Subset of Million Song Dataset

# Introduction:

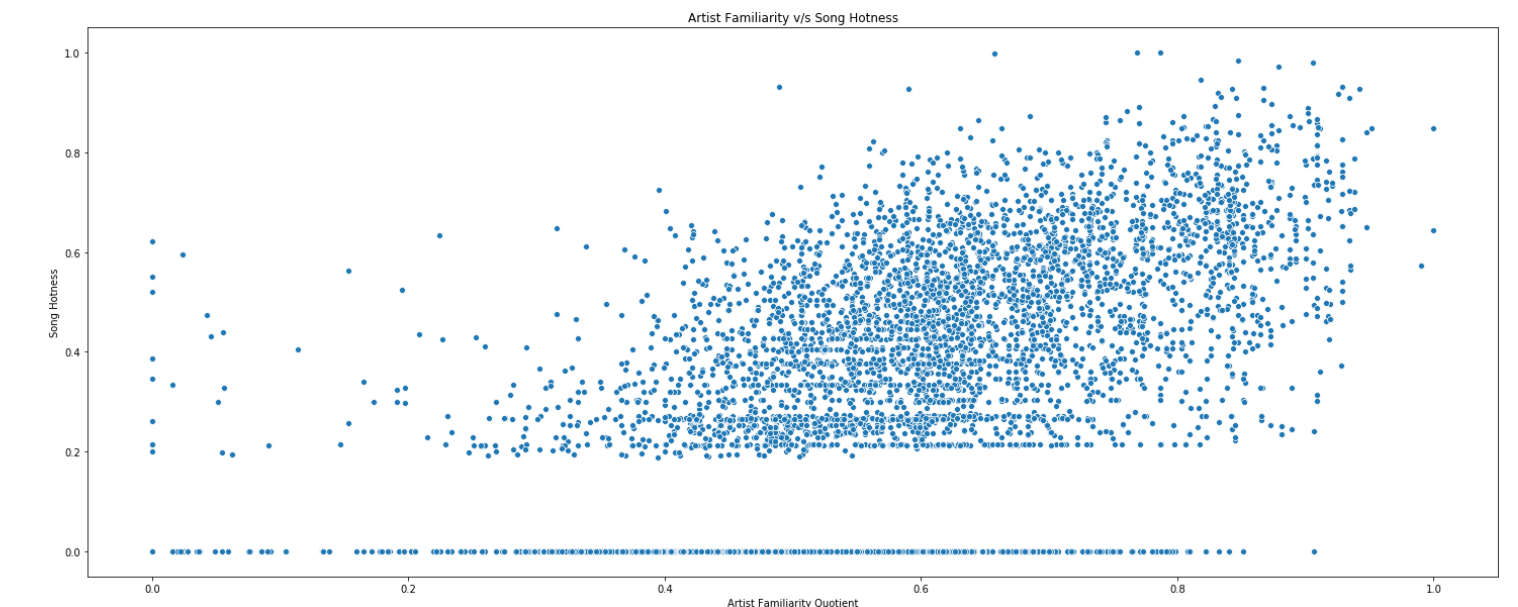
The document contains a detailed description of our analysis on the subset of Million Song dataset.

Our aim is to answer some basic questions like Are there certain characteristics for hit songs, what are the largest influencers on a song’s success, and can old songs even predict the popularity of new songs?

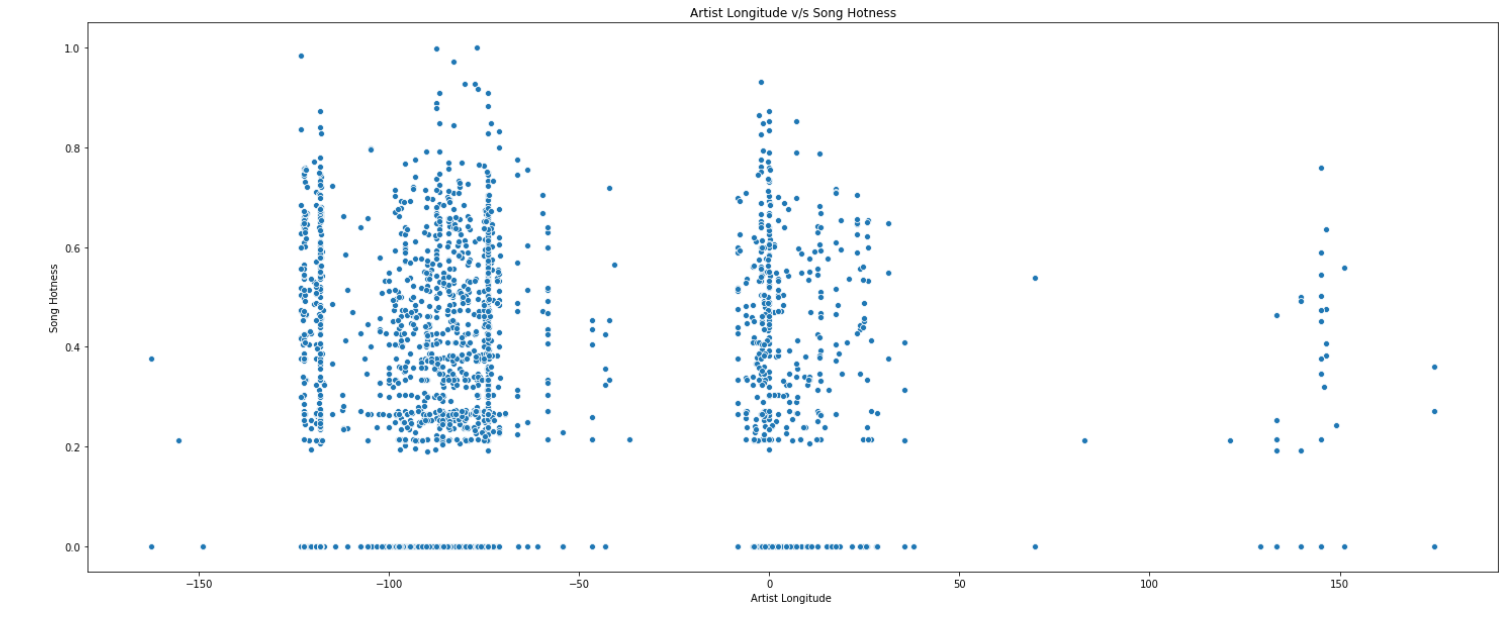
The [Million Song Dataset](https://labrosa.ee.columbia.edu/millionsong/)  was provided by Columbia, Spotify’s API .The dataset fetched from Spotify was quite huge and had lot many features , we took a subset of the same and some features that could help us train models.

The dataset contains features categorized by audio analysis, artist information, and song related features.

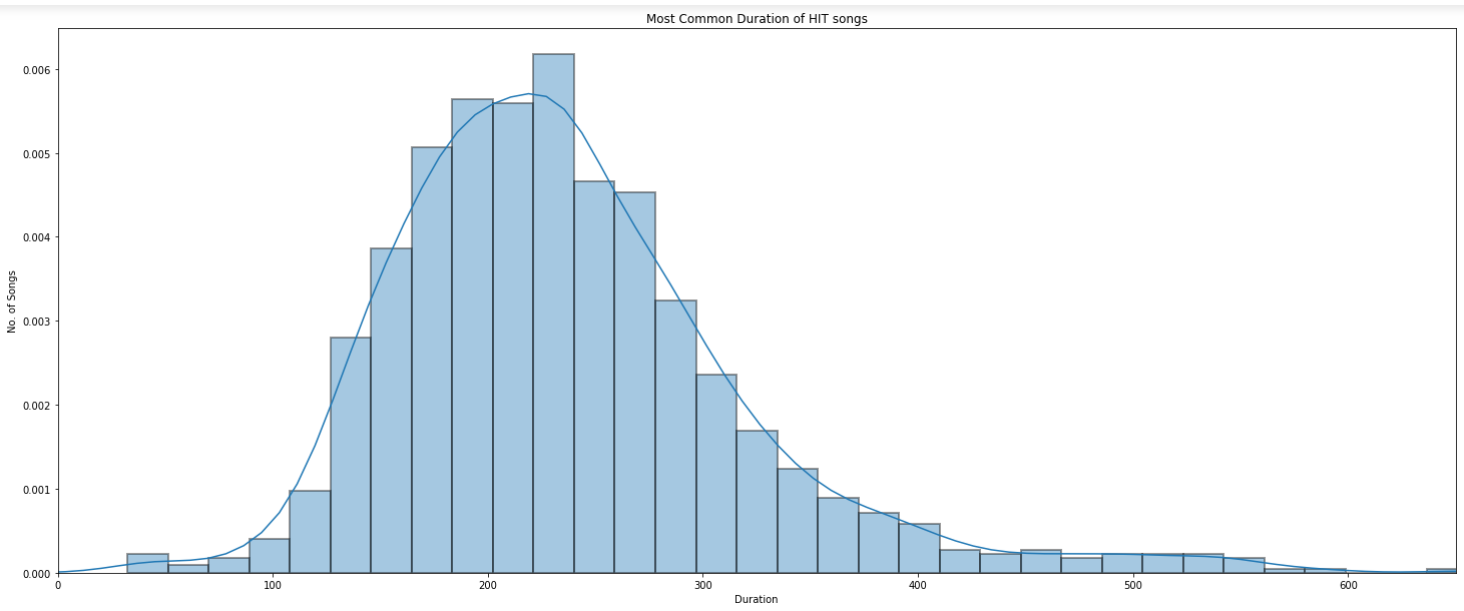
# Let’s look at all our graphs plotted as a part of Exploratory Data Analysis and see what we can infer from these.

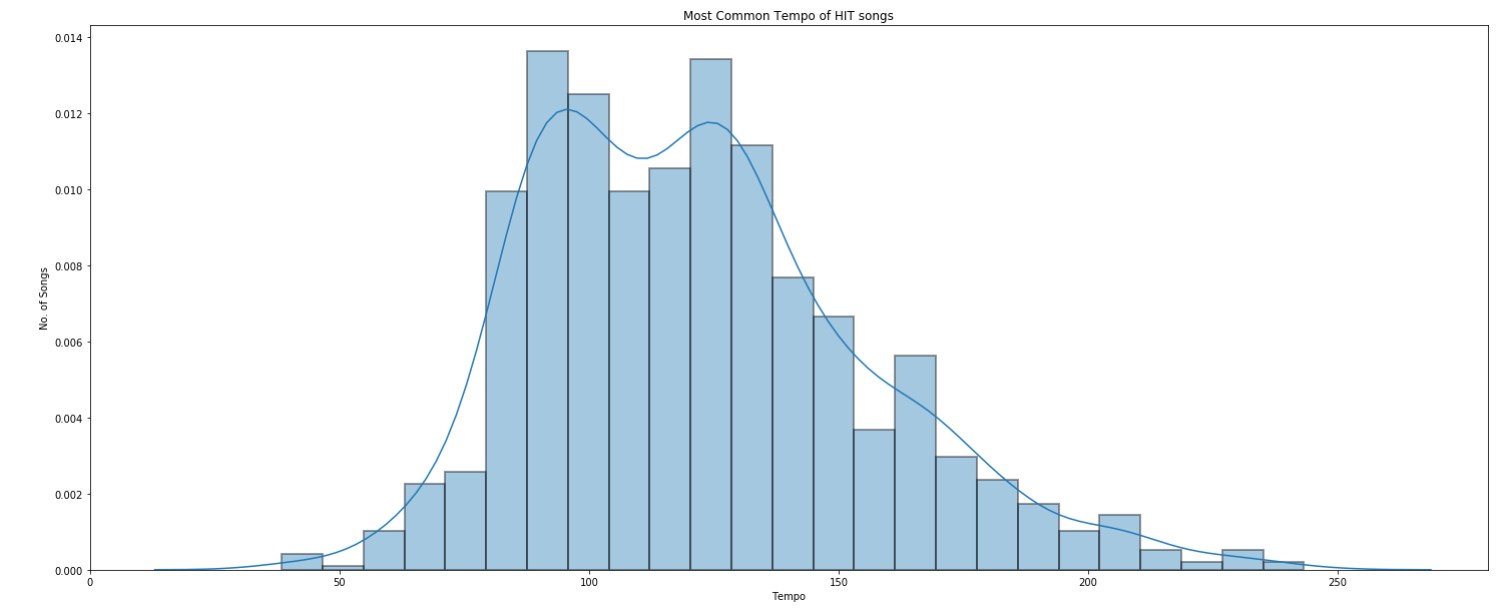


The above graph has been plotted between Artist Familiarity (describing how ‘familiar’ the artist is based) and song hotness, we can see that as the artist familiarity increases there is a higher chance of it being a hot song.

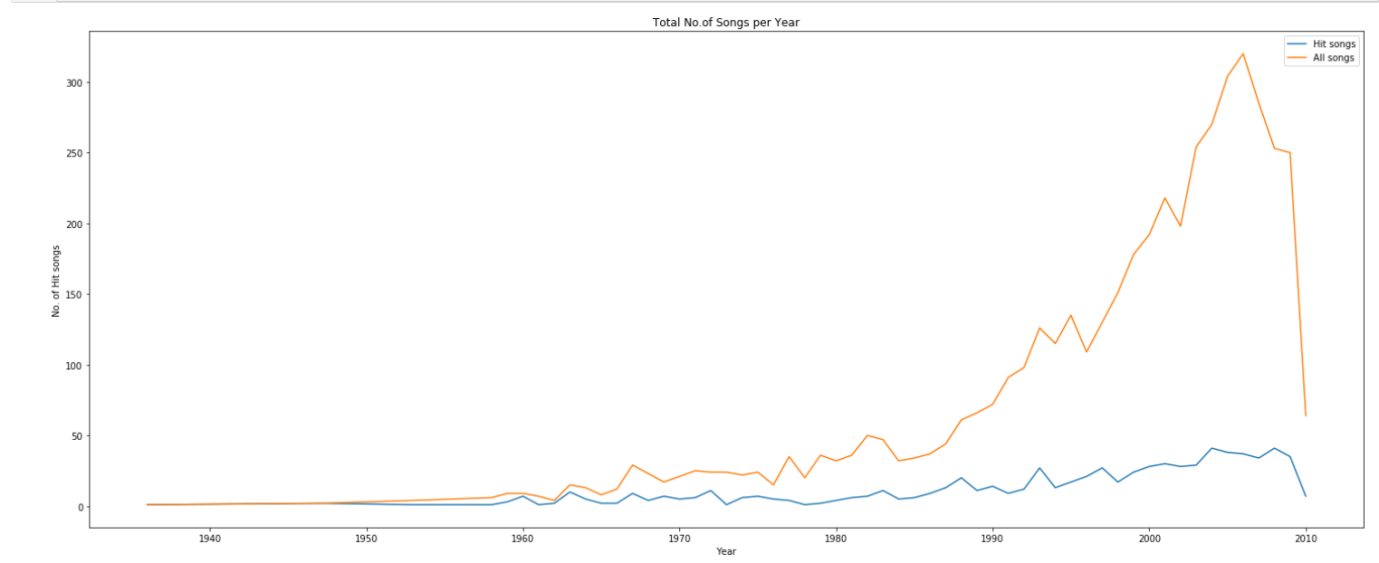


In the above graph we can see some interesting trends, there is generally more activity in the regions that also produce hits, we can see that the hits are centralized around these specific areas. Most of the activity is coming from the western side of the world, and on North America (as per the longitude values). Thus, we can expect the model to use this to predict whether a song is a hit.

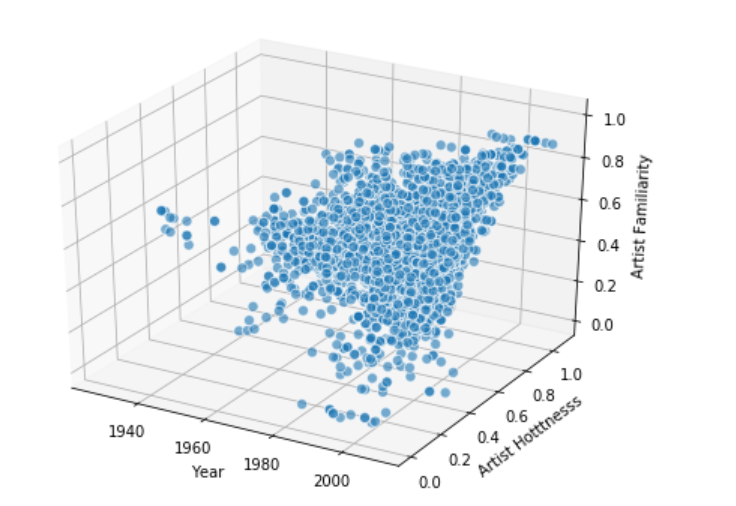




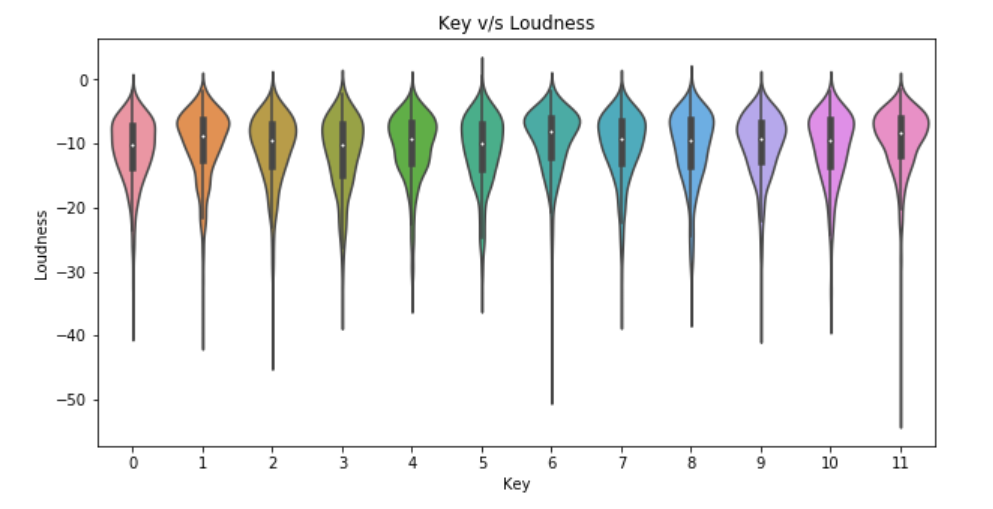
The above two graphs give us the distribution of features namely tempo and duration for Hit songs (determined based on bbhot feature). We can see that for tempo there was a range that hot songs commonly used, and there were two peaks within this range at about 100 bpm and 135 bpm. The duration of the hot songs was at about 200 seconds on average and this duration had a general range of 3 to 4 minutes.



The above plot shows us how the count of total songs and no. of hit songs is distributed. We can se the total no. of songs is quite huge in the range 2000-2010 and hence the no. of hits are also more , but overall the percentage of hit songs is less , as compared to years 1950-1960 the percentage of hit songs is more .Assuming that only this much songs are there we can say that in 1950’s the no. of songs were less and hence its probability to get hit was more as compared to the current scenario .



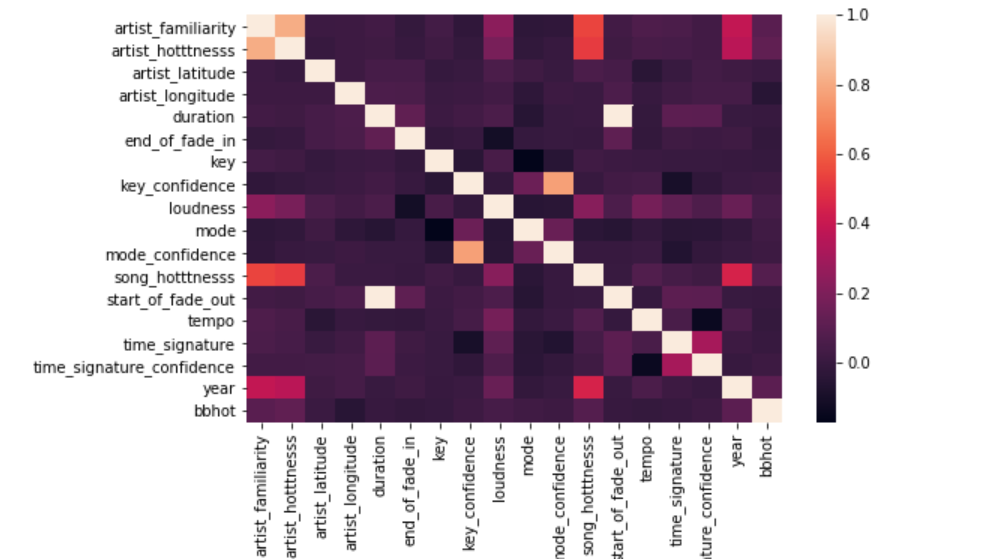
As per the above graph we can see that there is a growth in the artist familiarity value. We can see a larger density towards 1990’s. Artistfamiliarity value is approaching towards 1.0 around 2000’s.



Let’s see the relationship between key and loudness we can see that key is dense around loudness value -10, we can see that there is a loudness peak value for key 6, key11.

# Model Tuning and Pre-processing:

The prediction of popularity of a song is a classification problem, which can be handled by several machine learning algorithms like XGBOOST, RandomForest, K-Nearest Neighbours, Logistic Regression and lot more. Before applying any of these models one must be aware of the quality of data.



The above plot shows the correlation between the number of features. We can see that there is high correlation between start\_of\_fade\_out and duration i.e. .998, artist familiarity and artist hotness i.e.0.811. We should make sure to remove one of the columns to avoid the effect of multicollinearity on our data.

# Data Pre-processing:

Before applying any of the machine learning algorithms there are some keys points to look at and prepare data accordingly.

Handling Null Values: We need to make sure there are no null values in the model and drop the rows accordingly or go for imputations. While doing the same in our dataset, for the columns like artist\_latitude, artist\_location, title and artist\_longitude, the null values were not imputed and neither any rows were deleted it was not included while feature selection for model, whereas for column like artist\_familairty we have removed the null values(count=4) and imputed it with mean value.

Handling Categorical Data: Categorical variables are basically the variables that are discrete and not continuous. Ex — In our dataset key and mode though represented by a numerical value it’s a categorical data and needed to be handled in the appropriate manner. While some algorithms have the capabilities to handle categorical data appropriately and some do not, in our dataset we have used one hot encoding to handle the categorical columns like key and mode.

Multi-Collinearity: We could see from the above heatmap that there is a high correlation between start\_of\_fade\_out- duration and artist familiarity -artisthotness, to avoid the effect of multicollinearity we planned to use one of the columns from these two sets.

Rescaling Data: The great difference in the *scale* of the numeric features of data could cause problems when we attempt to combine the values as features during modelling and hence rescaling plays a major role, there are methods like min-max, standardization, normalization. For our dataset we use the scale method from sklearn module to scale the numeric features.

# Model Implementation:

The problem statement being considered for our dataset is a classification problem stating the fact whether a song will be popular or not depending on the feature of the song.

We started with the basic classification algorithms like Logistic regression, random forest, decision tree and later tried to implement XGboost to improve our model.

The main challenge is the dataset being considered is skewed and while implementing the algorithm we have to make sure there is a proper trade off between the accuracy and precision recall values, because if the skewness is ignoring the algorithms resulted in high accuracy but it doesn’t imply a good prediction for the minority class and hence ROC curve plays an important role in such scenario.

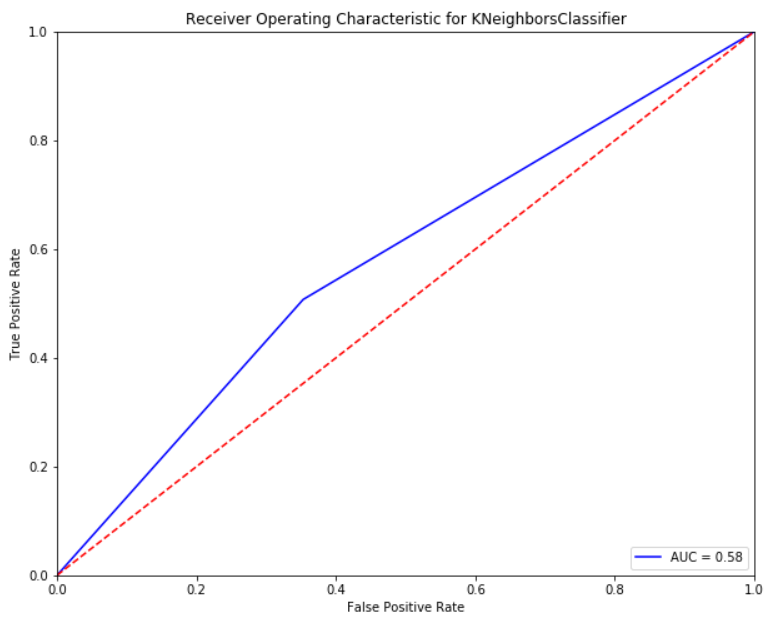
Since the dataset is skewed after splitting the dataset into train and test split, keeping ratio=0.7, we have applied smote on the training dataset. SMOTE works by creating new data points within the general sub-space where the minority class tends to lie. We could see better value in our ROC curve once we applied smote as now the skewness effect is reduced.

## Model Building Initial steps:

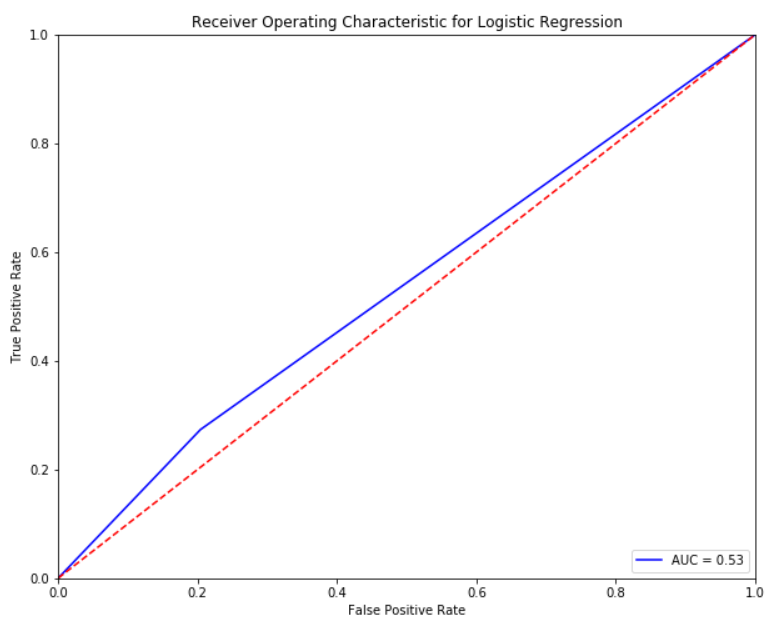
1. XGB Classifier: It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.
2. DecisionTree Classifier: It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision.
3. Logistic Regression: It is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.
4. KNeighbours Classifier: K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

|  |  |  |
| --- | --- | --- |
| Model Name | Accuracy | ROC/AUC Values |
| Logistic Regression | 0.59 | 0.58 |
| K-nearest Neighbours | 0.628 | 0.577 |
| XGBoost | 0.74 | 0.54 |
| Logistic Regression with Cross Validation | 0.88 |  |
| Decision Tree Classifier | 0.88 |  |
| XGB with Cross Validation | 0.87 |  |

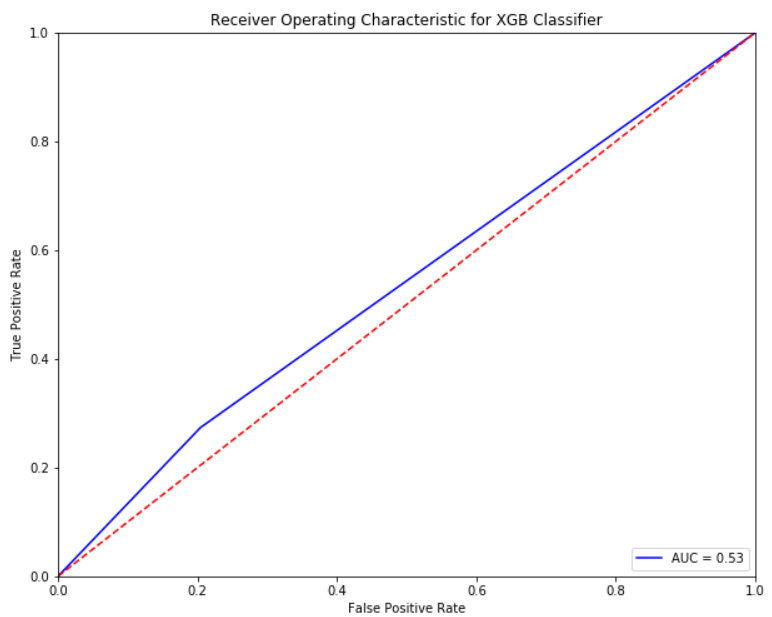
## Receiver Operating Characteristic for KNeighborsClassifier:



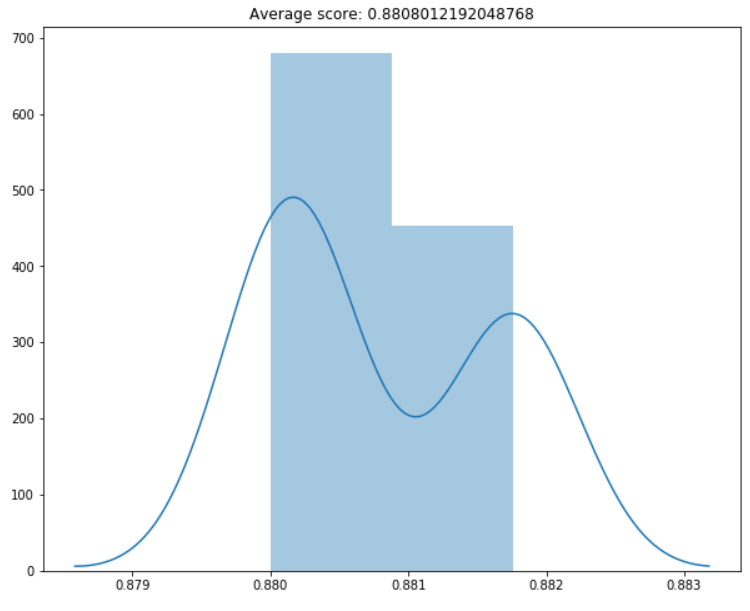
## Receiver Operating Characteristic for Logistic Regression:



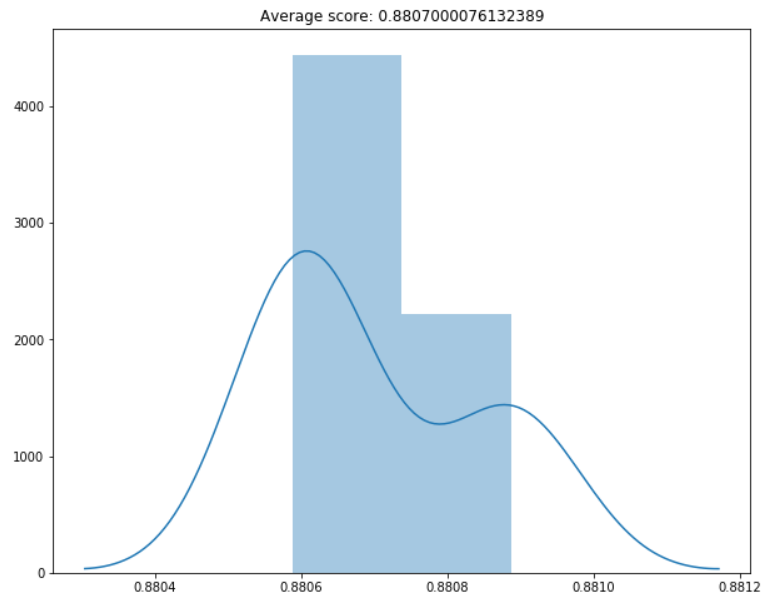
## Receiver Operating Characteristic for XGB Classifier:



## Logistic Regression along with cross validation:



## Decision Tree along with cross validation:



## XGB classifier along with cross validation:

