Prediction of Home Credit Loan Default with Machine Learning Models

Maria Mahbub
University of Tennessee, Knoxville

Abstract

This report reflects the implementations of several machine learning algorithms with a goal to predict loan applicants' repayment capability. For the analysis and prediction, data sets from one of the Kaggle competetions have been used. The implemented algorithms are statistical and parametric Gaussian Naive-Bayes (Gaussian NB), non-parametric k-nearest neighbors (kNN), logistic regression (LR), support vector machine classifier (SVC) and a gradient boosting framework based on non-statistical tree-based learning algorithm lightGBM (LGBM). Being exposed to a class imbalance problem, the models have been evaluated based on their Area Under Receiver Operating Characteristics Curve (ROC-AUC) which was also the evaluation metric for the Kaggle competition. After building the baseline models, two best performing ones, lightGBM and logistic regression have been improved upon their baseline AUC.

1. Introduction:

Getting rejected for loan applications due to insufficient or non-existent credit histories in spite of being capable of repaying loans, is a common struggle for many people. Many times, these people are taken advantage of their situation by untrustworthy lenders. Home Credit is an international consumer finance provider whose goal is to lend to these underserved population. In order to make sure a positive loan experience, Home Credit utilizes a variety of alternative data in order to build various statistical and machine learning models to predict their clients' ability to repay loans.

In 2018 they hosted a Kaggle competition with a challenge to reveal the full potential of their data which would eventually ensure them proper client selection and provide a platform for choosing suitable loan principal, maturity and repayment calendar for them [1]. This report explores their data extensively and experiments with different machine learning algorithms for loan default prediction in order to find the best-suited one.

Credit default is a regular phenomenon in the real world. It can cause huge loss to banks and other financial entities. Several research works have been done with a view to establishing an effective model to predict defaulters. Some of these works include implementation of logistic regression on a vast database of the Spanish companies in order to predict corporate bankruptcy by Bartual et al. [2], implementation of logistic regression and support vector machine by Nehrebecka in order to assess the credit risk of non-financial enterprises which has the potential to lessen the risk of negative effects on these sectors [3], assessment of credit default prediction on the banking market in Bosnia and Herzegovina nationwide together with its constitutional entities using traditional machine learning algorithms: logistic regression and multiple discriminant analysis by Memić [4] , implementation of random forests, logistic regression and SVM by Ying in order to study and analyze bank credit data set [5], and so on.

In this report, gradient boosting decision trees, logistic regression and several other classification algorithms have been used for default risk analysis and prediction from home credit database. As an evaluation metric Area Under Receiver Operating Characteristics Curve (ROC-AUC) have been used. The training set has been used to locally train, evaluate and test the models. In the end, the models have been submitted to Kaggle for final evaluation.

2 Technical Approach:

The algorithms that has been applied here for modeling are discussed briefly in this section.

2.1 Gaussian Naive Bayes:

Bayes theorem states that:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}$$

Here, $P(\omega_i|\mathbf{x})$ is the posterior probability, $p(\mathbf{x}|\omega_i)$ is Gaussian pdf, $P(\omega_i)$ is the prior probability distribution ranging from 0 to 1. Given a feature vector **x**, naive Bayes method assigns it to a class ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$. Here $g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i)$ [6].

2.2 k-Nearest Neighbors:

In kNN, in order to estimate $p(\mathbf{x})$ from n samples, a cell is centered at x and let grow until it contains k_n samples, and k_n is considered to be some function of n. Normally, $k_n = \sqrt{n}$. In order to employ kNN in classification, given c training sets from c classes, the total number of samples is calculated as: $n = \sum_{i=1}^{c} n_I$, Given a point x at which the statistics are to be determined, the hypersphere of volume V is found which only encloses k points from the combined set. If within that volume, k_i of those points belong to class i, then the density for class i is estimated as:

$$P(\omega_i|x) = \frac{k_i}{k}$$

The decision rule indicates to look in a neighborhood of the unknown feature vector for k samples. If within that neighborhood, more samples lie in class i than any other class, we assign the unknown as belonging to class i [6].

2.3 Support Vector Machines:

SVM finds a hyperplane with maximum margin in an n-dimensional feature space that classifies the data points. The goal here is to maximize the margin between the data points and hyperplane using hinge loss:

$$c\big(x, y, f(x)\big) = 1 - y * f(x) \text{ where, } \quad y * f(x) \geq 1$$

If the cost is not non-zero, then the loss value is calculated, and a regularization parameter is added to the cost function which balances the margin maximization and loss.
The cost function of SVM is given as:

$$\frac{1}{n}\sum_{i=1}^{n}\max(0,1-y_i(\mathbf{w}\mathbf{x_i}-b))]+\lambda\|\mathbf{w}\|^2$$

Here, λ is the tradeoff between increasing the margin size and ensuring that $\mathbf{x_i}$ is on the correct side of the margin. The weights $\mathbf{w}$ are updated by taking the gradients of the regularization parameter. In practice, the performance of SVM is controlled by several parameters among which the kernel function directly influences its capability of handling a data set. There are several kernel functions such as linear, polynomial, sigmoid, radial basis, Gaussian and so on. By trial and error, sigmoid kernel seemed to perform comparatively better and it is given by: $k(x,y)=\tanh(\alpha x^T y+c)$ [7].

2.4 Logistic Regression:

Logistic regression is another statistical method for classification. Its goal is to find the best fitting model to describe the relationship between predictors and response variable. Estimation using it selects parameters which maximize the likelihood of sample value observation. Logistic regression maps any real value into another value between 0 and 1 using the sigmoid function:

$$h_\theta(x)=\frac{1}{1+e^{\theta^T x}}$$

Here $h_\theta(x)$is the output between 0 and 1 and z is the input. The cost function of logistic regression is given by:

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{n}\text{cost}\left(h_\theta\left(x^{(i)}\right),y^{(i)}\right)$$

Here $\text{cost}(h_\theta(x),y)=-y\log\left(h_\theta(x)\right)-(1-y)\log\left(1-h_\theta(x)\right)$. In order to avoid overfitting, for this analysis, a regularization term L2-penalty term has been added to the cost function. Now the cost function becomes:

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{n}\left(\text{cost}\left(h_\theta\left(x^{(i)}\right),y^{(i)}\right)+\frac{\lambda}{2}\theta_i^2\right)$$

Here λ is the penalization parameter [8].

2.5 Gradient Boosting Decision Tree:

Decision trees used in classification consist of both leaf nodes and internal nodes. Each internal node represents a query on the features of the training samples. Children of these nodes represent divisions in the training samples determined by their responses to the query. Leaf or terminal nodes compete in the tree and are each assigned a class label corresponding to the majority vote of the training samples falling under the leaf. When constructing a decision tree, the objective is to select queries at each level of the tree that maximize the decrease in impurity between the child and parent nodes as follows:

$$\Delta i(N)=i(N)-P_L i(N_L)-(1-P_L)i(N_R)$$

Here i(N), i(N_L), and i(N_R) give the impurity of the parent, left child, and right child, respectively. The impurity measure may be calculated using entropy, variance, Gini, or misclassification impurity. However, a single decision tree is prone to overfitting and unlikely to generalize well [6].

Gradient boosting decision trees (GBDTs) resolve this overfitting or underfitting problem of simple decision trees. It combines the decisions of multiple decision trees by adding them together through an iterative training process. It minimizes the impurity by recursive splitting of data according to some new criteria. The features which have the most importance are accounted for the most impurity reduction. LightGBM by Microsoft [9] is one of the implementations of GBDT which is very suitable for a large set of data due to its faster execution and less memory usage. it uses leaf-wise growth strategy where it splits the leaf with minimal loss reduction. It can also efficiently treat missing value by allocating them to whichever split side reduces the loss most. LightGBM (LGBM) has various hyperparameters that need to be optimized for better modeling. This hyperparameter optimization process is discussed in the modeling section.

2.6 Model Evaluation:

The performance of each classifier has been evaluated based on Receiver Operating Characteristics curve with AUC (area under ROC curve). ROC curve shows how well a classifier can distinguish between two classes and what is the trade-offs between TPR and FPR. TPR and FPR are given by:

$$\text{True Positive Rate (TPR)} = TP/ (TP+FN)$$

$$\text{False Positive Rate (FPR)} = FP/ (TN+FP)$$

TPR is also known as sensitivity or recall. It explains the probability of positive class detection. On the other hand, FPR tells about the probability of false alarm. Note that, TP is the total number of true positives, TN is the total number of true negatives, FP is the total number of false positives and FN is the total number of false negatives. AUC measures the performance of a ROC curve. The higher the area, the better. A perfect ROC curve will have an AUC of 1 with TPR = 1 and FPR = 0.

3 Experiments and Results:

This section provides meticulous description of the data set, exploratory analysis, different preprocessing steps, model training, validation and testing process.

3.1 Description of Data Sets:

There were 7 different datasets to make use of in order to predict the repayment capability of each applicant, one of which is our primary data set and have been used to join other useful information from the rest of the data sets. Each row in the primary data set "application_{train|test}.csv" represents one loan application which is distinguished by a unique id namely, "SK_ID_CURR". There are 356,255 applications with 121 features in this set which is broken into a train (307511 data points with TARGET column) and a test (48744 data points without TARGET column) files. "bureau.csv" contains previous credits information of these clients from other financial institutions that were reported to the Credit Bureau. Each entry in this file is identified by a unique id, namely "SK_ID_BUREAU". It has 1716428 entries with 17 features. For every loan in the train-test sample, there are as many rows as the number of credits the client had in Credit Bureau

before the application date. The information of monthly balances of the above-mentioned previous credits is stored in "bureau_balance.csv" file where each row accounts for each month of observable history of every previous credit reported to Credit Bureau. In total, this data set has 27299925 row entries with 3 feature entries. The "previous_application.csv" file contains in total 1670214 previous applications of clients' for Home Credit loans. Each row in this file is identified by a unique id "SK_ID_PREV" and has 37 features. Monthly balance snapshots of applicant's previous POS (point of sales) together with cash loans and monthly balance snapshots of previous credit cards are contained in "POS_CASH_balance.csv" (10001358 row entries) and "credit_card_balance.csv" (3840312 row entries) respectively. Each row in these data sets represents each month's history of every previous consumer credit and cash loan in Home Credit with 8 and 23 features respectively. "Installments_payments.csv" has the repayment history for the previously disbursed credits. This data set has one row for every payment that was made together with a row for each missed payment. Each row here is equivalent to one payment of one installment totaling up to 13605401 payment information from various clients. This data set has 8 columns [1].

All of the 6 secondary data sets are related to the primary data set by either a primary (e.g. "SK_ID_CURR") or a secondary (e.g. "SK_ID_BURR") relationship which have later been used to join the data sets. This relationship is presented in the following diagram (figure 1):

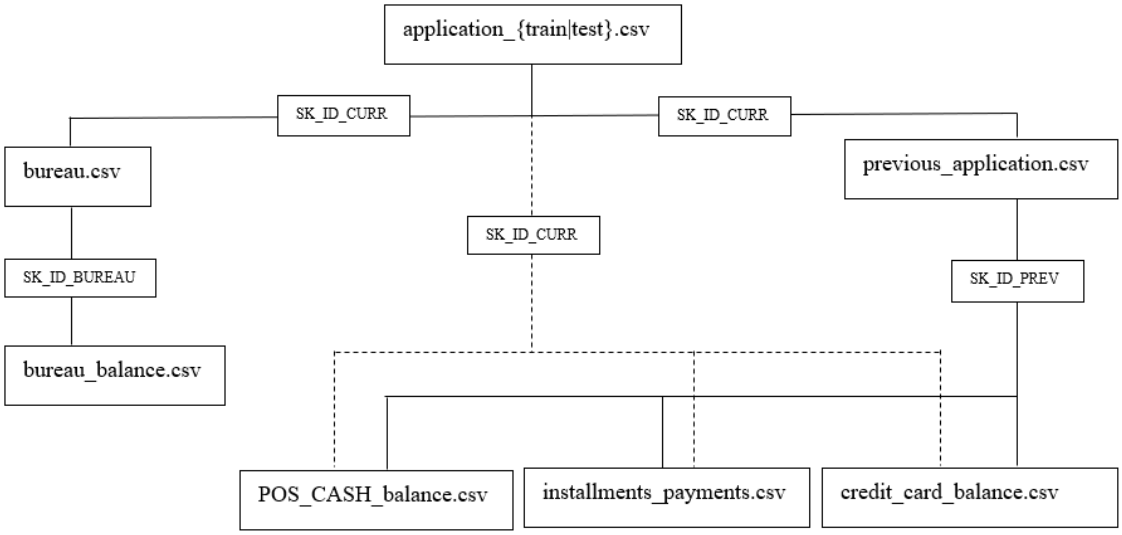


Figure 1: Relationship Among the Data Sets

For convenience, only train data has been used from current application train and test set for local model training, validating and testing. After building fair models, they have been tested on the actual testing data in Kaggle kernels.

3.2 Exploratory Data Analysis:

Before preprocessing and modeling, some exploratory analysis has been done in order to have a better understanding of the data sets. This has also provided a general idea of the effects of each feature on clients'

loan repayment capability. The data exploration is mainly done on the current and previous applications and bureau data based on ideas taken from Kaggle kernels [10].
As the very first step, the existence of any missing values in the data sets have been checked. Figure 2 represents the top 5 counts and percentage of missing values from each data set. RATE_INTEREST_PRIVILEGED and RATE_INTEREST_PRIMARY in the previous applications' data set have almost 100% missing values.

	Total	Percent
COMMONAREA_MEDI	214865	69.872297
COMMONAREA_AVG	214865	69.872297
COMMONAREA_MODE	214865	69.872297
NONLIVINGAPARTMENTS_MODE	213514	69.432963
NONLIVINGAPARTMENTS_MEDI	213514	69.432963

(a) application train

	Total	Percent
AMT_ANNUITY	1226791	71.473490
AMT_CREDIT_MAX_OVERDUE	1124488	65.513264
DAYS_ENDDATE_FACT	633653	36.916958
AMT_CREDIT_SUM_LIMIT	591780	34.477415
AMT_CREDIT_SUM_DEBT	257669	15.011932

(b) bureau

	Total	Percent
STATUS	0	0.0
MONTHS_BALANCE	0	0.0
SK_ID_BUREAU	0	0.0

(c) bureau balance

	Total	Percent
RATE_INTEREST_PRIVILEGED	1664263	99.643698
RATE_INTEREST_PRIMARY	1664263	99.643698
RATE_DOWN_PAYMENT	895844	53.636480
AMT_DOWN_PAYMENT	895844	53.636480
NAME_TYPE_SUITE	820405	49.119754

(d) previous application

	Total	Percent
AMT_PAYMENT_CURRENT	767988	19.998063
AMT_DRAWINGS_OTHER_CURRENT	749816	19.524872
CNT_DRAWINGS_POS_CURRENT	749816	19.524872
CNT_DRAWINGS_OTHER_CURRENT	749816	19.524872
CNT_DRAWINGS_ATM_CURRENT	749816	19.524872

(e) credit card balance

	Total	Percent
AMT_PAYMENT	2905	0.021352
DAYS_ENTRY_PAYMENT	2905	0.021352
AMT_INSTALMENT	0	0.000000
DAYS_INSTALMENT	0	0.000000
NUM_INSTALMENT_NUMBER	0	0.000000

(f) installments payments

	Total	Percent
CNT_INSTALMENT_FUTURE	26087	0.260835
CNT_INSTALMENT	26071	0.260675
SK_DPD_DEF	0	0.000000
SK_DPD	0	0.000000
NAME_CONTRACT_STATUS	0	0.000000

(g) POS cash balance

Figure 2: Missing Data in each Data Set

i) Application Train Data: Defaulter detection problems tend to be imbalanced due to the fact that being a defaulter is a very rare incident. Since most of the machine learning algorithms assume the fed data to be equally distributed over different classes, their classification tends to be more biased towards the majority classes causing the minority classes to be looked over. Hence detecting and fixing class imbalance is one of the important steps in data analysis process. By looking at the following pie chart (figure 3), it can be said that this training set is fairly imbalanced which needs to be handled before they are fed into a model.

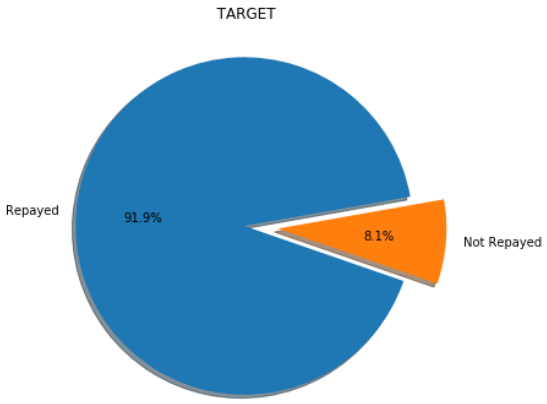


Figure 3: Class Imbalance Check

Since high negative or high positive correlation is considered to be one of the indicators of good predictors, as the next step Pearson correlation coefficients between TARGET and all the other variables in the training data set have been calculated and partially visualized (figure 4).

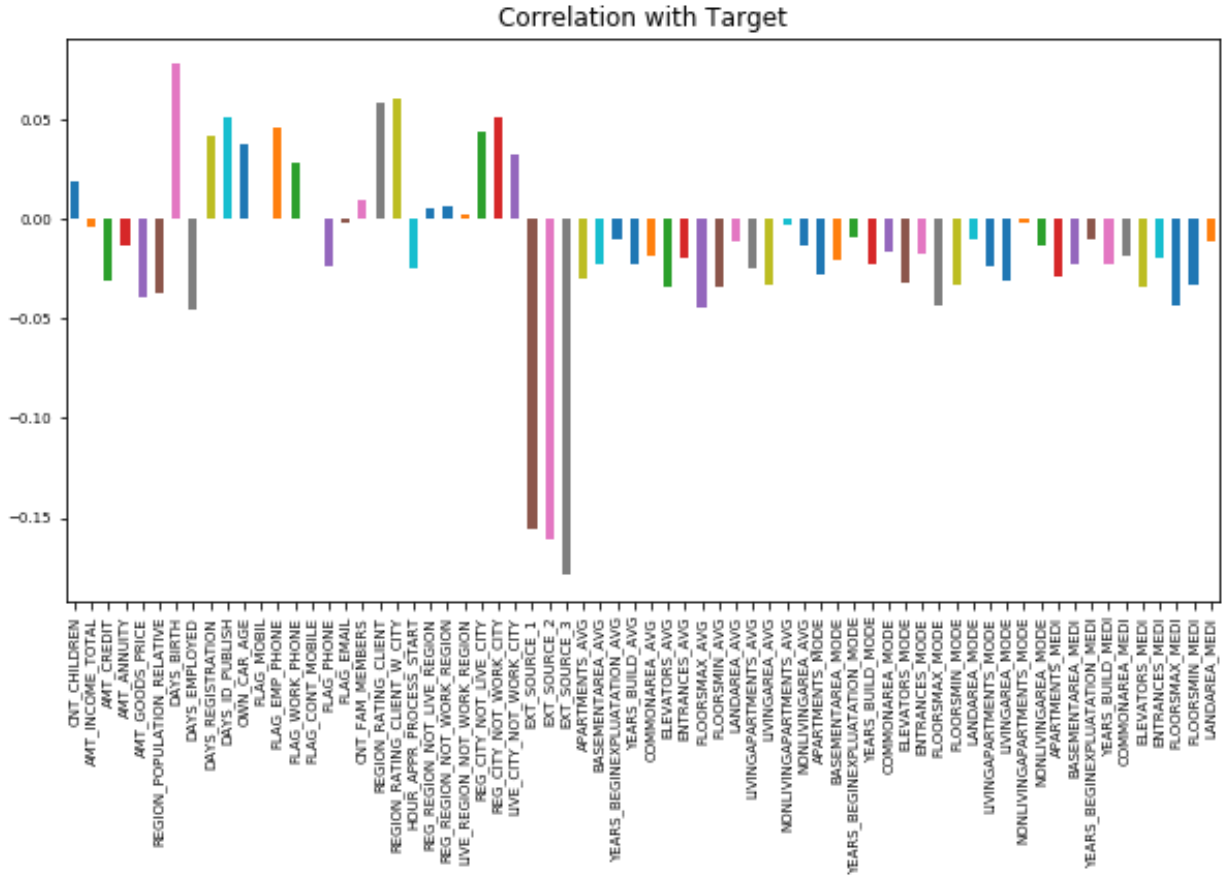
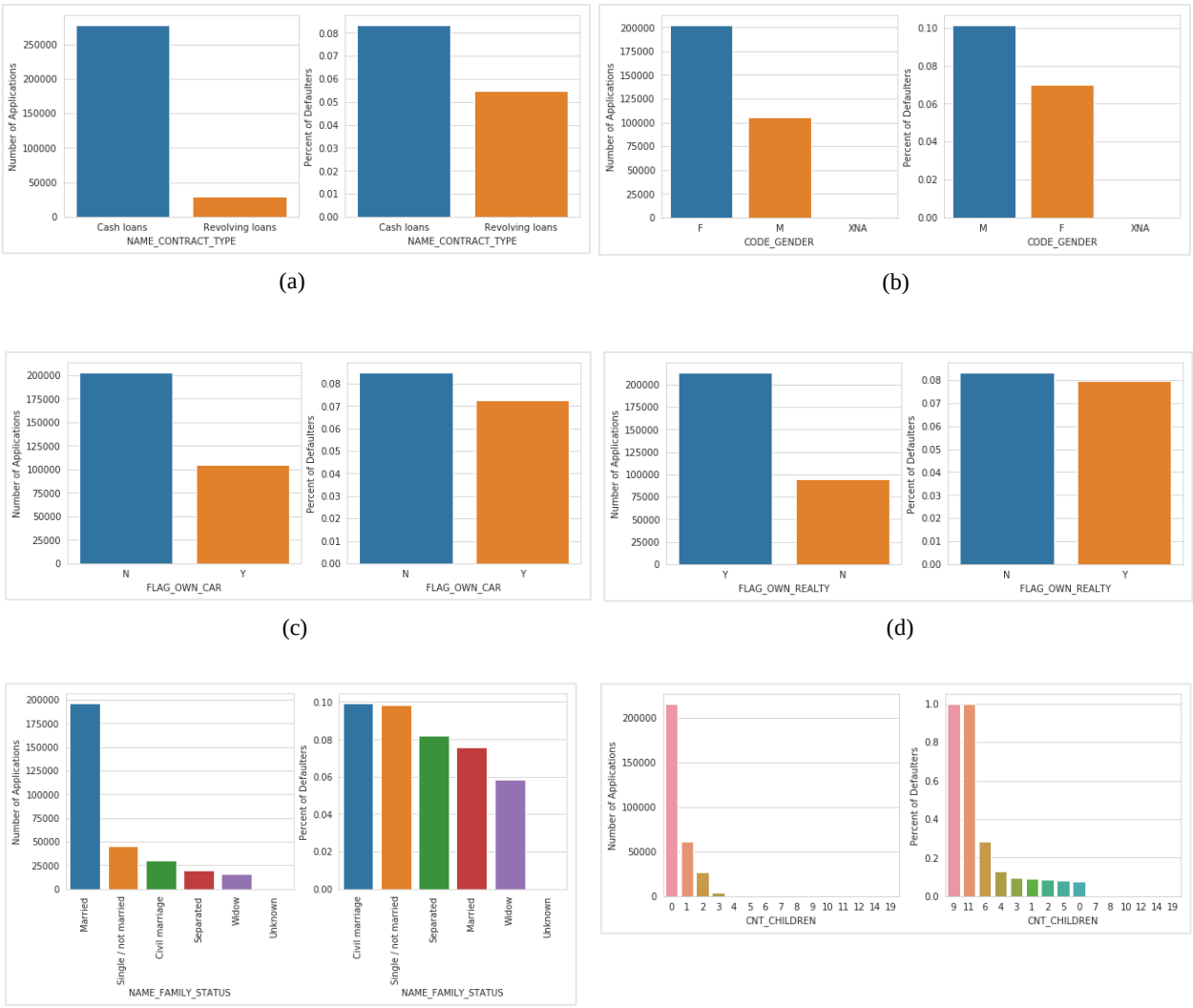


Figure 4: Pearson Correlation between Predictors and Response

It can be seen from the figure that EXT_SOURCE_3, EXT_SOURCE_2 and, EXT_SOURCE_1 are the top three correlated variables. On a hunch, it can be assumed that they are going to be 3 primary helpers on identifying defaulters.

In the next step, any sort of visible relationship between given features and clients' loan repaying capability have been analyzed. By keeping in mind, the scope of this report, only some of the important relationships are included below: In each bar graph left and right charts show frequency and percentage respectively.



(a) (b)

(c) (d)

(e) (f)

Figure 5: Exploratory Data Analysis (Application Train)

As it can be seen here (figure 5a), although clients are less likely to apply for revolving loans (~ 10%), a large part of them are not repaid compared to that. When looking at the gender (figure 5b), interestingly number of female clients is almost twice of male ones. But compared to a female (~7%), a male has a much higher probability (~ 10%) of being a defaulter. It has also been noticed that (figure 5c, 5d) whereas owning a car or not can slightly affect the chance of being a defaulter, owning a realty or not doesn't really have an effect on it. In terms of marital status, two categories in marriage have been noticed, married and civil

marriage. Whereas married people tend to take more loans than the others, civil marriage category tend to have the highest defaulting rate (~ 10%), followed by single category (figure 5e).

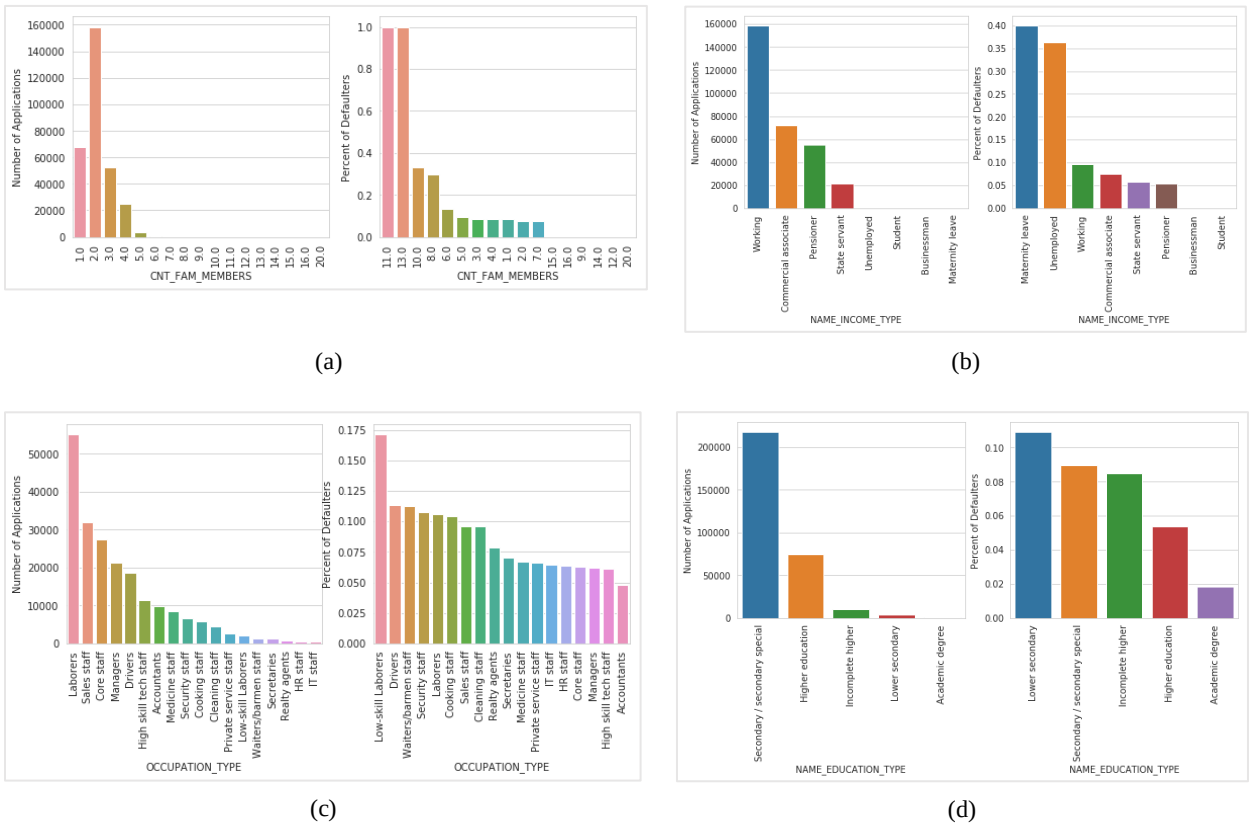


Figure 6: Exploratory Data Analysis (Application Train)

These two categories can be overlapping. But, since there is no such information regarding that, this has been disregarded for this analysis. Clients with no children seem to dominate in the CNT_CHILDREN category. Although when it comes the time to repay the loan, clients with 9 or 11 children have 100% chance of non-repayment followed by the clients with 4 and 6 children (~ 25%) (figure 5f). With a fair fraction of data this can be a good defaulter predictor. Clients with 2 membered families are inclined to take more loans than others, but their repayment rate is higher whereas clients with family members 11 and 13 have a 100% default rate (figure 6a). By looking at the income type feature, it can be noticed that working clients apply for more loans, but clients who are on maternity leave have the highest default rate (~ 40%) followed by unemployed clients (~ 37%) (figure 6b). In occupation chart the noticeable factor was, although low-skill-laborers apply for loans less frequently, their non-repayment rate is the highest in the chart (~ 17.5%) (figure 6c). People with low education tend to apply for more loans than people with a higher degree and the non-payment rate is seen to have an inverse relationship with increasing academic degree (figure 6d). Clients with organization type business entity have the highest applications in the data set (figure 7), whereas the highest percentage of non-repaid loans belongs to Transport: type 3 (~ 16%). Over 250K housing information entries have House/apartment. Despite having a lower proportion in terms of entries in the data set, clients who have rented apartments and who live with parents is dominating the non-repayment rate chart (~ 12%).

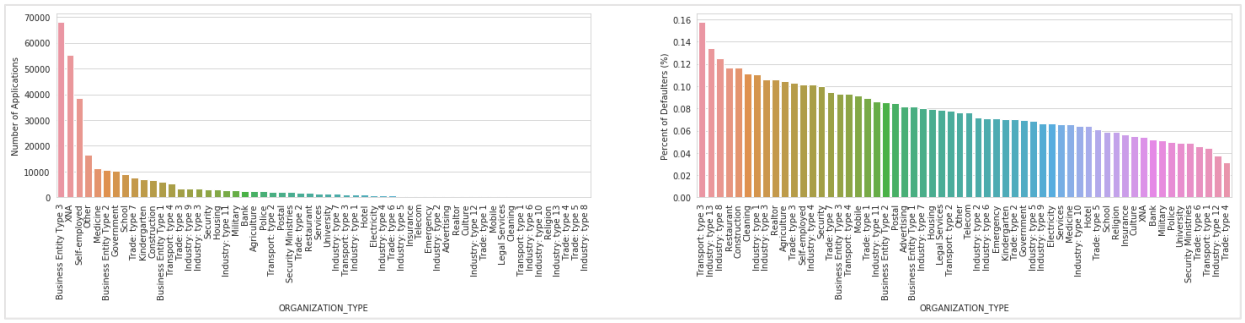


Figure 7: Exploratory Data Analysis (Application Train)

Other interesting features that caught attention were REG_REGION_NOT_LIVE_REGION, REG_REGION_NOT_WORK_REGION, REG_CITY_NOT_ LIVE_CITY, and REG_CITY_NOT_WORK_CITY. Interestingly, only a few of the clients are registered in either not live or not work region or city. But compared to others, they have higher non-returning rate (figure 8a-8d).

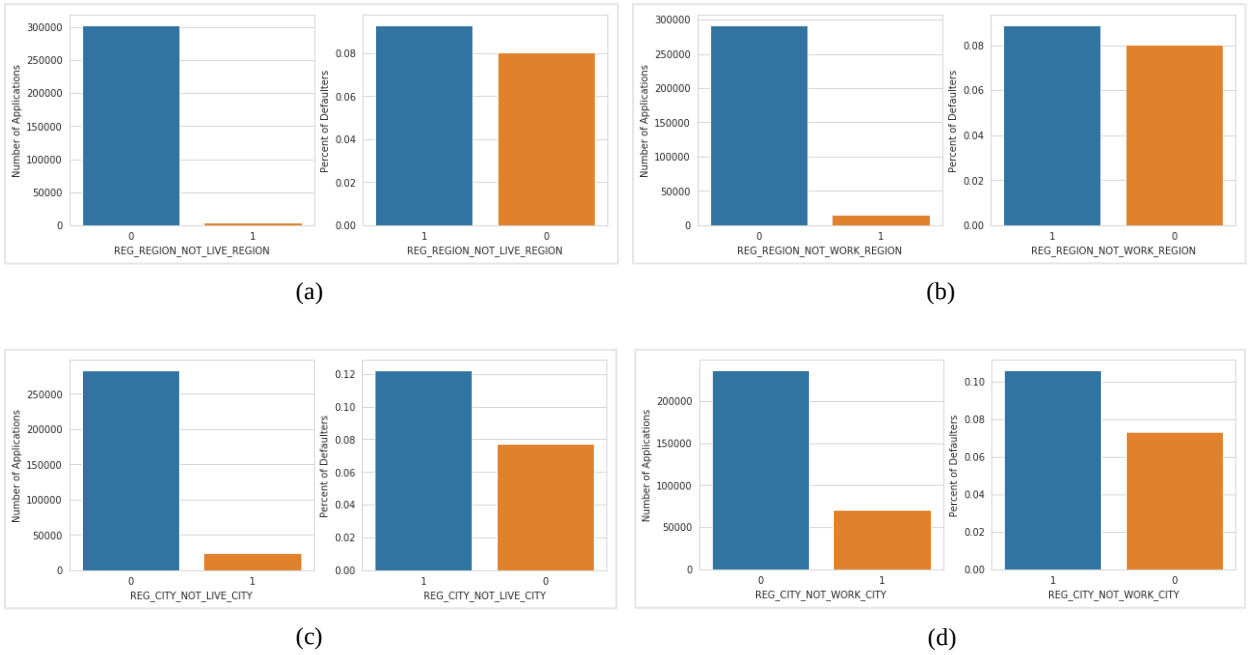


Figure 8: Exploratory Data Analysis (Application Train)

In figure 9, AMT_CREDIT, AMT_ANNUITY, AMT_GOODS_PRICE, DAYS_BIRTH, DAYS_EMPLOYED, DAYS_REGISTRATION and, DAYS_ID _PUBLISH have been plotted. It can be seen that most credit amount of the loans lies around $200K. Most of the loan annuity fall below $30K. The price of the goods for which the loan is given (consumer loans) has several peaks with the highest at around $500K. From the age distribution of the clients, it can be said that age ranges from around 20 to 68 years where the highest peak is around the younger clients. When looking at the employment days entry, a peak at the end of the distribution came into attention which means that there is a large set of clients who have been employed for more than 1000 years, which is unrealistic. This anomaly needs to be fixed in the data preprocessing step. The peaks from the distributions of number of days before application the clients

changed their registration or ID reveal that most registrations were changed before ~3 years and most IDs were changed before ~10 years of the application. When compared the distribution of the defaulters with non-defaulters, it has been noticed that younger clients have a lower chance of repaying loans than older clients.

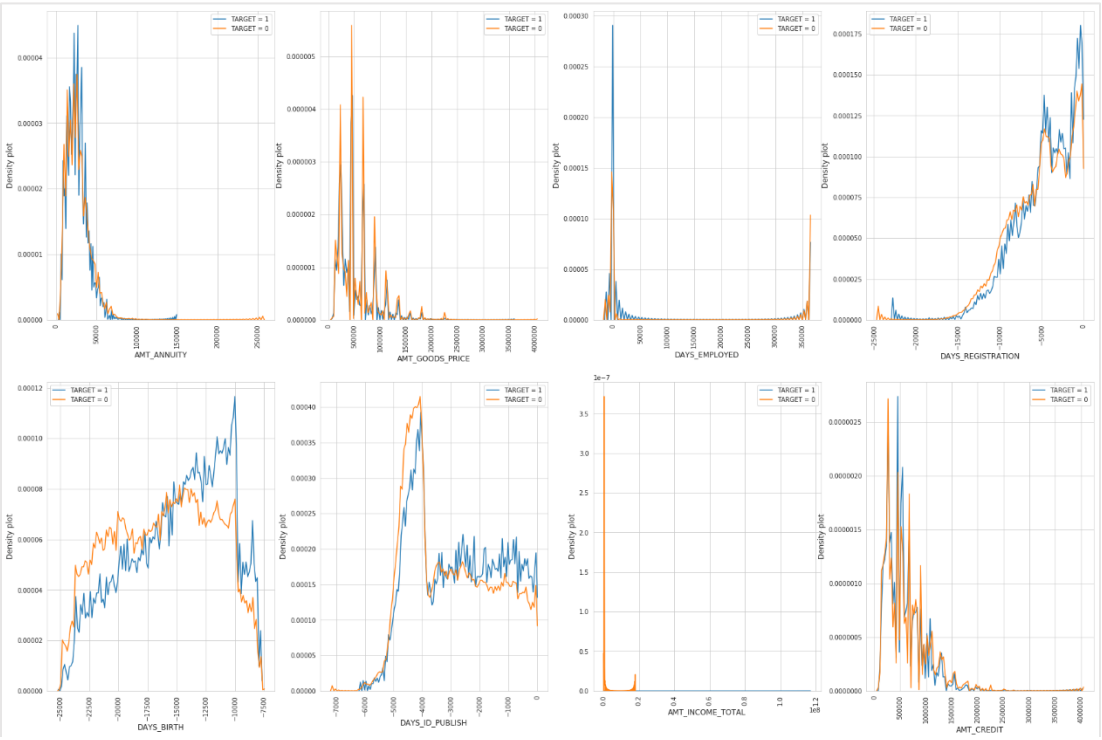
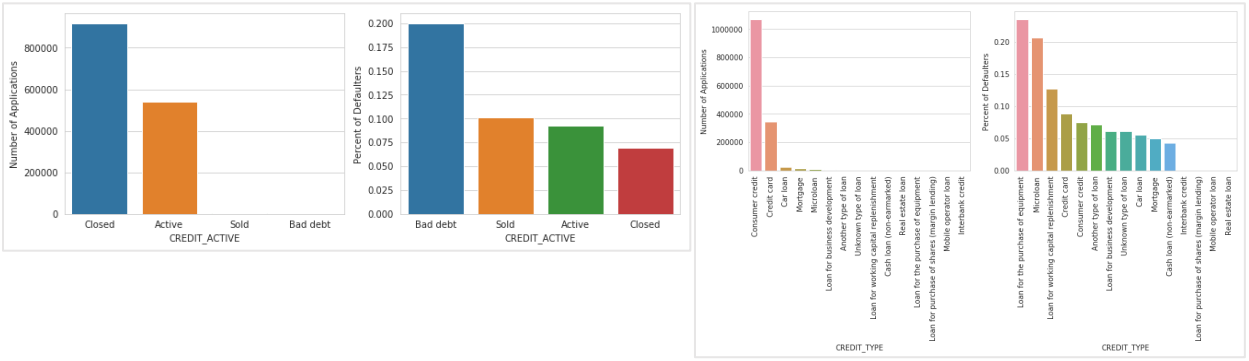


Figure 9: Exploratory Data Analysis (Application Train)

ii) Bureau Data: In the credit status chart (figure 10a) it has been noticed that although most of the credits registered at the credit bureau have a closed status (~ 900K), clients with bad debt status have the highest defaulting rate (~ 20%). This feature has the potential to be a good predictor for the target value in the training set. From the type of reported credit, it can be seen that (figure 10b) consumer credit has the highest entry in the data set and clients who took loans for the purchase of equipment and microloan have highest default rate.



(a) (b)

Figure 10: Exploratory Data Analysis (Bureau)

From the distributions (figure 11) it can be observed that the credit duration has a peak around ~1 year, most of the credits reported to bureau have nearly 0 days overdue, most of the current credit amounts are concentrated around $20K and maximum entries of reported current credit limit of credit card have values around 0. When these distributions have been compared for defaulters and non-defaulters, the four features seem to have similar distributions. So, it can be assumed that they are not going to be good predictors for the target variable.

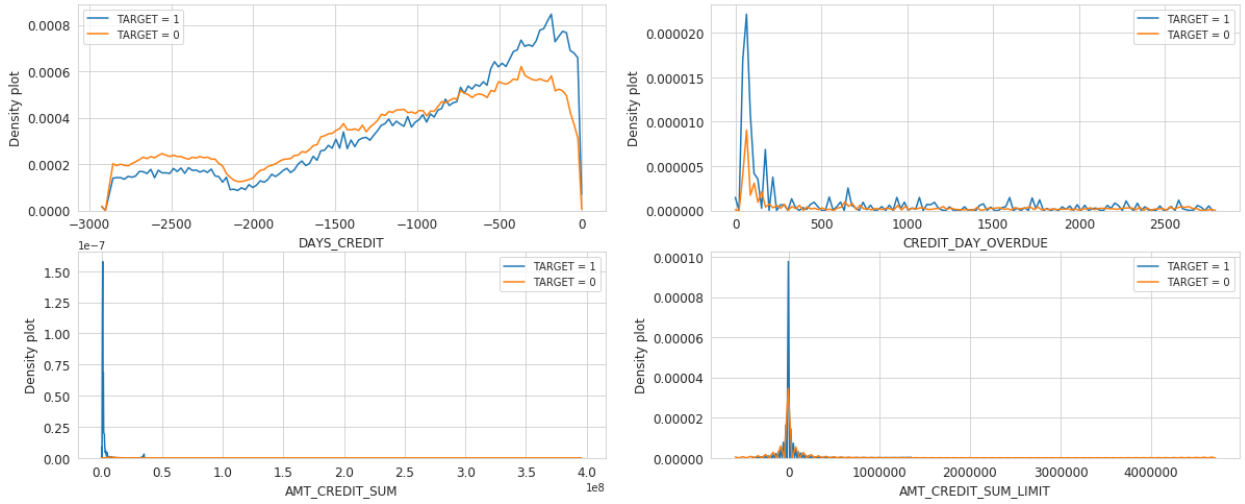


Figure 11: Exploratory Data Analysis (Bureau)

iii) Previous Application Data: In the previous application (figure 12), maximum clients applied for a contract type of cash loans and consumer loans. Very few clients who applied with XNA entry for this feature in their previous applications are most likely to be the defaulters. Also, clients who refused to name their purpose for the loan have the highest non-repayment rate (~ 23%) compared to others.

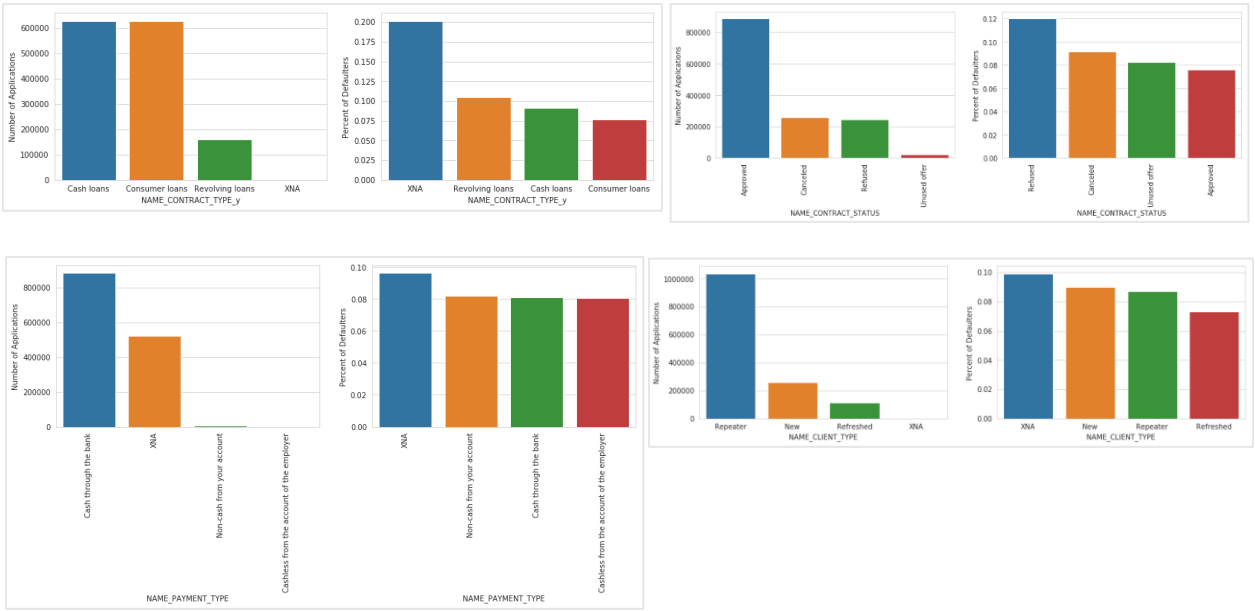


Figure 12: Exploratory Data Analysis (Previous Application)

Interestingly enough, clients who were refused to get loans in their previous applications have a higher default rate than others (~ 12%). Mostly the loans were paid by cash through the bank, but those who belonged to the category non-cash from the account (which is rare) have a higher default rate than others. New clients tend to have a higher non-payment rate than repeated or refreshed ones.

3.3 Data Pre-processing:

This section describes different data preprocessing steps which include fixing data anomalies, handling missing values, encoding categorical variables, feature engineering, standardizing, handling data imbalance, feature selection and, feature extraction.

3.3.1 Fixing Data Anomalies:

In the exploratory data analysis, an anomaly appeared in the days of employment column in both application train and, previous application data sets. When looked more closely, it has been noticed that some of the entries have 365243 days (more than 1000 years) which is not possible. Hence this value has been replaced with null values (NAN).

3.3.2 Primary Handling of Missing Value:

From the missing value percentage of the current application data set and their correlations with the TARGET variable, a threshold of 65% is chosen. That means it is not likely to affect the prediction results if the feature columns with more than 65% missing values are dropped from this set. With this threshold 17 out of 121 features have been dropped and they are mostly related to the apartments' information of the clients. This data set still has 50 columns with missing values to deal with. In bureau data set AMT_ANNUITY has more than 71% missing data which is why it seemed reasonable to drop this column from this set. Bureau data is now left with 6 columns with missing data. As it has been mentioned earlier, two features in the previous application data set has almost 100% missing values. Hence these two columns have been dropped from this data set which left 14 columns with missing values. Among the rest of the data sets, credit card balance data has 9 columns, installments payments data and point of sale cash balance data have 2 columns each, with missing values. However, these columns with missing values have been left untouched until aggregation of the training data using rest data sets has been performed. It is necessary to mention that if the features with missing values contain categorical or both numerical and categorical data, then they have been taken care of in the following encoding part, where all null values (NANs) have been considered to be a separate category.

3.3.3 Categorical Variable Encoding:

There are two types of categorical features in the data sets, features with 2 variables (e.g. CODE_GENDER) and with more than 2 variables (e.g. NAME_FAMILY_STATUS). For features with just 2 variables binary encoding has been performed. For features with more than 2 variables, one hot encoding has been performed where different columns have been created with dummy variables. There are also some features with mixed variable types (both numerical and categorical). For them, the numeric variables have been treated like the categorical ones and one hot encoding has been performed. In all the three cases above, if any null values (NANs) came into sight in any of the columns, they have been considered as a separate category during encoding.

3.3.4 Feature Engineering:

A little bit of domain knowledge has been applied to create some new features from the existing ones. Many of the features have been engineered here with the help of kernels from the Kaggle competition [10]. A list of newly created variables is given below with their relationship with the original variables.

Table 1: Newly created Variables:

Data Set	Derived features	Relationship with original ones
Application Train	DAYS_EMPLOYED_PERC	DAYS_EMPLOYED / DAYS_BIRTH
	INCOME_CREDIT_PERC	AMT_INCOME_TOTAL / AMT_CREDIT
	INCOME_PER_PERSON	AMT_INCOME_TOTAL / CNT_FAM_MEMBERS
	NEW_ANNUITY_TO_ INCOME_RATIO	AMT_ANNUITY / (1 + AMT_INCOME_TOTAL)
	PAYMENT_RATE	AMT_ANNUITY / AMT_CREDIT
	NEW_CREDIT_TO_ INCOME_RATIO	AMT_CREDIT / AMT_INCOME_TOTAL
Previous App.	APP_CREDIT_PERC	AMT_APPLICATION / AMT_CREDIT
Installments Pay.	PAYMENT_PERC	AMT_PAYMENT / AMT_INSTALMENT
	PAYMENT_DIFF	AMT_INSTALMENT - AMT_PAYMENT
	DPD (Days Past Due)	DAYS_ENTRY_PAYMENT- DAYS_INSTALMENT
	DBD (Days Before Due)	DAYS_INSTALMENT - AYS_ENTRY_PAYMENT

Once these derived features have been added to the original data sets, the primary application train data has been aggregated using 6 secondary datasets. The bureau credit history has been divided into active and closed credits, and the previous applications have been divided into approved and refused applications. Since each application ID has multiple entries in the secondary data sets, it has been decided to take the statistical summary of each secondary data set as an aggregation criterion. In order to avoid redundancy in the description, a table is added below with some of the variable summaries from each data set with a goal to demonstrate the process.

Table 2: Brief list of variable summaries:

Data Set	ID Based on	Summary	Variable
Bureau balance	SK_ID_BUREAU	min, max, mean, var, sum	MONTHS_BALANCE
Bureau	SK_ID_CURR	min, max, mean, var	DAYS_CREDIT
			AMT_CREDIT_MAX_OVERDUE
		min, max, mean, var, sum	AMT_CREDIT_SUM
			AMT_CREDIT_SUM_DEBT
Previous applications	SK_ID_CURR	min, max, mean, variance	AMT_ANNUITY
			AMT_CREDIT
			AMT_DOWN_PAYMENT
		min, max, mean	DAYS_DECISION
		min, max, mean, var, sum	CNT_PAYMENT
POS cash balance	SK_ID_CURR	min, max, mean	SK_DPD
			SK_DPD_DEF
Installments Pay.	SK_ID_CURR	nunique	NUM_INSTALMENT_VERSION
		min, max, mean, var	PAYMENT_PERC
		min, max, mean, sum	AMT_PAYMENT
Credit card balance	SK_ID_CURR	min, max, mean, var, sum	AMT_BALANCE

For encoded columns or for the ones which only have 0 and 1 entries, column sum (how many times that entry took place) and mean (how many times on average that appeared in the data set) have been taken. Also, the total number of POS cash accounts, installments accounts and credit card lines per current application have been calculated during aggregation.

Once this whole process has been completed, the application train data ended up with 307507 rows and 1069 features. For the next step, this data set has been divided into 2 parts as train and test sets. 90% of the data (train) is kept for model training and validation and 10% has been kept aside for testing (test).

3.3.5 Standardization:

In the next step, the data set has been normalized using sklearn's StandardScaler by removing mean and scaling to unit variance. The standard score of any sample is calculated as:

$$\mathbf{z} = \frac{\mathbf{x} - \boldsymbol{\mu}}{s}$$

Here, $\boldsymbol{\mu}$ is the mean of the training samples and s is the standard deviation of the training samples. In order to avoid data leakage to testing samples, testing data is standardized using the parameters calculated from the training set.

3.3.6 Missing Value Imputation:

Missing Value Imputation is one of the trickiest parts in any data analysis process. There are several ways to perform imputation. Some of the most common ones are, mean, median and mode imputation. They are easy and simple to use but oftentimes they can lead to biases in predictions. To resolve this issue, researchers have come up with more advanced techniques such as multiple imputation where the same missing values are imputed several times by using an appropriate model which incorporates random variations until the standard error is minimized, multivariate imputation by chained equations (MICE) where a series of regression models are run with each variable with missing values is conditionally modeled upon other variables [11] and k-nearest neighbors. Since the advanced ones are more efficient for data imputation, as a first attempt MICE and kNN have been applied one by one using the supported packages by python named fancyImpute. Unfortunately, due to memory error (probably because of the size of the data) and due to time constraints, the traditional approach (mean imputation) seemed to be the right one to go with for now. Before performing mean imputation on the columns with missing values, their distributions have been visualized to make sure that it would not produce any significant biases in the data distribution. Once all the columns have been visualized, the missing values are imputed with mean using scikit-learn's SimpleImputer. To avoid data leakage, the parameters calculated from the train set have been used to impute the test set.

3.3.7 Handle Data Imbalance:

Data imbalance is one of the major problems to deal with during the data analysis process. Since the traditional machine learning models, such as Gaussian NB, kNN, SVM and LR which will be used during modeling don't have any built-in parameter to handle unbalanced data set, the data set is resampled before modeling. In order to build a fair model, this has been performed only on the train set.

Keeping in mind the size of the training set 2 approaches were applied for resampling: a) Synthetic Minority Over-sampling technique (SMOTE) which randomly picked a point from the minority class, computed the k-nearest neighbors and added some synthetic points between the chosen and the neighboring points and then b) randomly undersampling which randomly chooses observations from majority class to eliminate until the data set gets balanced. Keeping in mind the loss of information during random undersampling, another method has been applied which performs under-sampling by removing similar samples from clusters based on centroids generating clustering methods. However, this couldn't be successfully executed because of the sample size. Figure 13 represents the imbalanced, oversampled and undersampled data (respectively) using 2-component PCA.

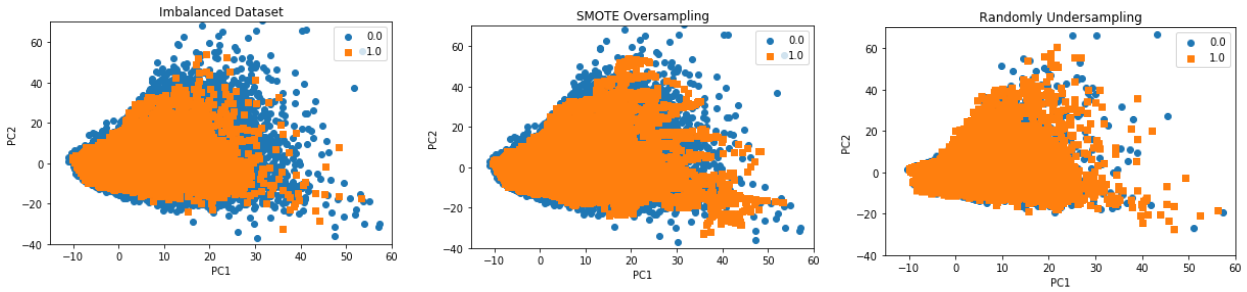


Figure 13: Handling Data Imbalance

3.3.8 Feature Selection:

Feature selection is another crucial component of data analysis workflow, especially when data with very high dimensions has to be dealt with (like this one). When presented with all the data at once, traditional models were having a hard time to navigate through all the dimensions which is why for this analysis and for traditional models the following two feature selection techniques have been used:

i) Pearson correlation coefficient: The absolute value of the correlation between features and target have been calculated and compared and the highly correlated features are selected one by one until the optimal subset of features has been achieved.

ii) Recursive feature elimination: This method has recursively fit a machine learning model (logistic regression in this case) and kept removing weakest features based on feature ranks derived from model coefficients or feature importance until it reached the desired number of features. The optimal number of features are calculated based on a cross validation technique implemented with RFE embedded with an elimination process with a predefined number of weakest features to eliminate at each step.

3.3.9 Feature Extraction:

Dimensionality reduction is one of the very known feature extraction techniques which converts a set of data with vast dimensions into data with lesser dimensions making sure that it contains similar information to the original data set. It also aims to reduce the complexity of data. In this analysis, two techniques have

been used to extract features which are: Fisher's Linear Discriminant (FLD) and Principal Component Analysis (PCA).

i) Fisher's Linear Discriminant: FLD used information from the dataset, hence labeled as 'supervised'. It calculates the directions, namely linear discriminants that represent the maximized separated axes among classes. FLD uses the calculation of within class and between class scatter matrices before Eigen decomposition [6]. In both algorithms, the reduced data set is derived as follows:

$$\mathbf{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

Here, **x** is the original data and **w** is the eigenvector. Using FLD, the data has been reduced to n = (no of classes -1) = 1 dimension.

ii) Principal Component Analysis: PCA is an 'unsupervised' technique which maximizes the variability in the dataset. The main idea that is used by PCA is computing the d dimensional mean vector ignoring the class labels and then computing the covariance matrix of the dataset. Once the covariance matrix has been determined, the next task is to calculate eigenvalues and eigenvectors for the matrix. After that, using an error rate some of the eigenvectors from the space are discarded [6]. The information from train data is used to reduce the dimensions of the test data. For PCA dimensionality reduction, in this project, the data set has been reduced to a range of different dimensions in order to find the best fitted one.

3.4 Modeling:

As a first step in modeling a variety of machine learning models have been implemented including Gaussian NB, kNN, SVM, LR and, LGBM. For each of them, a 5-fold cross validation has been performed for model training and validation and then they have been tested on the previously separated test data. Since the traditional Gaussian NB, SVM, kNN and, LR can't handle class imbalance by themselves, resampled data has been used for them. For modeling tree based LGBM, the unsampled data has been used with the hyperparameter scale_pos_weight = 11.

3.4.1 Baseline Model:

Building predictive models is an iterative process which usually helps from starting with some baseline models to compare before getting into any complex model tuning. Two baseline models have been created for each algorithm, one with only the current application data (259 features) and another one with the aggregated data but with dropped missing value columns (189 features). Following table represents how well each model performed on each of these two sets.

Table 3: Mean cross-validation AUC of 5 models on two baseline data sets:

Algorithms	Only Application Data	Aggregated Data with Dropped Missing Columns
Gaussian NB	0.55	0.52
SVC	0.53	0.53
kNN	0.66	0.62
LR	0.75	0.67
LGBM	0.76	0.68

By looking at the table, it can be concluded that logistic regression (mean cross-validation AUC = 0.75) and lightGBM (mean cross-validation AUC = 0.76) are the two best performing models. LightGBM is not only the best but also the fastest one to execute as it takes lower memory to run even with a large amount of data. In addition, it uses histogram-based algorithm which fastens the training process. It focuses on the accuracy of results by producing much more complex trees by following leaf-wise split approach. The performance of logistic regression supports the idea that it is one of the best performers for dichotomous response variables. Another reason can be its predictability of probabilities when compared to other classifiers such as Naive Bayes and its capability of demonstrating significant relationships between dependent and independent variables by indicating the strength of influence of multiple predictors on a response variable. It also performs well with large sample size. However, when modeling with logistic regression, one thing should always be kept in consideration which is collinearity between independent variables. How this has been handled in this analysis will be discussed later.

There can be a couple of reasons why the others performed poorly on this data set. Some of them are mentioned here. Naive Bayes assumes absolute feature independence which can often lead to poor performance. For SVM, the optimal kernel function has been chosen based on trial and error. However, it still didn't perform as expected. Its performance could be improved by tuning with more parameters along with kernel functions. Nevertheless, it took so long to run even with the undersampled data that it has not been considered for further improvement for now. kNN cannot handle high dimensions since it easily falls victim to curse of dimensionality. This means that Euclidean distance can be of no use in high dimensions because almost all the vectors will be almost equidistant from the query vector. This problem can be resolved by using different dimensionality reduction techniques such as PCA or FLD. However, kNN also takes a long time to execute over the large data set.

In the next part, these two models have been improved further upon their baseline AUC. One thing that has been observed is that dropping all the columns with missing values can lead to worst performance.

3.4.2 Improvement upon lightGBM:

The AUC achieved from lightGBM was the best and the fastest one in baseline models. A little bit of tweaking in the model parameters can change the accuracy a lot. Also, since the lightGBM can handle missing values and unbalanced data, the unimputed and unsampled aggregated train data with all the features has been applied during hyperparameter optimization and modeling.

i) Hyperparameter Optimization: Tuning hyperparameters is one of the rigorous and important tasks in the model building process. The learning process of the same model on the same data set is highly influenced by the predefined hyperparameters. In the tuning process, the objective is to minimize a predefined loss function on given data by implementing a cross-validation technique on a set of parameters.

There are several optimization techniques such as manual optimization where parameters are selected based on intuition or experience, grid search where a model is trained on a grid of parameters for each possible combination, random search where grid search is performed with randomized selection and automated hyperparameter tuning where algorithms such as Bayesian optimization or gradient descent are used to perform a more directed search for best hyperparameter. For this dataset randomized grid search with cross-validation has been used for optimizing parameters.

ii) Randomized Grid Search with Cross-Validation: Randomized grid search (from sklearn) is one of the most efficient ways to optimize hyperparameters. It takes lesser time and effort compared to other optimization techniques. It implements a fit and score method with a cross-validation optimization technique over a fixed number of different parameter settings sampled from specified distributions. This number can be controlled by the parameter n_iter. As an estimator LGBMClassifier has been used with 100 settings of parameters and with scoring criteria set as ROC-AUC.

iii) Hyperparameter Grid Selection for Optimization: Keeping in mind the time constraints, only the following parameters have been optimized: fraction parameter for random selection of part of features (colsample_ bytree), minimum number of data per leaf (min_child_samples), minimum sum hessian per leaf (min_child_weight), maximum number of leaves in one tree (num_leaves), L1 and L2 regularization parameters to control overfitting (reg_alpha, reg_lambda), bagging fraction (subsample) and weight of labels with positive class for imbalanced data (scale_pos_weight).

Following table represents their selected grid for optimization. This grid has been created by taking ideas from Kaggle kernels [10].

Table 4: Grid Selection for Hyperparameter Optimization

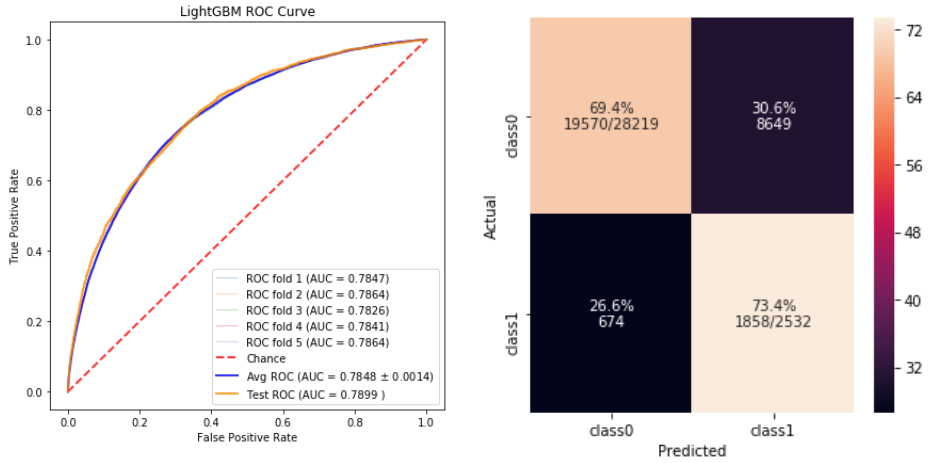
Parameter	Grid
colsample_bytree	continuous random variables uniformly distributed on [0.3, 0.5]
min_child_samples	List of random integers from 200 to 500
min_child_weight	[1e-5, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4
num_leaves	List of random integers from 4 to 40
reg_alpha	[1e-1, 1, 2, 5, 7, 10, 50, 100]
reg_lambda	[1e-1, 1, 5, 10, 20, 50, 100]
subsample	continuous random variables uniformly distributed on [0.3, 0.5]
scale_pos_weight	[1, 2, 6, 11, 12]

iv) Cross-Validation: Once the grid of each parameter has been specified, randomsearchcv has performed a 3-fold cross validation for 100 different parameter settings. In this process, each combination has been evaluated by scoring them on a validation set using Receiver Operating Characteristics Area Under the Curve (ROC AUC).

v) Early Stopping: In order to make this calculation less expensive, the number of sequentially trained decision trees (n_estimators) has been set as 10000. Also, an early stopping criterion has been set which basically decided after how many iterations the model needed to be stopped being trained if it's validation score doesn't improve.

After completion, the parameter values have been yielded as follows: colsample_bytree = 0.48263575356020577, min_child_samples = 311, min_child_weight = 1, num_leaves = 7, reg_alpha = 7, reg_lambda = 0.1, subsample = 0.3542761367404292 and scale_pos_weight = 2.

vi) Tuned LGBM Model: Figure 14 represents the performance of lightGBM which shows the mean cross-validation AUC (0.78) and the testing AUC (0.79) with the confusion matrix. The hyperparameter tuning and inclusion of more information or features in the data set have been able to increase the model AUC score.



(a) ROC Curve (b) Confusion Matrix for Test Data

Figure 14: Performance of LightGBM with Aggregated Data

This model has been able to correctly classify ~70% of non-defaulters and ~74% of defaulters. After that, the mean reduction in impurity across all trees in the model (namely feature importance) has been calculated. From the feature importance chart (figure 15), it can be concluded that EXT_SOURCE_1, EXT_SOU-RCE_3 and, DAYS_BIRTH are 3 of the 5 top predictors which comply with the assumptions made earlier during exploratory data analysis.

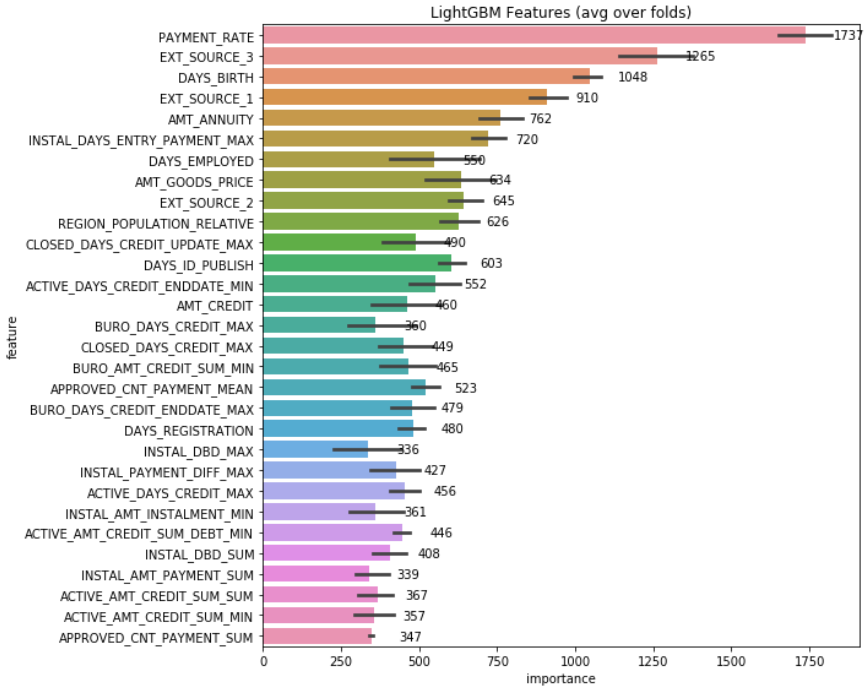


Figure 15: Feature Importance in LightGBM with All Features

3.4.3 Improvement upon Logistic Regression:

Although logistic regression was one of the best performers, it took longer to run than lightGBM. In order to improve the performance of logistic regression, feature selection and feature extraction techniques have been applied on the aggregated mean imputed train data. Another thing that has been considered before modeling with logistic regression is multicollinearity.

i) Detection and Removal of Multicollinearity: Multicollinearity is a phenomenon in which two or more predictors in a regression model are so highly correlated with each other that the values from one can predict the values of another. It causes model instability by variating regression coefficients. It also undermines the statistical significance of a predictor in the model [13]. In addition, with an increasing number of features regression models become more and more vulnerable to multicollinearity. In order to mitigate this problem, the correlation between every possible pair of features have been calculated and one of the features between every two highly correlated ones have been removed with a correlation threshold of 0.8. During removal of feature from each pair, their correlations with the response variable have been kept in consideration and the one with the lowest has been removed. After that, the data set has been left with 739 features.

ii) Pearson Correlation: In order to select the best set, a series of models have been trained in the following manner: first select the feature with highest correlation value, then select the second highest with the first one and continue in this way until there is no feature left.

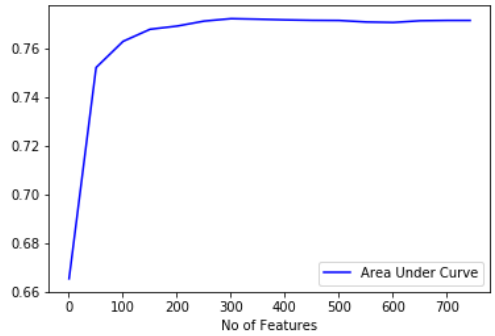
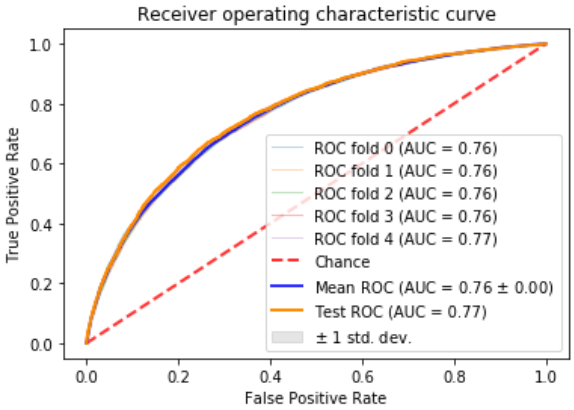
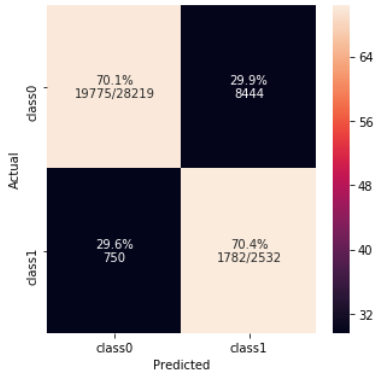


Figure 16: Change of AUC with Increasing Number of Correlated Features

Figure 16 represents how the model AUC score has been changed over the increasing number of correlated features. After the first 180 correlated features, the improvement in AUC score became almost constant. Hence, a set with the first 180 correlated features has been considered to train the final model. Figure 17 represents mean cross-validation and test AUC score of the model with the ROC curve and confusion matrix which shows improvement from the baseline model. It has been able to correctly detect both defaulters and non-defaulters ~70% of the time. For logistic regression, the feature importance has been calculated using absolute model coefficients for each feature (figure 18). In compliance with the initial assumptions, Interestingly, EXT_SOURCE_2 and EXT_SOURCE_3 are in top 5 important predictors. Also, CODE_GENDER in top 10 reflects strong gender dependency on clients' defaulting tendency.



(a) ROC Curve



(b) Confusion Matrix for Test Data

Figure 17: Performance of Logistic Regression with first 180 Correlated Features

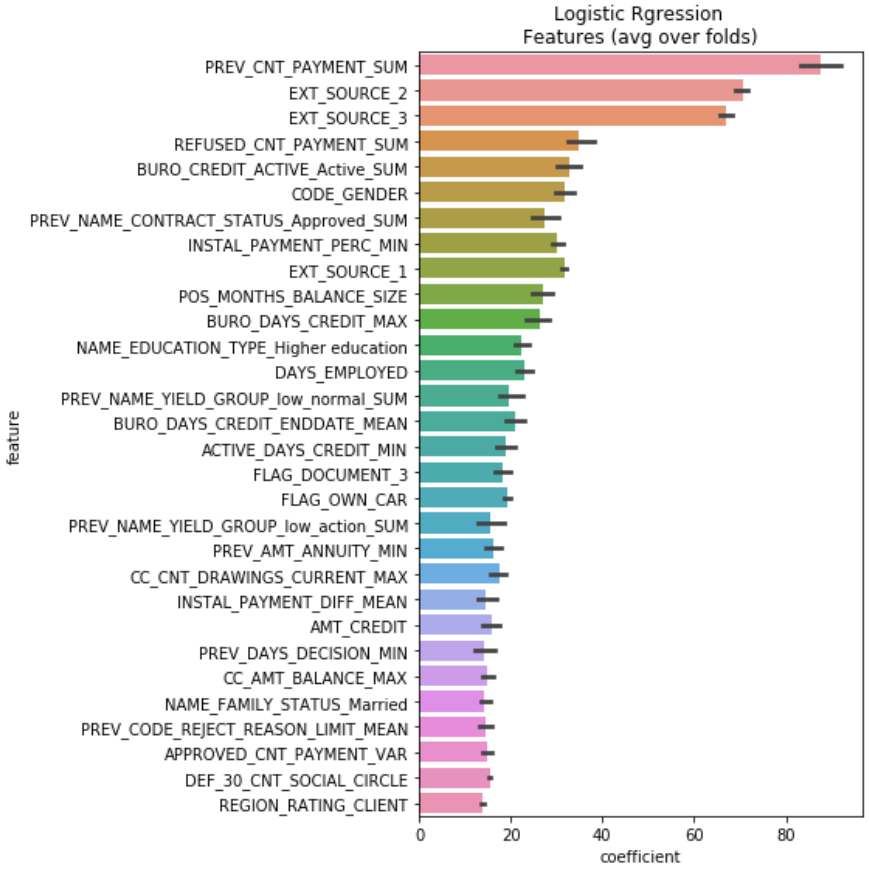


Figure 18: Feature Importance in Logistic Regression with Correlated Features

ii) Recursive Feature Elimination: For the next approach RFECV has been applied to the data set where it removed 50 (step size) least important features at each step and fit the rest in a logistic regression model. Figure 19 shows that the optimal number of features is 189. If the step size were finer, the result would have been more accurate. However, since it takes a long time to run even with a large step, 50 seemed to be the right one to select for now.

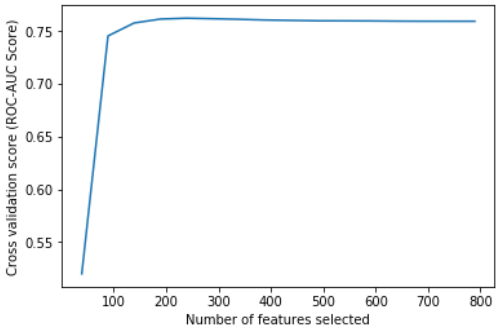
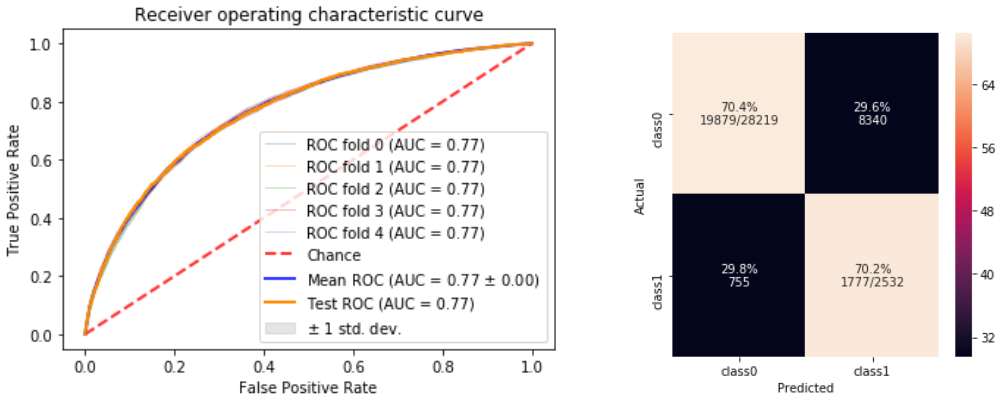


Figure 19: Change of AUC with Increasing Number of Features



(a) ROC Curve

(b) Confusion Matrix

Figure 20: Performance of Logistic Regression with RFE selected features

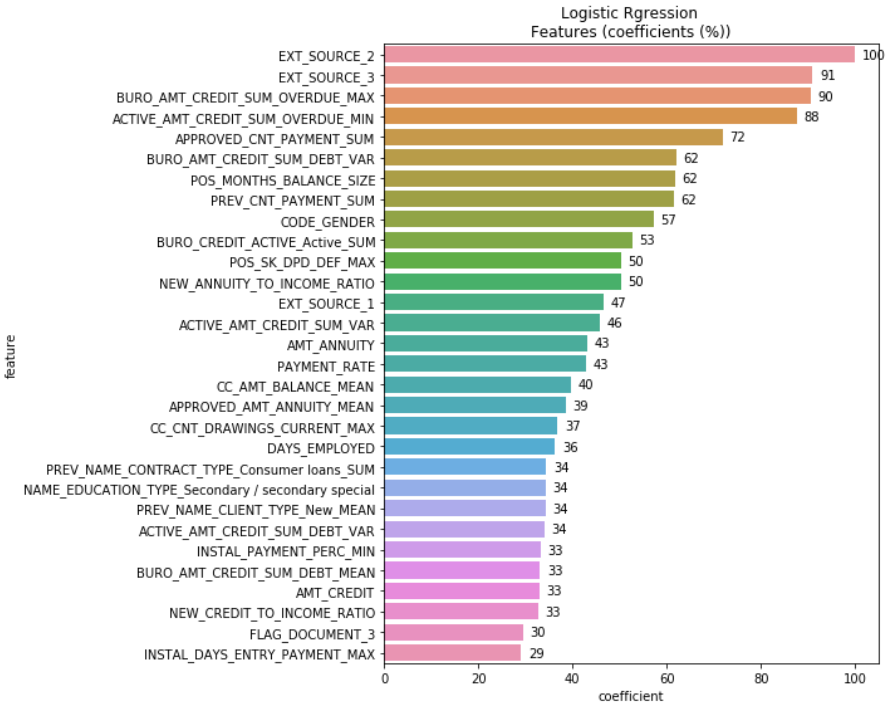
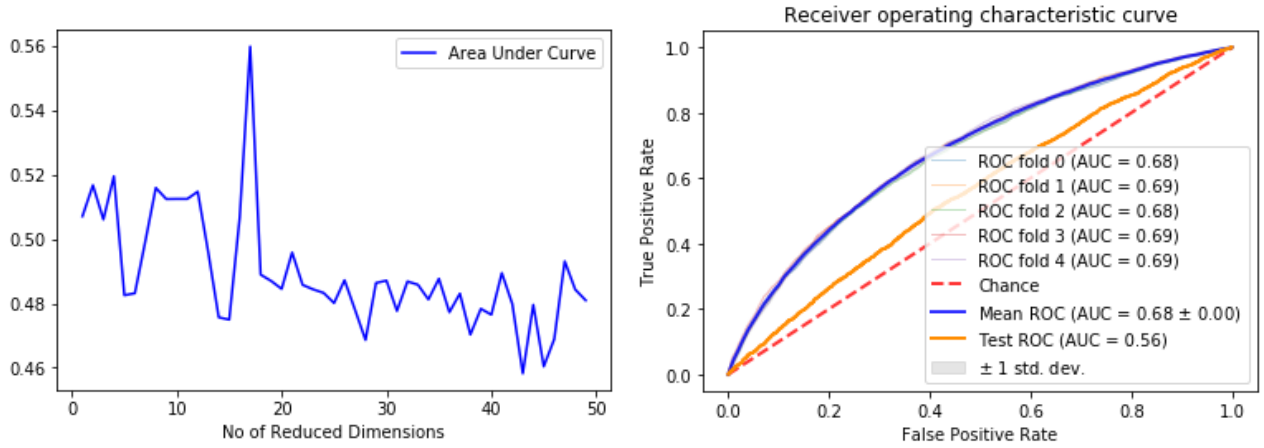


Figure 21: Feature Importance in Logistic Regression with RFE Selected Features

Figure 20 shows the results from the final model. The mean cross-validation ROC-AUC is 0.77 and test ROC-AUC is 0.77. This model has been able to identify ~71% loan payers and ~70% defaulters correctly. The feature importance for this model in figure 21 shows that EXT_SOURCE_2 and EXT_SOURCE_3 have again made in the top 5.

iii) PCA: Figure 22a represents how the AUC score has changed over different number of components in PCA. Although the best AUC has been achieved with 17 components, the performance (figure 22b) is still quite poor compared to other approaches.

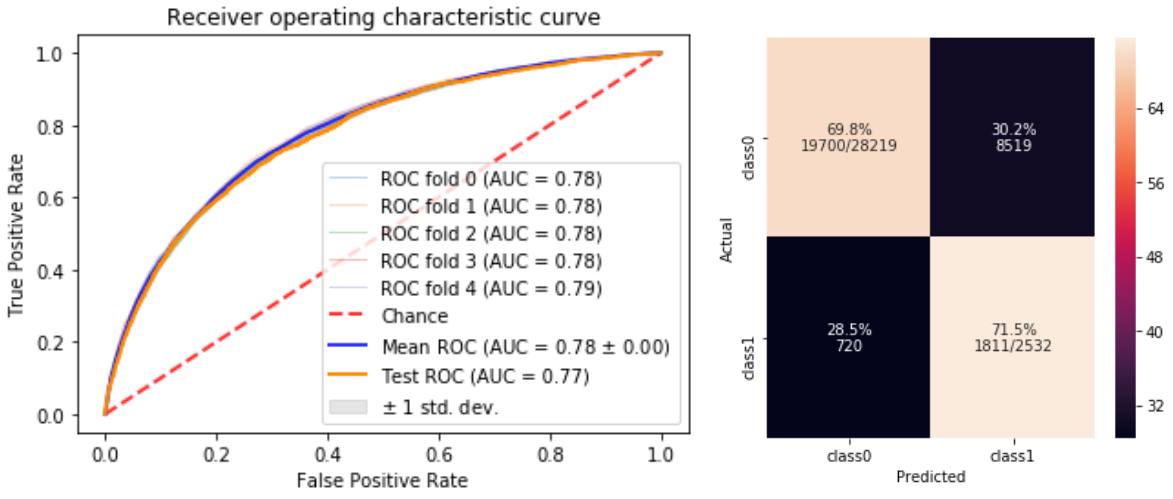


(a) Change of AUC with increasing number of components

(b) ROC Curve

Figure 22: Performance of Logistic Regression with PCA-reduced Data

iv) FLD: On the other hand, the performance achieved by FLD reduced data has outperformed all the previous performances of logistic regression with mean cross-validation AUC = 0.78 (figure 23). It has been able to predict ~70% of the loan repayers and ~72% of non-repayers.



(a) ROC Curve

(b) Confusion Matrix

Figure 23: Performance of Logistic Regression with FLD-reduced Data

3.4.4 Results from Kaggle competition: After these two models have been trained and tested with fair AUC score using the only training data, their generalizability to more unseen data have been tested by Kaggle provided testing data. The final predicted probabilities have been submitted to Kaggle to see where these models actually stand.

The results of both the models from there have been included in the following table. The public leaderboard is calculated with approximately 20% of the test data and the private one is calculated with 80% of the test data.

Table 5: Results from Kaggle Kernel

Model	Private Score	Public Score	Winning Score
LightGBM	0.77482	0.77204	0.80570 (private), 0.81724 (public)
Logistic Regression (Pearson Corr.)	0.75102	0.74863	N/A
Logistic Regression (LDA)	0.75203	0.74723	N/A

Although the scores are not as good as the winning ones, it can be considered as a fair achievement compared to the time frame.

4 Summary and Future Work:

The main purpose of this project was to determine loan defaulters based on certain criteria and to have a better understanding of how each factor influences clients' behavior in this regard using machine learning models. Statistical approach logistic regression and tree-based gradient boosting framework lightGBM performed the best in terms of both AUC and run-time. Although both of them had locally tested ROC-AUC score ~0.77, lightGBM gave the best performance when exposed to the actual testing data from Kaggle.

The reduction of generalizability beyond the training data due to its tendency to overfit with a large number of features may have caused the discrepancy in performances of logistic regression. Modeling with mean-imputed data can be another reason for this inconsistency, as it can easily induce biases in the predictions. Nonetheless, a careful further investigation will provide a clearer idea on what is indeed going on.

From feature importance in different models, it can be said that normalized scores from external data sources, clients' age at the time of application, clients' gender and amount of loan annuity have immense influence on the response variable. The result also somewhat corroborates the initial assumptions made in the exploratory data analysis part. However, some engineered features have also appeared in the top 30. Some of them are rate of payment, maximum how many days before application the installments of previous credit were actually paid, maximum how many days before application the last information about the closed credit bureau reported credits came, total number of payments made before application, how many credit bureau reported credits were active at the time of application and so on.

Although the results are fair enough, there are plenty of scopes to improve the models. Some of the advanced missing value imputation techniques can be adapted so that the data set is induced with minimum biases. To avoid losing any important information from the data set, in place of random undersampling, cluster-based undersampling method can be used with a more efficient way of handling large data set. For

further improvement in lightGBM model, some other hyperparameters can be optimized with extended grids.

Last but not least, this project has been a great learning experience which served its purpose. This provided a better understanding of how machine learning model works and what should be the workflow in every data science project.

References:

1) https://www.kaggle.com/c/home-credit-default-risk/data

2) Bartual, C., Garcia, F., Guijarro, F. and Moya, I. (2013). Default Prediction of Spanish Companies. A Logistic Analysis. *Intellectual Economics*, 7(3), pp.333-343.

3) Nehrebecka, N. (2018). PREDICTING THE DEFAULT RISK OF COMPANIES. COMPARISON OF CREDIT SCORING MODELS: LOGIT VS SUPPORT VECTOR MACHINES. *ECONOMETRICS*, 22(2), pp.54-73.

4) Memic, D. (2015). Assessing Credit Default using Logistic Regression and Multiple Discriminant Analysis: Empirical Evidence from Bosnia and Herzegovina. *Interdisciplinary Description of Complex Systems*, 13(1), pp.128-153.

5) Ying, L. (2018). Research on bank credit default prediction based on data mining algorithm. *The International Journal of Social Sciences and Humanities Invention*, 5(6), pp.4820-4823.

6) http://web.eecs.utk.edu/~qi/ece471-571/syllabus.htm

7) https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-9 34a444fca47

8) https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

9) Meng, Qi. (2018). LightGBM: A Highly Efficient Gradient Boosting Decision Tree.

10) https://www.kaggle.com/c/home-credit-default-risk/kernels

11) Azur, M., Stuart, E., Frangakis, C. and Leaf, P. (2011). Multiple imputation by chained equations: what is it and how does it work?. International Journal of Methods in Psychiatric Research, 20(1), pp.40-49.

12) Allen, M. (1997). The problem of multicollinearity. In: Understanding Regression Analysis. Springer, Boston, MA