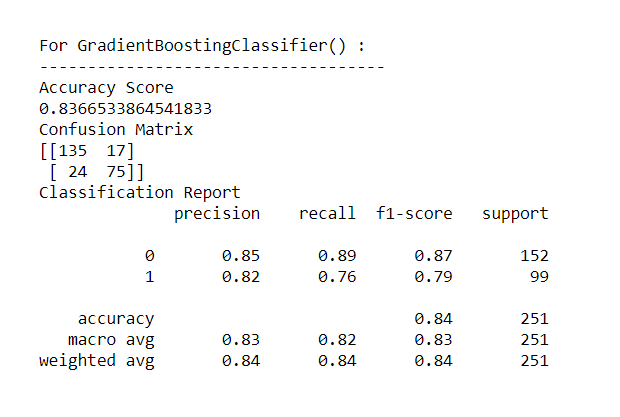
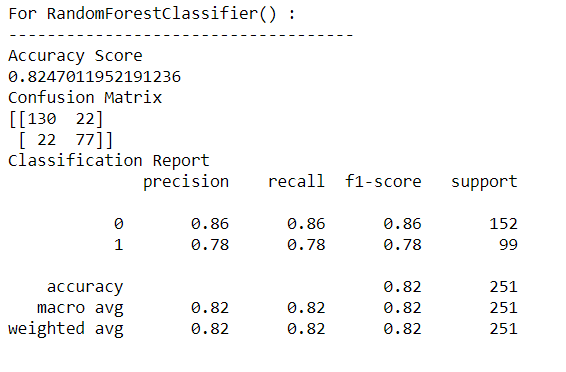
We will now build the machine learning model using different machine learning Classification algorithms that are **Logistic Regression, SVC, RandomForestClassifier, GradientBoostingClassifier**. Logistics regression comes from linear models, SVC from Support vector machine whereas random forest ang Gradient boosting are ensemble methods. We will first import these and then will pass the training data to both the models. After it gets trained, we will compute predictions over testing data and store in different variables.

Here we use for loop and we get the result in one step. Use the below code to the same.



The Output is shown below-



Accuracy score of LogisticRegression model is 86.45%

Accuracy score of SupportVectorClassifier is 83.26%

Accuracy score for RandomForestClassifier is 82.47%

Accuracy score for GradientBoostingClassifier is 83.66%

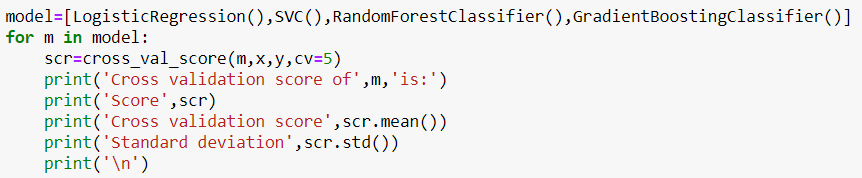
Accuracy score is maximum for the model LogisticRegression. Let’s check the accuracy score after the Cross validation and we can determine which is our best model.

Cross Validation:

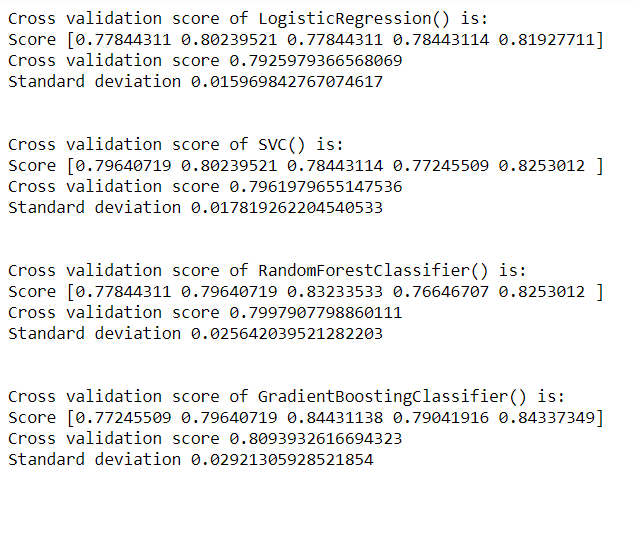
In machine learning, we can’t fit the model on the training data and can’t able to say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. For this purpose, we use the cross-validation technique.

Cross-validation is a technique in which we train our model using the subset of the dataset and then evaluate using the complementary subset of the dataset.

Here in this dataset we use the following command to find out the cross validation score.



The output is also shown below-



Cross validation score for Logistic Regression =79.25%

Cross validation score for SVC=79.61%

Cross validation score for RandomForestClassifier=79.97%

Cross validation score for GradientBoostingClassifier=80.93%

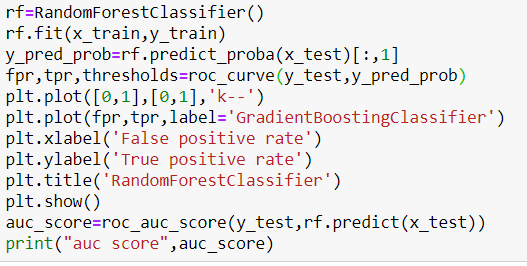
After comparing the cross-validation score with the accuracy score of corresponding models we reached into the conclusion that our best model is **RandomForestClassifier**, because the difference between the accuracy score and cross validation score is minimum for this model.

**AUC-ROC Curve:**

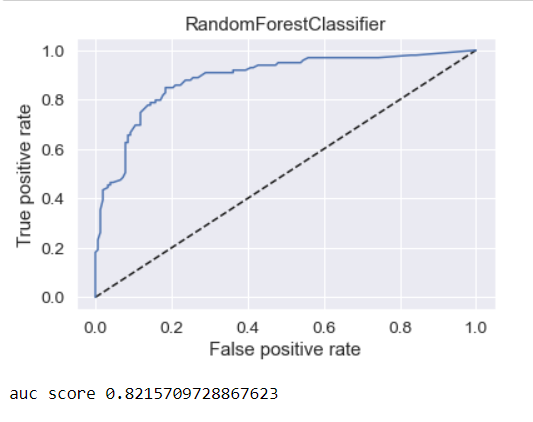
We will import roc\_auc\_score and roc\_curve from sklearn.metrics for findind auc score of model and plotting the roc curve of the model.



In this dataset we have plotted the roc curve of RandomForestClassifier, our best model and find out its auc score using the command given below-



And the output is



The AUC score for the RandomForestClassifier is 82.15

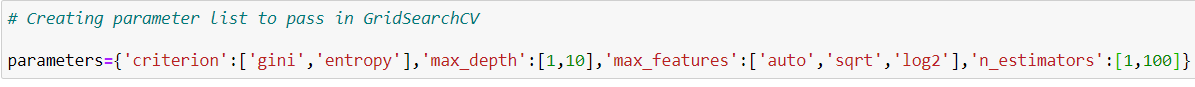
**Hyper Parameter Tuning:**

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.

However, there is another kind of parameters, known as **Hyperparameters**, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

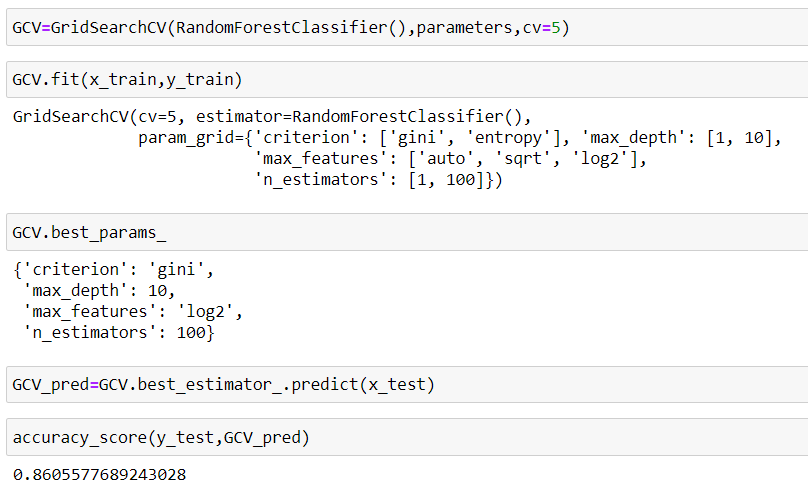
Usually we are using GridSearchCV for the purpose of tuning.

First we create a parameter list as shown below



Then we will fit it on to training set and finding out best parameters then prediction and finding out the accuracy score.

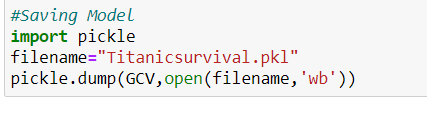
All these steps are given below



After Hyperparameter tuning our accuracy score improved a lot and becomes 86.05%.

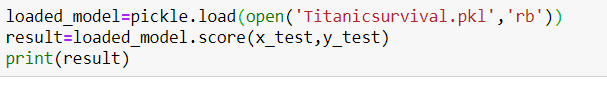
**Saving the Model:**

Here in this step we will save our model by importing pickle and using the following code

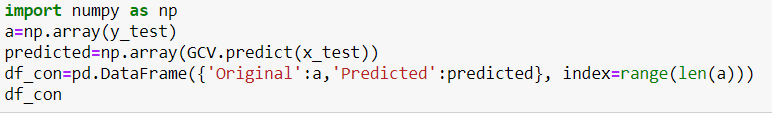


**Conclusion:**

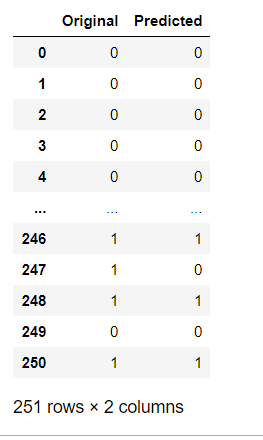
We will load the file that we saved and we made prediction using training set and compare it with the original test set.



The command for prediction is



The output is given below-



We can see that most of the values are matching. So our model with accuracy score around 86% performing well.

**Concluding Remarks:**

We made the entire journey in a small data science project. We explored the data, cleaned up the data, then we modified features and created new ones and in a last step we made a prediction with a random forest tree classifier.