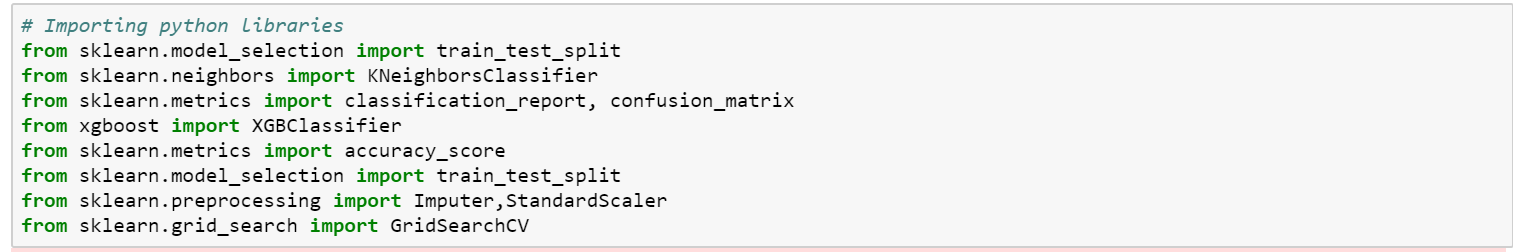
**Assignment 1**

**Dataset: Los Angeles Metro Bike share trip Data (**[**https://www.kaggle.com/cityofLA/los-angeles-metro-bike-share-trip-data**](https://www.kaggle.com/cityofLA/los-angeles-metro-bike-share-trip-data)**)**

**Step 1:**

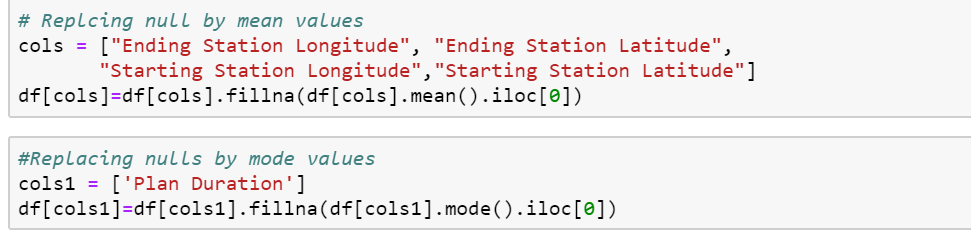
**Import Python Libraries:**





**Handling Missing Data:**

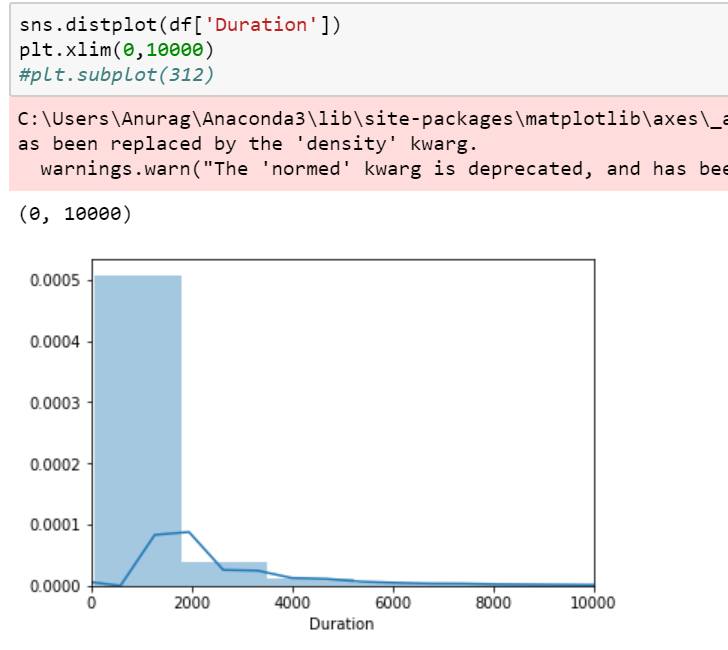
* Count the Number of Nulls present in the Dataset
* Replace Null from Numeric columns using Mean
* Replace Null from Categorical columns using mode function



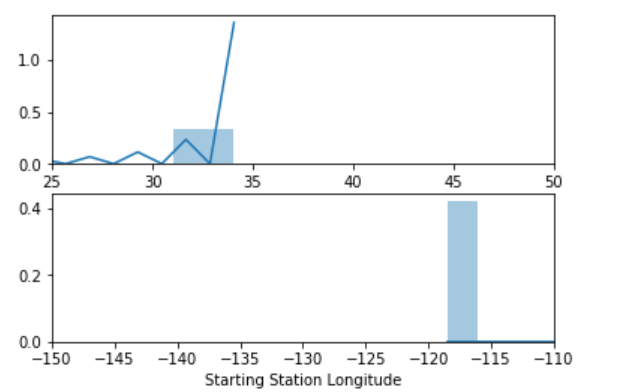
**Data Distribution:**

* Plot all the columns based on their data type

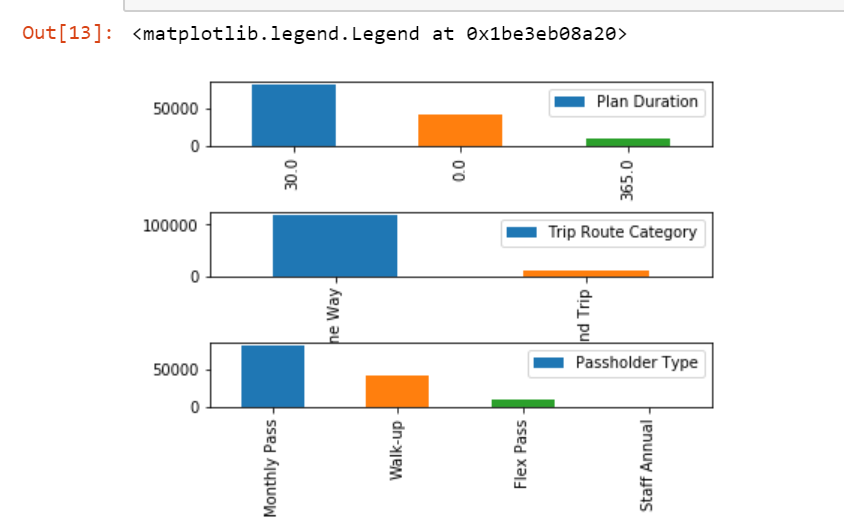
Distribution Plot for Duration:



Starting Station Latitude and Longitude:



Categorical Columns:



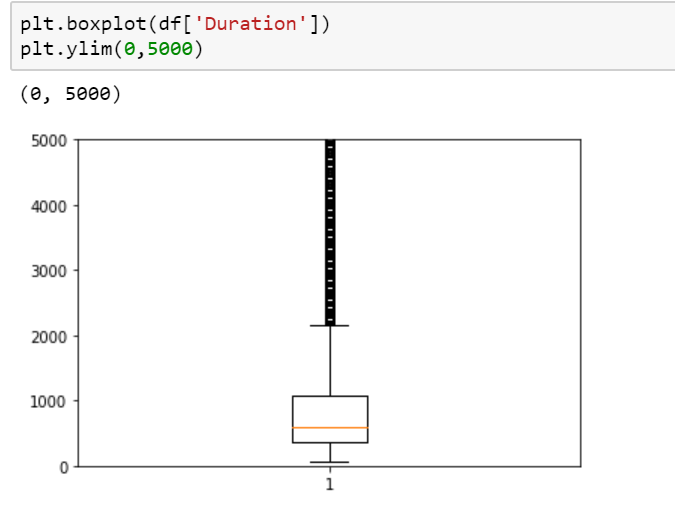
**Plot Summary:**

* Bar chart shows the frequency of each category present in the dataset
* There are 3 columns with the categorical values
* x-axis represents the unique values from each column
* y-axis represents the count of those values

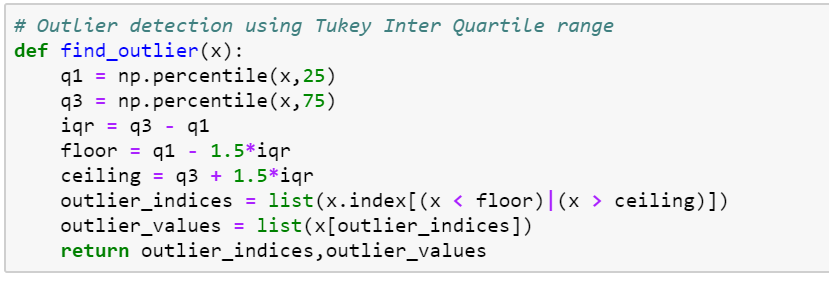
**Distribution Plot:**

* Data is more dispersed for 'Duration' columns as compared to 'Latitude' and 'Longitude' columns
* Values in 'Duration' column is distributed between 0 to 86000
* Values in 'Latitude' column is distributed between 30 to 34 while values in longitude column is distributed between -120 to -110

**Outlier Detection:**



**Outlier Detection Mathematically:**



**EDA Summary:**

* LA Bike share trip Data has total Missing Values = 2964
* Each Null value has been replaced by either **mean** or **mode**, depending on their data type
* Data set also has few columns with ID's, which is not relevant for predictive analysis
* Shape of the original dataset was Row=132427 and Column = 8
* Out of 8 columns Numerical columns = 5 and Categorical Columns = 3
* Categorical columns needed 0 and 1 encoding by creating dummy variables
* There were outliers present in Numerical columns which has been removed by calculating Tukey Inter Quartile range

**Analysis:**

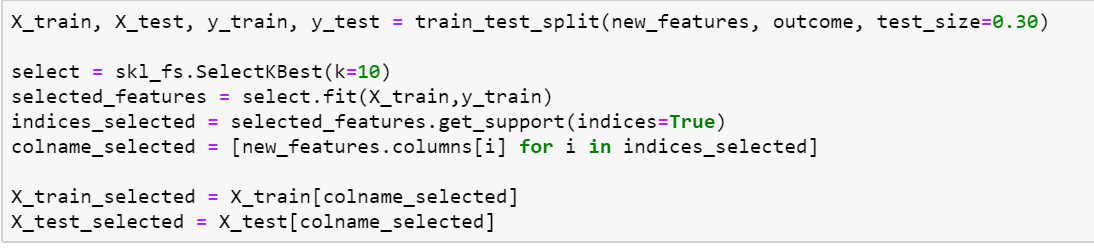
**Algorithms selected for predective analysis:**

* KNN(K-Nearest Neighbors)
* XGBoost(Extreme Gradient Boosting)

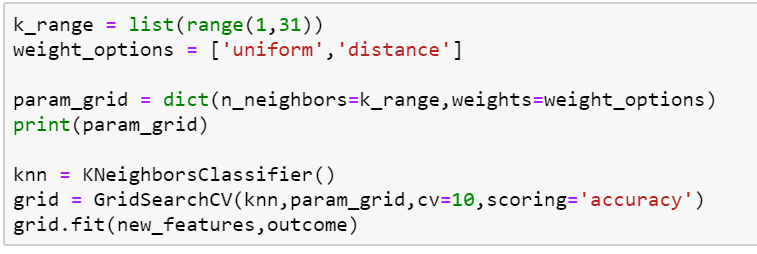
**Metrics to evaluate the algorithms:**

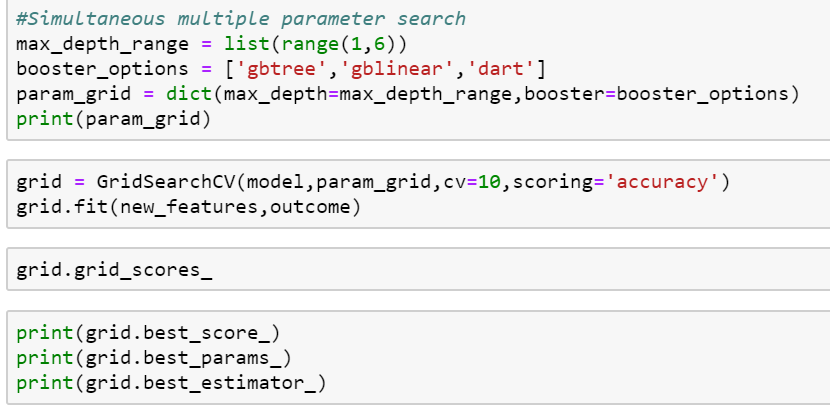
* classification report
* Accuracy
* Confusion Matrix

**Dimension Reduction and Feature Selection:**

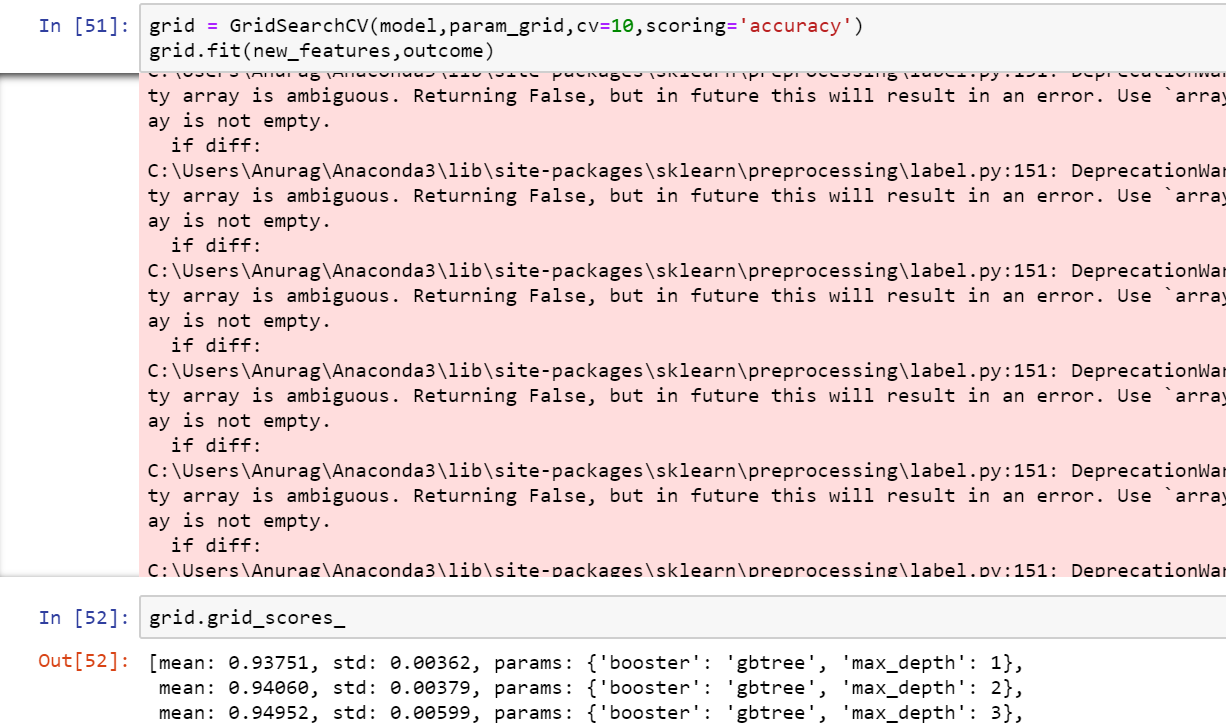


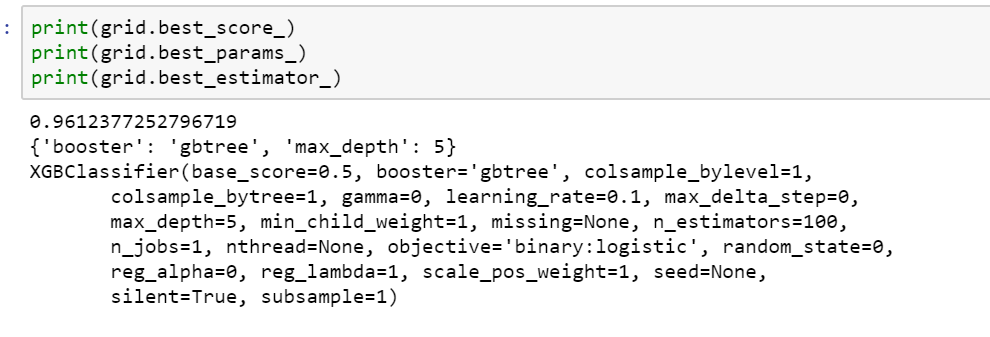
**Grid Search:**



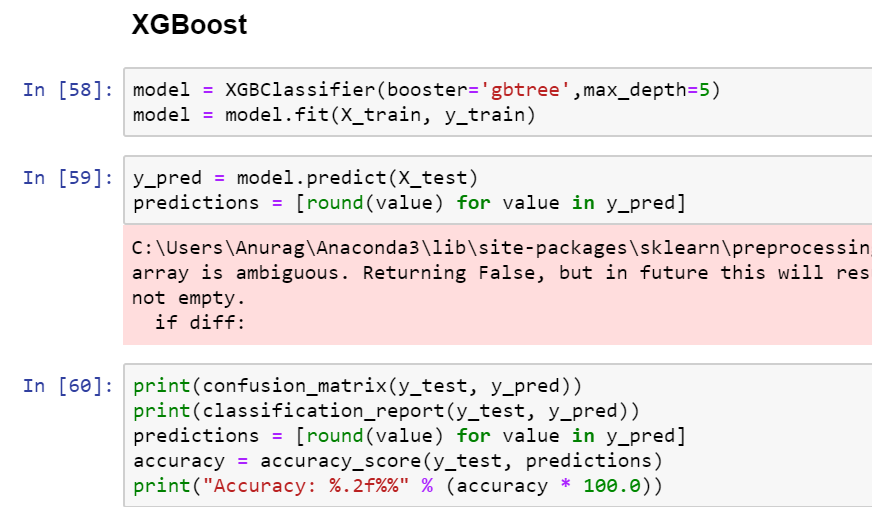


**Grid Search Result:**





**Re-Building the Model with Best Parameter:**



**SUMMARY:**

* Divided the data into train [70%] and test [30%] set
* Trained my model with the default parameters and with all the columns of training set
* Tested the accuracy using the test set
* Performed feature selection using \*\*sklearn.feature\_selection\*\* library
* Performed grid search operation using \*\*GridSearchCV\*\* library to find the best parameters
* Trained my model again based on the best features and best parameters
* Tested the accuracy using the test set

|  |  |  |
| --- | --- | --- |
| Algorithms | Accuracy | Accuracy after Features selection and grid search |
| K-Nearest Neighbors | 95.36% | 97.59% |
| XG Boost | 95.54% | 97.69% |

**Conclusion:**

**1. Based on the Accuracy and Classification reports it is clear that, XGBoost algorithm gives better result for this dataset**

**2. The accuracy and algoritm table indicates that, Accuracy of both the algorithms has been improved by more than 2% after selecting the best features and the best parameters**

**3. In order to achieve the improved accuracy these are the steps that we can follow:**

* Replace all the missing values using mean(Numerical values) and mode(Categorical Values)
* Remove the columns which contains IDs
* Remove the outliers from the dataset
* Select the best features from the dataset
* Choose the best hyper-parameters for algorithms