

A Machine Learning Approach to Credit Risk Default Analysis

Atul Singh, NMIMS, MPSTME, Mumbai, atulsingh20.nmims@gmail.com

Abstract: Increasing number of loan default is a major issue that the various financial institutions are facing today. These institutions are working towards identifying whether an applicant would default or not, based on the various data points that are available about the applicant. Data points can be of very high dimensions and need not be only financial in nature. The data is provided by the Home Credit Group which is a leading consumer durable loan provider, offers a mobile loan, two-wheeler, home appliances loans. This project aims to unlock the full potential of the real time data by doing an in-depth analysis of various parameters taken when considering an applicant and building a Machine learning model to classify an applicant whether he would default or not.

Keywords —Anomaly Detection, Credit Risk, EDA, Feature Engineering, Feature Selection, Machine Learning

I. PROCESS FLOW

For predicting the defaults, Home Credit Group provided the consumer records for the loans taken. This data is highly granular and contains data points which can significantly predict the loan repayment ability of a consumer.

A literature review has been done which involved looking into different approaches and metrics that had been used in the past to tackle the credit default problem.

Extensive Data Analysis has been done and important insights have been gathered. Using these insights, new variables (features) have been developed. To help with the process of understanding the data, an applicant analysis report is created which allows the end user to view all the information related to a consumer.

Feature Selection i.e., choosing those features that were highly significant has been done using techniques like filtering and embedded techniques.

After the final table is created, various techniques like Anomaly Detection and Classification have been applied to predict the possibility of default.

Once we select the best model, we create a User Interface, allowing the user (Company Employee) to enter consumer information and as a result the user gets a probability score depicting the probability of default for the consumer.

II. EXPLORATORY DATA ANALYSIS

Exploratory data analysis (EDA) is an approach for analyzing data set to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

The data is provided by the Home Credit Group which is a leading consumer durable loan provider, offers a mobile loan, two-wheeler, home appliances loans. During the analysis we looked into various aspect of the data set like:

- Structure of Data
- Feature Data Distribution
- Analyzing and Handling Missing Values
- Applicant Analysis

Structure of Data

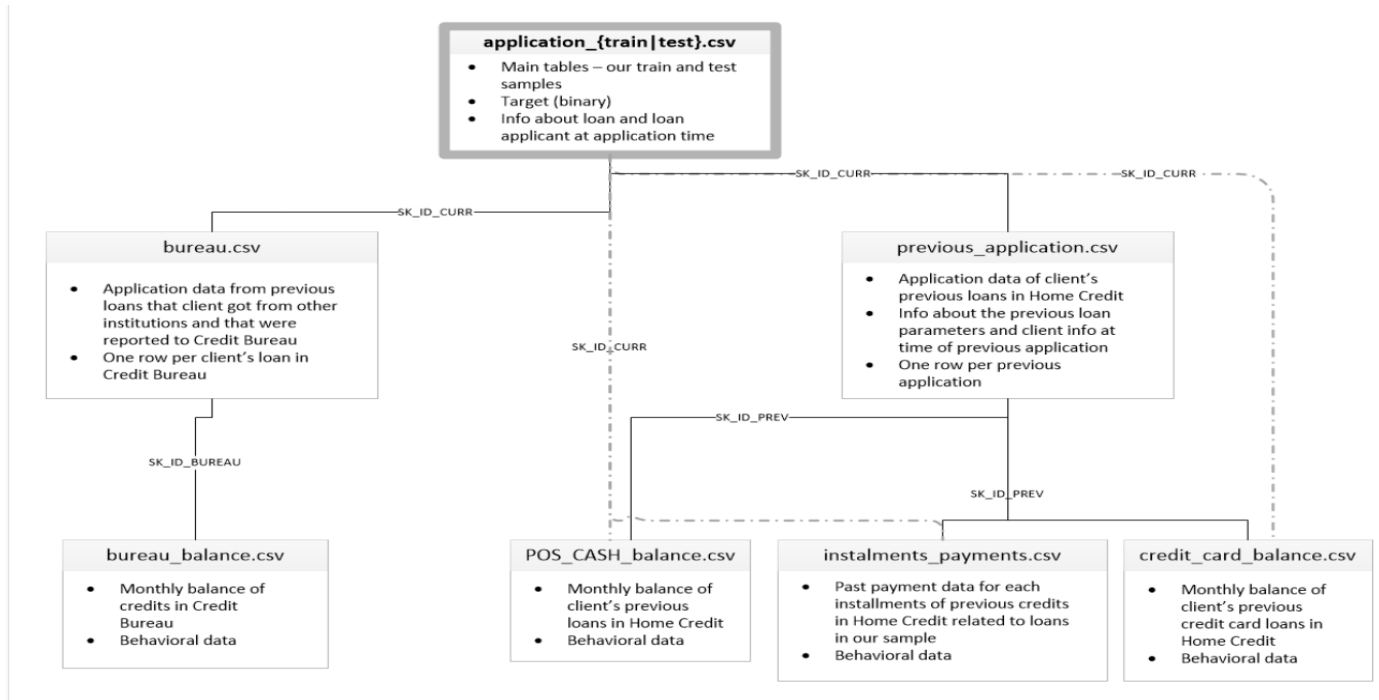


Figure 1: Dataset flowchart

application_{train|test}.csv

This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET).

Static data for all applications. One row represents one loan in our data sample.

bureau.csv

All clients' previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample).

For every loan in our sample, 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 – i.e. the table has (#loans in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.

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 our sample – i.e. the table has (#loans in sample * # of relative previous credits * # of months in which we have some history observable for the previous credits) rows.

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 our sample – i.e., the table has (#loans in sample * # of relative previous credit cards * # of months where we have some history observable for the previous credit card) rows.

previous_application.csv

All previous applications for Home Credit loans of clients who have loans in our sample.

There is one row for each previous application related to loans in our data sample.

instalments_payments.csv

Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample.

There is a) one row for every payment that was made plus b) one row each for missed payment.

One row is equivalent to one payment of one instalment OR one instalment corresponding to one payment of one previous Home Credit credit related to loans in our sample.

HomeCredit_columns_description.csv

This file contains descriptions for the columns in the various data files.

Data Distribution

1). Application_train data

We have a total of 67 unique variables (columns) in total in data set and not every variable will be required to build a machine learning model or will significantly affect the final prediction/result of the model. Some feature engineering needs to be done to identify significant variables and to understand if we need to make new variables by making some combination of existing variable to improve the accuracy

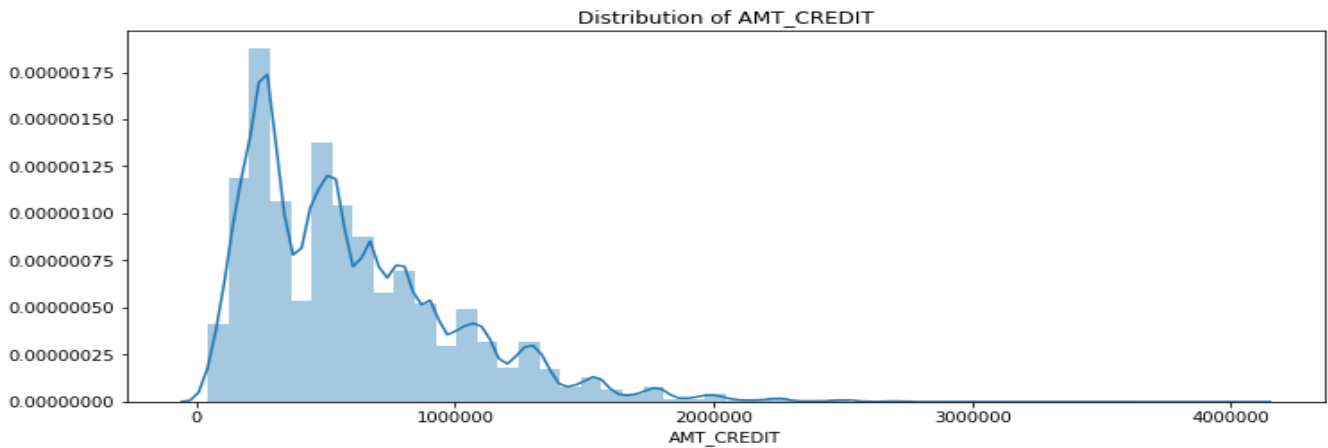


Figure 2: Distribution of Amt_Credit

We see that the data is right skewed we need to correct before using it we can correct it by taking log of the values or using scaling function.

As shown in below figure majority of population lies between 0.04-0.85 million. So, majority (237833) of credits issued are between 0.04-0.85 million

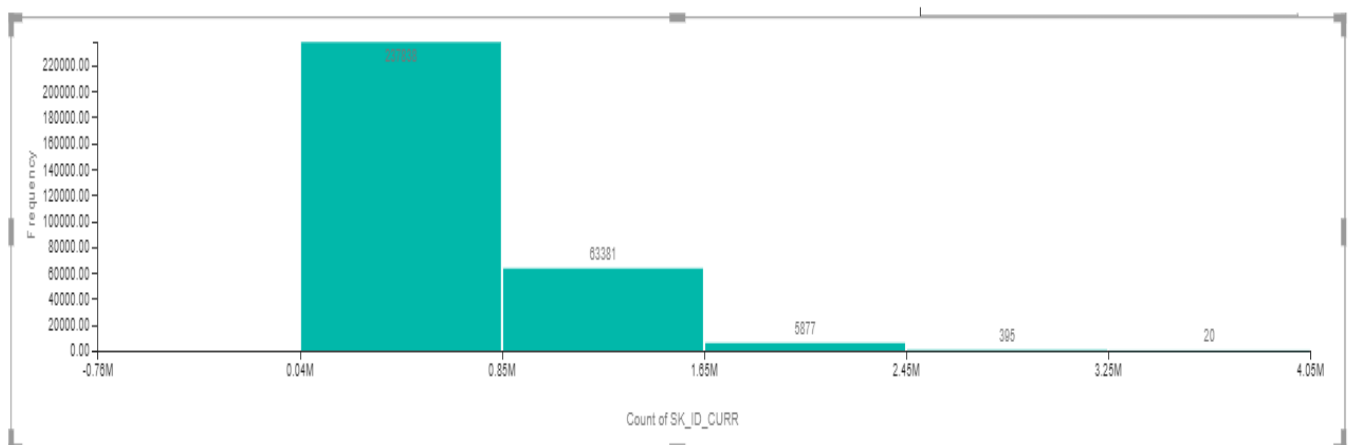


Figure 3: AMT_CREDIT distribution

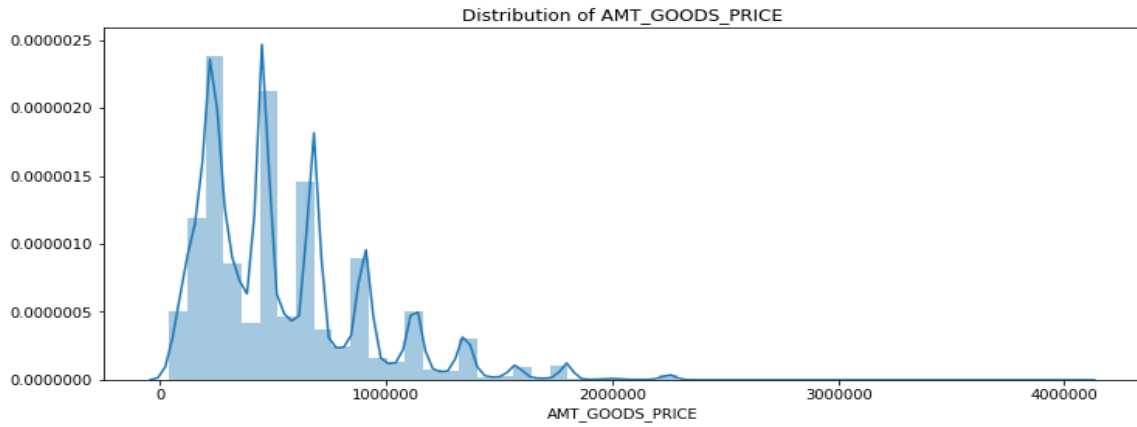
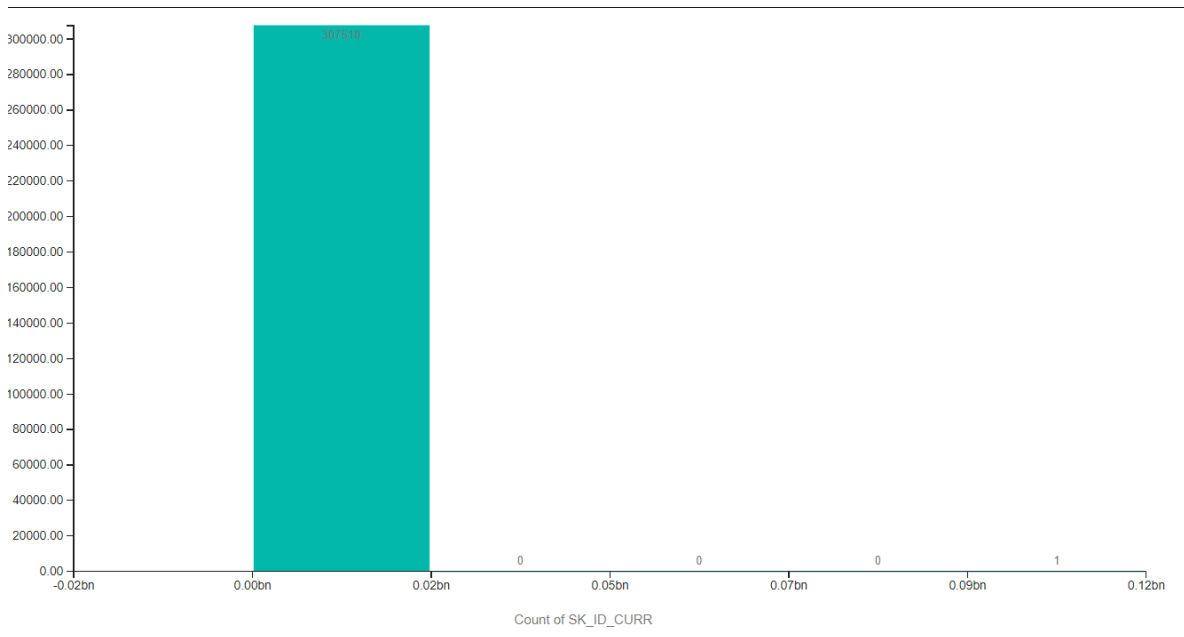


Figure 4: Distribution of Amt_Goods_Price

We see that the data is right skewed we need to correct before using it we can correct it by taking log of the values or using scaling function



I.

Figure 5: Amt_Income distribution

The above figure depicts the income distribution of various borrowers. From this figure we can conclude that majority of population annual income is less than 0.02bn. Except there is 1 outlier whose income lies in 0.09bn range.

Data is balanced or imbalanced

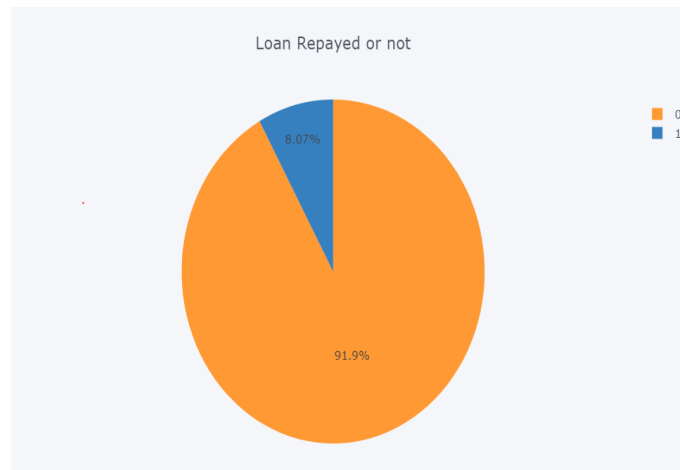


Figure 6: Percentage of loan repaid and non-repaid

The above figure depicts percentage of loan repaid or non-repaid. The data is imbalanced (91.9% (Loan repayed-0) and 8.07 % (Loan not repayed-1))

Types of loan

Revolving loans: Arrangement which allows for the loan amount to be withdrawn, repaid, and redrawn again in any manner and any number of times, until the arrangement expires. Credit card loans and overdrafts are revolving loans. Also called evergreen loan

Cash Loan: A cash basis loan is one in which interest is recorded as earned when payment is collected. Ordinarily, interest income is accrued on loans, as regular payment of both principal and interest is assumed.

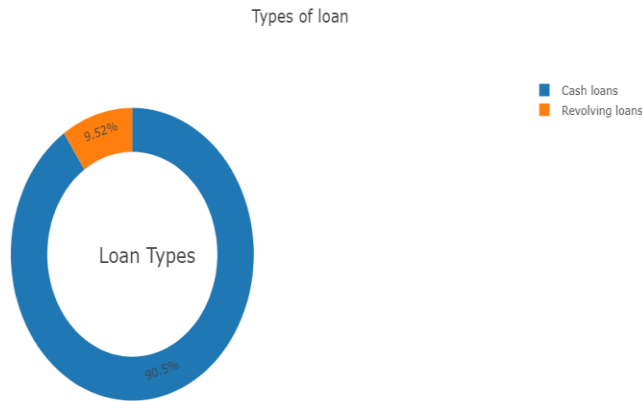


Figure 7: Percentage of type of loan

The above figure depicts the type of loan issued by bank. Most of the loans are Cash loans which were taken by applicants. 90.5 % loans are Cash loans.

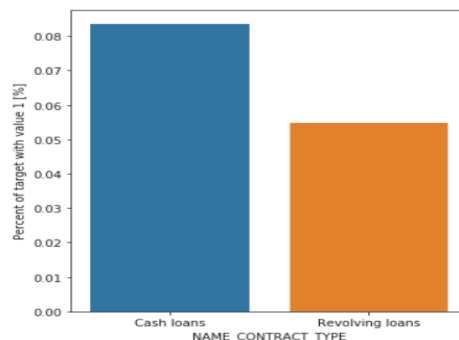


Figure 8: Default rate of loans contract wise

As shown in the above graph, Cash loans have high default rate of 8% as compare to revolving loans.

Client Gender

The number of female clients is almost double the number of male clients. Looking to the percent of defaulted credits, males have a higher chance of not returning their loans (~10%), comparing with women (~7%).

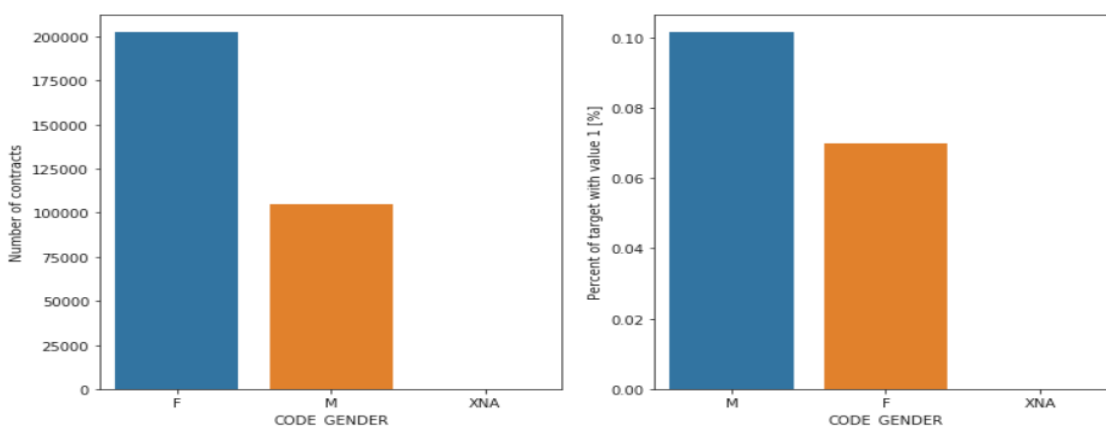


Figure 9: Number of contract and default rate based on client gender

Number of children

Let's see the distribution of the number of children of the clients.

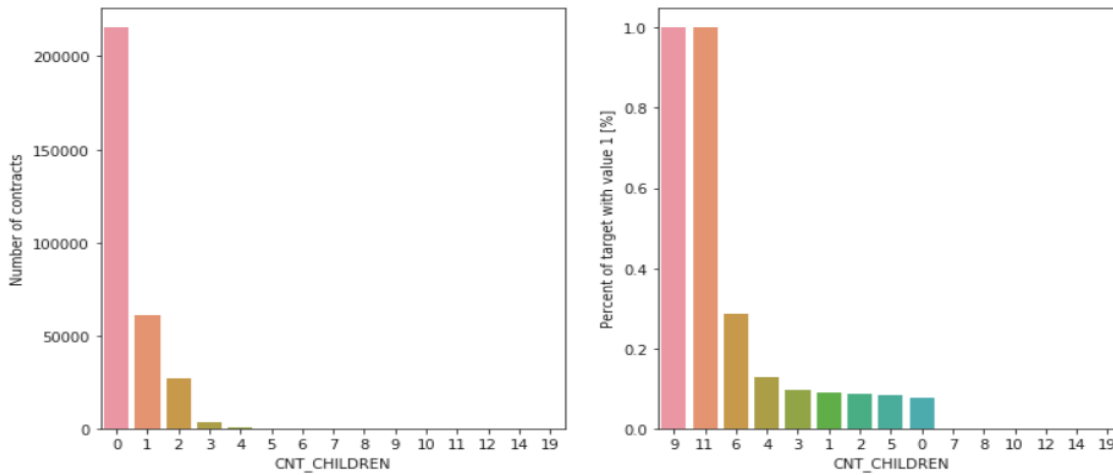


Figure 10: Number of contract and default rate based on count of children

Most of the clients taking a loan have no children. The number of loans associated with the clients with one child are 4 times smaller, the number of loans associated with the clients with two children are 8 times smaller; clients with 3, 4 or more children are much rarer.

As for repayment, clients with no children, 1, 2, 3, and 5 children have percent of no repayment around the average (10%). The clients with 4 and 6 children are above average in terms of percent of not paid back loans (over 25% for families with 6 children).

As for clients with 9 or 11 children, the percent of loans not repaid is 100%.

Income sources of Applicant's who applied for loan

Let's investigate the numbers of clients with different income type. As well, let's see the percent of not returned loans per income type of applicants.

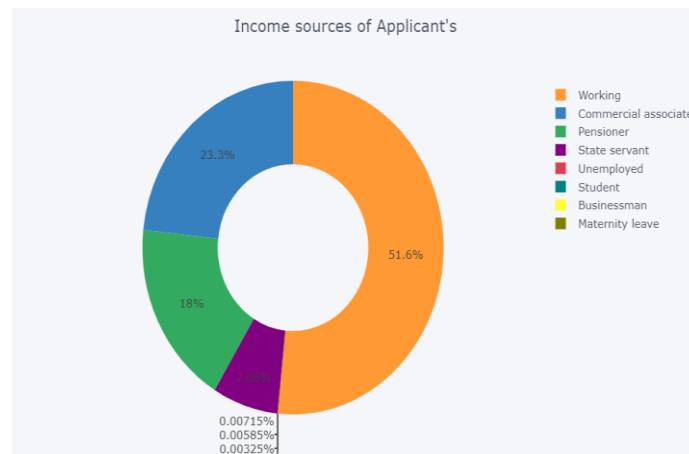


Figure 11: income source distribution

The above figure shows income source of applicants. Majority of population who applied for the loan are working (51.6%), while 23.3% of population are commercial associate, 18% of applicant's income source is pension and 7.08 % of applicants are state servant

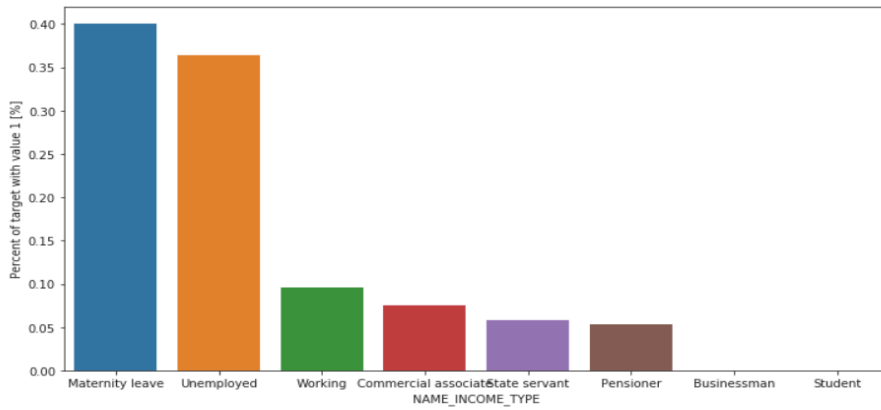


Figure 12: Default rate based on income source of applicant

The applicants with the type of income Maternity leave have almost 40% ratio of not returning loans, followed by Unemployed (37%). The rest of types of incomes are under the average of 10% for not returning loans.

Occupation of Applicant's who applied for loan

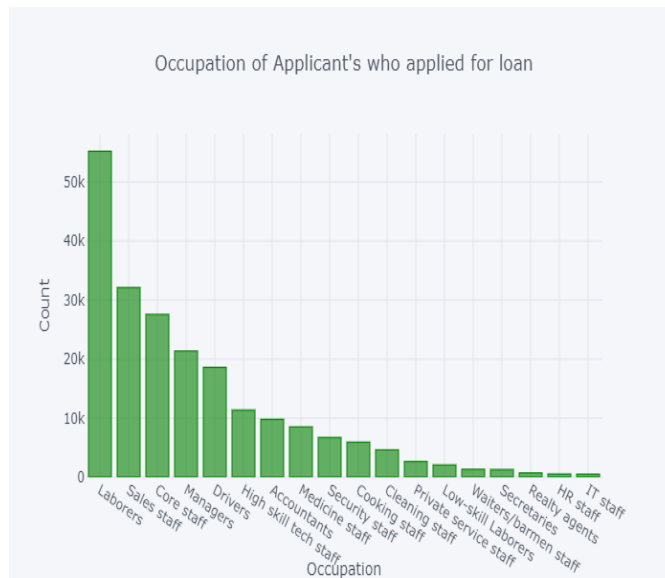


Figure 13: Applicant occupant distribution

The above figure depicts the occupation of applicants who applied for the loan. Highest no of applicant's occupation is laborers followed by sales staff, core staff, managers, drivers, high skill tech staff, accountants, medicine staff

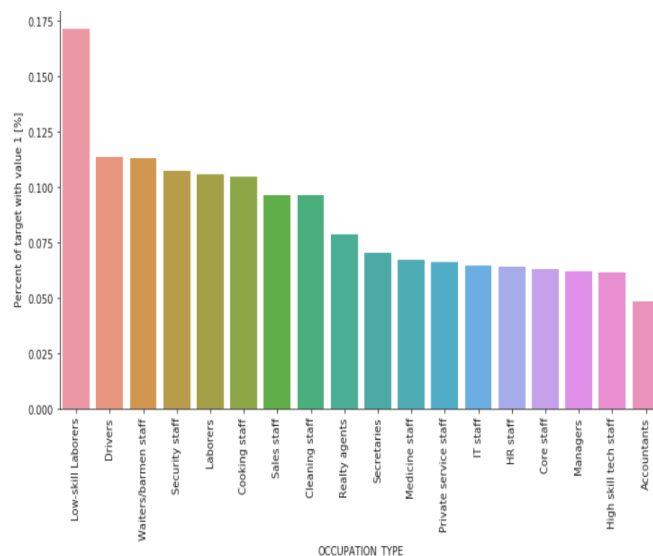


Figure 14: Default rate based on occupation type

Most of the loans are taken by Laborers, followed by Sales staff. IT staff take the lowest amount of loans. The category with highest percent of not repaid loans are Low-skill Laborers (above 17%), followed by Drivers and Waiters/barmen staff, Security staff, Labourers and Cooking staff.

Education of Applicant's who applied for loan

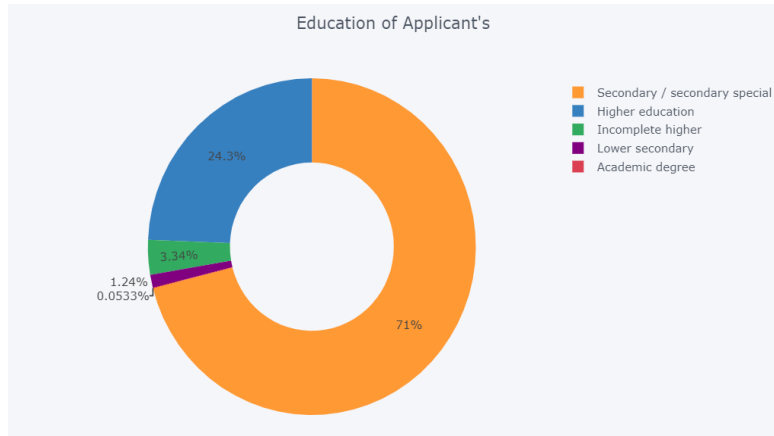


Figure 15: Education of applicant distribution

The above figure shows the education of applicants. 71 % applicants have secondary and 24.3 % having higher education while rest have incomplete higher education and lower secondary.

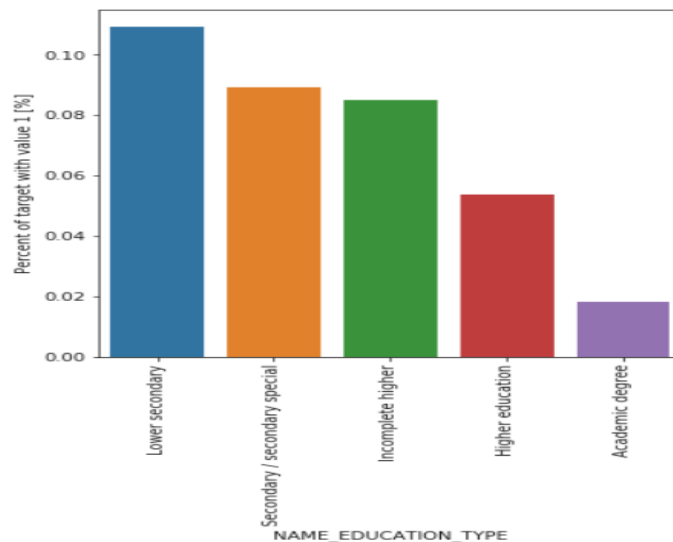


Figure 16: Default rate based on education type

The Lower secondary category, although rare, have the largest rate of not returning the loan (11%). The people with Academic degree have less than 2% not-repayment rate.

2) Bureau Data

Bureau data contains 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). For every loan in the sample, there are as many rows as number of credits the client had in Credit Bureau before the application date. SK_ID_CURR is the key connecting application_train|test data with bureau data.

Credit status

Let's see the credit status distribution. We show first the number of credits per category (could be Closed, Active, Sold and Bad debt).

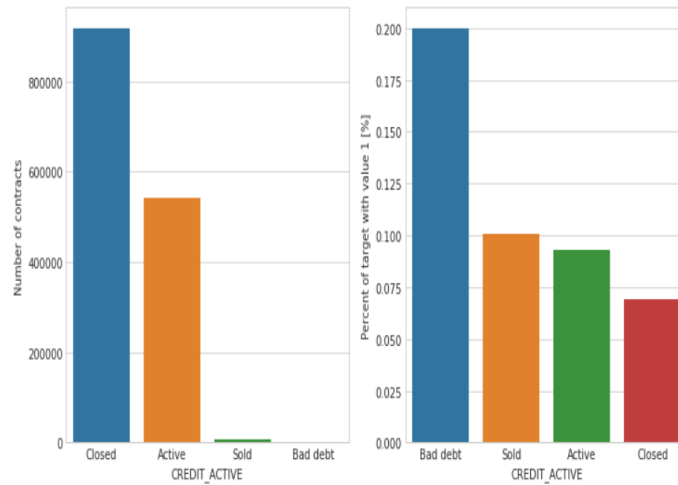


Figure 17: Number of contract and default rate based on credit status

Most of the credits registered at the Credit Bureau are in the status Closed (~900K). On the second place are the Active credits (a bit under 600K). Sold and Bad debt are just a few.

In the same time, as percent having TARGET = 1 from total number per category, clients with credits registered to the Credit Bureau with Bad debt have 20% default on the current applications.

Clients with credits Sold, Active and Closed have percent of TARGET == 1 (default credit) equal or less than 10% (10% being the rate overall). The smallest rate of default credit have the clients with credits registered at the Credit Bureau with Closed credits.

That means the former registered credit history (as registered at Credit Bureau) is a strong predictor for the default credit, since the percent of applications defaulting with a history of Bad debt is twice as large as for Sold or Active and almost three times larger as for Closed

Credit type

Majority of historical credits registered at the Credit Bureau are Consumer credit and Credit card. Smaller number of credits are Car loan, Mortgage and Microloan.

Looking now to the types of historical credits registered at the Credit Bureau, there are few types with a high percent of current credit defaults, as following:

Loan for the purchase of equipment - with over 20% current credits defaults;

Microloan - with over 20% current credits defaults;

Loan for working capital replenishment - with over 12% current credits defaults.

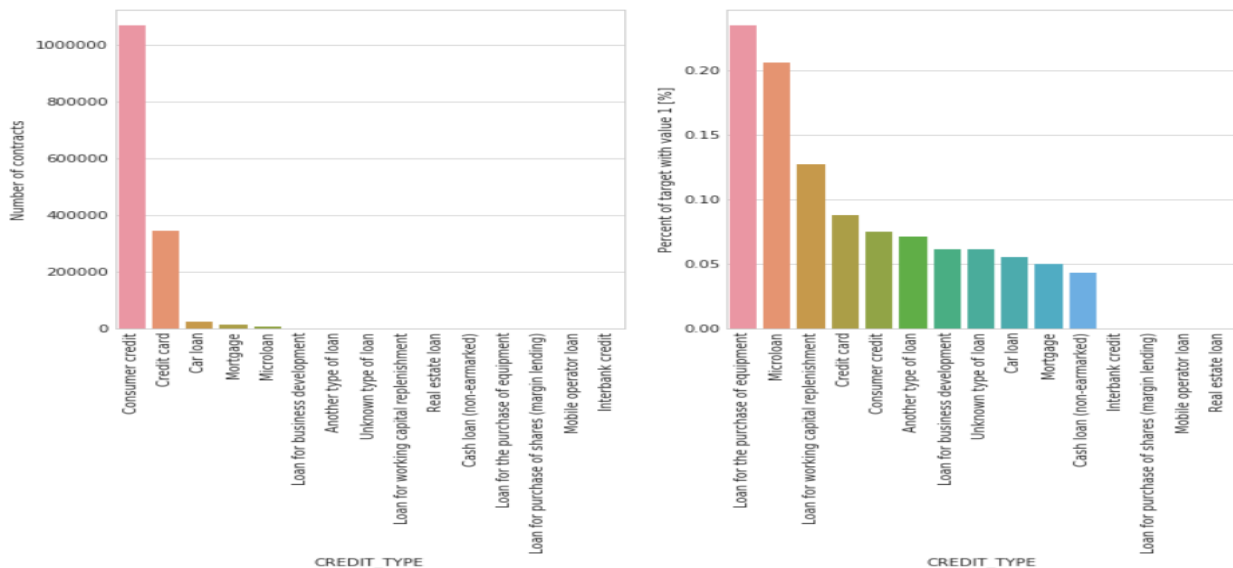


Figure 18: Number of contract and default rate based on credit type

3). Previous application data

The data frame previous_application contains information about 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 our data sample. SK_ID_CURR is the key connecting application_train|test data with previous_application data

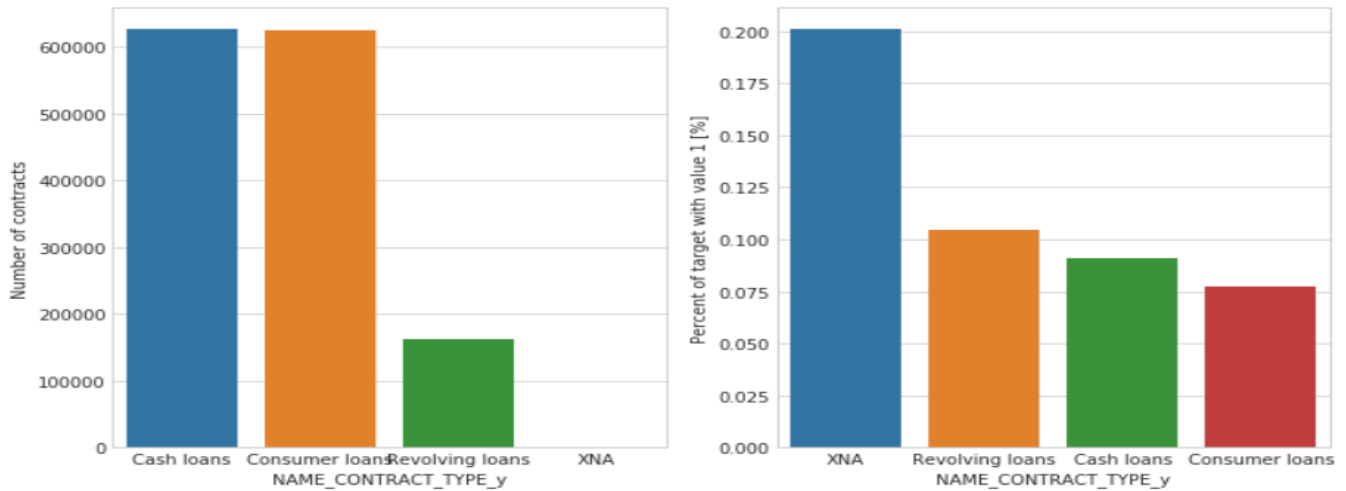


Figure 19: Number of contract and default rate based on contract type

Contract type

There are three types of contract in the previous application data: Cash loans, Consumer loans, revolving loans. Cash loans and Consumer loans are almost the same number (~600K) whilst revolving loans are ~150K.

The percent of defaults loans for clients with previous applications is different for the type of previous applications contracts, decreasing from ~10% for Revolving loans, then ~9.5% for Cash loans and ~8% for Consumer loans.

Contract status

Most previous applications contract statuses are Approved (~850K), Cancelled and Refused (~240K). There are only ~20K in status Unused offer.

In terms of percent of defaults for current applications in the sample, clients with history of previous applications have largest percent of defaults when in their history contract statuses are Refused (12%), followed by Cancelled (9%), Unused offer (~8%) and Approved (lowest percent of defaults in current applications, with less than 8%).

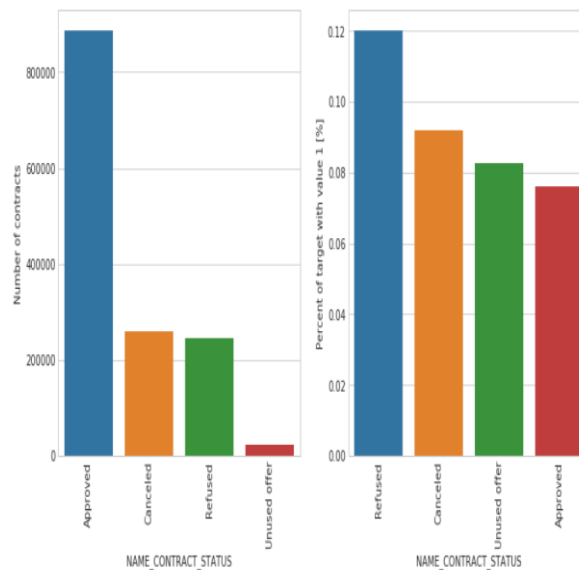


Figure 20: Number of contract and default rate based on contract status

Most previous applications contract statuses are Approved (~850K), Cancelled and Refused (~240K). There are only ~20K in status Unused offer.

In terms of percent of defaults for current applications in the sample, clients with history of previous applications have largest percent of defaults when in their history contract statuses are Refused (12%), followed by Cancelled (9%), Unused offer (~8%) and Approved (lowest percent of defaults in current applications, with less than 8%).

Client type

The below given graph tells us the client types for previous applications

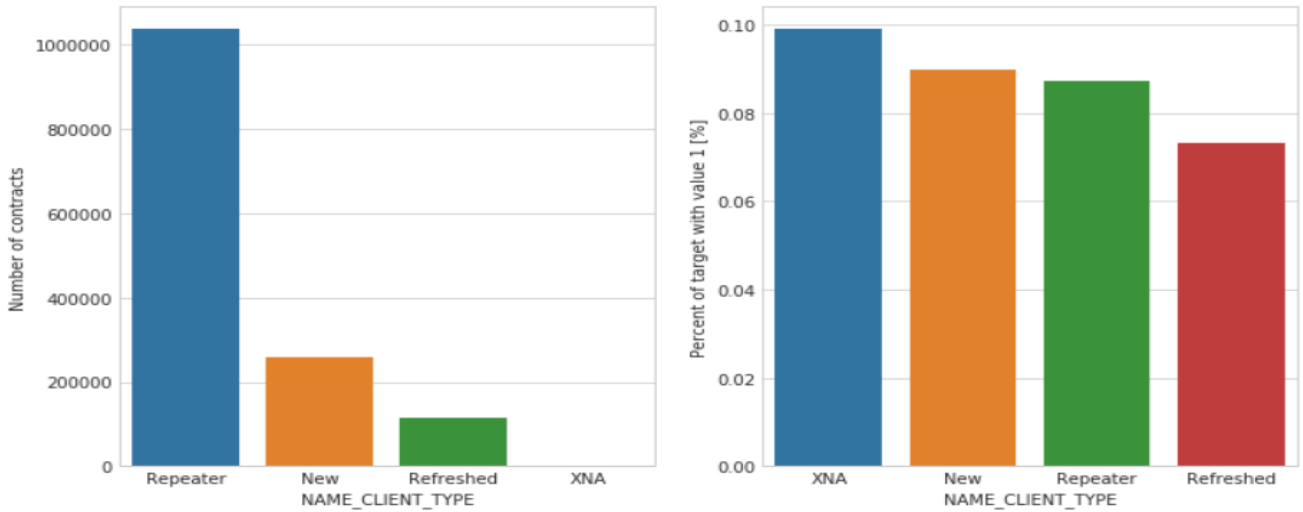


Figure 21: Number of clients and default rate based on client type

4). Applicant Analysis

In this part of the project, we performed an in-depth analysis of an application who has taken loan, in order to understand the data and the parameters available to us in the dataset. And also, to check what kind of loans the applicant takes and the behavior of the applicant with respect to payment of the loan.

As an example, we are going to study the applicant i.e., the demographics, earning patterns and answering questions like when and why he/she took a loan and what have been the behavior of the applicant when repaying the loan.

Analyzing Non-Defaulter

Analysis of an applicant who has not defaulted:

The application id 100048 has taken a loan of Rs. 604,152 (assuming it is INR) for a good costing Rs. 540,000. He/she has to pay Rs. 29,196 as annuity to the Home Credit Company. The applicant has an income of Rs. 202,500.

The applicant is 47 years old and married. He/she does not have any child and lives in a house/apartment which is owned but does not own a car.

The applicant has completed secondary education and is employed for 1.5 years approx. and works in a Type 1 Industry.

Application ID: 100048				0 TARGET
Application Details				
29,196.00 AMT_ANNUITY	604,152.00 AMT_CREDIT	540000 AMT_GOODS_PRICE	202,500.00 AMT_INCOME_TOTAL	
0 CNT_CHILDREN	2 CNT_FAM_MEMBERS	-16971 DAYS_BIRTH	-475 DAYS_EMPLOYED	
N FLAG_OWN_CAR	Y FLAG_OWN_REALTY	Secondary / secondary spec... NAME_EDUCATION_TY...	Married NAME_FAMILY_STATUS	
House / apartment NAME_HOUSING_TYPE	Working NAME_INCOME_TYPE	Laborers OCCUPATION_TYPE	Industry: type 1 ORGANIZATION_TYPE	

Figure 22: Applicant No. 100048- Application Train

Application ID

100048

4

Count of SK_ID_PREV

SK_ID_PREV	NAME_GOODS_CATEGORY	NAME_CONTRACT_STATUS	AMT_CREDIT	AMT_ANNUITY	AMT_APPLICATION	CNT_PAYMENT	DAYS_FIRST_DUE	DAYS_LAST_DUE
1060558	XNA	Approved	225,000.00	11,250.00	225,000.00	0	-878	365243
1366912	XNA	Approved	856,431.00	57,466.53	814,500.00	18	-284	365243
1366913	XNA	Approved	649,345.50	46,850.27	540,000.00	18	-428	-308
1827758	XNA	Canceled	0.00		0.00			
Total			1,730,776.50	115,566.80	1,579,500.00	36	-1590	730178

SK_ID_PREV	NAME_GOODS_CATEGORY	NAME_PRODUCT_TYPE	CHANNEL_TYPE	CNT_PAYMENT	PRODUCT_COMBINATION	NAME_SELLER_INDUSTRY	NAME_YIELD_I
1060558	XNA	walk-in	Credit and cash offices	0	Card Street	XNA	XNA
1366912	XNA	x-sell	Credit and cash offices	18	Cash X-Sell: low	XNA	low_normal
1366913	XNA	x-sell	Credit and cash offices	18	Cash X-Sell: middle	XNA	middle
Total				36			

Figure 23: Applicant No. 100048 - Previous Application

The applicant has taken 4 loans in the past from Home Credit Company out of which 3 were approved but 1 was not approved. Out of the three loans 1 loan has been completed and the rest 2 loans are still active.

2 loans were on the basis of cross-selling and were cash loans.

The active loans were taken 2 years and 0.7 years ago and the applicant pays a total of Rs.98000 (11250+57500+30000) as annuity of the total income of Rs. 200,000 that he/she earns, which is 50% of the total income earned. The higher is the percentage the riskier is the customer.

Let us analyze the loan no. 1366913

As seen in the below figure, the loan was taken by Credit and Cash Offices which was a result of cash cross-selling. The loan was repaid in 120 days (408-308 days). Hence we see a pre-payment behavior of the loans.

Pre-payment is a good sign as the Company gets back the loan, but it does affect its cash inflows as it receives a lesser interest than what was expected.

In the figure we notice the repayment pattern of the applicant who paid the amount due 10 months beforehand in a lump-sum.

Application ID

100048

Application ID

1366913

6

Count of SK_ID_PREV

SK_ID_CURR	SK_ID_PREV	CNT_INSTALMENT_FUTURE	CNT_INSTALMENT	MONTHS_BALANCE	NAME_CONTRACT_STATUS
100048	1366913	0	5	-10	Completed
100048	1366913	18	18	-15	Active
100048	1366913	17	18	-14	Active
100048	1366913	16	18	-13	Active
100048	1366913	15	18	-12	Active
100048	1366913	14	18	-11	Active
600288		80			

Figure 24: Applicant No. 100048- POS Cash Data

Let's take a look at the credit card loan taken by the customer:

SK_ID_CURR	SK_ID_PREV	AMT_CREDIT_LIMIT_ACTUAL	NAME_CONTRACT	MONTHS_BALANCE	AMT_DRAWINGS_ATM_CURRENT	AMT_DRAWINGS_CURRENT
100048	1060558	225000	Active	-31		0.00
100048	1060558	225000	Active	-30	0	212,013.00
100048	1060558	225000	Active	-29	0	100,120.50
100048	1060558	225000	Active	-28	0	47,457.00
100048	1060558	225000	Active	-27	0	0.00
100048	1060558	225000	Active	-26	0	31,837.50
100048	1060558	225000	Active	-25	0	0.00
100048	1060558	225000	Active	-24	0	17,172.00
100048	1060558	225000	Active	-23	0	0.00
100048	1060558	225000	Active	-22	0	60,696.00
100048	1060558	225000	Active	-21	0	0.00
100048	1060558	225000	Active	-20	0	0.00
100048	1060558	225000	Active	-19	0	0.00
100048	1060558	225000	Active	-18	0	0.00

Figure 25: Applicant No. 100048 - Credit Card Usage Data

The applicant has a credit card limit of Rs. 225,000. The above table shows how the customer has used the credit card and what his /her spending habits have been. The applicant had drawn Rs. 212,013 out of his/her Rs. 225,000 limit which was mostly repaid by the next month.

AMT_DRAWINGS_CURRENT	AMT_INST_MIN_REGULARITY	AMT_PAYMENT_CURRENT	AMT_RECEIVABLE_PRINCIPAL	AMT_RECVIVABLE	AMT_TOTAL_RECEIVABLE
0.00			0.00	0.00	0.00
212,013.00			212,013.00	212,013.00	212,013.00
100,120.50	10,682.28	213,750.00	0.00	0.00	0.00
47,457.00	5,045.85	59,850.00	28,673.64	32,288.85	32,288.85
0.00	4,610.34	4,950.00	82,306.98	86,429.39	86,429.39
31,837.50	4,585.14	4,950.00	113,639.94	116,600.31	116,600.31
0.00	6,106.14	11,250.00	99,622.49	103,836.20	103,836.20
17,172.00	5,772.56	5,850.00	126,772.79	130,607.15	130,607.15
0.00	6,557.45	18,000.00	95,148.77	99,113.00	99,113.00
60,696.00	8,930.21	13,500.00	165,104.24	170,969.49	170,969.49
0.00	8,576.69	9,000.00	162,533.66	168,022.80	168,022.80
0.00	8,431.25	11,250.00	146,124.63	151,633.62	151,633.62
0.00	8,172.27	9,000.00	145,445.81	150,683.63	150,683.63
0.00	8,012.61	9,000.00	142,251.80	147,500.73	147,500.73

Figure 26: Applicant No. 100048 - Credit Card Usage Data

In the above figure we can see the spending pattern of Applicant and how his minimum instalments are changing with time. We can say that the applicant has enough liquidity and seems worthy of credit, hence it should not default when given a loan. As we have seen some pattern of the person who has not defaulted, let's analyze the profile of an applicant who has defaulted: -

Application ID 1 TARGET

Application Details

24,700.50 AMT_ANNUITY	406,597.50 AMT_CREDIT	351000 AMT_GOODS_PRICE	202,500.00 AMT_INCOME_TOTAL
---------------------------------	---------------------------------	----------------------------------	---------------------------------------

0 CNT_CHILDREN	1 CNT_FAM_MEMBERS	-9461 DAYS_BIRTH	-637 DAYS_EMPLOYED
--------------------------	-----------------------------	----------------------------	------------------------------

N FLAG_OWN_CAR	Y FLAG_OWN_REALTY	Secondary / secondary spec... NAME_EDUCATION_TY...	Single / not married NAME_FAMILY_STATUS
--------------------------	-----------------------------	--	---

House / apartment NAME_HOUSING_TYPE	Working NAME_INCOME_TYPE	Laborers OCCUPATION_TYPE	Business Entity Type 3 ORGANIZATION_TYPE
---	------------------------------------	------------------------------------	--

Figure 27: Applicant No. 100002- Application Train

The application id 100002 has taken a loan of Rs. 406,597 (assuming it is INR) for a good costing Rs. 351,000. He/she has to pay Rs. 24,700 as annuity to the Home Credit Company. The applicant has an income of Rs. 202,500.

The applicant is 26 years old and not married. He/she does not have any child and lives in a house/apartment which is owned but does not own a car.

The applicant has completed secondary education and is employed for 1.75 years approx. and works in a Business Entity Type 3.

Analyzing Defaulter

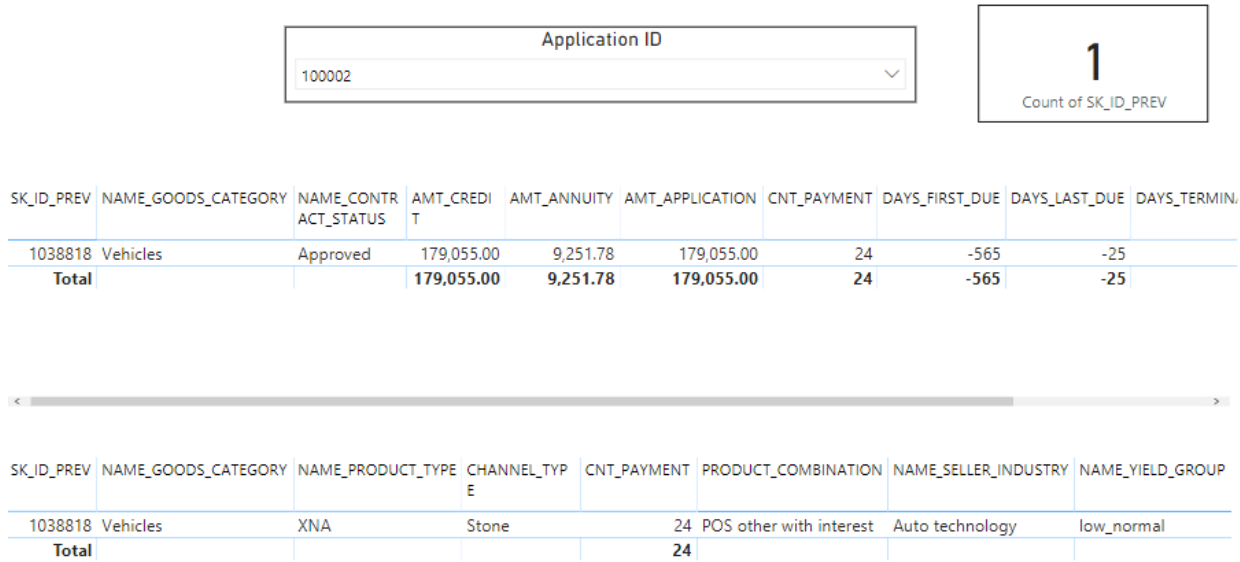


Figure 28: Applicant No. 100002- Previous Loans

We can see the applicant has taken 1 loan in the past from Home Credit Company. This loan was taken up for the purpose of buying a vehicle. He took the loan of Rs. 179,055 for which he had to pay Rs. 9,251 as annuity. He had to pay this amount in 24 instalments. The loan was taken 1.5 years ago and was terminated recently.

Let us analyze the loan no. 1038818 which was taken up by the applicant: -

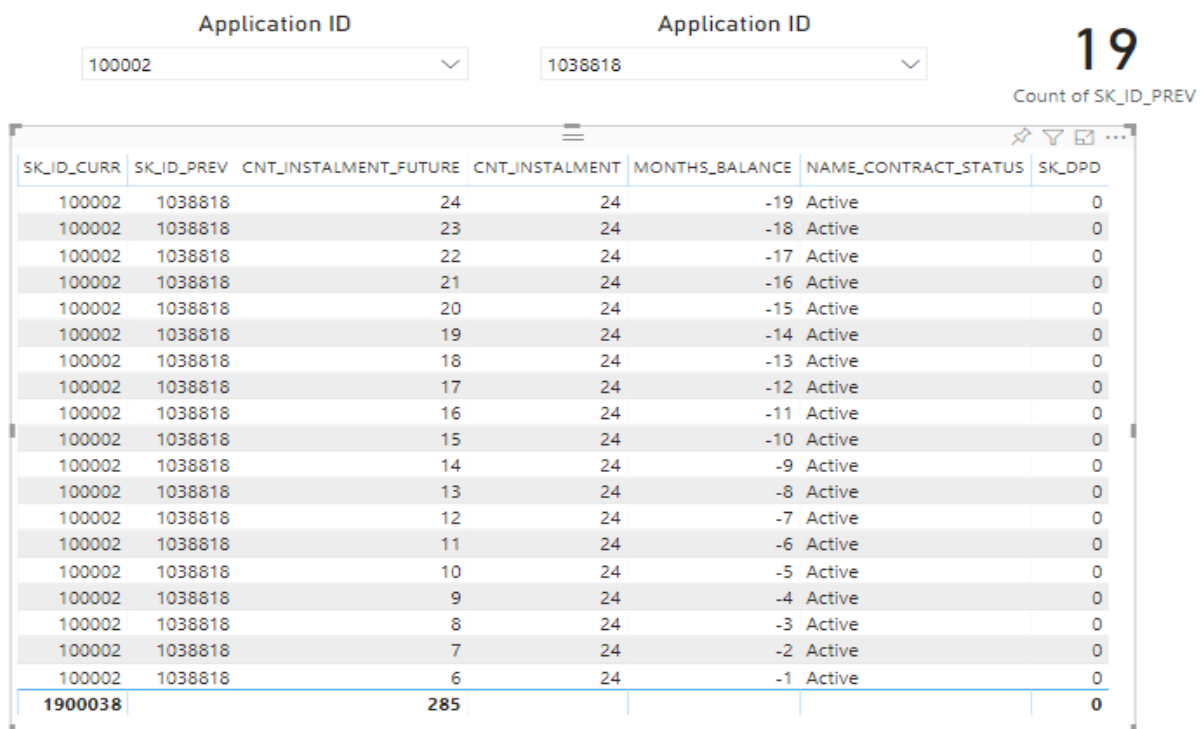


Figure 29: Applicant No. 100002- POS Loan Repayment

Here we see that the applicant is having 24 future instalments to pay in 19 months balance. Though the applicant seems to be paying the instalments regularly, he/she has 6 future instalments to pay in a moth which sums up to be 6*9,251 which is equal to Rs. 55,506.

After analyzing the behavior of the applicant, we see that due to the debt not being paid in the last month, the applicant would default in the future too. This factor i.e., how is the behavior of applicant seems to be important while analyzing and classifying whether applicant would default or not.

IV. MODEL BUILDING AND FEATURE IMPORTANCE

1). Evaluation Metrics

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. So, let’s talk about those four parameters first.

		Predicted Class	
		Class = Yes	Class = No
Actual Class	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Figure 30: Confusion Matrix

Accuracy, Precision, Recall & F1 Score

True positive and true negatives are the observations that are correctly predicted and therefore shown in green. We want to minimize false positives and false negatives so they are shown in red colour. These terms are a bit confusing. So let’s take each term one by one and understand it fully.

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells us the same thing.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. False positives and false negatives, these values occur when our actual class contradicts with the predicted class.

False Positives (FP) – When actual class is no and predicted class is yes.

False Negatives (FN) – When actual class is yes but predicted class in no.

Once we understand these four parameters then we can calculate Accuracy, Precision, Recall and F1 score.

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when we have

symmetric datasets where values of false positive and false negatives are almost same. Therefore, we have to look at other parameters to evaluate the performance of our model.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP + FP}$$

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

$$Recall = \frac{TP}{TP + FN}$$

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if we have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

$$F1\ Score = \frac{2 * (Recall * Precision)}{(Recall + Precision)}$$

In our case we prefer precision, recall and F1 score over accuracy.

2 Feature Selection

As the data is having a lot of features, we are going to select a subset of important features, using which we will proceed with applying different anomaly detection methods.

Using Embedded Techniques

Embedded feature selection help in identifying the most significant features so that our subset of feature consists of important feature only.

Considering only training data and the historical data of a customer applying for loan. We replaced null numerical values by mean, normalized the data to remove skewness in the data and null categorical values by a new category (“No Data”) then we encoded categorical variable by label encoding.

For selection a decision tree approach is used which is a flowchart-like structure in which each internal node represents a “test” on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we

want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information. For each node of the tree, the information value measures how much information a feature gives us about the class. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.

Information theory is a measure to define this degree of disorganization in a system known as Entropy. If the sample is completely homogeneous, then the entropy is zero and if the sample is equally divided (50% – 50%), it has entropy of one.

Entropy can be calculated using formula: -

$$\text{Entropy} = -p \log_2 p - q \log_2 q$$

Here p and q are the probability of success and failure respectively in that node. Entropy is also used with categorical target variables. It chooses the split which has lowest entropy compared to parent node and other splits. The lesser the entropy, the better it is.

Steps to calculate entropy for a split:

- Calculate entropy of parent node
- Calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split.

We can derive information gain from entropy as 1- Entropy

We run XGBoost and Random Forest models to classify whether the customer will default or not.

Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Here “boosting” is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made

Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

This approach supports both regression and classification predictive modeling problems.

We also find out Important features using XGBoost `plot_importance ()`

Now running the models on Under Sampled Data we have 18513 observations we get the following top 10 features: -

Feature importance is calculated: either “weight”, “gain”, or “cover” default is weight

”weight” is the number of times a feature appears in a tree

”gain” is the average gain of splits which use the feature

”cover” is the average coverage of splits which use the feature where coverage is defined as the number of samples affected by the split.

Firstly, here we have asymmetrical distribution of data i.e., our data is unbalanced in order to train models on unbalanced data we can use Resampling techniques i.e., either use up-sampling (Over sampling) or down-sampling technique.

After resampling we have an equal ratio of data points for each class.

Running XGBoost and Random Forest again with the balanced training data.

Using filter-based methods

Filter methods are generally used as a pre-processing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. These are like wrapper methods where subset of features are created. But unlike in wrapper methods, where each subset is tested using different models, here a much simpler filter is used. The ones used in our project are:-

Pearson’s Correlation: It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1. Pearson’s correlation is given as:

$$\rho_{x,y} = \frac{\text{cov}(X,Y)}{\sigma_x \sigma_y}$$

Chi-Square: It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

Before using this method, we need to prepare our data at some level to impute missing values, encoding our categorical variables and conducting other such data preparation activities. Once our data is prepared, we make use of our filter methods.

Initially our data has 226 features. Now applying various ML algorithms to know which will be the best using cross validation, can become computationally very expensive for this data. So, it’s better to select those features that are the most suitable in creating our prediction model. We will select 100 features from 226 features in our data.

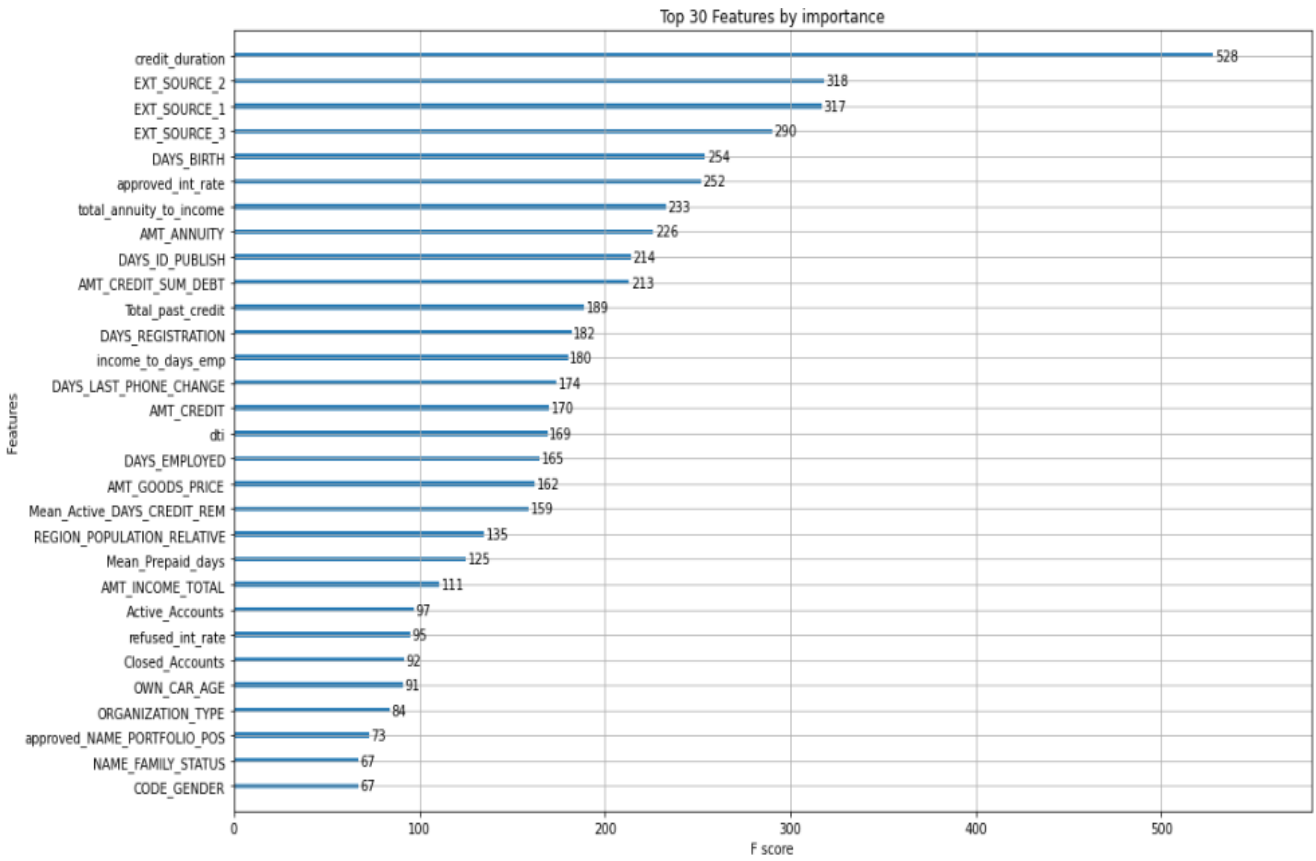


Figure 31: Top 30 Important features from XGBoost

We ran our code and found the following outcomes in the given screenshot below: -

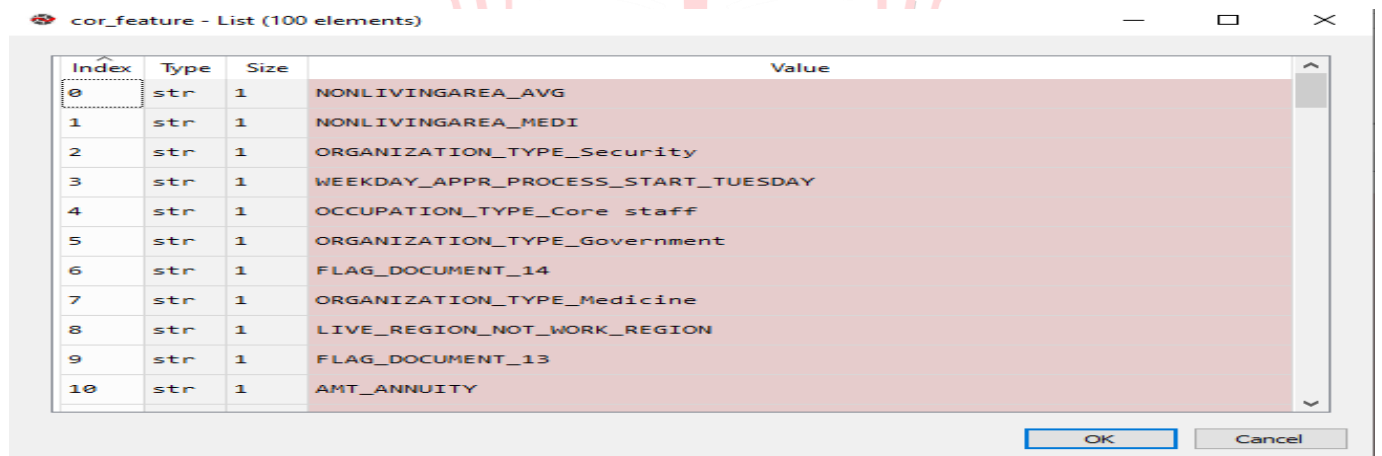


Figure 32: Significant features from Pearson's Correlation



Figure 33: Significant Features from Chi square Test

We will extend our feature selection by only choosing those features that were common in both the filter method's output. In this way, we will have less than 100 features for fitting various models on our data.

V. FEATURE ENGINEERING

Feature engineering, also known as feature creation, is the process of constructing new features from existing data to train a machine learning model. Typically, it is a manual process which relies on a person's intuition, domain knowledge and data manipulation. This process however is very tedious and energy consuming and the final feature set can be limited by both human subjectivity and time.

Feature engineering requires extracting the relevant information from the data and getting it into a single table. In many cases the data is spread across multiple tables. At the end of feature engineering, each client will still have only a single row, but with many more columns capturing information from the other data tables.

There can be two operations for feature creation: Transformation and Aggregation

A transformation acts on a single table by creating new features out of one or more of the existing columns.

Aggregations are performed across tables, and use a one-to-many relationship to group observations and then calculate statistics.

Below are some methods we have used to identify and create features that would increase the prediction power of the model.

1) Credit Default analysis in terms of Financial Ratios

Mortgage default is triggered by negative home equity which tends to occur for a particular combination of the several shocks that the household faces: house price declines in a low inflation environment with large nominal mortgage balances outstanding.

Mortgage origination such as loan-to-value (LTV), loan-to-income (LTI), and mortgage-payments-to-income (MTI) act default probabilities.

Mortgage payment measures the initial house affordability generally banks in India have limit of .10 above that value it is consider risky to approve loan

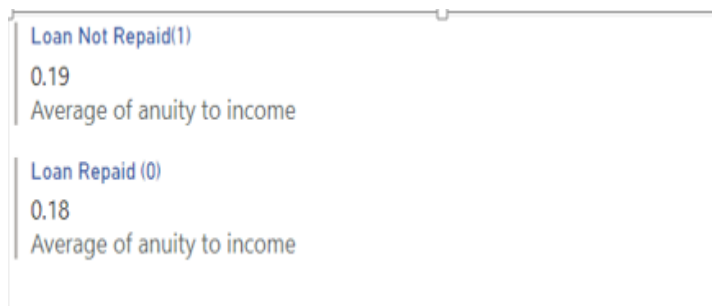


Figure 34: Average annuity to income

As we can see in the figure annuity to income ratio of the defaulters is higher (0.19) as compare to non-defaulters of loan (0.18). So from this analysis it can be concluded that if

annuity to income ratio is more than 0.18 then chances of default are higher.

The LTV ratio measures the household is initial equity stake, while LTI and MTI are measures of initial mortgage affordability. Our model allows us to understand the channels through which LTV and initial mortgage affordability ratios affect mortgage default. A higher LTV ratio (equivalently, smaller down payment) increases the probability of negative home equity and mortgage default, an effect that has been documented empirically by Schwartz and Torous (2003) and more recently by Mayer, Pence, and Sherlund (2009). The unconditional default probabilities predicted by our model become particularly large for LTV ratios in excess of 110 percent.

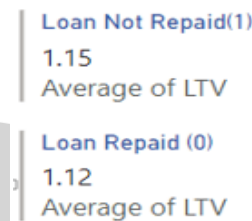


Figure 35: Average loan to value

As we can see in the figure Loan to value ratio of the defaulters is higher (1.15) as compare to non-defaulters of loan (1.12). So from this analysis it can be concluded that if loan to value ratio is more than 1.12 then chances of default are higher.

The LTI ratio affects default probabilities through a different channel. A higher initial LTI ratio does not increase the probability of negative equity; however, it reduces mortgage affordability making borrowing constraints more likely to bind. The level of negative home equity that triggers default becomes less negative, and default probabilities accordingly increases

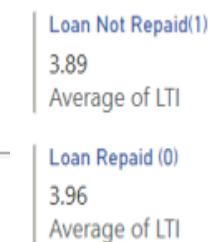


Figure 36: Average loan to income

As we can see in the figure Loan to income ratio of the defaulters is lower (3.89) as compare to non-defaulters of loan (3.96). Although Average LTI of defaulters should be higher this is because of imbalance data or because of willful defaulters.

2) Feature Engineering

Automated feature engineering can be used to create many candidates feature out of a dataset from which the best can be selected and used for training the model.

Feature-tools (an open-source Python library for automatically creating features out of a set of related tables) can automatically perform transformations and aggregations across multiple tables and combine the resulting data into a single table. It is based on a method called "deep feature synthesis". DFS stacks multiple transformation and aggregation operations to create features from data spread across many tables. And to formalize a relationship in feature-tools, we only need to specify the variable that links two tables together. Below is a list of some of the feature primitives:

name	type	description
num_true	aggregation	Finds the number of 'True' values in a boolean.
percent_true	aggregation	Finds the percent of 'True' values in a boolean feature.
time_since_last	aggregation	Time since last related instance.
num_unique	aggregation	Returns the number of unique categorical variables.
avg_time_between	aggregation	Computes the average time between consecutive events.
all	aggregation	Test if all values are 'True'.
min	aggregation	Finds the minimum non-null value of a numeric feature.
mean	aggregation	Computes the average value of a numeric feature.
seconds	transform	Transform a Timedelta feature into the number of seconds.
second	transform	Transform a Datetime feature into the second.
and	transform	For two boolean values, determine if both values are 'True'.
month	transform	Transform a Datetime feature into the month.
cum_sum	transform	Calculates the sum of previous values of an instance for each value in a time-dependent entity.
percentile	transform	For each value of the base feature, determines the percentile in relation
time_since_previous	transform	Compute the time since the previous instance.
cum_min	transform	Calculates the min of previous values of an instance for each value in a time-dependent entity.

Figure 37: Feature Primitives

A deep feature is simply a feature made of stacking multiple primitives and DFS is the name of process that makes these features. The depth of a deep feature is the number of primitives required to make the feature. Automated feature engineering has solved one problem, but created another: too many features. As the number of features increases it becomes more and more difficult for a model to learn the mapping between features and targets.

To overcome this curse of dimensionality we apply feature selection techniques to reduce the number of manual engineered features from the model. We will use three methods for feature selection:

1. Remove collinear features
2. Remove features with greater than a threshold percentage of missing values
3. Keep only the most relevant features using feature importance from a model

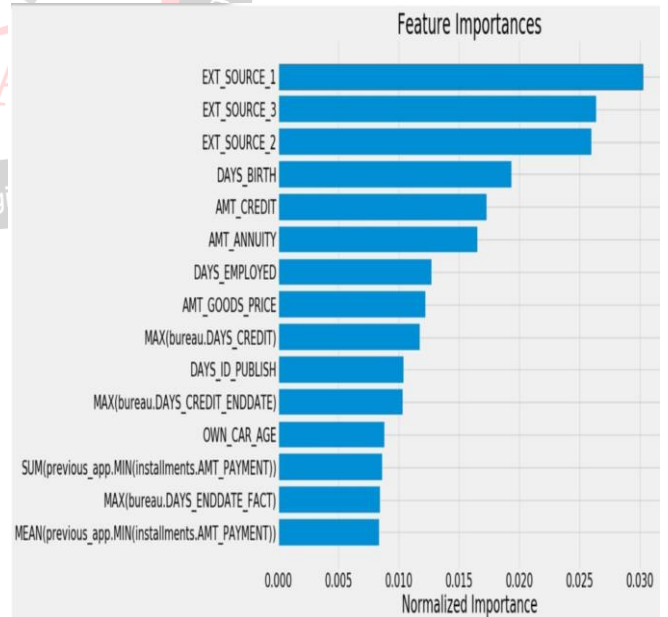


Figure 38: Feature Primitives

VI. MODEL SELECTION

We approached the problem of Credit Default from two separate angles: -

- Anomaly Detection Approach

- Classification Approach

Anomaly detection

- One Class SVM
- GMM

One Class SVM

A One-Class Support Vector Machine is an unsupervised learning algorithm that is trained only on the ‘normal’ data, in our case the non-default observations. It learns the boundaries of these points and is therefore able to classify any points that lie outside the boundary, i.e., outliers.

For this, a random sample of 5000 observations were taken with almost 10% of the data being default observations.

The nu parameter should be the proportion of outliers you expect to observe, the gamma parameter determines the smoothing of the contour lines.

After trying visualizations on various gamma values, the below given diagram is the closest we could get between the actual visualization and the predicted visualization. The gamma value used here is 0.00000000000001.

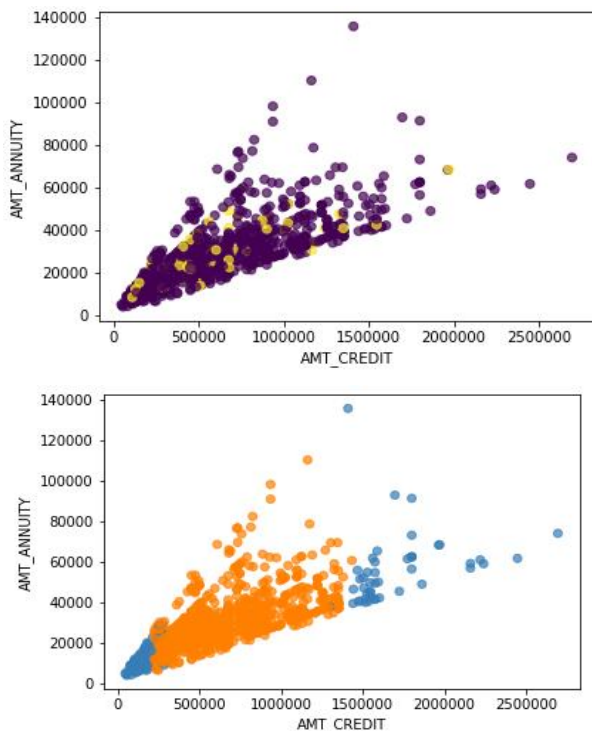


Figure 39: Actual vs Predicted Visualization using One Class SVM

We could clearly see that One class SVM isn’t able to draw boundaries well enough for our data to be used for prediction. There’s a significant difference between the predicted and the actual visualization.

Gaussian Mixture Models

When the number of observations in one class is much more than the other, it is difficult to train a vanilla CNN classifier.

The CNN classifier may consider that all observations are from the main class to achieve high accuracy.

One way to handle this problem is by using oversampling or down sampling to make data balanced. Also, adjusting class weights to force the classifier to handle data in the rare class is also a great idea.

However, using the above methods may sometimes cause model over fitting when data is extremely imbalanced. Therefore, we’ll look at another method, which is called anomaly detection.

In anomaly detection, we try to identify observations that are statistically different from the rest of the observations.

We will assume the observations in the main class as normal data, and use only these data to train our model. Then, we are able to predict whether a new observation is normal.

Gaussian Mixture Model which is the Unsupervised Clustering approach. Here we take the data for top 20 features to build the model. In this approach, unlike K-Means we fit ‘k’ Gaussians to the data.

Then we find the Gaussian distribution parameters like mean and Variance for each cluster and weight of a cluster. Finally, for each data point, we calculate the probabilities of belonging to each of the clusters.

Using GMM’s score_samples function, we can easily compute the likelihood of data. Assuming threshold as the interquartile range of the data, we can predict our testing data.

Then, we use the trained GMM to compute the likelihood of results. Finally, we can detect anomaly if the observation’s likelihood does not lie in the inter quartile range.

To set the number of components for the model we use Bayesian information criterion (BIC)

This criterion gives us an estimation on how much is good the GMM is in terms of predicting the data we actually have. The lower is the BIC, the better is the model to actually predict the data we have, and by extension, the true, unknown, distribution. We decided to take 9 components with the help of the following graph.

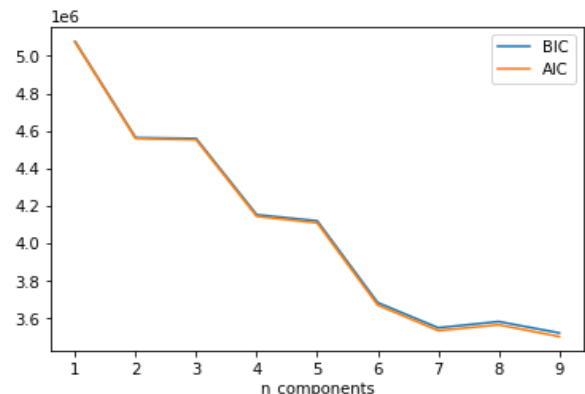


Figure 40: AIC BIC Graph to decide no. of components

Result

Now after we have trained a GMM model, we have the following results:

	GMM
Accuracy	0.140855
Precision	0.831142
Recall	0.140855
F1 Score	0.140855

Classification Method

In the traditional sense, Credit Default is seen as a binary classification problem. The data on which the model needs to run, consists of more than a million rows. We will be using tree-based models which are known for their fast computation and easy interpretation.

We used Gradient Boosting Decision Trees (GBDT) which is an ensemble technique. They involve multiple decision trees (weak learners) to be trained iteratively such that the result of the final model is the addition of multiple decision trees.

Among the Gradient Boosting Techniques, we had applied XGBoost and Light GBM technique, for which we had compared the performance on our data.

Hyper-Parameter Tuning

The different parameters which we used to tune and improve performance of the model, as the default values might not always work and might under/over-fit be depending on the dataset.

Below is a list of hyper parameters which had been tuned, these parameters improved the performance of the model.

num_leaves: This is the main parameter to control the complexity of the tree model. When trying to tune the num_leaves, we should let it be smaller than 2^{max_depth}. If we keep on increase this value, there are chances of overfitting

min_data_in_leaf: Its optimal value depends on the number of training samples and num_leaves. Setting it to a large value can avoid growing too deep a tree, but may cause under-fitting. In practice, setting it to hundreds or thousands is enough for a large dataset.

max_depth: It is used to limit the tree depth, more the depth more are the chance of overfitting

learning_rate: In gradient boosting, this is used to dampen the effect of each additional tree to the models

XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all

other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now.

The algorithm differentiates itself in the following ways:

Applications: Can be used to solve regression, classification, ranking, and user-defined prediction problems.

Portability: Runs smoothly on Windows, Linux, and OS X.

Languages: Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.

Cloud Integration: Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements

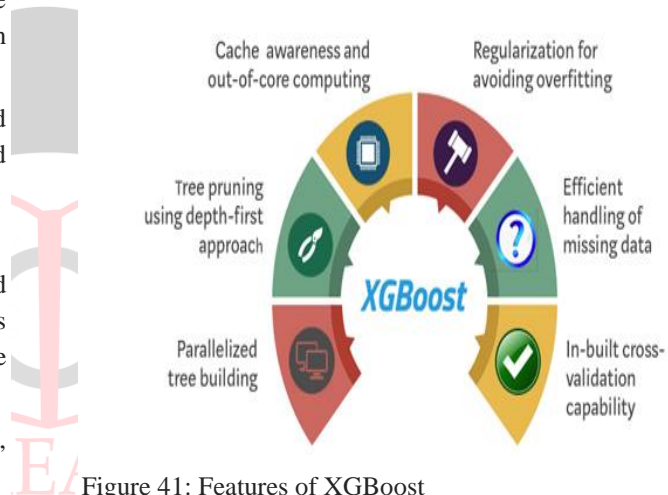


Figure 41: Features of XGBoost

System Optimization:

Parallelization: XGBoost approaches the process of sequential tree building using parallelized implementation. This is possible due to the interchangeable nature of loops used for building base learners; the outer loop that enumerates the leaf nodes of a tree, and the second inner loop that calculates the features. This nesting of loops limits parallelization because without completing the inner loop (more computationally demanding of the two), the outer loop cannot be started. Therefore, to improve run time, the order of loops is interchanged using initialization through a global scan of all instances and sorting using parallel threads. This switch improves algorithmic performance by offsetting any parallelization overheads in computation.

Tree Pruning: The stopping criterion for tree splitting within GBM framework is greedy in nature and depends on the negative loss criterion at the point of split. XGBoost uses 'max_depth' parameter as specified instead of criterion first, and starts pruning trees backward. This 'depth-first'

approach improves computational performance significantly.

Hardware Optimization: This algorithm has been designed to make efficient use of hardware resources. This is accomplished by cache awareness by allocating internal buffers in each thread to store gradient statistics. Further enhancements such as ‘out-of-core’ computing optimize available disk space while handling big data-frames that do not fit into memory.

Algorithmic Enhancements:

Regularization: It penalizes more complex models through both LASSO (L1) and Ridge (L2) regularization to prevent overfitting.

Sparsity Awareness: XGBoost naturally admits sparse features for inputs by automatically ‘learning’ best missing value depending on training loss and handles different types of sparsity patterns in the data more efficiently.

Weighted Quantile Sketch: XGBoost employs the distributed weighted Quantile Sketch algorithm to effectively find the optimal split points among weighted datasets.

Cross-validation: The algorithm comes with built-in cross-validation method at each iteration, taking away the need to explicitly program this search and to specify the exact number of boosting iterations required in a single run.

Result

Now after we have trained XGBoost model and having done hyper-parameter tuning (learning rate = 0.05, max_depth = 20, n_estimators = 200), we have the following results:

	XGBoost
Accuracy	0.692340931
Precision	0.696381289
Recall	0.681936041
F1 Score	0.689082969

Light GBM

Though XGBoost seemed to be the go-to algorithm, LightGBM which was released from Microsoft, this algorithm has been claimed to be more efficient (better predictive performance for the same running time) than XGBoost.

In addition to the benefits provided by XGBoost, LightGBM provides two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to deal with large number of data instances and large number of features respectively. The experimental results are consistent with the theory and show that with the help of GOSS and EFB, LightGBM can significantly

outperform XGBoost in terms of computational speed and memory consumption.

Gradient-based One-Side Sampling (GOSS)

This is a method that is employed exclusively in LightGBM. The essential observation behind this method is that not all data points contribute equally to training; data points with small gradients tend to be better trained (close to a local minima). This means that it is more efficient to concentrate on data points with larger gradients.

The most straightforward way to use this observation is to simply ignore data points with small gradients when computing the best split. However, this has the risk of leading to biased sampling, changing the distribution of data. For instance, if data that belonged to the "young" age group tended to be less well trained, the sampled data will have a much younger age distribution. This means that the split is likely to be younger than the optimal value.

In order to mitigate this problem, LightGBM also randomly samples from data with small gradients. This results in a sample that is still biased towards data with large gradients, so LightGBM increases the weight of the samples with small gradients when computing their contribution to the change in loss (this is a form of importance sampling, a technique for efficient sampling from an arbitrary distribution).

Exclusive Feature Bundling (EFB)

This is a method introduced in LightGBM that also takes advantage of the sparsity of large datasets. The essential observation behind this method is that the sparsity of features means that some features are never non-zero together. For instance, the words "Python" and "protein" might never appear in the same document in the data. This means that these features can be "bundled" into a single feature without losing any information. Suppose the tf-idf score for "Python" ranges from 0 to 10 and the tf-idf score for "protein" ranges from 0 to 20. In this case, the feature would range from 0 to 30, and can be converted back to the original tf-idf scores.

Result

Now after we have trained a Light GBM model and have done hyper-parameter tuning, we have the following results:

	Light GBM
Accuracy	0.718340931
Precision	0.698574537
Recall	0.683664649
F1 Score	0.686110809

After getting the results we can analyze the most important features according to Light GBM.

The below diagram shows the top 15 features, with most important one being represented in the bottom.

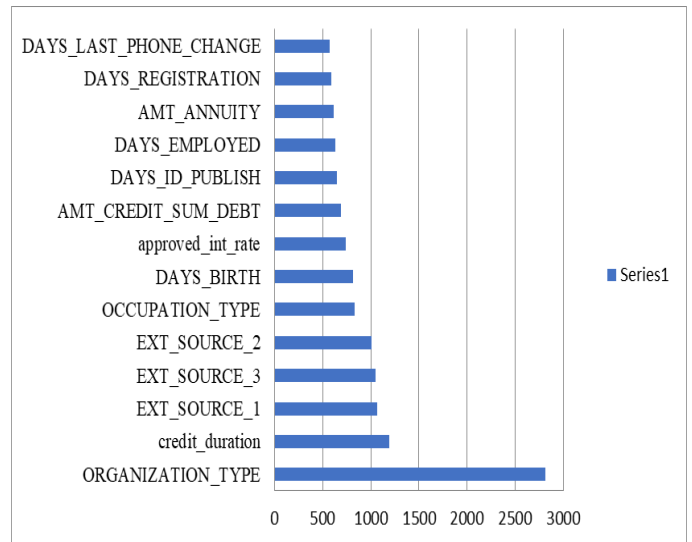


Figure 42: Feature importance using Light GBM
Comparing All Results

	GMM	XGBoost	Light GBM
Accuracy	0.140855	0.692340931	0.718340931
Precision	0.831142	0.696381289	0.698574537
Recall	0.140855	0.681936041	0.683664649
F1 Score	0.140855	0.689082969	0.686110809

We find that Light GBM gives us the best overall accuracy while precision recall and f1 score of XGBoost and Light GBM are similar.

VII. UI CREATION

As we have seen Light GBM provides the best performance with respect to speed as well as accuracy. We have built a UI application using Light GBM as our base model, allowing the end user to enter the customer details and receive a probability score representing the chances of a customer defaulting their loan.

To create a UI, we will use Tkinter, which is a standard Python interface to the Tk GUI toolkit shipped with Python.

Using the UI user can enter values such as Applicant Name, Financial Details, Demographics, and Credit Details and get an output probability for default for that customer.

Using this feature any Lending Organization can predict how that customer is going to behave. According to the perceived risk, it can plan out what measure needs to be taken in order to ensure timely payments of annuity.

For different probabilities different measures could be: -

- Probability 0-30%: Reminders in form of SMS
- Probability 30-50%: Reminders in form of Calls and SMS
- Probability 50-85%: Reminders in form of Calls and in-person meeting (if there is a delay)
- Probability 85-100%: To be prepared for strict measures for collections

Let’s consider two consumers who have taken up loan from Home Credit: -

The first customer is a 29 years old male, who took up loan from Home Credit Group

The details of the consumer and the loan taken are given below: -

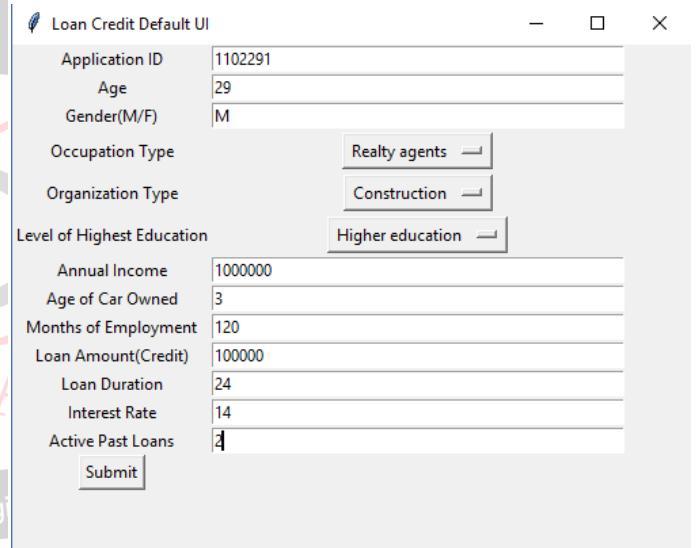


Figure 43: Screenshot of the UI – 1st Customer

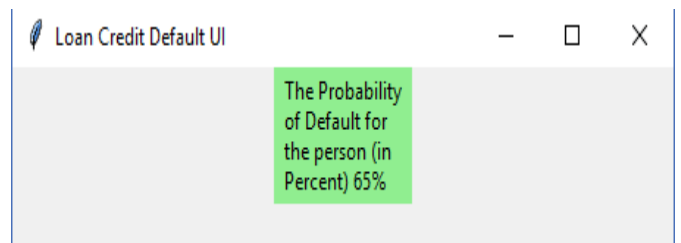
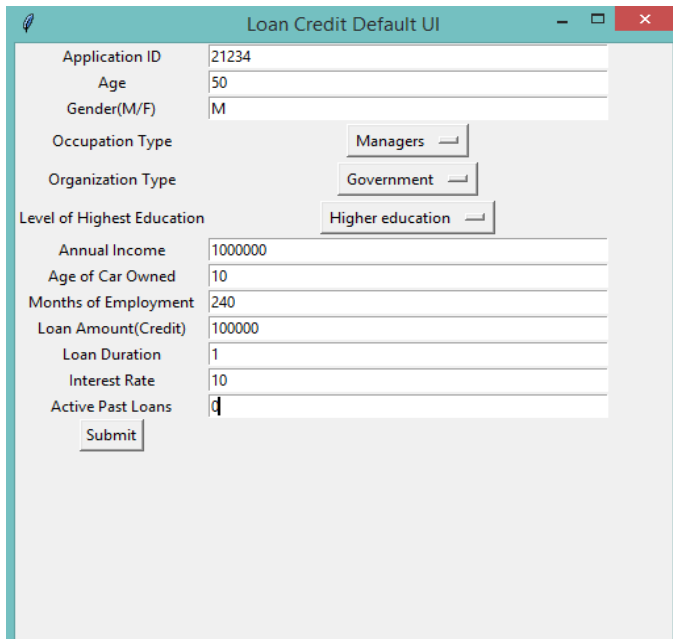


Figure 44: Screenshot of the probability of default - 1st Customer

The second customer is a 50 years old male, who took up loan from Home Credit Group

The details of the consumer and the loan taken are given below: -



Application ID	21234
Age	50
Gender(M/F)	M
Occupation Type	Managers
Organization Type	Government
Level of Highest Education	Higher education
Annual Income	1000000
Age of Car Owned	10
Months of Employment	240
Loan Amount(Credit)	100000
Loan Duration	1
Interest Rate	10
Active Past Loans	0

Figure 45: Screenshot of the UI - 2nd Customer

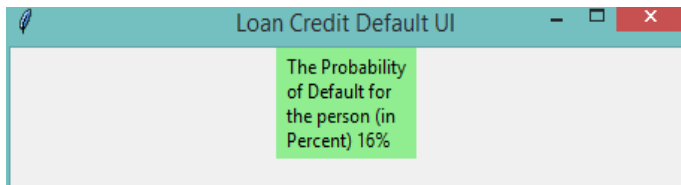


Figure 46: Screenshot of the probability of default -2nd Customer

Here we can see that our software is able to identify the risk associated with each consumer loan, giving us the probability score of the person defaulting.

VIII. CONCLUSION

As a part of this project, we have been able to answer interesting questions relating to consumer behavior when they take a loan from any bank. This was done using extensive EDA and developing an Applicant Analysis tool using Power BI visualization tool.

In addition to EDA we applied different machine learning techniques like Anomaly Detection and Classification techniques to our data and compared different models and their performance.

After creating a model, we choose the best model for creating a User Interface which allows the end user to input consumer details and get a probability of default for that individual. This application can help the bank identify the people who might default in the future and take necessary actions.

References

- [1] Bose, I., & Mahapatra, R. K. (2001). Business data mining — a machine learning perspective. *Information & Management*, 39(3), 211–225
- [2] Nupura Torvekar, Pravin S. Game (2019); Predictive Analysis of Credit Score for Credit Card Defaulters; *International Journal of Recent Technology and Engineering*; Volume 7; No. 5S2; Pg 283-286
- [3] Chih-Fong Tsai , Ming-Lun Chen(2010).Credit rating by hybrid machine learning techniques. *Applied Soft Computing* 10 (2010) 374–380
- [4] Li Ying (2018); Research on bank credit default prediction based on data mining algorithm; *The International Journal of Social Sciences and Humanities Invention*; Volume 5; No. 06; Pg 4820-4823
- [5] Li, S., Shiue, W., & Huang, M. (2006). The evaluation of consumer loans using support vector machines. *Expert Systems with Applications*, 30(4), 772–782
- [6] U Bhuvaneshwari, Sharon Sophia (2019); A Predictive Analysis of Credit Risk Evaluation and the Quality Decision Making using Different Predictive Models; *International Journal of Recent Technology and Engineering*; Volume 8; No. 1; Pg 1965-1969
- [7] Zhu, Y., Xie, C., Wang, G. and Yan, X. (2016). Comparison of individual, ensemble and integrated ensemble machine learning methods to predict China’s SME credit risk in supply chain finance. *Neural Computing and Applications*, 28(S1), pp.41-50.
- [8] Artem Bequé, Stefan Lessmann; Extreme learning machines for credit scoring: An empirical evaluation; *School of Business and Economics, Humboldt-University of Berlin, Unter-den-Linden 6, 10099 Berlin, Germany*
- [9] XiaoyunLiu, James Huang; Genetic Algorithm-based Feature Selection method for Credit Risk Analysis; *2012 2nd International Conference on Computer Science and Network Technology*
- [10] Khandani, A., Kim, A. and Lo, A. (2019). Consumer Credit-Risk Models Via Machine-Learning Algorithms. [online] Dspace.mit.edu. Available at: <https://dspace.mit.edu/openaccess-disseminate/1721.1/66301> [Accessed 12 Nov. 2019]
- [11] Julapa Jagtiani, Catharine Lemieux (2018); The Roles of Alternative Data and machine Learning in Fintech Lending: Evidence from the LendingClub consumer platform; Working Paper: Research Department, Federal reserve bank of Philadelphia
- [12] Dawn Iris Calibo1, Melvin A. Ballera(2019) Variable Selection for Credit Risk Scoring on Loan Performance Using Regression Analysis .*IEEE 4th International Conference on Computer and Communication Systems*.
- [13] Tony Van Gestel, Bart Baesens, Peter Van Dijke, Johan A. K. Suykens, Joao Garcia, Thomas Alderweireld (2005); Linear and non-linear credit scoring by combining logistic regression and support vector machines; *Journal of Credit Risk*; Volume 1; No. 4; Pg 31-60
- [14] Outlier Detection with One-Class SVMs - Carl Dawson
- [15] <https://towardsdatascience.com/outlier-detection-with-one-class-svms-5403a1a1878c>