GREAT LAKES
INSTITUTE OF MANAGEMENT, GURGAON
Global Mindset - Indian Roots

PGD-DSE (November-19)
Final Capstone Report

GROUP - E

FORECLOSURE PREDICTION BASED ON CUSTOMER DEMOGRAPHICS AND TRANSACTION BEHAVIOUR

Submitted By:

ANSHUMAN SILORI

LOHITAKSHA MOR

ARPIT SHARMA

BHAWNA KAPOOR

Under the guidance of:

DR. DIPANJAN GOSWAMI

INTRODUCTION

DATA DESCRIPTION

EXPLORATORY DATA ANALYSIS

ARCHITECTURE

TENTATIVE LIST OF ALGORITHMS USED & INITIAL APPROACH

RESULTS AND COMPARISON STUDY

IMPLICATION
LIMITATIONS
CONCLUSION
APPENDIX

1. INTRODUCTION

1.1 FORECLOSURE

Foreclosure refers to a legal process in which a bank attempts to recover the balance of a loan from a borrower who has stopped making payments to the bank by forcing the sale of the asset used as the collateral for loan or repaying the outstanding loan amount in a single payment instead of with EMIs while balance transfer is transferring outstanding Loan availed from one Bank / Financial Institution to another Bank / Financial Institution, usually on the grounds of better service, top-up on the existing loan, proximity of branch, saving on interest repayments, etc.

1.2 AIM OF THE PROJECT

India is a diversified financial sector undergoing rapid expansion, both in terms of strong growth of existing financial services firms and new entities entering the market. The sector comprises commercial banks, insurance companies, non-banking financial companies, co-operatives, pension funds, mutual funds and other smaller financial entities. The financial institutions provides a broad range of financial products and services to a substantial and diversified client base that includes corporations, institutions and individuals. Given the availability of various alternatives across the industry, customer has a propensity to move to another financial institution for Balance Transfer. Losing out on customers on grounds on foreclosure and balance transfer leads to revenue loss. According to the institution acquiring a new customer can cost up to five times more than retaining an existing customer and an increase in customer retention by 5% increases profits up to 25%. The firm is trying to reduce the foreclosures and improve customer retention rate.

1.3 PROBLEM STATEMENT

Predicting foreclosure cases for each of the data points in test dataset and to suggest the effective measure the firm can take to overcome a foreclosure case and enhance its customer retention rate that will yield to more profitability.

2. DATA DESCRIPTION

2.1 DATA SET

We have been provided the following dataset consisting:

- Customer demographics
- Customer transactions (i.e. repayments made by the client)
- List of foreclosed customers
- Test dataset

2.2 DATA MERGING

- The customer transactions datasetis merged with list of foreclosed customer dataseton AGREEMENTID and similarly with test dataset.
- Bothdatasets are then merged with Customer demographics dataset on CUSTOMERID.
- The resulted train dataset and test dataset consists of 19730 and 13163 instances respectively with 50 attributes.
- The two datasets were concat and resulted to 32893 and 50 attributes.

2.3DATASET DISCRIPTION

Below is the detailed description of each of the variable considered for analysis in the initial steps out of the total 53 variables.

FEATURES	DESCRIPTION
LOAN_AMOUNT	The gross/actual loan taken by the customer
NET_DISBURSED_AMNT	The net amount calculated by deducting processing fees, charges and other factors
CURRENT_ROI	Current rate of interest on principal amount
ORIGINAL_ROI	Original rate of interest on principal at the time of loan disbursement
CURRENT_TENOR	Present duration from the disbursement of loan till the last EMI payment of loan amount
ORIGINAL_TENOR	Original duration from the disbursement of loan till the last EMI payment of loan amount
DUEDAY	It is the last day of payment of EMI for the loan
PRE_EMI_DUE_AMT	It is the amount payable before the disbursement of the actual loan amount
PRE_EMI_RECEIVED_AMT	Amount paid by the borrower to the lender as Pre EMI interest amount before the disbursal of actual loan
PRE_EMI_OUTSTANDING_AMT	It the remaining Pre EMI amount that borrower needs to pay to the lender
EMI_DUEAMT	Amount that borrower needs to pay as EMIs on loan amount
EMI_RECEIVED_AMT	Amount received as EMI by the lender
EMI_OS_AMT	It is the remaining EMI amount that borrower needs to pay for previous EMI
EXCESS_AVAILABLE	The amount available to bank as the availability amount of collateral
EXCESS_ADJUSTED_AMT	The amount adjusted in LOAN_AMOUNT by the lender. However, the borrower still pays the entire Loan Amount. This is internal to lender only
BALANCE_EXCESS	The remaining amount calculated as (EXCESS_AVAILABLE - EXCESS_ADJUSTED_AMT)
NET_RECEIVABLE	Amount to be received by lender as per the amount in BALANCE_EXCESS
OUTSTANDING_PRINCIPAL	Principal amount payable by borrower
PAID_PRINCIPAL	Part of loan amount paid by borrower
PAID_INTEREST	Amount paid as interest on loan amount
MONTHOPENING	Total outstanding loan amount payable
LAST_RECEIPT_AMOUNT	Amount received in the latest payment by the borrower
NET_LTV	Loan-to-Value ratio (in percentage)
COMPLETED_TENURE	Duration up to which part or whole of the loan amount
BALANCE_TENURE	Remaining duration to pay the outstanding loan amount
DPD	Days Past Due shows worth of EMI is not paid by the borrower.
FOIR	Fixed Obligation to Income Ratio is the ratio that is used by lenders to give loans.
PRODUCT	The commodity or object on which the loan is given. The products included HL(House Loan), STHL(Short Term House Loan), LAP(Loan Against Property) and STLAP(Short Term Loan Against Property)
FORECLOSURE	Forced payment of Loan initiated by lender after selling the collateral or it is the early payment of loan amount resulting in early closure of all EMIs and full settlement.
AGREEMENTID	Unique Agreement id provided to customer for a sanctioned loan
CUSTOMERID	Unique Customer ID given to each customer
AGE	Age of the customer
SEX	Sex of the customer
MARITAL_STATUS	Marital status of the customer
NO_OF_DEPENDENT	Number of dependents of the customer

GROSS_INCOME	Gross income of the customer
NETTAKEHOMEINCOME	Net take home of the customer
CITY	Name of the city

2.4 TARGET VARIABLE

Foreclosure is thetarget variable and is characterized by '1' or '0' in our data set where '1' represents a foreclosure case and '0' indicates not a foreclosure case.

3. EXPLORATORY DATA ANALYSIS

3.1 INTRODUCTION

EDA is a general approach to explore datasets by means of simple summary statistics and graphic visualizations in order to gain a deeper understanding of the data.

3.2 DATA PREPROCESSING

3.2.1 DESCRIPTIVE STATISTICS:

It is one of significant step in order to draw conclusions about the statistical significance of the features explaining the measure of central tendency and spread of the data. It computes a summary of statistics pertaining to the Data Frame columns.

Features	Frequency	Mean	Standard Deviation	Minimum	IQR 1	Median	IQR 2
LOAN_AMT	5843930	13162295	37532.4	1546335	2664688	5179471	5.67E+08
NET_DISBURSED_AMT	5735300	12835929	37532.4	1532814	2619761	5098352	5.67E+08
CURRENT_ROI	14.68607	2.510337	9.901017	12.48552	14.35835	16.23118	29.96525
ORIGNAL_ROI	14.39668	2.596529	9.651307	12.48552	13.73407	16.16875	29.96525
CURRENT_TENOR	188.0143	56.94906	6	167	180	228	713
ORIGNAL_TENOR	183.3871	44.63828	14	180	180	228	300
PRE_EMI_DUEAMT	50820.66	355246.9	0	3377.428	8698.382	27295.49	31775396
PRE_EMI_RECEIVED_AMT	50565.32	352852	0	3368.042	8671.879	27203.29	31775396
PRE_EMI_OS_AMOUNT	255.3362	13952.82	0	0	0	0	2089129
EMI_DUEAMT	1703949	6707031	0	106241.3	424030.9	1241017	3.94E+08
EMI_RECEIVED_AMT	1674326	6640429	0	105252.4	419018.6	1215418	3.89E+08
EMI_OS_AMOUNT	29622.64	557490.4	0	0	0	0	58995309
EXCESS_AVAILABLE	346419.9	3466616	0	0	253.9835	1575.509	2.84E+08
EXCESS_ADJUSTED_AMT	291821.8	3341665	0	0	0	260.6091	2.84E+08
BALANCE_EXCESS	54598.03	873709.5	0	0	0	12.92003	53005248
NET_RECEIVABLE	-24720.1	907734.4	-5.1E+07	-2.20855	0	0	38643502
OUTSTANDING_PRINCIPAL	5207660	11759531	-0.75065	1424632	2388315	4548341	5.51E+08

PAID_PRINCIPAL	687033.9	27216756	0	10769.17	58135.43	234536.2	4.89E+09
PAID_INTEREST	850540.6	3005344	0	76110.07	243990.6	673394.6	1.85E+08
MONTHOPENING	5396868	11966685	0	1481066	2489023	4760564	5.51E+08
LAST_RECEIPT_AMOUNT	79022.85	709569.7	0.12	10999	19526	38385	84968812
NET_LTV	51.24617	21.20078	0.38	35.12	53.35	66.84	100
COMPLETED_TENURE	15.66841	16.82868	0	3	10	23	98
BALANCE_TENURE	172.3459	61.73766	0	137	177	214	691
DPD	7.443348	67.22668	0	0	0	0	2146
FOIR	22.87216	3151.114	-170.33	0.41	0.52	0.68	547616
MOB	17.07933	17.04018	0	5	12	25	98
AGE	40.84465	9.344151	18	34	40	48	76
NO_OF_DEPENDENT	0.519202	1.097127	0	0	0	0	10
GROSS_INCOME	154447.9	580127.5	0	45038.87	75064.79	128864.5	26917441
PRE_JOBYEARS	4.554526	6.25412	0	0	2	7	42
NETTAKEHOMEINCOME	153637.6	579753.1	0	45038.87	75064.79	128586	26917441

3.2.2 SKEWNESS AND KURTOSIS

Below mentioned variables has comparatively high skewness and kurtosis measure and is handled by applying Standard Scaler, z-score technique.

Features	Skewness	Kurtosis
Average(FOIR)	161.4408	27253.32
Average(PAID_PRINCIPAL)	159.4558	27110.35
%_CHANGE_IN_ROI	97.27011	13568.37
Average(BALANCE_EXCESS)	75.08289	7331.687
Average(EMI_OS_AMOUNT)	56.05292	4119.581
Average(PRE_EMI_OS_AMOUNT)	55.95357	3945.742
Average(EXCESS_ADJUSTED_AMT)	47.01108	3218.793
Average(EXCESS_AVAILABLE)	44.76305	2972.635
Average(PRE_EMI_RECEIVED_AMT)	31.09175	1436.509
Average(PRE_EMI_DUEAMT)	30.93218	1422.563
NETTAKEHOMEINCOME	27.81875	1035.796
GROSS_INCOME	27.77333	1033.347

3.3 FEATURE ENGINEERING

3.3.1 FEATURE EXTRACTION:

Some of the new features listed below are added in order to generalize the model well and improve the interpretability of model.

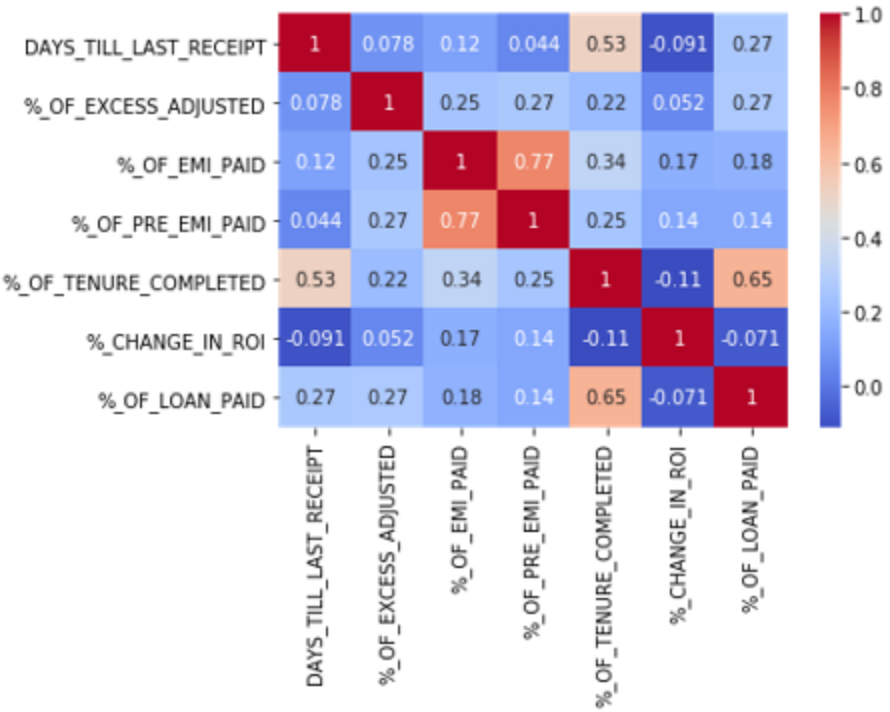
NEW FEATURE	FORMULA EXECUTED	DESCRIPTION
%_OF_LOAN_ PAID	(PAID_PRINCIPAL / LOAN_AMT) * 100	Percentage of loan paid by customer
%_CHANGE_IN_ROI	(CURRENT_ROI - ORIGINAL_ROI) / ORIGINAL_ROI * 100	Percentage of change in rate of interest
%_OF_TENURE_COMPLETED	(COMPLETED_TENURE) / (CURRENT_TENOR)* 100	Percentage of tenure completed
%_OF_PRE_EMI_PAID	(PRE_EMI_RECEIVED_AMT)/(PRE_EMI_DUEAMT+PRE_EMI_RECEIVED_AMT+PRE_EMI_OS_AMOUNT)* 100	Percentage of pre EMI paid
%_OF_EMI_PAID	(EMI_RECEIVED_AMT)/(EMI_DUEAMT+EMI_RECEIVED_AMT+EMI_OS_AMOUNT)	Percentage of EMI paid
%_OF_EXCESS_ADJUSTED	(EXCESS_ADJUSTED_AMT)/(EXCESS_AVAILABLE)	Percentage of excess adjusted
DAYS_TILL_LAST_RECEIPT	LAST_RECEIPT_DATE-AUTHORIZATIONDATE	No. of days passed from authorization date to last receipt date
FIXED_OBLIGATIONS	(FOIR) * (GROSS_INCOME)/30	Fixed Obligation

3.3.2 DESCRIPTIVE STATISTICS

NEW FEATURE	Frequency	Mean	Standard Deviation	Minimum	IQR 1	Median	IQR 2
%_OF_EXCESS_ADJUSTED	34.439066	46.81875	0	0	0	100	100
%_OF_EMI_PAID	40.179926	19.52462	0	50	50	50	50
%_OF_PRE_EMI_PAID	42.036007	18.28068	0	50	50	50	50
%_OF_TENURE_COMPLETED	10.236489	14.16836	0	1.5	5.263158	13.5514	100
%_CHANGE_IN_ROI	2.340835	5.958046	-43.478	0	0	4.683917	60
%_OF_LOAN_PAID	7.618043	18.94101	0	0.515519	2.212324	6.928401	1998.771

3.3.3 HEAT MAP

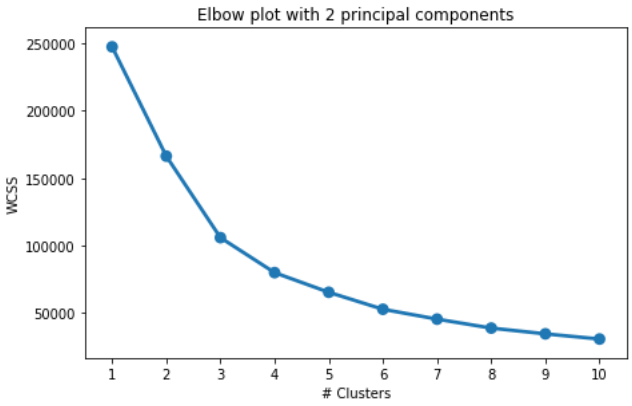
Less correlation can be observed among new features extracted.

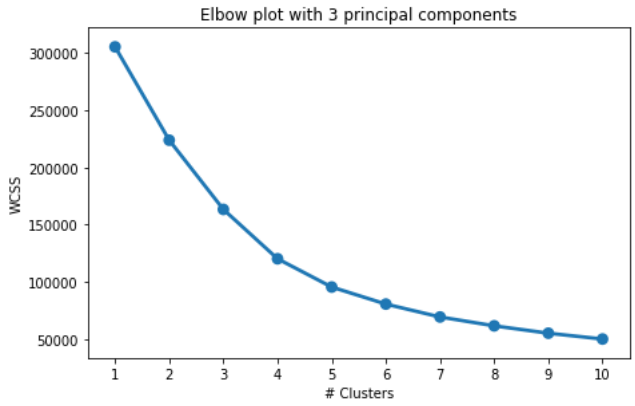


3.4 CLUSTERING FOR EDA

- Clustering has been applied using PCA, to get standardized labels (clusters), getting of which will be time consuming if applied manually. These labels are used to perform binning of variables to understand which type of cases among each variable are in majority among foreclosure.

- Using PCA we get good elbow plot with 2 principal components, clearly showing elbow point at 3 clusters. Same 3 clusters are been applied as labels using KMeans algorithm and EDA has been done w.r.t to those labels.





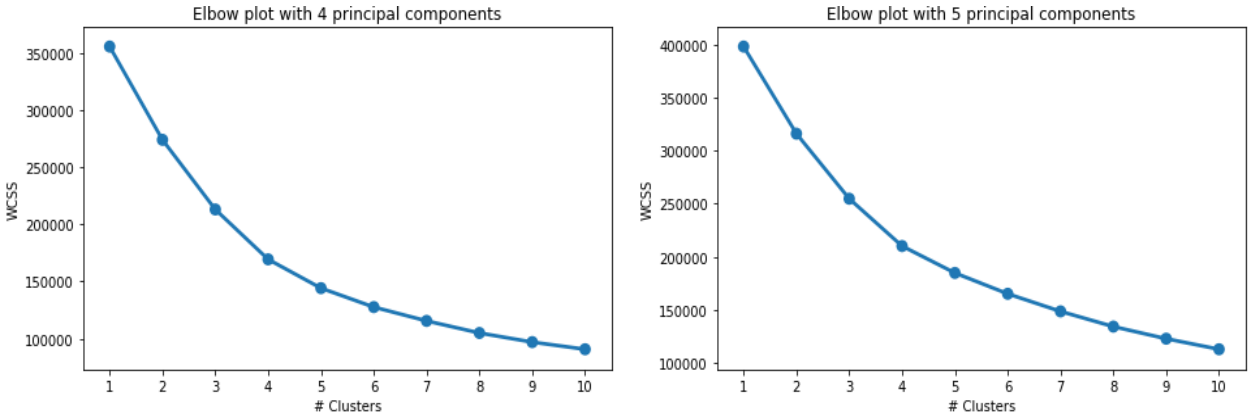


Fig 1 Graph showing elbow plots at different principal components

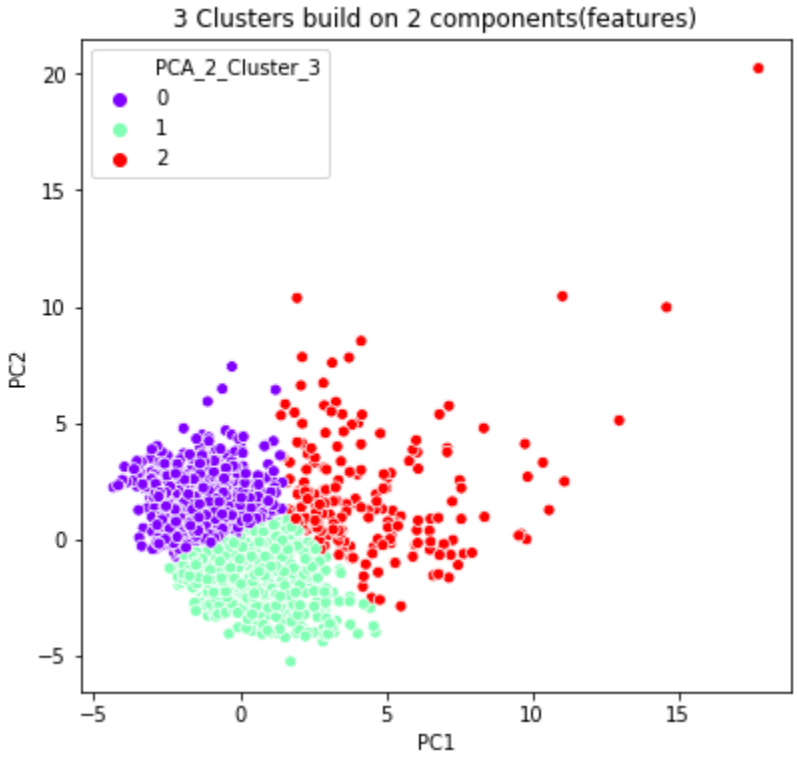


Fig .2 Scatter plot showing 3 clusters distribution with respect to two principal components

- Analysis and descriptive statistics is performed on resulted 3 clusters obtain from kmeans with respect to clusters Some of the derived results are listed below:

Cluster Label	Frequency	Mean	Standard Deviation	Minimum	IQR 1	Median	IQR 2	Maximum
0	718	2.334595	3.894268	0	0.776485	1.496432	2.683814	58.30656
1	878	4.25733	5.476943	0	1.353761	2.576464	4.787977	48.1055
2	199	26.39944	21.67366	0.348612	8.37382	20.13173	38.52673	86.76853

Fig: Descriptive statistics for % of loan paidwrt cluster label

Cluster Label	Frequency	Mean	Standard Deviation	Minimum	IQR 1	Median	IQR 2	Maximum
0	718	1.88301	5.158039	-29.9726	-0.68534	0	5.954198	33.04348
1	878	0.193039	2.336799	-13.8941	0	0	0	14.84375
2	199	4.395607	9.506919	-15.9877	-1.43415	2.671756	6.879559	43.11927

Fig: Descriptive statistics for % change ROIwrt cluster label

Cluster Label	Frequency	Mean	Standard Deviation	Minimum	IQR 1	Median	IQR 2	Maximum
0	718	9.151689	5.688899	0	4.915423	8.245123	12.44688	33.31812
1	878	14.44259	8.11606	0	8.333333	13.52083	18.88889	52.99252
2	199	34.17918	20.25051	1.921506	19.62	30.49626	41.67083	100

Fig: Descriptive statistics for %_of_tenure_completedwrt cluster label

Cluster Label	Frequency	Mean	Standard Deviation	Minimum	IQR 1	Median	IQR 2	Maximum
0	718	138320.8	241842.7	0	49607.88	75290.3	131685	3636129
1	878	107817.9	170425.6	0	53326.78	77303.06	119554.6	4261664
2	199	302705.9	605014.6	0	59918.22	113629	305039.3	4619963

Fig: Descriptive statistics for Gross Income wrt cluster label

3.5 UNIVARIATE, BIVARIATE AND MULTIVARIATE ANALYSIS

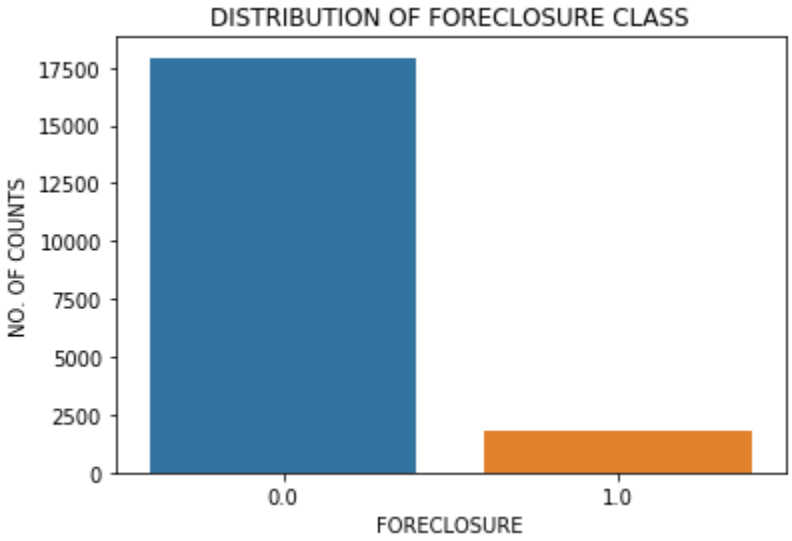
3.5.1 ATTRIBUTE CORRELATION

Through Pearson's correlation metrics, some of the features are observed highly correlated such as:

- CURRENT_TENOR is highly correlated with BALANCE_TENURE (ρ = 0.984895829)
- MONTHOPENING is highly correlated with LOAN_AMT (ρ = 0.9915407892)
- NET_DISBURSED_AMT is highly correlated with MONTHOPENING (ρ = 0.9942027115)
- DAYS_TILL_LAST_RECEIPT is highly correlated with COMPLETED_TENURE (ρ = 0.9694665006)
- EMI_RECEIVED_AMT is highly correlated with EMI_DUEAMT (ρ = 0.9980525048)
- PAID_INTEREST is highly correlated with EMI_RECEIVED_AMT (ρ = 0.953077511)
- FOIR is highly correlated with FIXED_OBLIGATIONS (ρ = 0.9515530014)
- OUTSTANDING_PRINCIPAL is highly correlated with NET_DISBURSED_AMT (ρ = 0.9839242541)
- PRE_EMI_RECEIVED_AMT is highly correlated with PRE_EMI_DUEAMT (ρ = 0.9986526659)

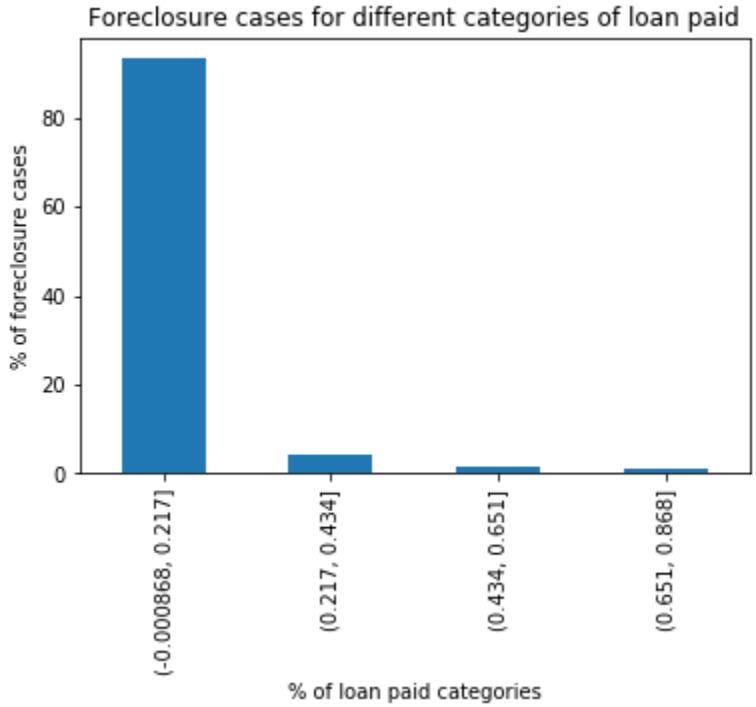
3.5.2DISTRIBUTION OF CLASS IN TARGET VARIABLE

The classes in the Target variables of the dataset are highly imbalanced with 90.9% and 9.01% as '0' and '1' class respectively.

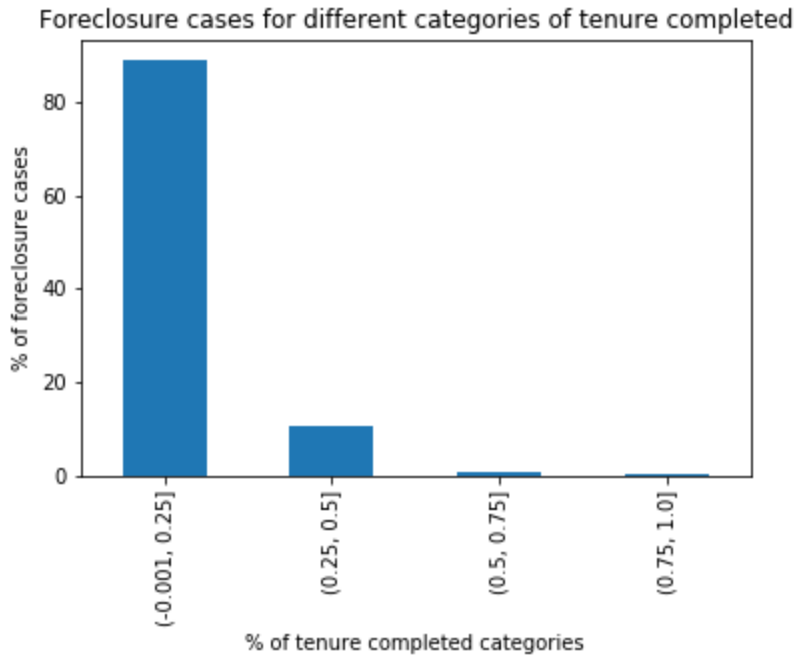


3.5.3 OTHER INFERENCES

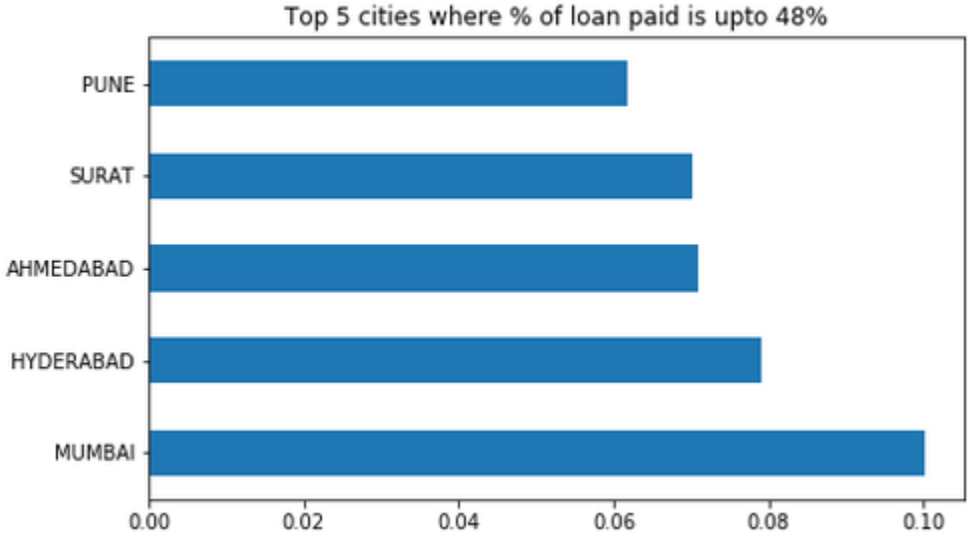
- Around 93% Foreclosure happens at 0-21.6% of loan payment. So, mostly people at initial stages of loan tenure show foreclosure.



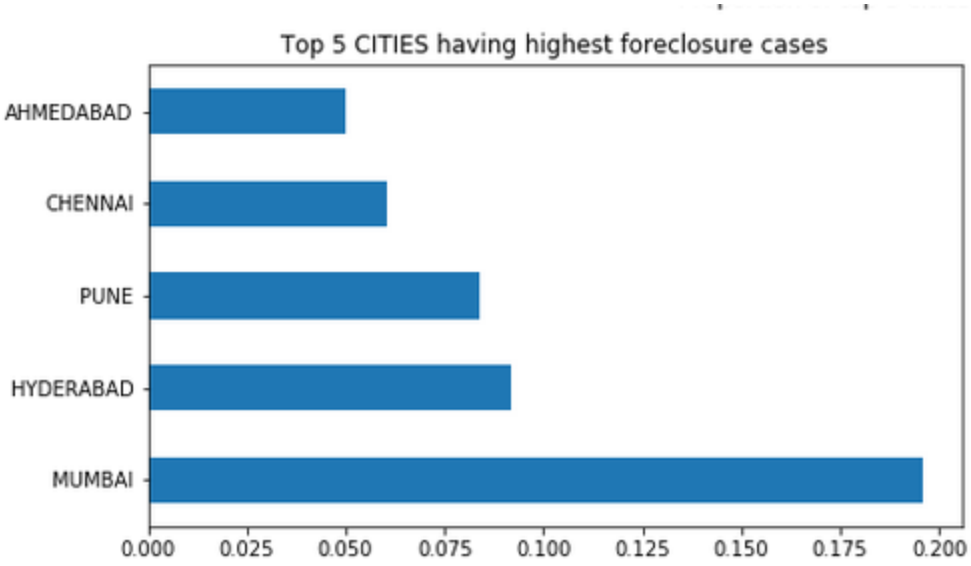
- Maximum Foreclosures cases were observed when percentage of tenure completed is less than 25%



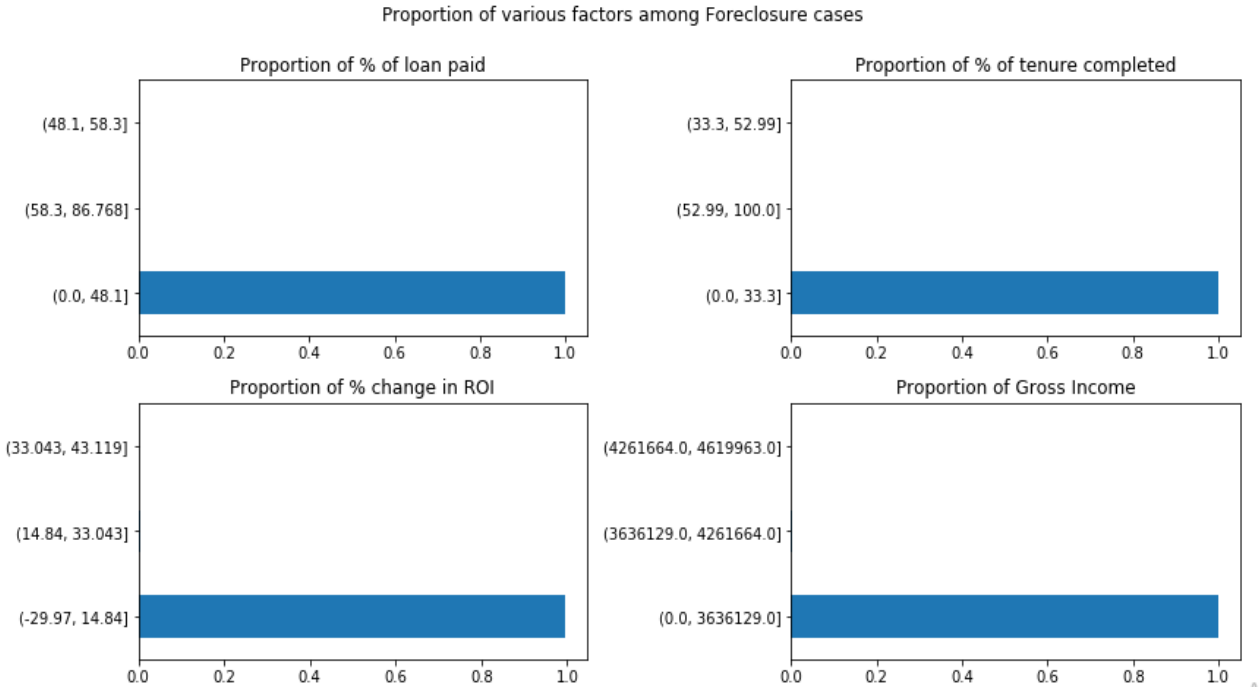
- Mumbai, Hyderabad, Ahmedabad, Surat and Pune are the top 5 cities where percentage of loan paid is up to 48%.



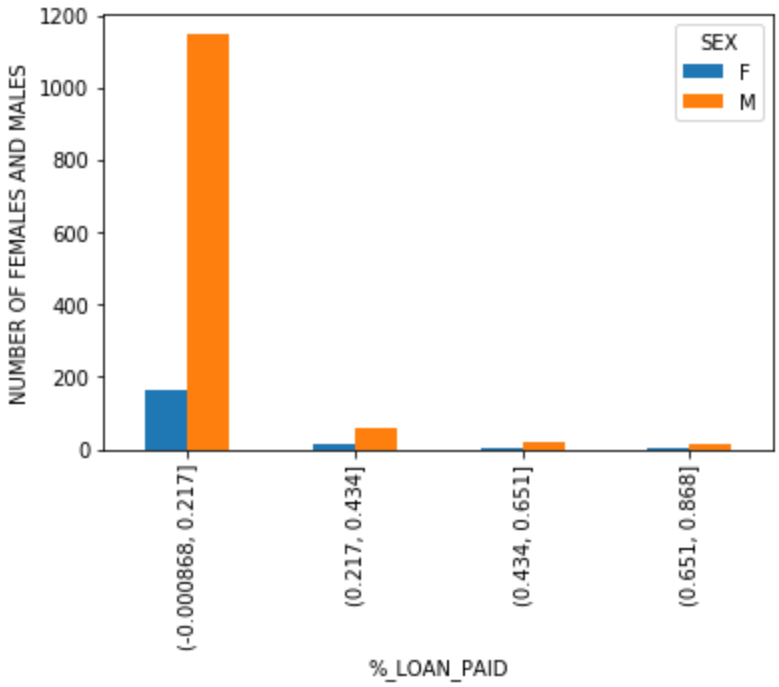
- Mumbai, Hyderabad, Pune, Chennai and Ahmedabad are the top 5 countries with highest foreclosure cases.



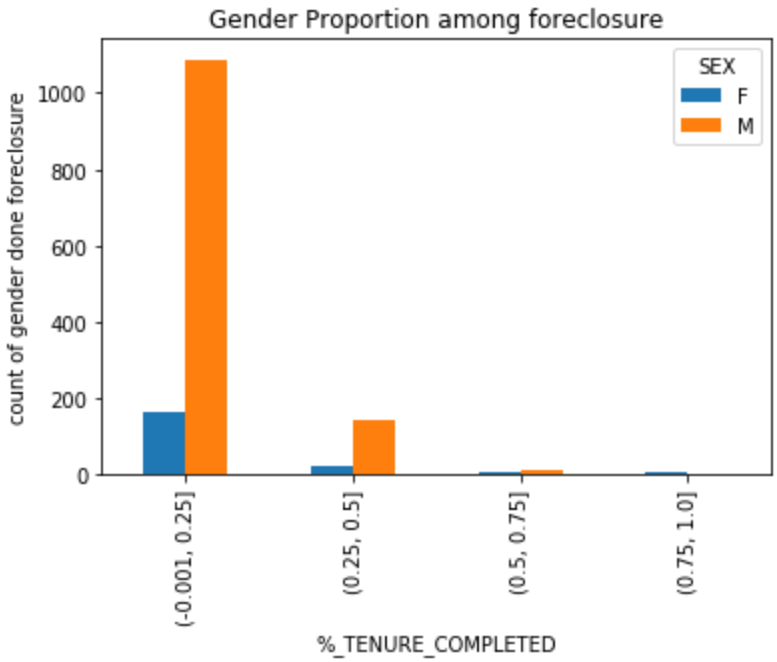
- Top 3 cities having highest foreclosure cases are Mumbai (19.6%), Hyderabad(9.1%) and Pune(8.4%).
- Below shown graphs depicts the proportion of various factors among foreclosure cases.



- There is 87.48% Male people who shows foreclosure after paying 0-21.6% of loan amount.



- Among Foreclosures, Married people paid 1.3 to 5% of total loan and single paid 1.2% to 3.7% of total loan.
- Among Foreclosure, one married guy paid maximum of 85.24% of total loan and among single maximum of loan paid is 56.26%.
- If ROI is increased, then on average Foreclosure happens at 7.2% increase in ROI, which is 3% more than non-foreclosure cases.
- If ROI is decreased, then on average Foreclosure happens at 2.3% decrease in ROI, which is 2.5% more than non-foreclosure cases.
- Mostly, Foreclosure happens (as per IQR) when ROI is increased in range [3.65%-7.46%].
- There is a foreclosure case that happens at a decrease of 29.97% ROI and case that happens at increase of 43.11% in ROI.
- There are 914 foreclosure cases that happens at no change in ROI.
- On average, at 14.7% of Tenure completed with median of 11.75%.
- There is foreclosure case, where Tenure completed is 100%.
- Mostly, foreclosure happens in range [7.2% - 18.2%] of tenure completed.
- There are 24 cases of foreclosure that happens where customer has no 0% tenure completed. It implies lumpsum repayment or customer has not initiated paying EMIs.
- Around 87% of foreclosures happen after completing 0-25% of total tenure.
- Approximately 98% of foreclosures happen after completing 0-50% of total tenure.
- There is 86.8% and 87.4% male people who show foreclosure after completing 0-25% and 0-50% tenure respectively.



- No difference between non-foreclosure and foreclosure cases. Both have mostly 50% PRE_EMI paid.
- No difference between non-foreclosure and foreclosure cases. Both have mostly 50% EMI paid.
- There is 15 cases of foreclosure, where no EMI is paid.
- There are 998 foreclosure cases where either excess is not available or nothing is adjusted.
- 55% of foreclosure cases arises for HL and 44% for STHL.
- Females show 59% foreclosure in STHL and remaining in HL, whereas males show 52.7% foreclosure in HL and remaining in STHL.
- Other people showing foreclosure with higher incomes might have paid the tenure by paying all in as lumpsum.

3.6 MISSSING VALUES

Dealing with missing values is an important aspect of EDA. Missing data can be anything from missing sequence, incomplete feature, files missing, information incomplete, data entry error etc. The techniques we will use to deal with such case includes filling null values by observing some significant pattern of that feature with other features. The other way could be replacing them with statistical measures such as mean, median or mode depending upon the data distribution and data type. The below mention table highlights the features consisting null values and to what extent.

FEATURES	MISSING VALUES	PERCENTAGE CALCULATED
PROFESSION	33354	100
OCCUPATION	33354	100
POSITION	32693	98.018229
PRE_JOBYEARS	30979	92.879415
QUALIFICATION	23957	71.826468
MARITAL_STATUS	23750	71.205852

AGE	23750	71.205852
SEX	23748	71.199856
NO_OF_DEPENDENT	23433	70.255442
BRANCH_PINCODE	23427	70.237453
NETTAKEHOMEINCOME	23354	70.018588
GROSS_INCOME	23354	70.018588
CUST_CATEGORYID	23354	70.018588
CUST_CONSTTYPE_ID	23354	70.018588
DPD	1	0.002998

- The features such as PROFESSION, OCCUPATION, POSITION and PRE_JOBYEARS consisting more than 90% values are dropped. Also the features such as QUALIFICATION, MARITAL_STATUS, SEX, NO_OF_DEPENDENT, BRANCH_PINCODE, CUST_CATEGORYID, CUST_CONSTTYPE_ID are excluded as the percentage of missing value is high and hence imputing those values with existing techniques will generate bias and lower the efficiency of model.
- For the column DPD, the null values were replaced by the mode. And null values in the numerical column like AGE were replaced by mean value.
- NETTAKEHOMEINCOME and GROSS_INCOME depicts a strong correlation. So we will consider keeping only one feature. Moving further the kept feature GROSS_INCOME is filled using GROSS_INCOME = (median (FIXED_OBLIGATION) / FOIR)*30.

3.7 OUTLIER TREATMENT

Since we are dealing with financial data, outlier removal is not advisable as each data point is significant and we need to include them as they are in our analysis. The extreme values in dataset are treated using Standard Scaler technique as it will end up with smaller standard deviations, which can suppress the effect of outliers to some extent.

4. ARCHITECTURE

4.1 HANDLING IMBALANCED DATA

While performing the conventional machine learning on the data which is imbalanced the model will be inaccurate and biased. In this case number of observations in one class will be significantly lesser than other. Standard classifier algorithms like Decision Tree and Logistic Regression have a bias towards classes which have number of instances. This happens because Machine Learning Algorithms are usually designed to improve accuracy by reducing the error. Thus, they do not take into account the class distribution / proportion or balance of classes. They tend to only predict the majority class data. The features of the minority class are treated as noise and are often ignored.

The main objective of balancing classes is to either increasing the frequency of the minority class or decreasing the frequency of the majority class. This is done in order to obtain approximately the same number of instances for both the classes.

4.2 TECHNIQUES

4.2.1 RANDOM UNDER-SAMPLING

Random Under-sampling aims to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are balanced out.

4.2.2 RANDOM OVER-SAMPLING

Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

4.2.3 SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE (SMOTE)

This technique is followed to avoid overfitting which occurs when exact replicas of minority instances are added to the main dataset. A subset of data is taken from the minority class as an example and then new synthetic similar instances are created. These synthetic instances are then added to the original dataset. The new dataset is used as a sample to train the classification models.

5. TENTATIVE LIST OF ALGORITHMS USED& INITIAL APPROACH

Since our problem is a classification problem, we will be using the following algorithms in modelling:

- Logistic Regression
- Naïve Bayes
- Tree Based Classifiers
    - Random Forest
- Ensemble Techniques
    - Ada Boost Classifier
    - Gradient Boosting Classifier

Before proceeding with the modelling techniques, some of features are dropped in order to make resulted model more generalized. These includes 'AGREEMENTID', 'CUSTOMERID', 'SCHEMEID', 'CITY', 'INTEREST_START_DATE' and 'AUTHORIZATIONDATE'.

Modelling using the listed algorithms was done under multiple scenarios and results compiled for overview.

5.1 SCENARIO DESCRIPTIONS:

ORIGINAL DATA& DERIVED VARIABLES

- Modelling was done on the original data after default data cleaning and scaling where necessary and new features were engineered and added from the intuitive understanding of the attributes in the data setas discussed under feature engineering section.

IMPORTANT FEATURES

- Feature Selection using Backward Elimination and feature importance metric is done and only features that were significant to predicting the target variable were considered for modelling.

SMOTE

- Oversampling using SMOTE not only increases the size of the training data set, it also increases the variety. It creates new (artificial) training examples based on the original training examples and adds them as synthetic data points on which the model can be build.

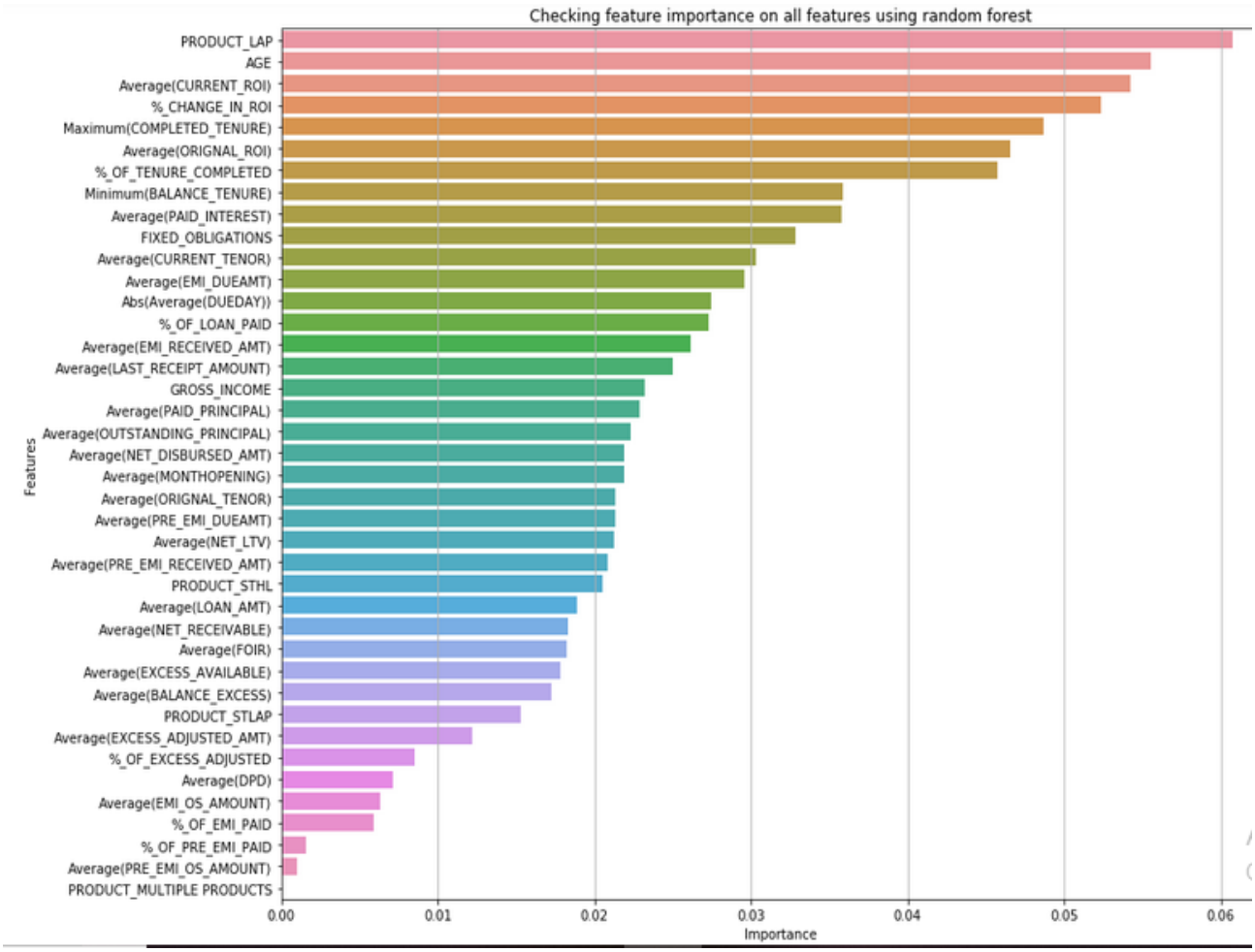
5.2 FEATURE SELECTION

- Feature Selection using Backward Elimination using Logit on Imbalanced Data is applied and following features are observed as most significant.

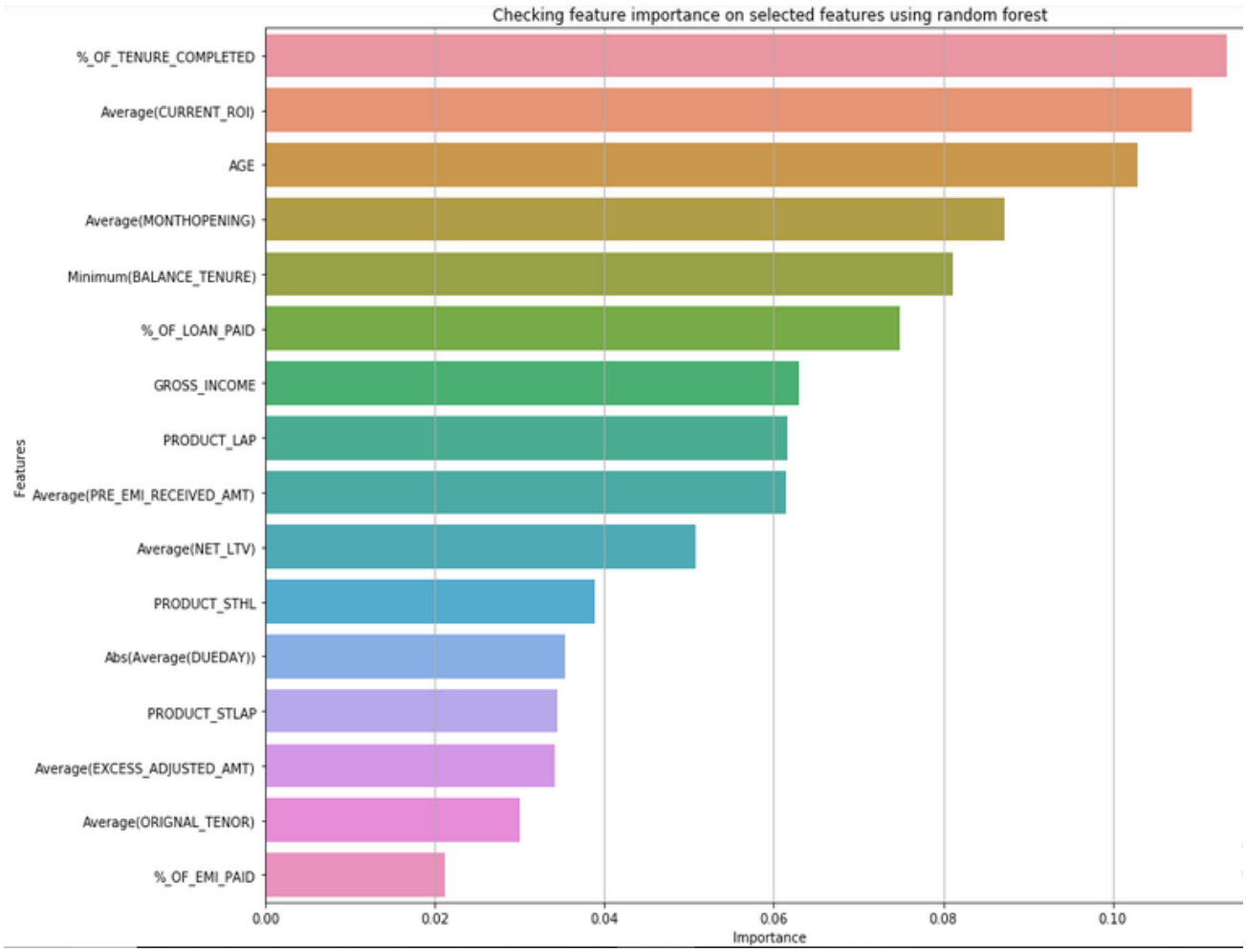
	coef	std err	z	P>|z|	[0.025	0.975]
Average(CURRENT_ROI)	0.4527	0.023	19.369	0.000	0.407	0.499
Average(ORIGNAL_TENOR)	0.1498	0.033	4.582	0.000	0.086	0.214
Abs(Average(DUEDAY))	0.1877	0.018	10.698	0.000	0.153	0.222
Average(PRE_EMI_RECEIVED_AMT)	-0.0434	0.019	-2.240	0.025	-0.081	-0.005
Average(EXCESS_ADJUSTED_AMT)	-0.0495	0.023	-2.157	0.031	-0.095	-0.005
Average(MONTHOPENING)	0.0484	0.018	2.724	0.006	0.014	0.083
Average(NET_LTV)	-0.0548	0.017	-3.260	0.001	-0.088	-0.022
Minimum(BALANCE_TENURE)	-0.1877	0.039	-4.760	0.000	-0.265	-0.110
AGE	-0.0311	0.015	-2.017	0.044	-0.061	-0.001
GROSS_INCOME	0.0477	0.022	2.150	0.032	0.004	0.091
%_OF_LOAN_PAID	0.1009	0.028	3.600	0.000	0.046	0.156
%_OF_TENURE_COMPLETED	-0.2058	0.027	-7.577	0.000	-0.259	-0.153
%_OF_EMI_PAID	0.1189	0.016	7.654	0.000	0.088	0.149
PRODUCT_LAP	-0.7423	0.024	-30.966	0.000	-0.789	-0.695
PRODUCT_STHL	-0.5191	0.024	-21.962	0.000	-0.565	-0.473
PRODUCT_STLAP	-0.8221	0.028	-29.366	0.000	-0.877	-0.767

- Checking feature importance on all features using random forest.

Below mentioned graph depicts the features in order of their importance in predicting the target variable.

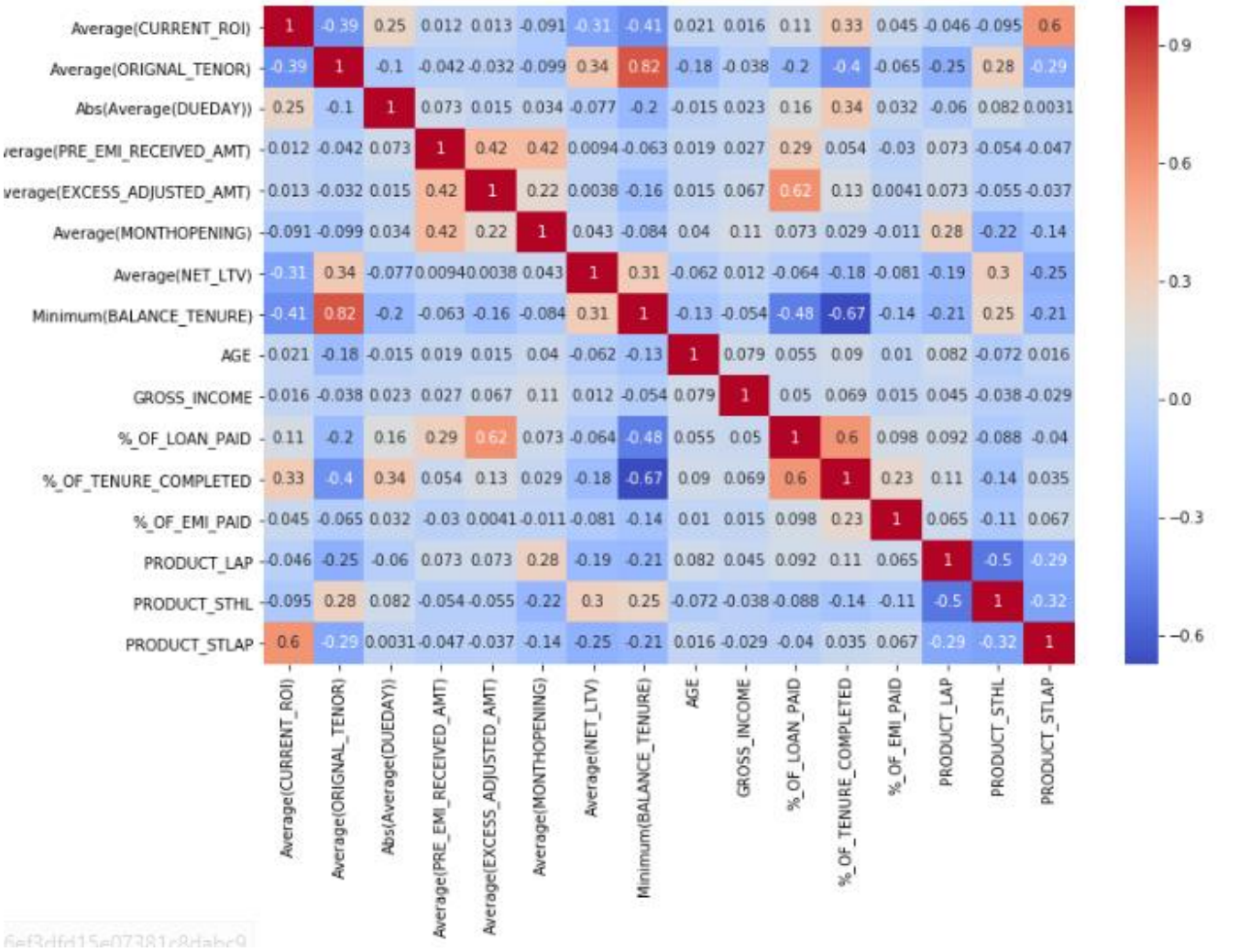


- Checking feature importance on selected features using random forest



5.2.1 Correlation matrix for features extracted using backward elimination

Correlation among selected 16 features through backward elimination is comparatively less.



6. RESULTS AND COMPARISON STUDY

Since we are dealing with foreclosure cases, we will be focusing on **Recall** rather than **Precision** i.e. we will allow Type I error to creep in to our models because in our case, Type II error is far costlier.

The results for all individual cases are compiled into tabular form in order to better compare between quality metrics for all learning algorithms.

Note: The applied models will be evaluated on the basis of ROC_AUC_Score as mentioned in problem statement. Therefore we will consider this score.

6.1. Model applied on imbalanced data.

Model performance on all features (Original Data & Derived variables)

Model	ROC_AUC_Score	Accuracy	f1_Score	Precision	Recall
LogisticRegression	0.710559	0.933773	0.544186	0.717791	0.438202
GaussianNB	0.747933	0.555161	0.285094	0.16672	0.983146
RandomForestClassifier	0.801429	0.959453	0.730337	0.912921	0.608614

Inference

In this case we have good accuracy for two model, but we either have good precision or good recall. The model is not able to classify the target correctly. The accuracy is high only because of the imbalance between the foreclosure and non-foreclosure. Therefore, we can't rely on these models.

Model performance on features extracted using backward elimination on Logit

Model	ROC_AUC_Score	Accuracy	f1_Score	Precision	Recall
LogisticRegression	0.696143	0.930563	0.515901	0.695238	0.410112
GaussianNB	0.771873	0.601791	0.307376	0.182293	0.979401
RandomForestClassifier	0.796113	0.954384	0.704595	0.847368	0.602996

Inference

On the selected variables, when important features were found out, these are the results we observe. It is not significantly different from the original data modelling. Drop in ROC_AUC_Score can be observed after feature selection for LogisticRegression and RandomForestClassifier. However, in case of GaussianNB, the score has increase little as more dependent variables are eliminated.

6.2Model Applied on Balanced data

6.2.1 SMOTE, Model Performance on all features

Model	ROC_AUC_Score	Accuracy	f1_Score	Precision	Recall
LogisticRegression	0.849429	0.828856	0.479712	0.330502	0.874532
GaussianNB	0.743244	0.555837	0.283065	0.165656	0.97191
RandomForestClassifier	0.887696	0.950667	0.747841	0.69391	0.810861

6.2.2 SMOTE, Model performance on features extracted using backward elimination on Logit

Model	ROC_AUC_Score	Accuracy	f1_Score	Precision	Recall
LogisticRegression	0.839471	0.81534	0.459179	0.312038	0.868914
GaussianNB	0.766998	0.602129	0.305105	0.181086	0.968165
RandomForestClassifier	0.86777	0.943572	0.712565	0.659236	0.775281

Inference

After SMOTE, here we get a reasonable accuracy and recall rate with decent AUC value. We can observe improvement in f1_score for GaussianNB and Random Forest Classifier as classes are balanced now. We achieved the best result with Random Forest Classifier having ROC_AUC metric (0.88) and F1 score (0.75). ROC_AUC_Score of Logistic Regression Improved from 0.69 to 0.83 but F1_Score (0.45) is too low in comparison to Random Forest Classifier (0.75).

6.3 ENSEMBLE METHOD

6.3.1 Model performance on all features

Model	ROC_AUC_Score	Accuracy	f1_Score	Precision	Recall
AdaBoostClassifier	0.89125	0.95253	0.75629	0.70436	0.81648
GradientBoostingClassifier	0.90716	0.92009	0.66807	0.53423	0.89139

6.3.2 Model performance on features extracted using backward elimination on Logit

Model	ROC_AUC_Score	Accuracy	f1_Score	Precision	Recall
AdaBoostClassifier	0.87067	0.94425	0.71649	0.6619	0.7809
GradientBoostingClassifier	0.88691	0.90319	0.61775	0.47979	0.86704

Inference

As we moved to Ensemble modelling technique we see AdaBoost Classifier giving almost same Roc_Auc_Score as Random Forest Classifier but F1_Score is not good as Random Forest Classifier. Gradient boosting's F1_Score (0.61) which is also not an improvement.

6.3.3 Model performance on features extracted using backward elimination on Logit

Model	ROC_AUC_Score	Accuracy	f1_Score	Precision	Recall
Light GBM	0.909	0.94069	0.70849	0.6219	0.8109
SVC	0.89691	0.95319	0.68775	0.597661	0.79704

Inference

Further we implemented Light Gradient Boosting and Support Vector Classifier modelling technique we see Light GBM giving almost same Roc_Auc_Score and Recall valiues only F! score of Random Forest outperforming Light GBM. SVC is almost performed like Light GBM as giving higher accuracy and ROC_AUC_Scores but Recall and F1_Score is not an improvement over Random Forest.

6.4 Cross Validation

Cross-validation is performed on training dataset to achieve average testing scores with standard deviation to know the estimate intervals defining range of test scores.

6.4.1 Cross Validation on all features

Model	Avg (ROC_AUC_Score)	Std (ROC_AUC_Score)	Avg (Accuracy)	Std (Accuracy)
LogisticRegression	0.9005	0.0733	0.8807	0.0795
GaussianNB	0.7604	0.0264	0.5602	0.0858
RandomForestClassifier	0.8426	0.0627	0.7808	0.1961

6.4.2 Cross Validation on features extracted using Backward Elimination on Logit

Model	Avg (ROC_AUC_Score)	Std (ROC_AUC_Score)	Avg (Accuracy)	Std (Accuracy)
LogisticRegression	0.9093	0.0701	0.8803	0.0823
GaussianNB	0.8547	0.0313	0.6064	0.0648
RandomForestClassifier	0.8661	0.0423	0.8617	0.0851

7. IMPLICATIONS

- The solution in the form of predicted probabilities and insights, will be an added edge to the bank. Bank can keep an eye on probable foreclosure cases and provide necessary support if required to overcome them and prevent revenue loss.
- Insights presented using the proportions of various quantities like percentage of loan paid, percentage of tenure completed, percentage change in ROI etc. will help the bank identifying at which stage of the total loan period foreclosure is more prominent to happen.
- Other micro level analysis on foreclosure cases using variables like Sex, Marital status, GrossIncome, City etc. will be helpful for bank to understand what kind of customers are more likely to do foreclosure, if loan was sanctioned to them.
- Final selected model 'Random Forest Classifier' achieved average roc_auc_score of 86% with 4% standard deviation.

8. LIMITATIONS

- The provided dataset was highly Imbalanced having 9:1 class ratio. The sampling techniques comes with many drawbacks. So even after balancing the data, there is a reasonable chance of biasness introduce in the data.

- The original independent variables are not enough for the identification of foreclosure indicators to a reasonable accuracy just by themselves.Actual foreclosure reasons could be different in real life.For example, Divorced, Location Change, changes in bank policies, job lost, variation in interest rates in other banks, market trends etc can significantly impact a foreclosure case.

- Also in the customer demographics dataset, many features were having more than 80% of missing values due to which we have to drop them. This led to loss in information which might play significant role in determining foreclosure cases.

9. CLOSING REFLECTIONS

- Further we can consider applying other feature selection techniques like Forward Selection, Bi-directional elimination, Recursive Feature Elimination or using statistical tests to drop features which are not significant.

- Exploring algorithms such as Support Vector Machine, Markov chain Monte Carlo, Neural Networks and more ensemble techniques like XGBoost and LGBM can also be applied to improve test metrics score.

- We can perform statistical tests like ANOVA to check difference in means and Chi-square to check dependency of categorical variables on each feature with target variable.

APPENDIX

Data Dictionary for all the 53 variables present in the dataset.

FEATURES	DESCRIPTION
LOAN_AMOUNT	The gross/actual loan taken by the customer
NET_DISBURSED_AMNT	The net amount calculated by deducting processing fees, charges and other factors
INTEREST_START_DATE	Start date of interest
CURRENT_ROI	Current rate of interest on principal amount
ORIGINAL_ROI	Original rate of interest on principal at the time of loan disbursement
CURRENT_TENOR	Present duration from the disbursement of loan till the last EMI payment of loan amount
ORIGINAL_TENOR	Original duration from the disbursement of loan till the last EMI payment of loan amount
DUEDAY	It is the last day of payment of EMI for the loan
AUTHORIZATIONDATE	Authorization of the loan date
CITY	Name of the city
PRE_EMI_DUE_AMT	It is the amount payable before the disbursement of the actual loan amount
PRE_EMI_RECEIVED_AMT	Amount paid by the borrower to the lender as Pre EMI interest amount before the disbursal of actual loan
PRE_EMI_OS_AMT	It the remaining Pre EMI amount that borrower needs to pay to the lender
EMI_DUEAMT	Amount that borrower needs to pay as EMIs on loan amount
EMI_RECEIVED_AMT	Amount received as EMI by the lender
EMI_OS_AMT	It is the remaining EMI amount that borrower needs to pay for previous EMI
EXCESS_AVAILABLE	The amount available to bank as the availability amount of collateral

EXCESS_ADJUSTED_AMT	The amount adjusted in LOAN_AMOUNT by the lender. However, the borrower still pays the entire Loan Amount. This is internal to lender only
BALANCE_EXCESS	The remaining amount calculated as (EXCESS_AVAILABLE - EXCESS_ADJUSTED_AMT)
NET_RECEIVABLE	Amount to be received by lender as per the amount in BALANCE_EXCESS
OUTSTANDING_PRINCIPAL	Principal amount payable by borrower
PAID_PRINCIPAL	Part of loan amount paid by borrower
PAID_INTEREST	Amount paid as interest on loan amount
MONTHOPENING	Total outstanding loan amount payable
LAST_RECEIPT_DATE	Last payment receipt date
LAST_RECEIPT_AMOUNT	Amount received in the latest payment by the borrower
NET_LTV	Loan-to-Value ratio (in percentage)
COMPLETED_TENURE	Duration up to which part or whole of the loan amount
BALANCE_TENURE	Remaining duration to pay the outstanding loan amount
DPD	Days Past Due shows worth of EMI is not paid by the borrower.
FOIR	Fixed Obligation to Income Ratio is the ratio that is used by lenders to give loans.
PRODUCT	The commodity or object on which the loan is given. The products included HL(House Loan), STHL(Short Term House Loan), LAP(Loan Against Property) and STLAP(Short Term Loan Against Property)
SCHEMEID	Scheme id
NPA_IN_LAST_MONTH	Non-performing assets in last month
NPA_IN_CURRENT_MONTH	Non-performing assets in current month
FORECLOSURE	Forced payment of Loan initiated by lender after selling the collateral or it is the early payment of loan amount resulting in early closure of all EMIs and full settlement.
AGREEMENTID	Unique Agreement id provided to customer for a sanctioned loan
CUSTOMERID	Unique Customer ID given to each customer
CUST_CONSTTYPE_ID	Specific type of each customer
CUST_CATEGORYID	Customer category
PROFESSION	Profession of the customer
AGE	Age of the customer
SEX	Sex of the customer
MARITAL_STATUS	Marital status of the customer
QUALIFICATION	Qualification of the customer
NO_OF_DEPENDENT	Number of dependents of the customer
OCCUPATION	Occupation
POSITION	Position of the customer
GROSS_INCOME	Gross income of the customer
PRE_JOBYEARS	Number of years of experience
NETTAKEHOMEINCOME	Net take home of the customer
BRANCH_PINCODE	Pincode of the branch

Codes

Data Merging

```
customer = pd.read_excel('../Finance/Raw datasets -DR DG-IIT D/Customers_31JAN2019.xlsx')
data = pd.read_excel('../PREPARED_FINAL_TRANSACTIONAL_DATA.xlsx')
train = pd.read_csv('../Finance/Raw datasets -DR DG-IIT D/train_foreclosure.csv')
test = pd.read_csv('../Finance/Raw datasets -DR DG-IIT D/test_foreclosure.csv')
```

```
f_train = pd.merge(pd.merge(train, data, left_on = 'AGREEMENTID', right_on = 'AGREEMENT_ID'), customer, on = 'CUSTOMERID',
                   how = 'left')
f_test = pd.merge(pd.merge(test, data, left_on = 'AGREEMENTID', right_on = 'AGREEMENT_ID'), customer, on = 'CUSTOMERID',
                  how = 'left')
```

```
# Combining train and test sets
df = pd.concat((f_train, f_test), sort = False, ignore_index = True)
```

```
df.describe()
```

Checking Skewness and Kurtosis

```
a=pd.DataFrame('Features':df.columns,
               'skewness':df.skew(),
               'kurtosis':df.kurt())
a=a.sort_values(by=['skewness'],ascending=False)
a[a['skewness']>28]
```

Correlation Heatmap

```
sns.heatmap(df.corr(),annot=True)
plt.show()
```

Univariate Analysis

```
# Fetching numerical and categorical columns
temp_df = df.drop(['AGREEMENTID', 'CUSTOMERID', 'INTEREST_START_DATE', 'AUTHORIZATIONDATE'], axis = 1)
numerical = temp_df.dropna(how = 'all', axis = 1).dtypes[temp_df.dropna(how = 'all', axis = 1).dtypes != 'object'].index
obj = data_info[(data_info['Unique_Values'] <= 10)].index
```

```
## Function to plot box and kde plots of numeric columns
def kde_box_pair(x):
    fig, ax = plt.subplots(len(x.columns), 2, figsize = (16, 140))
    cols = x.columns
    for i in range(2):
        for j in range(len(x.columns)):
            if i == 0:
                sns.kdeplot(x[cols[j]], ax = ax[j][i])
                ax[j][i].axvline(x[cols[j]].mean(), label = 'Mean', color = 'black')
                ax[j][i].axvline(x[cols[j]].median(), label = 'Median', color = 'red')
                ax[j][i].legend()
            else:
                sns.boxplot(x[cols[j]], ax = ax[j][i], showmeans = True)

    plt.tight_layout()
    plt.show()
```

```
# Plotting
kde_box_pair(df[numerical])
```

```
## Function to plot count plots of categorical and discrete columns
def count_plots(x):
    fig, ax = plt.subplots(len(x.columns), 1, figsize = (10, 30))
    cols = x.columns
    for i in range(len(x.columns)):
        sns.countplot(x[cols[i]], ax = ax[i])
    plt.tight_layout()
    plt.show()
```

```
# Plotting, we drop PROFESSION and OCCUPATION because all are null
count_plots(df[obj].drop(['PROFESSION', 'OCCUPATION'], axis = 1))
```

Feature Engineering

```
#1. percentage of loan paid
df['%_LOAN_PAID']=df['Average(PAID_PRINCIPAL)']/df['Average(NET_DISBURSED_AMT)']
#2. percentage change in ROI
df['%_CHANGE_ROI']=(df['Average(CURRENT_ROI)']-df['Average(ORIGNAL_ROI)'])/df['Average(ORIGNAL_ROI)']*100
#3. % of tenure completed
df['%_TENURE_COMPLETED']=df['Maximum(COMPLETED_TENURE)']/df['Average(ORIGNAL_TENOR)']
#4.percentage of pre emi paid
def percentage_pre_emi_paid(x):
    recieved,os,due=x[0],x[1],x[2]
    if(recieved==0 and os==0 and due==0):
        return 0
    else:
        return (recieved/recieved+os+due)*100
df['%_PRE_EMI_PAID'] = df[['Average(PRE_EMI_DUEAMT)', 'Average(PRE_EMI_RECEIVED_AMT)', 'Average(PRE_EMI_OS_AMOUNT)
apply(percent_of_emi_paid, axis = 1)
# 5. Creating %_OF_EMI_PAID
def percent_of_emi_paid(x):
    due, received, os = x[0], x[1], x[2]
    if (os == 0 and received == 0 and due == 0):
        return 0
    else:
        return received/(due + received + os) * 100

df['%_EMI_PAID'] = df[['Average(EMI_DUEAMT)', 'Average(EMI_RECEIVED_AMT)', 'Average(EMI_OS_AMOUNT)']].\
apply(percent_of_emi_paid, axis = 1)
#6. percntage of excess adjusted
def excess(x):
    excess,adj=x[0],x[1]
    if excess==0:
        return 0
    else:
        return adj/excess*100

df['%_EXCESS_ADJUSTED']=df[['Average(EXCESS_AVAILABLE)','Average(EXCESS_ADJUSTED_AMT)']].apply(excess)
```

One Hot Encoding, Scaling and Clustering

```
f_train = pd.get_dummies(f_train, drop_first = True)
f_test = pd.get_dummies(f_test, drop_first = True)

features = list(f_train.columns)
explain = features.copy()
```

```
# Scaling features using standard scalar
for i in features:
    sc = StandardScaler()
    f_train[i] = sc.fit_transform(f_train[i].values.reshape(-1,1))
    f_test[i] = sc.transform(f_test[i].values.reshape(-1,1))
```

```
from sklearn.cluster import KMeans
```

```
# Using KMeans without PCA (Let's check elbow plot between any two pairs of date)
clusters_range = np.arange(1,11)
clusters_error, clusters_error1 = [], []
for i in clusters_range:
    kmeans = KMeans(n_clusters = i, n_jobs = -1)
    kmeans.fit(f_train[features])
    clusters_error.append(kmeans.inertia_)
    kmeans.fit(f_train[features].iloc[:, -6:-10:-1])
    clusters_error1.append(kmeans.inertia_)

# Elbow plot
fig, ax = plt.subplots(1,2, figsize = (14,6))
ax[0].plot(clusters_range, clusters_error, marker = 'o')
ax[1].plot(clusters_range, clusters_error1, marker = 'o')
ax[0].set_title('Clustering on all features')
ax[1].set_title('Clustering on random subset of features')
plt.show()
```

```
# Elbow plots with different principal components (features)
def pca_elbow(x):
    fig, ax = plt.subplots(2,2, figsize = (16,10))
    for i,j,k in zip(range(2,6), np.repeat([0,1], 2), np.tile([0,1], 2)):
        pca = PCA(n_components = i)
        features = pca.fit_transform(x)
        clusters_range = np.arange(1,11)
        clusters_error = []
        for z in clusters_range:
            kmeans = KMeans(n_clusters = z, n_jobs = -1)
            kmeans.fit(features)
            clusters_error.append(kmeans.inertia_)

        # Elbow plots
        sns.pointplot(x = clusters_range, y = clusters_error, ax = ax[j][k])
        ax[j][k].set_title('Elbow plot with {} principal components'.format(i))
        ax[j][k].set_ylabel('WCSS')
        ax[j][k].set_xlabel('# Clusters')
    plt.show()

pca_elbow(f_train)
```

```
# Creating 3 clusters on 2 principal components
label_3 = []
pca = PCA(n_components = 2)
features = pca.fit_transform(f_train)
kmeans = KMeans(n_clusters = 3)
kmeans.fit(features)
label_3 = kmeans.labels_
```

```
f_train['PC1'] = features[:,0]
f_train['PC2'] = features[:,1]
f_train['PCA_2_Cluster_3'] = label_3
f_train['FORECLOSURE'] = target
```

```
fig, ax = plt.subplots(1,2, figsize = (14, 6))
sns.scatterplot(x = 'PC1', y = 'PC2', hue = 'PCA_2_Cluster_3', data = f_train, palette = 'rainbow', ax = ax[0], legend = 'ful
ax[0].set_title('3 Clusters build on 2 components(features)')
plt.show()
```

```
forec = f_train[f_train['FORECLOSURE'] == 1].drop(['PC1', 'PC2', 'PCA_2_Cluster_3'], axis = 1)
```

```
# Elbow plots with different principal components (features)
def pca_elbow(x):
    fig, ax = plt.subplots(2,2, figsize = (16,10))
    for i,j,k in zip(range(2,6), np.repeat([0,1], 2), np.tile([0,1], 2)):
        pca = PCA(n_components = i)
        features = pca.fit_transform(x)
        clusters_range = np.arange(1,11)
        clusters_error = []
        for z in clusters_range:
            kmeans = KMeans(n_clusters = z, n_jobs = -1)
            kmeans.fit(features)
            clusters_error.append(kmeans.inertia_)

        # Elbow plots
        sns.pointplot(x = clusters_range, y = clusters_error, ax = ax[j][k])
        ax[j][k].set_title('Elbow plot with {} principal components'.format(i))
        ax[j][k].set_ylabel('WCSS')
        ax[j][k].set_xlabel('# Clusters')
    plt.show()

pca_elbow(forec.drop('FORECLOSURE', axis = 1))
```

```
# Creating 3 clusters on 2 principal components
label_3 = []
pca = PCA(n_components = 2)
features = pca.fit_transform(forec)
kmeans = KMeans(n_clusters = 3)
kmeans.fit(features)
label_3 = kmeans.labels_
```

```
forec['PC1'] = features[:,0]
forec['PC2'] = features[:,1]
forec['PCA_2_Cluster_3'] = label_3
```

```
fig, ax = plt.subplots(1,2, figsize = (14, 6))
sns.scatterplot(x = 'PC1', y = 'PC2', hue = 'PCA_2_Cluster_3', data = forec, palette = 'rainbow', ax = ax[0], legend = 'full'
ax[0].set_title('3 Clusters build on 2 components(features)')
plt.show()
```

EDA w.r.t clustered labels on Foreclosure cases

```
fore_train = train[train['FORECLOSURE'] == 1]
fore_train['PCA_2_Cluster_3'] = label_3
```

```
fore_train['%_OF_LOAN_PAID'].groupby(fore_train['PCA_2_Cluster_3']).describe()
```

```
fore_train['%_CHANGE_IN_ROI'].groupby(fore_train['PCA_2_Cluster_3']).describe()
```

```
fore_train['%_OF_TENURE_COMPLETED'].groupby(fore_train['PCA_2_Cluster_3']).describe()
```

```
fore_train['GROSS_INCOME'].groupby(fore_train['PCA_2_Cluster_3']).describe()
```

```
pd.cut(np.round(fore_train['%_OF_LOAN_PAID'], 2), [0, 48.1, 58.3, 86.768]).value_counts(normalize = True)
pd.cut(np.round(fore_train['%_OF_TENURE_COMPLETED'], 2), [0, 33.3, 52.99, 100]).value_counts(normalize = True)
pd.cut(np.round(fore_train['%_CHANGE_IN_ROI'], 2), [-29.97, 14.84, 33.043, 43.119]).value_counts(normalize = True)
pd.cut(np.round(fore_train['GROSS_INCOME'], 2), [0, 3.636129e+06, 4.261664e+06, 4.619963e+06]).value_counts(normalize = True)
```

```
sns.countplot(dg['FORECLOSURE'])
plt.xlabel('FORECLOSURE')
plt.ylabel('NO. OF COUNTS')
plt.title('DISTRIBUTION OF FORECLOSURE CLASS')
plt.show()
```

```
print(pd.cut(df[df['FORECLOSURE'] == 1]['%_OF_LOAN_PAID'],4).value_counts(normalize = True)*100)
(pd.cut(df[df['FORECLOSURE'] == 1]['%_OF_LOAN_PAID'],4).value_counts(normalize = True)*100).plot(kind = 'bar')
plt.ylabel('% of foreclosure cases')
plt.xlabel('% of loan paid categories')
plt.title('Foreclosure cases for different categories of loan paid')
plt.show()
```

```
print(pd.cut(df[df['FORECLOSURE'] == 1]['%_TENURE_COMPLETED'],4).value_counts(normalize = True)*100)
(pd.cut(df[df['FORECLOSURE'] == 1]['%_TENURE_COMPLETED'],4).value_counts(normalize = True)*100).plot(kind = 'bar')
plt.ylabel('% of foreclosure cases')
plt.xlabel('% of tenure completed categories')
plt.title('Foreclosure cases for different categories of tenure completed')
plt.show()
```

```
fig, ax = plt.subplots(1,2, figsize = (14,4))
df[df['FORECLOSURE'] == 1]['CITY'].value_counts(normalize = True).head().plot(kind = 'barh', ax = ax[0])
ax[0].set_title('Top 5 CITIES having highest foreclosure cases')
df[(df['%_LOAN_PAID'] >= 0) & (df['%_LOAN_PAID'] <= 48.1)]['CITY'].value_counts(normalize = True).head(5).plot(kind = 'barh',
                                                                                                                ax = ax[1])
ax[1].set_title('Top 5 cities where % of loan paid is upto 48%')
plt.suptitle('Proportion of top 5 cities among foreclosure cases', y = 1.05)
plt.tight_layout()
plt.show()
```

```
print(df['CITY'].groupby(df['FORECLOSURE']).value_counts(normalize = True)*100)
```

```
# Plotting
fig, ax = plt.subplots(2,2, figsize = (12,6))
pd.cut(np.round(df['%_LOAN_PAID'], 2), [0, 48.1, 58.3, 86.768]).value_counts(normalize = True).plot(kind = 'barh',
                                                                                                    ax = ax[0][0])
ax[0][0].set_title('Proportion of % of loan paid')
pd.cut(np.round(df['%_TENURE_COMPLETED'], 2), [0, 33.3, 52.99, 100]).value_counts(normalize = True).plot(kind = 'barh',
                                                                                                         ax = ax[0][1])
ax[0][1].set_title('Proportion of % of tenure completed')
pd.cut(np.round(df['%_CHANGE_ROI'], 2), [-29.97, 14.84, 33.043, 43.119]).value_counts(normalize = True).plot(kind = 'barh',
                                                                                                             ax = ax[1][0])
ax[1][0].set_title('Proportion of % change in ROI')
pd.cut(np.round(df['GROSS_INCOME'], 2),[0, 3.636129e+06, 4.261664e+06, 4.619963e+06]).value_counts(normalize = True).plot(kin
                                                                                                    = 'barh', ax = ax[1][1])
ax[1][1].set_title('Proportion of Gross Income')
plt.suptitle('Proportion of various factors among Foreclosure cases', y = 1.05)
plt.tight_layout()
plt.savefig('Proportion_of_some_factors_among_foreclosures.png', pad_inches=0.5)
plt.show()
```

```
pd.crosstab(pd.cut(df[df['FORECLOSURE'] == 1]['%_LOAN_PAID'],4), df['SEX']).plot(kind='bar')
plt.ylabel('NUMBER OF FEMALES AND MALES')
plt.show()
```

```
df[df['%_LOAN_PAID']!=0]['%_LOAN_PAID'].groupby([df['FORECLOSURE'],df['MARITAL_STATUS']]).describe()
```

```
df[df['%_CHANGE_ROI']>0]['%_CHANGE_ROI'].groupby(df['FORECLOSURE']).describe()
```

```
df[df['%_CHANGE_ROI']<0]['%_CHANGE_ROI'].groupby(df['FORECLOSURE']).describe()
```

```
df[df['%_CHANGE_ROI']==0]['%_CHANGE_ROI'].groupby(df['FORECLOSURE']).describe()
```

```
df[df['%_TENURE_COMPLETED'] != 0]['%_TENURE_COMPLETED'].groupby(df['FORECLOSURE']).describe()
```

```
(pd.crosstab(pd.cut(df[df['FORECLOSURE'] == 1]['%_TENURE_COMPLETED'],4), df['SEX'])).plot(kind='bar')
plt.ylabel('count of gender done foreclosure')
plt.title('Gender Proportion among foreclosure')
plt.show()
```

```
df[df['%_PRE_EMI_PAID'] != 0]['%_PRE_EMI_PAID'].groupby(df['FORECLOSURE']).describe()
```

```
df[df['%_EMI_PAID'] != 0]['%_EMI_PAID'].groupby(df['FORECLOSURE']).describe()
```

```
df[df['%_EXCESS_ADJUSTED'] != 0]['%_EXCESS_ADJUSTED'].groupby(df['FORECLOSURE']).describe()
```

```
df['PRODUCT'].groupby([df['FORECLOSURE'], df['SEX']]).value_counts(normalize = True) * 100
```

```
df[['%_LOAN_PAID','%_CHANGE_ROI','%_TENURE_COMPLETED','%_PRE_EMI_PAID','%_EMI_PAID','%_EXCESS_ADJUSTED']].describe()
```

```
df[['%_LOAN_PAID','%_CHANGE_ROI','%_TENURE_COMPLETED','%_PRE_EMI_PAID','%_EMI_PAID','%_EXCESS_ADJUSTED']].corr()
```

Missing Value Imputation

```
#checking missing values
def check_missing(d):
    missing=d.isnull().sum().sort_values(ascending=False)
    percentage=(d.isnull().sum()/d.isnull().count()*100).sort_values(ascending=False)
    f=pd.concat([missing,percentage],axis=1,keys=['missing_values','percentage of missing values'])
    return(f)
a=check_missing(df)
a
```

```
df['AGE']=df['AGE'].fillna(df['AGE'].mean())
```

```
df['Average(DPD)'] = df['Average(DPD)'].fillna(df['Average(DPD)'].mode()[0])
```

```
df['FIXED_OBLIGATIONS'] = df['Average(FOIR)'] * (df['GROSS_INCOME']/30)
df['FIXED_OBLIGATIONS'] = df['FIXED_OBLIGATIONS'].fillna(df['FIXED_OBLIGATIONS'].median())

def null_gross_income(x):
    gross_med = df['GROSS_INCOME'].median()
    g, foir, obg = x[0], x[1], x[2]
    if pd.isnull(g):
        if(foir == 0):
            return gross_med
        else:
            return (obg/foir) * 30
    else:
        return g

df['GROSS_INCOME'] = df[['GROSS_INCOME', 'Average(FOIR)', 'FIXED_OBLIGATIONS']].apply(null_gross_income, axis = 1)
```

One Hot Encoding, Scaling and Feature Selection using Backward Elimination on Logit

```
f_train = pd.get_dummies(f_train, drop_first = True)
f_test = pd.get_dummies(f_test, drop_first = True)

features = list(f_train.columns)
explain = features.copy()
```

```
# Scaling features using standard scalar
for i in features:
    sc = StandardScaler()
    f_train[i] = sc.fit_transform(f_train[i].values.reshape(-1,1))
    f_test[i] = sc.transform(f_test[i].values.reshape(-1,1))
```

```
# Automating the selection of optimal features using backward elimination, here probability threshold is 0.05.
while(True):
    model = sm.Logit(endog = target, exog = f_train[explain]).fit()
    pvals = model.pvalues
    max_pval = pvals[(pvals == pvals.max()) & (pvals > 0.05)]
    if(max_pval.empty == False):
        explain.remove(max_pval.index[0])
        continue
    else:
        break

explain
```

```
# Checking model summary
model = sm.Logit(endog = target, exog = f_train[explain]).fit()
model.summary()
```

```
## Correlation matrix for features extracted using backward elimination
plt.figure(figsize = (12,8))
sns.heatmap(f_train[explain].corr(), annot = True, cmap = 'coolwarm')
plt.show()
```

Model Training and AUC plots

```
# Splitting data in train test
X_train, X_test, y_train, y_test = train_test_split(f_train[explain], target, test_size=0.3)
# Fitting default logistic regression model
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_proba = lr.predict_proba(X_test)[:,1]
print('Test roc auc score  : ', roc_auc_score(y_test, y_pred))
print('Confusion Matrix \n', confusion_matrix(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
print(classification_report(y_test,y_pred))

lr_fpr, lr_tpr, _ = roc_curve(y_test, y_proba)
b_fpr, b_tpr, _ = roc_curve(y_test, np.repeat(0, len(X_test)))

plt.figure(figsize = (10,8))
plt.plot(lr_fpr, lr_tpr, label = 'Logistic')
plt.plot(b_fpr, b_tpr, ls = '--', label = 'Base (all 0)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
# Fitting logistic regression model with class_weights as balanced
lr = LogisticRegression(class_weight = 'balanced')
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_proba = lr.predict_proba(X_test)[:,1]
print('Test roc auc score  : ', roc_auc_score(y_test, y_pred))
print('Confusion Matrix \n', confusion_matrix(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
print(classification_report(y_test,y_pred))

lr_fpr, lr_tpr, _ = roc_curve(y_test, y_proba)
b_fpr, b_tpr, _ = roc_curve(y_test, np.repeat(0, len(X_test)))

plt.figure(figsize = (10,8))
plt.plot(lr_fpr, lr_tpr, label = 'Logistic')
plt.plot(b_fpr, b_tpr, ls = '--', label = 'Base (all 0)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
y_proba = gnb.predict_proba(X_test)[:,1]
print('Test roc auc score  : ', roc_auc_score(y_test, y_pred))

sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
print(classification_report(y_test,y_pred))
lr_fpr, lr_tpr, _ = roc_curve(y_test, y_proba)
b_fpr, b_tpr, _ = roc_curve(y_test, np.repeat(0, len(X_test)))

plt.figure(figsize = (10,8))
plt.plot(lr_fpr, lr_tpr, label = 'Logistic')
plt.plot(b_fpr, b_tpr, ls = '--', label = 'Base (all 0)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
# Using cross validation with Naive Bayes
auc_roc = cross_val_score(gnb, f_train[explain], target, scoring = 'roc_auc', cv = 5)
print('ROC_AUC Scores : ', auc_roc)
print('Mean : {}   Std : {}'.format(auc_roc.mean(), auc_roc.std()))
```

```
# Using cross validation with Logistic Regression
auc_roc = cross_val_score(lr, f_train[explain], target, scoring = 'roc_auc', cv = 5)
print('ROC_AUC Scores : ', auc_roc)
print('Mean : {}   Std : {}'.format(auc_roc.mean(), auc_roc.std()))
```

```
# Splitting data in train test using all features
X_train, X_test, y_train, y_test = train_test_split(f_train[features], target, test_size=0.3)
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)
y_proba = dtc.predict_proba(X_test)[:,1]
print('Test roc auc score  : ', roc_auc_score(y_test, y_pred))
print('Confusion Matrix \n', confusion_matrix(y_test, y_pred))

sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
print(classification_report(y_test,y_pred))

lr_fpr, lr_tpr, _ = roc_curve(y_test, y_proba)
b_fpr, b_tpr, _ = roc_curve(y_test, np.repeat(0, len(X_test)))

plt.figure(figsize = (10,8))
plt.plot(lr_fpr, lr_tpr, label = 'Logistic')
plt.plot(b_fpr, b_tpr, ls = '--', label = 'Base (all 0)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
dtc = DecisionTreeClassifier(max_depth = 12)
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)
y_proba = dtc.predict_proba(X_test)[:,1]
print('Test roc auc score  : ', roc_auc_score(y_test, y_pred))
print('Confusion Matrix \n', confusion_matrix(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
print(classification_report(y_test,y_pred))

lr_fpr, lr_tpr, _ = roc_curve(y_test, y_proba)
b_fpr, b_tpr, _ = roc_curve(y_test, np.repeat(0, len(X_test)))

plt.figure(figsize = (10,8))
plt.plot(lr_fpr, lr_tpr, label = 'Logistic')
plt.plot(b_fpr, b_tpr, ls = '--', label = 'Base (all 0)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
rfc = RandomForestClassifier(n_estimators = 100)
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
y_proba = rfc.predict_proba(X_test)[:,1]
print('Test roc auc score  : ', roc_auc_score(y_test, y_pred))
print('Confusion Matrix \n', confusion_matrix(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
print(classification_report(y_test,y_pred))
lr_fpr, lr_tpr, _ = roc_curve(y_test, y_proba)
b_fpr, b_tpr, _ = roc_curve(y_test, np.repeat(0, len(X_test)))

plt.figure(figsize = (10,8))
plt.plot(lr_fpr, lr_tpr, label = 'Logistic')
plt.plot(b_fpr, b_tpr, ls = '--', label = 'Base (all 0)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
rfc = RandomForestClassifier(n_estimators = 100, max_depth = 12)
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
y_proba = rfc.predict_proba(X_test)[:,1]
print('Test roc auc score  : ', roc_auc_score(y_test, y_pred))
print('Confusion Matrix \n', confusion_matrix(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
print(classification_report(y_test,y_pred))
lr_fpr, lr_tpr, _ = roc_curve(y_test, y_proba)
b_fpr, b_tpr, _ = roc_curve(y_test, np.repeat(0, len(X_test)))

plt.figure(figsize = (10,8))
plt.plot(lr_fpr, lr_tpr, label = 'Logistic')
plt.plot(b_fpr, b_tpr, ls = '--', label = 'Base (all 0)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

Checking model performance using all metrics on all features and selected subset of features

```
def model_performance(models, train, label):
    """
    models : Accepts list of one or all of following models:
    logistic regression : 'lr',
    gaussian naive bayes : 'gnb',
    random forest classifier : 'rfc'

    train : contains dataframe having features
    label : contains series/array of target or label

    Return dataframe containing all performance metric scores.
    """
    auc_scores, accuracy, f1_scores, precision, recall, name = [], [], [], [], [], []
    X_train, X_test, y_train, y_test = '', '', '', ''
    model = ''
    if type(train) == pd.core.frame.DataFrame:
        if type(label) == pd.core.series.Series:
            X_train, X_test, y_train, y_test = train_test_split(train, label, test_size=0.3, random_state = 42)
        elif type(label) == np.ndarray:
            X_train, X_test, y_train, y_test = train_test_split(train, label.values.reshape(-1,1), test_size=0.3, random_stat
        else:
            raise Exception('Type Error : Enter train in dataframe format and label as pandas series or numpy array')
    else:
        raise Exception('Type Error : Enter train in dataframe format and label as pandas series or numpy array')
```

```
    for i in models:
        if i == 'lr':
            model = LogisticRegression()
            model.fit(X_train, y_train)
        elif i == 'gnb':
            model = GaussianNB()
            model.fit(X_train, y_train)
        elif i == 'rfc':
            model = RandomForestClassifier(n_estimators = 100)
            model.fit(X_train, y_train)
        else:
            raise Exception("models must contain only an iterable of one or all of 'lr'  'gnb'  'rfc' ")

        y_pred = model.predict(X_test)

        name.append(model.__class__.__name__)
        auc_scores.append(roc_auc_score(y_test, y_pred))
        accuracy.append(accuracy_score(y_test, y_pred))
        f1_scores.append(f1_score(y_test, y_pred))
        precision.append(precision_score(y_test, y_pred))
        recall.append(recall_score(y_test, y_pred))

    return pd.DataFrame({
        'ROC_AUC_Score' : auc_scores, 'Accuracy' : accuracy, 'f1_Score' : f1_scores,
        'Precision' : precision, 'Recall' : recall
    }, index = name)
```

```
model_performance(models = ['lr', 'gnb', 'rfc'], train = f_train[features], label = target)
```

```
model_performance(models = ['lr', 'gnb', 'rfc'], train = f_train[explain], label = target)
```

```
rfc = RandomForestClassifier(n_estimators = 100)
rfc.fit(f_train[features], target)
feature_importance = pd.DataFrame({
    'Features' : features,
    'Importance' : rfc.feature_importances_
}).sort_values(by = 'Importance', ascending = False)
```

```
plt.figure(figsize = (14, 12))
sns.barplot(x = feature_importance['Importance'], y = feature_importance['Features'])
plt.grid(b = True, axis = 'x')
plt.title('Checking feature importance on all features using random forest')
plt.show()
```

```
rfc = RandomForestClassifier(n_estimators = 100)
rfc.fit(f_train[explain], target)
feature_importance = pd.DataFrame({
    'Features' : explain,
    'Importance' : rfc.feature_importances_
}).sort_values(by = 'Importance', ascending = False)
```

```
plt.figure(figsize = (14, 12))
sns.barplot(x = feature_importance['Importance'], y = feature_importance['Features'])
plt.grid(b = True, axis = 'x')
plt.title('Checking feature importance on selected features using random forest')
plt.show()
```

Checking model performance after applying SMOTE

```
# Checking model performance after applying SMOTE
def model_performance(models, train, label):
    """
    models : Accepts list of one or all of following models:
    logistic regression : 'lr',
    gaussian naive bayes : 'gnb',
    random forest classifier : 'rfc'

    train : contains dataframe having features
    label : contains series/array of target or label

    Return dataframe containing all performance metric scores.
    """
    smote = SMOTE()
    X_res, y_res = '', ''
    auc_scores, accuracy, f1_scores, precision, recall, name = [], [], [], [], [], []
    X_train, X_test, y_train, y_test = '', '', '', ''
    model = ''
    if type(train) == pd.core.frame.DataFrame:
        if type(label) == pd.core.series.Series:
            X_train, X_test, y_train, y_test = train_test_split(train, label, test_size=0.3, random_state = 42)
            X_res, y_res = smote.fit_resample(X_train, y_train)
        elif type(label) == np.ndarray:
            X_train, X_test, y_train, y_test = train_test_split(train, label.values.reshape(-1,1), test_size=0.3, random_stat
            X_res, y_res = smote.fit_resample(X_train, y_train)
        else:
            raise Exception('Type Error : Enter train in dataframe format and label as pandas series or numpy array')
```

```
    for i in models:
        if i == 'lr':
            model = LogisticRegression()
            model.fit(X_res, y_res)
        elif i == 'gnb':
            model = GaussianNB()
            model.fit(X_res, y_res)
        elif i == 'rfc':
            model = RandomForestClassifier(n_estimators = 100)
            model.fit(X_res, y_res)
        else:
            raise Exception("models must contain only an iterable of one or all of 'lr'  'gnb'  'rfc' ")

        y_pred = model.predict(X_test)

        name.append(model.__class__.__name__)
        auc_scores.append(roc_auc_score(y_test, y_pred))
        accuracy.append(accuracy_score(y_test, y_pred))
        f1_scores.append(f1_score(y_test, y_pred))
        precision.append(precision_score(y_test, y_pred))
        recall.append(recall_score(y_test, y_pred))

    return pd.DataFrame({
        'ROC_AUC_Score' : auc_scores, 'Accuracy' : accuracy, 'f1_Score' : f1_scores,
        'Precision' : precision, 'Recall' : recall
    }, index = name)
```

```
model_performance(models = ['lr', 'gnb', 'rfc'], train = f_train[features], label = target)
```

```
model_performance(models = ['lr', 'gnb', 'rfc'], train = f_train[explain], label = target)
```

Ensemble methods

```
# Checking model performance after applying SMOTE
def model_performance(models, train, label):
    """
    models : Accepts list of one or all of following models:
    logistic regression : 'lr',
    gaussian naive bayes : 'gnb',
    random forest classifier : 'rfc'

    train : contains dataframe having features
    label : contains series/array of target or label

    Return dataframe containing all performance metric scores.
    """

    smote = SMOTE()
    X_res, y_res = '', ''
    auc_scores, accuracy, f1_scores, precision, recall, name = [], [], [], [], [], []
    X_train, X_test, y_train, y_test = '', '', '', ''
    model = ''
    if type(train) == pd.core.frame.DataFrame:
        if type(label) == pd.core.series.Series:
            X_train, X_test, y_train, y_test = train_test_split(train, label, test_size=0.3, random_state = 42)
            X_res, y_res = smote.fit_resample(X_train, y_train)
        elif type(label) == np.ndarray:
            X_train, X_test, y_train, y_test = train_test_split(train, label.values.reshape(-1,1), test_size=0.3, random_stat
            X_res, y_res = smote.fit_resample(X_train, y_train)
        else:
```

```
            raise Exception('Type Error : Enter train in dataframe format and label as pandas series or numpy array')
    else:
        raise Exception('Type Error : Enter train in dataframe format and label as pandas series or numpy array')

    for i in models:
        if i == 'rf_boost':
            RF=RandomForestClassifier(n_estimators=100,criterion='entropy',random_state=42)
            model = AdaBoostClassifier(base_estimator=RF,n_estimators=100)
            model.fit(X_res, y_res)
        elif i == 'g_boost':
            model =GradientBoostingClassifier(n_estimators=100)
            model.fit(X_res, y_res)
        elif i == 'l_bag':
            LR=LogisticRegression()
            model = BaggingClassifier(base_estimator=LR,n_estimators=25,random_state=42)
            model.fit(X_res, y_res)
        else:
            raise Exception("models must contain only an iterable of one or all of 'lr'  'gnb'  'rfc' ")

        y_pred = model.predict(X_test)

        name.append(model.__class__.__name__)
        auc_scores.append(roc_auc_score(y_test, y_pred))
        accuracy.append(accuracy_score(y_test, y_pred))
        f1_scores.append(f1_score(y_test, y_pred))
        precision.append(precision_score(y_test, y_pred))
        recall.append(recall_score(y_test, y_pred))
```

```
    return pd.DataFrame({
        'ROC_AUC_Score' : auc_scores, 'Accuracy' : accuracy, 'f1_Score' : f1_scores,
        'Precision' : precision, 'Recall' : recall
    }, index = name)
```

```
# on selected features
model_performance(models=['rf_boost','g_boost','l_bag'],train = f_train[explain], label = target)
```

```
# on all features
model_performance(models=['rf_boost','g_boost','l_bag'],train = f_train[features], label = target)
```

Checking model performance using cross validation

```
def model_performance(models, train, label):
    """
    models : Accepts list of one or all of following models:
    logistic regression : 'lr',
    gaussian naive bayes : 'gnb',
    random forest classifier : 'rfc'

    train : contains dataframe having features
    label : contains series/array of target or label

    Return dataframe containing all performance metric scores.
    """
    auc_scores, accuracy, f1_scores, precision, recall, name = [], [], [], [], [], []
    std_auc, std_accuracy, std_f1, std_precision, std_recall = [], [], [], [], []
    X_train, y_train = '', ''
    model = ''
    if type(train) == pd.core.frame.DataFrame:
        if type(label) == pd.core.series.Series:
            X_train, y_train = train, label
        elif type(label) == np.ndarray:
            X_train, y_train = train, label.values.reshape(-1,1)
        else:
            raise Exception('Type Error : Enter train in dataframe format and label as pandas series or numpy array')
    else:
        raise Exception('Type Error : Enter train in dataframe format and label as pandas series or numpy array')
```

```
    for i in models:
        if i == 'lr':
            model = LogisticRegression()
            #model.fit(X_train, y_train)
        elif i == 'gnb':
            model = GaussianNB()
            #model.fit(X_train, y_train)
        elif i == 'rfc':
            model = RandomForestClassifier(n_estimators = 100)
            #model.fit(X_train, y_train)
        else:
            raise Exception("models must contain only an iterable of one or all of 'lr'  'gnb'  'rfc' ")

        #y_pred = model.predict(X_test)

        name.append(model.__class__.__name__)

        score = cross_val_score(model, X_train, y_train, cv = 5, scoring = 'roc_auc', n_jobs = -1)
        auc_scores.append(score.mean())
        std_auc.append(score.std())

        score = cross_val_score(model, X_train, y_train, cv = 5, scoring = 'accuracy', n_jobs = -1)
        accuracy.append(score.mean())
        std_accuracy.append(score.std())

        score = cross_val_score(model, X_train, y_train, cv = 5, scoring = 'f1', n_jobs = -1)
        f1_scores.append(score.mean())

        std_f1.append(score.std())

        score = cross_val_score(model, X_train, y_train, cv = 5, scoring = 'precision', n_jobs = -1)
        precision.append(score.mean())
        std_precision.append(score.std())

        score = cross_val_score(model, X_train, y_train, cv = 5, scoring = 'precision', n_jobs = -1)
        recall.append(score.mean())
        std_recall.append(score.std())

    return pd.DataFrame({
        'Avg(ROC_AUC_Score)' : auc_scores, 'Std(ROC_AUC_Score)' : std_auc, 'Avg(Accuracy)' : accuracy,
        'Std(Accuracy)' : std_accuracy, 'Avg(f1-score)' : f1_scores, 'Std(f1-score)' : std_f1,
        'Avg(Precision)' : precision, 'Std(Precision)' : std_precision, 'Avg(Recall)' : recall,
        'Std(Recall)' : std_recall
    }, index = name)
```

```
model_performance(models = ['lr', 'gnb', 'rfc'], train = f_train[features], label = target)
```

```
model_performance(models = ['lr', 'gnb', 'rfc'], train = f_train[explain], label = target)
```