**Titanic Project**

Everyone in the world must be aware of the Titanic Ship incident. RMS Titanic was a British passenger liner operated by the White Star Line that sank in the North Atlantic Ocean on 15 April 1912, after striking an iceberg during her maiden voyage from Southampton to New York City. So we have a dataset of passengers in Titanic while it was sinking. It gives us information about multiple people like their ages, sexes, sibling counts, embarkment points, and whether or not they survived the disaster. Based on these features, we have to predict if an arbitrary passenger on Titanic would survive the sinking or not.

For analysing Titanic dataset, we will use Jupyter Notebook for writing Python codes. Always starting with a python code, we firstly import pandas library and numpy into the notebook. Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages. Pandas works well with many other data science modules inside the Python ecosystem and is typically included in every Python distribution. Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data, including Loading the dataset, Data cleansing, Data normalization, Data visualization and much more. In fact, with Pandas, you can do everything that makes world-leading data scientists vote Pandas as the best data analysis and manipulation tool available. Numpy is a Python package which we have mainly used for linear algebra, apart from this Numpy is used for various other purposes.

After importing libraries for mathematical and statistical analysis, libraries for visualization are imported, which include matplotlib and seaborn. Seaborn utilises fascinating themes, while matplotlib is used for making basic graphs. Also we can import warnings.filterwarnings(‘ignore’) which is called to not to print any warnings for the user in the program. Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program.

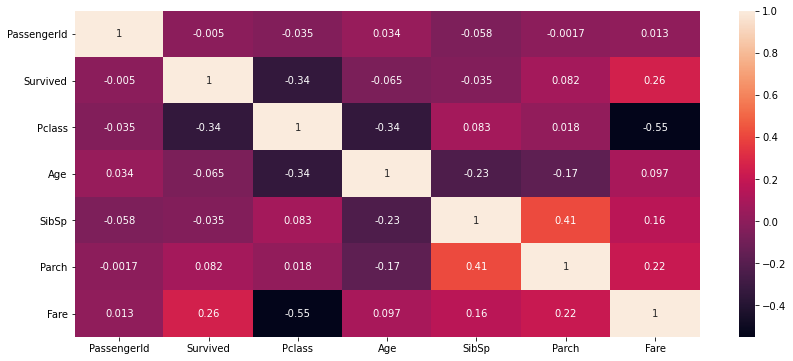
For analysing the data which we want to analyse, first thing we do is import the dataset into the notebook. Dataset come in various formats, but we convert them into an excel file that is “.xlsx” or a “.csv” file and upload it into Jupyter.

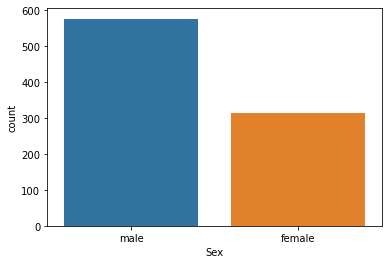
Once we upload it as a csv file, we can directly import using “ pandas.read\_csv(‘ Name of the file.csv ’) ”. If we want to import an excel file we can import the file by using “ pandas.read\_excel(r ” PATH OF THE FILE.xlsx ”) ”. In my file I have used “ csv ” file. I named the file as “ df ”. So my code goes as - df=pd.read\_csv(“ titanic.csv ”) where pd is pandas as I have imported Pandas as pd and “ titanic “ is the name of the dataset csv file. We start with df.head() and df.tail() which gives the output as the first 5 rows of the dataset and last 5 rows of the dataset respectively. The output contains the dimensions of the table at the bottom i.e., 5 rows x 12 columns. The features (columns) of the dataset are : PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, ticket, Fare, Cabin and Embarked and the shape (dimension) of the dataset is 891 rows x 12 columns which is checked using df.shape() where df is name of th dataset and .shape() is a pre-defined function in python which gives shape of the dataset as output.

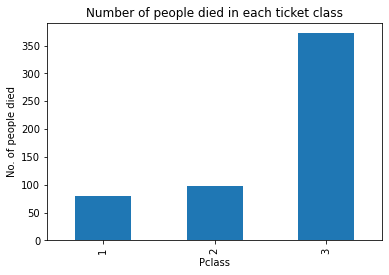
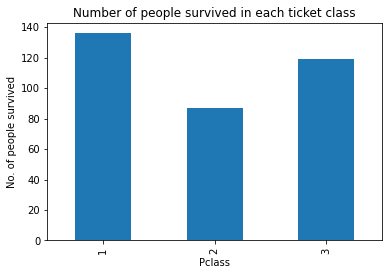
For gaining more information of our dataset df.info() is used, which shows the number of columns along with the names of the features, non null counts, types of data present in the columns and how much memory is the dataset using. As the output we got the names of all 12 features, data types such as float64 (2 columns), int64 (5 columns) and object (5 columns) and memory usage is 83.7+ KB. To check only the data types df.dtypes is used, using df.columns all the features of the dataset can be called and df.values shows all the values in the dataset in the form of an array.

Now let us focus on data cleaning and data processing, starting with dealing with null or missing values. To check if there are any null or missing values df.isnull() is used which gives output in a DataFrame format and the values are True where the data is missing or null value is present and False in place of any other value. To check value counts of null values we will use df.isnull().sum(). There were 2 features with null values ‘ Age ‘, ‘ Embarked ’ and ‘ Cabin ‘, 177, 2 and 687 respectively. Replacing the null values in ‘ Age ‘ with median of the ‘ Age ‘ column using .fillna() method and replacing null values with .fillna(method= ‘ pad ‘) in ‘ Embarked ‘ column. Didn’t remove or replace null values from Cabin column has it will be dropped from the dataset as it won’t be that important for predicting the survival. To check mean, median, standard deviation, 25th percentile, 75th percentile, total counts and maximum values of each columns I have used df.describe(). df.descirbe() helps to view all the above mentioned into a table altogether. There is no need to call all functions for individual columns.

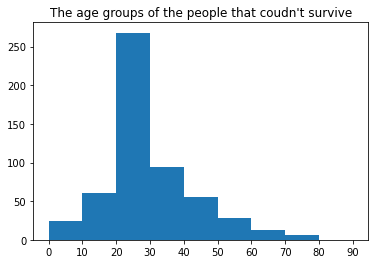
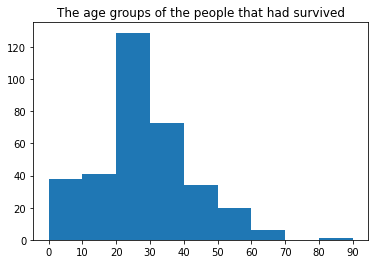
Grouping the dataset into 0 and 1 which are the values of our target column ‘ Survived ‘. To check the correlation sns.heatmap(df.corr()) was used as it shows correlation between the dataset columns using colour scheme. Lighter the cell colour more the correlation between those two columns and vice versa. Figure mentioned below.



**

**

Checking the value counts of feature ‘ Sex ‘ i.e. male and female with values nearly 600 and 300 respectively. More than 530 died and more than 300 survived. In our target column ‘ Survived ‘, 0 means died and 1 means survived. In the third image we are trying to visualise number of people survived in different ticket class and we can conclude that highest people survived are from class 1 ( nearly 140) followed by class 3 (120) and only 80-85 survived in class 2. In the last image we can see that people in class 3 died the most (more than 350) and in class 2 100 died and less than 100 died in class 1.

**

The images shows age wise distribution of number of people who survived and who died.

0-10 -> Nearly 40 people survived and around 25 died from this age group.

10-20 -> Approx. 40 people survived from this age group and more than 50 died.

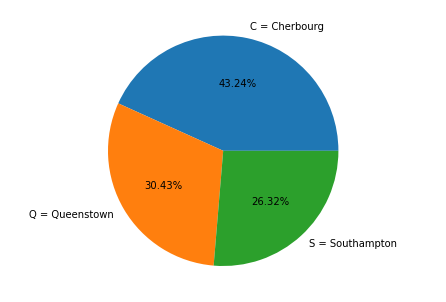
20-30 -> 120+ people from this age group survived which is highest and also highest deaths are from this group that is 250+.

30-40 -> Nearly 75 survived and approx. 100 died from this age group.

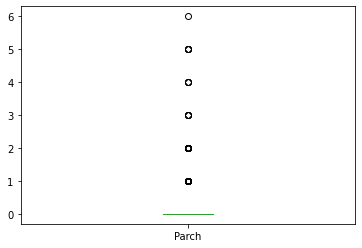
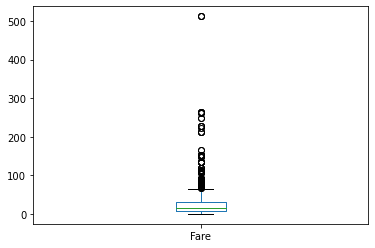
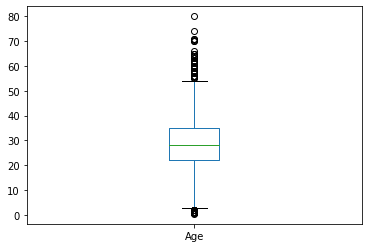
40-50 -> Around 50 died and 50 survived in this age group.

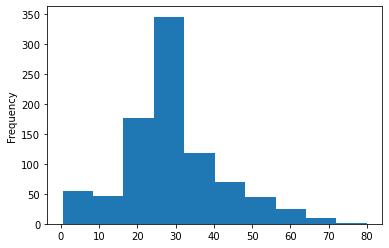
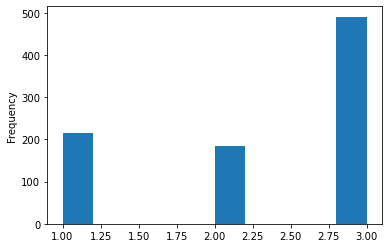
50-90 -> There are less 40 people who survived and approx. 50 who died.

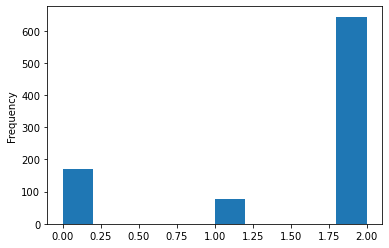
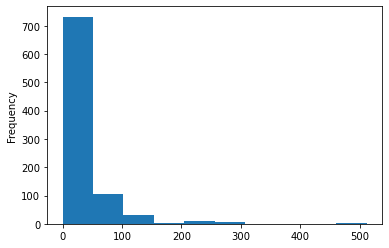
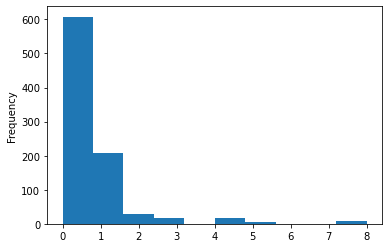
Checking ‘ SibSp ‘ w.r.t ‘ Survived ‘, 1 in ‘ SibSp ‘ had 53% data which is the highest and 8 in ‘ SibSp ’ has the lowest 0% data. Now checking ‘ Pclass ‘ w.r.t ‘ Survival ‘, class 1 has 62% and 3 has 24%. Below figure shows the percentage of people who were embarked from different location. 43.24% from Cherbourg, 30.43% from Queenstown and 26.32% from Southampton.

**

Now let’s separate target column and independent columns by naming y and x respectively. From x dropping ‘ PassengerID ‘, ‘ Name ‘, ‘ Ticket ‘, ‘ Cabin ‘. Also will drop ‘ Survived ‘ as it is the target column and y will contain only ‘ Survived ‘. Now x has shape 891 rows x 7 columns and y has shape 891 rows x 1 column. Using lambda function encoding values of C, Q and S into 1, 2 and 3 resp., and also male and female into 1 and 2 in column ‘ Sex ‘. There are some outliers usually present in the dataset, so I have checked if there are outliers present in our dataset with the help of boxplot and found some outliers in few columns of the dataset. To handle outliers I took help of zscore imported from scipy.stats.. Also checking skewness and handling skewed data is as important as handling outliers as they affect the results. The images with box plots are used for checking the outliers and histograms are used for checking skewness.

**

**

**

Let us separate our independent variables altogether from our dependent variable i.e. x and y for train test split. So to all our independent variables I have denoted x and for dependent variable as y that is ‘Attrition’. It’s time for splitting the data into training data and testing data. For splitting the data train\_test\_split was used where test\_size was 30% and random state=45. So finally we got out x\_train, y\_train, x\_test and y\_test and their shape is 623x7, 623x1, 268x7 and 268x1 respectively. x\_train contains the training independent variables where as x\_text contains testing independent variables, y\_train contains training dependent variable and y\_test contains testing dependent variable.

For modelling the data, I have used Logistic Regression, Random Forest Classifier, Ada Boost Classifier, Decision Tree Classifier, SVC and GaussianNB. While checking the model most importance is given to accuracy score followed by confusion matrix and to rest. So, while evaluating most accuracy score was of Random Forest Classifier with 83% accuracy. Worst performing model was SVC (svmkernel = ‘ poly ‘) as its accuracy was 68%. Rest of the models such as Logistic Regression, Decision Tree Classifier, Ada Boost Classifier and GaussianNB had accuracy score of 82%, 79%, 82.08% and 81% respectively. Also, while using cross validation Random Forest Classifier came out to be the best performing model with score mean of 80.8% and worst was SVC with an improved score mean of 67.4%. Saving the best ML model i.e. Random Forest Classifier using joblib function.

After using Hyperparameter Tuning, SVC with parameters such as 'C': 1, 'kernel': 'linear' was best model with score of 78.8%. Random Forest Classifier’s best parameters were 'max\_features': 'auto' and best score was 77.5% and Logistic Regression’s score was 78.5%. We found tpr, fpr and threshold to get the AUC-ROC Curve. AUC-ROC Curve for Logistic Regression looks good as its score is more than 80.7%.

