Home Credit Default Risk – Prediction

MSDS - 7335 Machine Learning

Final Project Document

Hari Narayan Sanadhya

[hsanadhya@mail.smu.edu](mailto:hsanadhya@mail.smu.edu)

Master of Science in Data Science,

Southern Methodist University, Dallas, Texas, 75275, USA

**Abstract.** Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders. This paper discusses a few machine learning algorithms to analyze this dataset and generate value out of it.

1. Introduction

Credit is the trust which allows one party to provide money or resources to another party where that second party does not reimburse the first party immediately (thereby generating a debt), but instead promises either to repay or return those resources (or other materials of equal value) later. The resources provided may be financial (e.g. granting a loan), or they may consist of goods or services (e.g. consumer credit).

To be sure with the borrowers financial health to decide the clients capability to repay the amount borrowed, lenders often pull the credit history of the client from mostly the credit bureaus. A good credit score can mean the difference between your loan being approved or denied. But when the client does not have a credit history or a bad one, then it becomes challenging for the lenders to predict clients loan capability to ensure that the loan is approved only to customers who will be able to repay full and on time.

So in the case, when the credit history is poor or non-existing, lenders may try to find other sources of financial data for the customer and generate value out of it. In this paper, we use such sources of data like data from credit cards, POS etc and then analyze and model the same to make prediction regarding clients capability to repay the loans.

1. Home Credit Organization overview:

Home Credit [http://www.homecredit.net/] Group is an international consumer finance provider with operations in 10 countries. They focus on responsible lending primarily to people with little or no credit history.

**Business model** of Home Credit starts with offering sales finance through consumer finance and then to branch-enabled lending. Customers typically start with in-store point-of-sales (POS) financing then advancing to broader consumer credit products and ultimately to fully fledged branch-based consumer lending.

**Aim** of Home Credit is to:

* Provide innovative retail financial services with a focus on mass-retail lending
* Help clients realize their dreams and ambitions in a financially responsible way
* Offer long-term, stable and interesting employment to our employees
* Encourage economic development through supporting domestic consumption, thereby improving living standards

Home Credit is an international financial institution which provides loan to customers mostly having insufficient or non-existent credit histories. Due to the poor credit background, these individuals don’t get loan approved from most of the big financial institutions like banks. Since credit history for customers is not good enough to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data - including telco and transactional information--to predict their clients' repayment abilities.

1. Data Set overview:

Data is available at the web address [https://www.kaggle.com/c/home-credit-default-risk] and is a part of an active Kaggle competition (active till 08/30/2018 – Last day of submission is 08/22/2018). Below is brief summary of all the data files that’s provided by Home Credit as a part of this competition.

* **application\_{train|test}.csv**
  + This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET). TARGET is the variable for prediction.
  + Static data for all applications. One row represents one loan in our data sample.
* **bureau.csv**
  + All client's previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in the sample provided).
  + For every loan in the sample data provided, there are as many rows as number of credits the client had in Credit Bureau before the application date.
* **bureau\_balance.csv**
  + Monthly balances of previous credits in Credit Bureau.
  + This table has one row for each month of history of every previous credit reported to Credit Bureau.
* **POS\_CASH\_balance.csv**
  + Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.
  + This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in the sample.
* **credit\_card\_balance.csv**
  + Monthly balance snapshots of previous credit cards that the applicant has with Home Credit.
  + This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in the sample.
* **previous\_application.csv**
  + All previous applications for Home Credit loans of clients who have loans in the sample.
  + There is one row for each previous application related to loans in the data sample.
* **installments\_payments.csv**
  + Repayment history for the previously disbursed credits in Home Credit related to the loans in the sample.
  + There is
    - one row for every payment that was made plus
    - one row each for missed payment.
  + One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit related to loans in the sample.

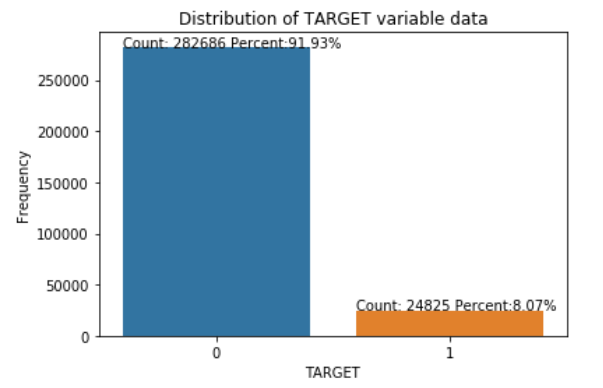


**Figure 1**: Entities and Relationship – This figure gives a brief summary of the Home Credit dataset entities and how the various entities are connected to each other.

1. Problem Statement

For the clients that have insufficient or non-existent credit histories, predict the clients repayment capabilities to ensure that clients capable of repayment are not rejected and have a great customer experience.

TARGET variable in the application file is the variable that needs to be predicted. Train data has this value prefilled so that the machine learning models can learn from the existing dataset and find out the patterns or relationship within the dataset that would help in deciding the client's repayment capabilities.



**Figure 2**: Distribution of the TARGET variable– This figure shows the count and percentage of each distinct value of TARGET variable in the training dataset.

As shown in figure 2, the dataset is highly imbalanced with 8% of the entire dataset having TARGET value of 1 and remaining 92% having value of 0. TARGET value of 0 means that the client had paid the loan on time and there were no hick-ups in doing so. TARGET value of 1 mean that the client not able to pay the loan in timely manner and there were issues with his/her loan repayment. The machine learning models were trained with the imbalanced nature of the dataset considered by either using:

1. **class-weight** option with value as **"balanced"** while initializing the machine learning model so that the machine learning model automatically adjust weights inversely proportional to class frequencies in the input data.
2. **oversampling** where new samples were generated in the dataset for the class (TARGET=1) which are under-represented. Used SMOTE (Synthetic Minority Over-sampling Technique) to perform oversampling.
3. Classification Performance Measurement

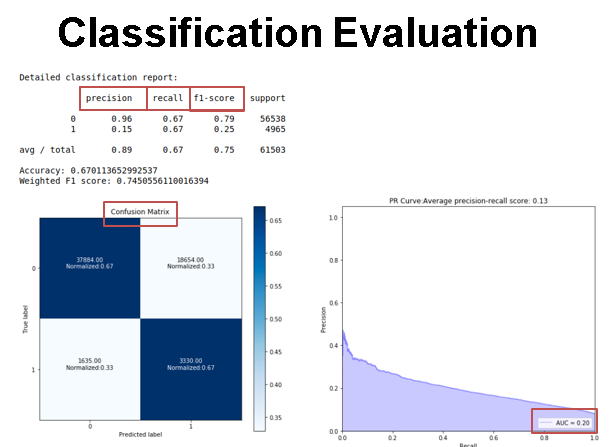
To evaluate the performance of various machine learning models implemented to classify the dataset for the TARGET variable prediction, below metrics were used:

* **Confusion Matrix**: It’s a specific table layout that allows visualization of the performance of classification algorithm. It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions. This table shows visually if the model is confusing with the class of the TARGET variable i.e. is it mislabeling one as another and vice-versa.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True Condition | |
|  |  | Positive | Negative |
| Predicted Condition | Positive | True positive (TP), Power | False positive (FP), Type I error |
| Negative | False Negative (FN), Type II Error | True Negative (FP) |

* **Accuracy**: the accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's true value.
* **Precision**: It's the number of correct positive results divided by the number of all positive results returned by the classifier
* **Recall**: It's the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).
* **F1-score**: It’s the measure of test accuracy and is the harmonic mean of precision and recall.
* **PR Curve (Precision recall curve)**: Precision and Recall are inversely related, ie. as Precision increases, recall falls and vice-versa. A balance between these two needs to be achieved by the Machine Learning system, and to achieve this and to compare performance, the precision-recall curves come in handy. With Precision plotted on the y-axis and recall on x-axis, the area-under-curve (AUC) for this curve shows how much balanced precision and recall metrics are. Ideally it should be one meaning that they are completely balanced.

**NOTE**: Precision, recall and F1-score don't consider true negatives and thus won't be affected by the relative imbalance. This is the main reason to use them for performance evaluation.



**Figure 3**: Classification Evaluation Metrics– This figure shows the various Machine Classification measurement metrics that were used to evaluate the performance of Machine Learning models used in this paper.

1. ETL and Deep Feature Synthesis:

**Application Data file analysis:-**

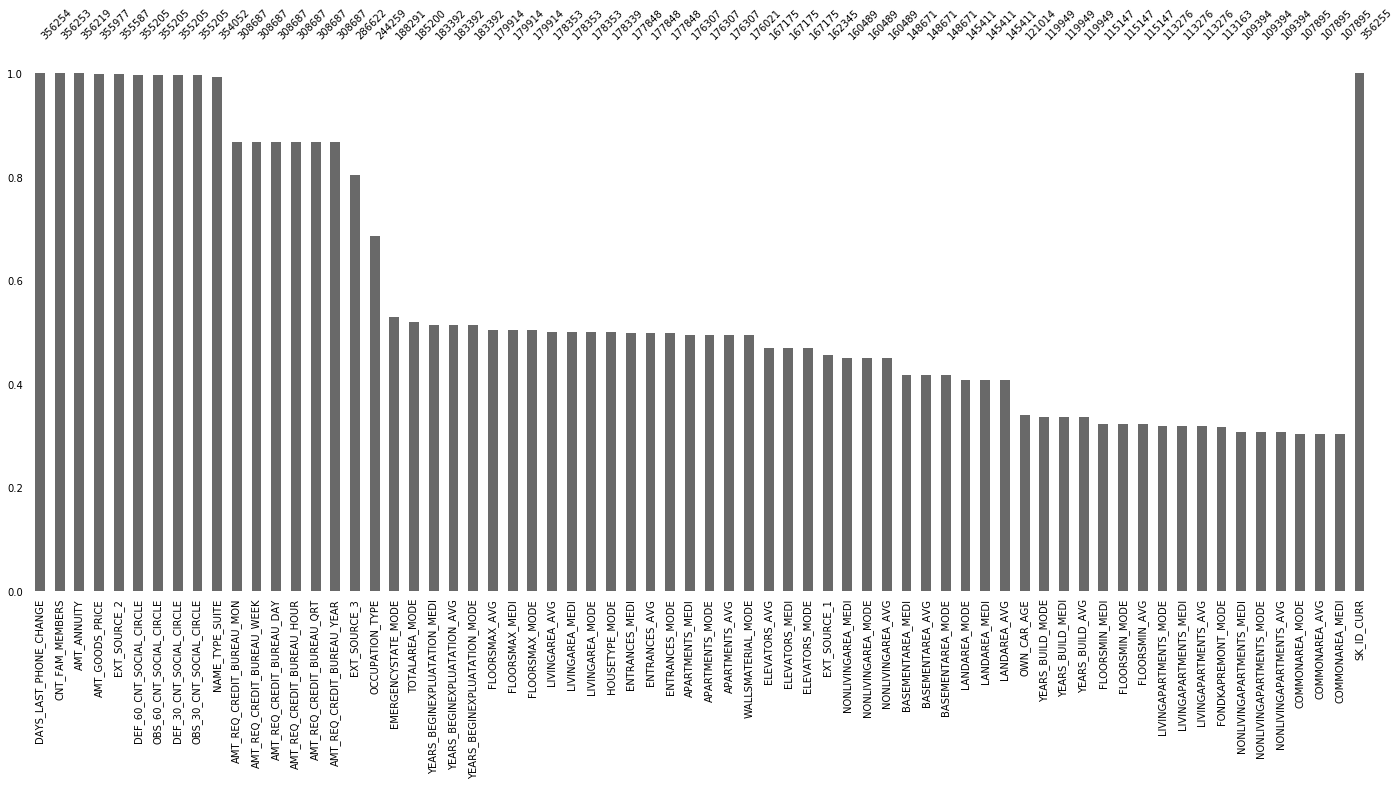
Since application file is the main table in the dataset which has unique record for every loan application and has the TARGET variable (prediction variable) as a part of the dataset, this file was analyzed separately.

Shape of Train data: (307511, 122)

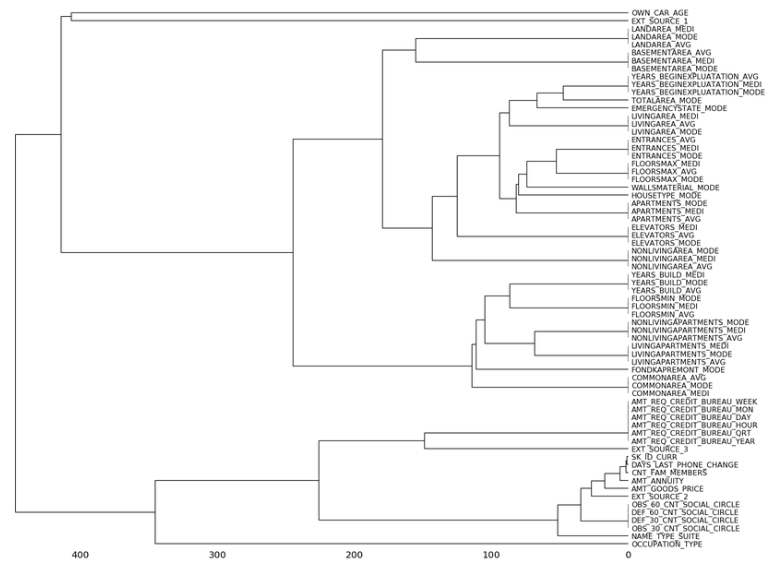
Shape of Test data: (48744, 121) # TARGET variable not present in the test file

The TARGET variable in the train data has two distinct values i.e. 0 and 1 so in order to do the analysis on the entire application data, TARGET variable was created in the test file with a value of -1 and then both the test and train data was joined together. After the analysis, they were separated back by using the TARGET variable value.

Shape of Combined Application data: (356255, 122)



**Figure 4**: Barchart– This figure shows a barchart of the application data with the bar height represents the percent of the non-missing values in the dataset.



**Figure 5**: Dendrogram– This figure shows a Dendrogram of the application data which tries to group variables together in similar group if the missing data in those variables always happen at the same time i.e. if two or more variables have missing data always at the same time, then they end up in the same group.

As shown in figure 4, the application data has many columns with missing data.

Out of 122 columns in the dataset, 67 have missing data with minimum 0.00028 and maximum 69.71411 missing data

|  |  |
| --- | --- |
| Field/Fields | Missing data imputation |
| NAME\_TYPE\_SUITE | Missing data filled with existing category 'Unaccompanied' |
| OCCUPATION\_TYPE | Missing data imputed with new value 'Unemployed/Not Specified' |
| EMERGENCYSTATE\_MODE | Missing data filled with existing category 'No' |
| FONDKAPREMONT\_MODE | Missing data filled with existing category 'not specified' |
| HOUSETYPE\_MODE, WALLSMATERIAL\_MODE | Missing data imputed with new value 'Not Specified' |
| OWN\_CAR\_AGE | For the records where OWN\_CAR\_AGE is missing but the FLAG\_OWN\_CAR is Y, changed the flag to N. Missing data filled with -1. |
| EXT\_SOURCE\_1, EXT\_SOURCE\_2, EXT\_SOURCE\_3 | Since these fields shows the standardized credit score of the applicant from external sources like credit bureau's, new flag field was created indicating the presence or absence of the external source. Missing data was then imputed with zero. |
| 'DEF\_60\_CNT\_SOCIAL\_CIRCLE', 'OBS\_60\_CNT\_SOCIAL\_CIRCLE', 'DEF\_30\_CNT\_SOCIAL\_CIRCLE', 'OBS\_30\_CNT\_SOCIAL\_CIRCLE' | As analyzed from figure 5, all these fields are in the same group in the figure which means that the fields have missing data at the same time, These fields are indicative of the social circle of the applicant and if the data is missing, then its indicative of the social circle information of the applicant is not available. So created a new flag variable which is indicative of whether the data in all these fields is present or missing. The missing data was then imputed with Zero. |
| DAYS\_LAST\_PHONE\_CHANGE, CNT\_FAM\_MEMBERS, AMT\_ANNUITY, AMT\_GOODS\_PRICE | Missing data was imputed with zero. |
| APARTMENTS\_AVG, BASEMENTAREA\_AVG, YEARS\_BEGINEXPLUATATION\_AVG, YEARS\_BUILD\_AVG, COMMONAREA\_AVG, ELEVATORS\_AVG, ENTRANCES\_AVG, FLOORSMAX\_AVG, FLOORSMIN\_AVG, LANDAREA\_AVG, LIVINGAPARTMENTS\_AVG, LIVINGAREA\_AVG, NONLIVINGAPARTMENTS\_AVG, NONLIVINGAREA\_AVG, APARTMENTS\_MODE, BASEMENTAREA\_MODE, YEARS\_BEGINEXPLUATATION\_MODE, YEARS\_BUILD\_MODE, COMMONAREA\_MODE, ELEVATORS\_MODE, ENTRANCES\_MODE, FLOORSMAX\_MODE, FLOORSMIN\_MODE, LANDAREA\_MODE, LIVINGAPARTMENTS\_MODE, LIVINGAREA\_MODE, NONLIVINGAPARTMENTS\_MODE, NONLIVINGAREA\_MODE, APARTMENTS\_MEDI, BASEMENTAREA\_MEDI, YEARS\_BEGINEXPLUATATION\_MEDI, YEARS\_BUILD\_MEDI, COMMONAREA\_MEDI, ELEVATORS\_MEDI, ENTRANCES\_MEDI, FLOORSMAX\_MEDI, FLOORSMIN\_MEDI, LANDAREA\_MEDI, LIVINGAPARTMENTS\_MEDI, LIVINGAREA\_MEDI, NONLIVINGAPARTMENTS\_MEDI, NONLIVINGAREA\_MEDI, TOTALAREA\_MODE | All the fields listed are associated to the place where the client lives. So if its missing, it may mean that the client has not provided this information or that information is not applicable to the place where the client currently stays. So, the missing values in all these fields were replaced by zero. |

**Deep Feature Synthesis (DFS)**

Since the data provided was in form of different files which were not grouped together logically, it was required to intelligently merge the dataset into one which would then be used as the input features by the machine learning models.

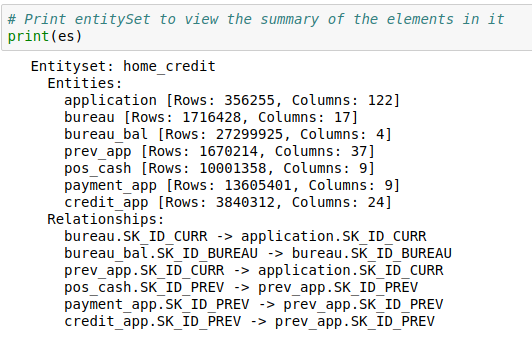
Automating feature engineering optimizes the process of building and deploying accurate machine learning models by handling the tedious task of feature generation. FeatureTools Deep Feature Synthesis (DFS) is an automated feature engineering method developed by MIT. To implement DFS, there are a few additional steps that are needed to be performed.

1. Define an EntitySet which is similar to defining schema in RDBMS.
2. All the different data files are treated as different entity and is added into the EntitySet. This is like table creation in RDBMS. Here in DFS, the entities must have a unique identifier (single column) which if not there in the dataset, can be created.
3. Define relationship between entities listing the fields that relates data in one file to another. This is similar to foreign key creation in RDBMS.

FeatureTools DFS application was used to group data in these entities into a single data file. Sum, Mean and Max were the primitives that were used in this analysis which means that while grouping the data from different entities together, many to one grouping was done to generate the sum, average and maximum values of all the different features from all the entities that were present in the EntitySet.

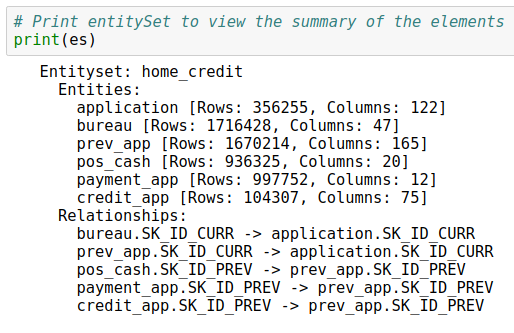
DFS was performed in two separate ways: -

**DFS Approach 1**: -Without performing any manual analysis on the other data files, DFS was performed on the data. For this, since the data structure was not altered, the entities and relationship were defined by using the ER diagram provided by Home Credit (Figure 1)



**Figure 6**: EntitySet for approach 1– This figure shows the summary of the EntitySet created using approach 1. The EntitySet home\_credit has 7 entities and they were linked with each other using 6 different relationships.

**DFS Approach 2**: - Manual Data analysis followed by DFS: This was done because the summarization of the dataset needed to be performed at distinct levels. The bureau balance file had applications with different status. So, the grouping needed to be done for each of the status separately. This means that when the DFS completes its execution, the outcome should have bureau balance data mentioned separately for applications with different status. Here for each dataset, one-hot encoding was done to each of the dataset before creating the entities. Records with different status were separated and then the records with the same status were grouped manually to generate the sum, max and average field values. These different datasets created were then regrouped together so that for each application, the sum, mean and max values for the data fields will be listed in different columns based on the status.

****

**Figure 7**: EntitySet for approach 2– This figure shows the summary of the EntitySet created using approach 2. This EntitySet home\_credit has 6 entities and they were linked with each other using 5 different relationships.

Comparing Figure 6 with Figure 7, the second EntitySet has more columns in each of the entities due to one-hot encoding followed by separation of the dataset based of the status of the record. Also bureau and bureau\_bal entities (with a total of 21 columns) in figure 6 was reduced to single entity (with total 47 columns).

1. Data Scaling and conversion:-

The output dataset from the two DFS approaches has fields with varying range of values. Some of the fields had very large value which caused the algorithms to break while training. The error message received was "ValueError: Input contains NaN, infinity or a value too large for dtype('float64')." Rechecking the data confirmed that there was no missing or infinite value in the dataset but still this error popped up every now and then. Also, very large values of some of the columns caused them to present themselves as more important while performing feature reduction. So instead of using the dataset as it is, scaling was done to reduce the dataset to smaller range.

For DFS approach 1 dataset, Scandard Scaling, Minimum Maximum scaling and Maximum Absolute scaling were the scaling algorithms from sklearn library that were used to do the analysis.

For DFS approach 2 dataset, Logarithm conversion of data columns with high range was performed. The logarithm scaled data was then normalized as well to see the impact on the result.

1. Feature Reduction:-

The DFS Approach 1 had generated the final output dataset with total of 716 features whereas the DFS approach 2 resulted in final output dataset with 1925 features. To reduce the number of features to the machine learning classification model, the algorithms used were:

1. Principle Component Analysis (PCA): - Linear Dimension reduction algorithm

To find the number of principle components that should be used in the machine learning model, use the Scree Plot.

|  |  |
| --- | --- |
| DFS Approach 1 | DFS Approach 2 |
|  |  |
| Number of Principle components to use with   1. Unscaled Data – 8 2. Standard Scaled Data – 110 3. MinMax Scaled Data – 35 4. MaxAbs Scaled Data - 35 | Number of Principle components to use with   1. Log scaled Data – 8 2. Standard Scaled Data – 8 |

1. BernoulliRBM (Restricted Boltzmann machines): - Unsupervised Neural network Algorithm and Non-Linear feature learner. This algorithm trains the unsupervised Neural network based on a probabilistic model. It assumes the input to be binary or between 0 and 1 so only the scaled data was used with this algorithm.
2. Output of Classification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification algorithm output using only the Application file data** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Adaboost Oversampling | 0 | 0.93 | 0.44 | 0.6 |  |
| 1 | 0.09 | 0.64 | 0.16 |  |
| total | 0.87 | 0.46 | 0.57 | 0.46 |
| RandomForest oversampling | 0 | 0.91 | 0.44 | 0.6 |  |
| 1 | 0.09 | 0.64 | 0.16 |  |
| total | 0.87 | 0.46 | 0.56 | 0.46 |
| Standard Scaled Adaboost Oversampling | 0 | 0.93 | 0.93 | 0.93 |  |
| 1 | 0.19 | 0.18 | 0.18 |  |
| total | 0.87 | 0.87 | 0.87 | 0.87 |
| Standard Scaled RandomForest oversampling | 0 | 0.92 | 0.87 | 0.96 |  |
| 1 | 0.26 | 0.02 | 0.04 |  |
| total | 0.87 | 0.92 | 0.88 | 0.91 |
| Standard Scaled Decision Tree oversampling | 0 | 0.93 | 0.9 | 0.91 |  |
| 1 | 0.14 | 0.18 | 0.16 |  |
| total | 0.86 | 0.84 | 0.85 | 0.84 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 1 – PCA + Logistic Regression Output** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Logistic Regression With PCA – Unscaled data | 0 | 0.94 | 0.37 | 0.53 |  |
| 1 | 0.09 | 0.73 | 0.16 |  |
| Total | 0.87 | 0.4 | 0.5 | 0.394 |
| Logistic Regression With PCA – StandardScaled data | 0 | 0.96 | 0.67 | 0.79 |  |
| 1 | 0.15 | 0.67 | 0.25 |  |
| total | 0.89 | 0.67 | 0.75 | 0.67 |
| Logistic Regression With PCA – MinMaxScaled data | 0 | 0.95 | 0.63 | 0.76 |  |
| 1 | 0.13 | 0.64 | 0.22 |  |
| total | 0.89 | 0.63 | 0.72 | 0.632 |
| Logistic Regression With PCA – MaxAbsScaled data | 0 | 0.95 | 0.62 | 0.75 |  |
| 1 | 0.13 | 0.63 | 0.21 |  |
| total | 0.88 | 0.62 | 0.71 | 0.622 |

Hyperparameter of Logistic regression were tuned using Sklearn Pipeline with 3 fold cross validation.

The best model performance after tuning with StandardScaled and MinMaxScaled data came out as below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 1 – PCA + Logestic Regression Output** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Logistic Regression With PCA – ScandardScaled data | 0 | 0.96 | 0.67 | 0.79 |  |
| 1 | 0.15 | 0.67 | 0.24 |  |
| Total | 0.9 | 0.67 | 0.75 | 0.67 |
| Logistic Regression With PCA – MinMaxScaled Data | 0 | 0.96 | 0.66 | 0.78 |  |
| 1 | 0.15 | 0.68 | 0.24 |  |
| total | 0.89 | 0.66 | 0.74 | 0.66 |

Hyperparameter tuning of Logistic regression didn’t help much with the improvement in the model performance and the result is the almost the same as the result with default parameters. So, ended up using the default parameters only as the best model for PCA with Logistic Regression.

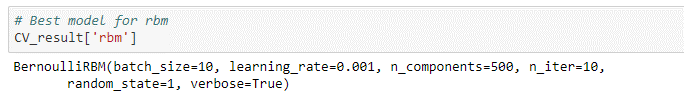
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 1 - Standard Scaled data with other classification algorithms (no parameter reduction)** | | | | | |
| RandomForest oversampling | 0 | 0.92 | 0.99 | 0.95 |  |
| 1 | 0.24 | 0.04 | 0.06 |  |
| total | 0.87 | 0.91 | 0.88 | 0.91 |
| RandomForest oversampling | 0 | 0.93 | 0.96 | 0.94 |  |
| 1 | 0.22 | 0.11 | 0.15 |  |
| total | 0.87 | 0.9 | 0.88 | 0.9 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 2 - PCA + Logestic Regression Output** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Logistic Regression with log reduced data | 0 | 0.94 | 0.61 | 0.74 |  |
| 1 | 0.12 | 0.58 | 0.19 |  |
| total | 0.88 | 0.61 | 0.7 | 0.61 |
| Logistic Regression with log reduced normalized data | 0 | 0.94 | 0.61 | 0.74 |  |
| 1 | 0.12 | 0.58 | 0.19 |  |
| total | 0.88 | 0.61 | 0.7 | 0.61 |

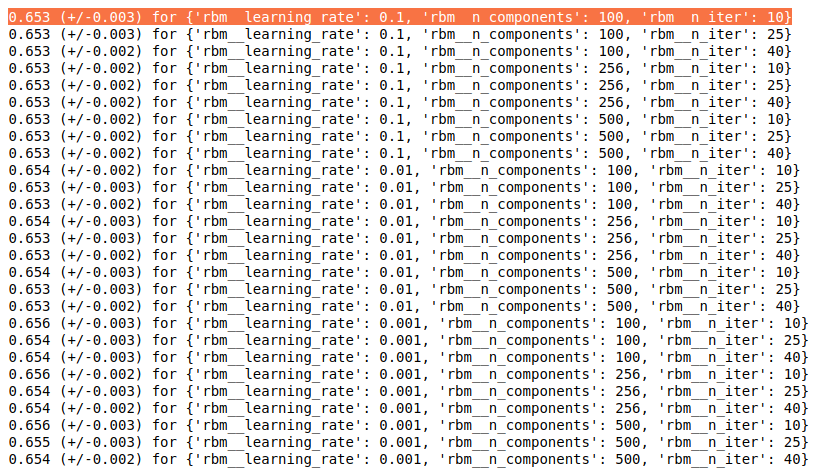
1. **BernoulliRBM feature extractor and a LogisticRegression classifier**

The RBM neural network was tuned for optimal hyperparameter selection. The input dataset used was the standard scaled dataset as this algorithm requires the input to be wither binary or between 0 and 1.

|  |  |  |
| --- | --- | --- |
| **Tuning Parameters** | | |
| **learning\_rate** | **[0.1, 0.01, 0.001]** |
| **#\_epochs** | **[10, 25, 40]** |
| **hidden\_layers** | **[100, 256, 500]** |



Though the above snip showed the best model for the RBM neural network, on analyzing the performance of each iteration of hyperparameter tuning combination (figure 8), since the improvement was not significant, we decided to go with the default configuration.

****

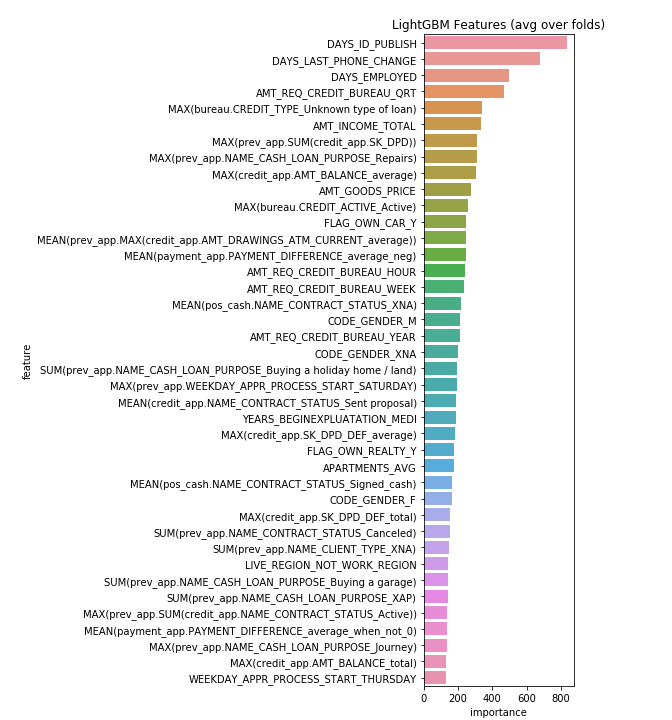
**Figure 8**: Output of RBM Hyperparameter Tuning– This figure shows mean test score when different combinations of hyperparameters are put to work.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 1 - Standard Scaled data with RBM combined with other classification algorithms** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Logistic Regression with class\_weight option | 0 | 0.92 | 0.68 | 0.78 |  |
| 1 | 0.09 | 0.36 | 0.14 |  |
| Total | 0.86 | 0.66 | 0.73 | 0.66 |
| QuadraticDiscriminantAnalysis with class\_weight option \*\* | 0 | 0.9 | 0 | 0 |  |
| 1 | 0.08 | 1 | 0.15 |  |
| total | 0.84 | 0.08 | 0.01 | 0.08 |
| Logistic Regression oversampling | 0 | 0.91 | 0.32 | 0.48 |  |
| 1 | 0.08 | 0.64 | 0.14 |  |
| total | 0.84 | 0.35 | 0.45 | 0.35 |
| RandomForest oversampling | 0 | 0.91 | 0.32 | 0.48 |  |
| 1 | 0.08 | 0.64 | 0.14 |  |
| total | 0.84 | 0.35 | 0.45 | 0.35 |
| Adaboost oversampling | 0 | 0.91 | 0.32 | 0.48 |  |
| 1 | 0.08 | 0.64 | 0.14 |  |
| total | 0.84 | 0.35 | 0.45 | 0.35 |

\*\* - QuadraticDiscriminantAnalysis performed poorly because RBM learns the non-linear relationship between the features and since the input to the QuadraticDiscriminantAnalysis model is the output of the RBM model, a non linear classifier with a non-linear feature learner is redundant and cause this issue. Python also gives a warning message in this case stating " Variables are collinear "

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 2 - RBM combined with other classification algorithms** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Normalized data with Logistic Regression with class\_weight option | 0 | 0.92 | 0.08 | 0.15 |  |
| 1 | 0.08 | 0.92 | 0.15 |  |
| Total | 0.86 | 0.15 | 0.15 | 0.15 |
| Normalized data with QuadraticDiscriminantAnalysis with class\_weight option \*\* | 0 | 0.93 | 0 | 0 |  |
| 1 | 0.08 | 1 | 0.15 |  |
| total | 0.86 | 0.08 | 0.01 | 0.08 |
| Normalied data with randomforest classifier | 0 | 0.92 | 1 | 0.96 |  |
| 1 | 0.3 | 0.01 | 0.14 |  |
| total | 0.87 | 0.92 | 0.88 | 0.92 |
| Normalized data with Light Gradient Boosting Machine | 0 | 0.92 | 1 | 0.96 |  |
| 1 | 0.53 | 0.04 | 0.07 |  |
| total | 0.89 | 0.92 | 0.89 | 0.92 |

\*\* - QuadraticDiscriminantAnalysis performed poorly because RBM learns the non-linear relationship between the features and since the input to the QuadraticDiscriminantAnalysis model is the output of the RBM model, a non linear classifier with a non-linear feature learner is redundant and cause this issue. Python also gives a warning message in this case stating " Variables are collinear "



**Figure 9**: Feature importance from LightGBM model – The figure shows the top 40 features that were used by LightGBM model to make the classification.

1. Output of Clustering – Unsupervised learning

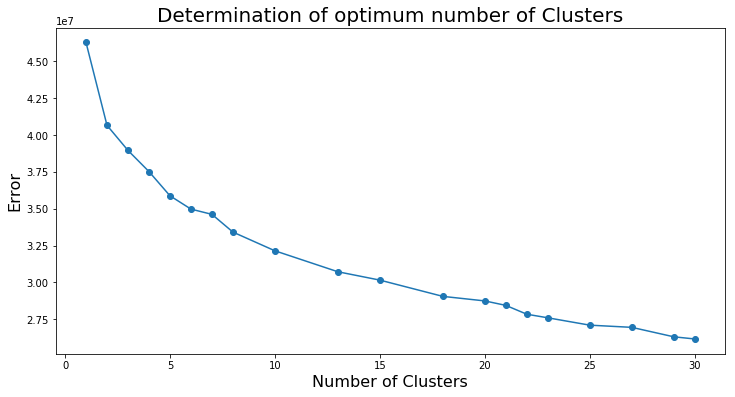
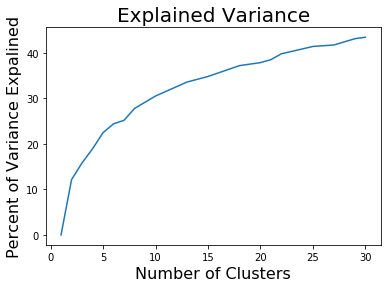
The following approaches were used to determine the number of clusters in the dataset (Algorithms implemented but didn’t work): -

1. DBSCAN algorithm (Density-Based Spatial Clustering of Applications with Noise): This algorithm finds core samples of high density and expands clusters from them. It is good for clusters with high density. With the given dataset, DBSCAN always returned the number of clusters as 1.
2. AffinityPropagation - AffinityPropagation creates clusters by sending messages between pairs of samples until convergence. Affinity Propagation chooses the number of clusters based on the data provided so generalization is achieved by adjusting the two parameters namely preference and damping factor. With different values of the parameters for affinity propagation, the number of clusters were in high hundreds.

|  |  |  |  |
| --- | --- | --- | --- |
| Damping | Convergence\_iter | Preference | #\_of\_clusters |
| 0.5 | 15 | None | 1538 |
| 0.75 | 15 | None | 1408 |
| 0.75 | 50 | None | 1408 |
| 0.75 | 5 | None | 1574 |
| 0.75 | 15 | -50 | 10753 |
| 0.75 | 15 | 50 | 17813 |

**DFS approach 1**

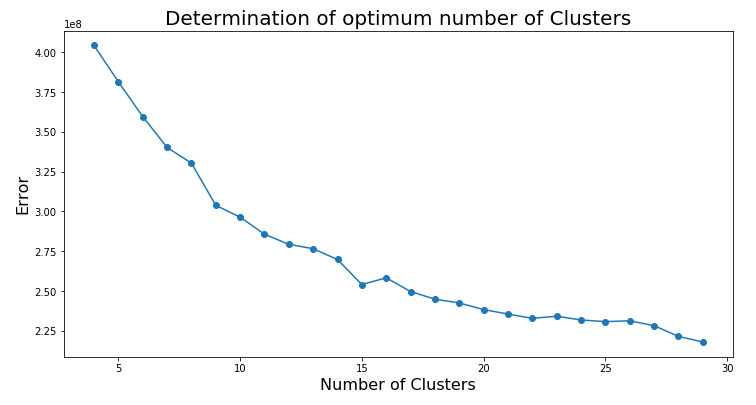
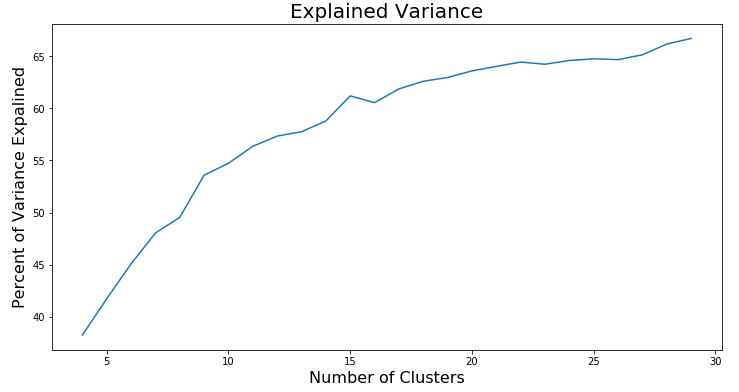
1. Use of elbow method with k-means to determine the number of clusters in the dataset.

****

**Figure 10**: K-means elbow method – The elbow method is used to determine the optimal number of clusters into which the dataset can be divided. Any of the above two figures can be used to determine the same. Had chosen n\_clusters=6 after looking into the plots.

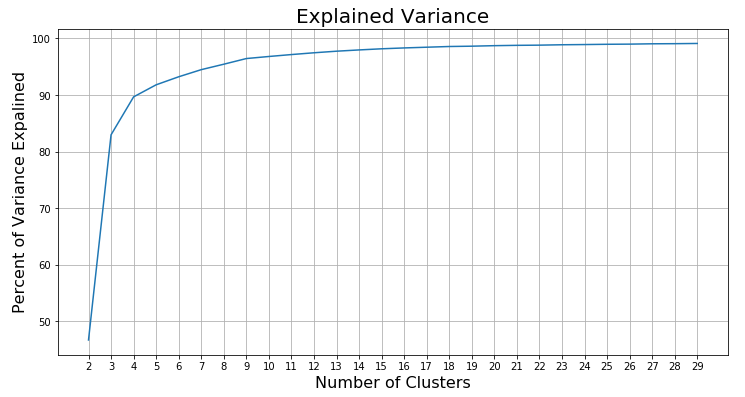
**DFS approach 2**

1. Use of elbow method with MiniBatchKMeans to determine the number of clusters in the dataset.

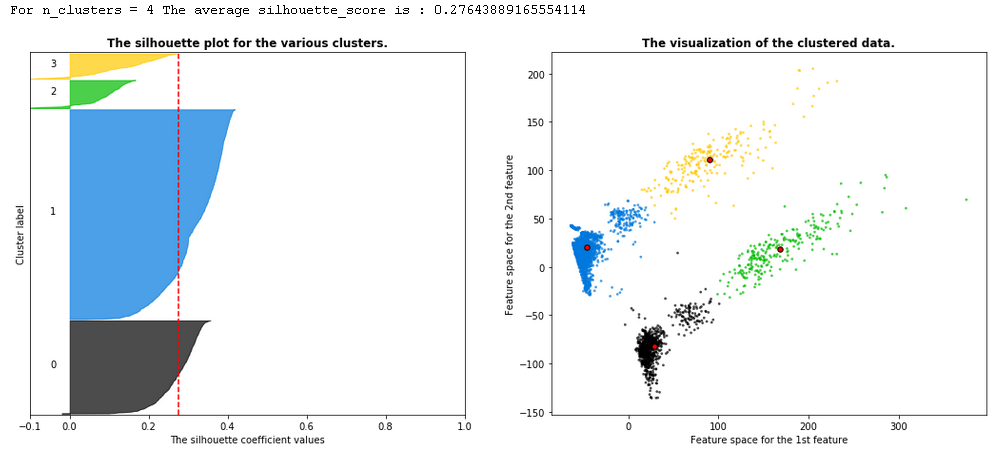


**Figure 10**: MiniBatchKMeans elbow method – The elbow method is used to determine the optimal number of clusters into which the dataset can be divided. Any of the above two figures can be used to determine the same. Had chosen n\_clusters=9 after looking into the plots.

1. Use of elbow method with k-means to determine the number of clusters in the dataset.



**Figure 10**: K-means elbow method – The elbow method is used to determine the optimal number of clusters into which the dataset can be divided. Had chosen n\_clusters=4 after looking into the plots.



**Figure 11**: K-means Silhouette Plot and clustered data visualization – Using the n\_clusters as 4 obtained from elbow method, the above figure shows the silhouette plot and the clustered data visualization.

1. Cluster data as feature for classification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 1 - Cluster information as input features to classification algorithms** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Probability of data point belonging to each cluster obtained from K-means clustering as feature to random forest algorithm with oversampling | 0 | 0.92 | 1 | 0.96 |  |
| 1 | 0.24 | 0.01 | 0.03 |  |
| Total | 0.87 | 0.92 | 0.88 | 0.92 |
| Probability of data point belonging to each cluster obtained from K-means clustering as feature to decision tree algorithm with oversampling | 0 | 0.92 | 0.9 | 0.91 |  |
| 1 | 0.13 | 0.17 | 0.14 |  |
| total | 0.86 | 0.84 | 0.85 | 0.84 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DFS Approach 2 - Cluster information as input features to classification algorithms on log scaled data** | | | | | |
|  |  | precision | recall | f1-score | Accuracy |
| Probability of data point belonging to each cluster obtained from MiniBatchKMeans clustering as feature to random forest algorithm | 0 | 0.92 | 1 | 0.96 |  |
| 1 | 0.12 | 0.01 | 0.01 |  |
| Total | 0.86 | 0.92 | 0.88 | 0.96 |
| Predicted cluster number of data point from MiniBatchKMeans clustering as feature to random forest algorithm with class\_weight='balanced' option | 0 | 0.93 | 0.64 | 0.76 |  |
| 1 | 0.1 | 0.48 | 0.17 |  |
| Total | 0.87 | 0.62 | 0.71 | 0.62 |
| Probability of data point belonging to each cluster obtained from MiniBatchKMeans clustering as feature to LGBMClassifier with class\_weight='balanced' option | 0 | 0.93 | 0.83 | 0.88 |  |
| 1 | 0.12 | 0.25 | 0.16 |  |
| Total | 0.86 | 0.79 | 0.82 | 0.79 |
| Probability of data point belonging to each cluster obtained from MiniBatchKMeans clustering as feature to random forest algorithm with oversampling | 0 | 0.92 | 0.89 | 0.91 |  |
| 1 | 0.11 | 0.16 | 0.13 |  |
| Total | 0.86 | 0.83 | 0.84 | 0.83 |
| Probability of data point belonging to each cluster obtained from MiniBatchKMeans clustering as feature to decision tree algorithm with oversampling | 0 | 0.92 | 0.79 | 0.85 |  |
| 1 | 0.09 | 0.25 | 0.14 |  |
| Total | 0.86 | 0.75 | 0.79 | 0.75 |
| Probability of data point belonging to each cluster obtained from K-means clustering as feature to LGBMClassifier algorithm with oversampling | 0 | 0.93 | 0.76 | 0.84 |  |
| 1 | 0.11 | 0.33 | 0.16 |  |
| Total | 0.86 | 0.73 | 0.78 | 0.73 |
| Standard Scaled Probability of data point belonging to each cluster obtained from MiniBatchKMeans clustering as feature to random forest algorithm with oversampling | 0 | 0.92 | 0.89 | 0.91 |  |
| 1 | 0.11 | 0.16 | 0.13 |  |
| Total | 0.86 | 0.83 | 0.84 | 0.83 |
| Standard Scaled Probability of data point belonging to each cluster obtained from MiniBatchKMeans clustering as feature to decision tree algorithm with oversampling | 0 | 0.93 | 0.79 | 0.85 |  |
| 1 | 0.11 | 0.27 | 0.15 |  |
| Total | 0.86 | 0.75 | 0.8 | 0.75 |
| Standard Scaled Probability of data point belonging to each cluster obtained from K-means clustering as feature to LGBMClassifier algorithm with oversampling | 0 | 0.93 | 0.76 | 0.84 |  |
| 1 | 0.11 | 0.32 | 0.16 |  |
| Total | 0.86 | 0.73 | 0.78 | 0.73 |