Machine Learning Engineer Nanodegree

Capstone Project

Soham Mukherjee
June 22, 2018.

Santander Customer Satisfaction

Project Overview

This is partial project to classify a customer of Santander Bank to be satisfied or not.

Throughout the project I have tried to use different techniques of feature engineering and feature selection with the use of some basic classifying techniques to compare the results of different classifiers.

I did come across some links that talked about how a marketing agency conducted a survey and gave results based on their survey. The references of these are given in the survey section. However, none of these links gave a very detailed description of the important features that might describe a customer's satisfaction. The papers also did not give an automated approach to determining customer satisfaction.

An automated algorithm is necessary for a bank to determine its customer's satisfaction with the exponentially increasing data size. The banks cannot rely on statistical surveys alone as in today's day and age none of us would prefer to spend even a few minutes answering some questions. This is why it is important to come up with an automated algorithm that learn from the customer's behavior along with other features to come up with a targeted profile of the customer to determine whether the customer is happy or not. This gives the bank a chance to build their business around each and every customer to guarantee maximum satisfaction. This is where machine learning comes in. Thus , I have tried to come up with a standard machine learning model on m y own as I was unable to find any such literature on this project.

The dataset provided by the bank does not contain a data definition which makes it harder to research on the features and there usage. So without trying to find out what these features are , I tried to use statistical methods to determine important features and use them into my classifiers.

Even though this is my submission for the project but this project can be extended to include a lot of other techniques in order to provide a much greater accuracy score in terms of ROC AUC.

Background

Santander Customer Satisfaction is a competition posted by Santander Bank on Kaggle asking it’s user base for a possible solution to the problem declaring about $60,000 as prize money. As for any business, customer satisfaction is the key to its success. Similarly, Santander is asking Kagglers to find a possible machine learning algorithm to identify unhappy customers early in their relationship.

Problem Statement

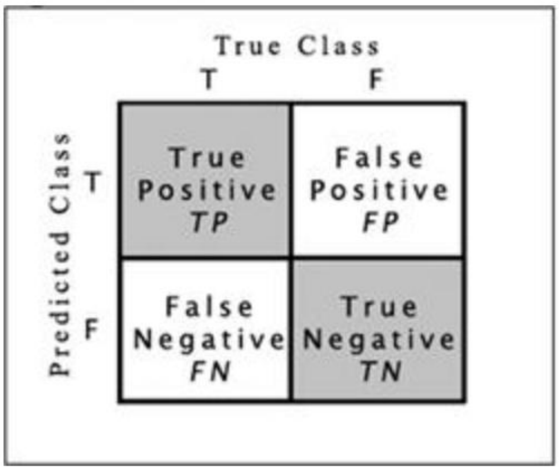
Santander Bank asks Kagglers to help them identify dissatisfied customers early in their relationship. For any business, customer satisfaction is the key for its success. Santander wants to identify unhappy customer so that it can take appropriate measures in order to improve their satisfaction before it’s too late. Identifying unhappy customers don’t tend to stick around and the challenge in this situation is that, unhappy customers don’t tend to voice their dissatisfaction before leaving. Thus, the only way to find out is through the historical data containing hundreds of features which might help in predicting if a customer is satisfied or not.

The solution to this problem is to apply classification algorithms to the training data set to train the model then use the model to test it against the given test.csv. The challenge would be to find what the features describe and engineer the features and find out a suitable classification algorithm by evaluating some algorithms namely – Logistic regression ,Decision trees, Ensemble methods and also neural nets and to find out whether a single algorithm works well or a combination would work best.

Evaluation Metrics

According to the project description provided by the bank at Kaggle, the submission are to be evaluated on area under the ROC curve.

The ROC or Receiver Operating Characteristic curve plots the true positive rate against the false positive rate. This is used to diagnose the ability of a binary classifier system. It can be used to find possibly optimal models and to discard suboptimal ones from one another. The true positive rate is known as the recall and the false positive is known as the fall-out. Thus, the ROC curve can be thought of as a plot of Power as a function of Type 1 error of the decision rule.



		True Class	
		T	F
Predicted Class	T	True Positive *TP*	False Positive *FP*
	F	False Negative *FN*	True Negative *TN*

$$ACC = accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$SP = specificity = \frac{TN}{FP + TN}$$

$$TPR = sensitivity = \frac{TP}{TP + FN}$$

$$FPR = (1 - specificity) = \frac{FP}{FP + TN}$$

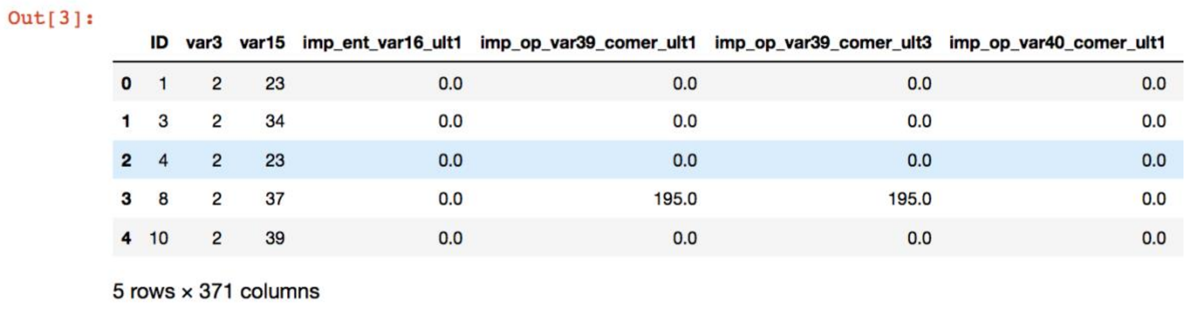
Even though there are other ways of measuring classification accuracy ROC AUC gives a very comprehensive accuracy score by plotting sensitivity vs (1-specificity) which are derived from the confusion matrix.

The popularity of ROC AUC over standard accuracy measure comes from the fact that in any unbalanced dataset it is easy to achieve 99% accuracy if 99% of the data belongs to one class. Whereas ROC AUC depends on both True Positives as well as False Positives to determine a score thus making the metrics fairly independent of the state of unbalance in the data.

Project Analysis

Exploratory Data Analysis and Visualization

The data provided by Santander is an anonymized dataset which means, there is no way to tell which column corresponds to which feature and what it describes. This makes it significantly difficult to determine the meaning of each features and correspondingly it becomes difficult to identify its usefulness to determine its effect on the target variable. Below is a sample of the dataset.

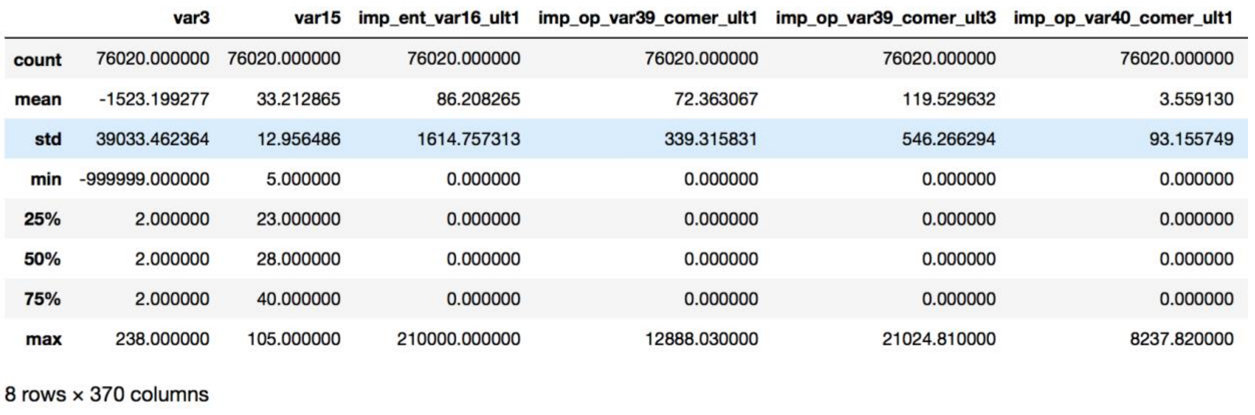


Out[3]:

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_var40_comer_ult1
0	1	2	23	0.0	0.0	0.0	0.0
1	3	2	34	0.0	0.0	0.0	0.0
2	4	2	23	0.0	0.0	0.0	0.0
3	8	2	37	0.0	195.0	195.0	0.0
4	10	2	39	0.0	0.0	0.0	0.0

5 rows × 371 columns

However, following are some basic exploration that I could do. The data set contains 371 columns / features which makes it significantly illogical to explore each and every features and determine its importance. Some explorations were done which are listed below.



	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_var40_comer_ult1
count	76020.000000	76020.000000	76020.000000	76020.000000	76020.000000	76020.000000
mean	-1523.199277	33.212865	86.208265	72.363067	119.529632	3.559130
std	39033.462364	12.956486	1614.757313	339.315831	546.266294	93.155749
min	-999999.000000	5.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	23.000000	0.000000	0.000000	0.000000	0.000000
50%	2.000000	28.000000	0.000000	0.000000	0.000000	0.000000
75%	2.000000	40.000000	0.000000	0.000000	0.000000	0.000000
max	238.000000	105.000000	210000.000000	12888.030000	21024.810000	8237.820000

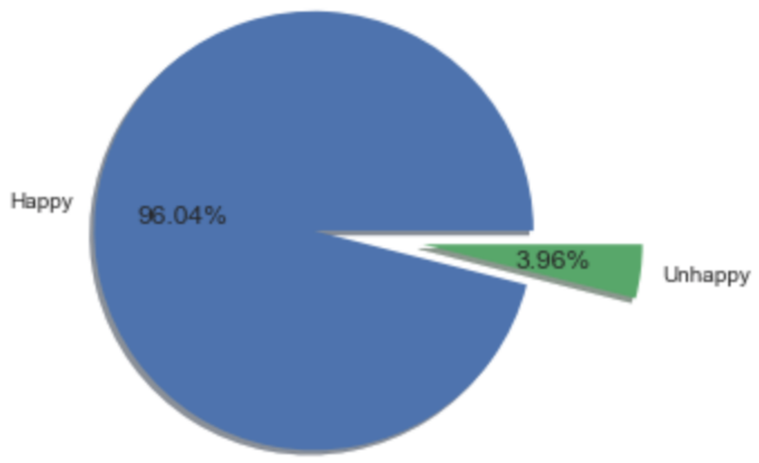
8 rows × 370 columns

Above is a partial screenshot of the data description a determined by the describe() method of the python pandas library. This description suggests:

- There are 76020 data points
- 370 columns including the Target variable
- Also, individual feature statistics.
- The code would show us that many features has a standard deviation of 0 and many features are duplicated or identical.

Some analysis was possible for example, var3 is a categorical variable , var15 is an integer variable which might suggest the age of the customers.

Another preliminary analysis that was possible was to determine the distribution of the Target variable.



	TARGET	Percentage
0	73012	96.043147
1	3008	3.956853

The Target variable consists of about 4% of unhappy customers and 96% happy customer. The dataset is highly unbalanced. SMOTE (Synthetic Minority Over-sampling Technique) was used to balance the dataset and the results where compared using the same final classifier.

Feature names suggests that var3, var15, num_var38 are features out of which no other feature is derived from whereas other features have nam es which suggests they are being derived from the same category or like wise for example

```
'imp_op_var40_comer_ult1','imp_op_var40_comer_ult3', 'imp_op_var40_e
fect_ult1','imp_op_var40_efect_ult3', 'imp_op_var40_ult1'
```

The names of the above features suggests that these where derived from a the same imp_op_var40 variable. There are other features like this as well.

As there are 370 features, no further exploration was conducted. However, feature selection techniques where used to determine which feature had the most impact on the Target variable and a list of features where drawn.

Feature Selection

Even though there are a number of ways features selection and engineering can be done, I chose to use Chi – squared and ANOVA F-value tests to determine the features which are most important or rather the features on which the Target variable is dependent on. Columns with standard deviation = 0 and identical or duplicate columns where deleted from the dataset.

Chi-squared - The idea is to compute chi-squared stats between each non-negative feature. Chi-square test measures dependence between stochastic variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification. This is done for categorical variables.
ANOVA F-Value - F-Test checks for and only captures linear relationships between features and labels. A highly correlated feature is given higher score and less correlated features are given lower score.

After running these tests and selecting the features we were left with 60 features out of 370.

The variable var38 was revisited after Feature Selection. It seemed that this feature is heavily right skewed. I decided to use the logarithmic value of this variable instead.

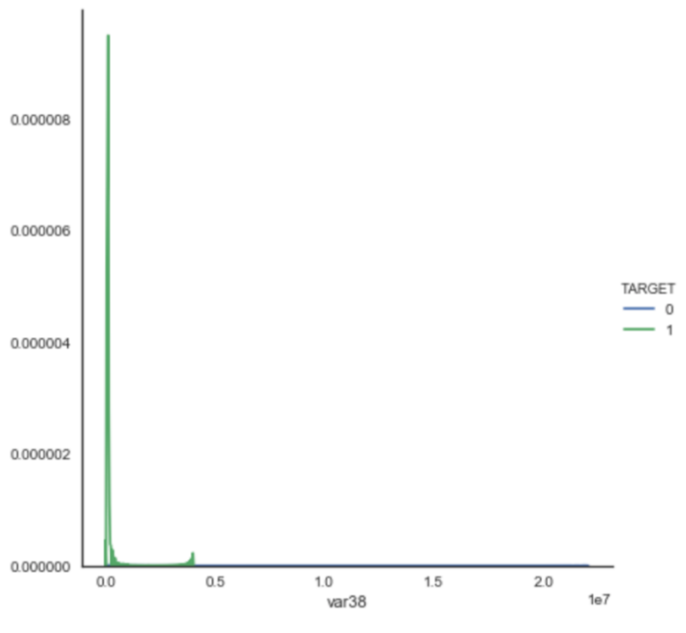


Fig. - right skewness of var38

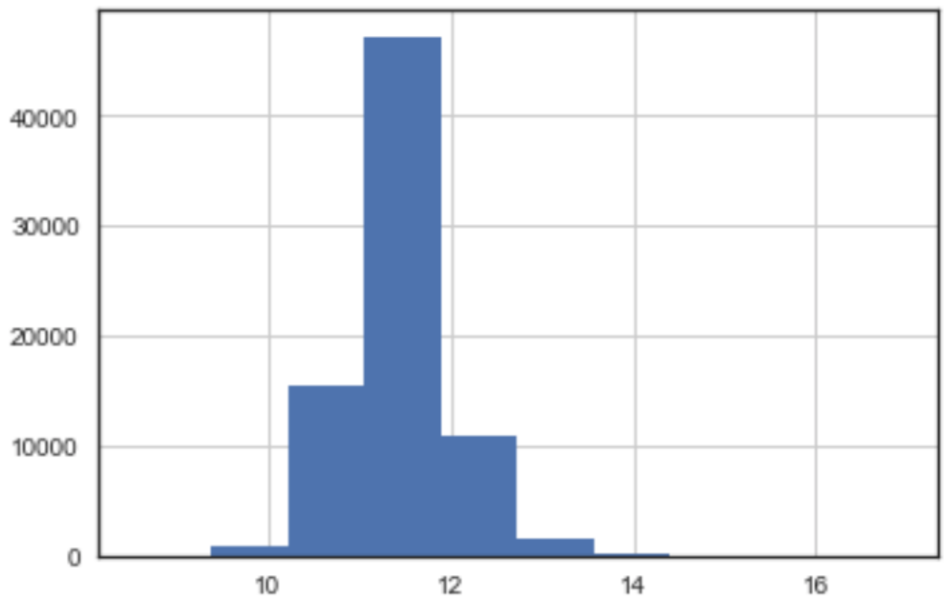
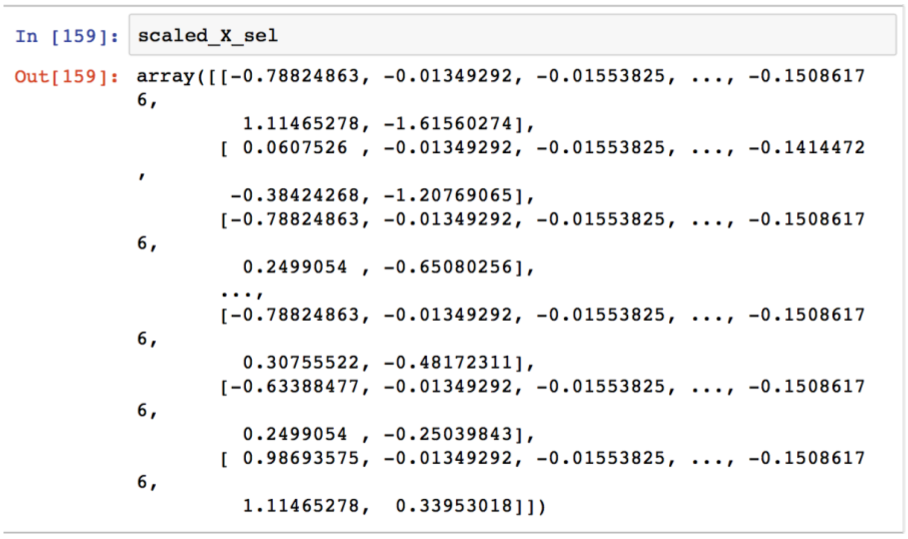


Fig. – Skewness avoided with log_var38

Scaling Data

I used Standard scaler from sklearn preprocessing to standardize the data set. StandardScaler standardizes the dataset by removing the mean and scaling to unit variance.

Below is the sample of the scaled dataset of the selected features.



```
In [159]: scaled_X_sel

Out[159]: array([[-0.78824863, -0.01349292, -0.01553825, ..., -0.1508617
          6,
                   1.11465278, -1.61560274],
                 [ 0.0607526 , -0.01349292, -0.01553825, ..., -0.1414472
          ,
                  -0.38424268, -1.20769065],
                 [-0.78824863, -0.01349292, -0.01553825, ..., -0.1508617
          6,
                   0.2499054 , -0.65080256],
                 ...,
                 [-0.78824863, -0.01349292, -0.01553825, ..., -0.1508617
          6,
                   0.30755522, -0.48172311],
                 [-0.63388477, -0.01349292, -0.01553825, ..., -0.1508617
          6,
                   0.2499054 , -0.25039843],
                 [ 0.98693575, -0.01349292, -0.01553825, ..., -0.1508617
          6,
                   1.11465278,  0.33953018]])
```

Data Splitting

Even though the dataset provided by Santander has a test dataset, but it does not have any TARGET variable to test our classification results. I decided to split the training dataset into train and valid datasets with 70% and 30% ratio. Once the model is created on the 70% of the training set, the model can be tested against the validation set containing the rest of the 30%.These splitting is done from scaled dataset of the selected features.

Benchmark Model

I used logistic regression as the benchmark model for my project. The details of the logistic regression model is given the next section of **Model Selection and Tuning**.

Model Selection and Tuning

Models Used

For this particular problem four types of classifiers where used to classify the dataset

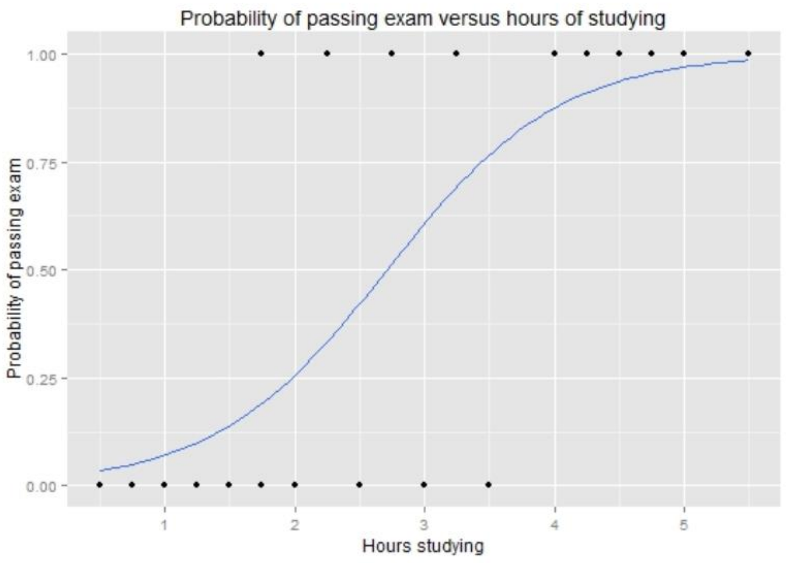
- MLPClassifier (Multilayer Perceptron Classifier)
- LogisticRegression
- XGBoost
- RandomForest

Logistic Regression (Benchmark Model)

Logistic Regression was primarily developed to describe data and understand the relationship between dependent and independent variables. It uses a logistic function as below –

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

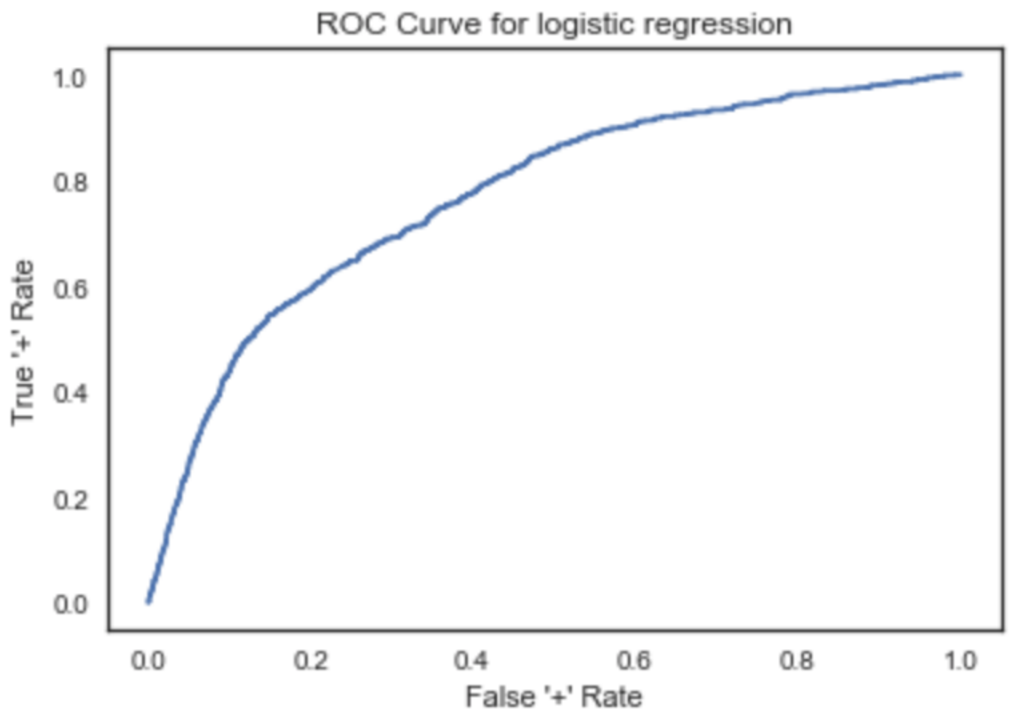
This function produces a sigmoid curve as below –



[diagram taken from Wikipedia]

This sigmoid function can be used as a classification technique as well. By setting a threshold value of probability (say 0.50) we can say that any point with probability above 0.5 belongs to one class whereas the rest belongs to the other class.

I used the logistic regression package from sklearn. Also, I used GridSearchCV to find the most optimal values for C – inverse of the regularization strength with solver as newton-cg. This gave C = 1.0 as the most optimal value with 79.2% AUC score on the training set and 76.9% AUC score on the validation set.



Following where the hyperparameters that I used for the model building

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_i
ntercept=True,
          intercept_scaling=1, max_iter=10000, multi_class='ov
r', n_jobs=1,
          penalty='l2', random_state=555, solver='newton-cg',
tol=10,
          verbose=0, warm_start=False)
```

Even though logistic regression gave a score of 76.9% on the validation , MLP Classifier had already gave a better score of 80.9%. Thus, I decided to use the Logistic regression results as the base model for my analysis.

MLPClassifier

The MLP classifier or the multi-layer perceptron classifier is a neural network which is readily available with sklearn. These are a type of feed forward artificial neural network consisting of three layers of perceptron – *input, output* and *hidden layers.* Each layer consists of perceptron or nodes. Weights are associated with each node. The weights are adjusted using the *backpropagation* technique and *gradient descent* to lower the final output error.

The final error is given by –

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n).$$

here, e is the error at nth data point.

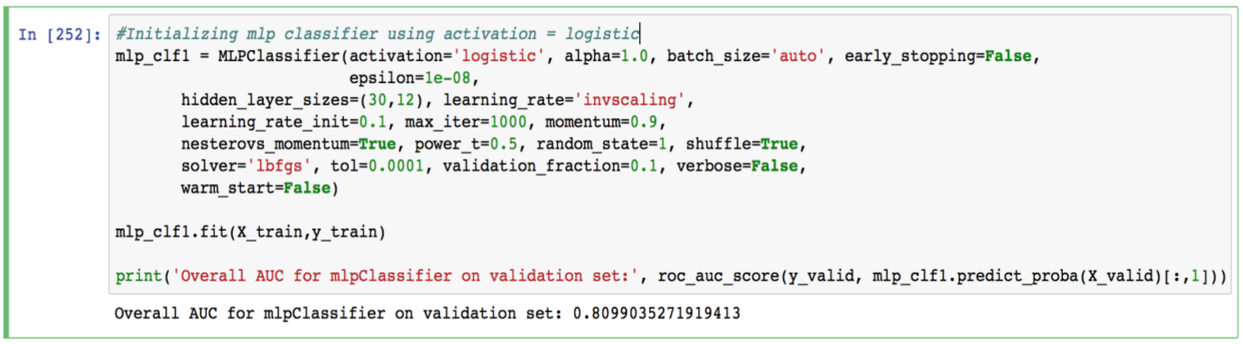
The weights are given by –

$$\Delta w_{ji}(n) = -\eta\frac{\partial\mathcal{E}(n)}{\partial v_j(n)}y_i(n)$$

here, η is the learning rate and y_i is the output of the previous perceptron.

Initially, a basic MLPClassifier was created with hidden_layer_sizes as 30,30 and 30. This model gave an AUC score of 80.7% on the validation set.

The hyperparameters of the model was then tuned using GridSearchCv method to find alpha or the L2 regularization parameter. The result gave use alpha =1000 as the most optimal one and an AUC score of 96% on the training set itself. However, on running this model on the validation set it gave a very low AUC. Thus, I manually changed the alpha to 1 with the following hyperparameters which gave the best AUC score on the validation set.



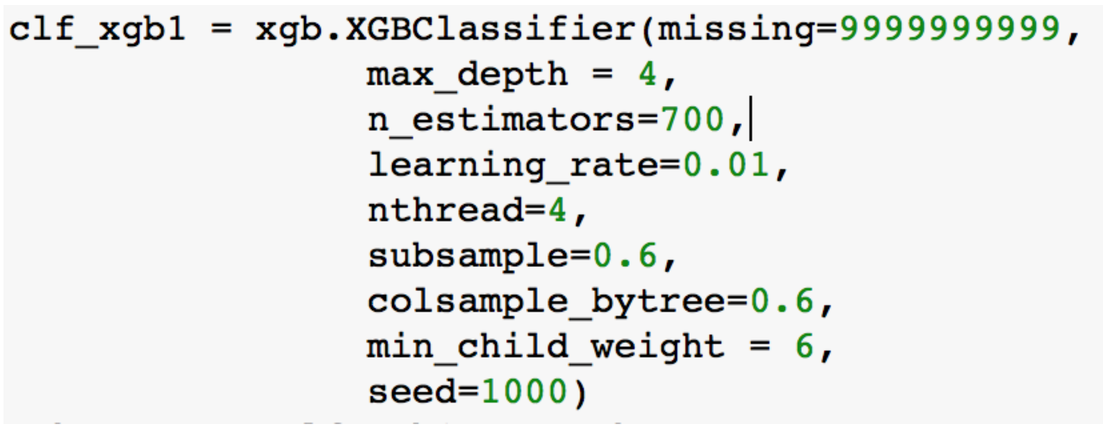
```
In [252]: #Initializing mlp classifier using activation = logistic
          mlp_clf1 = MLPClassifier(activation='logistic', alpha=1.0, batch_size='auto', early_stopping=False,
                                   epsilon=1e-08,
                 hidden_layer_sizes=(30,12), learning_rate='invscaling',
                 learning_rate_init=0.1, max_iter=1000, momentum=0.9,
                 nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
                 solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
                 warm_start=False)

          mlp_clf1.fit(X_train,y_train)

          print('Overall AUC for mlpClassifier on validation set:', roc_auc_score(y_valid, mlp_clf1.predict_proba(X_valid)[:,1]))

          Overall AUC for mlpClassifier on validation set: 0.8099035271919413
```

XGBoost

XGBoost or eXtreme Gradient Boosting algorithm is a newer implementation of the already existing gradient boosting decision tree algorithm. Unlike any other gradient boosting algorithm, XGBoost was made to be a highly scalable, fast and high performance gradient boosting algorithm.

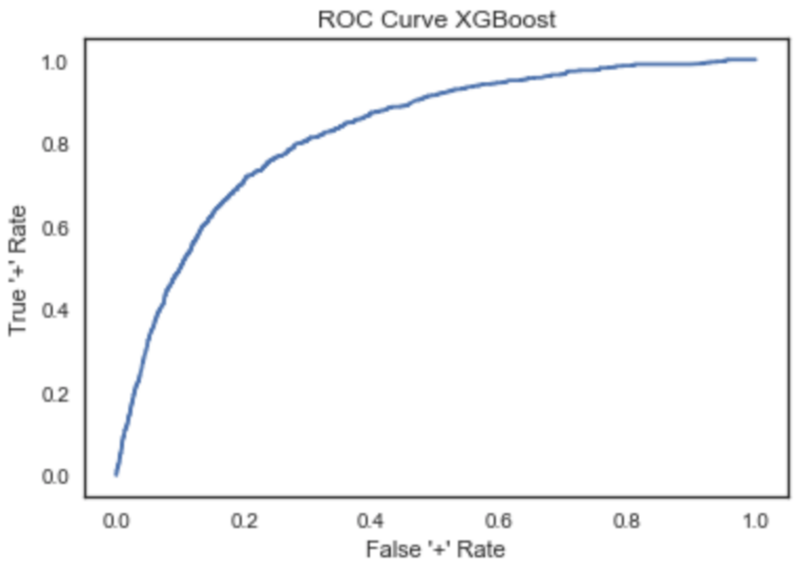
Boosting in essence is an ensemble technique that creates new models to correct the error made by existing models. These models are added sequentially. In Gradient Boosting technique , new models are created to predict the residual errors of prior models which are added together to make the final output. This uses gradient descent to minimize the loss when adding new models.

A general XGBoost model with the following hyperparameters gave an AUC score of 94.7% on the training set itself and 80.69% on the validation set indicating overfitting. Thus, I started tuning the hyperparameters using GridSearchCV. The Jupyter notebook attached will show the overall process of tuning. The final model that I kept was with the following hyperparameters.



```
clf_xgb1 = xgb.XGBClassifier(missing=9999999999,
                max_depth = 4,
                n_estimators=700,
                learning_rate=0.01,
                nthread=4,
                subsample=0.6,
                colsample_bytree=0.6,
                min_child_weight = 6,
                seed=1000)
```

This model gave an AUC score of 85.3% on the training set. Even though it is lower than the previous one, it gave a better AUC score on the validation set of 82.58%.



Random Forest

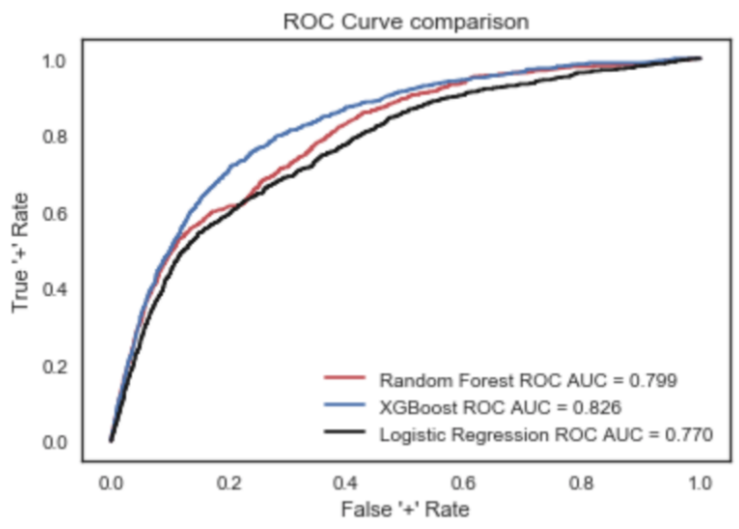
Random Forest classifier is an ensemble learning model that fits multiple decision tree algorithms on various sub samples of the given dataset. The final model is obtained by averaging to improve predictive accuracy and reduce overfitting.

I used GridSearchCV to tune the hyperparameters of the Random Forest model to obtain the following model.

```
Out[311]: RandomForestClassifier(bootstrap=True, class_weight=None, crit
          erion='entropy',
                      max_depth=6, max_features='auto', max_leaf_nodes=N
          one,
                      min_impurity_decrease=0.0, min_impurity_split=None
          ,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100, n_
          jobs=1,
                      oob_score=False, random_state=None, verbose=0,
                      warm_start=False)
```

This Random Forest classifier gave an ROC AUC Score of 79.9% on the validation dataset.

The following diagram shows the ROC curve for Random Forest, XGBoost and Logistic Regression.



Voting Classifier

This is a majority voting classifier that I used to build an ensemble model out of the models I built to further improve the ROC AUC score.

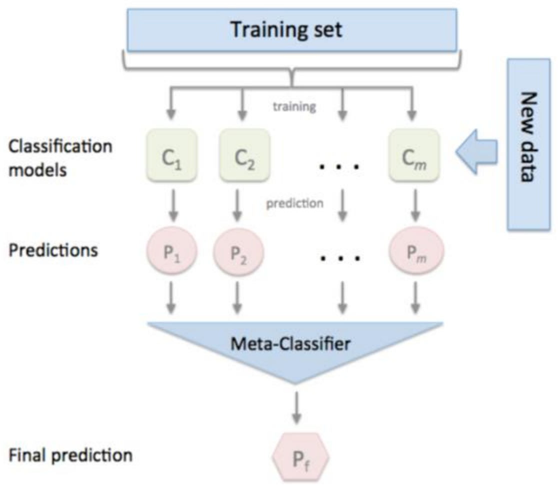
Suppose there are two classes of classifiers. Each classifier is able to predict the probability of a data point belonging to a particular class. In the voting classifier technique , these probabilities are presented to the voting classifier where it averages them and the class with the highest probability is given as the input. The average is done based upon the weights specified to the classifiers.

I used the MLP Classifier and XGBoost classifier with weights of 1 and 9 respectively to give a final ROC AUC score of 82.62 % as compared to 82.5% of the XGBoost model.

Stacking Classifier

I also tried the Stacking Classifier from mlxtend.classifier package to create a Stacked classifier using XGBoost, MLP and the voting classifier previously built.

Stacking classifier uses two levels of predictions. First a first-level of predictions are generated from a few basic classifiers then, these predictions are used along with another model at a second-level to predict the output.



This is the following model that I used as a Stacked Classifier.

```
stck_clf = StackingClassifier(classifiers=[clf_xgb2,mlp_clf2,vot_clf],
                              use_probas=True,
                              average_probas=False,
                             meta_classifier=mlp_clf2)
```

This Stacked Classifier gives an ROC AUC score of 82.61%.

Note: I decided to not use the random forest classifier in the voting or stacking classifier as it kept on bringing down the ROC AUC score.

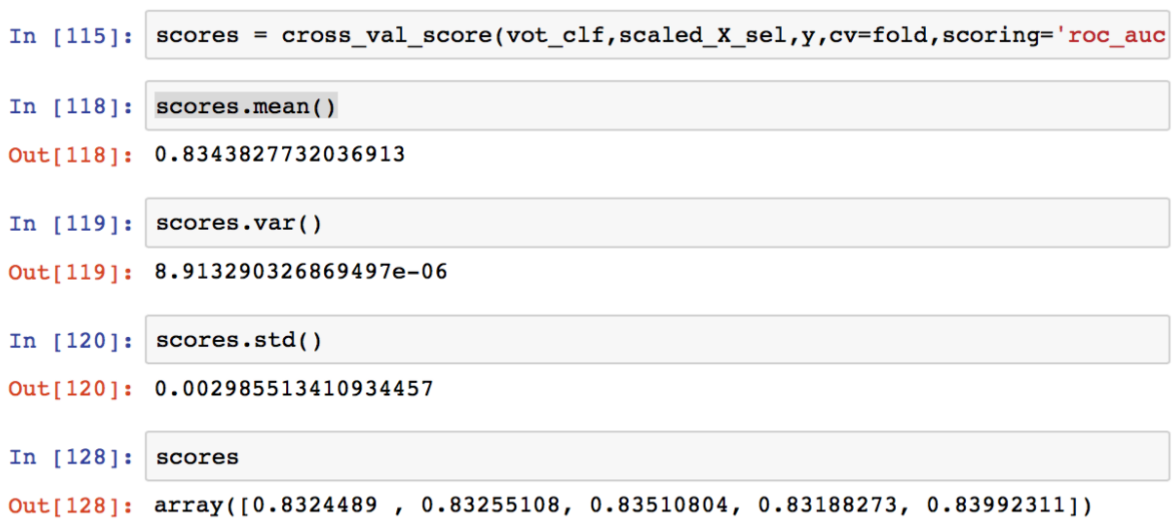
Results

The comparison of the models can be summarized as below :

MODEL	ROC AUC SCORE ON VALIDATION SET (UNSEEN DATA)
LOGISTIC REGRESSION **(BENCHMARK MODEL)**	76.9%
MLPCLASSIFIER	80.7%
XGBOOST	82.5%
RANDOM FOREST	79.9%
VOTING CLASSIFIER **(FINAL MODEL)**	82.62%
STACKING CLASSIFIER	82.61%

The voting classifier will give better result with a more tuned MLP Classifier model. Both XGBoost and MLP classifier work well with unseen data and are not much affected by outliers. Furthermore, any outliers in data can be recognized through EDA and an algorithm can be written to clean the data before running the model. Thus , this method of creation need to take place at intervals periodically to adjust with the newly found data.

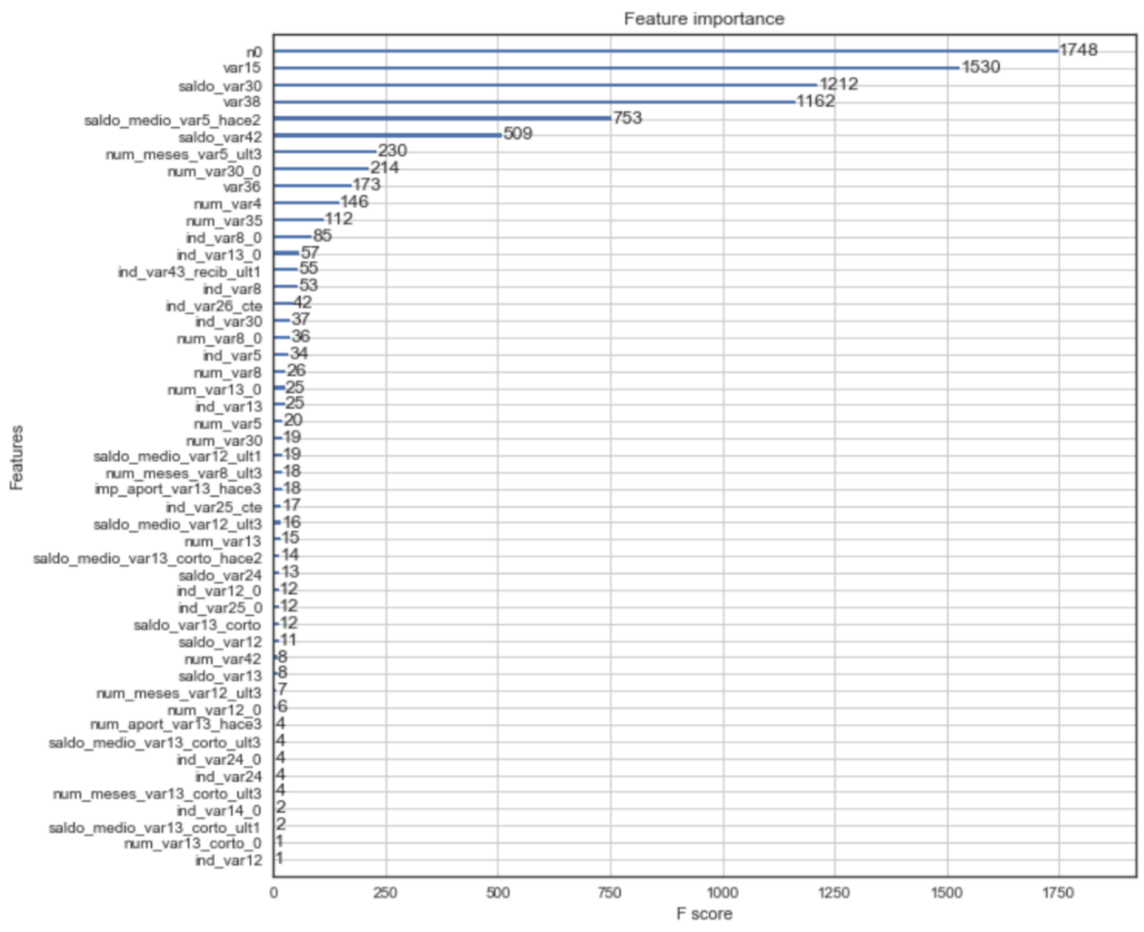
The below figure shows the result of a cross validation done over 5 folds of the training data for the scoring mechanism of roc_auc. The result gives a mean score of 83.4% with a standard deviation of .003



```
In [115]: scores = cross_val_score(vot_clf,scaled_X_sel,y,cv=fold,scoring='roc_auc

In [118]: scores.mean()

Out[118]: 0.8343827732036913

In [119]: scores.var()

Out[119]: 8.913290326869497e-06

In [120]: scores.std()

Out[120]: 0.002985513410934457

In [128]: scores

Out[128]: array([0.8324489 , 0.83255108, 0.83510804, 0.83188273, 0.83992311])
```

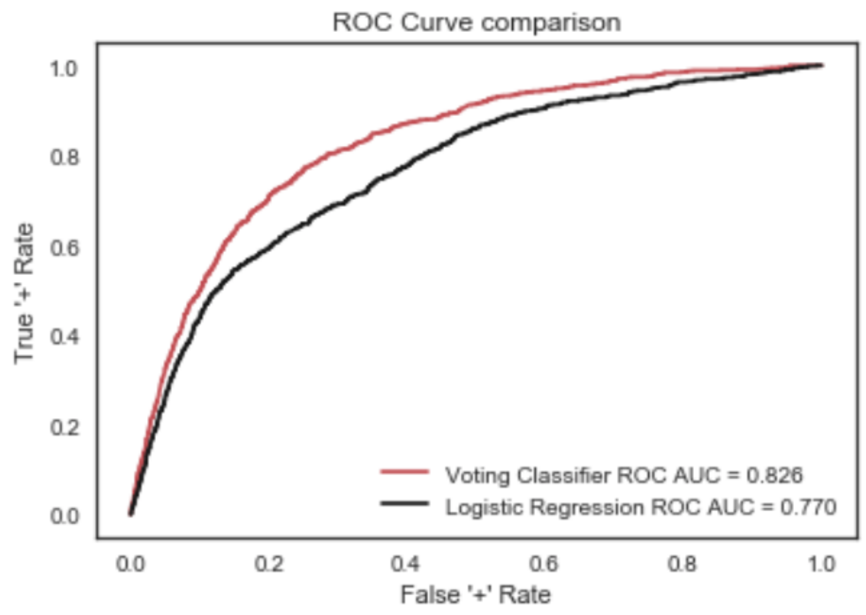
Visualizations

Feature importance according to xgboost –



The above feature importance shows that the added feature n0 is the most important feature out of all the 59 features selected using the statistical techniques.

The ROC curve below shows the difference between the final model and the benchmark logistic regression model.


ROC Curve comparison
True '+' Rate
False '+' Rate
0.0
0.2
0.4
0.6
0.8
1.0
Voting Classifier ROC AUC = 0.826
Logistic Regression ROC AUC = 0.770

Reflection

In this project I followed the following methodology to build the desired model.

- Imported the datasets.
- Visualized the dataset to get a basic understanding of the data set.
- Ran Statistical analysis on the datasets to understand distribution of the features.
- Used two types of the feature engineering methodology to reduce the number of features.
- Used Chi -2 and F-classif to reduce the number of features.
- Used Standard scaler to standardize the datasets.
- Selected some Machine learning models to use.
- Built a benchmark model of logistic regression to try and increase the ROC AUC score.
- Built basic models of MLPClassifier, XGBoost and Random Forest.
- Tuned the hyperparameter of each of these models using GridSearchCV to get the optimal values of the hyperparameters.
- Used Voting Classifier and Stacking Classifier to build ensemble models out of the already built models.

The voting classifier has been selected as the final model consisting of MLPClassifier and XGBoost with 1 : 9 weight ratio. Both of these models are optimized , but, MLPClassifier can be tuned more to adjust the ROC AUC score on the training set (96 %) and the validation score of 80.7%. This difference of score can result from overfitting. Thus, tuning the mlpclassifier can give a better result.

Challenges

I did not face any grave difficulties while working on this project except the fact that there was a deficiency in literature in this kind of study and the data definition was not provided. The absence of a SME was also felt while working on this project. This is why most of the dependency was on statistics and personal judgement.

Another issue that I faced was the time taken by GridsearchCV to find the optimal hyperparameter values for XGBoost and Random Forest. Due to lack of understanding and experience in parallel processing I was unable to speed up the process of parameter tuning and model training. A lot of time was lost while waiting for the GridsearchCV algorithm to finish finding the optimal values.

Improvement

This project gave me a chance some different feature engineering techniques and get accustomed to hyperparameter tuning. That being said, more feature engineering techniques can be used to select better features to develop more high scoring models.

Models such as XGBoost , MLP Classifier though gives a satisfying result, other classifying techniques can be used. Keras can provide a very good platform to use neural networks to produce better scoring models.

Exploratory analysis can be done on the other features which will provide more information on the features. As the features are anonymized, more research on the banking sector can provide us with a much more clearer understanding of the features. Thus, availability of a subject matter expert is very crucial for this project.

Reference

- https://www.kaggle.com/c/santander-customer-satisfaction
- Google
- https://www.kaggle.com/c/santander-customer-satisfaction/discussion
- https://stackoverflow.com/questions/38600813/names-features-importance-plot-after-preprocessing/38616018
- http://www.jdpower.com/press-releases/jd-power-2018-us-retail-banking-satisfaction-study
- https://rmsresults.com/2013/08/27/customer-satisfaction-survey-for-banks-market-research/
- https://surveymethods.com/blog/measuring-customer-satisfaction-in-the-banking-industry/