**Titanic Survival Machine Learning End-to-End Project Analysis**

In this article, I am going to walk you through all the major steps involved in building and completing an end-to-end Machine Learning (ML) project. For this project, I have chosen the popular Titanic dataset and I will show you the detailed procedure and techniques I used to predict whether an arbitrary passenger survived the shipwreck or not.

The topics that I am going to cover are listed down:

* Defining and understanding the problem
* Data Collection and Data Exploration
* **Exploratory Data Analysis (EDA)**
* Data pre-processing
* Selecting and Training a few Machine Learning Models
* **Cross-Validation and Hyperparameter Tuning**
* Saving the final ML model
* Concluding Remarks

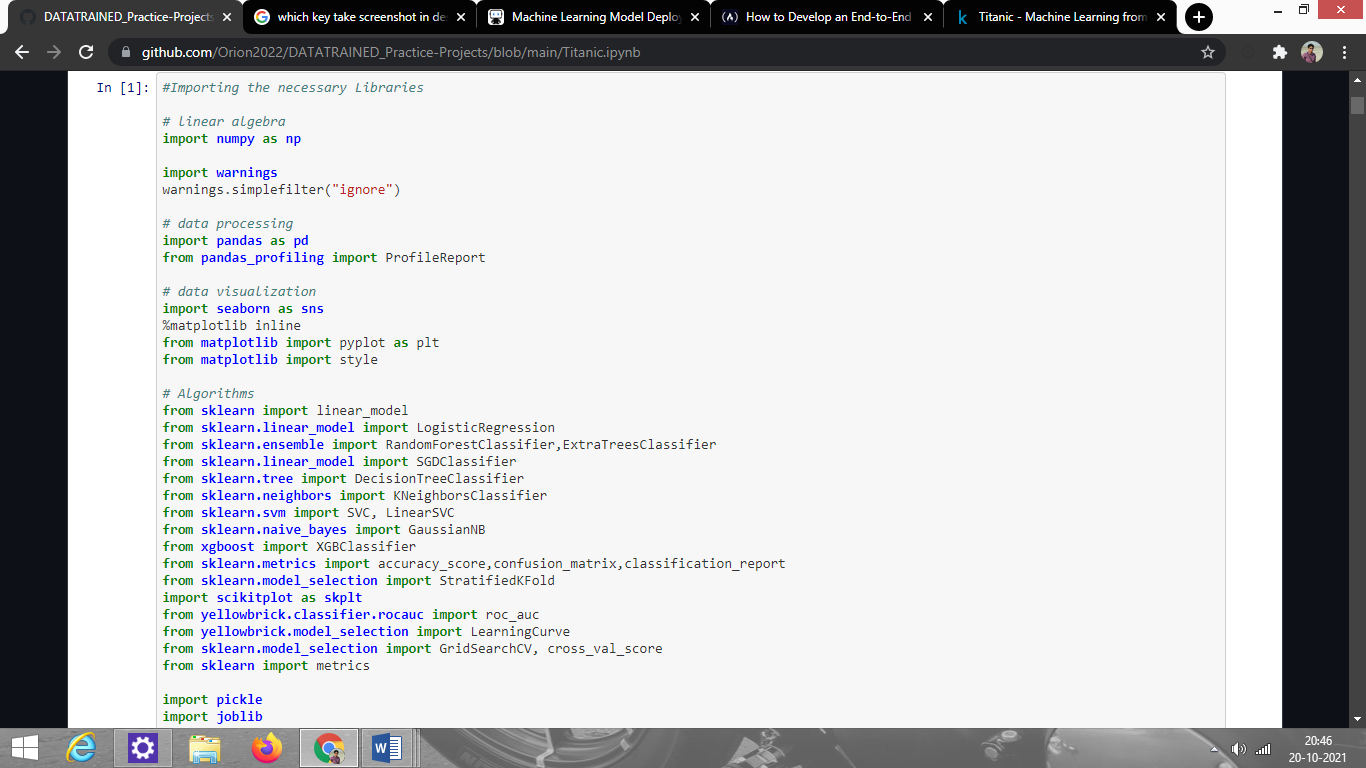
Let’s not waste any time and dive right into it.

**Defining and understanding the problem:**

The first step in any machine learning project is to define the problem that you are going to solve. If you don’t have the data then you have to collect the data first. You can easily get access to the Titanic dataset on the kaggle.com website. As I have already acquired the dataset so I am going to define the problem statement: In early 1912, during her maiden voyage, the widely considered “unsinkable” ship Titanic sank after colliding with an iceberg. Unfortunately, as there weren’t enough lifeboats for everyone on board, this resulted in the death of passengers and crew. However, many passengers survived this disaster.

The dataset 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 build a predictive model that could predict if an arbitrary passenger on Titanic would survive the sinking or not.

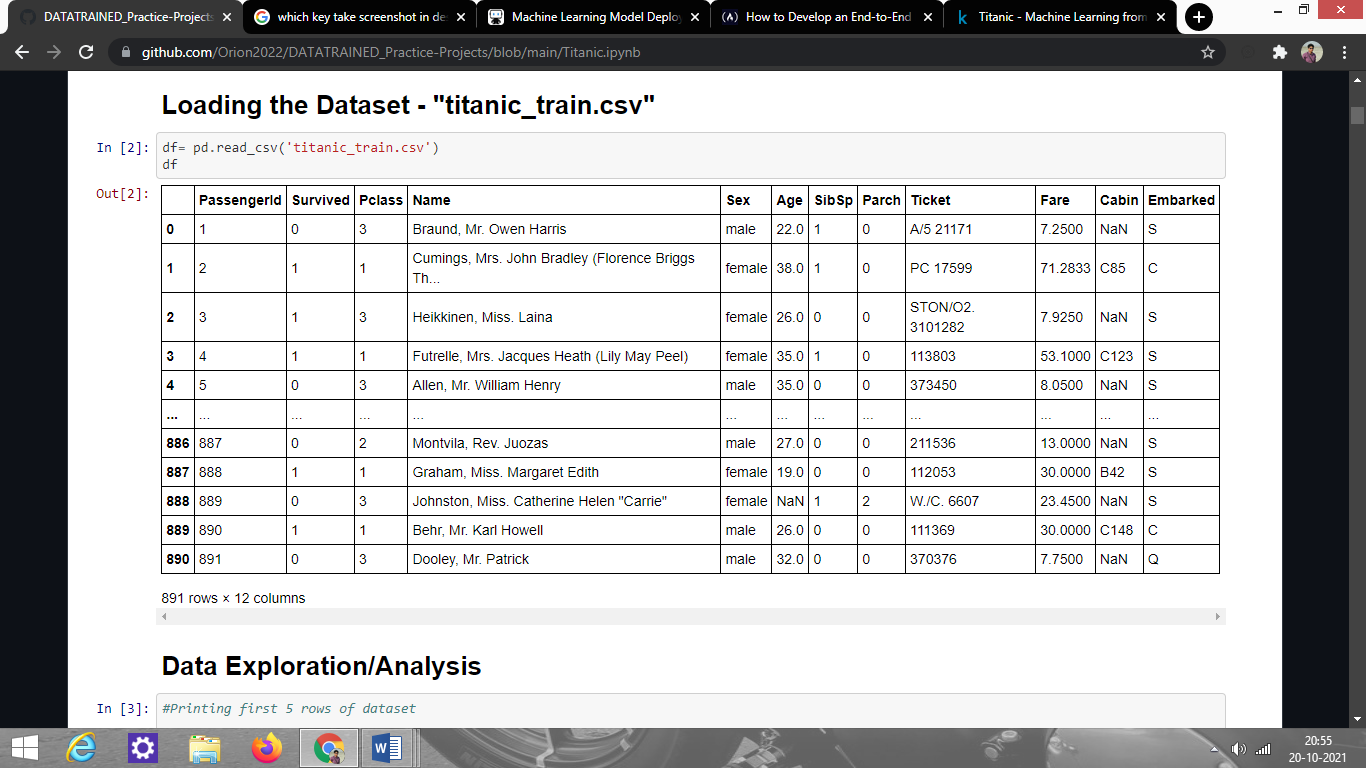
First of all, let’s import all the necessary libraries/packages that will be used in our model building.



**Data Collection and Data Exploration**

* ***Loading and reading the dataset***:

After importing the libraries, I have loaded and read the Titanic dataset into a dataframe in my Jupyter Notebook which can be achieved through the pandas read\_csv method as shown below:



Next, I take a look at the rows and columns of the dataframe ‘df’ and try to understand the data. From the line of code shown above, we can see that there are 891 rows and 12 columns present in our dataset.

* ***Understanding the dataset***:

Before going into data analysis, we have to understand the data i.e., each column present in the dataset. Below I have listed the features with a short description:

1. PassengerId: Unique Id of a passenger.
2. Survived : Survival of a passenger (0 = No, 1 = Yes)
3. Pclass: Ticket class (1 = 1st, 2 = 2nd, 3 = 3rd)
4. Name: Passenger's name
5. Sex: Sex of a passenger
6. Age: Age in years
7. sibsp: Number of siblings/spouses aboard the Titanic
8. Parch: Number of parents/children aboard the Titanic
9. Ticket: Ticket number
10. Fare: Passenger fare
11. Cabin: Cabin number
12. Embarked : Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

The column named “Survived” is the target column/label or the dependent variable that I am going to predict and the remaining columns are the features or independent variables that will help in the model prediction.

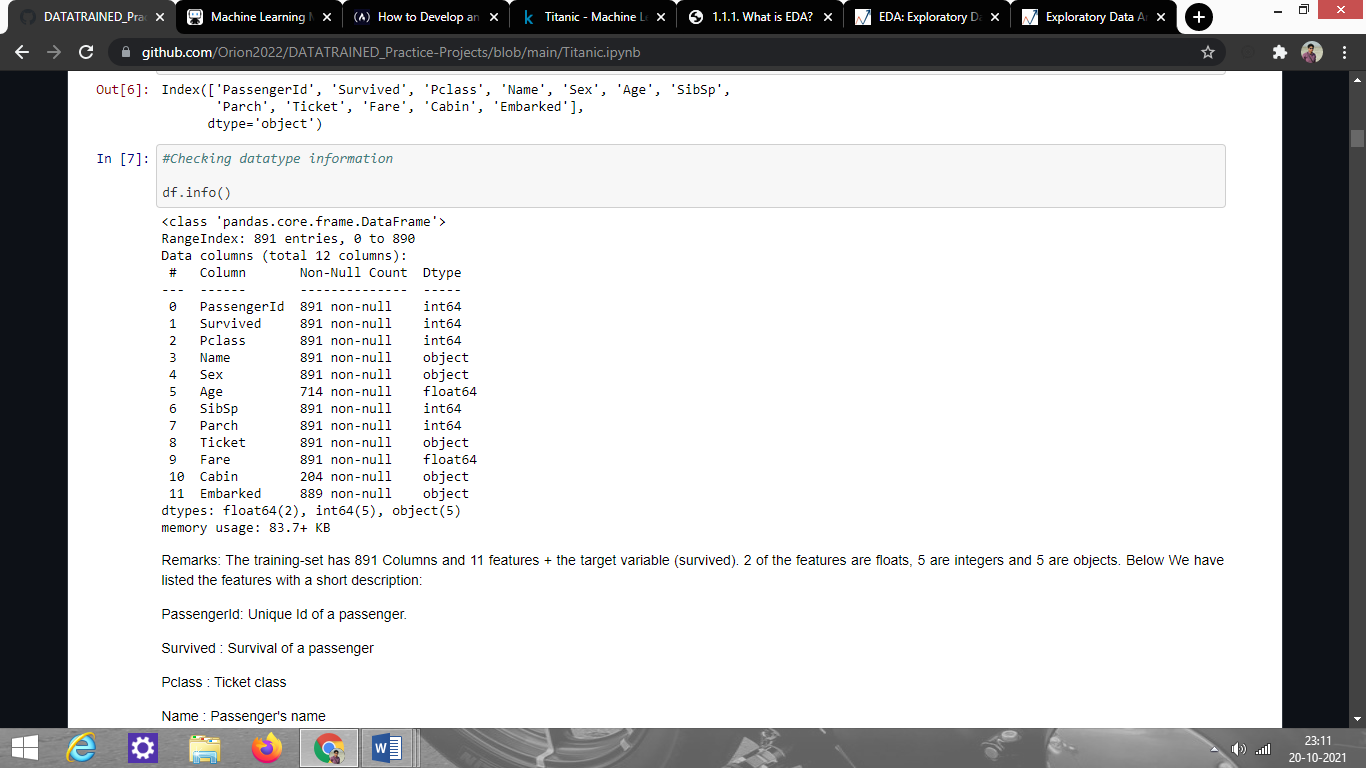
**Exploratory Data Analysis (EDA)**

Exploratory Data Analysis or EDA is considered the most important step in any Data Analysis or Data Science project. It is a process of examining the data and extracting insights from the data. It is generally classified into two types: one is the graphical analysis and the other is non-graphical analysis.

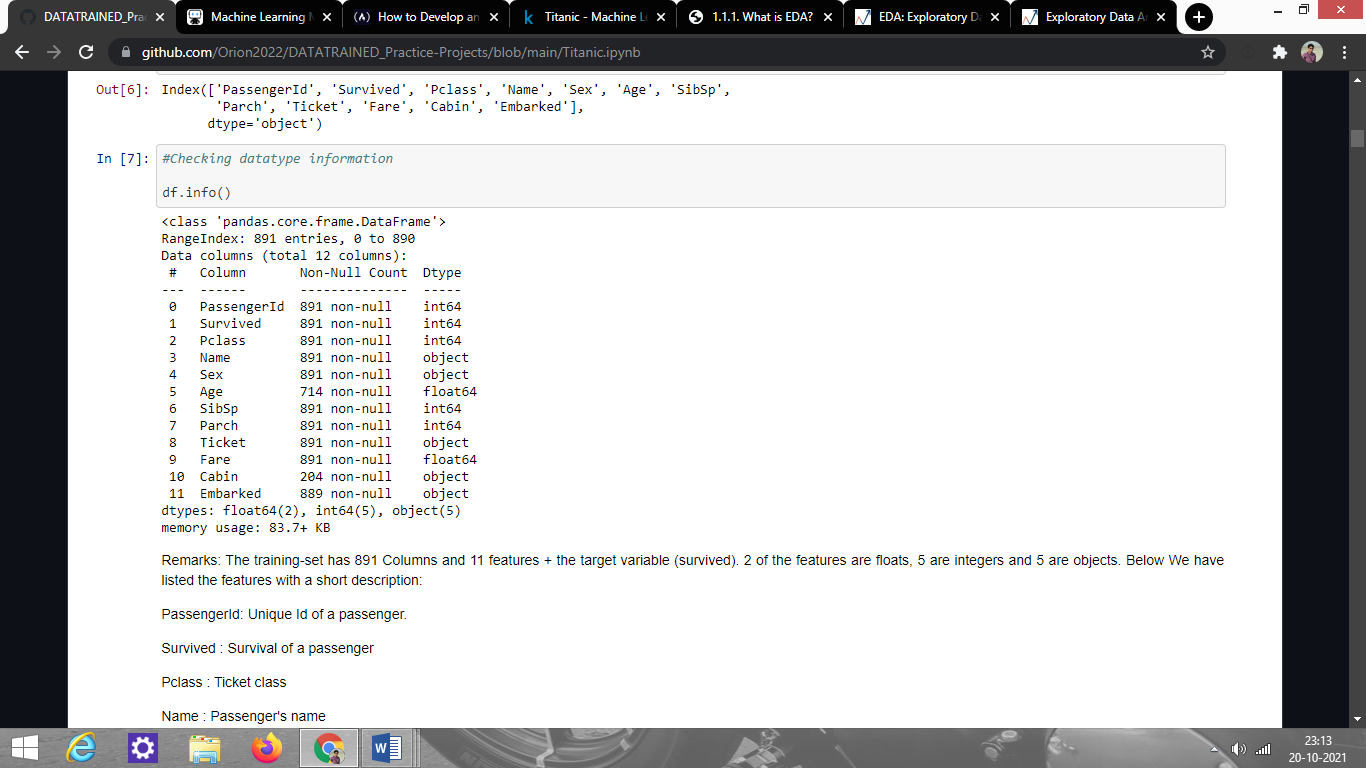
Non-graphical EDA involves generating summary statistics for numerical data in the dataset, checking missing values, and many more that I’ll be showing later. However, graphical EDA involves creating various graphical representations such as barplot, pie charts, etc, to have a better understanding of the data.

* ***Non-Graphical EDA***:

To begin with, let's check the data type of the columns which can be done with the code shown below:

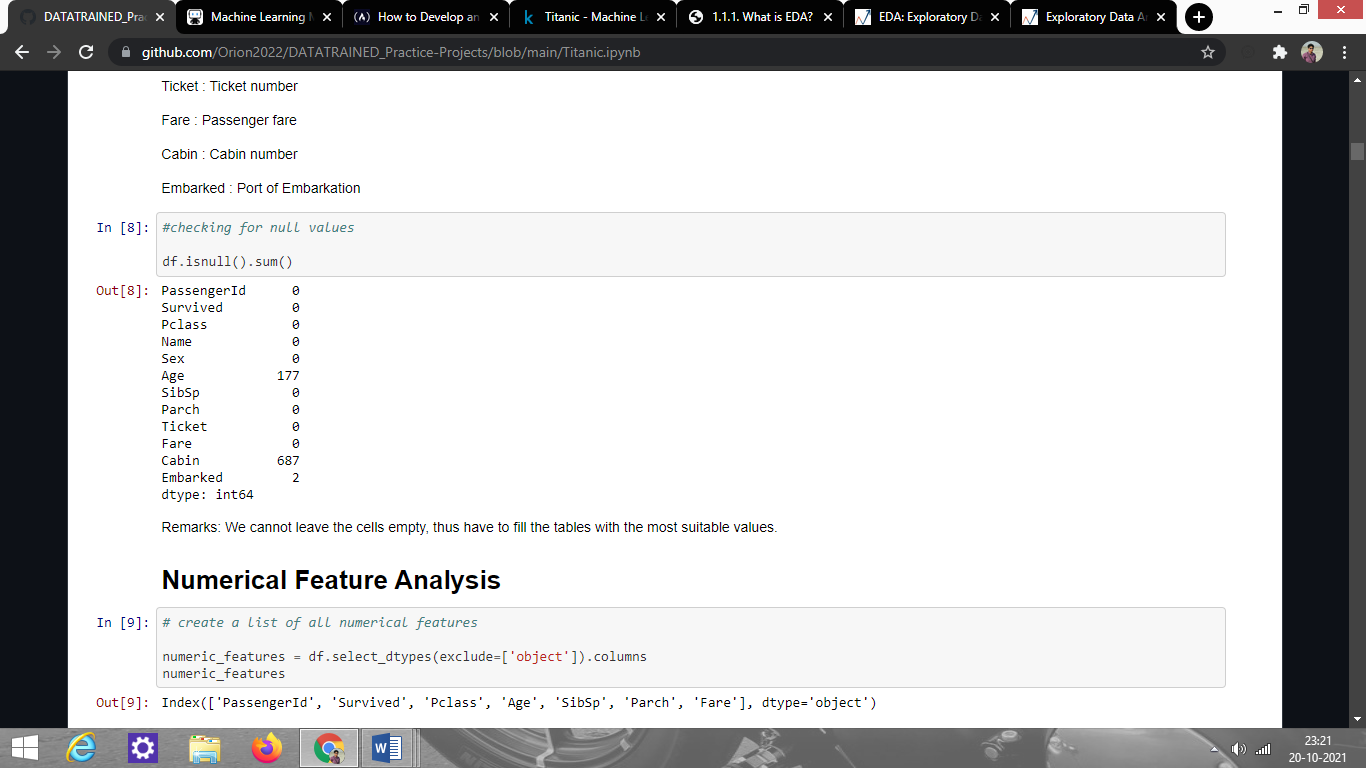


Running this code, we get the output as:

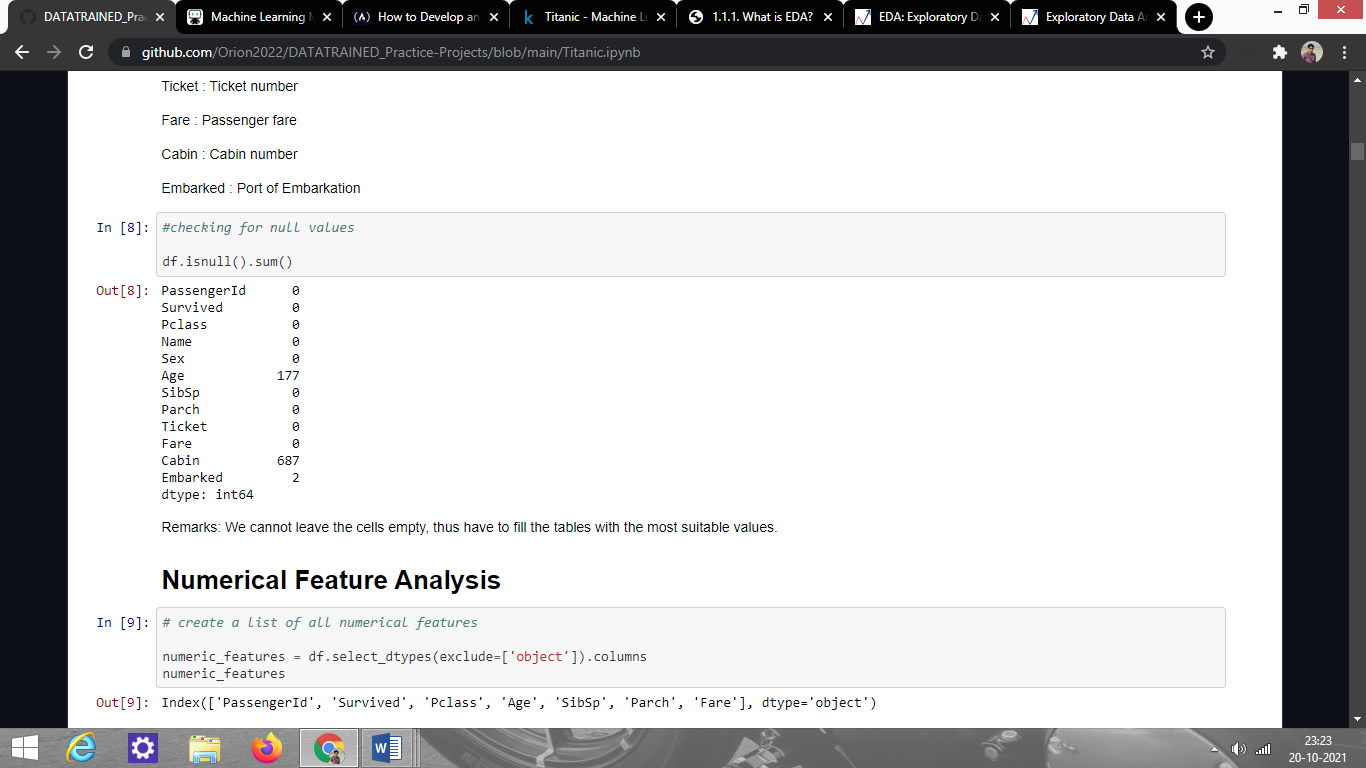


Here, we can see that the training set has 891 Columns and 11 features + the target variable (Survived). 2 of the features are floats, 5 are integers and 5 are objects.

Next, I will check if the dataset has any missing or null values using the piece of code shown below:

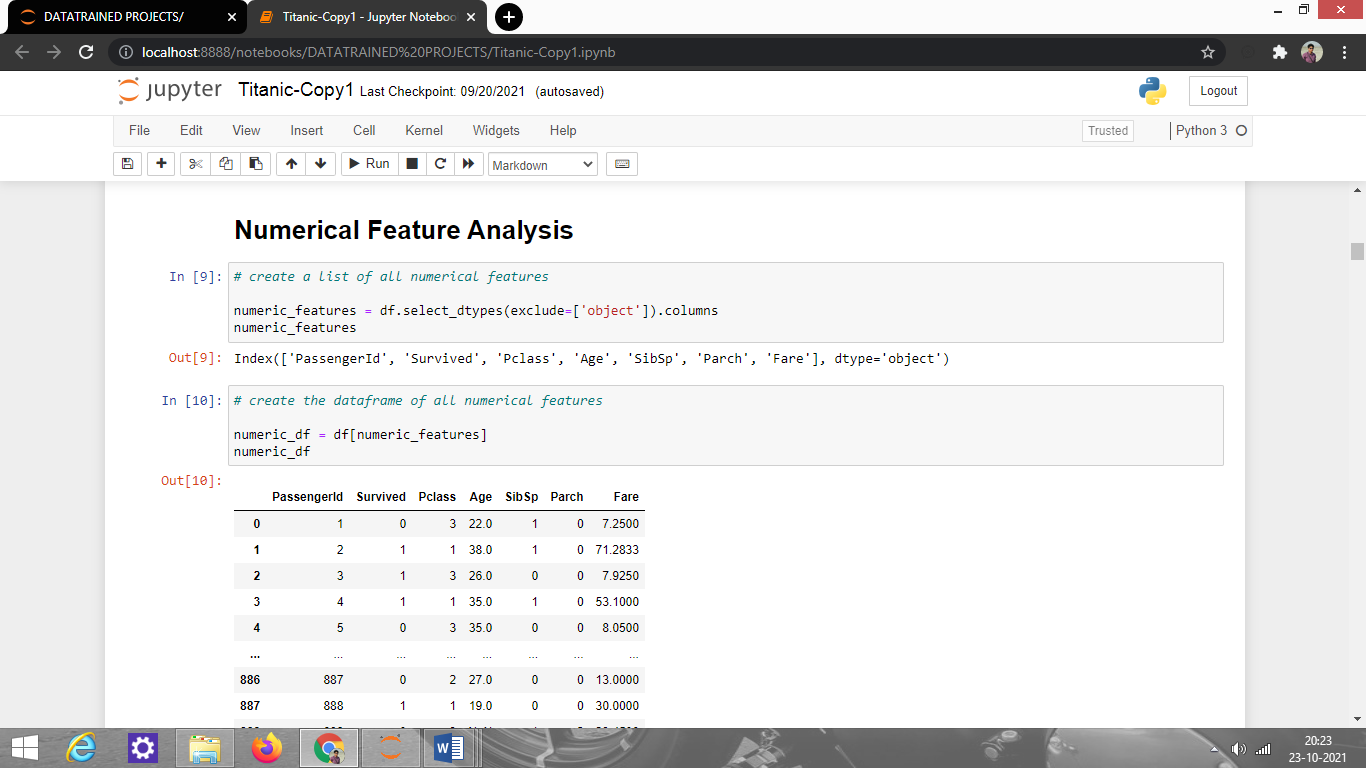


The output is-

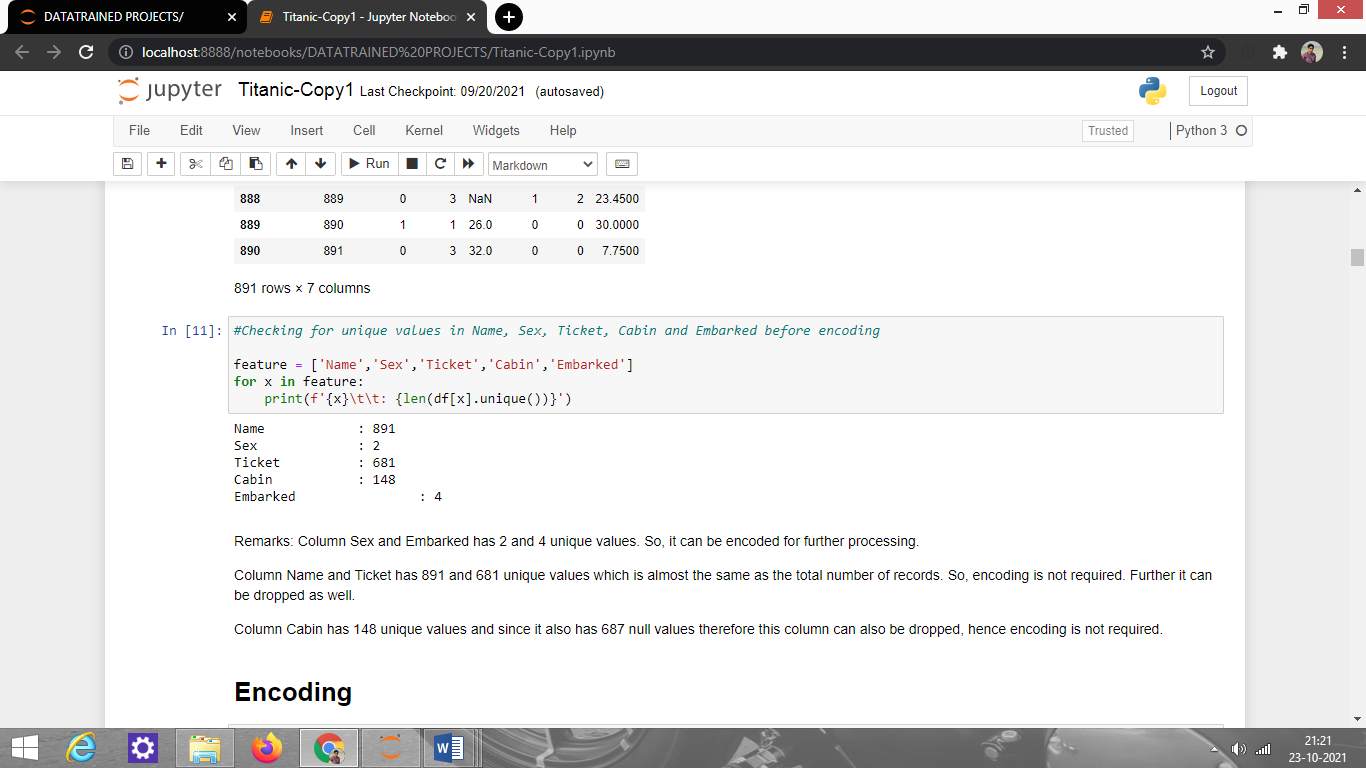


From the above output, I find that there are null values present in the dataset and I cannot leave the cells empty, thus have to fill the tables with the most suitable values.

**Numerical Data Analysis**: As the dataset is a combination of both numerical and categorical features, I am going to make a dataframe consisting of the numerical features for further analysis. The codes are shown below-



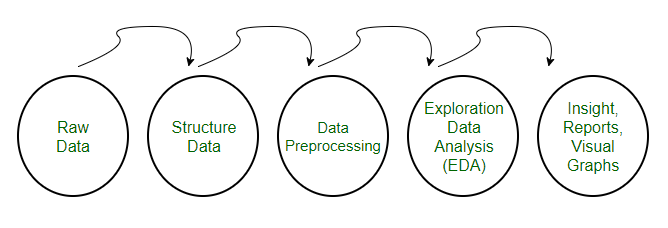
Next, I have checked the unique values present in the categorical columns using the code-



The insights that can be drawn from the above line of codes:

1. Sex and Embarked have 2 and 4 unique values. So, it needs to be encoded for further processing.
2. Name and Ticket have 891 and 681 unique values which are almost the same as the total number of records. So, encoding is not required. Further, these features can be dropped as well.
3. Cabin has 148 unique values and since it also has 687 null values, therefore, this column can also be dropped, hence encoding is not required.

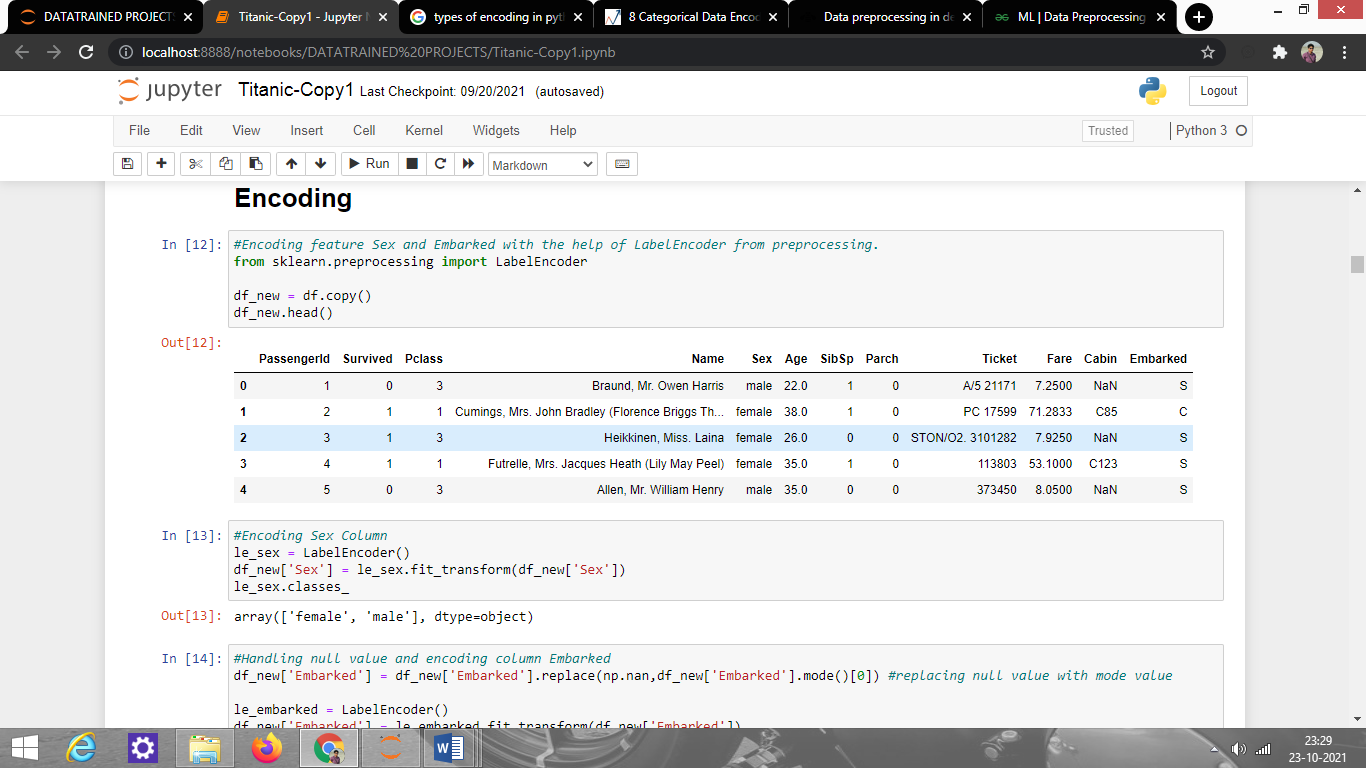
***Pre-processing of the data***: The data that I have is in raw format and to feed it to the algorithm to build the model it is necessary to convert the raw data into a clean data set which can be achieved through data pre-processing.

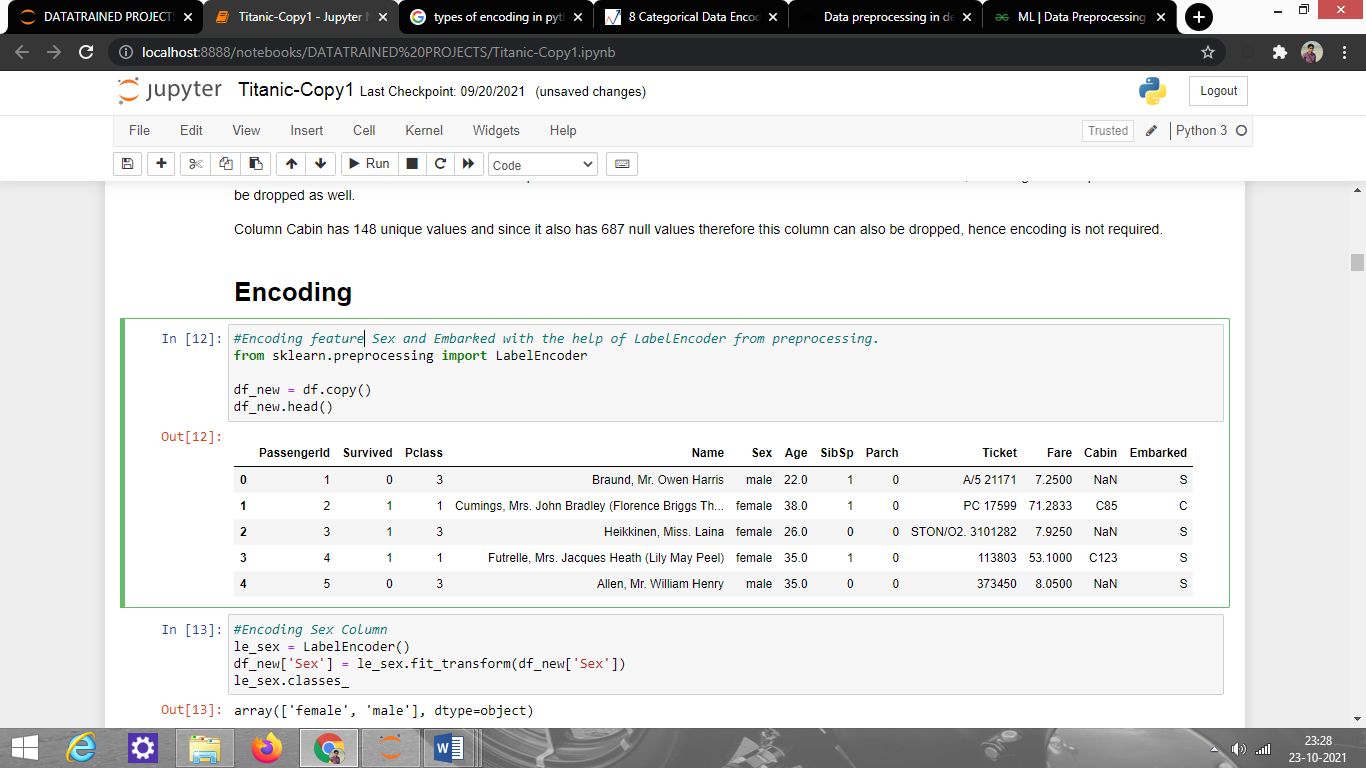


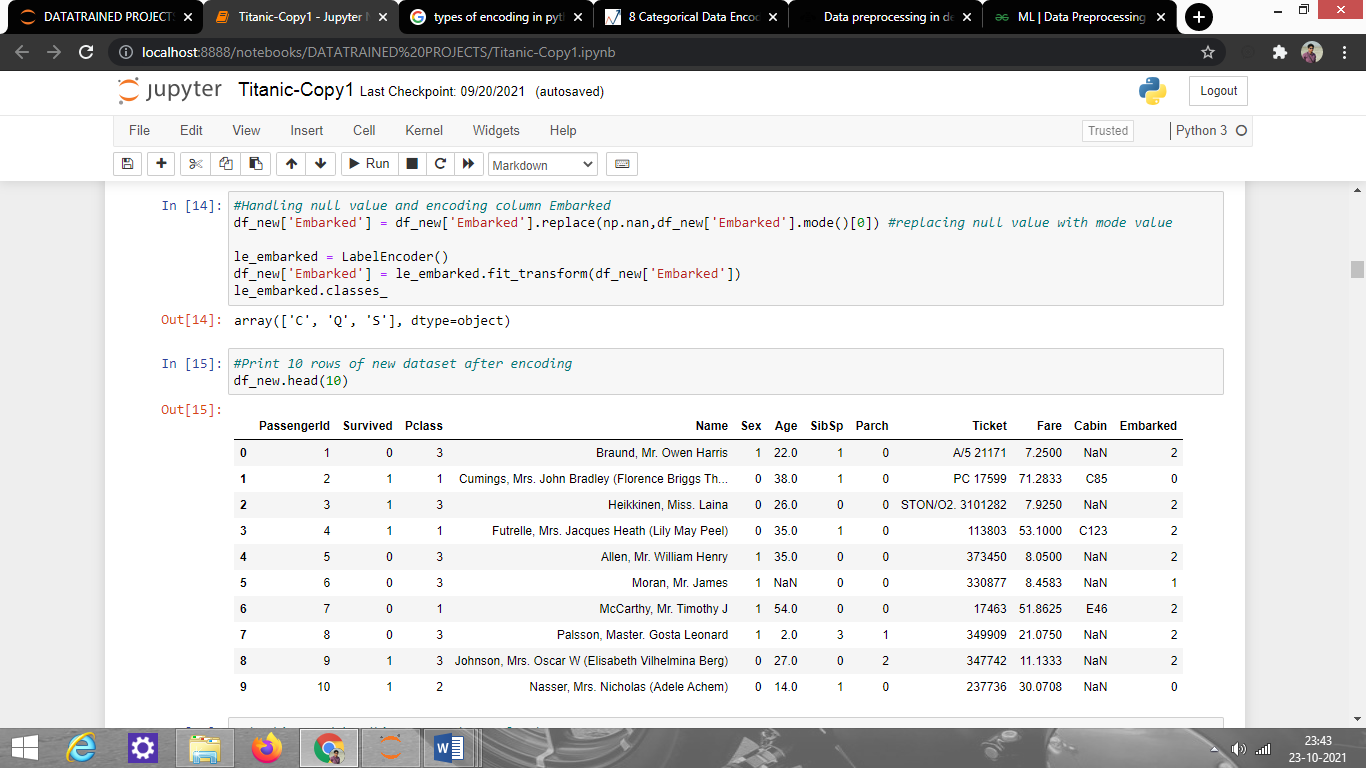
The data has to be in a proper format for achieving better results from the applied model in Machine Learning projects.

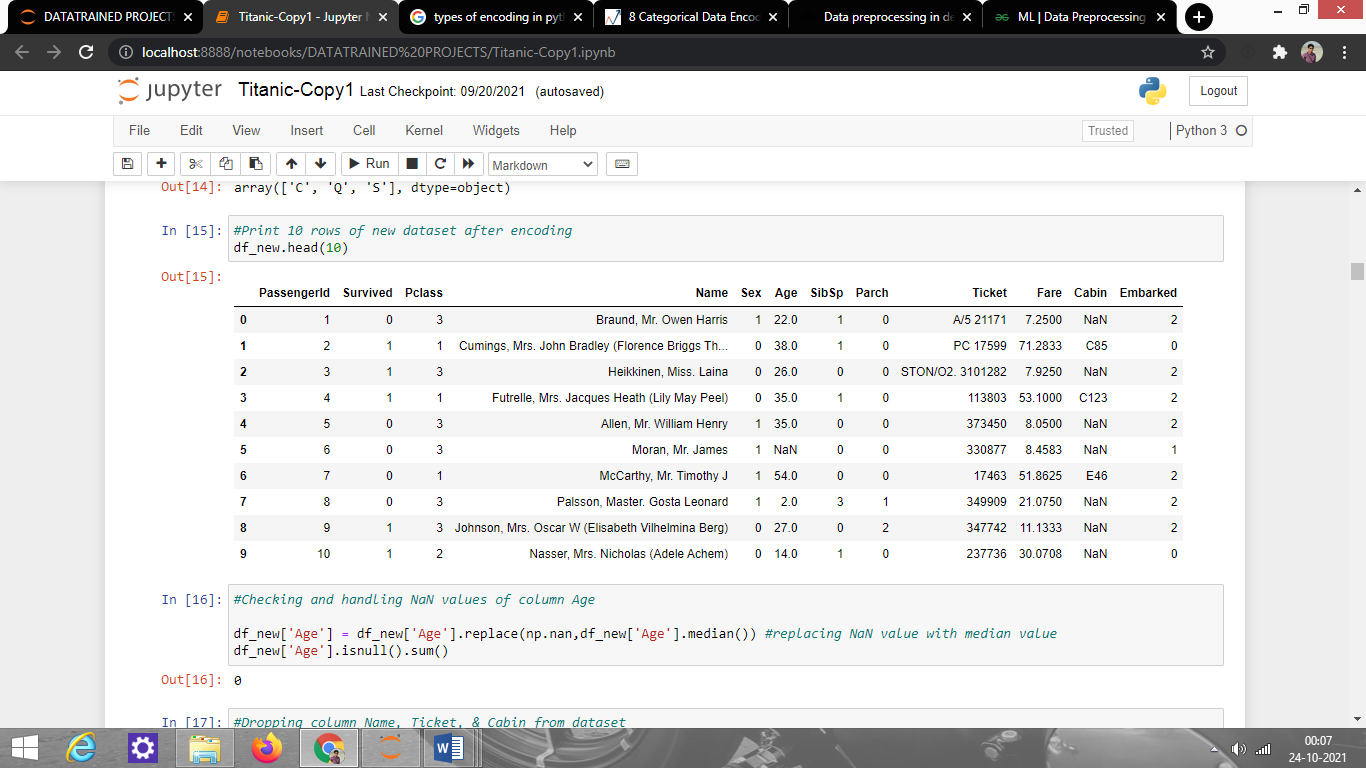
As the Titanic dataset contains object data columns that need to be converted into numeric values to make them understandable to Machine Learning models, I am going to encode those object data. I am using LabelEncoder to encode those data along with that I am also going to drop a few columns that do not help much in model building and fill the missing/null data present with mean and mode values.

Code:

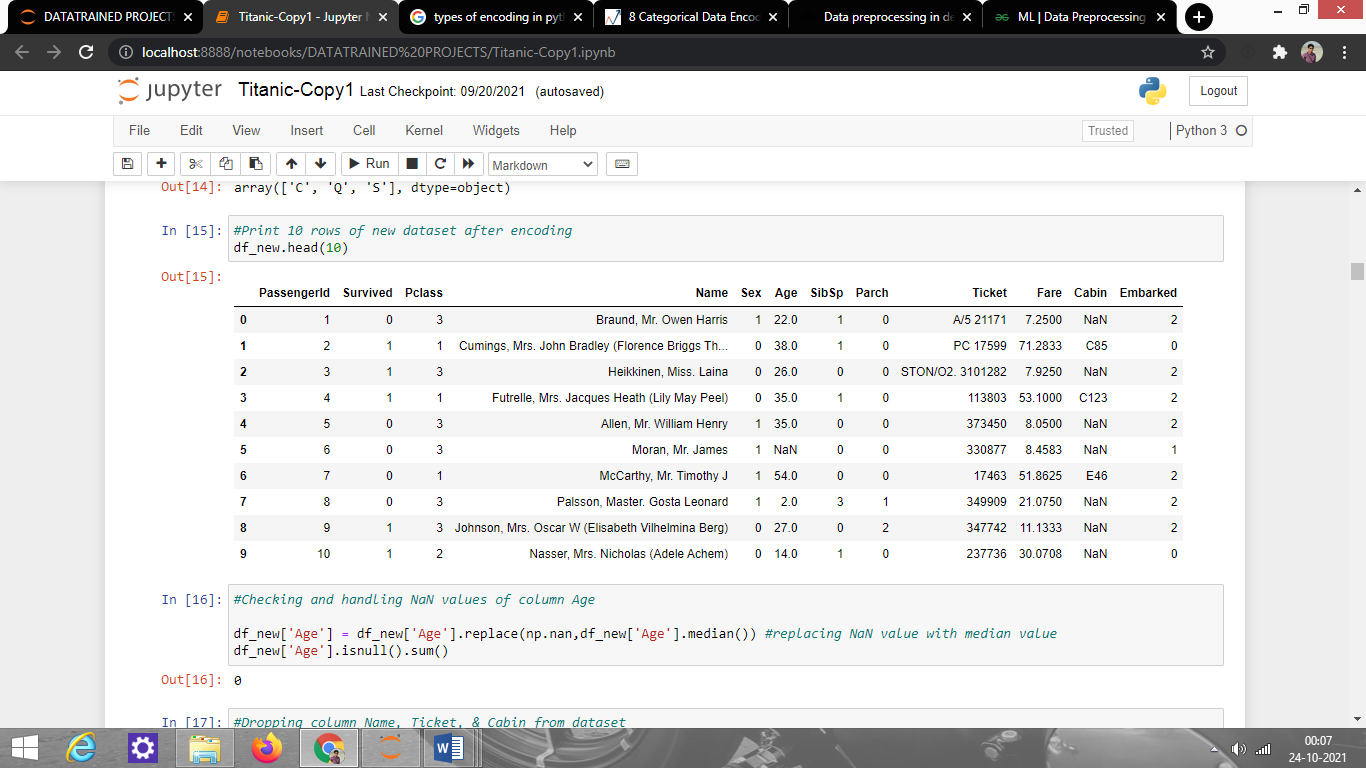




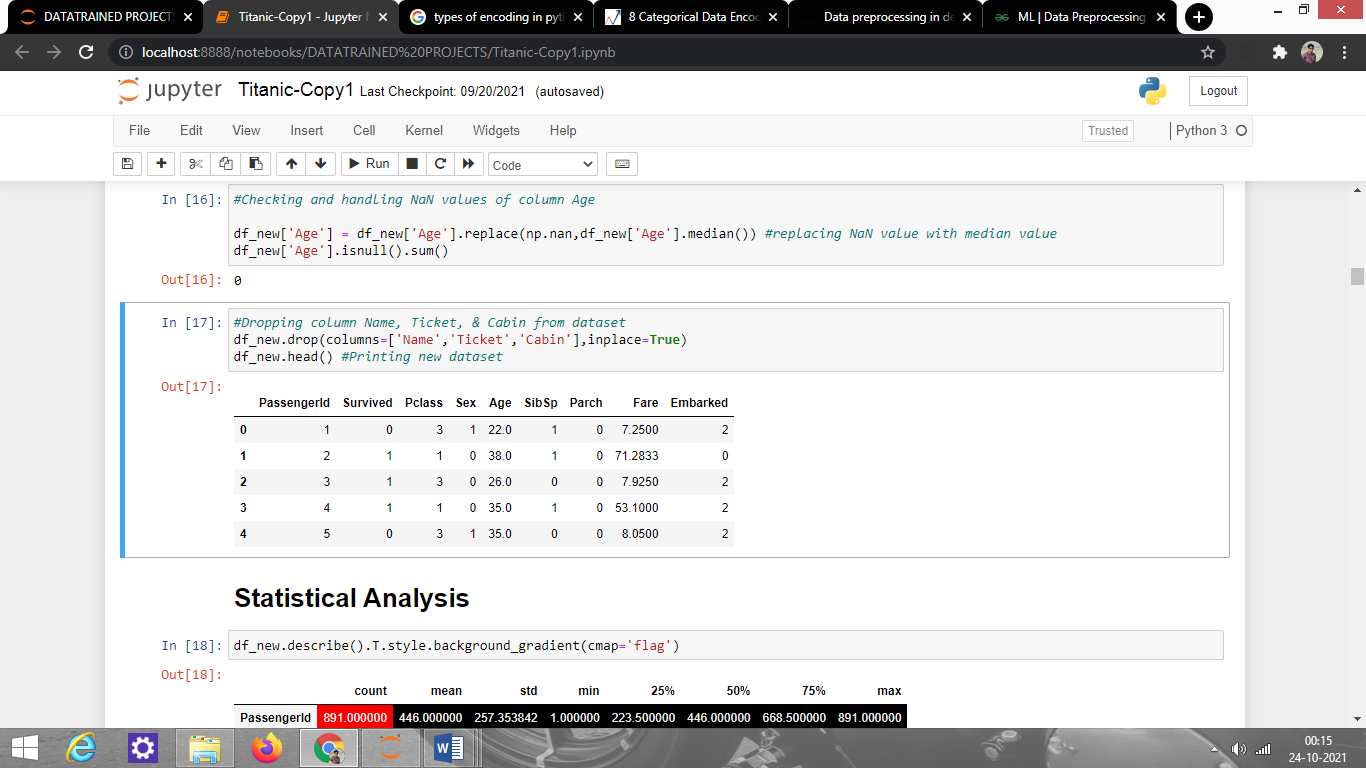




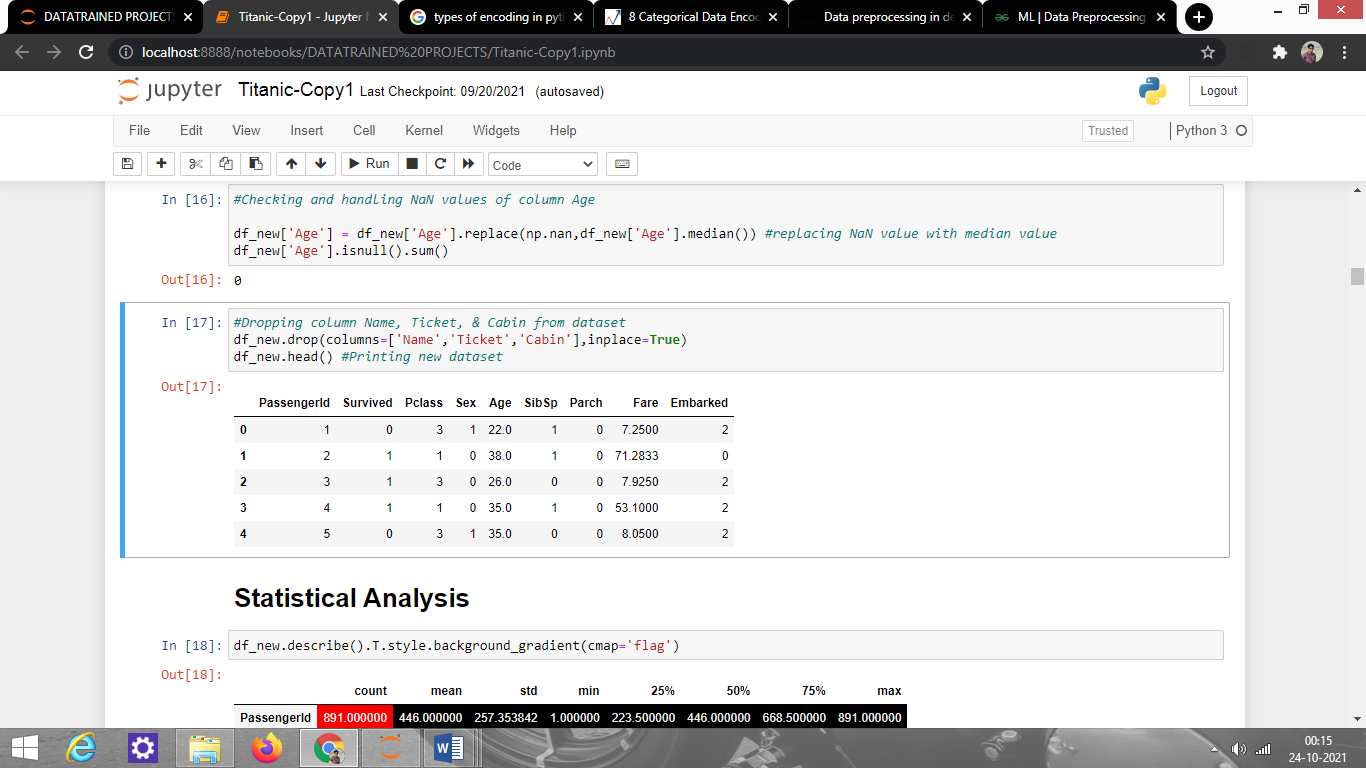
The output:



Code to drop columns:



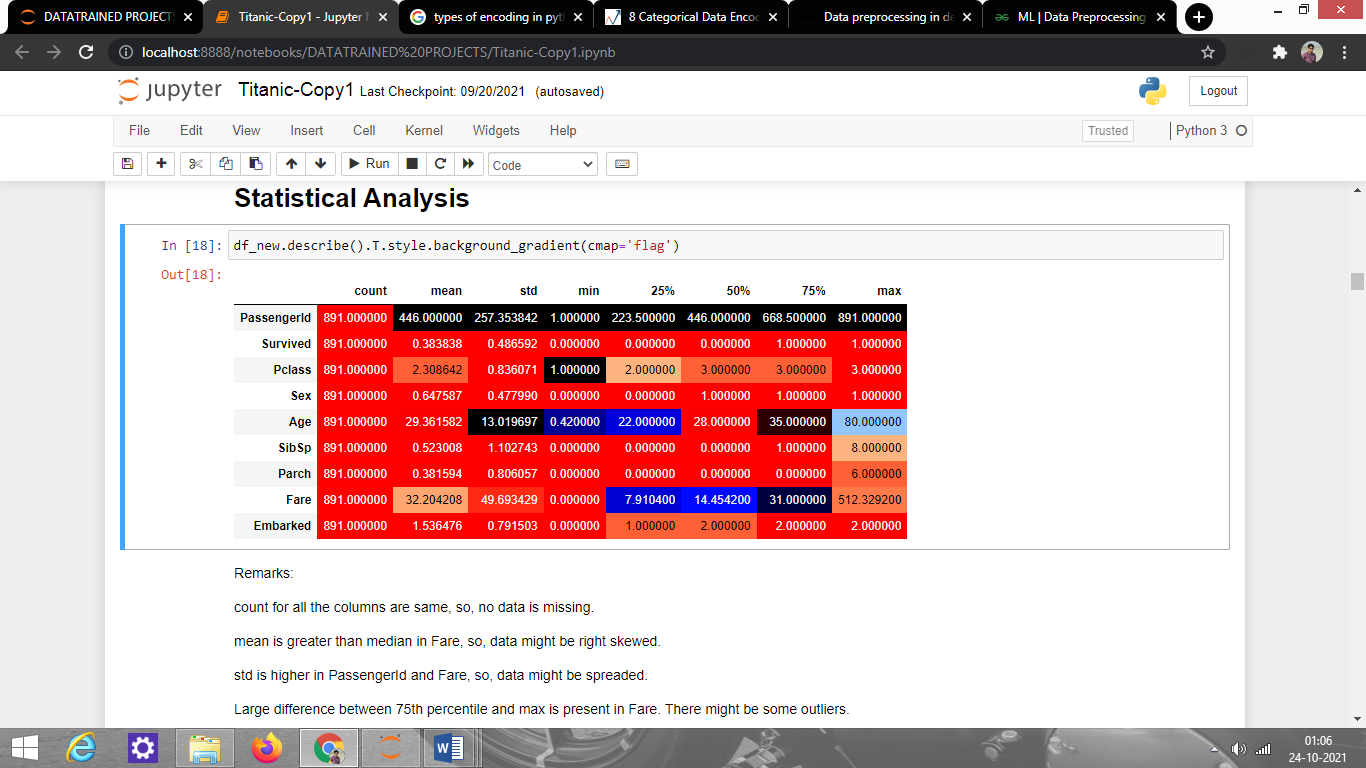
The output:



The above output shows the dataframe after encoding the object data, filling the missing values, and dropping the columns not required for model prediction.

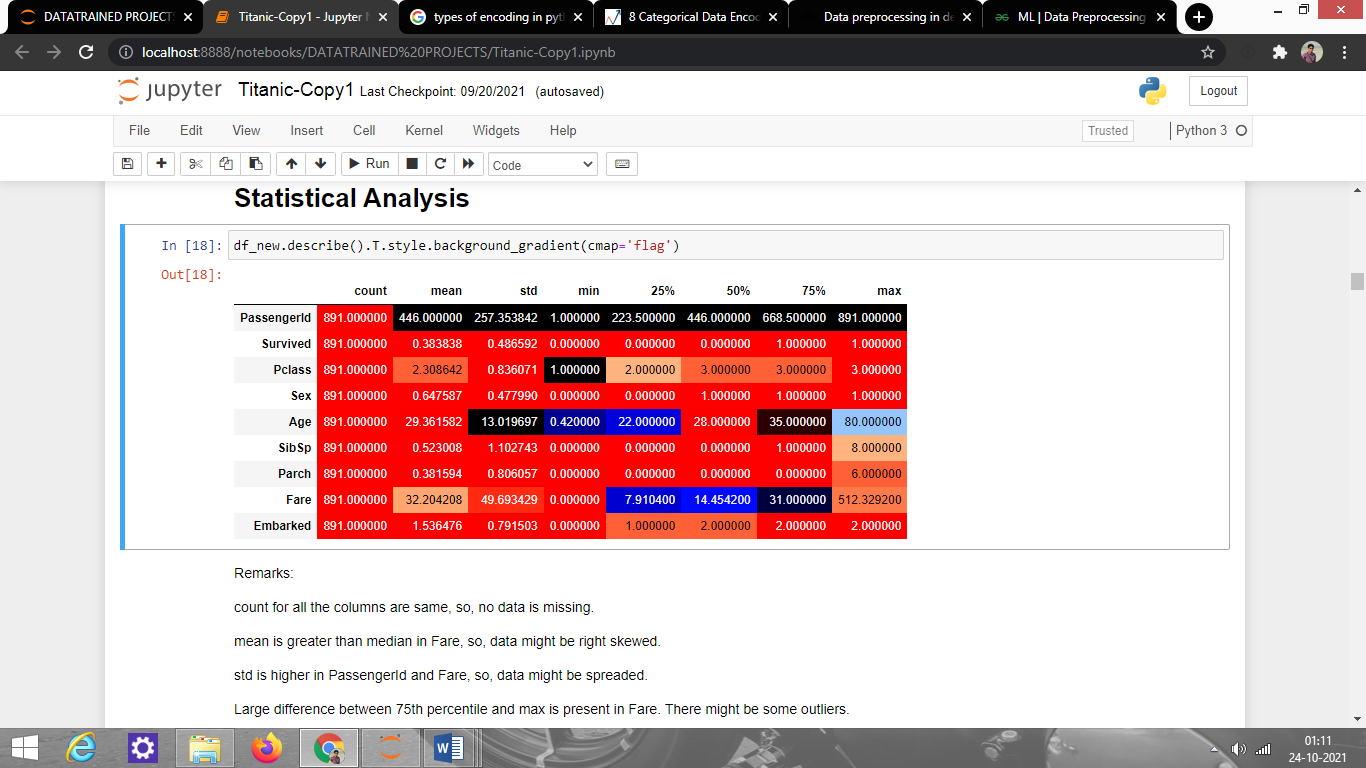
Now, let’s do a Statistical Analysis of the dataset using the ‘describe method’ to take a look at the count value, mean data, standard deviation information, and minimum, maximum, 25% quartile, 50% quartile, and 75% quartile details.

Code:



Once I have used the above line of code the output provided is in transpose format to accommodate all the columns from our dataset in tabular as well as visual format.

The Output:



The insights that can be drawn from the above output are-

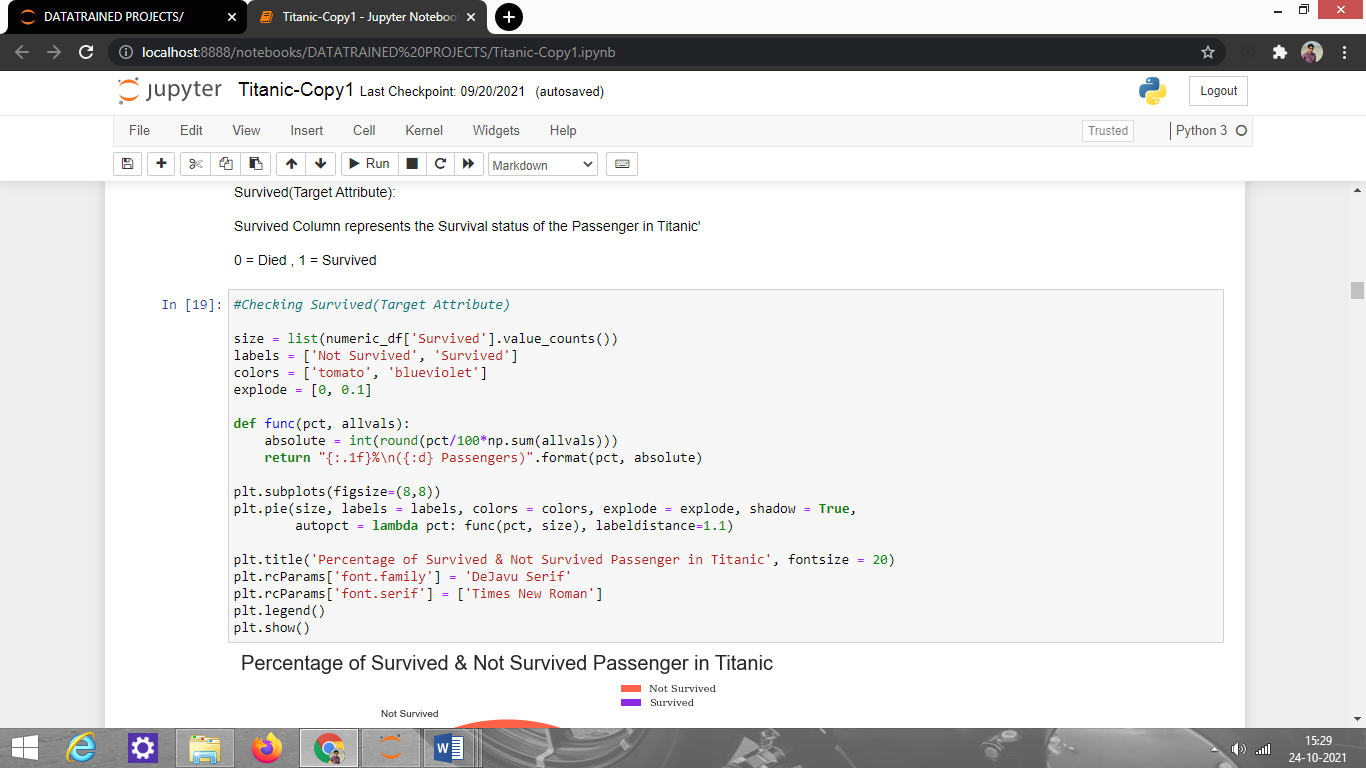
1. count for all the columns are the same, so, no data is missing.
2. mean is greater than the median in Fare, so, data might be right-skewed.
3. std is higher in PassengerId and Fare, so, data might be spreaded.
4. A large difference between the 75th percentile and max is present in Fare. There might be some outliers present.

* ***Graphical EDA***:

Non-graphical methods don’t provide a full picture of the data. Graphical methods are therefore required to understand the data in a much better way. Usage of various visualization techniques allows us to optimize and analyze the features further. Let’s move ahead and list down all the visualization codes and their output that I have indulged in my project analysis.

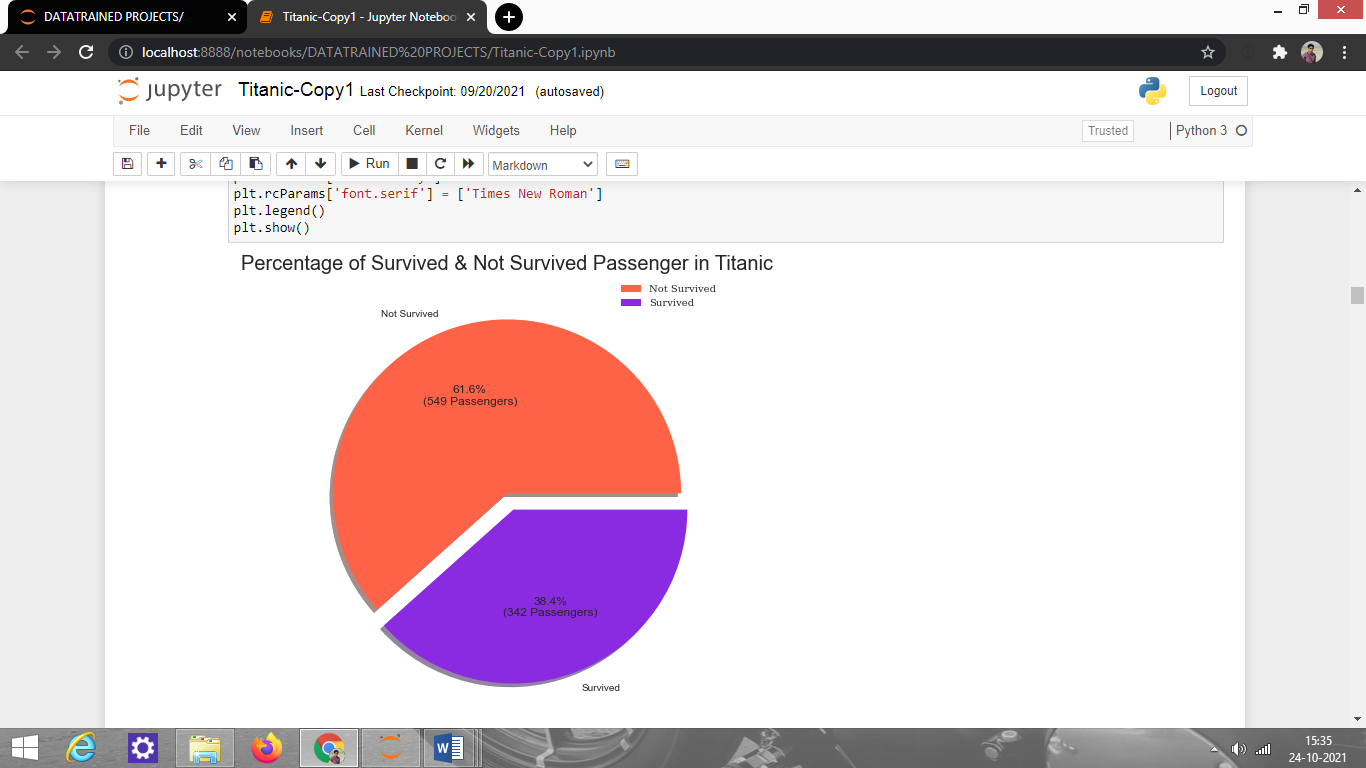
1. Numerical Feature Analysis:

Code:



This is the function I defined to check the target attribute “Survived” to visualize the percentage of passengers that survived and those who did not survive.

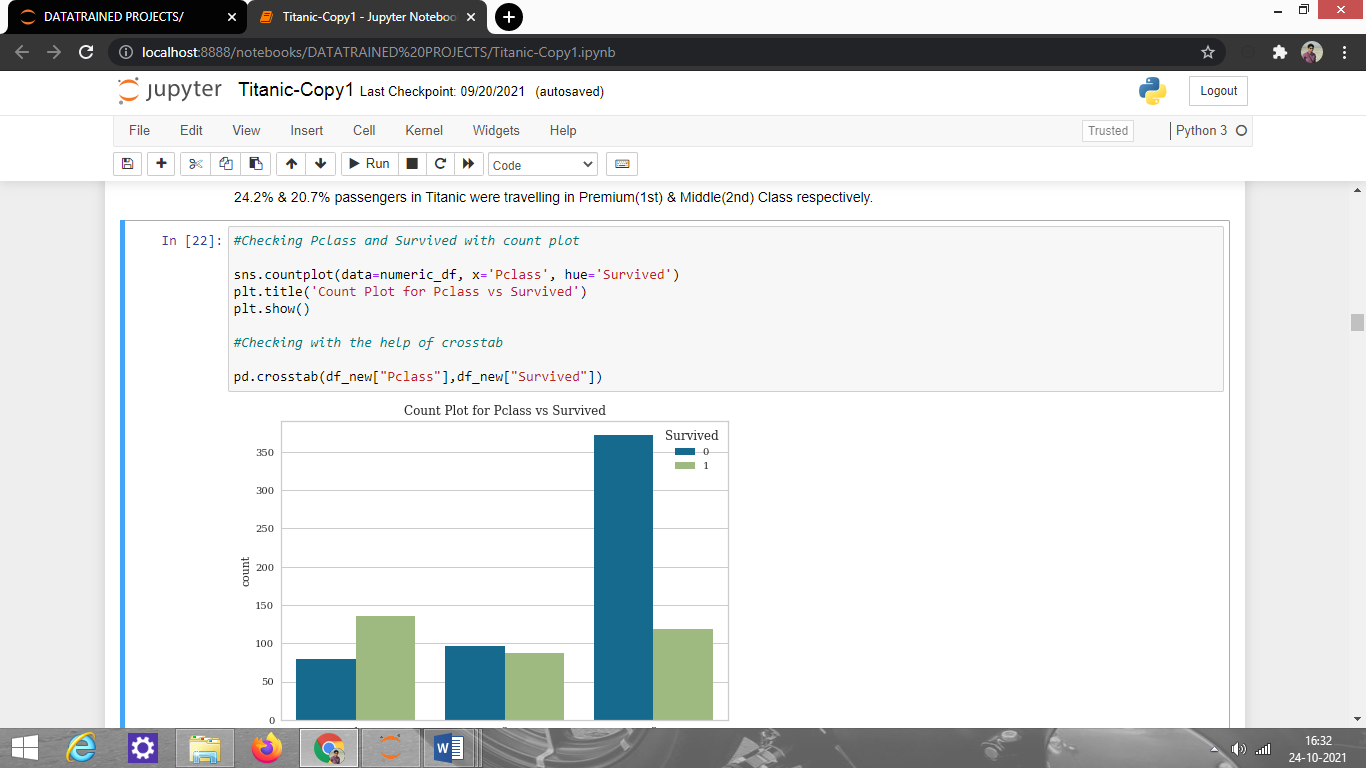
The output:



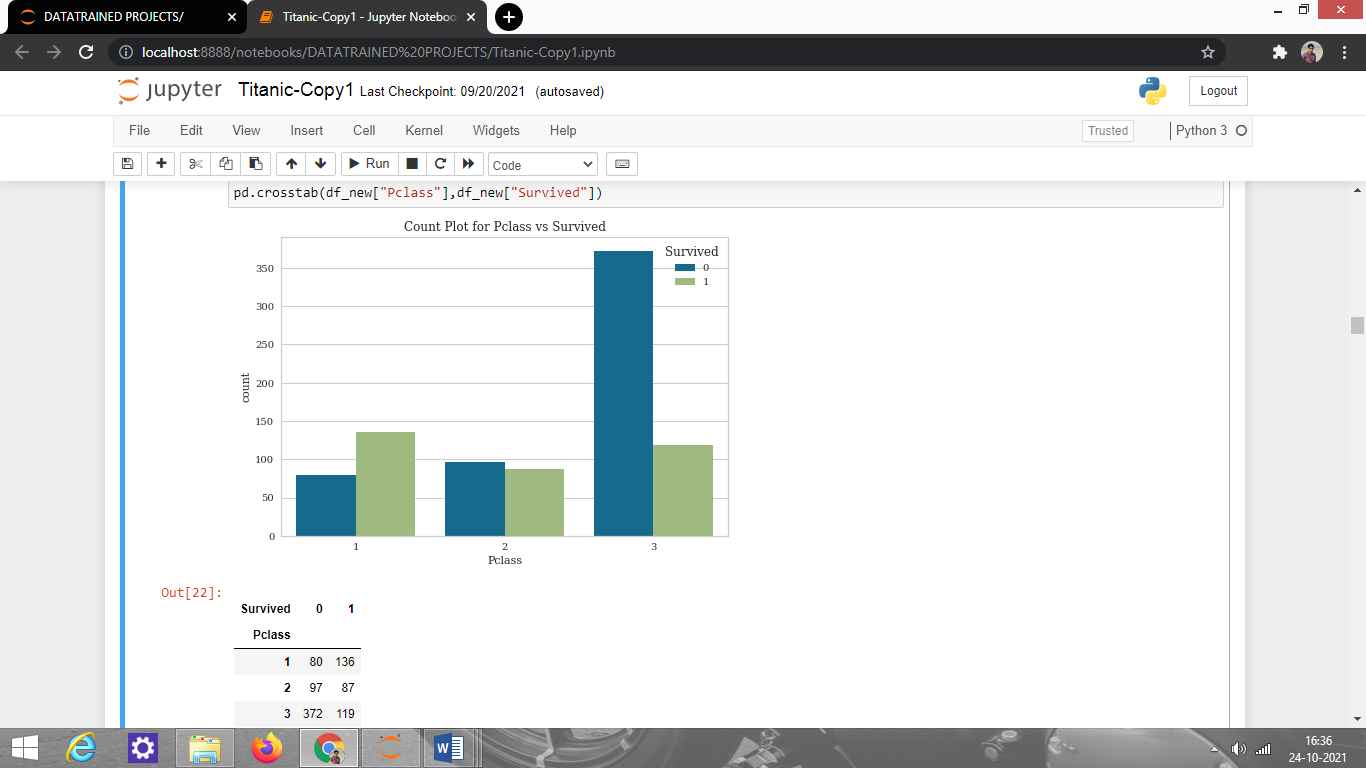
The observations that I found from the above pie-chart are-

1. majority of the passengers (61.6%) in Titanic, were not able to survive.
2. Only 38.4% passengers were able to survive.

Code:

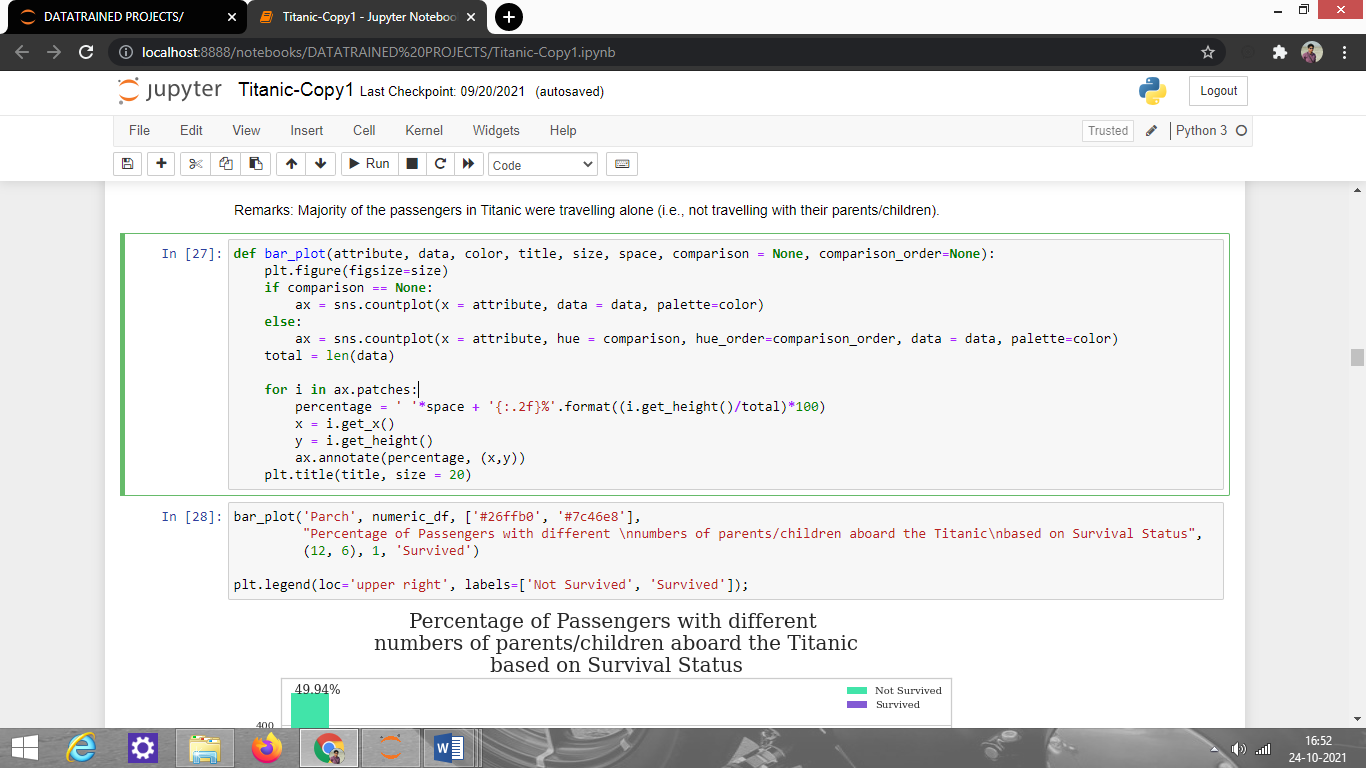


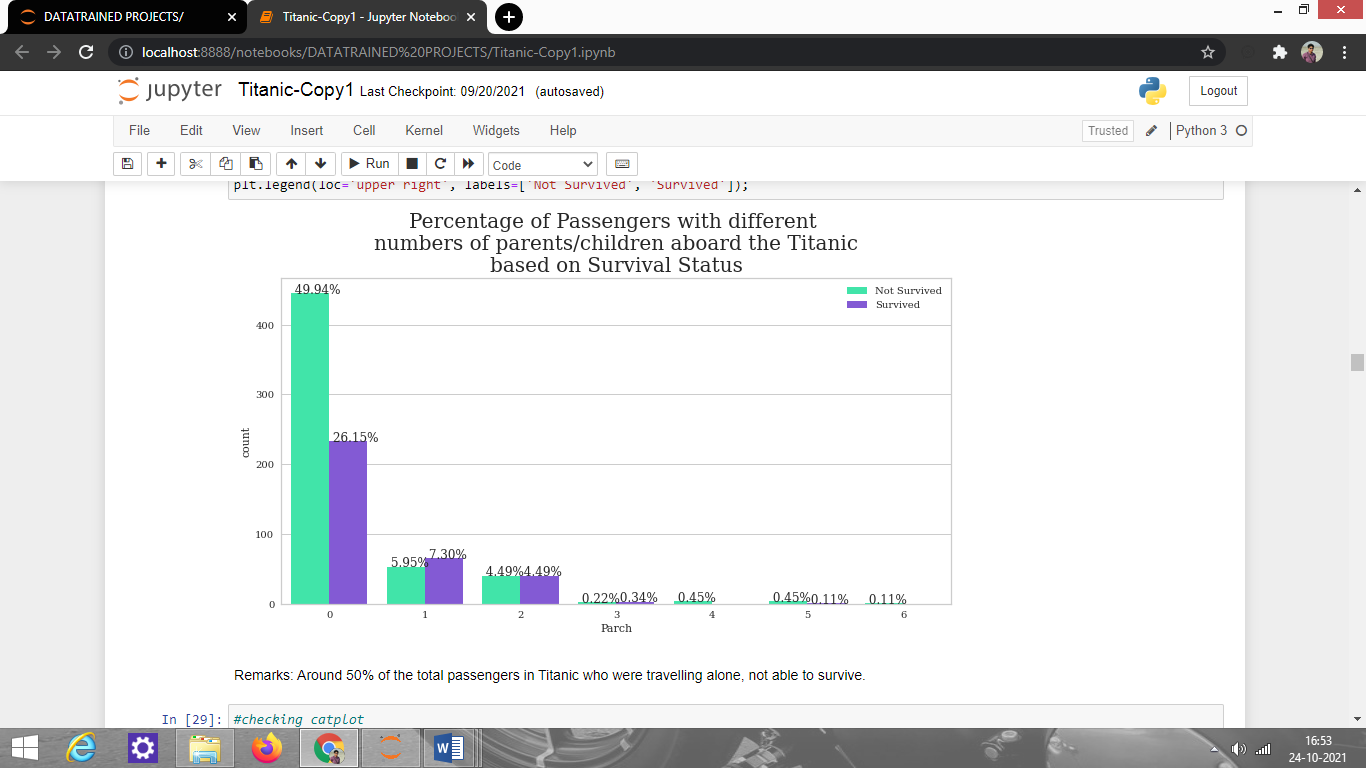
The output:



Observation: In the above plot, I have checked the relationship between the target attribute “Survived” and the feature ‘Pclass’ and noticed that majority of the passengers who were traveling in Economy (3rd) Class of Titanic were not able to survive.

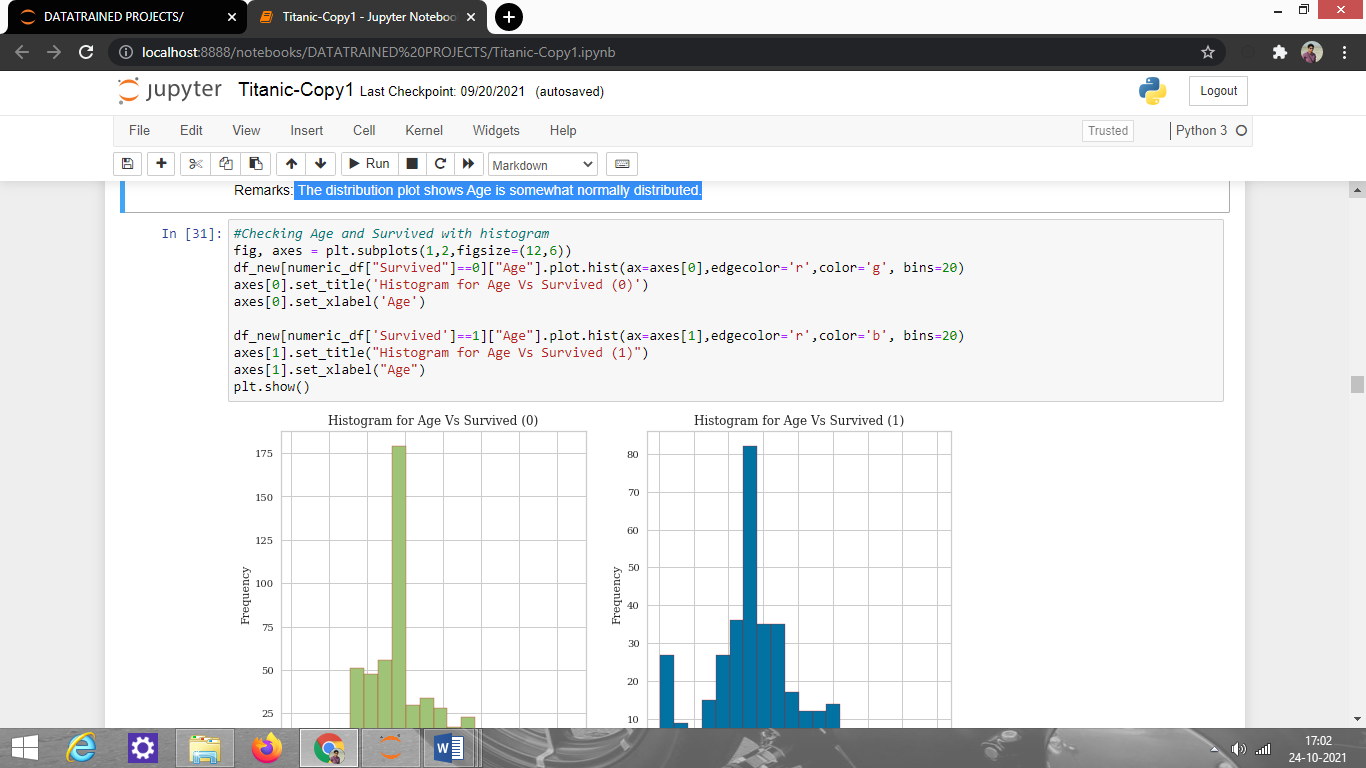
Code and the Output:

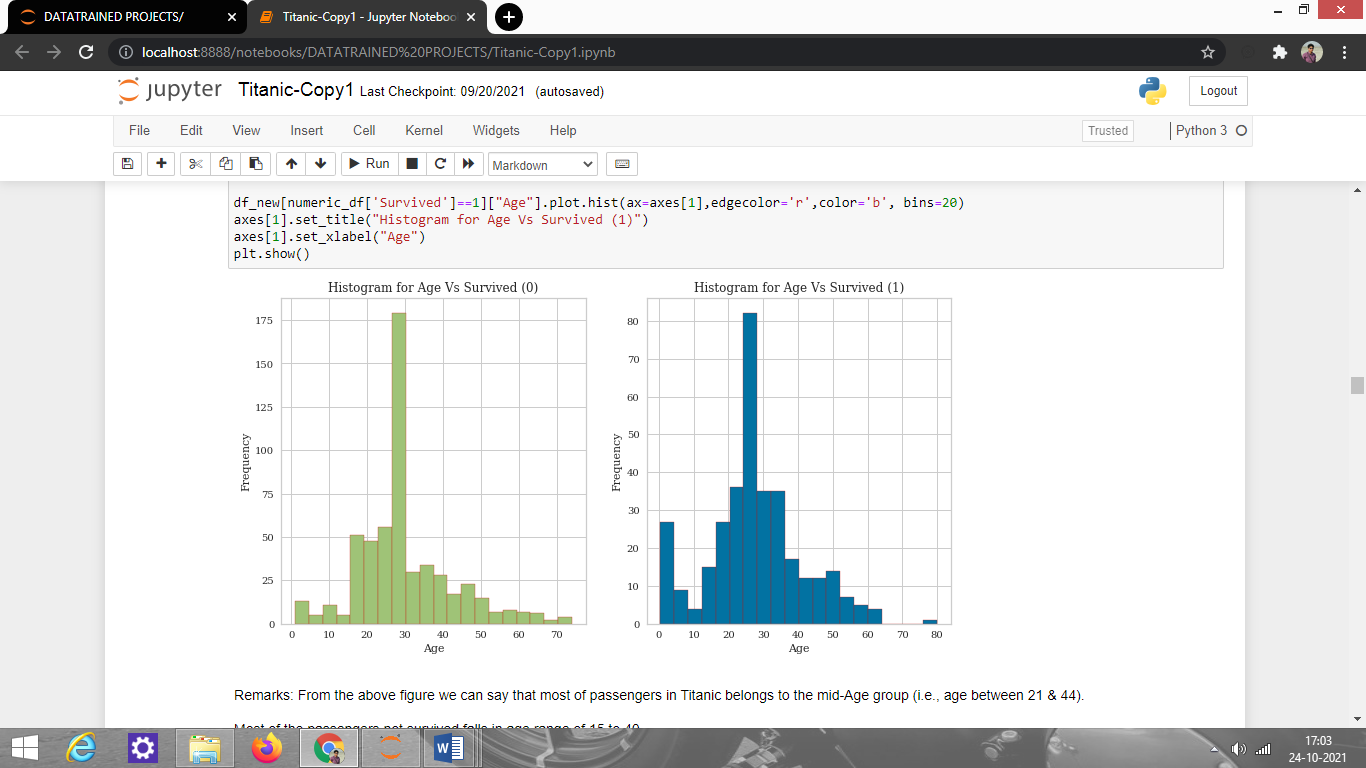




Observation: Around 50% of the total passengers in Titanic who were traveling alone, not able to survive.

Code and the output:



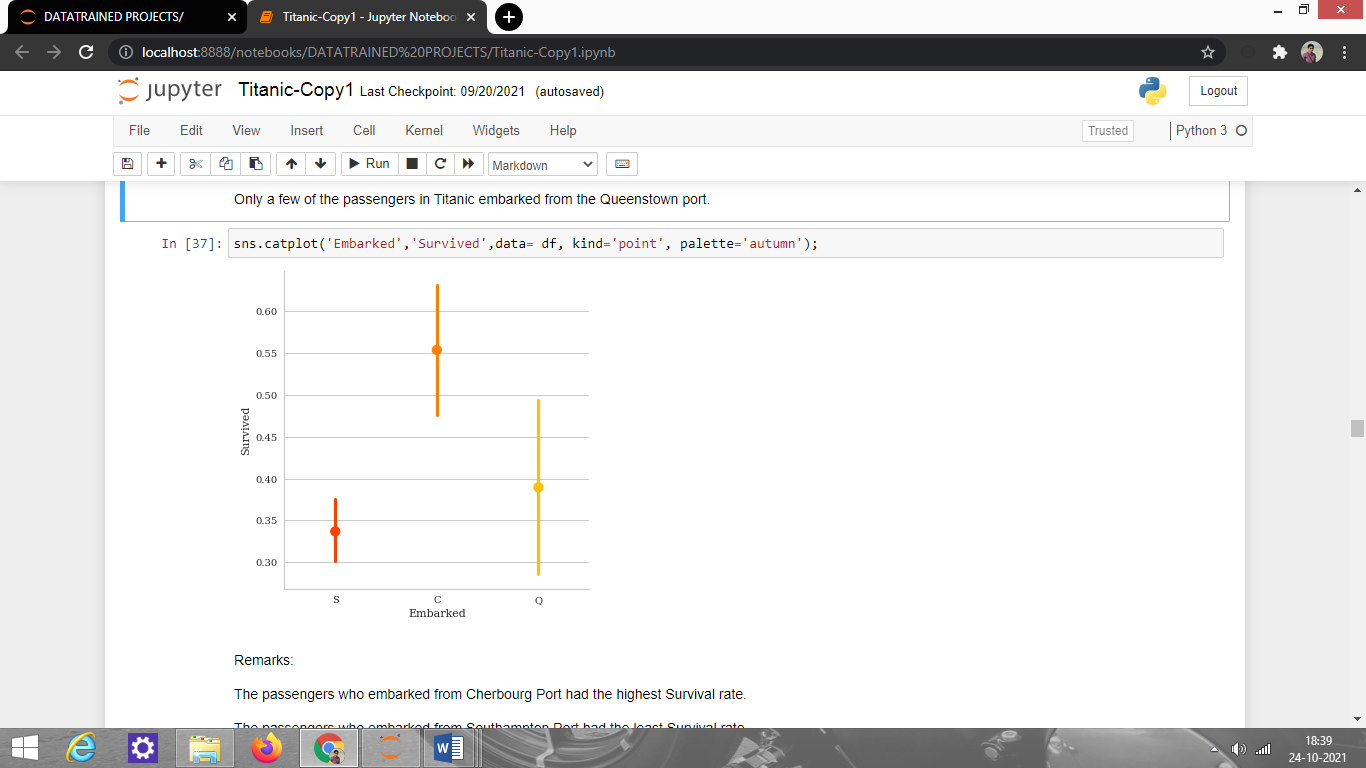


Observation:  From the above figure we can say that most of the passengers in Titanic belong to the mid-Age group (i.e., age between 21 & 44).

1. Most of the passengers not survived falls in the age range of 15 to 40.
2. Most of the passengers survived falls in the age range of 20 to 35.

[2] **Categorical feature Analysis:**

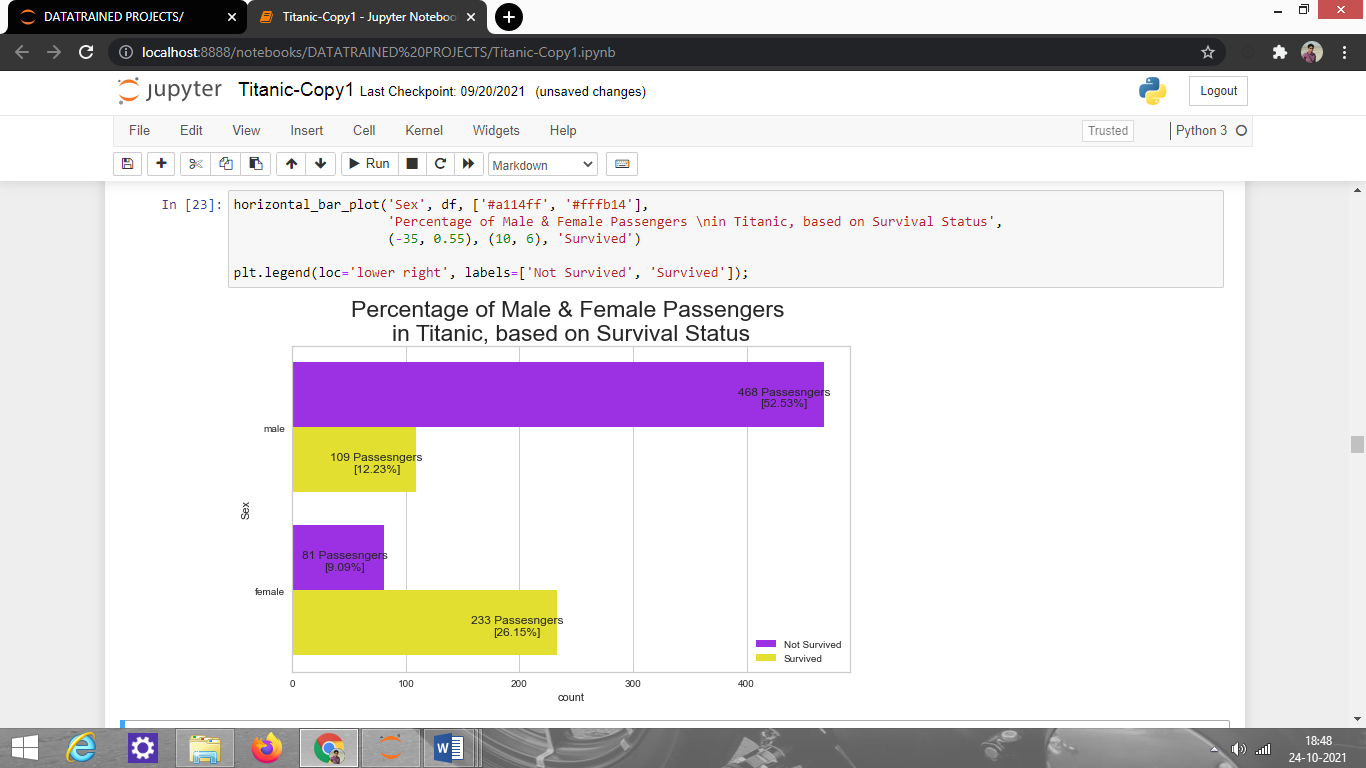
Code and the Output:



Observation:

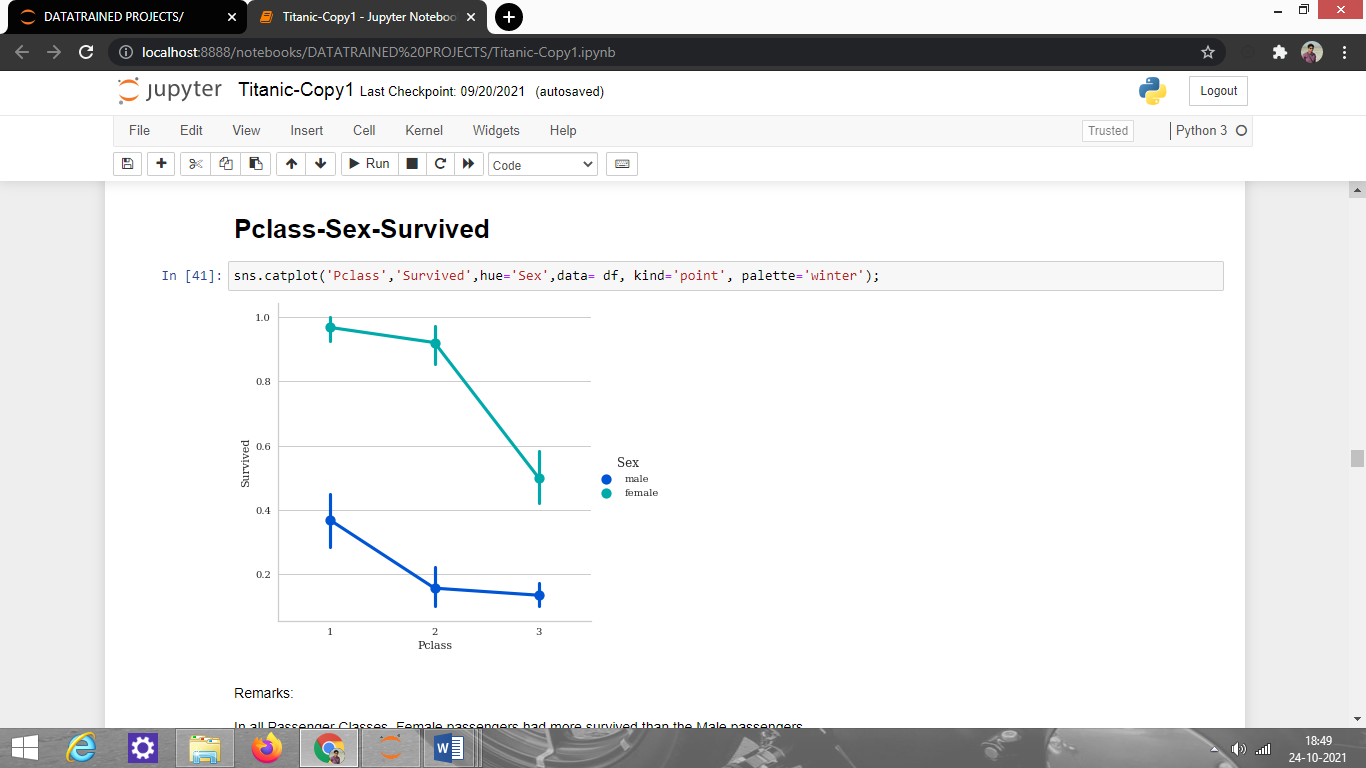
1. The passengers who embarked from Cherbourg Port had the highest survival rate.
2. The passengers who embarked from Southampton Port had the least Survival rate..

Code and the output:



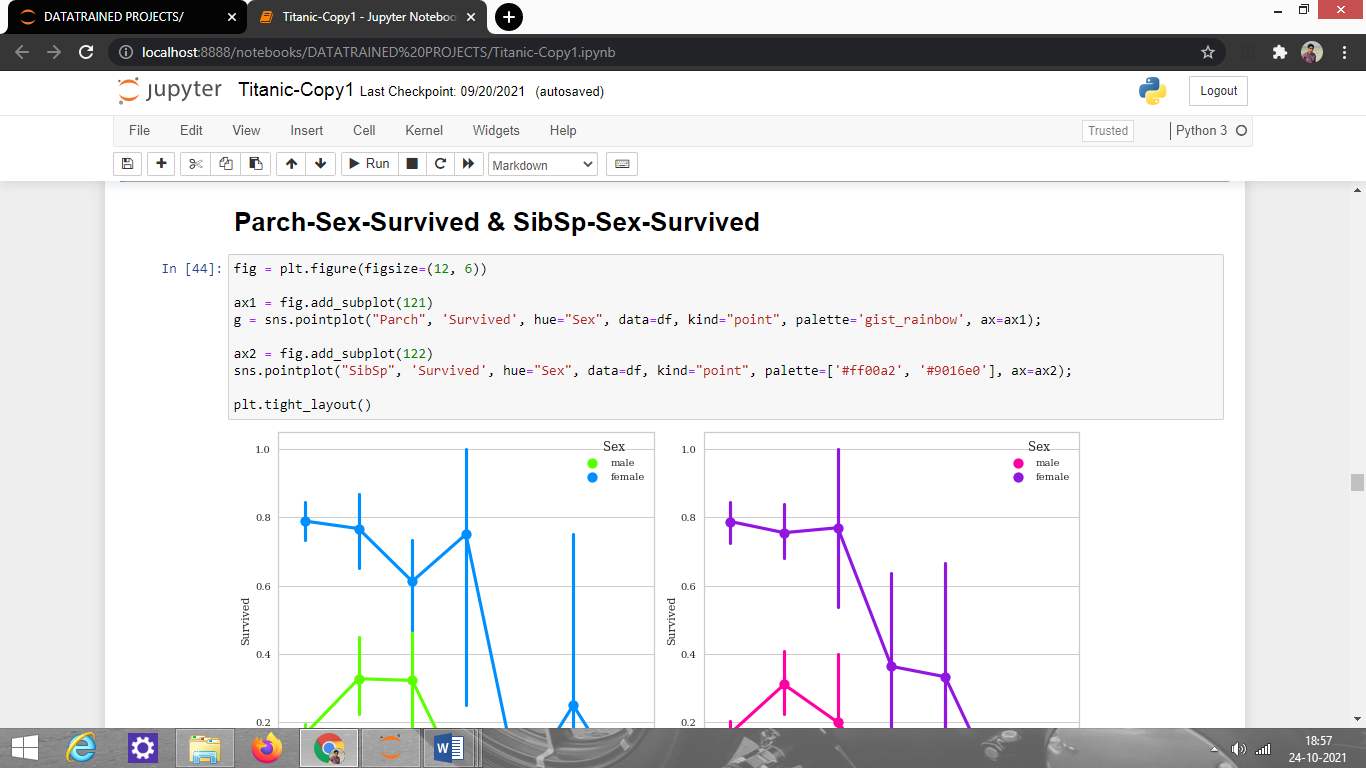
Observation: 52.53% of the total passenger in Titanic who were not able to survive was Male.

Code and the output:



Observation: In all Passenger Classes, Female passengers had more survived than Male passengers.

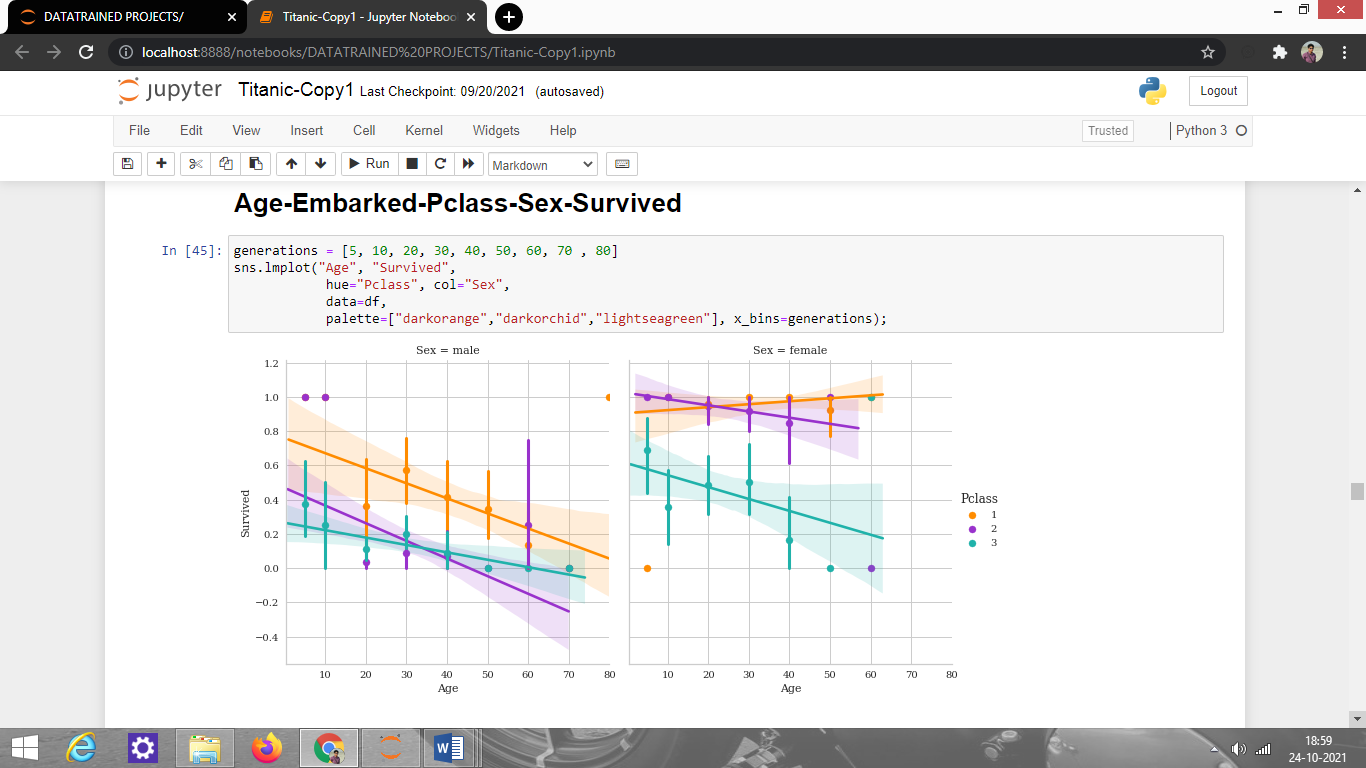
Code and the output:





Observation: The Female passengers who were traveling alone in Titanic had the highest Survival Rate compared with the other Male and Female passengers who were traveling alone as well as with their families.

Code and the output:



Observation: As the Age Increases, passenger's Survival Rate also increases for all kinds of passengers except for the Female passengers of the middle(2nd) Class.

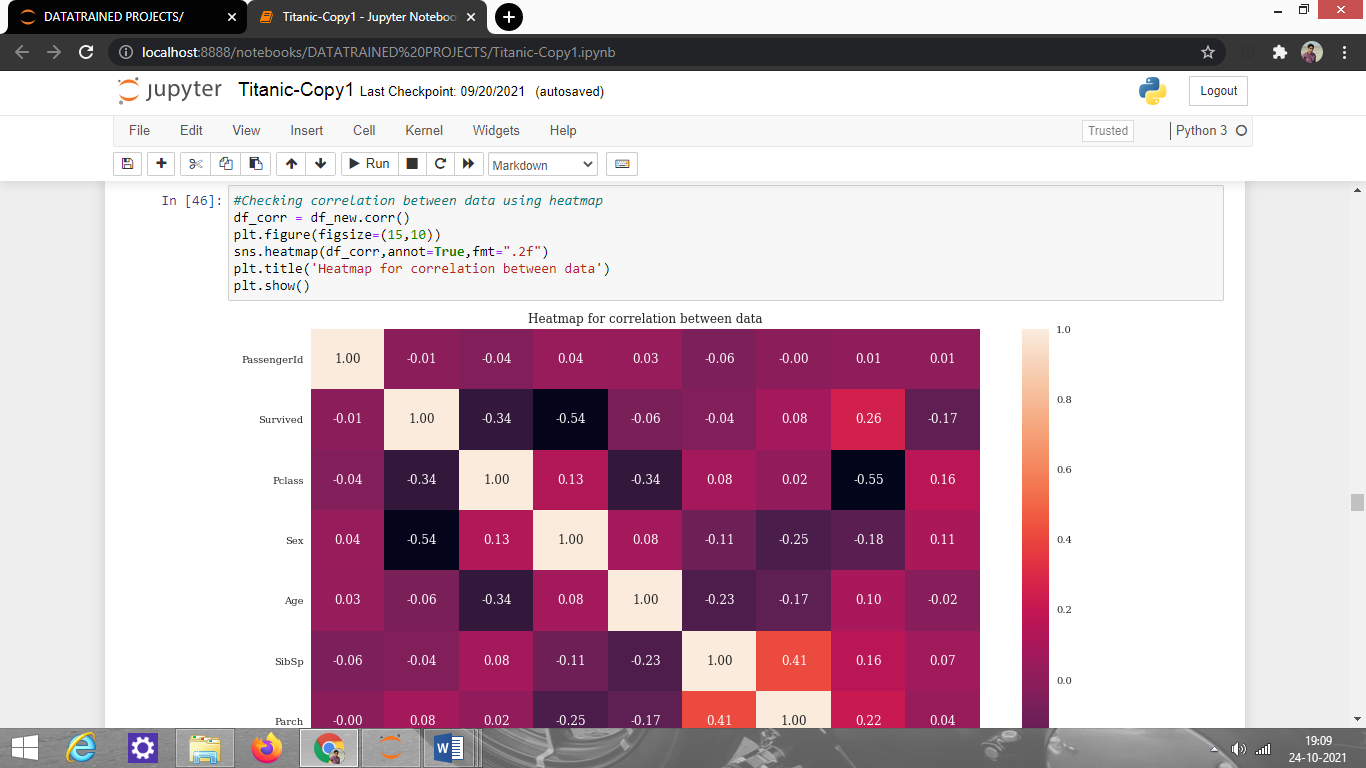
These are some of the visualization techniques I implemented to understand how the target attribute varies with the feature attributes.

**Data Pre-Processing:**

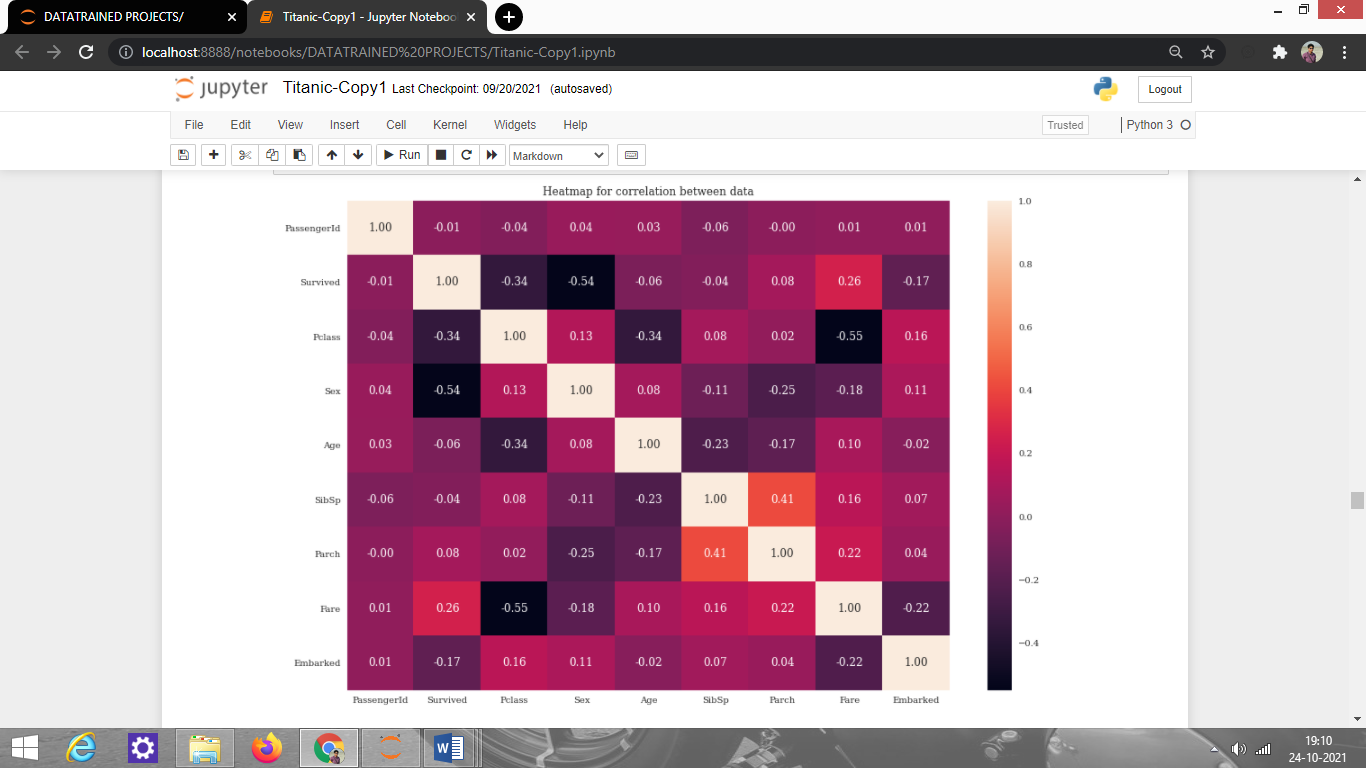
* **Checking Multicollinearity:**

Multicollinearity is a big problem in machine learning projects where two features have high correlation which will affect our model result so in the way to handle it we have many methods like we can use PCA and if I have two features that are highly correlated I can drop one of them. With the help of a heatmap visualization, I am going to check the multicollinearity which shows the correlation between features. correlation value varies between +1 to -1 where +1 indicates a positive correlation and -1 indicates a negative correlation and 0 indicates no correlation.

Code:



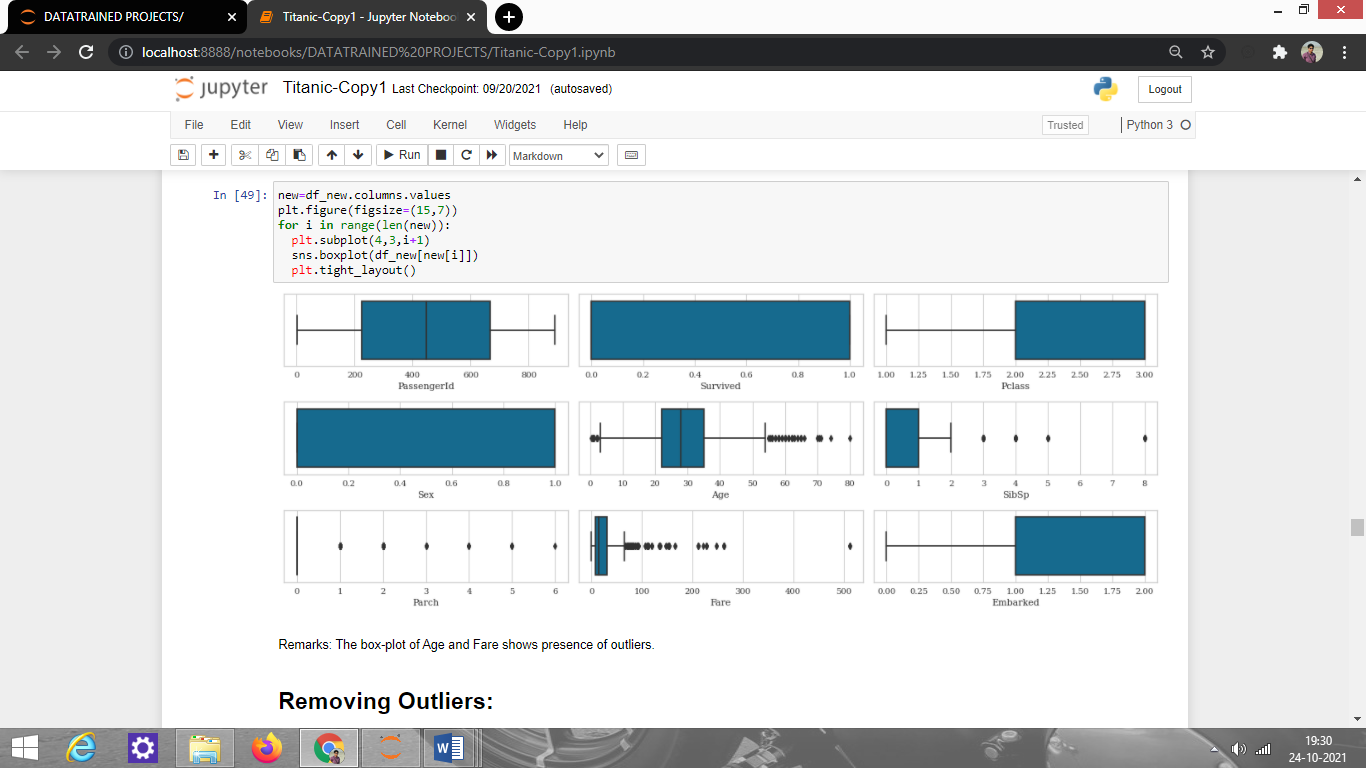
Output:



It seems that there is no multicollinearity present among the feature attributes so I can now move to the next step which is checking for outliers and skewness in the dataset.

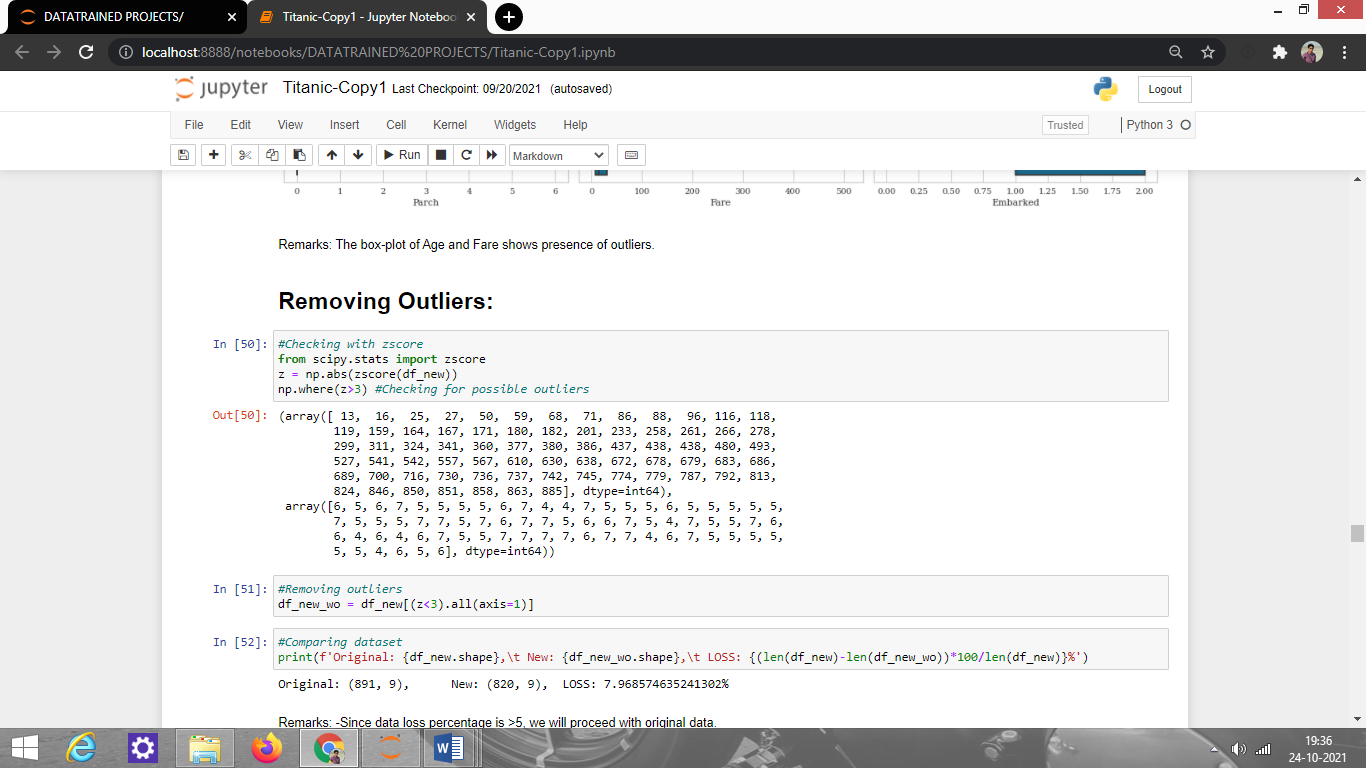
* **Outliers:** Outliers are extreme values that fall a long way outside of the other observations. There Are many ways to remove outliers and to detect them but the best way is BOX-PLOT which shows if there are any outliers or not.

Code and the output:

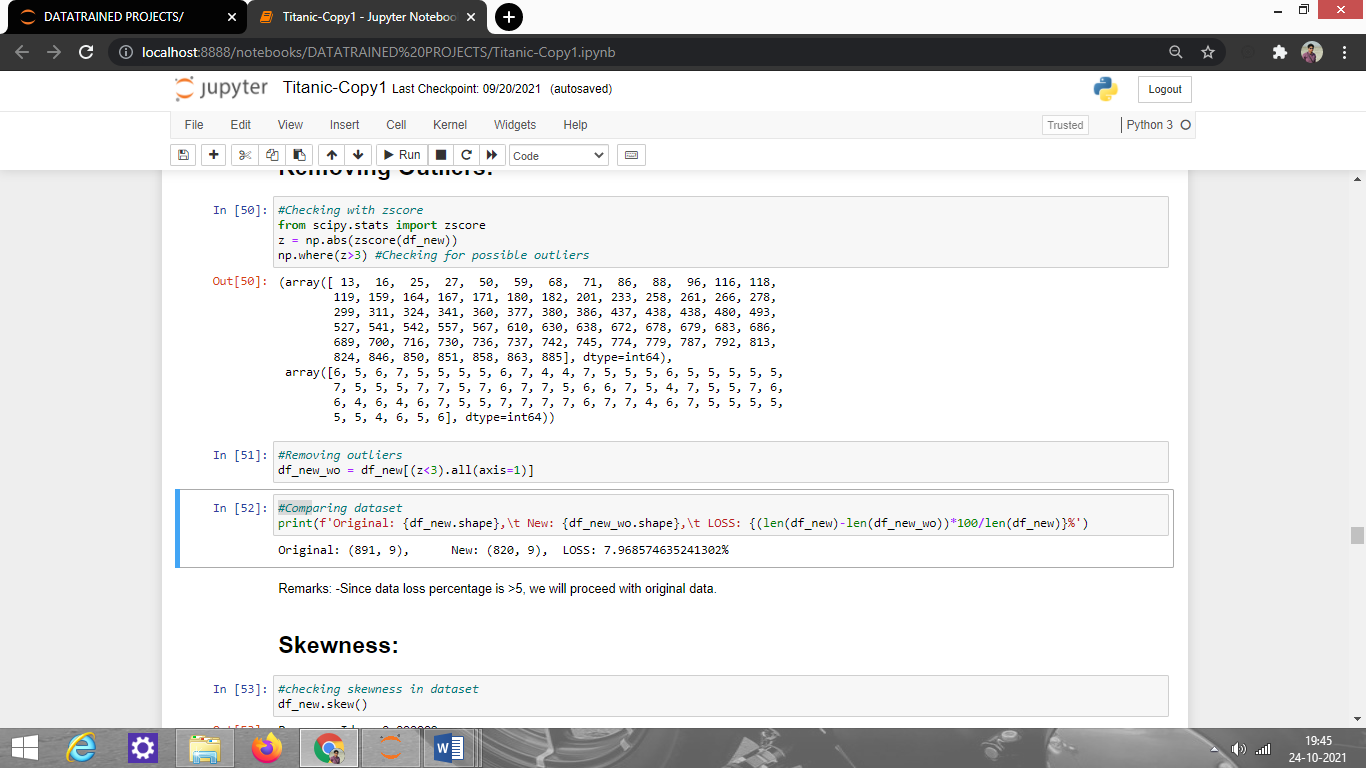


It can be seen that there are outliers present in features like Age, Fare, Parch, and SibSp. Hence, I have to remove these outliers. There are many ways to remove outliers like Z-Score, IQR-Method but I will be using the Z score method to handle these outliers.

Code:

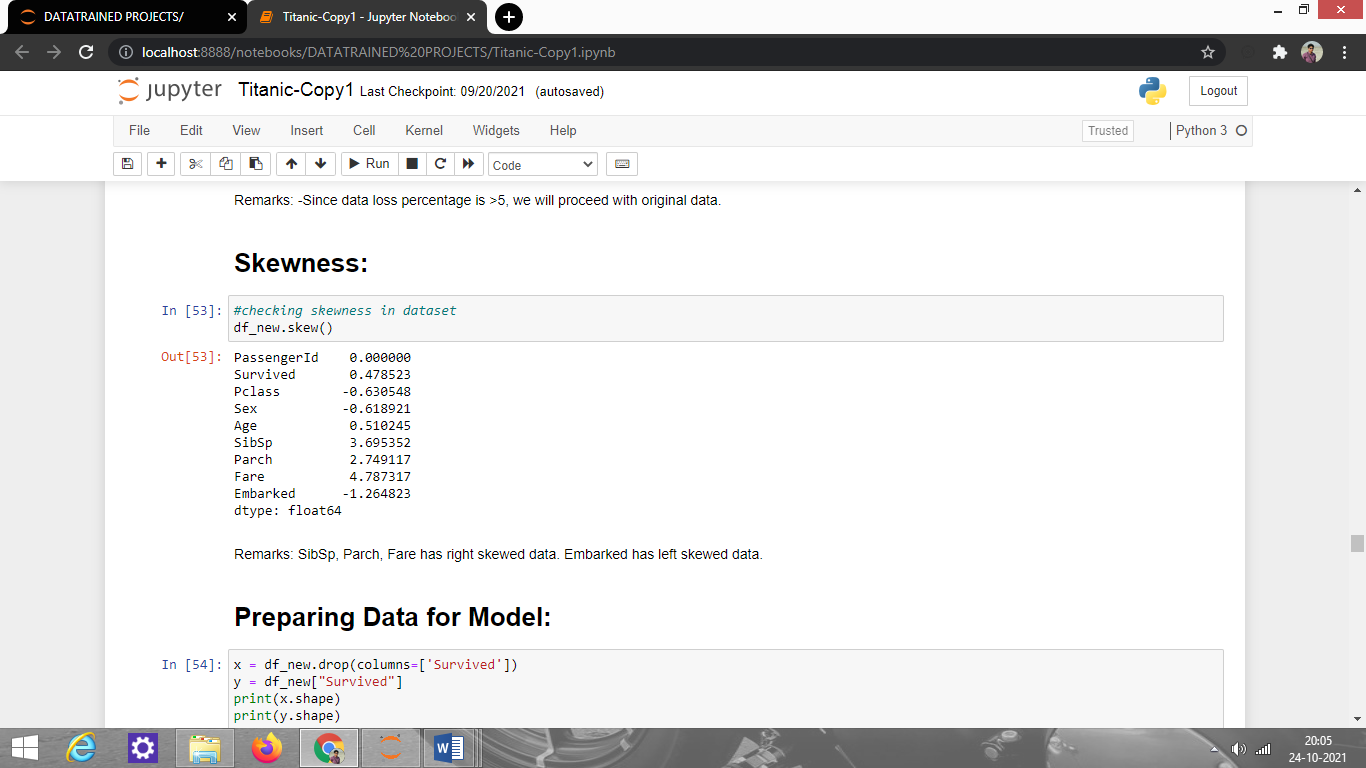


After the usage of the Z score technique, I check the percentage of data that I lost and found that 7.96% of data has been lost. However, in Data Science retaining valuable data always takes priority and fixing it rather than simply deleting it unless it is the last resort and only data loss of less than 5% is considerable. Thus, I am going to proceed with the original data that have some outliers present because data is valuable. The piece of code I used to check the percentage of data loss is:

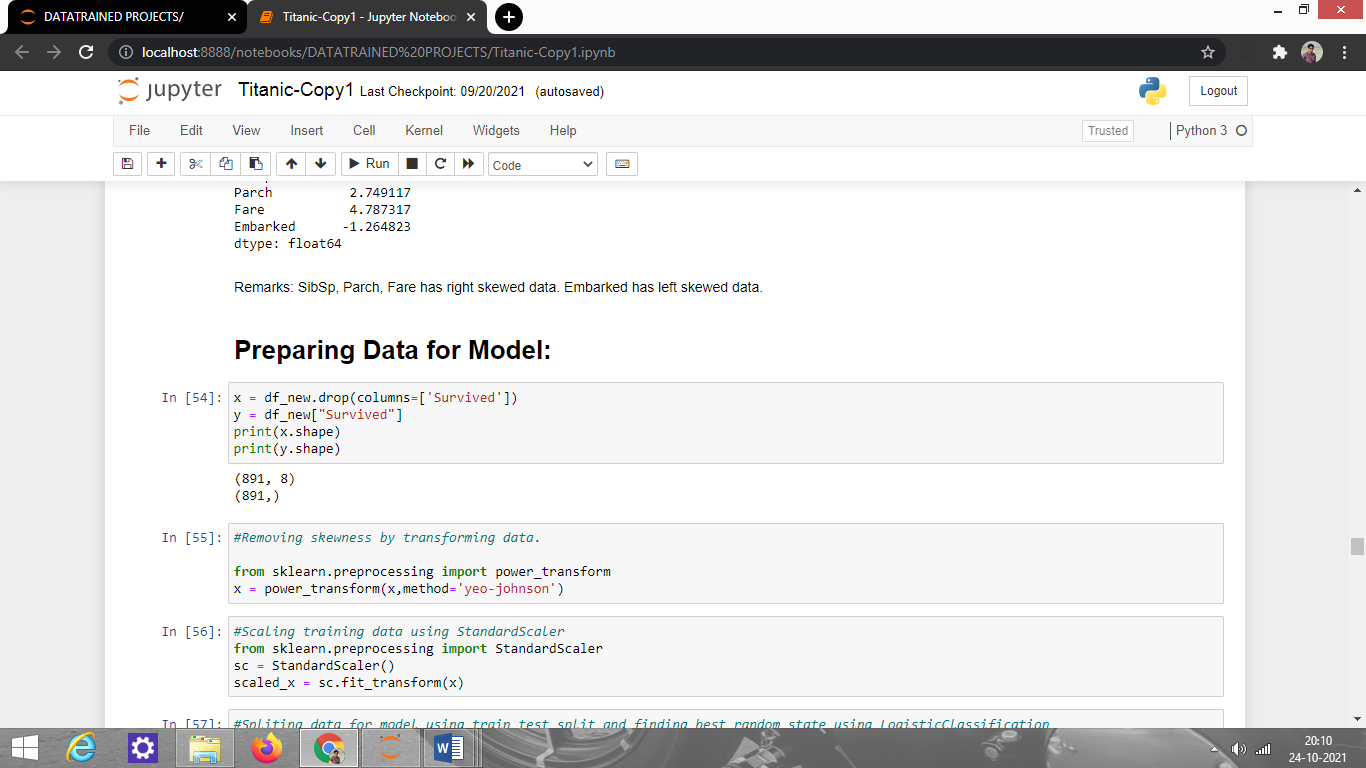


* **Checking skewness**: Skewness is the degree of distortion from a normal distribution for a machine learning model. A transformation may be used to reduce skewness . The acceptable skewness range lies between +/-0.5 value for each column and we can only reduce the skewness of continuous features.

Code to check skewness:



After running this code, I found that the data is skewed. I will be removing skewness with the help of Power transformer using method=’yeo-johnson’ because it will also deal with the negatively skewed data. The code is-



The skewed data has been successfully removed.

# *Splitting the data into dependent and independent variables:* In machine learning, the concept of dependent and independent variables is important to understand. In our dataset, the feature attributes are the independent variables while the target attribute is the dependent variable. With this in mind, we need to split our dataset into the matrix of independent variables and dependent variable. I am storing the independent variables in X and the dependent variable in Y.

# Code:

# 

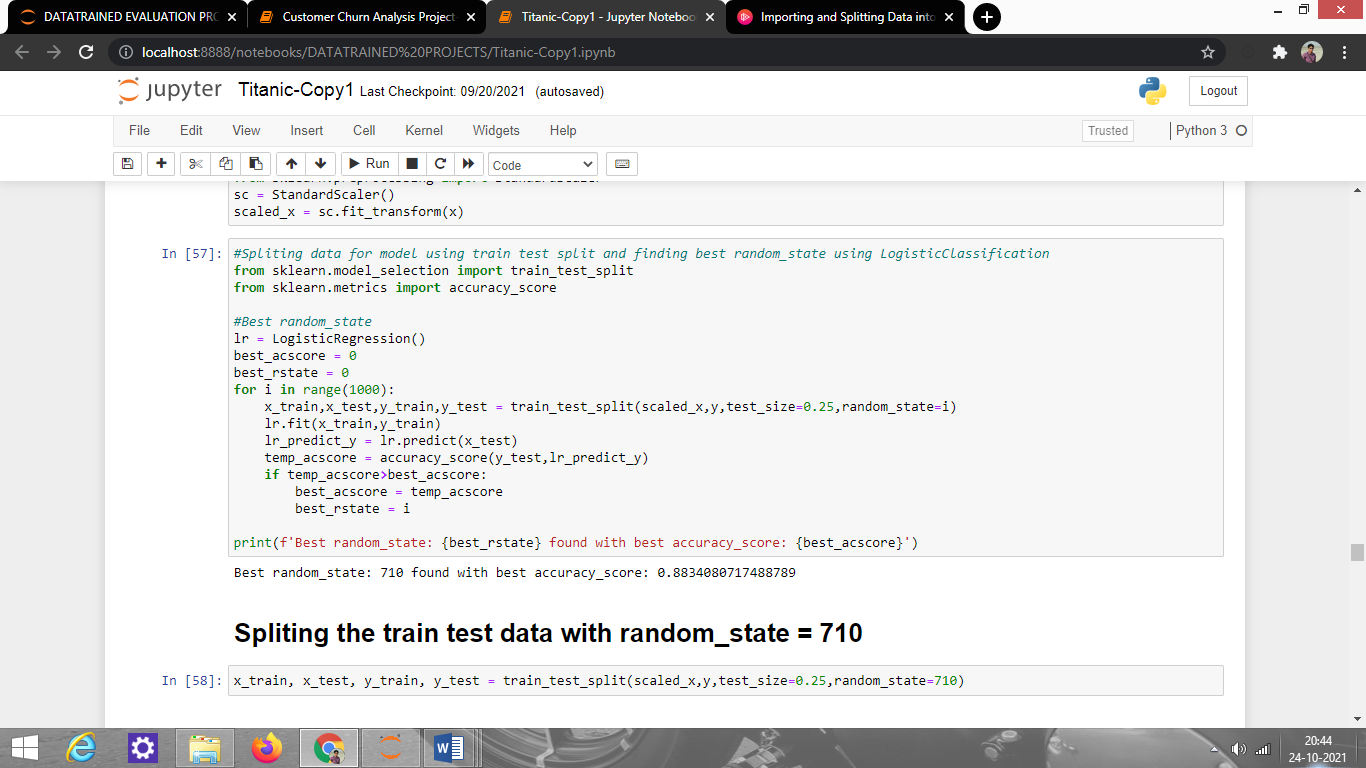
* ***Feature Scaling:*** Feature scaling is a very important step in machine learning. In datasets, we usually have some extremely high values and some minimum values so feature scaling is used to convert them into 0–1 range so our model can easily interpret values. I will be using StandardScaler to scale values.

The code is:



Next, I am going to write a simple piece of code to choose a fitting random state for the machine learning models.

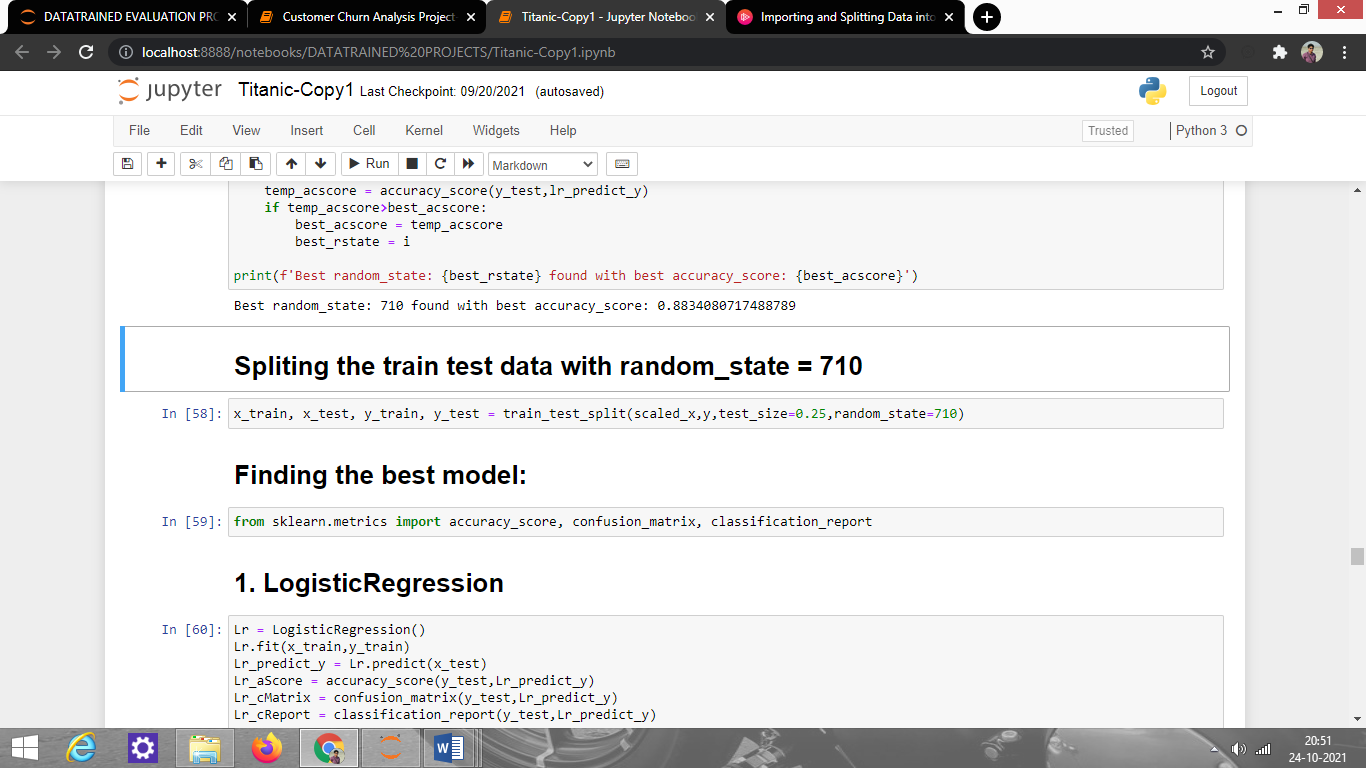
Code:



The above code gave me the best random state to be 710.

* **Splitting Data into training and testing :** Now, I will use the train test split to bifurcate the entire dataset into training data and testing data. Here I am using 75% data for training purpose and 25% data for testing purpose.

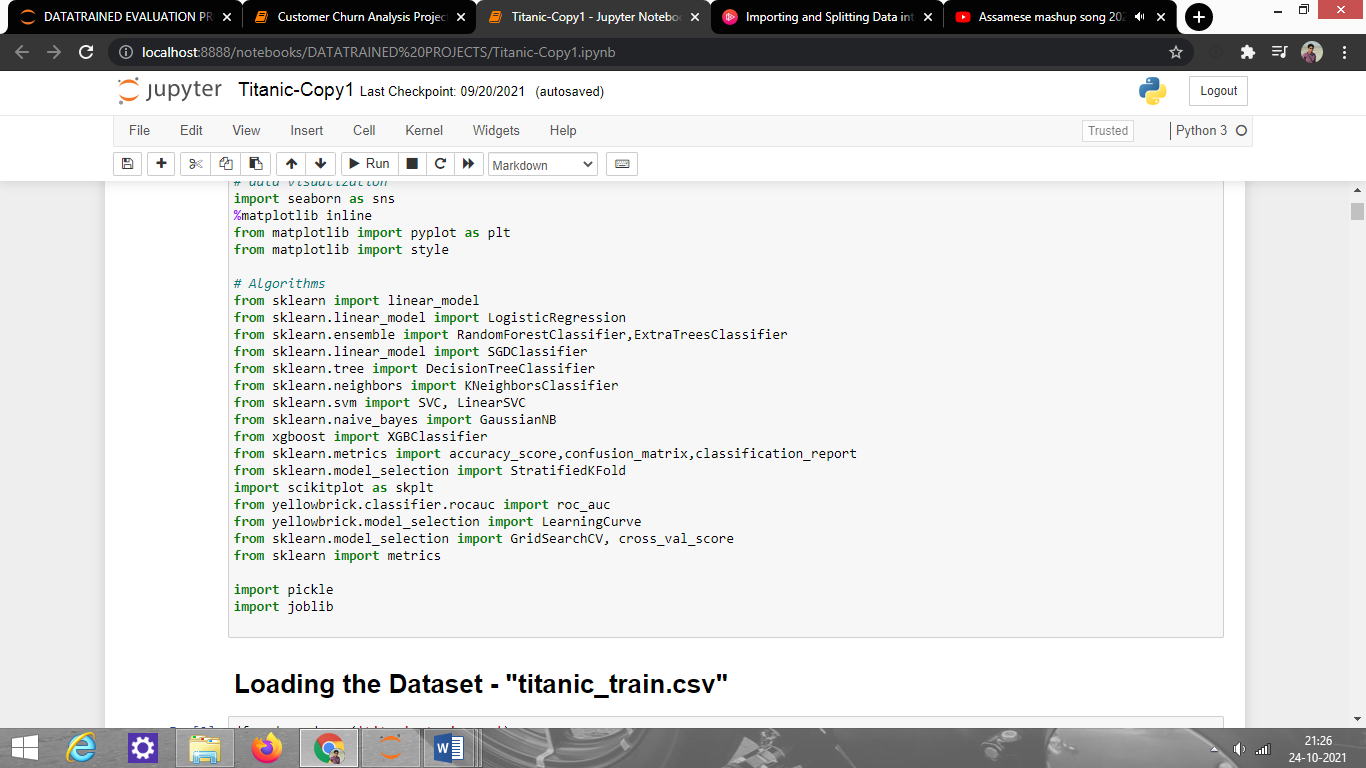
Code:



**Selecting and Training Machine Learning Models:**

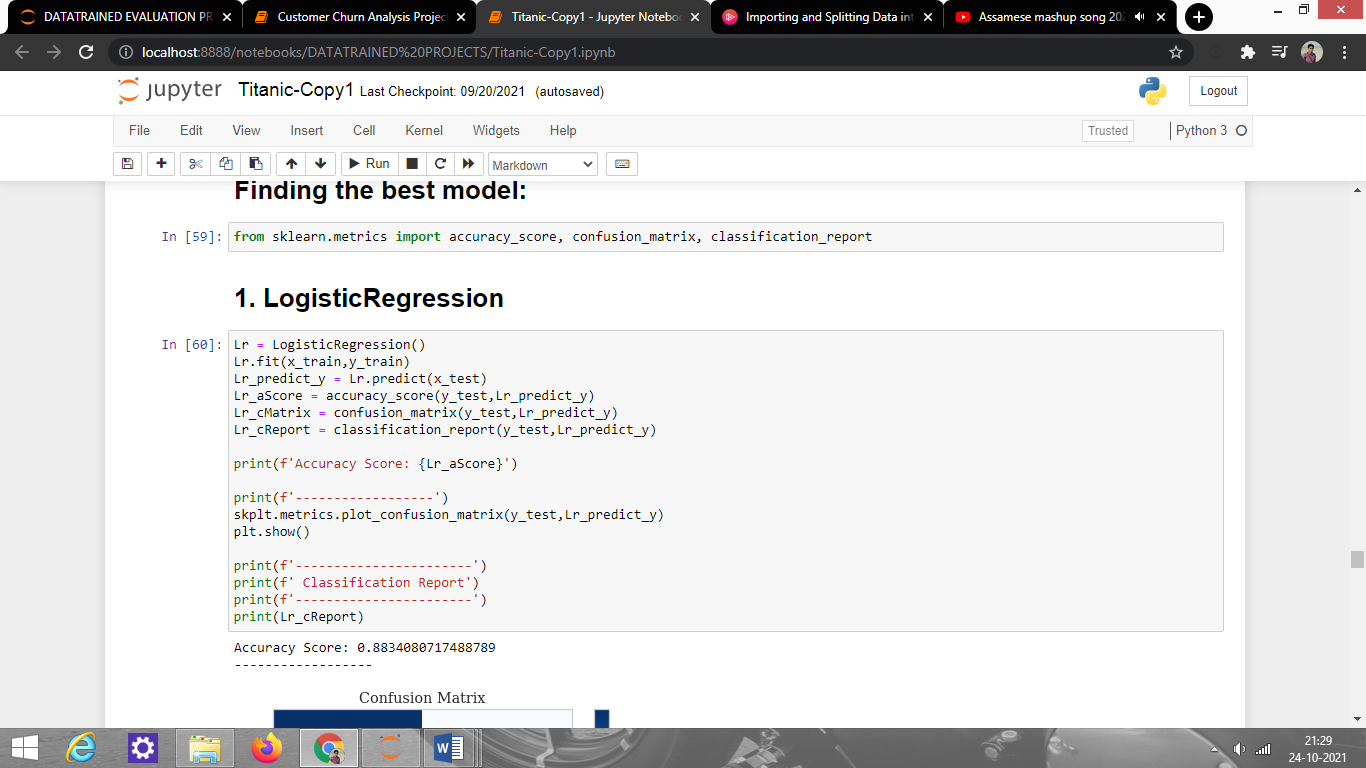
I have imported the necessary libraries for the machine learning model creation and its evaluation metrics steps. I am selecting a few classifiers such as LogisticRegression, DecisionTreeClassifier, RandomForestClassifier,etc, to train and build the ML models and later on,

Importing Libraries:

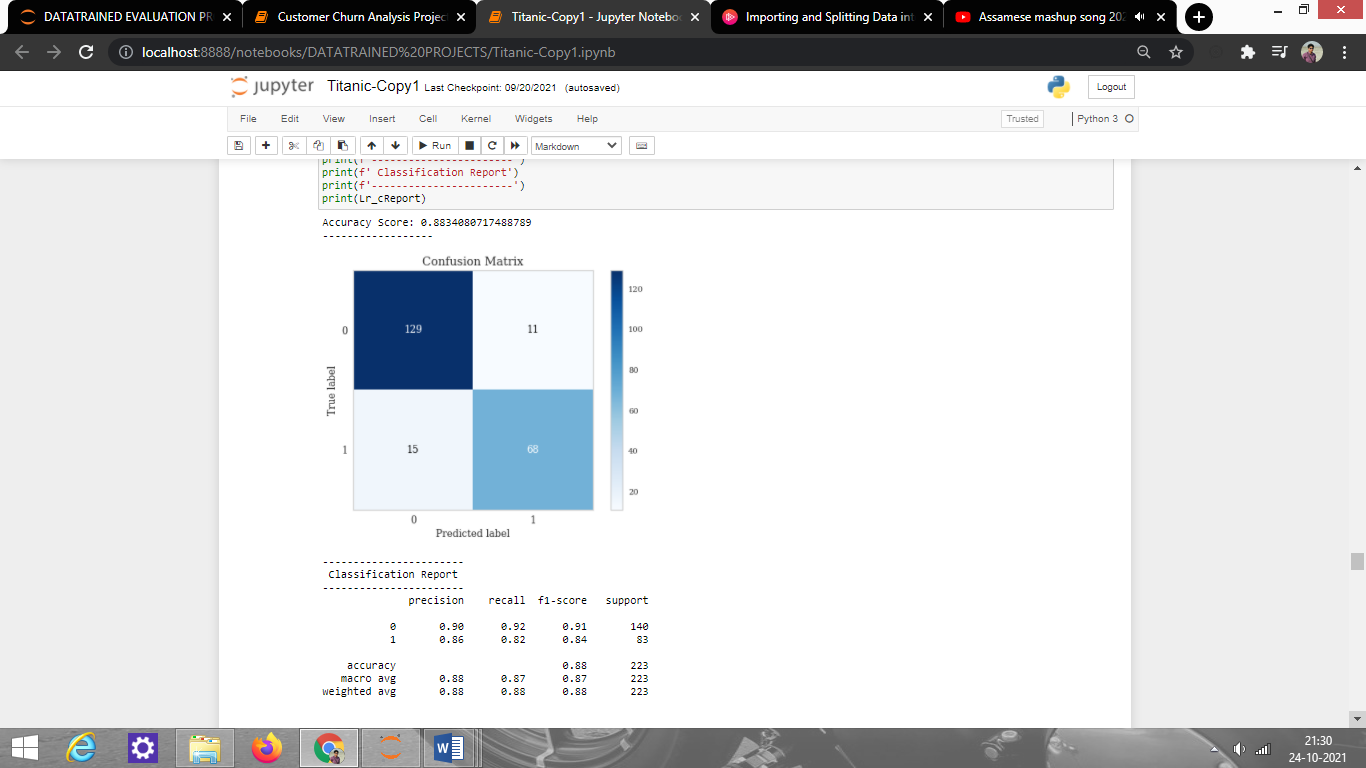


It is important to build more than 5 ML models so that you can choose from the best performing model and then apply hyperparameter tuning to make it perform even better. I am going to select the LogisticRegression as my choice of classification model as it is providing good accuracy and evaluation metrics than the other models I trained.

Code:



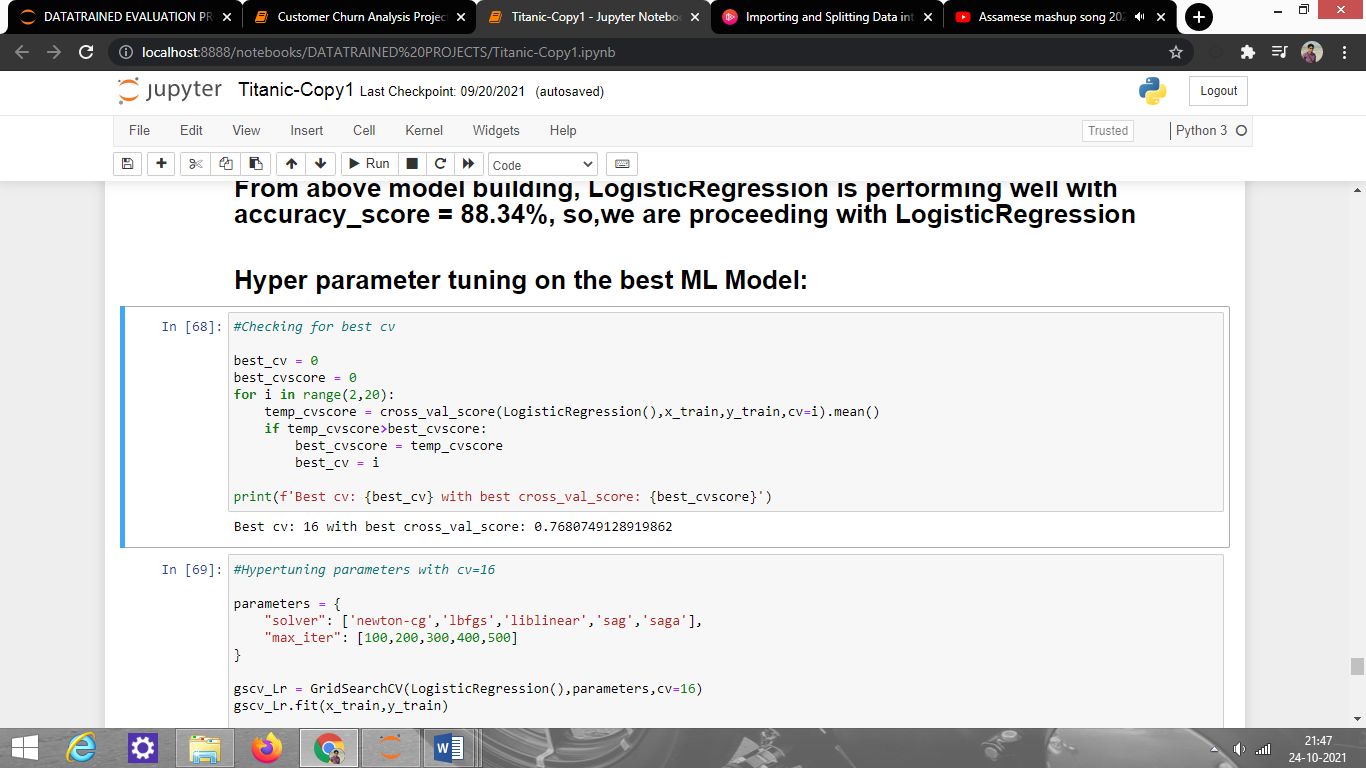
Output:



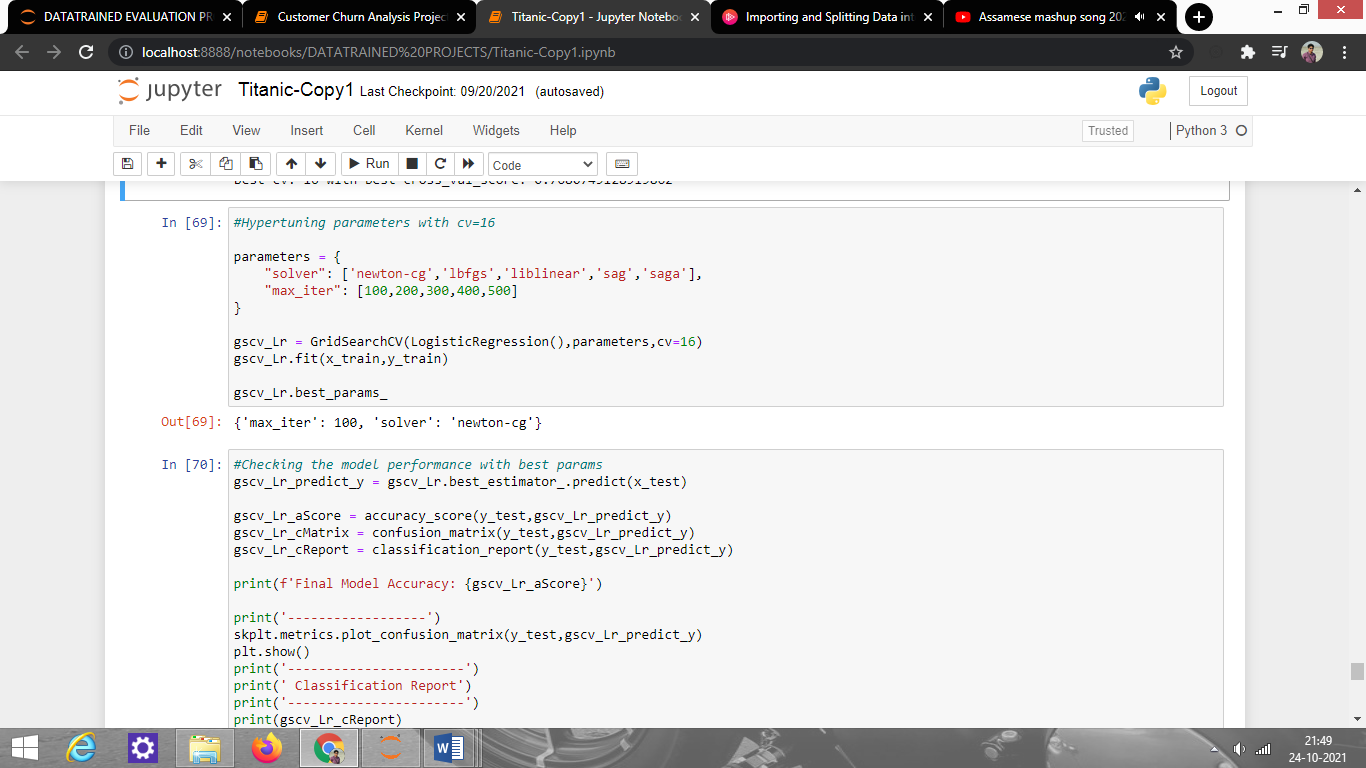
**Cross-Validation and Hyperparameter Tuning:**

I will now conduct cross-validation and Hyperparameter tuning of my best model i.e, LogisticRegression to get an improved cross-validation score and better accuracy.

Code for best CV score:



Code for HyperParameter tuning:

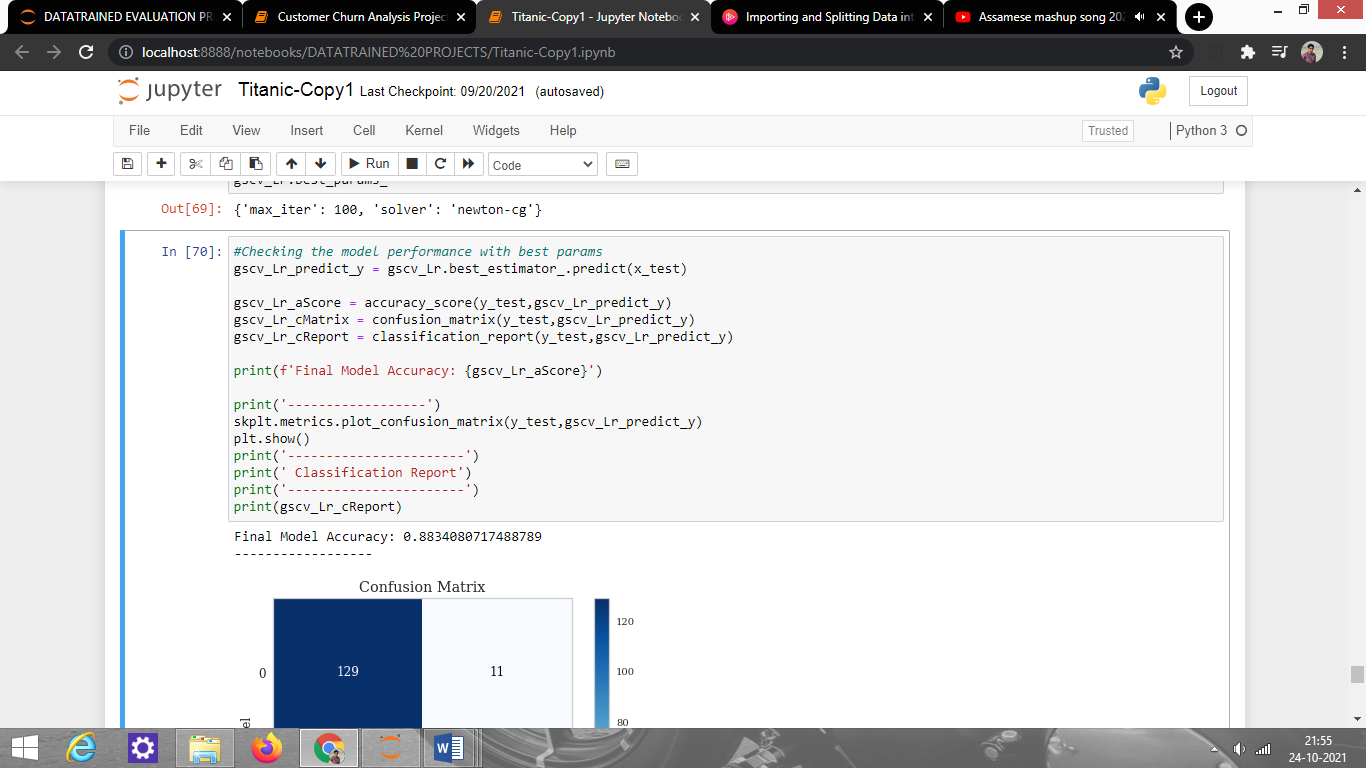


Output:

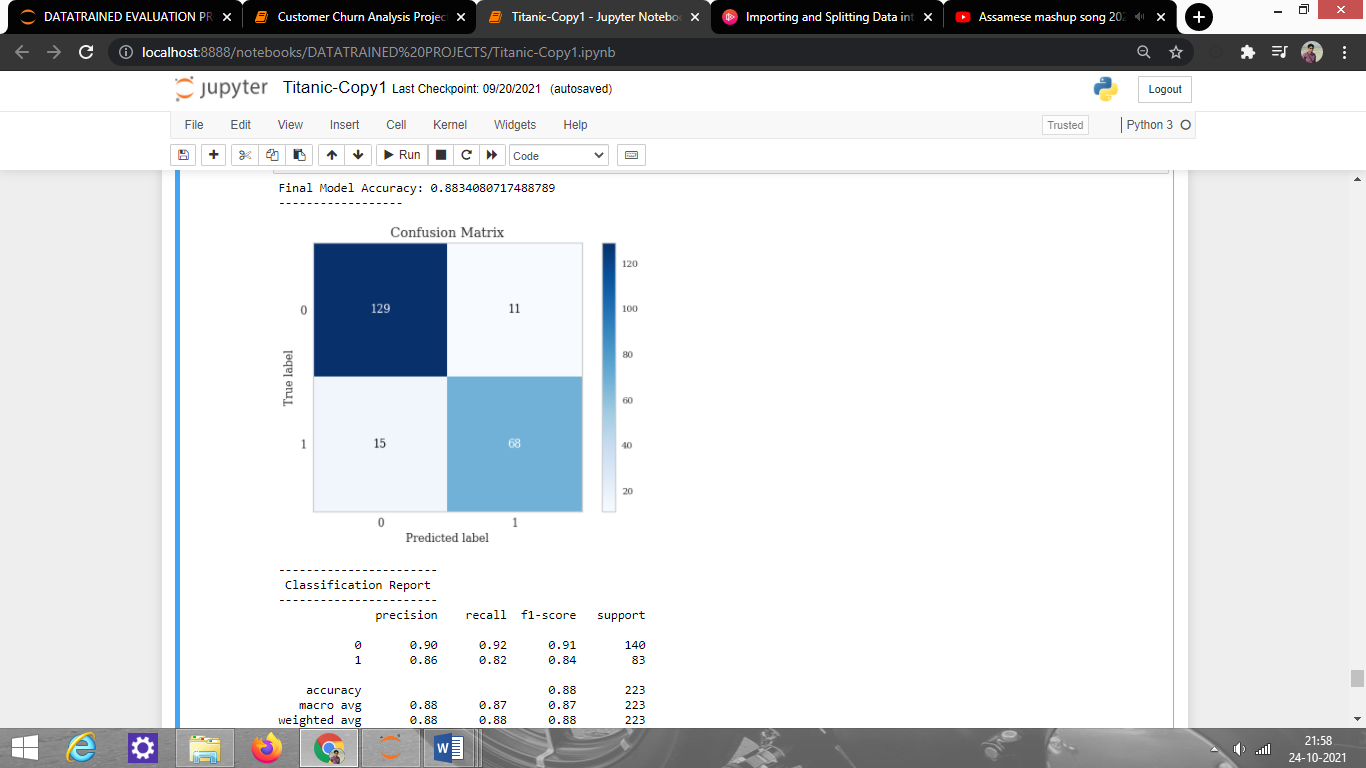
{'max\_iter': 100, 'solver': 'newton-cg'}

After applying the above steps to get the best parameters list, I have simply plugged these into my final model and received the output for it. I have created an AUC ROC curve plot and Confusion matrix for the final model.

Code:



Output:



# Code for plotting AUC ROC Curve and the output:

# 

# The AUC score for the final model is 93%

# Saving the Model:

# I am going to save the final model using either joblib or pickle. I have used the joblib method to save and then load my model from the same saved filename.

# Code to save the model:

# 

# Code to load the model:

# 

# Final predicted values:

# 

# Table above shows the original target value and predicted target value.

# Concluding remarks:

# Let’s do a quick recap of the entire process that we went through starting from understanding the Problem Statement then going through the Data Analysis and EDA processes We went through the necessary Pre-processing Data steps before training a few models. Later we selected the Machine Learning Model with the best accuracy and evaluation metrics. Also, we plotted the AUC-ROC curve then saved the final classification model. Lastly, we compared the original target value with the target values predicted by the final Machine Learning model. The model was able to predict the values with 88.34% accuracy.