**Project N43**

Predict if a customer is satisfied or dissatisfied with their banking experience

**Done by**

**Umesh Raj Baidya (40AIML643-21/1)**

**Index**

* 1. **Chapter 1**

1.1. Introduction

1.2. Problem Definition

1.3. Dataset

1.4. Key metric/Key Performance Indicator to optimize

1.5. Real world challenges and constraints

1.6. Similar problems solved in Literature

* 1. **Chapter 2**

2.1. Introduction

2.2. Output Variable Analysis

2.3. Input Variable Analysis

2.4. Feature Engineering and Encoding

2.5. PCA and t-SNE representation of the data

* 1. **Chapter 3**

3.1. Introduction

3.2. Training and Test Data

3.3. Build Simple Models

3.4. Using Cost Sensitive Method for classifying Imbalanced Dataset

3.5. Conclusion and Going Forward

* 1. **Chapter 4**

4.1. Introduction

4.2. Advanced Modelling

4.3. Ensemble Modelling

4.4. Feature Importance

* 1. **Chapter 5**

5.1. Introduction

5.2. How to use Heroku for deployment

5.3. Tools and Library to solve this problem

5.4. Demo of this working application

5.5. Limitations of system and how we can improve

**Chapter 1**

**1.1. Introduction**

The project that I will work on is to ‘predict if a customer is satisfied or dissatisfied with their banking experience’. The dataset used for this project is from the Kaggle competition conducted by Santander Bank in the year 2016. To speak a little about Santander Bank, it was formerly known as Sovereign Bank, which is subsidiary owned by a Spanish company called Santander Group. It is multinational bank based out of Boston which operates mainly in North America, South America and Europe. To list a few of its services that includes retail banking, mortgages, corporate banking, cash management, credit card, capital markets, trust and wealth management, and insurance. As I could observe from the business model of the bank from their website, one of the point that it says it follows is ‘Customer-centric approach’, hence I can understand them coming up with such a problem in Kaggle, as I do feel that customer satisfaction is key measure of the success for any bank and helps them stand out. In this project I will work with hundreds of features to analyse the data and come up with a machine learning model to help predict if a customer is satisfied or dissatisfied with their banking experience.

**1.2. Problem Definition**

**1.2.1. What is the problem all about?**

The world is growing very fast with its many developments especially in technological front, this is indeed affecting how the businesses are getting done. The banking sector is also moving forward with this technological aspect providing my new services like mobile banking, internet banking, digital wallets etc. As we know that banking sector is a commoditized space, where everyone is providing almost same products and services without much room for rivals to compete in price. As a result they can’t just offer basic services to increase their tendency to have monopoly for their businesses. Banks now have to rely on other aspects like customer satisfaction to attract more customers and knowing more about their customer satisfaction will also help them prevent losing customers bringing a competitive advantage over others.

**1.2.2. Why is this an important problem to solve?**

Customers are the centre point for every business and banking sector is no different here. With the coming age in banking sector it is getting more competitive and there is a continuous need for them the get new customers. This makes customer satisfaction a very important aspect in the future of banking. A satisfied customer will help preach the bank’s services, which in turn benefits the bank. In addition to this a satisfied customer will also increase the profitability as the current customer enjoys the bank’s services. This is crucial as studies have shown that its much more cheaper and efficient to keep current customers than acquiring new ones. Apart from all this, a satisfied customer leads to loyal customer, and if the banks caters the needs for such customers properly then in time they will not be interested in the offerings by other competitive banks.

**1.2.3.** **Business/Real-world impact of solving this problem**

It is without any doubt that identifying which customer is satisfied and which customer is not satisfied will result in better financial for the banks. With this information they can help target their their future services/products for certain masses, and this also help identify which issues that they can overcome in their services to increase the customer satisfaction and retain more customers.

Identifying a satisfied and unsatisfied customer is a two fold process, where firstly for satisfied customer the bank can target new products and services which the customer will be more willing to take as he had an overall positive experience and this customer is more likely to bring new business to the bank by recommending the services to others. Secondly by identifying unsatisfied customers early, the bank can now take proactive steps to improve the customer’s happiness before its too late.

**1.3. Dataset**

**1.3.1. Source of the dataset**

The dataset for this project is taken from the Kaggle. It was problem given by Santander Bank in the year 2016. The train dataset consist of hundreds of anonymized columns both numerical and categorical, and the output is the TARGET column that we to predict. The task that we have in hand is to predict if a customer is satisfied or unsatisfied. This is clearly a binary classification problem when 1 in TARGET it represent that the customer is unsatisfied and when 0 in TARGET represent that the customer is satisfied. We are also given a submission dataset which is has all the columns same as train dataset except the TARGET column which we have to predict and submit.

**1.3.2. What is the positive class of interest over here?**

For us the positive class of interest is the unsatisfied customers which is also the minority class in our case.

**1.3.3. Explanation of each feature and data-point available**

This dataset is a black-box dataset where we don't have much information about the what each features represent. I did try some basic analysis like PCA and t-SNE on the data to see how much separable the classes are in two dimensions and also check minimum how many features are need to explain 95% or higher variance of the dataset (this can be seen in the python notebook attached).

To speak about the each data-point in the dataset, I will say that here each data-point represent a single customer and the features might represent some information about the customer like his age, salary, where he/she is currently living, employed or not etc. But we cant say this with 100% confidence as the dataset is designed specifically so that no information of the customer are leaked in any form and I am planning to use it as this with going deep in finding what each feature represent, but instead using univariate, bivariate and feature importance

**1.3.4. Data size, Data acquisition, Tools used and challenges foresee to process it**

The data I have used in this project is from Santander 2016 Kaggle competition, it is an open source dataset which has a shape of 76020 rows and 370 features, with all the features are either numerical data or categorical data. I could not find any null values in the dataset. I did find some columns which have a constant value those can be removed as they don’t bring any useful information in modelling, I could also see that few columns have exactly same values or you can say that say that their correlation is exactly 1 and I plan to go forward and remove them also. The main challenge that I foresee is the feature engineering part as the dataset we have is a black box.

**1.3.5. As the cost of misclassifying an unsatisfied customer is high, how am I going to tell this to the model or in other words, how am I going to assess my model's performance in this case?**

As I feel the cost of misclassifying an unsatisfied customer is high. We can maybe use the ‘class\_weight’ parameter in scikit learn to give more weightage to the minority class (unsatisfied customers) and less to the majority class (satisfied customer), which will penalize any misclassification on the minority class.

**1.4. Key metric/Key Performance Indicator to optimize**

**1.4.1. Business Metric definition**

While researching the internet for KPI for customer satisfaction I came up with various new terms to measure it like ‘Customer Satisfaction Score (CSAT)’, ‘Net Promoter Score (NPS)’, ‘Customer Effort Score (CES)’, etc. But these metrics seems to be grade based system, i.e. the customer give a score in between 0 to 10, I was not able to find any such scoring feature in the dataset given in Santander’s dataset. Also the features are anonymized hence coming up with any customized KPI is also a little difficult. Hence I decided to go with the already existing metrics in machine learning rather than coming up with specific business metric for this particular issue and explore which is best suited for the task at hand and would be easier to explain to the stakeholders in meeting.

As a side note it is worth mentioning that metric is different from loss function. Loss functions are functions which measure a model performance and train a ML model using optimization, and should be differentiable. Whereas metrics are used to monitor and measure performance with comparison to another model and don't need to be differentiable. However in some cases if the metric is differentiable then it can be used both as loss function and a metric.

**1.4.2. Metrics available for the task of classification**

The problem that we have to solve is a binary classification one, but we do have to keep in mind that the dataset we have is highly skewed towards the majority class which in our case is satisfied customers. First I will go through few of the options available to us then we will go through them to find which is the best for the task at hand.

1. **Accuracy**

Using ‘accuracy’ as evaluation metric in case of imbalanced dataset is generally known as a bad judgement, because for an imbalanced data of say 1:1000 (unsatisfied to satisfied) if we have a classifier which gives satisfied customer all the time, then also we will have a model accuracy of 99.9%. Hence it is not suitable for our dataset and our problem.

1. **Precision, Recall and f1-score**

As our objective in this project is to classify Next we have precision, recall and f1-score. Before beginning lets start with the definition of each:

**a. Precision**: Of the total predictions that were detected positive by the model, how many were actually positive.

**b. Recall**: Of the total observation of the positive class, how many were actually predicted as positive by the model.

**c. f1-score**: It is the harmonic mean of Precision and Recall.

It is observant that both Precision and Recall deals the minority class(positive class or unsatisfied customer in our case), hence it can be said that they are a much better choice of using as evaluation metric.

Both Precision and Recall range from 0 to 1 and our goal is to maximize both these values, and here f1-score is important as it seeks to balance both.

1. **Sensitivity, Specificity and G-mean**

If I talk about the definition of each:

**a. Sensitivity**: It is same as recall.

**b. Specificity**: Of the total observation of negative classes, how many were actually predicted as negative by the model.

**c. G-mean**: It is geometric mean of Sensitivity and Specificity.

For imbalanced classification Sensitivity makes more sense than specificity, but the G-mean can be used to find the sweet spot between the two.

1. **Ranking Metrics for Imbalanced Classification**

This metric is more concerned with evaluating classifiers based on how effective they are in separating classes. These require classifier to give a probability score, and using these scores different thresholds can be applied to test effectiveness of classifiers.

**a. Receiver Operating Characteristic (ROC) Curve**: It is a plot between true positive rate and false positive rate for set of predictions by the model under different thresholds. Each threshold is a point. Here the perfect model will be a point on the top left of the plot. The area under ROC Curve (ROC AUC) can be used as a single score to summarize the plot and can be used to compare models.

**b. Precision-Recall Curve**: Similar to ROC Curve in lots of way this curve focuses more on performance of the classifier on the minority class. Here the perfect classifier is represented as a dot in the top right corner. And similar to ROC AUC we can have Precision Recall AUC as single score to summarize and can be used to compare models.

1. **Probabilistic Metrics** **for Imbalanced Classification**

These are useful when we are measuring not only when the model fails but whether they have selected the wrong class with high or low probability.

**a. Log-loss**: For the expected values as y and predicted values as yhat, log-loss is given by the formula = - ((1- y) \* log(1- yhat) + y \* log(yhat))

**b. Brier Score**: It is more focused on the positive class (minority class), which makes it more preferable than Log-loss. It is given by the formula = (1 / N) \* Σ (yhat - y)^2

**1.4.3. Which metric is best? Its pro and cons with other alternatives**

As we have a highly imbalanced dataset it is not fruitful to use method like accuracy. An important disadvantage of using metrics like precision, recall, specificity, sensitivity, f1-score and g-means is that they assume full knowledge of the conditions under which the classifier will be deployed. In particular, they assume that the class imbalance present in the training set is the one that will be encountered throughout the operating life of the

Classifier. Next as I think we do not need the probability scores justifying the classification whether the customer is satisfied or not, hence I assume we can also skip the probabilistic method like log-loss and brier score. Lastly we have the Ranking metrics like ROC and Precision Recall Curves and AUCs. These do not have the limitation of the making assumption that about the class distribution as in case of threshold metrics like f1-score, g-mean etc.

**1.4.4. ROC Curve vs Precision-Recall Curve**

Finally if speaking about ROC curve, it is widely used but is not with problems. For imbalanced dataset with severe skewness and few data-point for minority class the ROC AUC score can be misleading. This is mainly because a small number of correct or incorrect predictions can result in a large change in ROC curve and ROC AUC score. And here the alternative of using Precision-Recall Curve comes into play.

Precision Recall curve is better than ROC curve in case of imbalanced dataset and also has the advantage that it focuses more on minority class.

Finally the main question is “If your question is: ‘*How meaningful is a positive result from my classifier given the baseline probabilities of my problem?’*, use a PR curve. If your question is, ‘*How well can this classifier be expected to perform in general, at a variety of different baseline probabilities?’*, go with a ROC curve.”

To me I thinks its best to use Precision-Recall Curve and PR AUC as the KPI of our model.

As we are indeed focused on how well we can classify the minority class(unsatisfied customers) and take necessary actions to prevent them losing as a customer. If in this process we make mistake some satisfied customers as unsatisfied it will not be that much of any issue, but we strictly don’t want to miss any unsatisfied customers.

**1.4.5. My choice for PR Curve over ROC- AUC Curve?**

As we use False Positive Rate (FP / TN+FP) in ROC AUC curve, and in case of imbalanced class we will have high True Negatives making FPR close to 0, pushing the ROC curve to the top left side and giving a higher AUC score which can be misleading when comparing different models or judging the accuracy of the model.

But in Precision Recall Curve we are not using FPR, we will use Precision (TP / TP+FP) and Recall (TP / TP+FN) which doesn’t have FPR in their formullas hence we will not have that misleading information as we have in ROC AUC Curve and is more suitable for ‘imbalanced’ data.

And as we don't have any True Negative in Precision and Recall, hence using PR AUC score is better in cases where True Negative doesn’t play an important role in our business problem or we may say that the majority class (class 0 or satisfied customers for our project) is not playing any important role, thus giving more emphasis on Minority class(unsatisfied customers or class 1).

**1.5. Real world challenges and constraints**

I do not see any as such real work challenge with the data set used, number of features are indeed high but not very high to create problems of high dimensionality. The dataset is also of not very high size that it will have processing with with normal 16gb of RAM in my laptop. The only problem I see is the black box anonymized features, but in the next of phase of the project “EDA and feature extraction” I plan to go through all the features and find some helpful insights to help in feature engineering.

**1.6. Similar problems solved in Literature**

**1.6.1. My first cut very naive approach to problems**

Till now we all know this is a binary classification problem, with a very highly imbalanced data. The very basic approach for my to solve this problem would as following:

1. Clean the dataset:

Find null points in dataset, use imputation to fill those missing data. Also find data which seems odd or misplaced and maybe be replace them using imputation.

2. Balance the dataset:

Use various methods for upsampling the data minority dataset. (SMOTE, Cluster based over sampling, Ensemble Techniques etc.)

3. Do some basic univariate or bivariate feature analysis.

Box plots, KDFs, Histogram, scatter plots between features etc.

4. Pre process the data:

Remove features which are highly correlated with other features. Remove constant features, duplicate features. Scale the dataset as most are numerical dataset. Do one hot encoding for categorical features.

5. Train classification models

I would train various classification models and check which model will give the best value for the evaluation metric

**1.6.2. Some existing solutions**

I found few very good solutions done by other on this very same topic in the internet. Providing the links below:

a. https://github.com/diefimov/santander\_2016/blob/master/README.pdf

b. https://github.com/kweonwooj/kaggle\_santander\_customer\_satisfaction/tree/master/34\_wpppj

c. https://cs230.stanford.edu/projects\_spring\_2018/reports/8290384.pdf

d. https://towardsdatascience.com/santander-customer-satisfaction-a-self-case-study-using-python-5776d3f8b060

e. <https://www.kaggle.com/c/santander-customer-satisfaction/discussion/20811>

f. <https://www.linkedin.com/pulse/santander-bank-customer-satisfaction-soham-mukherjee/>

g. https://medium.com/@nissanttiwari/santander-customer-satisfaction-82eacb41a2b3

**Chapter 2**

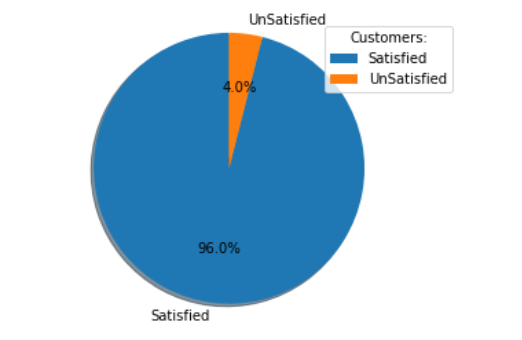
**2.1. Introduction**

In this phase I will perform exploratory data analysis on the dataset provided by Santander Bank in its 2016 Kaggle competition. We have total 371 features or columns in this dataset. In this 371 columns we also have a column for named ‘TARGET’ which contains the output of satisfied or unsatisfied customers that we are interested to predict. I will perform various univariate and multivariate analysis to find out which features are important for us to detected the target variable and will remove the features which are not that useful for our problem. I will also find which encoding method is useful for different features and why I will use them.

**2.2. Output Variable Analysis**

**2.2.1. Explore the ‘TARGET’ column.**

The TARGET column is what we want to predict after the final modelling. It equals one for unsatisfied customer and zero for satisfied customers. There are only two unique values for the TARGET column, hence we can say that problem we are solving is a ‘Binary Classification’ task. Below is a pie chart for TARGET column

Fig. 1: Pie chart of the TARGET column

In the pie chart we can see that of the total of 76020 customers 96% of customers are satisfied and only 4% are unsatisfied. Which clearly tells us that the dataset we have is highly imbalanced.

**2.3. Input Variable Analysis**

**2.3.1. Variables without the keyword ‘imp’, ‘ind’, ‘num’ and ‘saldo’**

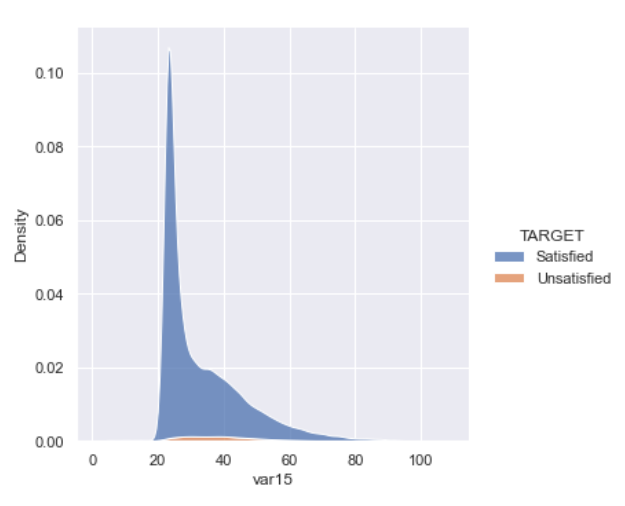
There are 5 variables which don’t have these four keywords. These 5 variables are ‘var3’, ‘var15’, ‘var36’, ‘var21’ and ‘var38’. I will try to dig deep into these 5 variables first and try analysing them using Univariate analysis and try to do some feature extraction on them.

1. **Var3**

Doing value\_counts on var3, I could see that we have 116 data-points in this feature which have value ‘-999999’. To my understanding these might be NaN and during imputing them someone gave them this value. Although this 116 data-points are small compared to total observation, I feel that removing them is not advisable. That is why I have replaced them with the mode of var3.

1. **Var15**

The density plot for var15 shows a spike at between 20 and 30 for satisfied customers. Which was because the value 23, 24 and 25 had constitute 40% of the total observations.

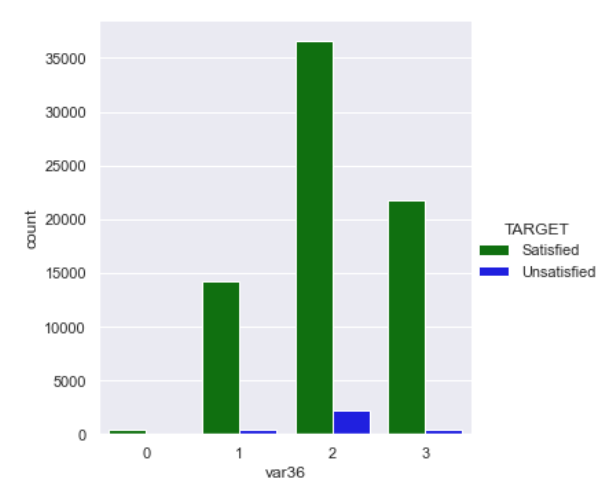
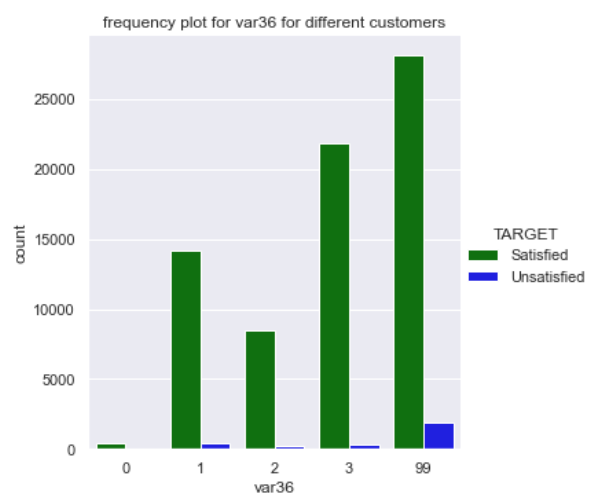
Fig. 2: Density Plot for var15 (age)

Looking further into the range of var15 I have come to the conclusion that this field could denote the age of the customers. Hence the KDE plot also shows that the most of customers of this bank will be in the age range of 23 to 35.

Another important insight about this field is that for unsatisfied customers this field has a minimum value of var15 is 23, which tells us that customers below the age of 23 are always satisfied customers.

1. **Var36**

Cardinality of var36 was simpler with only 4 values (0, 1, 2, 3, 99), but the value 99 in this is an outlier. We can see the count of each value in different classes with below image.

 Fig. 3: Frequency of var36 for different customers Fig. 4: Frequency plot after KNNImputation

As the value 99 is very high we cannot simply remove it, hence I will use the KNNImputer to find the 10 nearest and try to come up with a sensible value to impute this outlier.

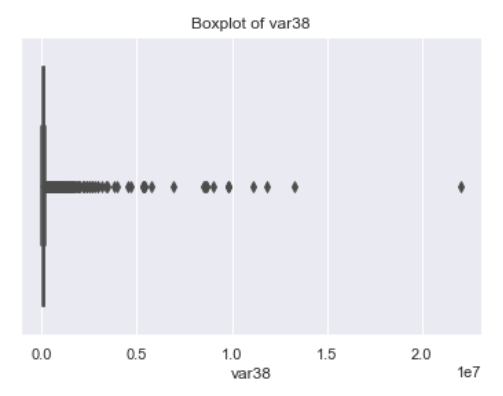
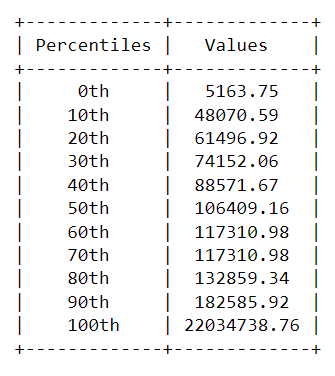
The value assigned using KNNImputer is ‘2.14’. Hence I will replace the 99 in this column with 2 (because its closest to the value available) for both classes. Which gives new frequency plot as below.

1. **Var21**

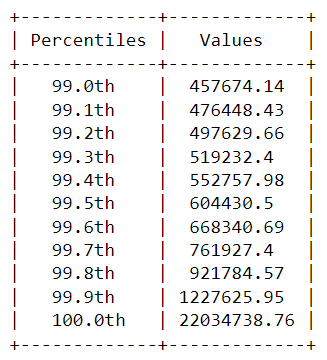
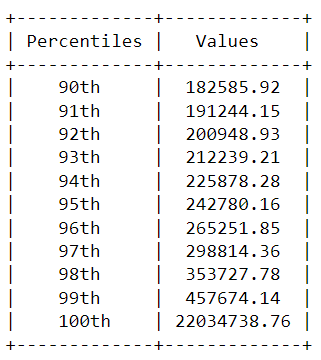
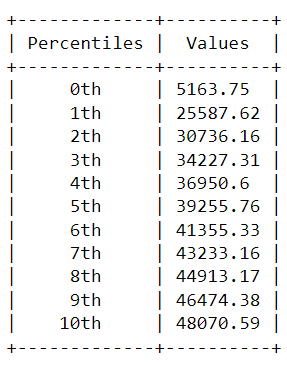
This field is kind of quasi constant ie the feature is almost constant. With most of its value as 0. The distribution of value 0 is almost same in both classes at about 98.8%. The only feature extraction I can use this feature is to distribute it to 0 for 0 and 1 for everything else.

1. **Var38**

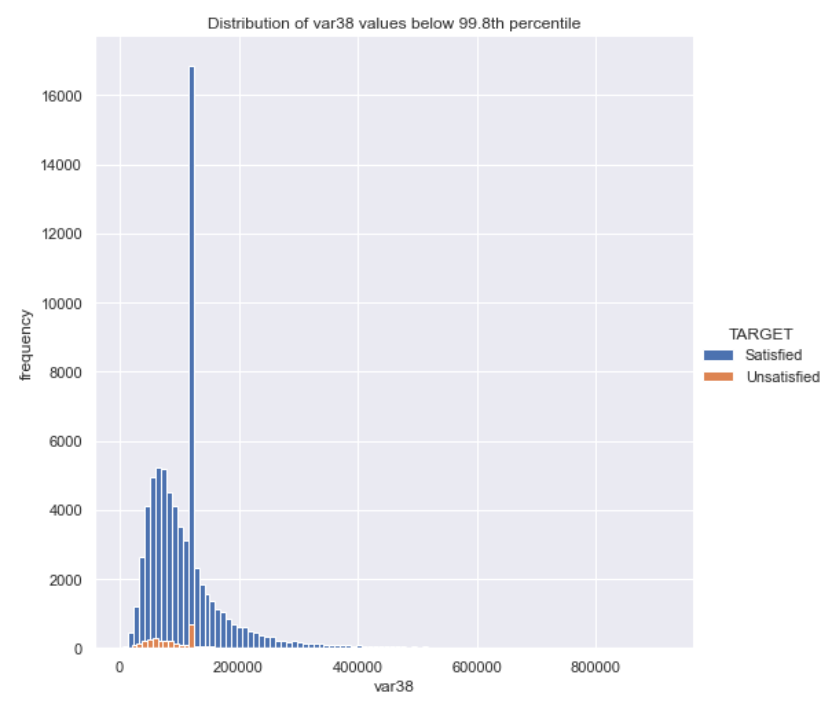
The mode of var38 is 117310, and according to a few literature available this value is considered close to average mortgage in Spain. As this is numerical variable we will start printing the percentile and see it.

Table 1: 0th to 100th Percentile of var38 Fig. 5: Boxplot on var38 (showing extreme values)

We can see a sharp jump between 0th and 10th percentile and also between 90th and 100th percentile. So lets zoom in between these two spaces.

 Table 2: 0th to 10th percentile Table3: 90th to 100th percentile Table 4: 99th to 100th percentile

As we still had a high jump between 99th and 100th percentile, I zoomed further. There was still a jump from 99.8th and 99.9th percentile, hence to I plot a histogram with this 99.8 percentile values to see the distribution of var38. Below is the image where I can see the histogram. It is a right skewed and looked similar to log normal distribution. Hence I went forward and did a log transform on those values also removed the mode value and plotted the distribution again and found it closer to Gaussian distribution. The shape of the distribution suggests that this might be net worth, or income of customer or other measure of money value making the suggestion of mortgage correct.

  Fig. 6: Distribution of var38 Fig. 7: Log transformed distribution of var38

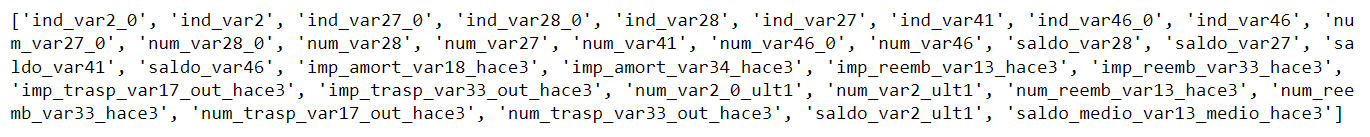
Since this Gaussian distribution makes sense and is more useful in ML models we will use the log transformed values for var38.

**2.3.2 Feature reduction**

There are certain features which make no useful contribution in our analyses. These features can be removed going forward and doing further analysis. Removing these features also help us in feature reduction process. Problems with these features are as below:

**2.3.2.1. Constant Features**

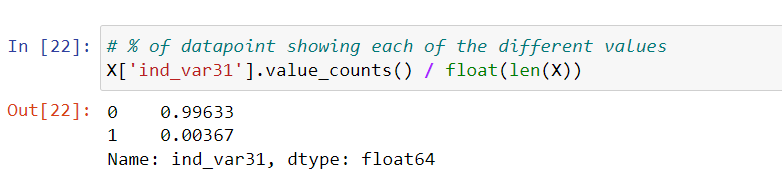
There are few features which only have one unique value in the entire column and for both the classes. These features do don’t help our model learn anything to help distinguish the classes, hence such features can be removed before hand. There were 34 such features. Below is a list of them:



**2.3.2.2. Quasi Constant Features**

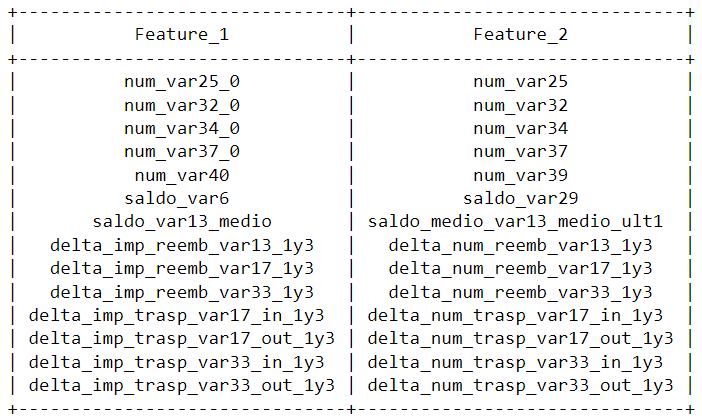
Quasi-constant features are those that show the same value for the great majority of the observations of the dataset. In general, these features provide little if any information that allows a machine learning model to discriminate or predict a target. But there can be exceptions. Identifying and removing quasi-constant features, is an easy first step towards feature selection and more easily interpretable machine learning models. I am taking my threshold to be 99% and I’ll remove all those features where we have same value for higher than 99% of the total data points.

Below is an example of quasi constant feature. We can see that greater than 99% of the observations show one value, 0. Therefore, this features is almost constant and can be removed from the dataset.

 Fig. 8: Example of a Quasi constant variable

**2.3.2.3. Duplicate Features**

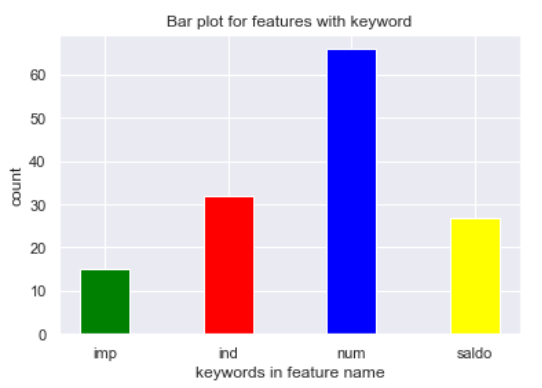
Certain features have exactly similar set of values. In such a scenario these two features are saying exactly similar things, and our machine learning model won’t learn anything insightful by keeping both these features. We can better off delete one of these features. Below is a list of duplicate feature pairs:

 Table 5: List of duplicate features

I have removed the feature\_2 column features and kept only the feature\_1 column features.

**2.3.3. Variables with the keyword ‘imp’, ‘ind’, ‘num’ and ‘saldo’**

If we simply get the count of feature name with these keywords, we can see that there is maximum features have keyword ‘num’ and minimum feature have keyword ‘imp’.

 Fig. 9: Count of features with keywords

I will go ahead and pick each keyword and then analyse features in those groups.

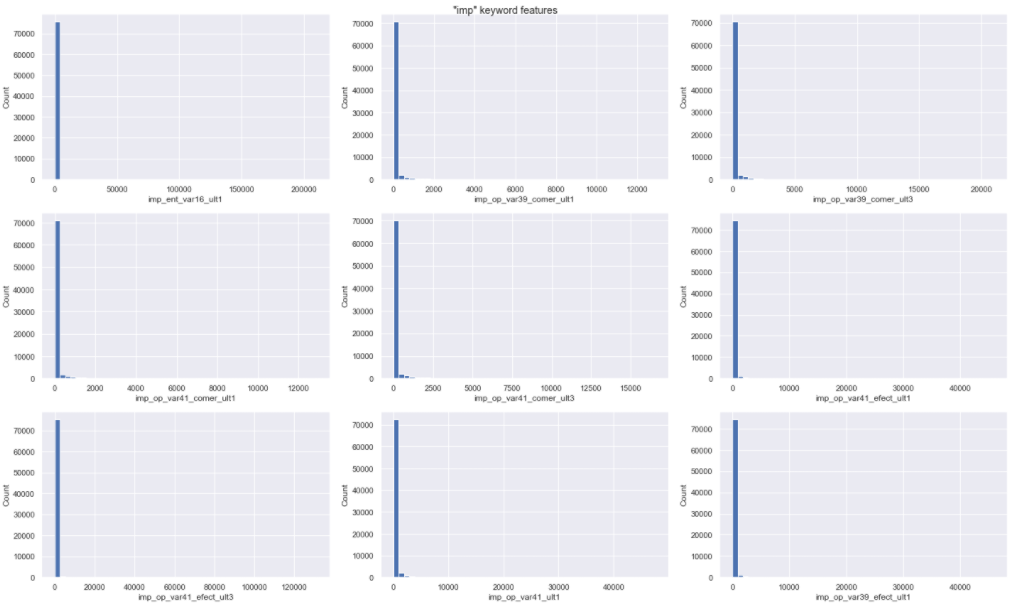
**2.3.3.1 imp keyword features**

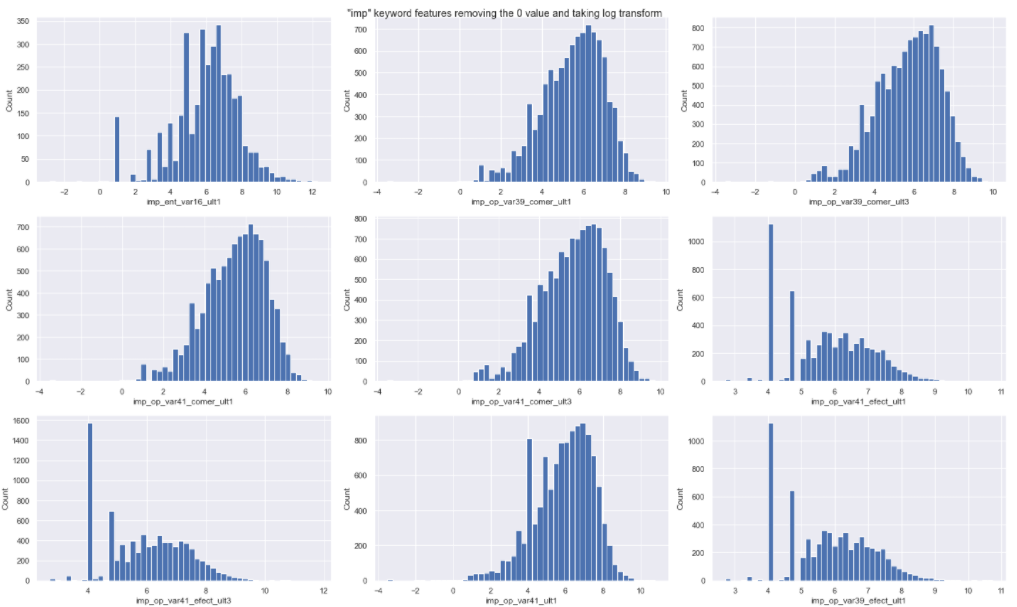
There are total of 15 features with this keyword and all of these features are numerical features. When plotting the histogram for all these features I could see a high concentration of the values around the value 0.

So I went ahead and removed the value 0 and then again plotted this histogram to see somewhat of a Pareto distribution. Next I tried using the log transformation to get a Gaussian like distribution for the features. It was evident that the features did show Gaussian like distribution after the log transformation.

It was only for the “delta\_imp\_aport\_var13\_1y3” that the distribution could not be that useful after the log transformation. The final conclusion is that I will using log for the transformation of these features which have the word ‘imp’ in its name.

Below is the histogram of these features before and after the log transformation.

Fig. 10: Histogram of 9 features with keyword imp in their name

Fig. 11: Histogram of 9 features after removing the 0 value and doing log transformation on them

**2.3.3.2. ind keyword features**

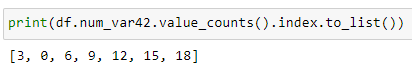
There were total of 32 features which had the keyword ‘ind’ in its name. Upon investigating it is found that all of these 32 features are categorical value. The cardinality of these features are just 2 for both the types of customers, so we don’t need to any one hot encoding on them.

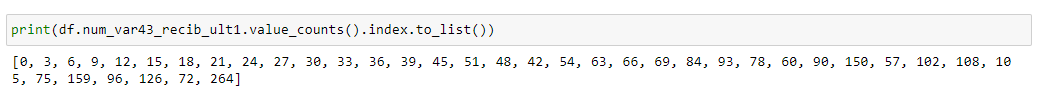
In conclusion I am planning to use these features as it is and do not apply any transformation or encoding on them.

**2.3.3.3. num keyword features**

There are total 66 features with the keyword num in them, and all of these 66 features are actually categorical features.

There was strange behaviour of this feature that I found. There unique values of these features are actually in multiple of 3. For e.g. we can see with these two features:





Hence to encode this behaviour I was planning to add a new feature which is count of value which are multiple of 3 across the row.

Another thing with these features are that the cardinality of few of these features are actually very high. For e.g. for num\_var45\_ult3 has a cardinality of 172. Having such high cardinality can result in very high feature space when we do one hot encoding, which might make the model run into the problem of curse off dimensionality and also increase the size of the dataset.

So I will reduce this cardinality by using a simple aggregating function. The idea is simple leave instances belonging to a value with high frequency as they are and replace the other instances with a new category which I will call others.

**2.3.3.4. saldo keyword features**

We have total 27 features with the keyword saldo in its name. All these 27 features are numerical feature with almost similar behaviour like we have for imp keyword feature hence I will use similar transformation for these features as well. I will be using log transformation to get a Gaussian like distribution for these features as well.

**2.4. Feature Engineering and Encoding**

**2.4.1. Correlated Features**

There is the correlation between the features which we want to remove to avoid overhitting in our modelling. I am planning to use a library feature engine to remove all the high correlated features. DropCorrelatedFeatures() works only with numerical variables. Categorical variables will need to be encoded to numerical or will be excluded from the analysis.

Fig. 12: Heat map of all the features showing some high correlated features

**2.4.2. Feature Engineering**

There are quite a few feature engineering tricks I found in the Exploratory Data Analysis that I will be applying over here and listing them. Also I will applying some other feature engineering method I found when reading the literature of this problem.

1. Age of customers below 23.

2. The value of var38 is its mode value or not.

3. Number of 0 in each column for a particular row.

This is useful as the data is very sparse and could contain some useful information.

4. Number of non 0 values in each column for a particular row.

5. New feature to store where we have 99 or -999999 in features var3 and var36.

This is useful as this gives us some information for missing or outlier value.

6. Features which are multiple of 0 for a particular row.

7. Var21 is 0 or not.

8. Added K-Mean clusters as suggested on one of the literature reviews at I had read.

Here I have used the clusters [2,4,6,8,10].

9. One hot encode all the categorical features.

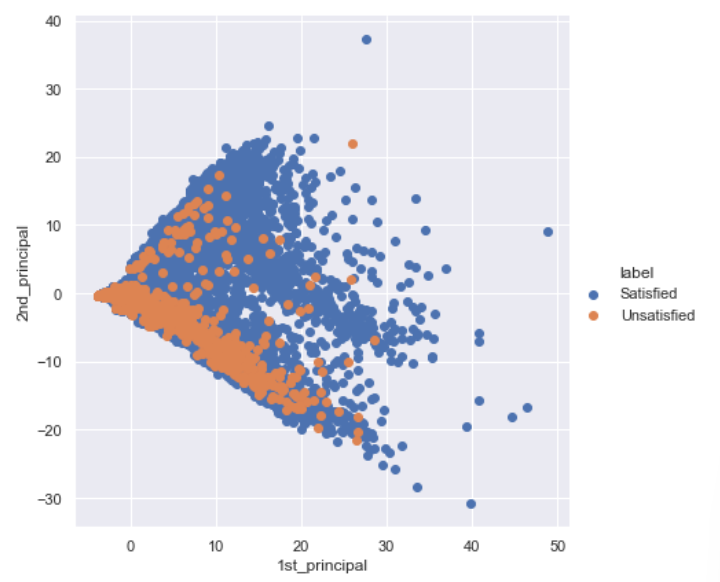
10. Scale the numerical features using standardization.

**2.5. PCA and t-SNE representation of the data**

Now after the feature engineering and standardization of data we can finally check the PCA and tSNE representation to see how the data of high dimensionality looks in 2 dimensionality.

**2.5.1. PCA:**

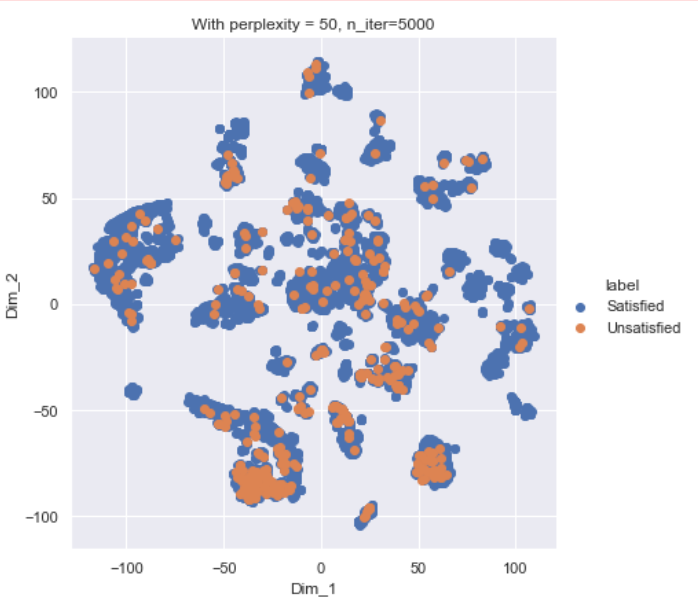
Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analysing data much easier and faster for machine learning algorithms without extraneous variables to process. So to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

Fig. 13: Final PCA representation

**2.5.2. t-SNE:**

t-SNE is also a dimentionality reduction techniwue like PCA, but it is newer and developed in 2008.In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.

t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance.

Fig. 14: t-SNE representation of the data

**Chapter 3**

**3.1. Introduction**

In this phase I will train various models from with and without the class weights to handle the imbalance in the dataset. I will try to plot the metric of ROC AUC with both train and test data and see which cases we have a better analysis. I will also do hyper para meter tuning to get the best values of class weights to penalize the models if they deviate.

**3.2. Training and Test Data**

Finally after the EDA and Feature Engineering done in phase 2 I have come up with 3 different datasets to work with. Below is the description for these three types of dataset:

1. First Dataset is the simple dataset which I receive after feature engineering. It has total of 225 features.
2. Second Dataset is received after the result of reducing the cardinality of the ‘num’ keyword from the first dataset. It has total of 167 features. This reduction of features is mainly due to reducing the cardinality of ‘num’ features.
3. Third Dataset is received after I take log on the features ‘var38’, ‘imp’ keyword features and ‘saldo’ keyword feature as discussed in phase 2. This is done on dataset 2. It also has total of 167 features.

As part of train and test data I have done stratified sampling so as to have similar ratio of satisfied and unsatisfied customers in both training and testing dataset.

**3.3. Build Simple Models**

I’ll start by building simple Logistic Regression, SVC, Decision Tree Model plot the ROC curve and chec the value for AUC and will also be looking at the confusion matrix for the same.

**3.3.1. Logistic Regression**

Bluntly looking at the plot and AUC score we may say that the score looks good, but if we then look at the Confusion Matrix on training and test data it helps Identify that the model is behaving as a dumb model and always or mostly predicting class 0 or satisfied customers.

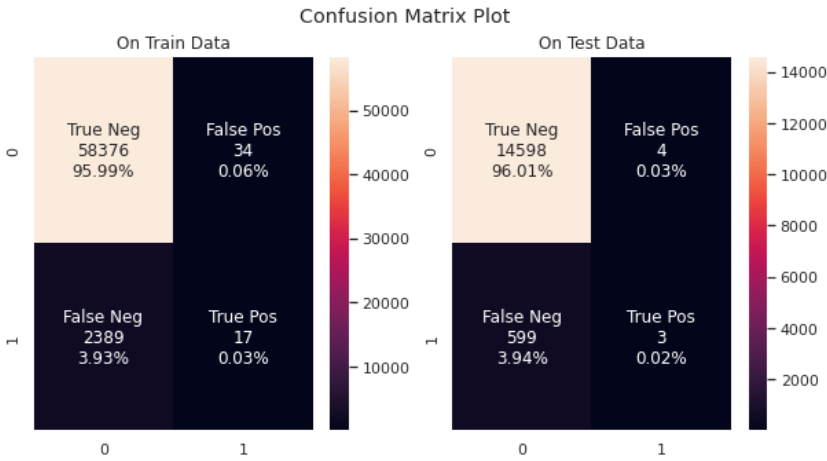
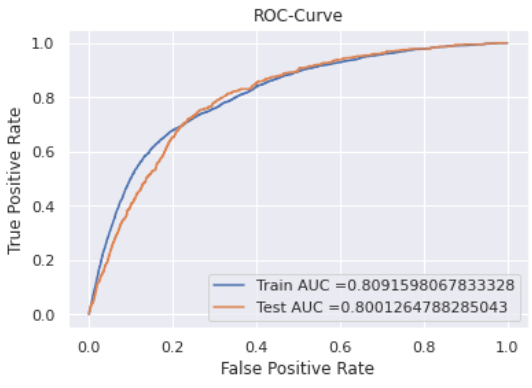


Fig 1: ROC Curve and Confusion Matrix after Logistic Regression on dataset 3

**3.3.2. Support Vector Classifier**

Similar to Logistic regression using simple SVC for imbalanced does not give the best result. The ROC curve does look promising but the result that we get in confusion matrix really bring the picture into place that the model trained is a dumb model which is giving Class 0 all the time.

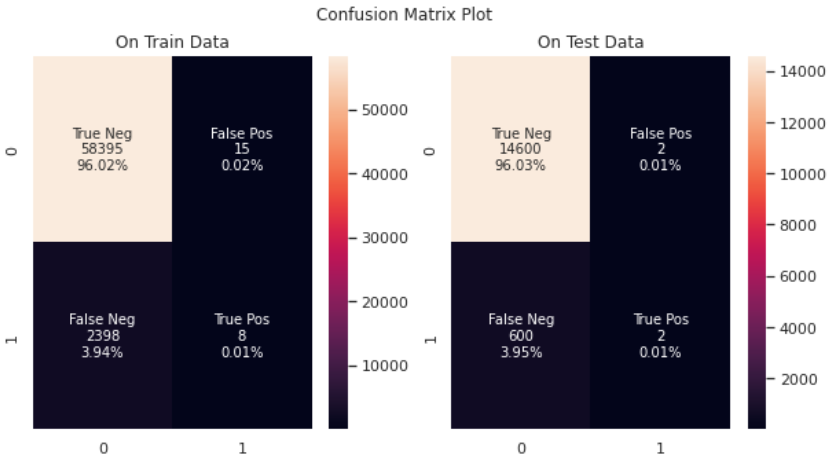
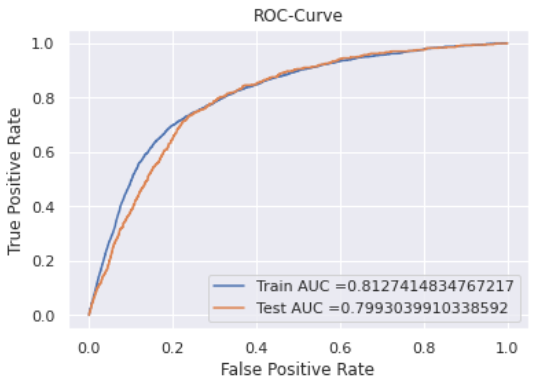


Fig 2: ROC Curve and Confusion Matrix after SVC on dataset 3

**3.3.3. Decision Tree**

With the class weights I could see a very high difference between the AUC score for train and test, this clearly states that the model is over-fitting on training data, and the low value on test data for True Positive clearly shows us that the model is not behaving as needed due to the high imbalance on the data.

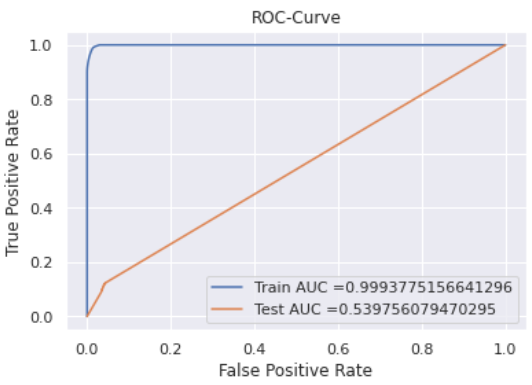


Fig 3: ROC Curve and Confusion Matrix after Decision Tree on dataset 3

**3.4. Using Cost Sensitive Method for classifying Imbalanced Dataset**

In the above models and their results it is clearly evident that the models were not working at all. The results were heavy driven by the majority class and this leading to abrupt and misleading ROC curves and AUC scores.

Cost-sensitive learning is a subfield of machine learning that takes the costs of prediction errors (and potentially other costs) into account when training a machine learning model. It is a field of study that is closely related to the field of imbalanced learning that is concerned with classification on datasets with a skewed class distribution. As such, many conceptualizations and techniques developed and used for cost-sensitive learning can be adopted for imbalanced classification problems.

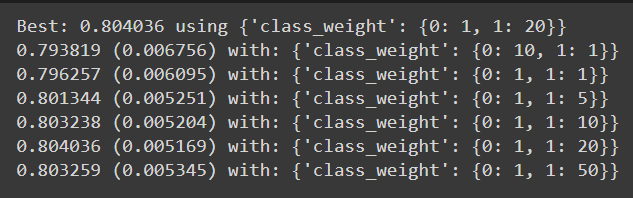
In cost-sensitive learning, a penalty associated with an incorrect prediction and is referred to as a cost. We could alternately refer to the inverse of the penalty as the benefit, although this framing is rarely used.

**3.4.1. Grid Search Weighted Logistic Regression**

Using a class weighting that is the inverse ratio of the training data is just a heuristic. It is

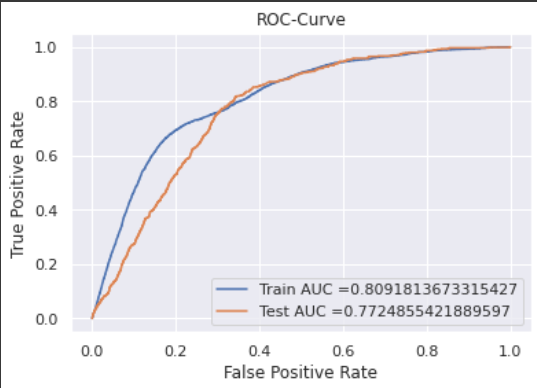
possible that better performance can be achieved with a different class weighting, and this too will depend on the choice of performance metric used to evaluate the model. In this section, we will grid search a range of different class weightings for weighted logistic regression and discover which results in the best ROC AUC score. These values for weights will be based on our prior knowledge that the class 1 is minority class We will try the following weightings for class 0 and 1:

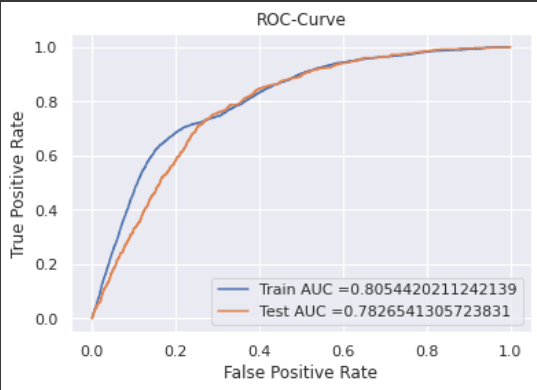
* Class 0:10, Class 1:1
* Class 0:1, Class 1:1.
* Class 0:1, Class 1:5.
* Class 0:1, Class 1:10.
* Class 0:1, Class 1:20.
* Class 0:1, Class 1:50.



With the above data it is evident that the best class weights according to the model turns to weight 1 to class 0 (Satisfied Customers) and weight 20 to class 1 (Unsatisfied Customers).

Next we will look at all the Logistic Regression Model build on all the three datasets we have with the best class weights and see how the roc\_auc\_score looks for each.





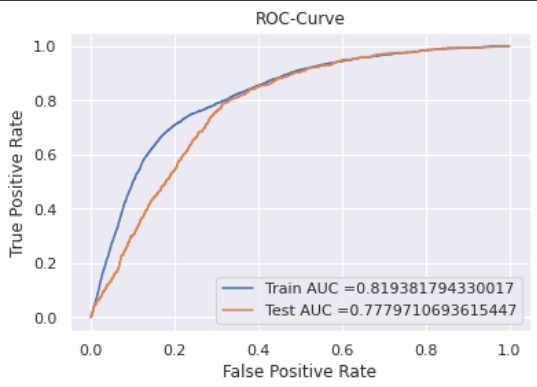
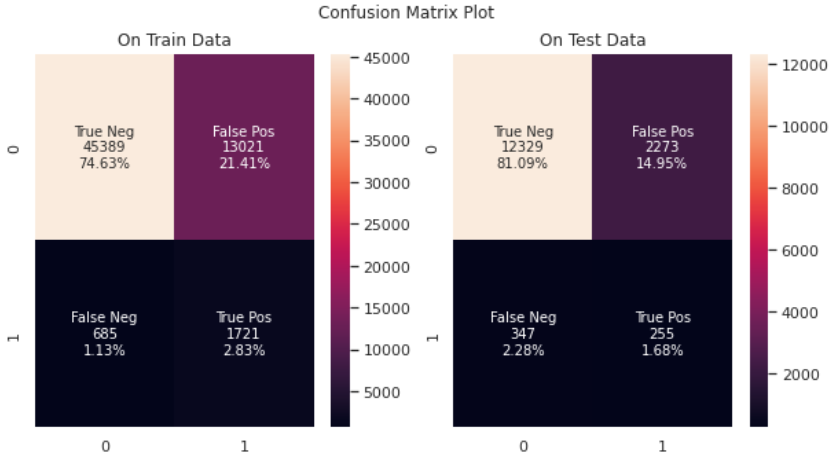
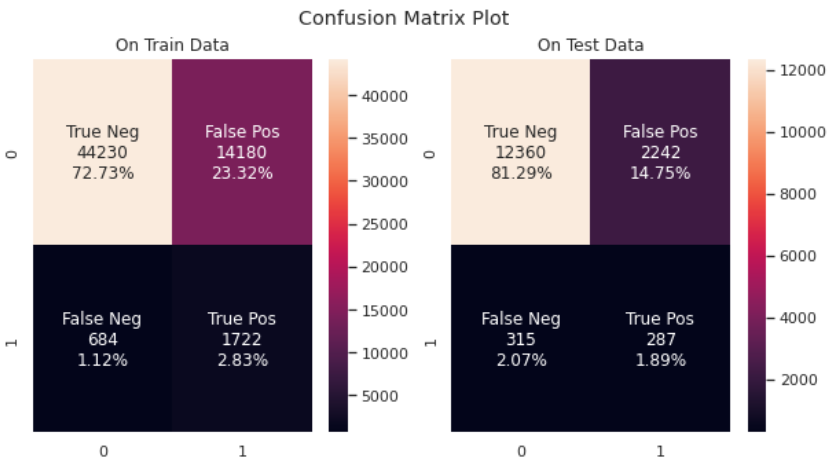
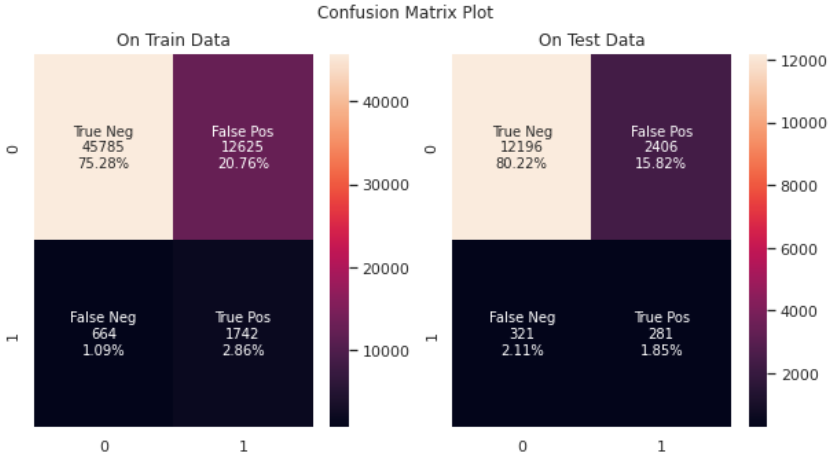


Fig 4: LR Model on Dataset 1 (Top Left), Dataset 2 (Top Right), Dataset 3 (Below Center)

It is kind of evident that the train test AUC score for all the three dataset are comparable, we can also notice that on Dataset 3 the Train AUC is slight higher to others, but the best result on test dataset could be seen on through the model on dataset 1.

Below we can see the confusion matrix in all the 3 dataset with class weights for the Logistic Regression Model:



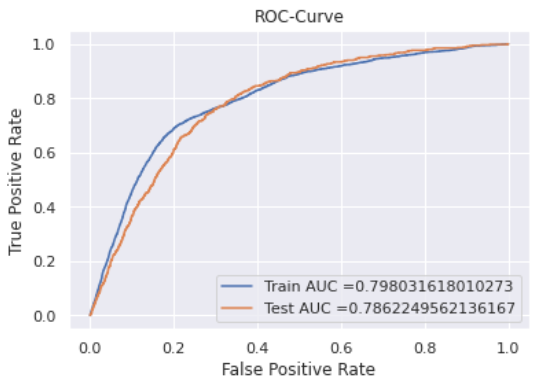
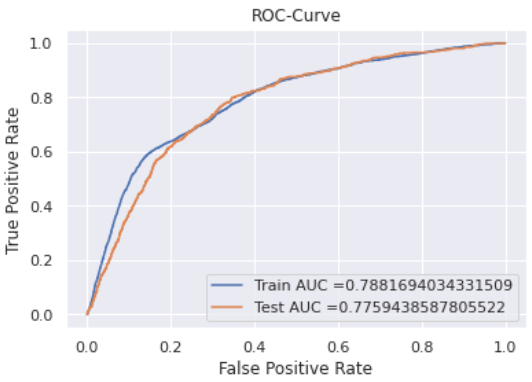
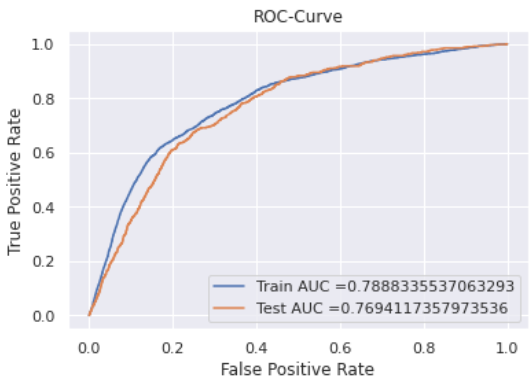
Fig 5: Confusion matrix for LR on all the three dataset (1-top left, 2-top right, 3-bottom centre)

**3.4.2. Grid Search Weighted Support Vector Classifier**

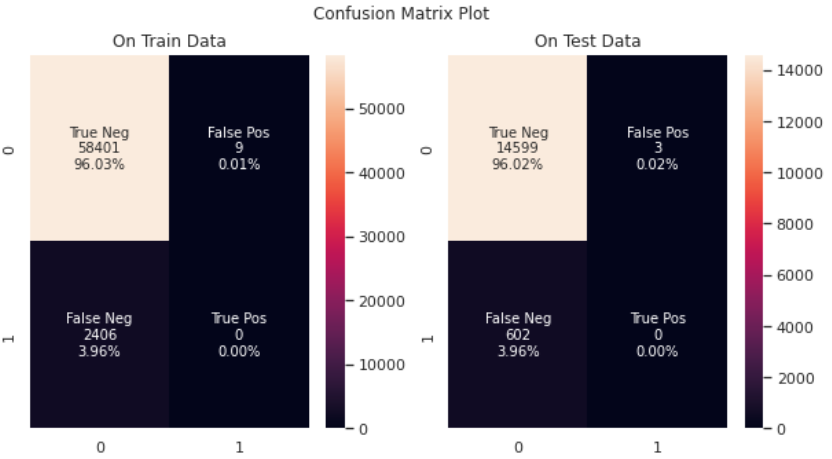
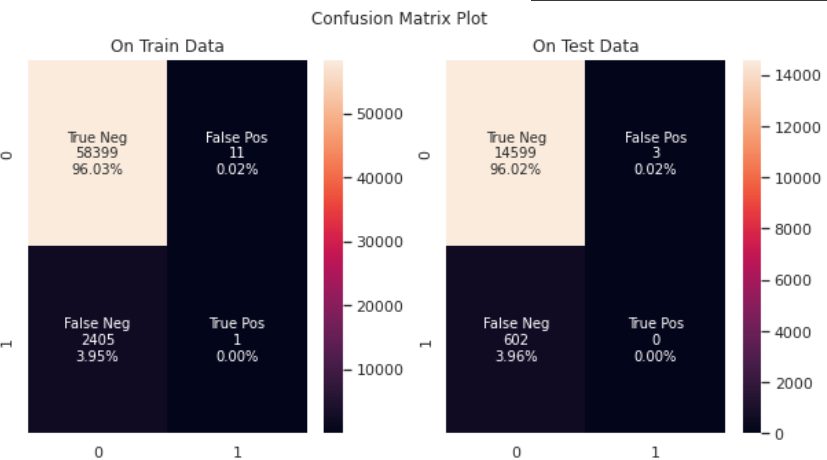
The Support Vector Machine algorithm is effective for balanced classification, although it does not perform well on imbalanced datasets. The SVM algorithm finds a hyperplane decision boundary that best splits the examples into two classes. The split is made soft through the use of a margin that allows some points to be misclassified. By default, this margin favours the majority class on imbalanced datasets, although it can be updated to take the importance of each class into account and dramatically improve the performance of the algorithm on datasets with skewed class distributions.

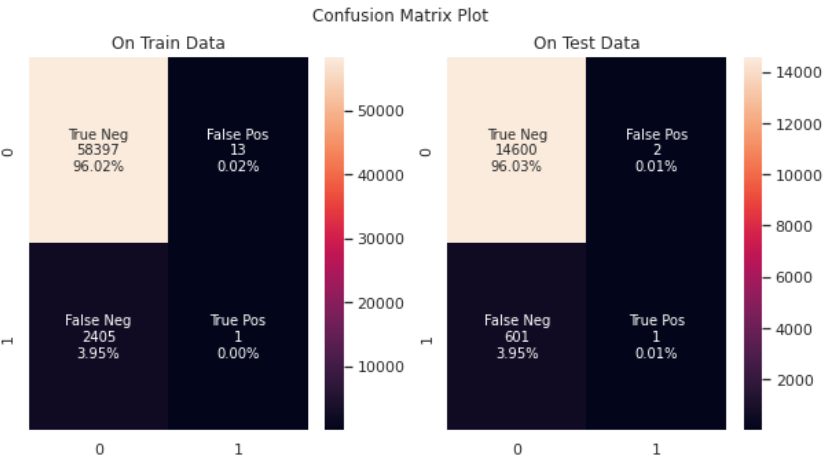
On the similar note I have tried class weights in the SVC model. And try to plot with the ROC curve in all the three datasets. These model will help with understand how the cost sensitive model will work in case of support vector classifiers with the rbf kernel as default.

One of the main issues I faced was that the SVC classifier was taking forever to training. I had been 3 hours and still going on. I tried to find the methods to reduce this and the best result I found was to use LinearSVC and use calibrated model above it as LinearSVC does not give us probability score, but still I was not able to do Grid search to find best class weights.

Fig 6: LinearSVC Model on Dataset 1 (Top Left), Dataset 2 (Top Right), Dataset 3 (Below Center)

This is them most misleading AUC score that I have got, looking at the score we can say the score are comparable but lets see what the confusion matrix says:



Fig 7: Confusion matrix for LinearSVC on all the three dataset (1-top left, 2-top right, 3-bottom centre)

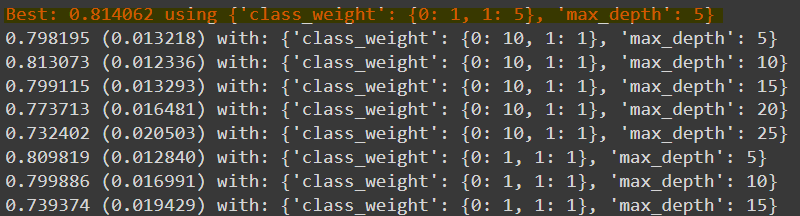
This is the most worst data I could have gotten in terms of result. It seems to me that the model did not able to work at all using the class weights all.

**3.4.3. Grid Search Weighted Decision Tree Classifier**

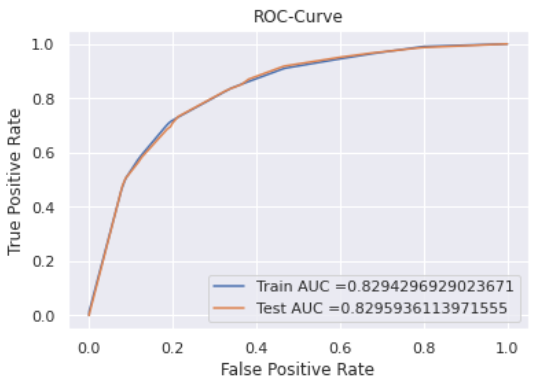
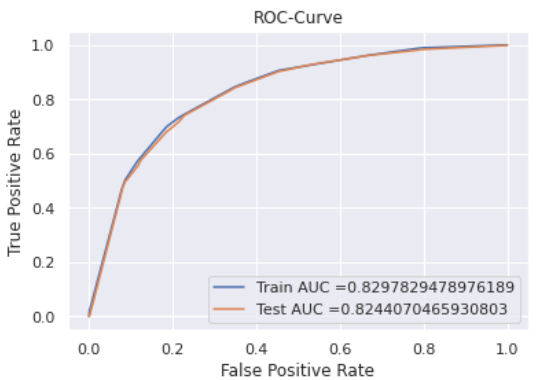
The decision tree algorithm is effective for balanced classification, although it does not perform well on imbalanced datasets. The split points of the tree are chosen to best separate examples into two groups with minimum mixing. When both groups are dominated by examples from one class, the criterion used to select a split point will see good separation, when in fact, the examples from the minority class are being ignored.

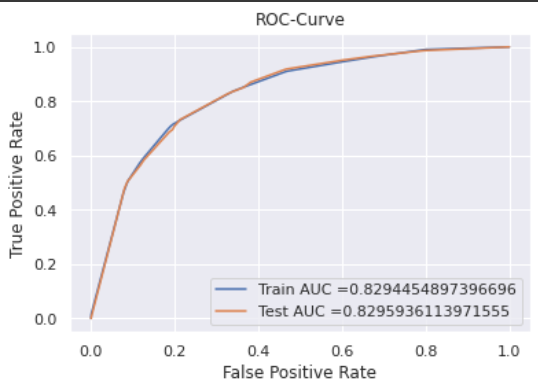
This problem can be overcome by modifying the criterion used to evaluate split points to

take the importance of each class into account, referred to generally as the weighted split-point or weighted decision tree.

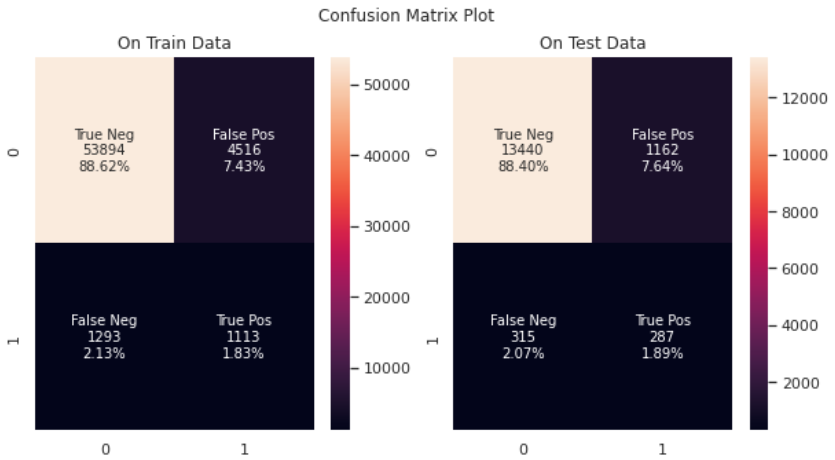
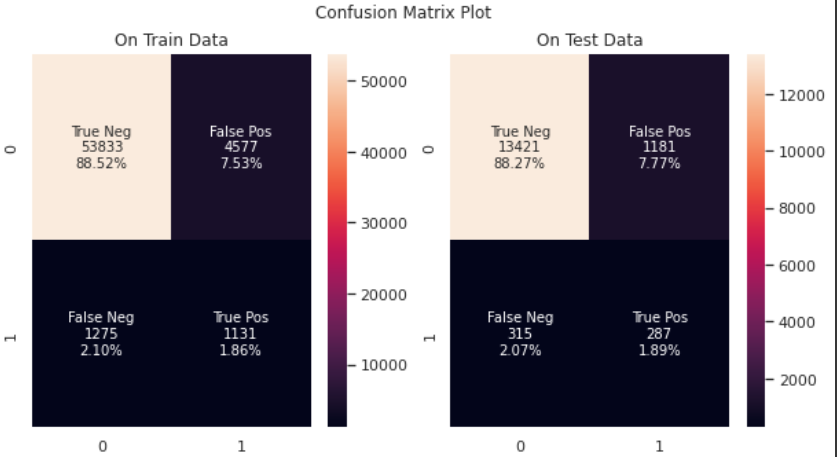


Using the best value for class weight and max\_depth below is the plot for ROC auc curve for all the 3 dataset.



Fig 5: Decision Tree Model on Dataset 1 (Top Left), Dataset 2 (Top Right), Dataset 3 (Below Center)

Looking at the ROC Curve we can say that this model has resulted in the best result for the three dataset compared to the above models.



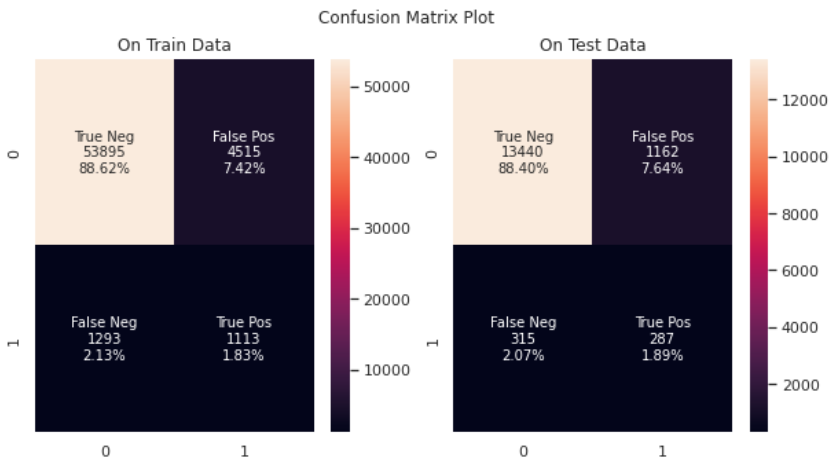


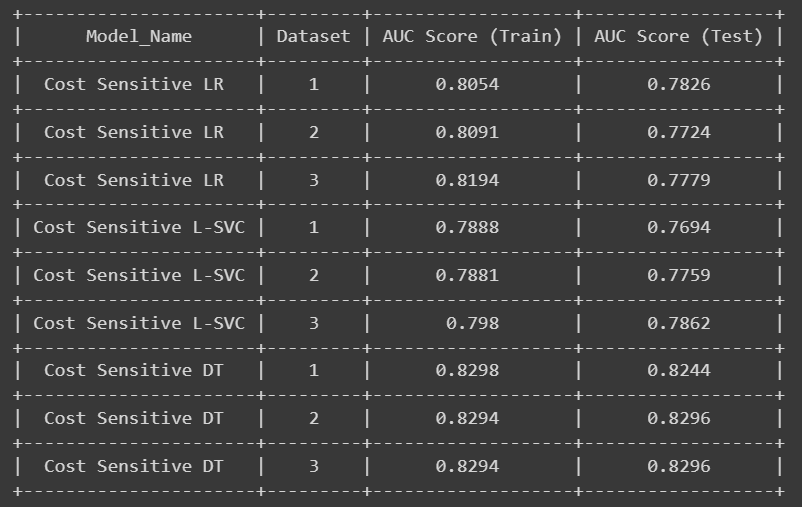
Fig 6: Confusion matrix for Decision Tree on all the three dataset (1-top left, 2-top right, 3-bottom centre)

It clearly evident that the DT model has increased the True negative higher to the previous model of Linear SVC and Logestic Regression. Hence over all reducing the False Positive.

**3.5. Conclusion and Going Forward**

So with the analysis of the above three models the best it is conclusive that the Decision Tree Models gave a best result on all the three dataset I had planned. Also looking at the confusion matrix also the model tried reducing False Positive a lot compared to the other models. Next step going forward would be try ensemble methods and see how the imbalanced dataset we have pans out those kind of models.

I also plan to try with precision recall score as the metric to check once if there is better result with score.



This is the final values for all the AUC Score for all the three cost sensitive models that i have trained and for all the three different types of dataset on both the training and test data. There was improve for LR and Decision Trees from there simple model which was not cost sensitive. But to Linear SVC there was not improvement.

**Chapter 4**

**4.1. Introduction**

In this phase I will use various advanced modelling methods to train and get the result and compare the finding with final kaggle score.

There is a new dataset I added which has response encoding instead of one hot encoding. This will be my 4th dataset. So in total now there are 4 datasets available for me to run my models on.

**4.2. Advanced Modelling**

I tried to get one more different modelling techniques. Below are the details for those:

**4.2.1. Random forest:**

So firstly I tried random forest as the advanced machine learning method and found the below results:

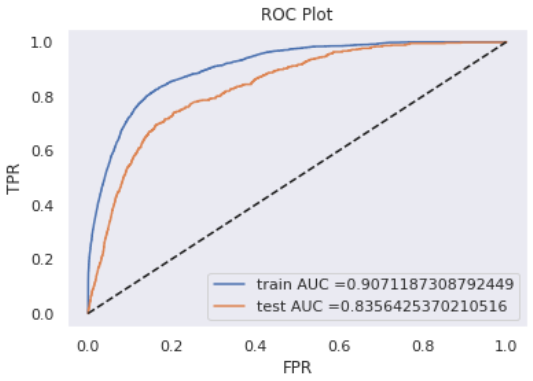
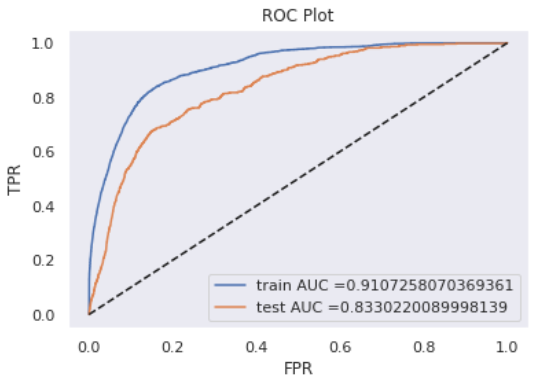
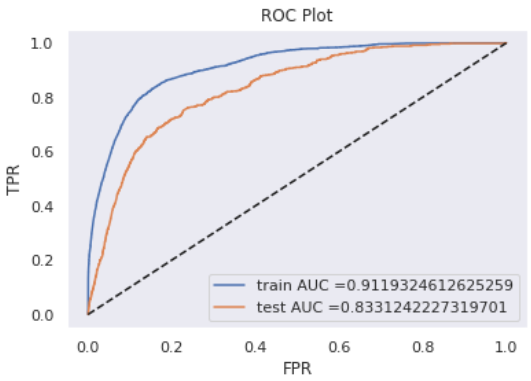
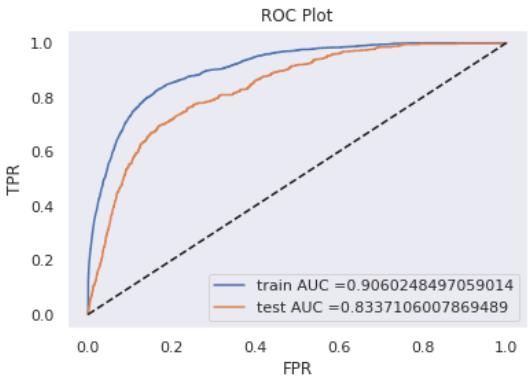


Fig 1: The ROC plot for the 4 datasets on Random Forest models.

This result was a major improvement over the simple models that I had trained over the last phase. Also it seems that the model trained on the 4th dataset (one with response encoding) is better compared in all the four models.

Next I used the model and predicted the final test dataset and got this result from Kaggle:

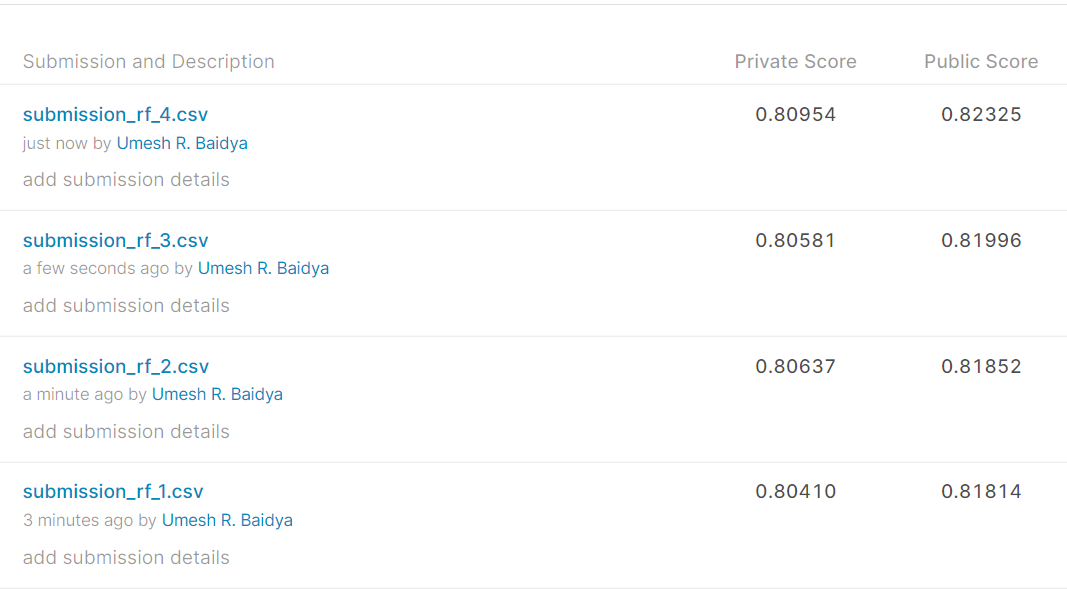


Fig 2: This is the Kaggle score on 4 models on final test data.

As evident it can be seen that the 4th model also behaved well on final test data compared to the rest models.

The random forest model is built on decision trees, and decision trees are sensitive to class imbalance. Each tree is built on a "bag", and each bag is a uniform random sample from the data (with replacement). Therefore each tree will be biased in the same direction and magnitude (on average) by class imbalance. In Python, weighted tree splitting is implemented in the Scikit-learn class RandomForestClassifier, as the class\_weight parameter.

**4.2.2. XGBoost Classifier:**

The next model I used was xgboost and found the below results when I tested on the four datasets:

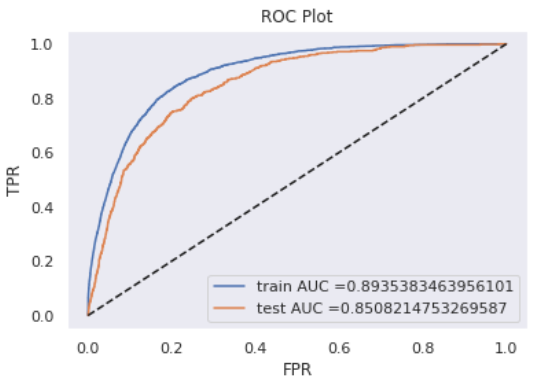
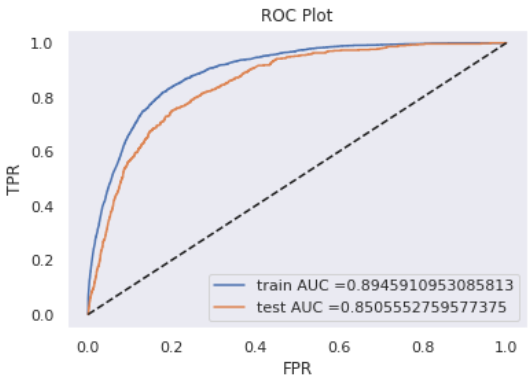
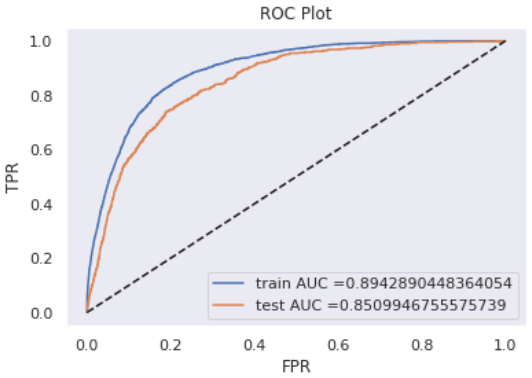
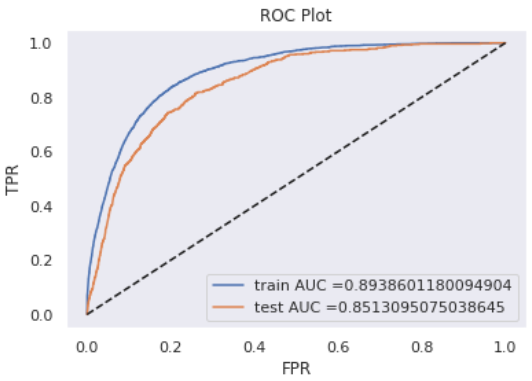


Fig 3: The ROC plot for the 4 datasets on XGBoost models.

There is better improvement from the Random Forest to XGBoost model. Also different than the Random Forest here model is better on the 1st dataset. Next lets find the result on the final test dataset and compare the result in Kaggle score:

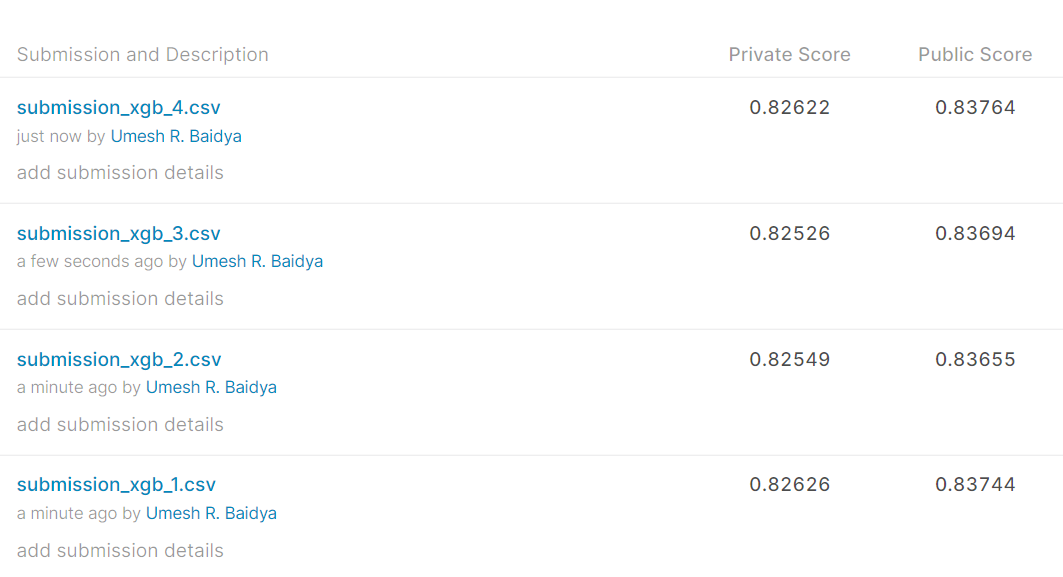


Fig 4: This is the Kaggle score on 4 models on final test data.

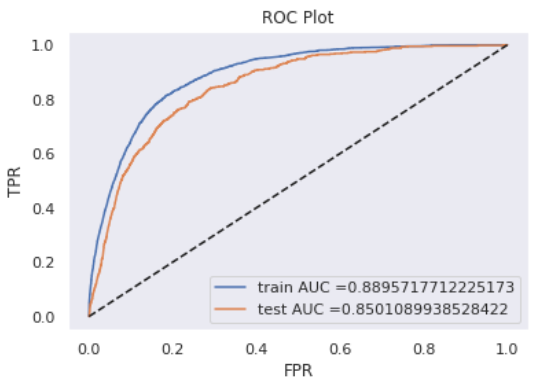
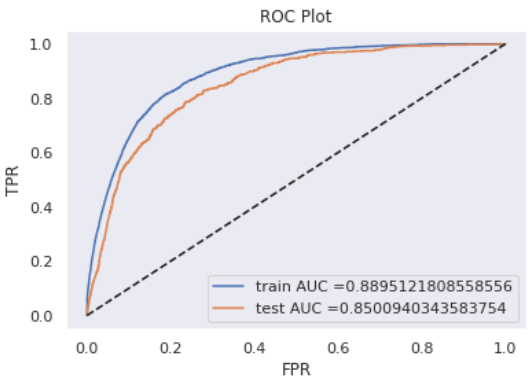
Here we can see that the model trained on the 4th dataset gave a better result compared to the remaining model.

Although the XGBoost algorithm performs well for a wide range of challenging problems, it offers a large number of hyperparameters, many of which require tuning in order to get the most out of the algorithm on a given dataset.

The implementation provides a hyperparameter designed to tune the behavior of the algorithm for imbalanced classification problems; this is the scale\_pos\_weight hyperparameter. By default, the scale\_pos\_weight hyperparameter is set to the value of 1.0 and has the effect of weighing the balance of positive examples, relative to negative examples when boosting decision trees.

**4.2.3. Light-GBM Classifier:**

The next model I used was xgboost and found the below results when I tested on the four datasets:



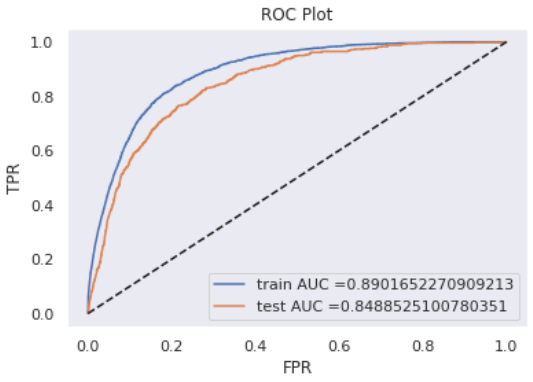


Fig 5: The ROC plot for the 4 datasets on Light GBM models.

And below is the Kaggle score for Light GBM models:

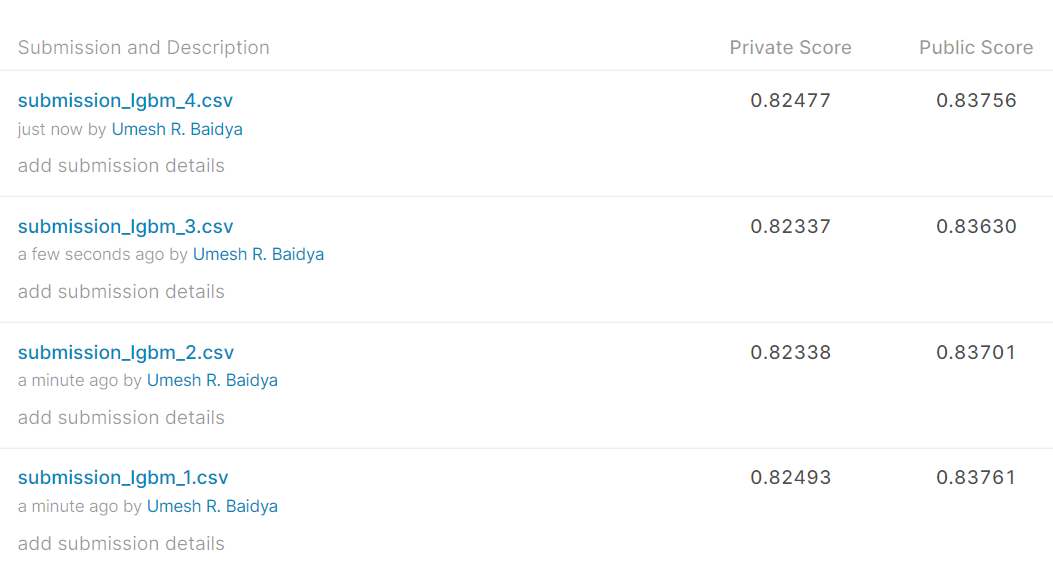
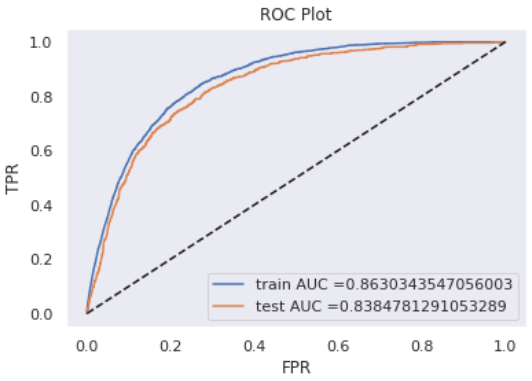
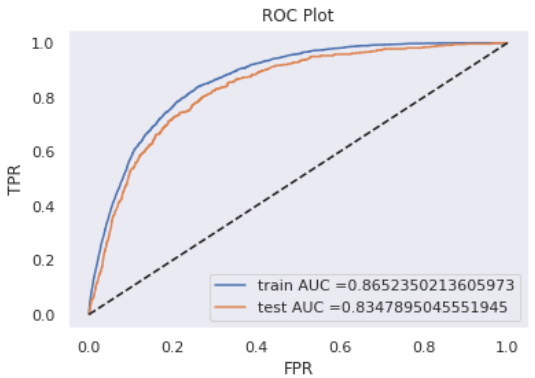


Fig 6: This is the Kaggle score on 4 models on final test data.

**4.2.4. AdaBoost Classifier:**

The last model I tried was the AdaBoost Classifier. And here are the results on these four dataset with this model:



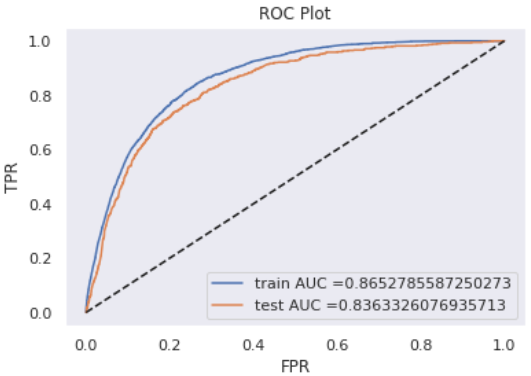
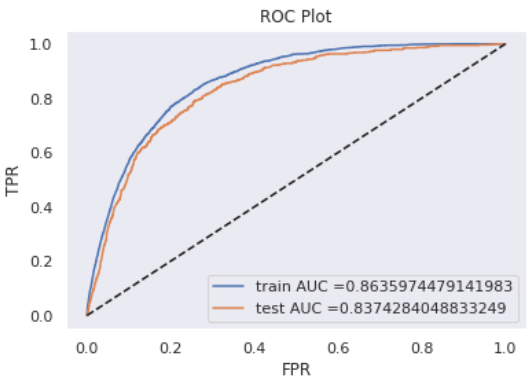


Fig 7: The ROC plot for the 4 datasets on Adaboost models.

This was a slight drop in performance compared to the other three but still better than the simple models tested in the phase 3.

The closest values for train and validation datasets were received for the Adaboost models.

Next lets see the result of final test data and compare the Kaggle scores:

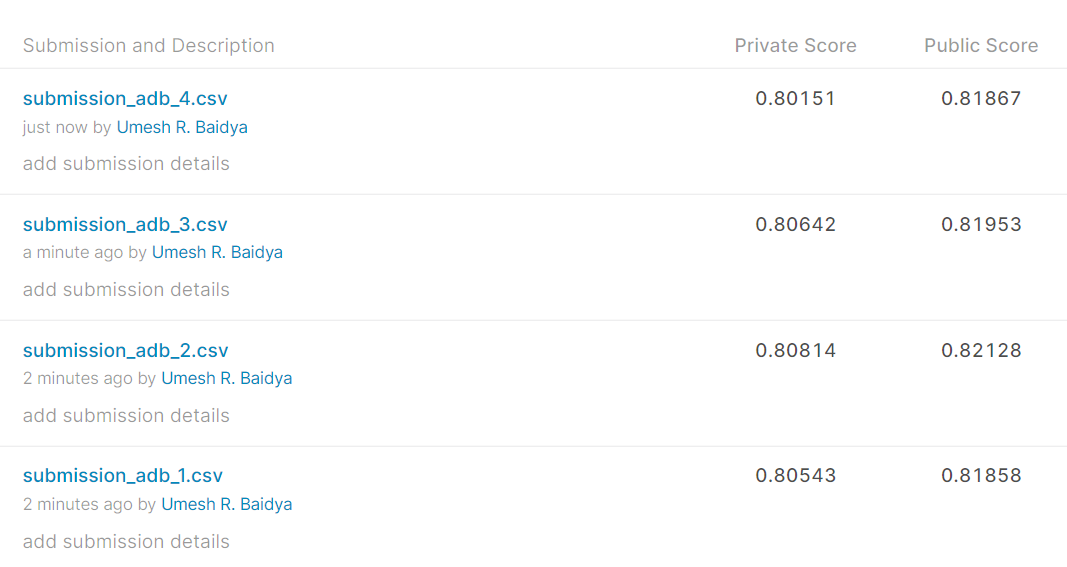
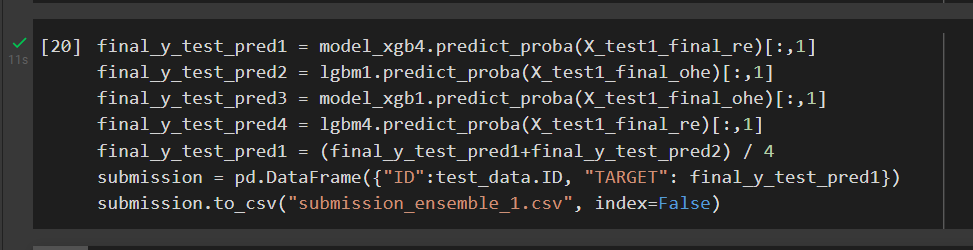


Fig 7: This is the Kaggle score on 4 models on final test data.

The final test Kaggle score is also slightly below compared to the previous models.

**4.3. Ensemble Modelling**

Next idea was to combine top 4 models among all the models and create a simple ensemble model on top of this where we take average of score on the probability score for them:



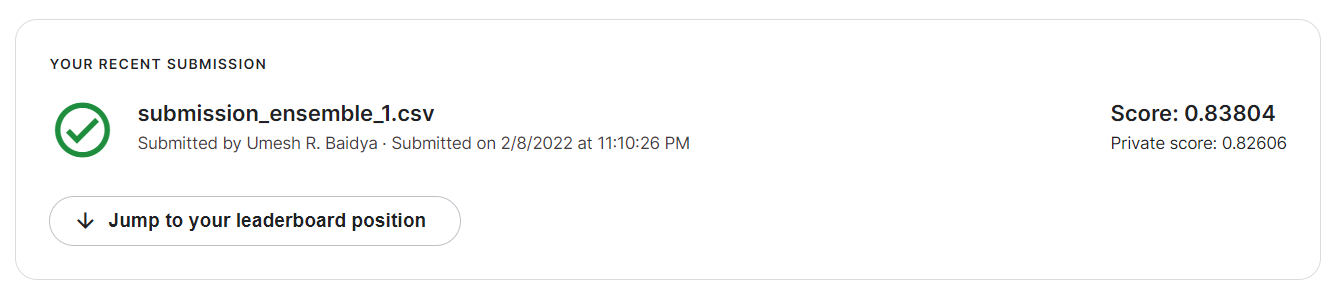
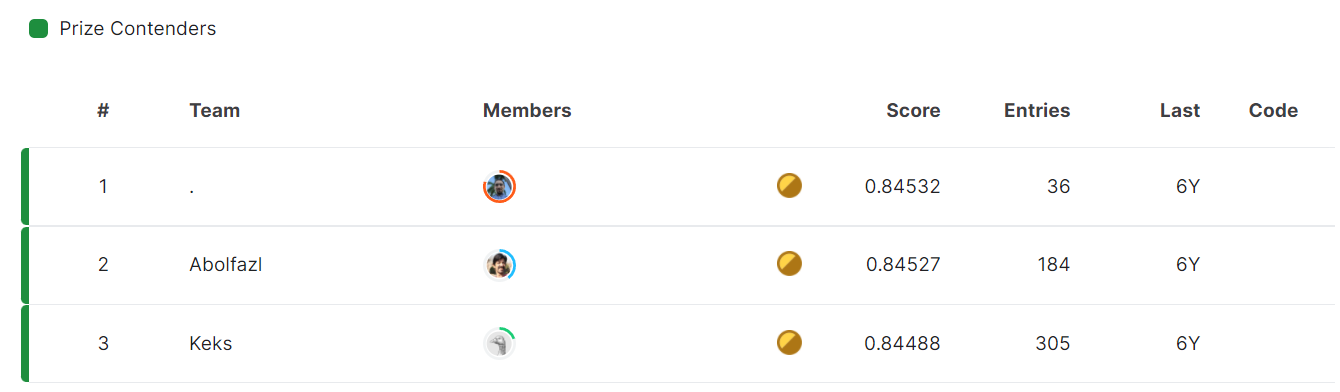


Fig 8: Kaggle result from the ensemble model

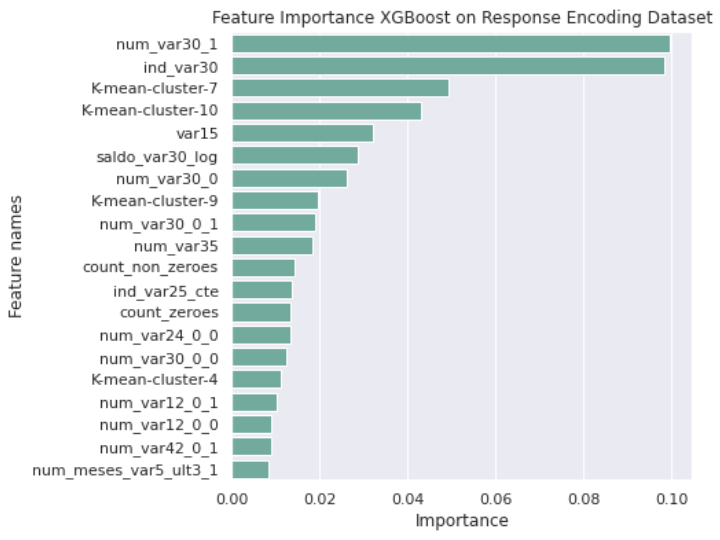
The ensemble model was the very best result that I got on the Kaggle, which is just almost 1% less than the Kaggle top scorers.



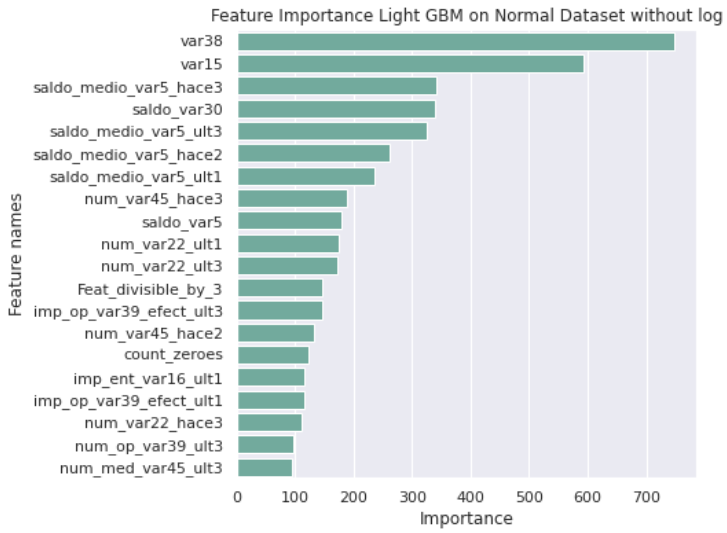
**4.4. Feature Importance**

Next we look at the feature importance for the 4 top models:

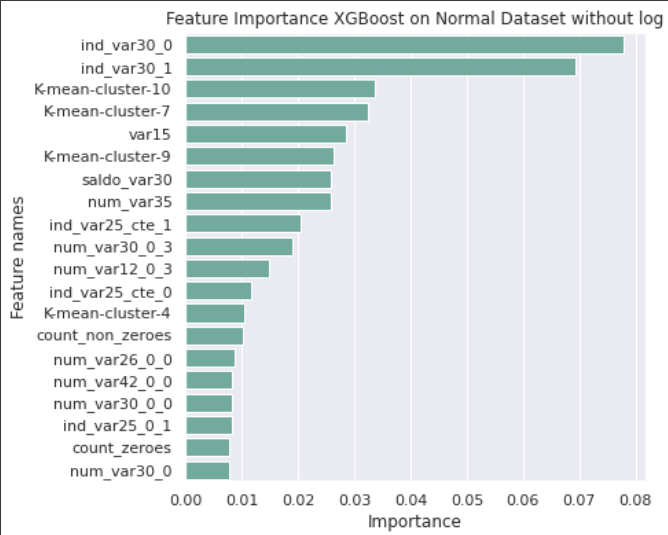
**Model 1: XGBoost with Response Encoded dataset**



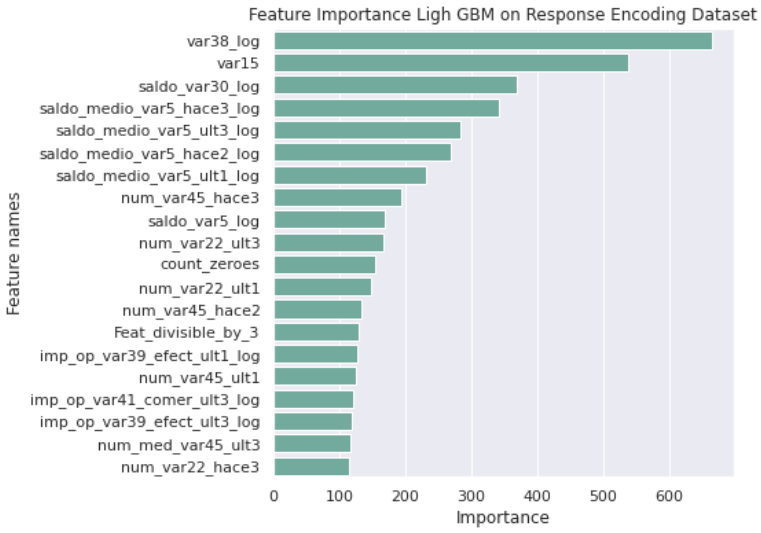
**Model 2: Light GBM with Normal dataset without log**

****

**Model 3: XGBoost with Normal dataset without log**

****

**Model 4: Light GBM with Response Encoded Dataset**

****

We can see that there are several features like var38, var15, count\_zeros, count non zero, var\_30/ind or num, K means features common here. Meaning these features are overall helpful in identifying the satisfied and unsatisfied customers.

**Chapter 5**

**5.1. Introduction**

So the objective with this phase will be to deploy the model that I have built on Heroku Cloud platform. Heroku is a container-based cloud Platform as a Service (PaaS). We as developers can use Heroku to deploy, manage, and scale modern apps.

**5.2. How to use Heroku for deployment**

Heroku gives us the platform to run our python app where we give all it all the dependencies and steps of how to execute and Heroku can very easily run our application.

The advantage that I can see for see from AWS and GCP is that they use virtualization where we have to lease a computer and give different details like the processor, memory and computing power etc., it was comparatively easier for a newbie like me to actually use Heroku.

Steps I followed for the deployment would be as below:

1. Create a folder (for me it was named ‘deploy-heroku’) which will store all the files and images needed for my application to run.
2. Create the “requirements.txt” file. The need for this to tell Heroku which all libraries it needs with their version number for our application to run.

To create this we just need to run this command:

**pip freeze > requirements.txt**

After running the above command it will create this requirements file in my deploy directory.

1. Next we need to create a “Procfile”. This files tell Heroku how to execute my application.

As this is web application we will write following line in Procfile:

“**web: gunicorn app:app**”

1. Next step for me was to install “git” and “Heroku CLI” in my windows.

1. After this I opened git in my deploy-heroku folder and login into Heroku by typing “**Heroku Login**”, this opens browser where we have to login into the Heroku account that we have created.

1. Next step will be to create a new git repository using the below commands:

a) git init . --> this steps initializes a git repository

b) git add . --> this step will add all untracked changes to the staging area

c) git commit -am “<message>” --> this will commit our changes.

1. Next we add this local repo to remote repo. For this we run:

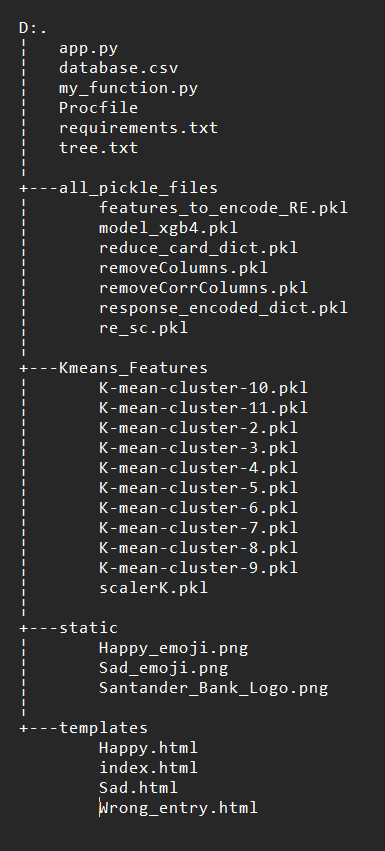
“**heroku git:remote -a umesh-proj-uoh**”

1. Finally we will push this to heroku using below command.

“**git push heroku master**”

**5.3. Tools and Library to solve this problem**

To create the web application I have used flask. Here I have created app.py file which will start my web application. I have also created a simple html also for the output. Below is the structure of the deploy-heroku directory:

Fig 1: Directory Structure

All the HTML files are present in templates folder which is need by the flask application. Also all the images are present in the static folder. Apart from this I have Kmeans\_Features and all\_pickle\_files to store the pickle files for their respective tasks.

**5.4. Demo of this working application**

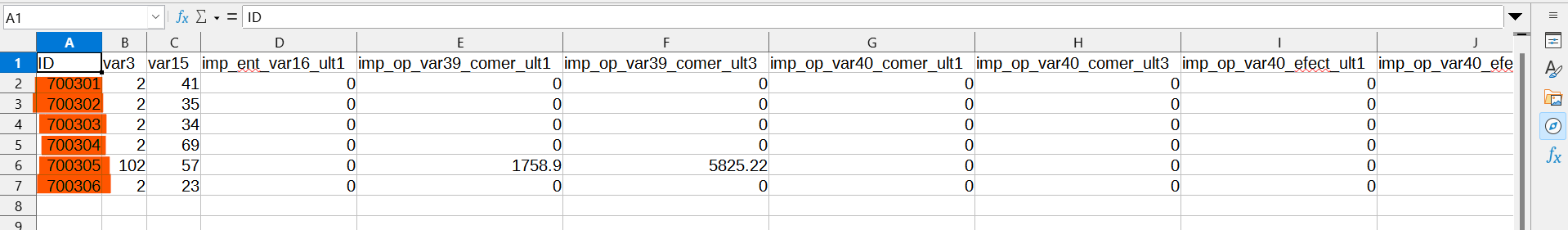
This application can we accessed using this link:

<https://umesh-proj-uoh.herokuapp.com/>

This is the main screen:



I have created a csv file called database.csv which contains different customers details and I have given each of the customer a unique ID as you can see below:

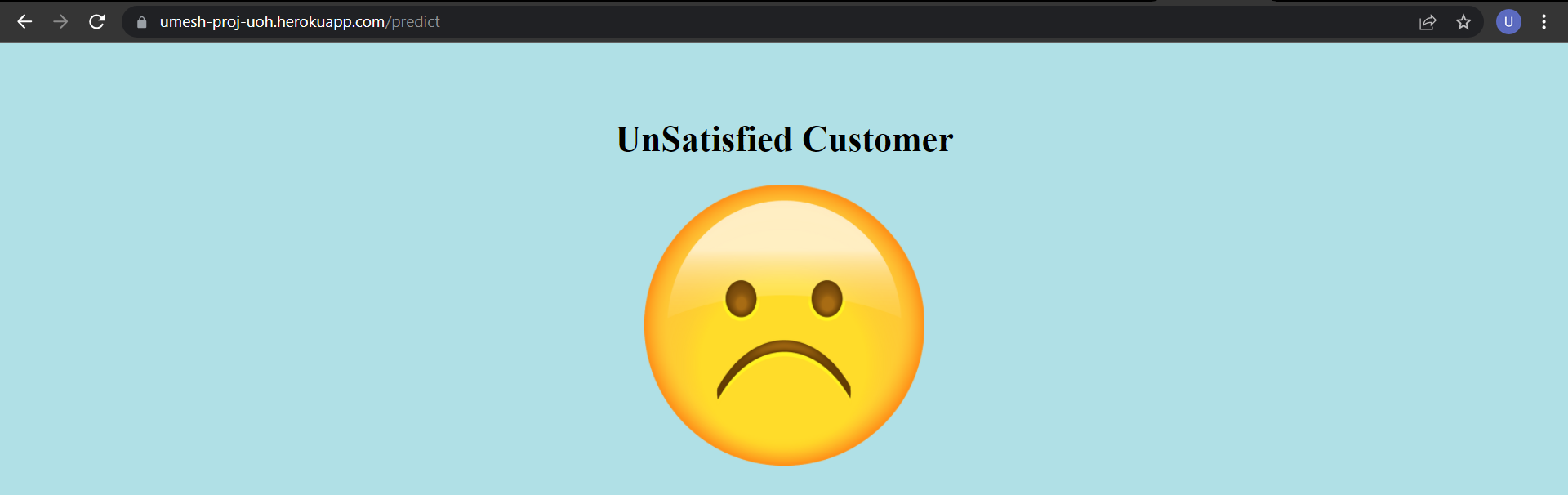


Next we will provide the Employee ID as input to the our web application.



After entering the customer ID and submitting it I will call my application and depending on the output that the customer is satisfied or unsatisfied we get different out like you can see below:



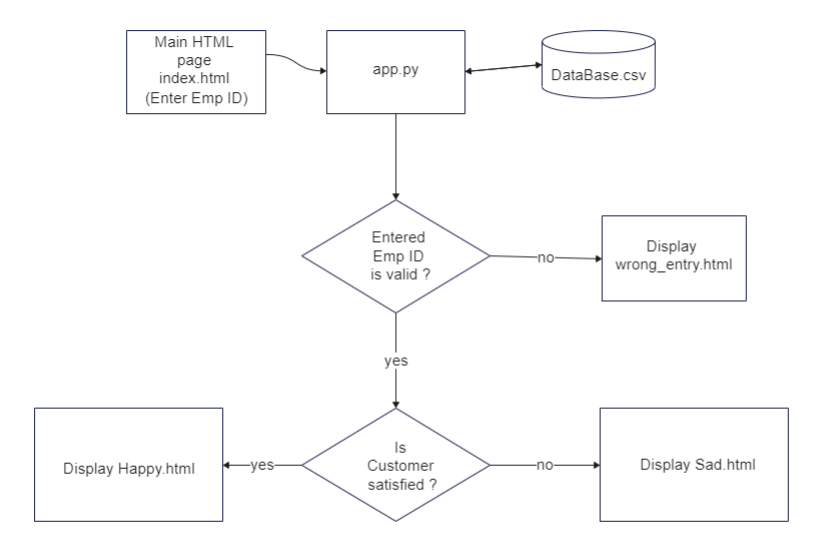


Finally if we enter a customer which is not present in the database.csv file or some other gibberish value in customer id field we get this error:



**Chapter 4**

**Architecture diagram of the system**



**5.5. Limitations of system and how we can improve**

The few limitation of the system that I can think of are as below:

1. Using the csv file as the database. I will be faster and more efficient to use show kind of database like mongo db or MySQL to store all the customer details. Using the csv we have to load the complete csv into the ram which is highly wasteful as we can millions and millions of customers.

1. We are basically under the impression that the data in the database is accurate and without any mistake which is not true as per me and it could so happen that the data we have in the database is having some outlier value or invalid value which can cause some issues.

Remedies to above issue:

1. We can use some proper database like MqSQL to store the customer details and when we enter the emp id we should directly select the record using the sql query which will be faster as well as memory efficient for us.
2. Using the SQL query like this can lead to SQL injection problem hence we should take care of that also.
3. We should check the different values for each columns which are present in the database before using to predict the satisfaction. This can however lead to the problem of latency as we will need to check each and every columns value before making even a single prediction.

**References**:

* <https://arxiv.org/pdf/1505.01658v2.pdf>
* <https://arxiv.org/pdf/1812.01388.pdf>
* <https://www.udemy.com/course/machine-learning-with-imbalanced-data/>
* <https://www.researchgate.net/publication/263913891_Classification_of_imbalanced_data_a_review>
* https://www.getfeedback.com/resources/csat/how-to-measure-customer-satisfaction-kpis/
* https://towardsdatascience.com/metrics-for-imbalanced-classification-41c71549bbb5
* https://machinelearningmastery.com/imbalanced-classification-with-python/
* https://towardsdatascience.com/santander-customer-satisfaction-a-self-case-study-using-python-5776d3f8b060
* https://stats.stackexchange.com/questions/7207/roc-vs-precision-and-recall-curves
* https://www.daniweb.com/programming/computer-science/tutorials/520084/understanding-roc-curves-from-scratch
* https://towardsdatascience.com/random-forest-for-imbalanced-dataset-example-with-avalanches-in-french-alps-77ffa582f68b
* https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/
* https://stackoverflow.com/questions/37627923/how-to-get-feature-importance-in-xgboost
* https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/
* <https://docs.google.com/document/d/1jeJTMffi-9fixbDtxj_63joOEFsmq7na/edit>
* https://www.analyticssteps.com/blogs/what-light-gbm-algorithm-how-use-it
* https://machinelearningmastery.com/xgboost-for-imbalanced-classification/
* https://seaborn.pydata.org/generated/seaborn.barplot.html
* <https://medium.com/@nissanttiwari/santander-customer-satisfaction-82eacb41a2b3>
* https://medium.com/analytics-vidhya/tutorial-exploratory-data-analysis-eda-with-categorical-variables-6a569a3aea55
* https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.RandomizedSearchCV.html
* [https://scikit-learn.org/stable/modules/grid\_search.html#randomized-parameter-search](https://scikit-learn.org/stable/modules/grid_search.html" \l "randomized-parameter-search)
* <https://rstudio-pubs-static.s3.amazonaws.com/190147_69763c0c50fa430c9e774c3527dd253e.html>
* <https://www.geeksforgeeks.org/how-to-set-a-seaborn-chart-figure-size/>
* <https://www.kaggle.com/solegalli/feature-selection-with-feature-engine>
* [https://www.kaggle.com/adarshsng/extensive-advance-feature-selection-tutorial/notebook](https://towardsdatascience.com/santander-customer-satisfaction-a-self-case-study-using-python-5776d3f8b060)
* <https://stackoverflow.com/questions/40273067/replace-specific-values-in-a-dataframe-column-using-pandas>
* <https://www.analyticsvidhya.com/blog/2020/07/knnimputer-a-robust-way-to-impute-missing-values-using-scikit-learn/>
* <https://www.kaggle.com/c/santander-customer-satisfaction/discussion/20811>
* <https://www.kaggle.com/c/santander-customer-satisfaction/discussion?sort=votes>
* <https://www.analyticsvidhya.com/blog/2020/07/univariate-analysis-visualization-with-illustrations-in-python/>
* <https://medium.com/analytics-vidhya/tutorial-exploratory-data-analysis-eda-with-categorical-variables-6a569a3aea55>
* <https://analyticsindiamag.com/beginners-guide-to-tabulate-python-tool-for-creating-nicely-formatted-tables/>
* <https://towardsdatascience.com/dealing-with-features-that-have-high-cardinality-1c9212d7ff1b>
* <https://stackoverflow.com/questions/37292872/how-can-i-one-hot-encode-in-python>
* https://www.edrawmax.com/online/en/
* https://www.appliedroots.com/lecture/6/diploma-in-ml-ai-with-uoh/4877/hands-on-session-deploy-an-ml-model-using-flask-apis-on-aws/38/course-4-advanced-machine-learning-7-credits
* https://www.appliedroots.com/lecture/6/diploma-in-ml-ai-with-uoh/4880/ml-model-productionization-using-heroku/38/course-4-advanced-machine-learning-7-credits
* https://devcenter.heroku.com/categories/reference
* https://www.w3schools.com/html/