**TITANIC PROJECT**

SUBMITTED BY: ANKUSH CHAUDHARI

**Developing Machine Learning Model to predict ‘Survived the Sinking'**

**Introduction**

For this assignment, I'd like to look into the tragic sinking of the Titanic. The film "Titanic," which I saw as a child, has made a lasting impression on me. The ship crashed with an iceberg early in the morning of April 15, 1912, killing more than 1500 passengers out of a total of 2,224.

In this blog, I will go through the whole process of creating a machine learning model on the famous Titanic dataset, which is used by many people all over the world. It provides information on the fate of passengers on the Titanic, summarized according to economic status (class), sex, age, and survival.

# **The Problem Statement**

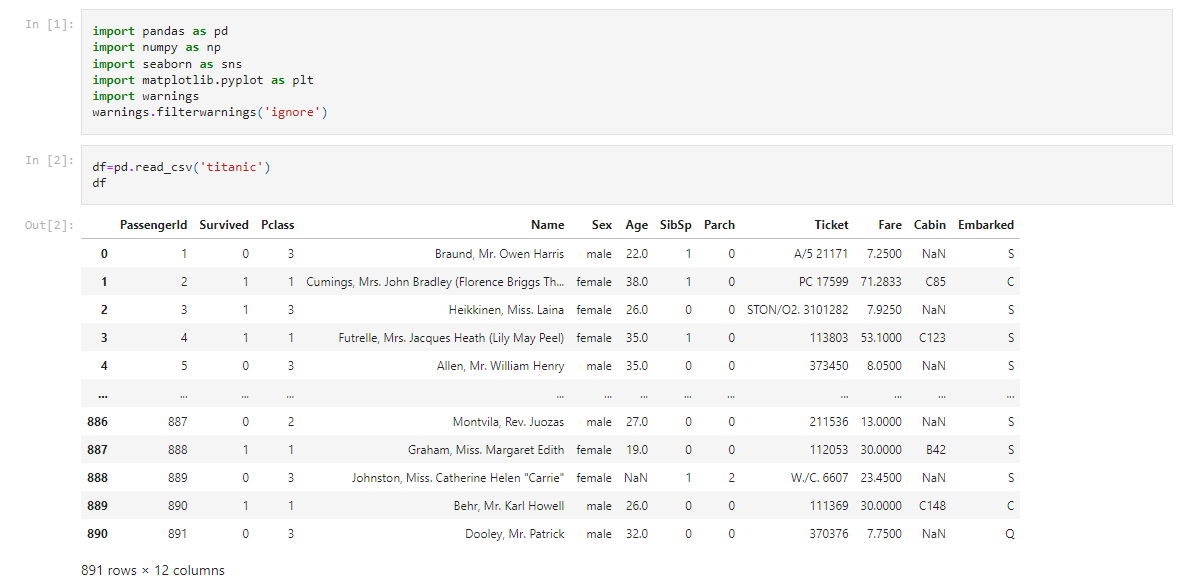
The dataset I'm working with includes demographic data as well as other details such as ticket class, cabin number, and flight amount for 891 travelers. The fundamental subject that interests me is: What are the factors that are associated with passenger survival?

# **About Dataset**

**Dataset Link:** https://github.com/dsrscientist/dataset1/blob/master/titanic\_train.csv

# **Load Dataset**

Before delving deeper, I'd like to gain a general perspective of the data and see if there is any additional data cleaning or wrangling that needs to be done. To begin, I import the CSV file into a Pandas Dataframe.



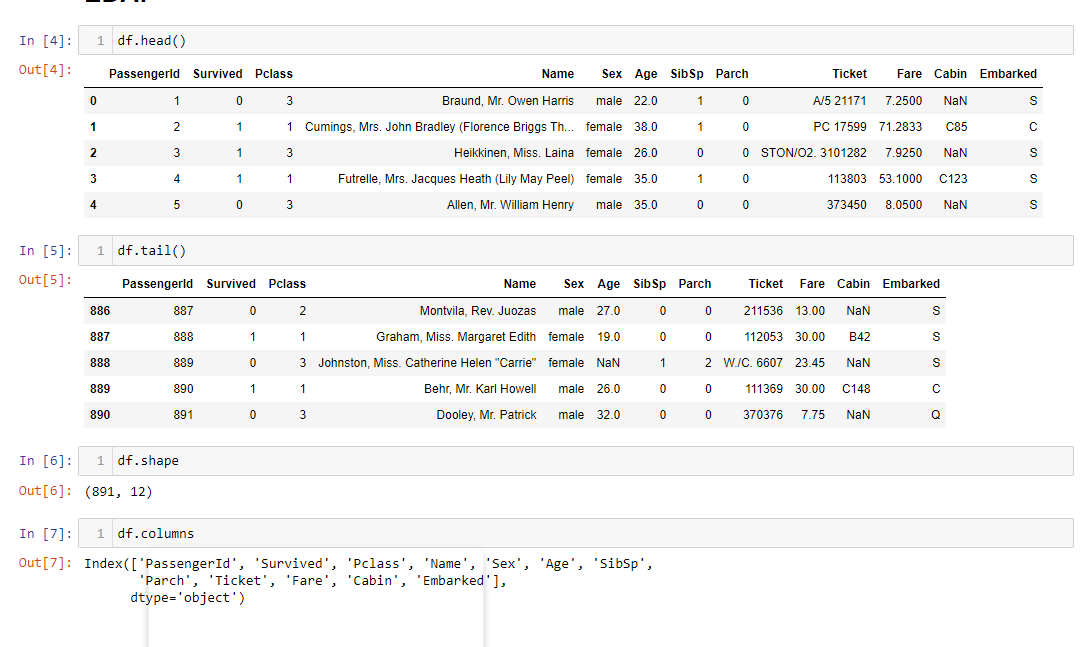
After loading the dataset and taking close look, we can see that there are 891 rows (examples) and 12 columns. Out of 12, 11 columns are independent features and 1 is a target feature or variable i.e. Survived

We need to think practically for each and every feature with respect to the relationship with our target feature and hence, deal accordingly.

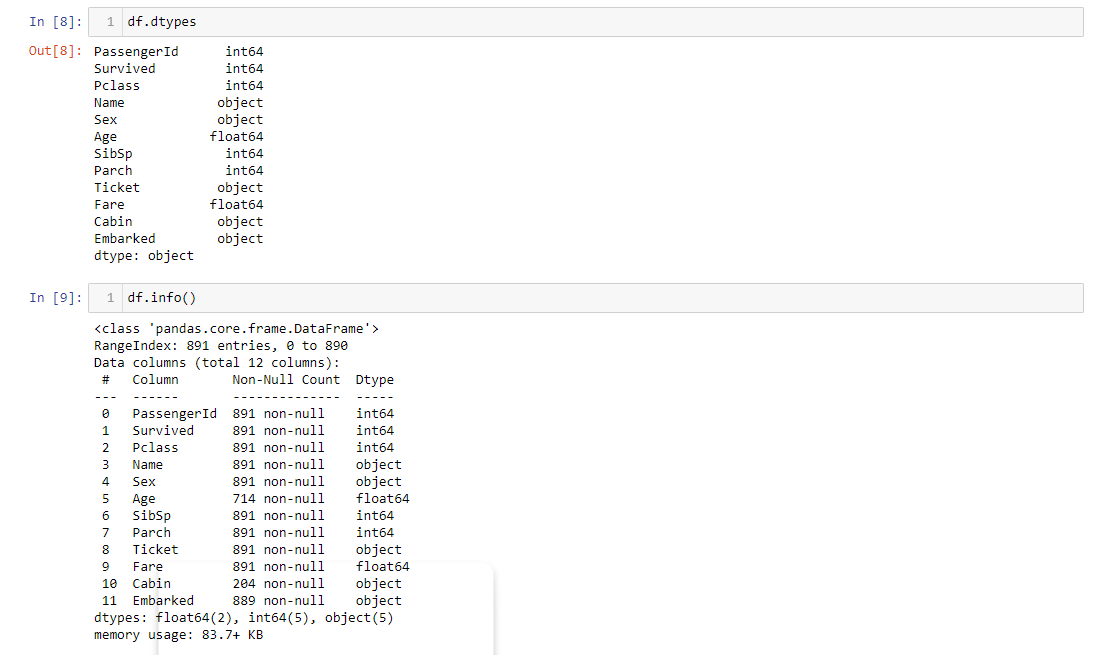
For example – As you can see above, at 0 index there is one feature called ‘PassengerID’. We can remove that column because it will not help in defining our target.

# **Data Analysis**

In this part, we analyze the data by checking its data type, data info, missing values or null values, statistics, and std. deviation, value count, and uniqueness in every column etc.

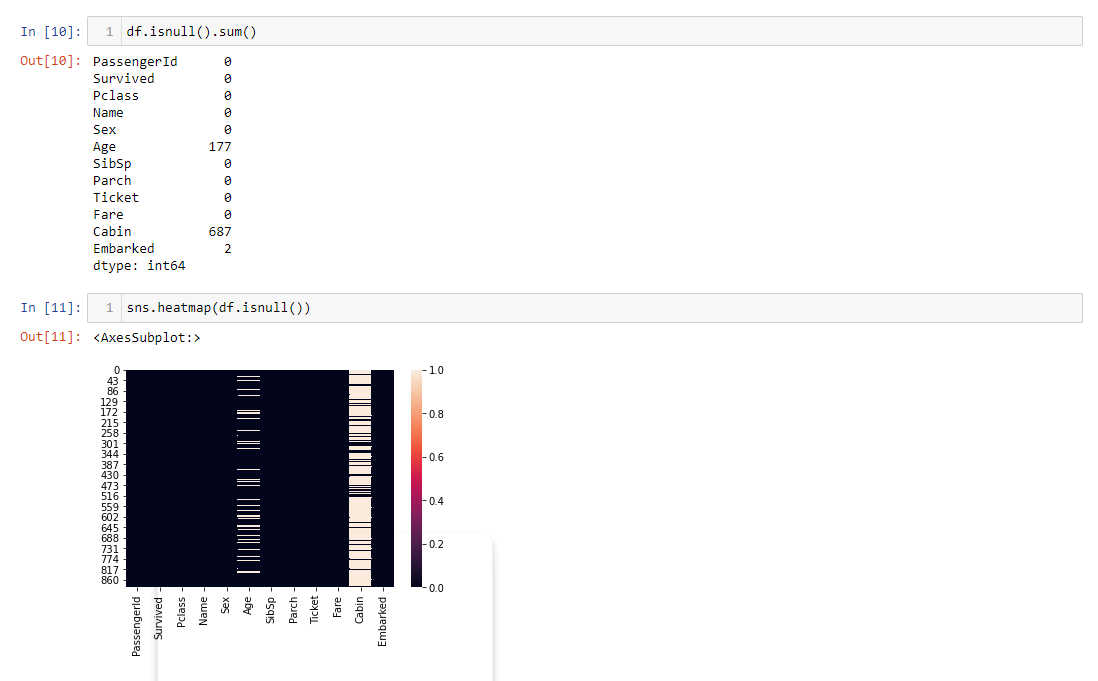


* We can see that we have, 891 rows and 12 columns in our dataset



* The columns at index 0, 1, 2, 5, 6, and 7, 9 are with numeric (float or integer) datatypes, and at index 3, 4, 8, 9, and 10 columns consisting object data type. Hence, we need to convert them into numeric for our model.

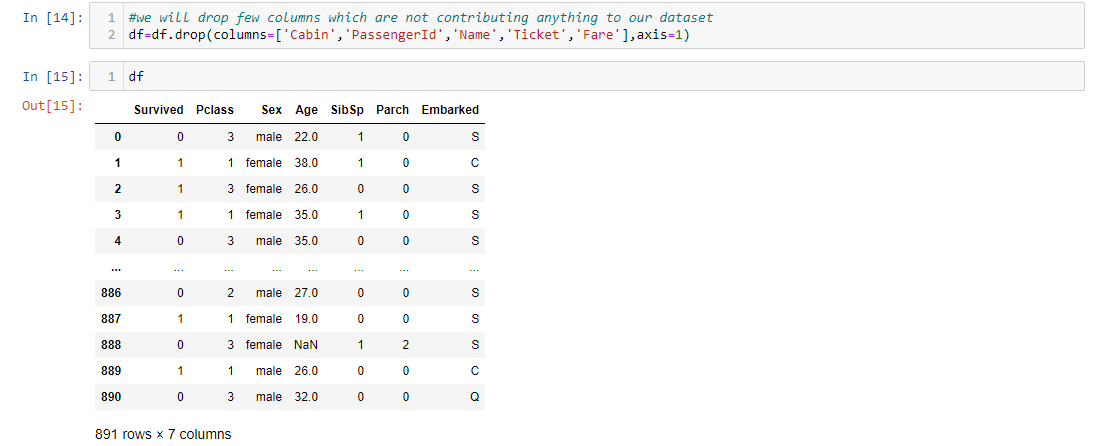
**Checking Missing values**



* There are null values present in the Age, Cabin, and the Embarked column, we will fill missing values in Age and Embarked column only because we are going to drop the Cabin column as it is not contributing to our dataset.

**Dropping the column**

* We are dropping the columns which are not contributing anything to our dataset and not necessary to keep those for model training.
* The columns we are going to drop are- Cabin, PassengerID, Name, Ticket, and Fare.



* Now we can see above that we have dropped the column and we have only 6 features left along with 1 target column. (Total 7 columns we have)

**About the columns**

1. PassengerID- It is the unique ID of the passenger.

2. Survived- Total no. of passengers survived during the incident.

3. Pclass- It is the Passenger's class (1st, 2nd, or 3rd class).

4. Name- Name of individual Passenger.

5. Sex- Passenger's sex, whether (Male or Female).

6. Age- Age of passenger.

7. Sibsp- Number of Siblings or number of Spouses Aboard.

8. Parch- Number of Parents or number of children Aboard.

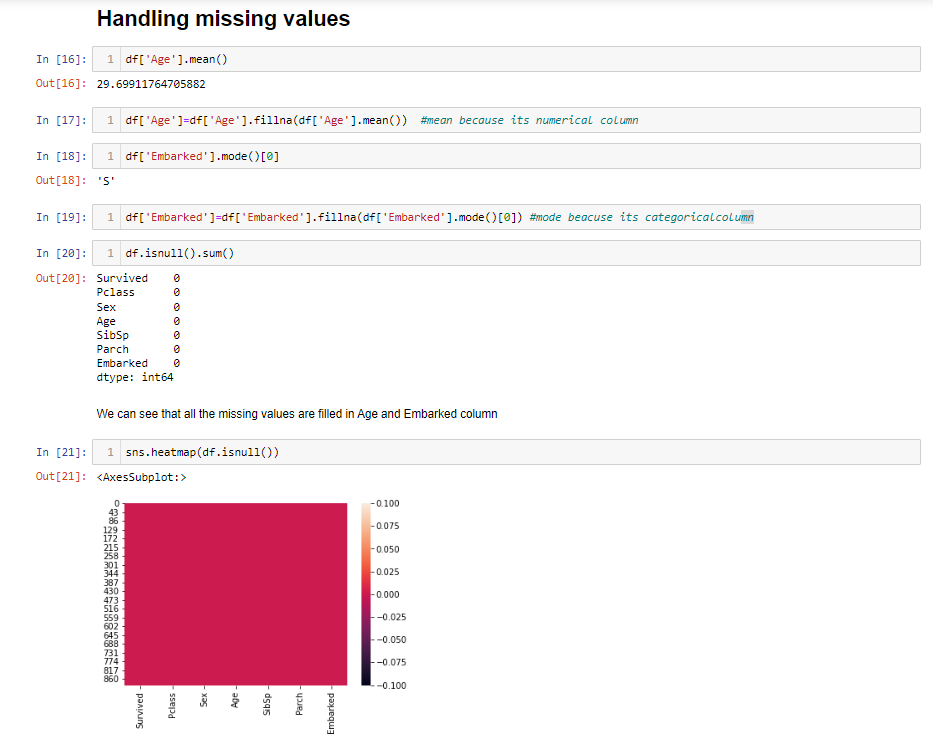
9. Ticket- Ticket number of passenger.

10. Fare- Ticket fare.

11. Cabin- Cabin number allotted to the respective passenger of 1st class.

12. Embarked- Port from where the passenger boarded the ship.

**Handling Missing Values**

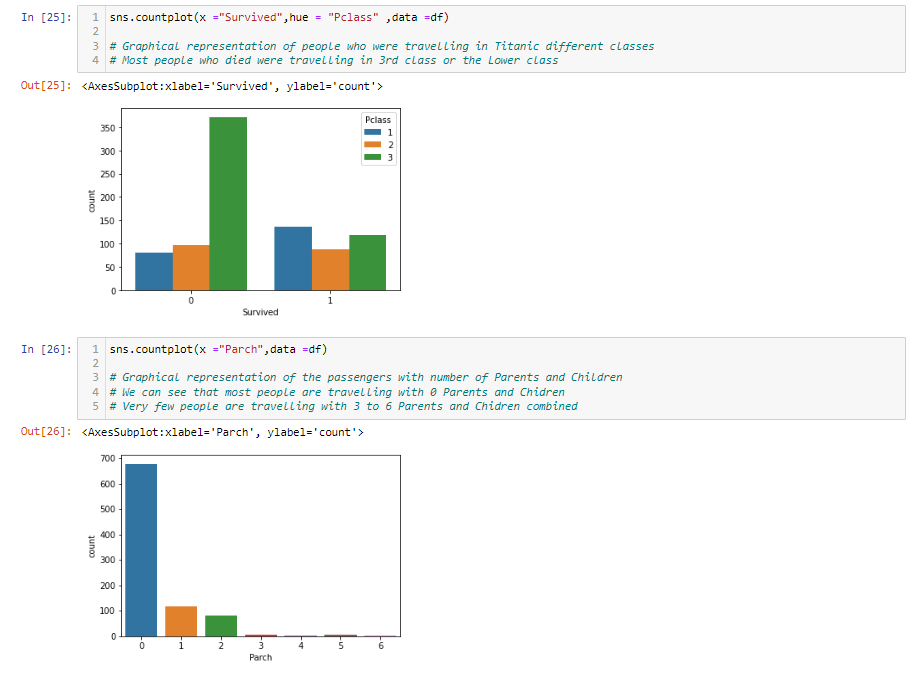


* We calculated the mean of the Age column as it includes numerical datatype and replaced all the missing values with the mean using the fillna method.
* Same way, we found the mode of the Embarked column as it includes object datatype and replaced all the missing values with the mode using fillna method.
* We can see that there are no null values in our dataset now.

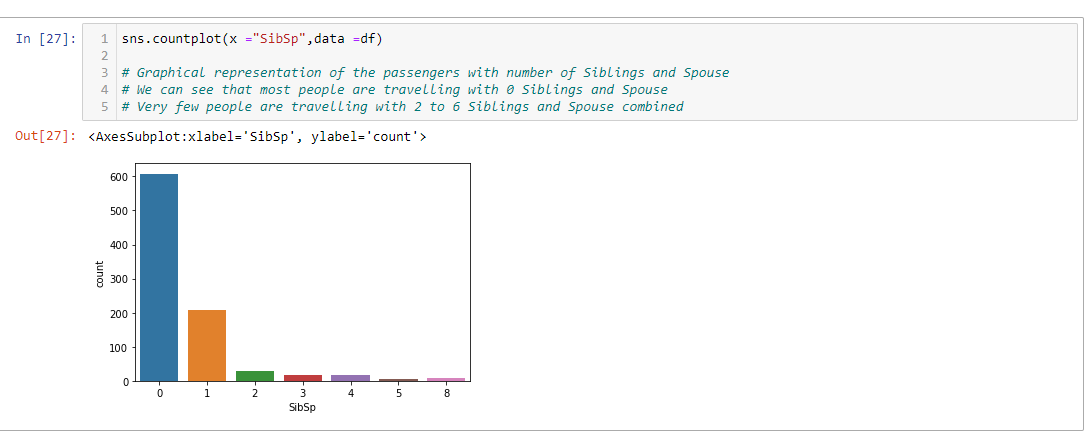
**Data Visualization**



* In the first plot we can see that more people have died during this event and fewer people survived.
* In the second plot we can see that more males died as compared to females and more females survived as compared to males.



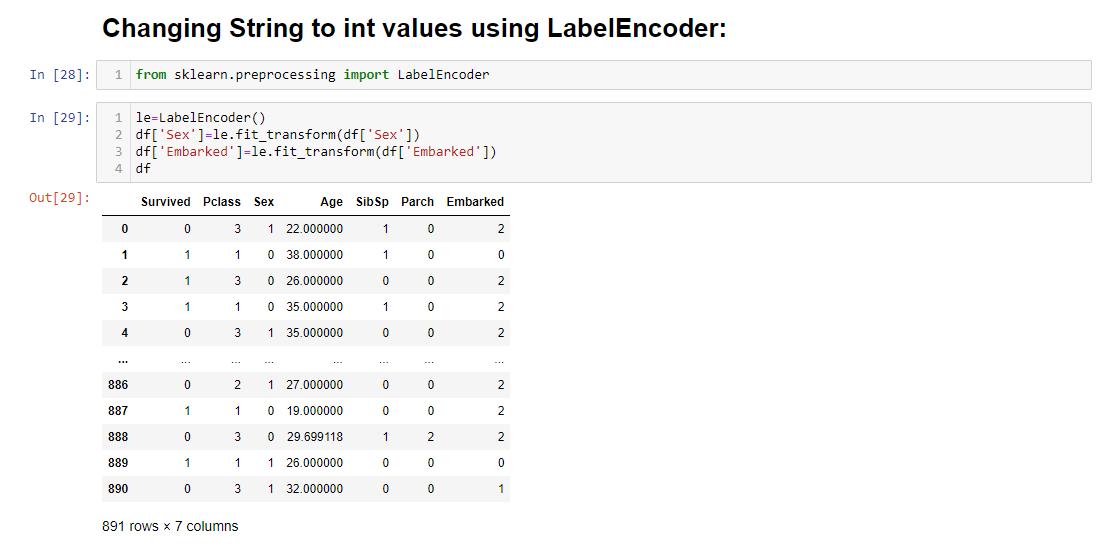
* From the first graph we can see that people from the third class died more as compared to other classes.
* From the second graph we can see that the number of travelers with no partner is more.



* From the above plot we can see that most of the travelers were traveling with 0 siblings and spouse and very few people were traveling with 2 to 6 siblings and spouse combined.

**Label Encoder**

The performance of a machine learning model is determined not only by the model and hyperparameters but also by how different types of variables are processed and fed into the model. Because most machine learning models only accept numerical variables, categorical variables must be pre-processed. We must transform these categorical variables to integers in order for the model to comprehend and extract useful information.

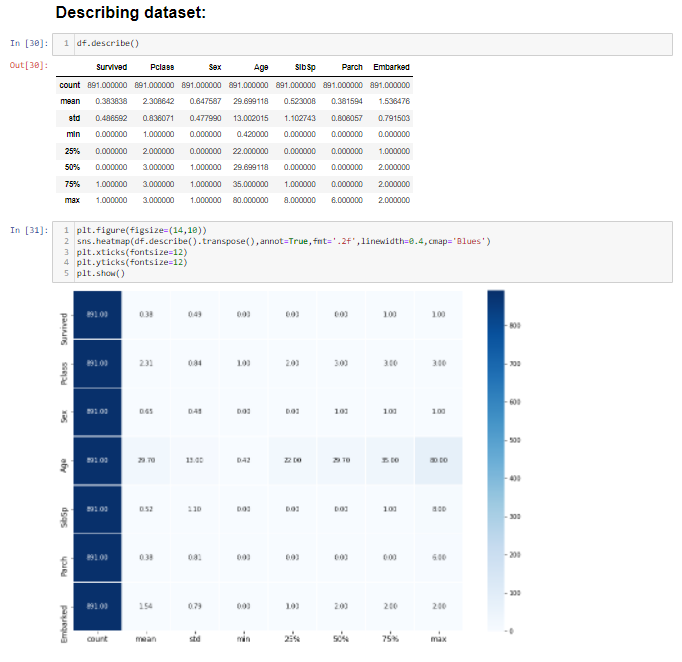


* Here we can see that using the Label Encoder library from sklearn, we have changed the objective datatypes into numerical values.

# **Exploratory Data Analysis (EDA) and Visualisation Concluding Remarks**

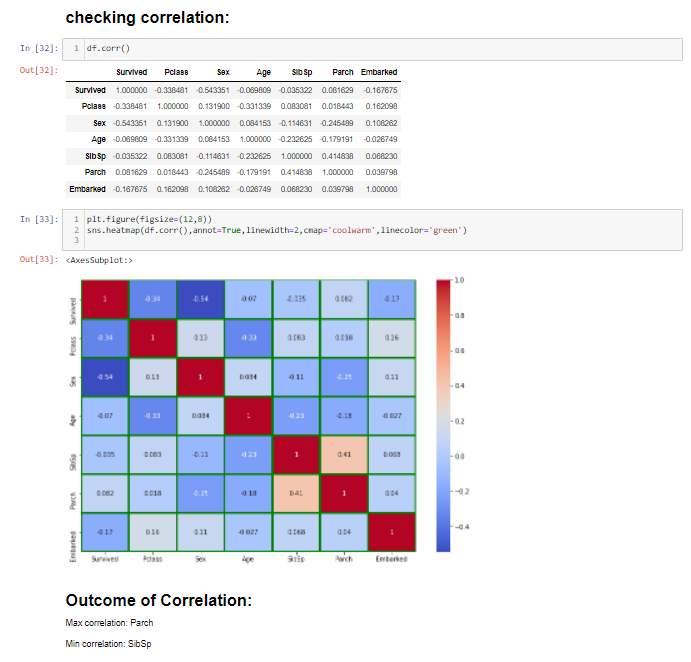
Exploratory Data Analysis (EDA) is a way of analyzing data sets in order to summarise their primary characteristics, which frequently involves the use of statistical graphics and other data visualization techniques. Univariate, Bivariate, and Multivariate Analysis are the three types of analysis.

**Describing Dataset**



* The description of the dataset gives important information such as mean, std deviation, 25%, 50% and 75%, as well as minimum and maximum value of each and every column.

**Correlation of Features**



* Checking the correlation of the dataset is very important as it gives us information about how each column is correlated to another column in our dataset. High correlation means it is contributing more to the target column and less correlation means it is not contributing to the target column.
* Here we can see that Parch has maximum correlation and SibSp has less correlation to our target Survived column.

## **Outlier(s): Detection & Removal**

* Outliers are extreme values ​​that are far from other observations. It can be detected and removed using either the ZScore or Interquartile Range (IQR) methods. We will use zscore for this purpose here.



* We have checked the outliers present in the dataset using boxplot method. There are many outliers present in the Age column and also in SibSp and Parch.
* To perform the model well, we have to remove these outliers in further steps.

**Outlier Removal**

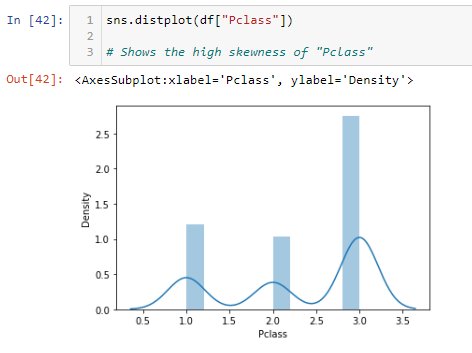
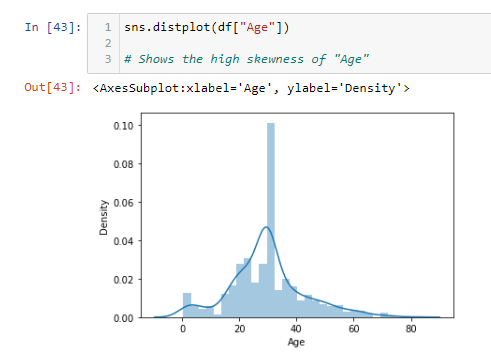
## 

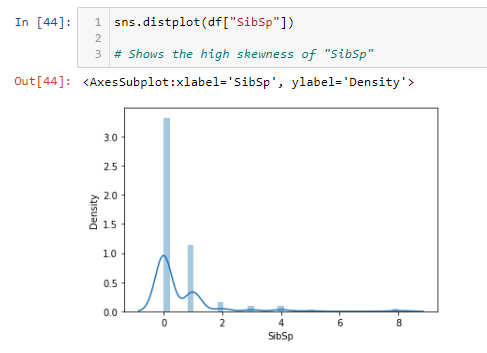
* We have imported zscore from scipy.stats for outlier removal.
* After removal of outliers, we have 839 rows left out of 891 rows but the columns are the same.
* We have calculated the percentage of data loss and found that we have lost 5.8% of data during outlier removal.

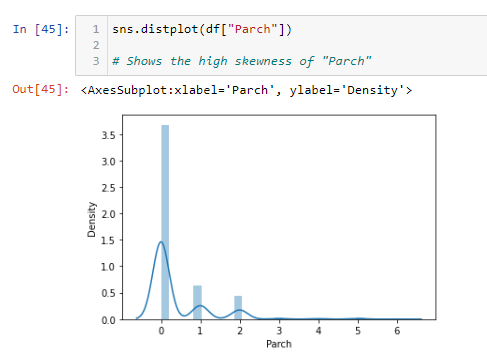
**Data Distribution using Distplot**

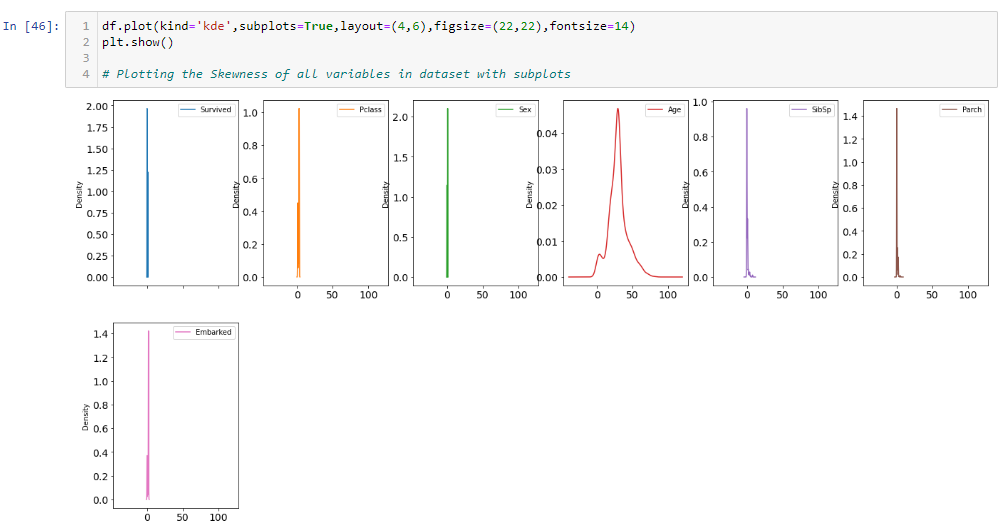
The distribution of data basically tells us about the mean, median, mode, maximum, minimum, standard deviation, and skewness of data. We can get and visualize them using the distribution plot of the seaborn library as shown below:

* We can see that the distribution plot shows, that the data is not distributed normally in all of the features and this is understandable because all of the features are of categorical type.

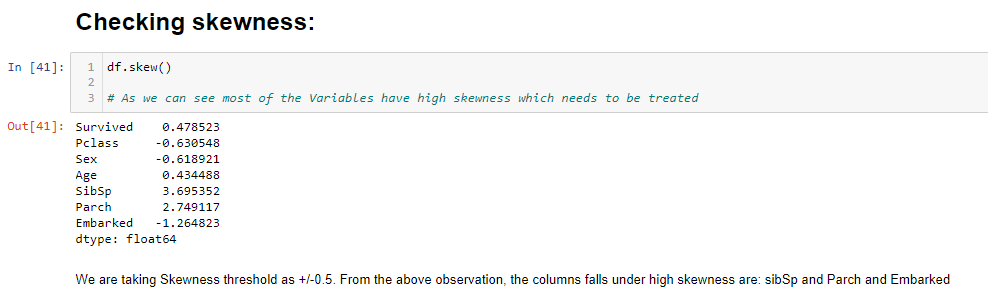
 







## **Skewness: Detection & Treatment**



* We can see that the columns with high skewness are SibSp, Parch, and Embarked.
* We are taking a threshold of +/-0.5 here.

## **Separate Input and Output/Target Variable and Scale Feature Data for Model Training**

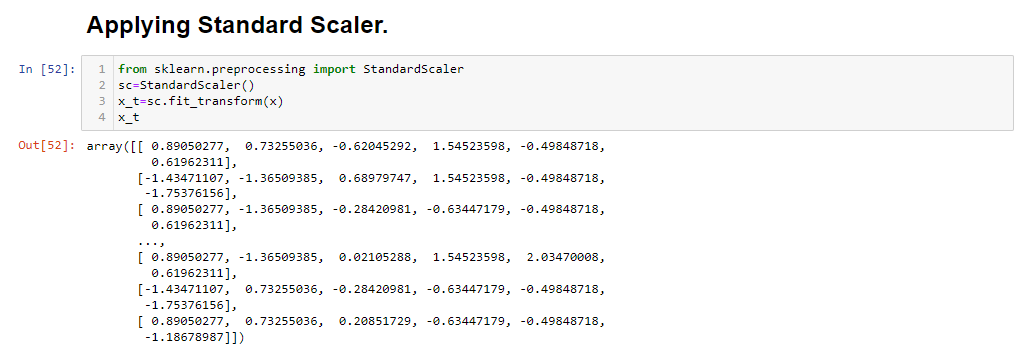
* Now, we can separate the features into the input as X and output/target as Y to continue further with data preparation.

# 

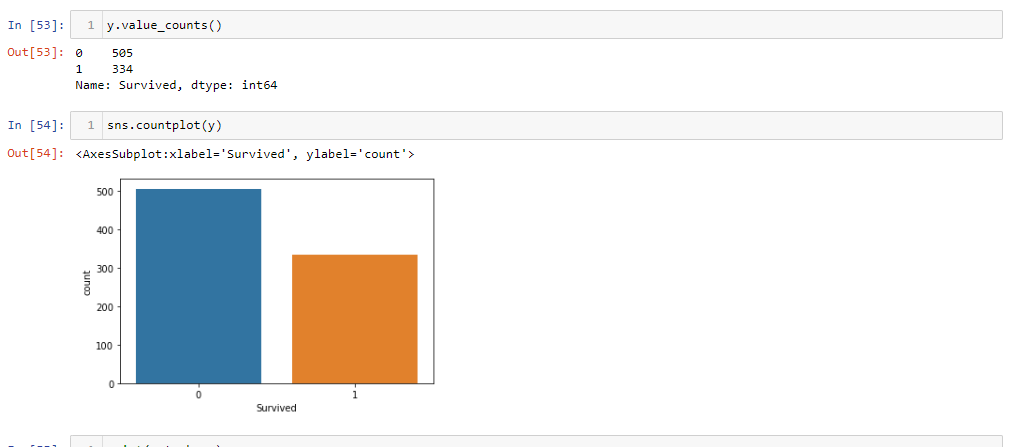
* Above we can see that we have removed skewness using power transform method.

**Standard Scaler**

* Scaling data for model training refers to normalizing the range of independent variables or features of data. Here we are using StandardScaler for this purpose



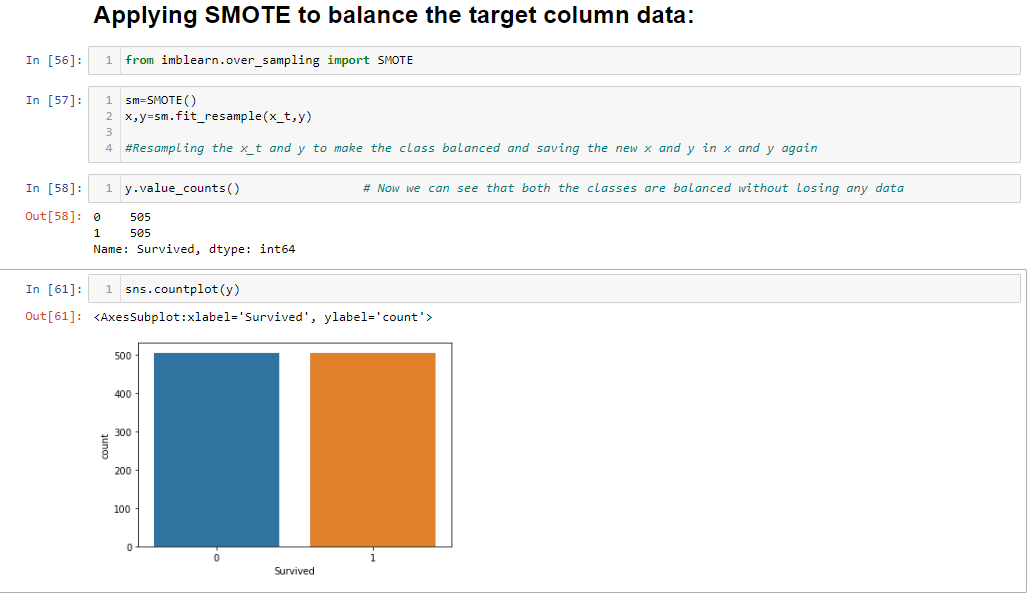
**Verifying Class Imbalance in the target column**



* Using the value counts function we have checked the numbers of people who survived and numbers of people who died.
* We can see that 505 people died and 334 people survived.
* Using the count plot also we can easily see that there is a huge class imbalance issue here.
* In this case we have to balance the class before building the model.
* We will use SMOTE method to balance the target variable

**SMOTE**

* We have imported SMOTE from the imblearn oversampling to balance the dataset
* We can see below that the target column is balanced now.
* The number of survived and died people are equal now.
* Treating unbalanced data is necessary before the model training as unbalanced data can predict the biased result. So the accuracy of the model will be not true.



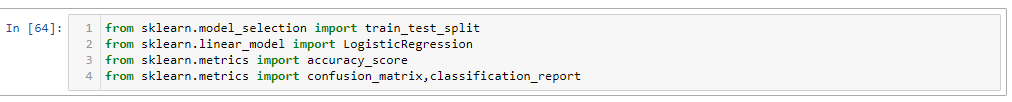
# **Prepare Dataset for Model Training**

Preparing dataset Model training is the one of core parts of machine learning model building and includes different types of data modification and transformation to achieve better model performance.

# **Model Training: Finding the best model**

The models that I have decided to train for this dataset are LogisticRegression, SVC (Support Vector Classifier), KNeighboursClassifier, DecisionTreeClassifier, RandomForestClassifier, GaussianNB models. The goal here is to find the best hyper-tuned models for further processing.

**Importing Libraries**

**Applying loop for getting best random\_state**

## 

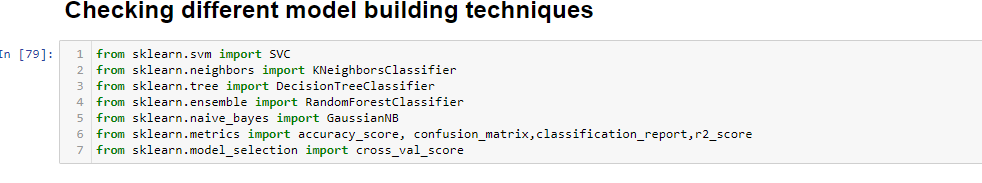
* We found our best random state 287 which is giving us high accuracy of 86%
* Now, we are going to use this random state for creating train\_test\_split

## **Prepare Model List and Test to get Best Model**

### 

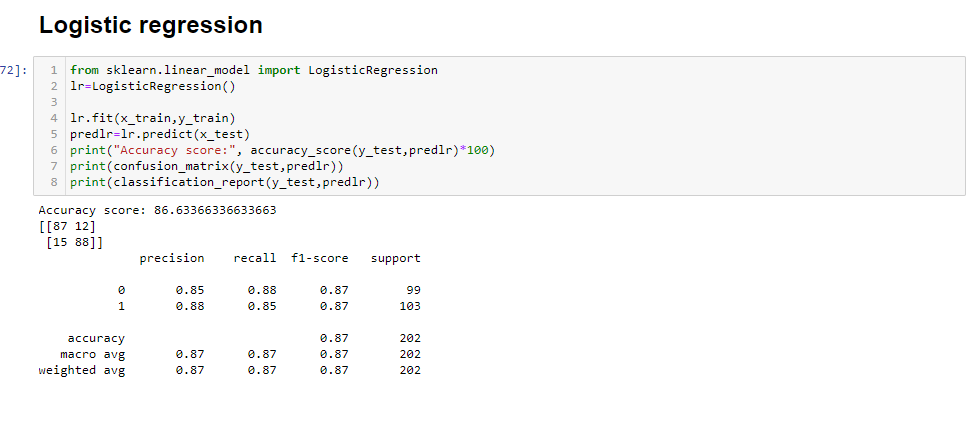
* We have created train test\_test\_split using the random state 287
* We will train all our above-listed model with this train test split
* After creating train\_test\_split we have checked the shape of the x\_train, x\_test, y\_train and y\_test.

**Importing all the models**



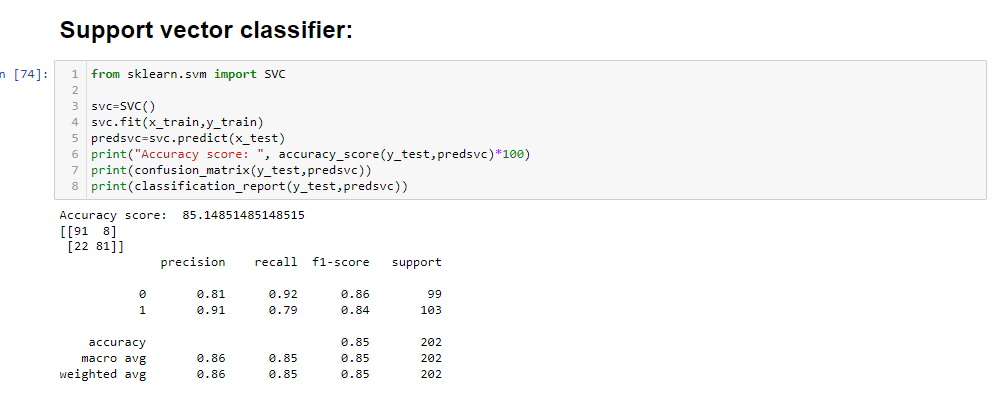
**Model Building**

1. **Logistic Regression**

****

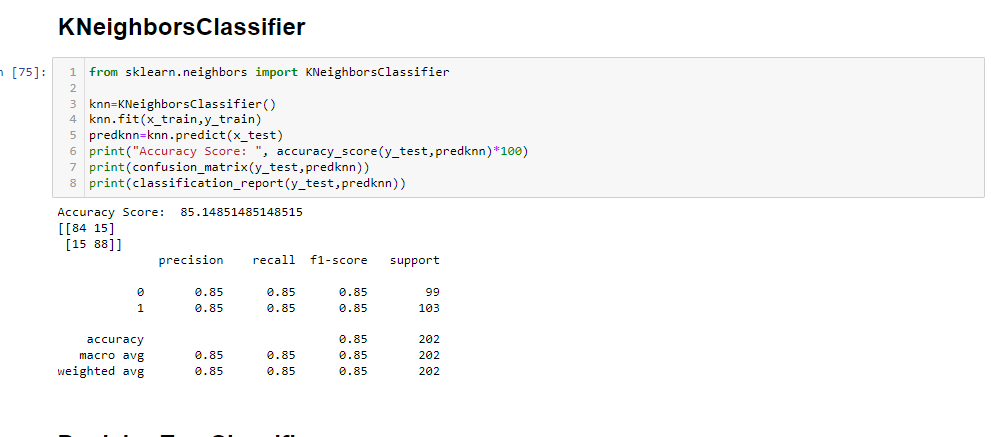
* Logistic regression model is giving us 86.6% accuracy score

1. **Support Vector Classifier**



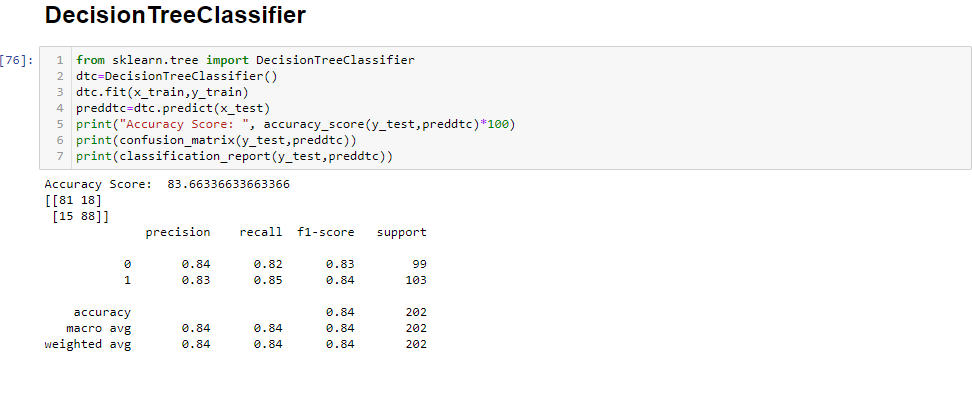
* Support vector classifier model is giving us an accuracy score of 85.1%

1. **KNeighbors Classifier**



* KNeighbors classifier model is giving us an accuracy score of 85.1%

1. **Decision Tree Classifier**



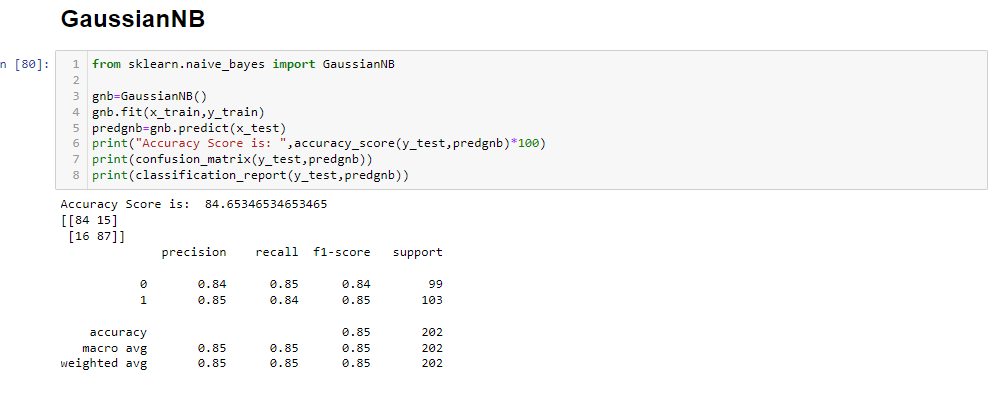
* Decision Tree classifier model is giving us an accuracy score of 83.6%

1. **Random Forest Classifier**



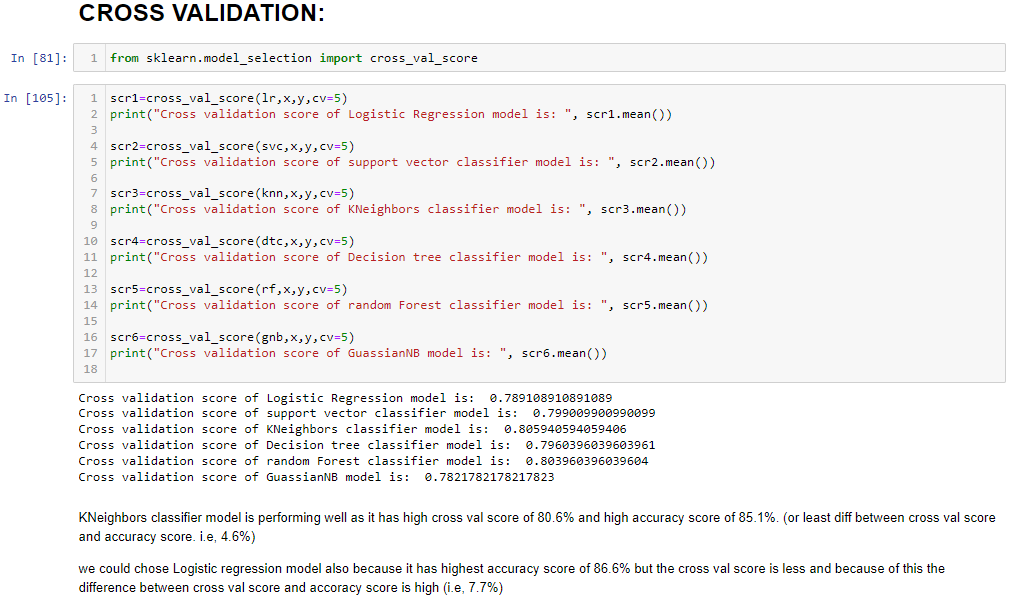
* Random Forest classifier model is giving us an accuracy score of 85.1%

1. **GaussianNB**



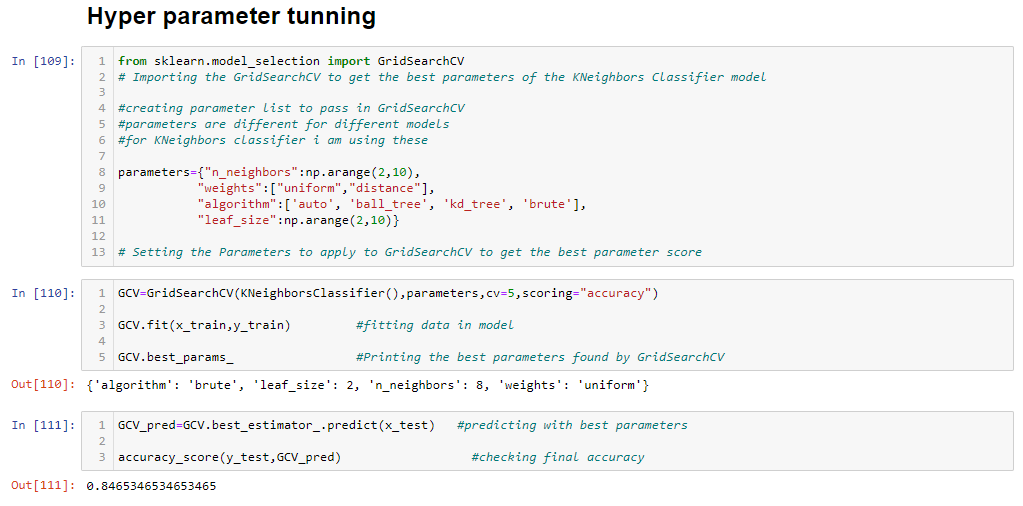
* GaussianNB model is giving us an accuracy score of 84.6%

**CROSS VALIDATION SCORE**



* After getting accuracy from all the models we have also checked the cross-validation score of all the models.
* We have given CV=5 so that every model will predict accuracy 5 times and give us an average of the 5 accuracies.
* In this case, KNeighbors Classifier model is performing well as it has a high cross-validation score of 80.6% and high accuracy score of 85.1% (or the least difference between cross-validation score and accuracy score. i.e. 4.6%).
* We could choose the Logistic regression model also because it has the highest accuracy score of 86.6% but the cross-validation score is less and because of this the difference between the cross-validation score and accuracy score is high (i.e., 7.7%)

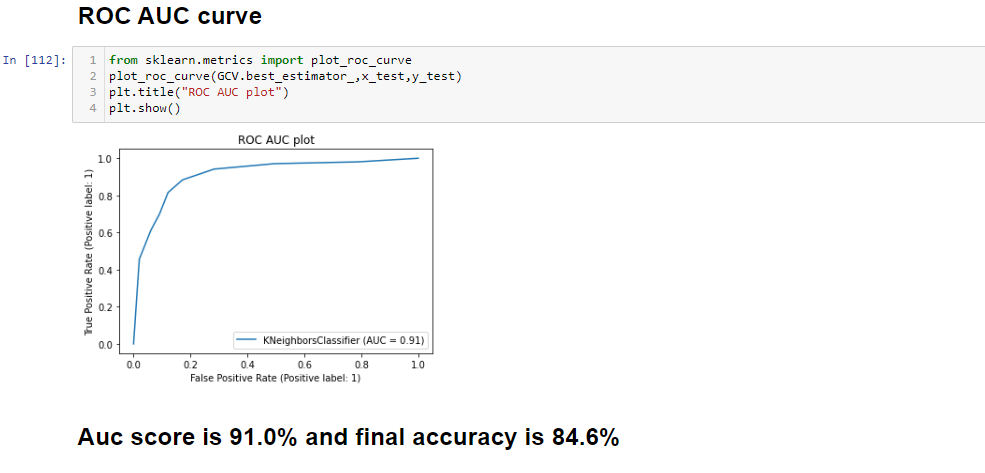
**HYPERPARAMETER TUNNING USING GRID SEARCH CV**



* In machine learning, optimizing or adjusting hyperparameters is a problem in choosing the best set of hyperparameters for the learning algorithm. Hyperparameters are the parameters whose values ​​are used to control the learning process. In contrast, the values ​​of other parameters (usually node weights) are learned.
* The same type of machine learning model may require different constraints, weights, or learning rates to generalize to different data patterns. These measurements are called hyperparameters and need to be adjusted so that the model can optimally solve machine learning problems. Hyperparameter optimization finds a tuple of hyperparameters that produces an optimal model that minimizes the predefined loss function of a given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss. Cross-validation is often used to estimate this generalization performance.
* Hyperparameters are important as they manipulate the general behavior of a gadget studying model. The remaining purpose is to discover the most advantageous aggregate of hyperparameters that minimizes a predefined loss feature to present higher results.
* For KNeighbors classifier model we have done Hyperparameter tunning and found that the accuracy score is 84.6%

# **PLOT ROC-AUC CURVE AND DETERMINE SCORES**

Measuring performance is an important task in machine learning. Therefore, for classification issues, you can rely on the AUC-ROC curve. If you need to see or visualize the performance of a multiclass classification problem, use the AUC curve (under-curve area) ROC curve (receiver operating characteristic). This is one of the most important assessments to check the performance of your classification model. It is also referred to as AUROC (area under receiver operating characteristics).



##### **As per the ROC AUC curve score, KNeighbors Classifier is the best-fit model.**

# **Model Selection: The Final Model**

In this step, we will save or serialize the final model which gives the highest performance into an object or pickle file.



## **Conclusion**

The final model performance is good with **Accuracy 84.6% and the ROC AUC score is 91%**

# **Concluding Remarks**

In order to extend this project to get a better final model, I would like to see if similar data can be used for further analysis. Doing so will allow you to further test your model and gain more insight into what works best in real-world situations. My model worked perfectly with my dataset, so it's worth asking if the dataset could be at risk. Therefore, testing the model on a different dataset may provide further validation.