CredRisk: A Credit Risk Predictor Model Using Machine Learning

Group 36
Yashasvi Sharma
Pranjal Kalekar

857-350-2818
857-294-2305

sharma.yasha@northeastern.edu
kalekar.p@northeastern.edu

Percentage of Effort Contributed by Student 1: 50

Percentage of Effort Contributed by Student 2: 50

Signature of Student 1: 

Signature of Student 2: 

Submission Date: 04/21/2023

Index:

1. Background:

LendingClub is a peer-to-peer lending service provider that allows individual investors to partially fund personal loans as well as buy and sell notes backing the loans on a secondary market. LendingClub makes historical data available to the public via an API. Financial datasets containing information on credit history, income, employment status, and other relevant data from a sample of individuals or organizations.

Furthermore, by utilizing the most effective machine learning techniques such as regressions, Random Forest, or Neural Networks, we can create models to predict potential borrowers' credit risk. This model can be used to identify patterns and trends that will help financial institutions evaluate the creditworthiness of potential borrowers.

2. Problem Definition:

The intention of this project was to identify the best classification model and predictors for correctly predicting borrowers' credit risk. The intention is to seek how attributes impact classification and how hyper-parameter tunning impacts classification of borrowers.

3. Data Cleaning and Data Exploration:

We used the following steps to identify and correct or remove errors, inconsistencies, and inaccuracies in a dataset in-order to improve its quality and usefulness:

The data sets for year 2020-2021 are sourced from the following API's:

- https://resources.lendingclub.com/LoanStats_2021Q1.csv.zip
- https://resources.lendingclub.com/LoanStats_2021Q2.csv.zip
- https://resources.lendingclub.com/LoanStats_2021Q3.csv.zip
- https://resources.lendingclub.com/LoanStats_2021Q4.csv.zip
- https://resources.lendingclub.com/LoanStats_2020Q4.csv.zip

This data set contains 110048 instances. There are 143 attributes in total, with 1 target attribute, 'loan status'. There are 107 numerical attributes and 37 categorical attributes. For analysis, the data must be cleaned.

For data cleaning:

- All the rows and columns which had all its values null were removed
- Columns with more than 75 % of missing values are also removed from the dataset.
- For the columns consisting less than 75% missing data, we implemented different techniques depending the amount of the missing data, and the type of the data (Numerical/Categorical)
    - We dropped the following columns(respective missing values) instead of filling it to avoid misleading data:

Col_name	No of Missing Values

mths_since_recent_revol_delinq	79386
next_pymnt_d	63886
mths_since_last_delinq	63869
emp_title	16270
emp_length	9496

Table 3.1 Dropped Columns and corresponding missing values

- For the following Columns we removed rows which had null values in it

Col_name	No of Missing Values
il_util	13088
mths_since_recent_inq	10926

Table 3.2 Columns with dropped rows and corresponding missing values

- For the following columns we tried filling the missing data with the median of the column as some values are symmetrical in some columns whereas some are skewed in others.

Col_name	No of Missing Values
bc_util	865
percent_bc_gt_75	820
bc_open_to_buy	819
mths_since_recent_bc	765
dti	121
revol_util	101
mths_since_rcnt_il	12
mo_sin_old_il_acct	12

Table 3.3 Columns with mean filling the missing values and corresponding missing values

- For a couple of columns, we tried filling the missing values with the most frequent value in the column. As the affecting values will be less than 1% of the data, the data updation won't be misleading.

Col_name	No of Missing Values
hardship_flag	143
num_tl_120dpd_2m	637

Table 3.4 Columns and corresponding missing values being filled with most frequent value

- For column "last_pymnt_d" we forward filled the missing values
- After checking for correlation, we find 25 columns which have correlation of more than 0.8 and were removed.

After cleaning the dataset has 88892 entries and 69 columns. In the 69 columns there are float64(52) columns and object(17) i.e categorical columns.

The finalized 69 columns with their description after data cleaning and feature engineering:

loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
int_rate	Interest Rate on the loan
annual_inc	The self-reported annual income provided by the borrower during registration.
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
open_acc	The number of open credit lines in the borrower's credit file.
pub_rec	Number of derogatory public records
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
total_acc	The total number of credit lines currently in the borrower's credit file
out_prncp	Remaining outstanding principal for total amount funded
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
recoveries	post charge off gross recovery
last_pymnt_amnt	Last total payment amount received
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
tot_coll_amt	Total collection amounts ever owed
tot_cur_bal	Total current balance of all accounts
open_acc_6m	Number of open trades in last 6 months
open_act_il	Number of currently active installment trades

open_il_12m	Number of installment accounts opened in past 12 months
open_il_24m	Number of installment accounts opened in past 24 months
mths_since_rcnt_il	Months since most recent installment accounts opened
total_bal_il	Total current balance of all installment accounts
il_util	Ratio of total current balance to high credit/credit limit on all install acct
max_bal_bc	Maximum current balance owed on all revolving accounts
all_util	Balance to credit limit on all trades
total_rev_hi_lim	Total revolving high credit/credit limit
inq_fi	Number of personal finance inquiries
total_cu_tl	Number of finance trades
inq_last_12m	Number of credit inquiries in past 12 months
bc_open_to_buy	Total open to buy on revolving bankcards.
chargeoff_within_12_mths	Number of charge-offs within 12 months
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
mo_sin_old_il_acct	Months since oldest bank installment account opened
mo_sin_old_rev_tl_op	Months since oldest revolving account opened
mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened
mo_sin_rcnt_tl	Months since most recent account opened
mort_acc	Number of mortgage accounts.
mths_since_last_delinq	The number of months since the borrower's last delinquency.
num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
num_actv_bc_tl	Number of currently active bankcard accounts
num_bc_tl	Number of bankcard accounts
num_il_tl	Number of installment accounts
num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)
num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)
num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
pct_tl_nvr_dlq	Percent of trades never delinquent
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
grade	LC assigned loan grade
sub_grade	LC assigned loan subgrade

emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
issue_d	The month which the loan was funded
purpose	A category provided by the borrower for the loan request.
title	The loan title provided by the borrower
addr_state	The state provided by the borrower in the loan application
earliest_cr_line	The month the borrower's earliest reported credit line was opened
initial_list_status	The initial listing status of the loan. Possible values are – W, F
last_pymnt_d	Last month payment was received
last_credit_pull_d	The most recent month LC pulled credit for this loan
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
hardship_flag	Flags whether or not the borrower is on a hardship plan
loan_status	Current status of the loan

Table 3.5 Columns and with their descriptions

3.1 Data Visualizations:

- Heat Map of dataset before Data Cleaning for missing values:

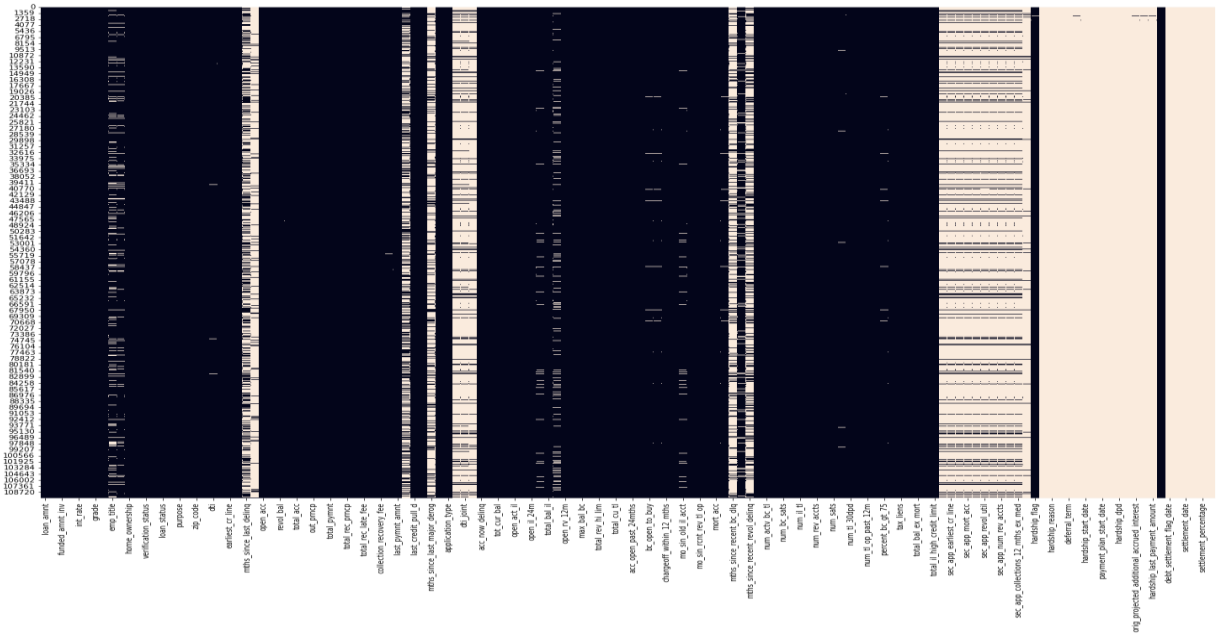


Fig 3.1.1: Heatmap with missing values

The white spaces shows us that null values exist in that attribute before the dataset is cleaned

- Exploratory visualizations to determine the loan risk associated with employment length

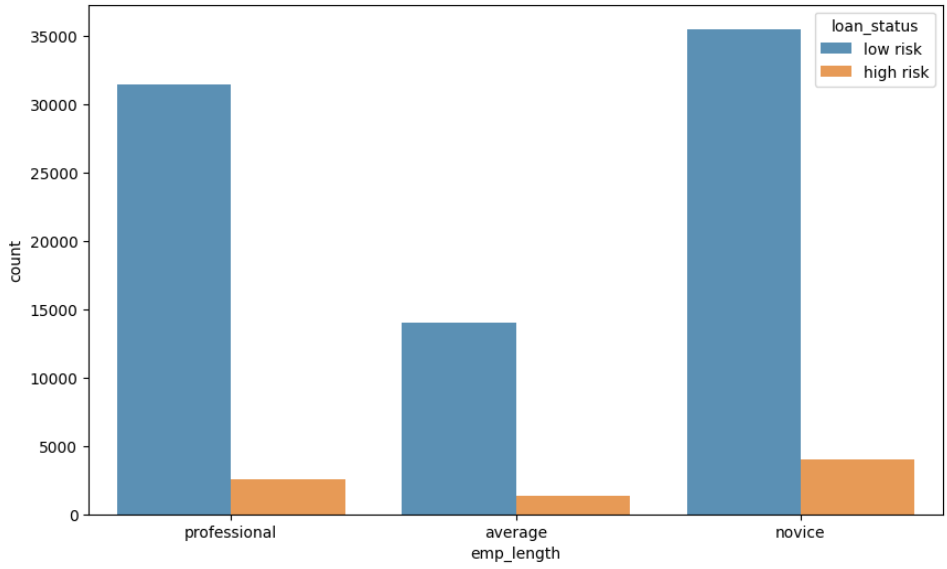


Fig 3.1.2: Countplot of loan risk associated with employment length

We find that novice had the maximum risk associated with them where as average employment had the lowest high risk. However we find that novice take the maximum number of loans

- Exploratory visualizations to determine the sate with maximum loan lending risk.

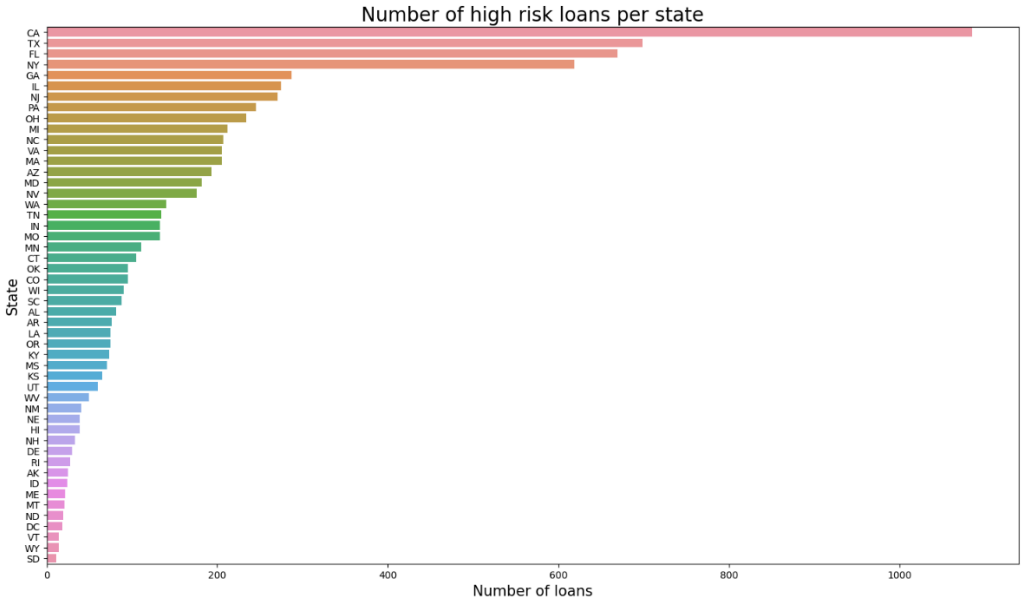


Fig 3.1.3: Bar Plot of high risk loan per state

We infer that California and Texas states has the most number of high risk loans in the US

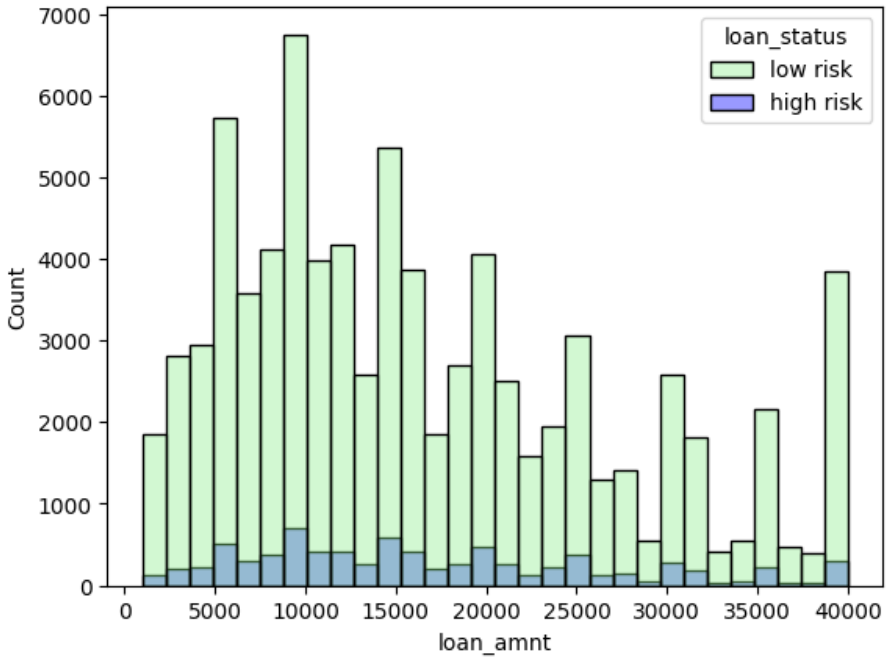


Fig 3.1.4: HistPlot of loan_amnt and ‘count’

We infer that about 10000$ is the amount that is most frequently requested for a loan and the risk associated with it.

- Using Box Plot to display the relationship between loan grade and Interest rate

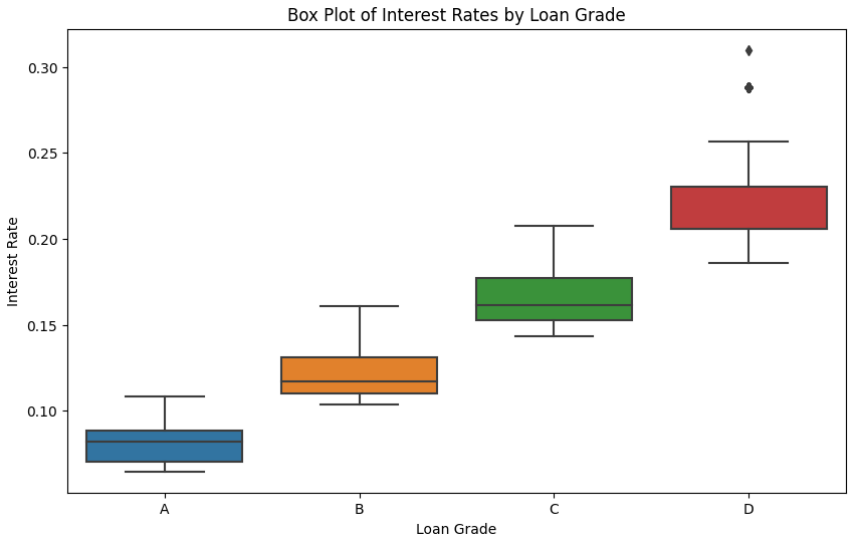


Fig 3.1.4: Box Plot between loan grade and Interest rate

We infer that Loan Grade C is charged the maximum interest rate while Loan Grade A is charged the least interest rate.

- Using log Scatter Plot to display the relationship between Annual Income vs. Debt-to-Income Ratio

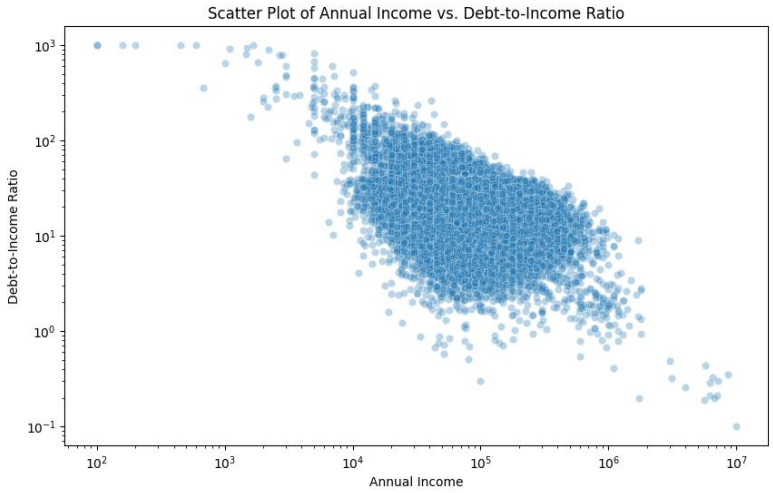


Fig 3.1.5: Scatter Plot of Annual Income vs. Debt-to-Income Ratio

We infer that there is a negative correlation between Annual income and debt to income ratio which means there is an inverse relationship between these two variables, where one tends to go up when the other goes down.

- Correlation heatmap:

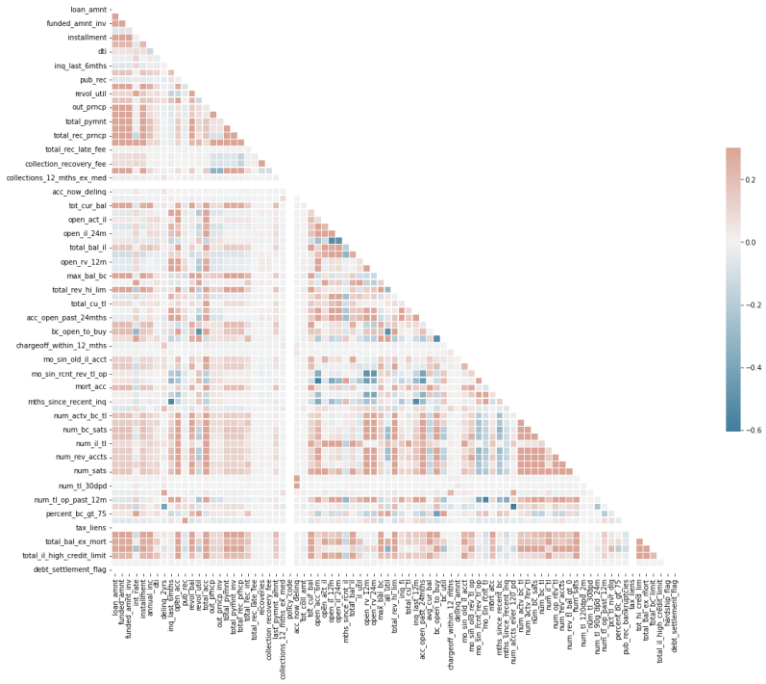


Fig 3.1.6: Correlation Heatmap for column values

The correlation heatmap shows the correlation columns in the dataset with red color indicating high correlation

3.2. Data Processing:

- Visualizing outliers using box plot:

    - A box plot is a statistical representation of a variable's distribution through its quartiles. The lower and upper quartiles are represented by the box's ends, while the median (second quartile) is indicated by a line inside the box.

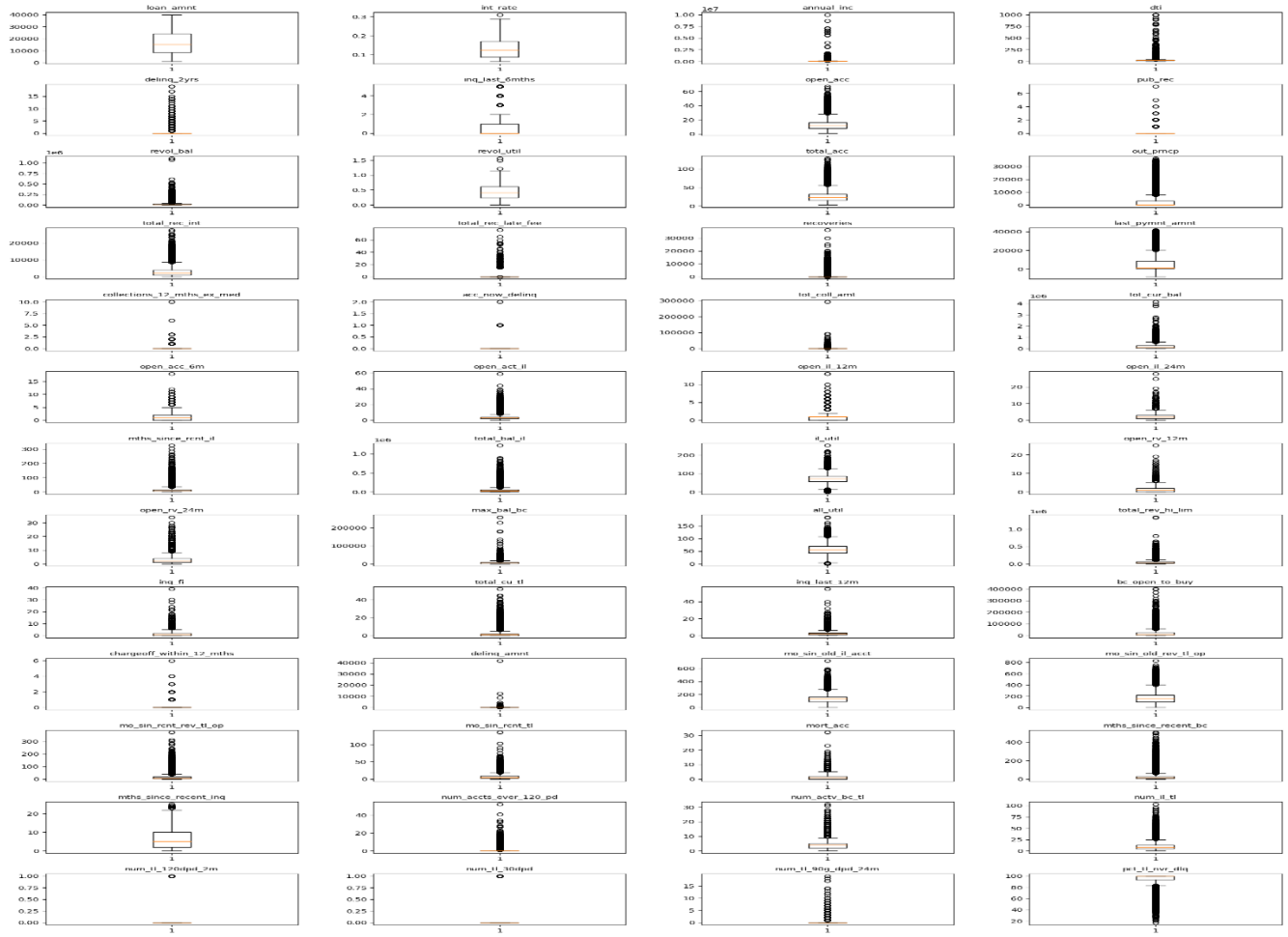


Fig. 3.2.1: Box plot of all 52 numerical Columns

- Removing Outliers:

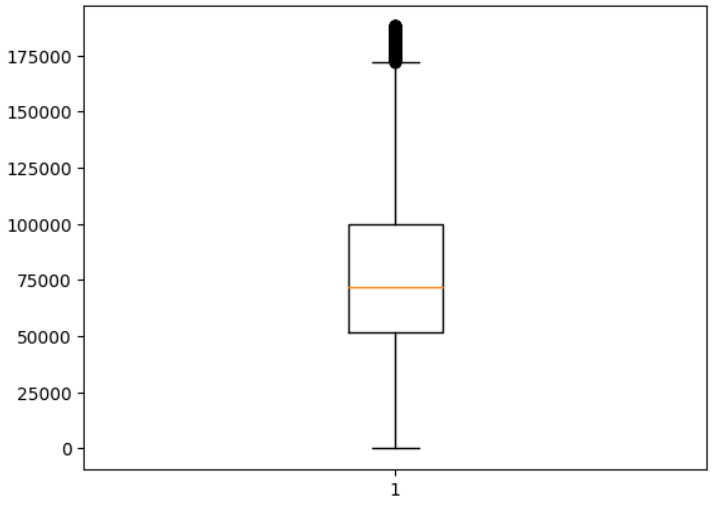


Fig 3.2.2: Box plot of mo_sin_old_rev_tl_op showing outliers

The Inter Quartile Range approach to finding outliers is the most widely used and trusted method in the research field. We have used it to remove outliers in reported income.

- Computing Categorical Variables:

For following categorical values we are using LabelEncoder to encode to numerical values

  - 'hardship_flag'
  - 'emp_length'
  - 'sub_grade'
  - 'grade'
  - 'loan_status'

- Statistics of the variables:

	loan_amnt	int_rate	annual_inc	dti	delinq_2yrs	inq_last_6mths	open_acc	pub_rec	revol_bal	revol_util	total_acc	out_prncp	total_rec_int	total_rec_late_fee	recoveries	last_
count	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	88892.00	
mean	16571.54	0.13	90949.73	22.56	0.21	0.62	12.80	0.10	18642.52	0.43	25.19	2953.50	2919.76	0.03	122.11	
std	10261.09	0.05	102456.13	19.03	0.71	0.85	6.12	0.30	23252.41	0.25	12.62	5595.10	2705.46	0.98	789.82	
min	1000.00	0.06	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	0.00	
25%	8500.00	0.09	53000.00	14.65	0.00	0.00	8.00	0.00	6552.00	0.24	16.00	0.00	962.29	0.00	0.00	
50%	15000.00	0.12	75000.00	20.69	0.00	0.00	12.00	0.00	12761.50	0.41	23.00	0.00	2071.77	0.00	0.00	
75%	23931.25	0.17	107110.52	27.63	0.00	1.00	16.00	0.00	22789.50	0.62	32.00	3121.77	4039.90	0.00	0.00	
max	40000.00	0.31	9999999.00	999.00	19.00	5.00	67.00	7.00	1107809.00	1.56	129.00	36356.11	27746.85	75.88	36195.88	

Fig 3.2.3: Statistics of the dataset

On statistical analysis on the dataset following observation was made:

- Average loan amount that was awarded is 16571.54 with a minimum amount of 1000 and maximum amount of 40000 dollars
- Average interest rate was found to be 13.15 %
- Total loan amount disbursed during the period was found to be 1473077350 $

- Correlation analysis of the variables:

	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	delinq_2yrs	inq_last_6mths	open_acc	pub_rec	revol_bal	revol_util	total_acc	out_prncp
loan_amnt	1.00	1.00	1.00	0.07	0.93	0.22	0.06	0.02	0.06	0.17	0.06	0.35	0.11	0.18	0.41
funded_amnt	1.00	1.00	1.00	0.07	0.93	0.22	0.06	0.02	0.06	0.17	0.06	0.35	0.11	0.18	0.41
funded_amnt_inv	1.00	1.00	1.00	0.07	0.93	0.22	0.06	0.02	0.06	0.17	0.06	0.35	0.11	0.18	0.41
int_rate	0.07	0.07	0.07	1.00	0.01	0.05	0.09	0.10	0.10	0.01	0.04	0.01	0.32	0.02	0.01
installment	0.93	0.93	0.93	0.01	1.00	0.22	0.07	0.00	0.05	0.17	0.05	0.35	0.16	0.16	0.28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
pub_rec_bankruptcies	0.06	0.06	0.06	0.04	0.05	0.03	0.01	0.04	0.06	0.02	1.00	0.08	0.06	0.02	0.04
tot_hi_cred_lim	0.32	0.32	0.32	0.12	0.29	0.31	0.04	0.04	0.03	0.32	0.07	0.46	0.03	0.38	0.10
total_bal_ex_mort	0.28	0.28	0.28	0.02	0.27	0.26	0.14	0.01	0.04	0.40	0.05	0.51	0.14	0.45	0.11
total_bc_limit	0.36	0.36	0.36	0.24	0.33	0.23	0.03	0.07	0.04	0.45	0.12	0.49	0.20	0.37	0.13
total_il_high_credit_limit	0.23	0.23	0.23	0.01	0.22	0.24	0.17	0.03	0.05	0.36	0.04	0.20	0.05	0.44	0.09

Fig 3.2.4: Correlation of the dataset

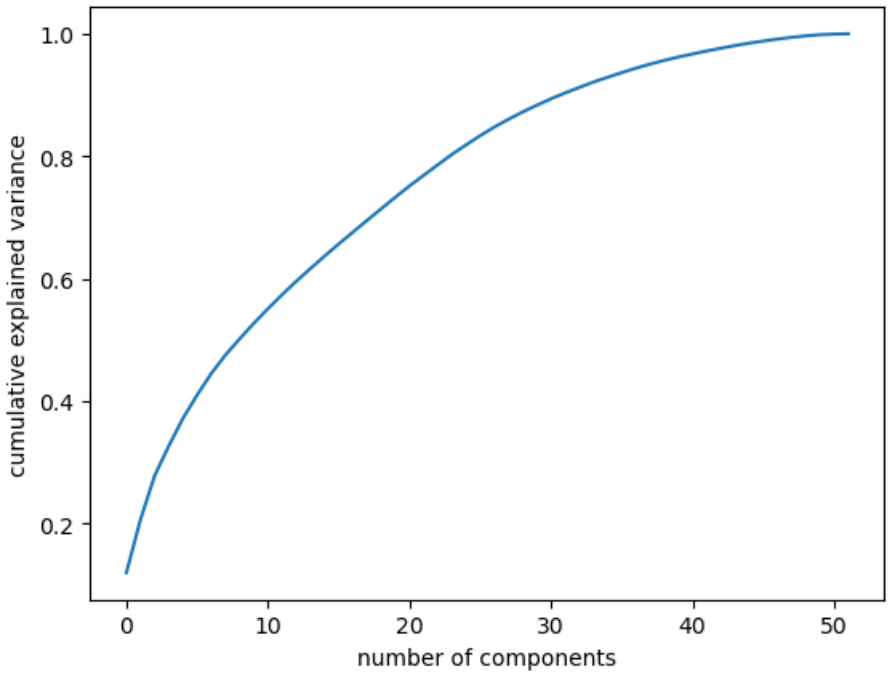
In order to investigate the relationship between the numerical variables in the dataset, a correlation analysis was performed. Columns with a correlation of more than 0.8 and less than -0.8 can be considered has having high correlation and were removed on further analysis
Following are the columns which were found to have high correlation:
'acc_open_past_24mths',
'avg_cur_bal',
'bc_util',
'collection_recovery_fee',
'funded_amnt',
'funded_amnt_inv',
'installment',
'num_actv_rev_tl',
'num_bc_sats',
'num_bc_tl',
'num_op_rev_tl',
'num_rev_accts',
'num_rev_tl_bal_gt_0',
'num_sats',
'num_tl_op_past_12m',
'out_prncp_inv',
'percent_bc_gt_75',
'pub_rec_bankruptcies',
'tot_hi_cred_lim',
'total_bal_ex_mort',
'total_bc_limit',
'total_il_high_credit_limit',
'total_pymnt',
'total_pymnt_inv',
'total_rec_prncp'

3.3 PCA (Principal Component Analysis):

Pca was performed on the dataset and following PC components were found:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16	PC17	PC18	PC19	PC20	PC21	PC22	PC23	PC24	PC25	PC26	PC27	PC28	PC29	PC30
0	3.06	5.49	-1.80	1.49	0.08	-3.36	0.49	-0.34	1.53	2.58	0.41	-0.92	0.58	0.41	-0.47	-1.07	0.07	0.11	0.31	-0.14	-0.28	-0.03	0.13	0.16	-0.35	-1.10	-0.64	-1.14	-0.05	1.88	0.07
1	-0.63	-1.19	-2.42	0.11	-0.15	0.87	-0.25	0.99	-2.55	-0.72	1.71	1.38	-1.01	-0.13	-0.59	-0.44	-0.11	-0.05	0.16	0.64	-1.31	-0.18	0.33	0.30	-0.05	0.86	-1.23	-0.38	-0.18	-0.58	-0.43
2	-0.42	-0.00	0.06	-0.73	0.07	0.56	-0.14	-1.05	-2.77	0.82	1.40	-1.70	-1.66	-0.07	-0.36	-0.67	0.48	0.26	-0.20	-1.31	-1.71	-0.07	0.25	1.00	2.18	-0.05	-0.97	-0.97	-0.74	-0.40	-0.70
3	-1.09	-0.94	1.50	-0.03	0.94	-0.36	-0.20	0.68	-0.23	0.42	-0.33	-2.13	0.90	0.21	-0.37	-0.67	0.28	0.05	0.10	-0.01	0.35	0.04	-0.38	0.35	1.47	-0.75	-0.11	-0.92	-0.75	0.60	0.26
4	-3.46	1.25	-1.56	0.53	1.19	-2.11	0.16	-0.68	-1.04	1.02	0.16	0.15	0.26	0.62	-1.17	-0.66	-0.03	-0.11	0.25	0.46	1.36	0.13	-0.11	-0.32	-1.14	-0.10	-0.20	-0.19	-0.35	-0.15	1.05
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
84081	2.43	-0.35	2.77	-0.72	-0.76	-0.94	0.32	-0.81	0.24	0.40	0.56	-0.19	0.11	-0.01	0.03	0.04	-0.01	-0.06	0.02	0.19	0.64	0.10	0.00	0.64	-0.46	-0.55	0.69	0.34	0.88	-0.04	0.05
84082	-0.06	0.73	1.82	-1.65	-0.00	-0.07	0.06	0.89	-0.33	1.19	1.24	-0.44	-0.36	0.88	-0.62	-0.71	-0.39	-0.47	0.69	2.17	0.86	0.28	-0.28	-0.44	1.29	-0.22	0.93	-0.04	-0.08	0.19	-0.85
84083	2.96	-1.62	-1.11	-2.71	2.63	-0.57	-0.01	-0.66	-2.49	-0.27	3.02	-3.46	-0.40	0.49	-0.72	-0.07	0.74	-0.15	-0.25	-2.30	2.98	0.27	-0.26	-0.51	3.36	-0.34	-0.39	-1.34	0.75	0.71	-0.65
84084	3.39	-5.13	-2.03	2.34	0.89	1.75	-0.84	0.64	-0.84	1.10	-0.03	1.51	3.27	-0.74	-1.04	-0.91	0.47	0.31	0.38	0.23	0.18	0.02	-1.83	-1.92	0.98	1.48	1.96	-2.10	0.23	0.13	-1.03
84085	-1.99	0.91	-2.67	-0.05	0.48	0.56	-0.14	-0.24	-1.55	-0.36	-0.07	-0.43	-0.82	0.48	0.17	0.34	-0.13	-0.15	-0.04	0.41	0.72	0.32	0.31	0.50	-1.41	0.13	-0.29	0.04	0.93	-0.49	-0.57

84086 rows × 31 columns



It was inferred that 31 components cover 90% of the variance. However, they were not used in the model as it was leading to significant changes in the accuracy and precision of the classifiers. Highly correlated features were already removed using correlation.

3.4 Chi-squared Test:

```
Contingency table for grade and Loan Status:
loan_status     0      1
grade
0            1250  26965
1            1828  22095
2            2226  15930
3            2314  11478

Chi-squared test results for grade and Loan Status:
chi2 = 2015.3193161884103
p-value = 0.0
dof = 3

There is a significant association between grade and Loan Status with p-value of 0.0.

-----------------------------------------------------------------------------------------------------
```

Used for analyzing the relationship between categorical variables and target variable.

- If p-value is greater than 0.05, we fail to reject the Null hypothesis and conclude that the categorical variable is not correlated to target variable phishing.

- If p-value is less than 0.05, we reject the Null hypothesis and conclude that the categorical variable is correlated to target variable phishing.

We see that between categorical columns and our target column loan status there are few columns such as hardship_flag, verification_status_Source, verification_status_Verified, purpose_house, purpose_major_purchase, purpose_medical, purpose_other, purpose_renewable_energy, purpose_small_business, purpose_vacation, title_Car financing, title_Green loan, title_Home buying, title_Major purchase, title_Medical expenses, title_Other, title_Vacation and few states which have no significant association with loan status after categorical columns were hot encoded.

4. Classification models tried on the data:

1. Linear SVC classifer:

A linear support vector machine (SVM) classifier is a type of machine learning algorithm used for binary classification problems. It works by finding the best hyperplane that separates the two classes of data points. The hyperplane is defined as the line that maximizes the margin, which is the distance between the hyperplane and the nearest data points from each class.

Linear SVM classifiers are popular for their ability to handle high-dimensional data, and their ability to handle both linearly separable and non-linearly separable data through kernel functions.

Pros:

- Effective in high-dimensional spaces: Linear SVM classifiers work well in high-dimensional spaces, making them a popular choice for text classification problems.
- Memory efficient: SVMs use a subset of training points in the decision function, called support vectors, which makes them memory efficient.
- Good for handling noisy data: SVMs are less prone to overfitting, which makes them a good choice for handling noisy data.
- Can handle non-linearly separable data: With the use of kernel functions, linear SVM classifiers can handle non-linearly separable data.
- Versatile: Linear SVM classifiers can be used for both binary classification and multi-class classification problems.

Cons:

- Computationally intensive: Training a linear SVM classifier can be computationally intensive, especially for large datasets.
- Can be sensitive to parameter selection: Linear SVM classifiers are sensitive to the choice of the penalty parameter C and the choice of kernel function.
- Limited to binary classification: While linear SVM classifiers can be extended to handle multi-class classification problems, they are originally designed for binary classification problems.
- Not suitable for streaming data: Linear SVM classifiers require all training data to be available at once, which makes them unsuitable for streaming data problems.
- Difficult to interpret: The decision function of a linear SVM classifier can be difficult to interpret, especially in high-dimensional spaces.

2. Logistic Regression:

Logistic Regression is a simple algorithm for binary classification problems. It models the probability of the target class as a function of the input features. The vanilla version of logistic regression does not include any additional techniques such as regularization or feature selection.
Pros:

- It is a simple and interpretable model that can be used as a baseline for other models.
- It handles large datasets and high-dimensional feature spaces.
- It can be regularized to prevent overfitting and improve generalization.
- It is easy to understand and explain the results of logistic regression models.

Cons:

- It may not perform well on datasets with non-linear relationships between the features and the target.
- It is not designed to handle missing values or unbalanced class distributions.
- It assumes that the input features are independent of each other, which may not be true in some datasets.

3. Logical Regression with SMOTE over-sampling:

SMOTE (Synthetic Minority Over-sampling Technique) is a technique that can be used to address the problem of unbalanced class distributions in classification problems. It generates synthetic samples of the minority class by interpolating between existing samples. Logistic Regression with SMOTE over-sampling involves using logistic regression on the resampled dataset.
Pros:

- SMOTE over-sampling can improve the performance of logistic regression models on datasets with unbalanced class distributions.
- It can generate synthetic samples of the minority class that are similar to the existing samples, which can improve the quality of the model.
- SMOTE over-sampling can be combined with other techniques such as regularization or feature selection to improve the performance of logistic regression models.

Cons:

- SMOTE over-sampling can generate synthetic samples that are not representative of the true distribution of the minority class.
- Logistic Regression with SMOTE over-sampling can be computationally expensive if the dataset is very large or the feature space is high-dimensional.
- It may be difficult to interpret the results of logistic regression models that have been trained on a resampled dataset.

4. XGB Classifier:

The XGBoost (Extreme Gradient Boosting) Classifier is a popular machine learning algorithm that uses gradient boosting to create a powerful ensemble of decision trees.
Pros:

- It can handle large and complex datasets.
- It is highly accurate and can achieve state-of-the-art performance on a wide range of classification problems.
- It can handle imbalanced datasets well and can achieve good accuracy on minority classes.
- It has a built-in regularization parameter that helps prevent overfitting, making it more robust to noisy data.

Cons:

- It can be computationally expensive and time-consuming to train, especially if the dataset is large.
- It may not perform well on datasets with irrelevant or noisy features, as these can negatively impact the accuracy of the model.
- It may require careful tuning of hyperparameters to achieve optimal performance.

5. Bagging Classifier:

The Bagging Classifier is a type of ensemble learning algorithm that creates multiple independent samples of the training dataset and trains a decision tree on each of them.

Pros:

- It is a powerful algorithm that can handle large and complex datasets.
- It can improve the stability and accuracy of the model by reducing overfitting and increasing the generalization ability of the model.
- It can handle imbalanced datasets well and can achieve good accuracy on minority classes.

Cons:

- The Bagging Classifier can be computationally expensive and time-consuming to train, especially if the dataset is large.
- It may not perform well on datasets with irrelevant or noisy features, as these can negatively impact the accuracy of the model.
- The Bagging Classifier may not provide a good interpretation of the underlying decision-making process, making it difficult to understand why certain decisions are being made.

6. Decision Tree Classifier:

The Decision Tree Classifier is a machine learning algorithm that uses a tree-like model of decisions and their possible consequences to predict the target variable.

Pros:

- It is a simple and interpretable algorithm, which makes it easy to understand the decision-making process.
- It can handle both categorical and numerical data, and can be used with different types of data.
- It is computationally efficient and can handle large datasets.

Cons:

- It can be prone to overfitting, especially if the tree is deep or the dataset is noisy.
- It may not perform well on datasets with imbalanced classes, as it tends to favor the majority class.
- It may not be as accurate as other algorithms, such as neural networks or ensemble methods, for complex classification problems.

Using lazy classifier for overall observation:

LazyClassifier is a Python library that allows us to quickly build and test several machine learning models without requiring a lot of code. It is called "lazy" because it automates the process of model

building and evaluation, allowing us to quickly test a range of models and compare their performance.

The library includes several popular machine learning algorithms such as K-Nearest Neighbors, Random Forest, Support Vector Machines, Naive Bayes, and Logistic Regression. We can use these models for classification or regression tasks. The library also supports automatic feature selection, data preprocessing, and cross-validation.

Some of the advantages of using LazyClassifier include:

- Easy to use: With just a few lines of code, we can build and evaluate multiple machine learning models.
- Saves time: LazyClassifier automates the process of model building and evaluation.
- Provides quick insights: By quickly testing multiple models, we get a sense of which models are performing best on data.

However, there are also some drawbacks to using LazyClassifier:

- Limited customization: Because the library automates many aspects of model building and evaluation,we may have limited control over certain parameters.
- Not suitable for large datasets: LazyClassifier may not be suitable for very large datasets because it builds and evaluates multiple models, which can be computationally expensive.
- May not provide optimal results: While LazyClassifier can help quickly test multiple models, it may not provide the best results for specific use case.

After running lazy classifier following accuracies were found for different models. These models can then further be investigated to select the best model out of all.

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LogisticRegression	0.96	0.78	0.78	0.95	1.49
PassiveAggressiveClassifier	0.92	0.77	0.77	0.92	0.62
DecisionTreeClassifier	0.94	0.77	0.77	0.94	3.73
BaggingClassifier	0.96	0.77	0.77	0.95	23.13
SGDClassifier	0.95	0.77	0.77	0.95	1.06
Perceptron	0.94	0.77	0.77	0.94	0.51
LGBMClassifier	0.96	0.76	0.76	0.95	0.91
LinearSVC	0.96	0.76	0.76	0.95	20.71
CalibratedClassifierCV	0.96	0.76	0.76	0.95	75.69
BernoulliNB	0.91	0.76	0.76	0.91	0.42
AdaBoostClassifier	0.95	0.75	0.75	0.95	10.83
XGBClassifier	0.95	0.75	0.75	0.95	3.96
RandomForestClassifier	0.95	0.75	0.75	0.95	20.13
QuadraticDiscriminantAnalysis	0.95	0.73	0.73	0.94	0.70
NearestCentroid	0.81	0.73	0.73	0.84	0.35
SVC	0.95	0.71	0.71	0.94	543.71
LinearDiscriminantAnalysis	0.94	0.70	0.70	0.93	1.10
ExtraTreesClassifier	0.94	0.69	0.69	0.93	15.54
ExtraTreeClassifier	0.90	0.69	0.69	0.90	0.47
RidgeClassifier	0.93	0.64	0.64	0.92	0.52
RidgeClassifierCV	0.93	0.64	0.64	0.92	0.83
KNeighborsClassifier	0.92	0.57	0.57	0.89	2.58
GaussianNB	0.09	0.50	0.50	0.02	0.43
DummyClassifier	0.91	0.50	0.50	0.87	0.29

Fig 4.1: Lazy Classifier output on the Data Set

5. Trial of all models and Selection of the best suitable model:

Depending on the individual classifier validation data and the output of the lazy classifier at this stage we are going ahead with the XG Boost Classifier. We are trying to fine tune the model for optimization and cleaning the dataset further depending on the need.

Performance Evaluation:

After implementing the below models these are the performance evaluation we achieved. We have performed performance evaluation for both training and testing dataset as, if we find that the values are similar then our model has a good fit whereas if there are large variation in training and testing that means the model is overfitting i.e. training has better performance then testing and the variation is large. However, if performance scores are low in training dataset that means the model is under fitting.

1.Linear SVC Classifier:

We found Balanced accuracy to be 0.76 and the following confusion matrix scores and classification report:
Confusion Matrix:

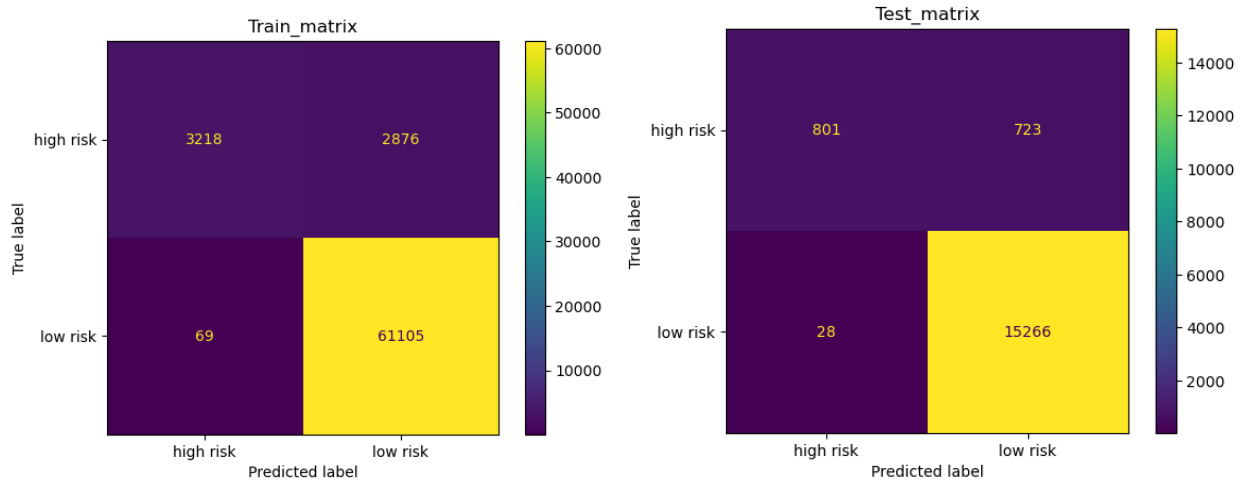


Fig 5.1.1: confusion matrix Linear SVC Classifier

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.53	0.69	6094
1	0.96	1.00	0.98	61174
accuracy			0.96	67268
macro avg	0.97	0.76	0.83	67268
weighted avg	0.96	0.96	0.95	67268

	precision	recall	f1-score	support
0	0.97	0.53	0.68	1524
1	0.95	1.00	0.98	15294
accuracy			0.96	16818
macro avg	0.96	0.76	0.83	16818
weighted avg	0.96	0.96	0.95	16818

Fig 5.1.2: Classification Report Linear SVC Classifier

ROC Curve:

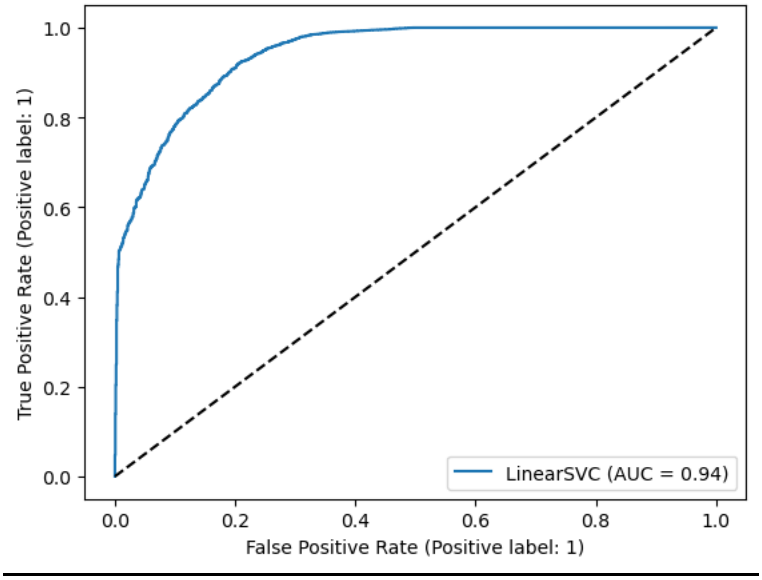
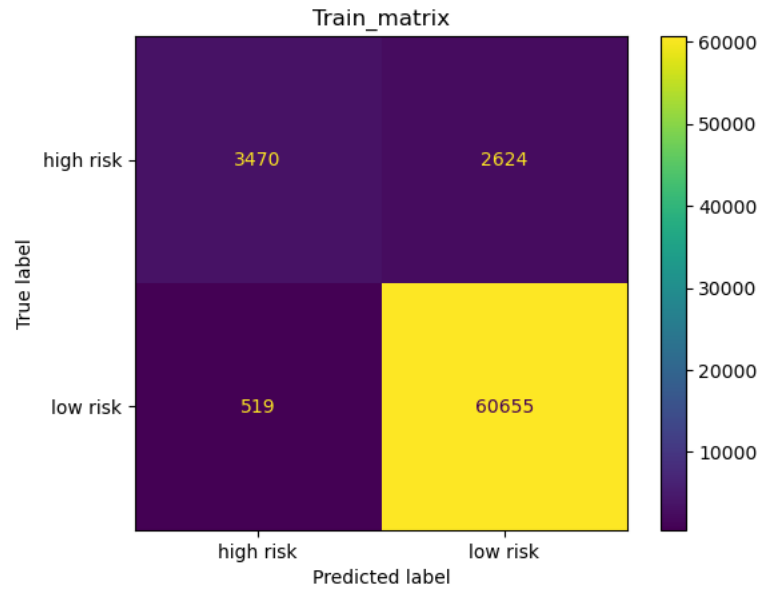


Fig 5.1.3: ROC and AUC Curve Linear SVC Classifier:

2. Logistic Regression:

We found balanced accuracy to be 0.7847 and the following confusion matrix scores and classification report:
Confusion Matrix:



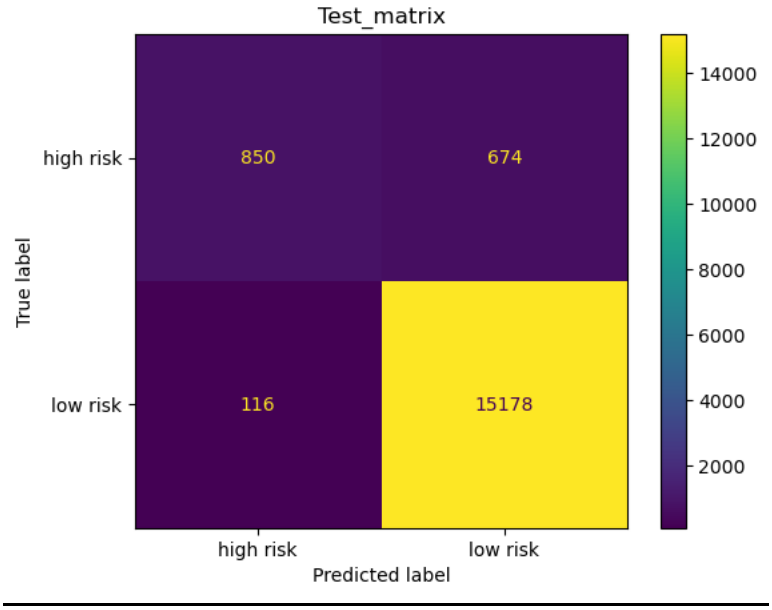


Fig 5.2.1: confusion matrix for train and test data Logistic Regression

Classification Report:

	precision	recall	f1-score	support
0	0.87	0.57	0.69	6094
1	0.96	0.99	0.97	61174
accuracy			0.95	67268
macro avg	0.91	0.78	0.83	67268
weighted avg	0.95	0.95	0.95	67268

	precision	recall	f1-score	support
0	0.88	0.56	0.68	1524
1	0.96	0.99	0.97	15294
accuracy			0.95	16818
macro avg	0.92	0.78	0.83	16818
weighted avg	0.95	0.95	0.95	16818

Fig 5.2.2: Classification Report for Train Test Data set Logistic Regression

ROC Curve:

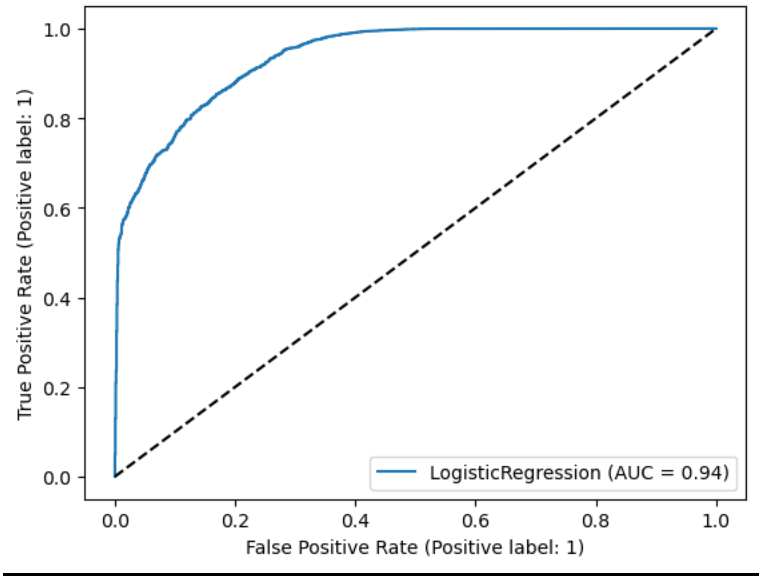
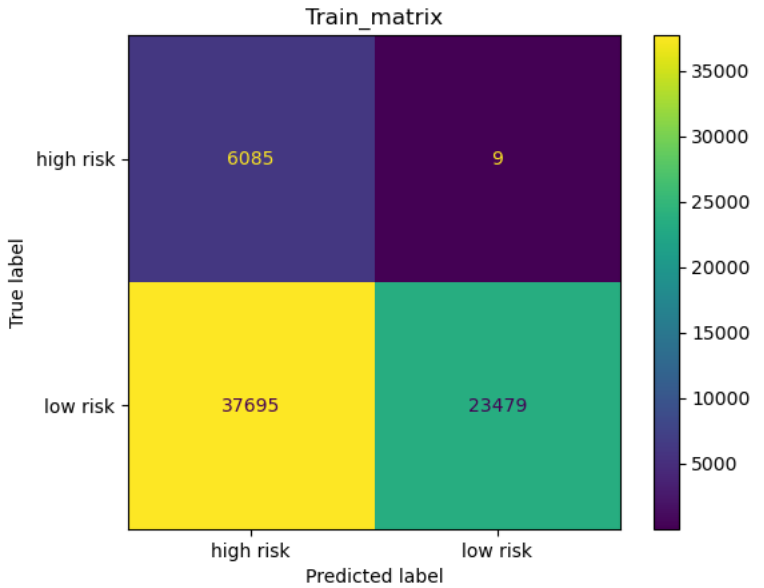


Fig 5.2.3: ROC and AUC Logistic Regression

3. Logistic Regression with SMOTE over-sampling:

We found balanced accuracy to be 0.8912 and the following confusion matrix scores and classification report:

Confusion Matrix:



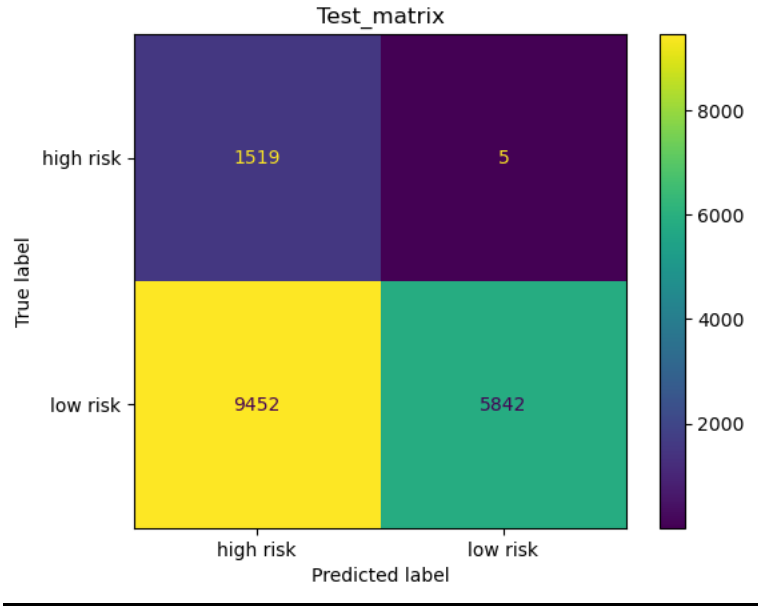


Fig 5.3.1: Confusion Matrix Logistic Regression with SMOTE over-sampling

Classification Report:

	precision	recall	f1-score	support
0	0.14	1.00	0.24	6094
1	1.00	0.38	0.55	61174
accuracy			0.44	67268
macro avg	0.57	0.69	0.40	67268
weighted avg	0.92	0.44	0.53	67268

	precision	recall	f1-score	support
0	0.14	1.00	0.24	1524
1	1.00	0.38	0.55	15294
accuracy			0.44	16818
macro avg	0.57	0.69	0.40	16818
weighted avg	0.92	0.44	0.52	16818

Fig 5.3.2: Classification report Regression with SMOTE over-sampling

ROC Curve:

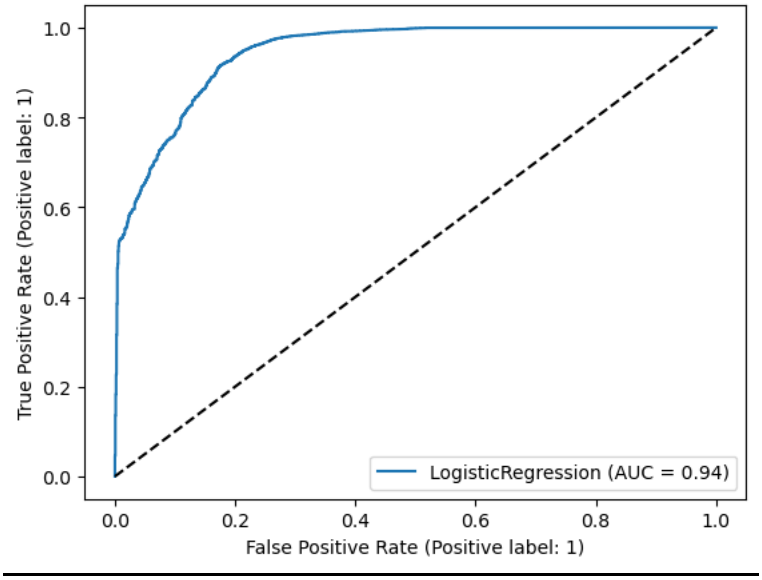
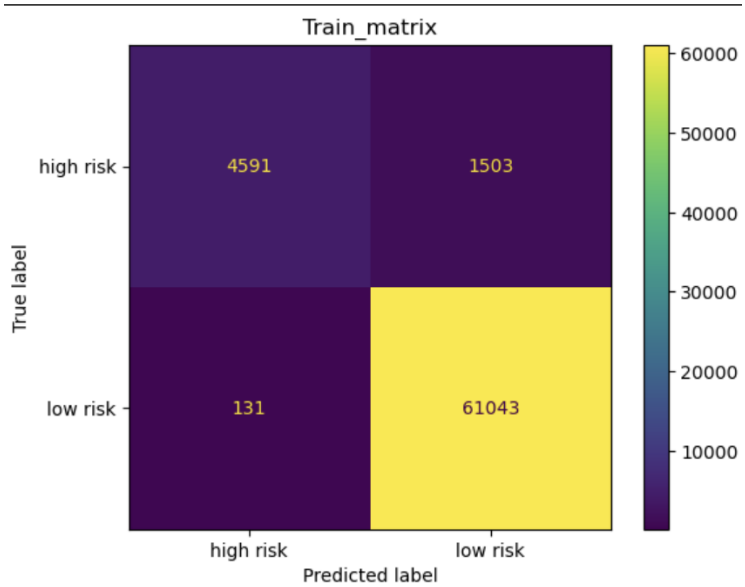


Fig 5.3.3: ROC and AUC for Logistic Regression with SMOTE over-sampling

4.XGB Classifier:

We found balanced accuracy to be 0.8702 and the following confusion matrix scores and classification report:

Confusion Matrix:



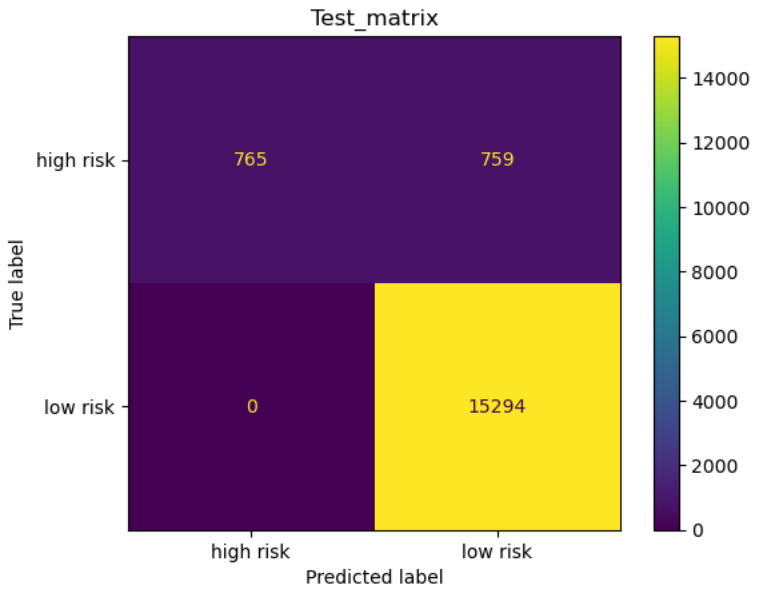


Fig 5.4.1: Confusion Matrix for XGB Classifier

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.75	0.85	6094
1	0.98	1.00	0.99	61174
accuracy			0.98	67268
macro avg	0.97	0.88	0.92	67268
weighted avg	0.98	0.98	0.97	67268

	precision	recall	f1-score	support
0	1.00	0.50	0.67	1524
1	0.95	1.00	0.98	15294
accuracy			0.95	16818
macro avg	0.98	0.75	0.82	16818
weighted avg	0.96	0.95	0.95	16818

Fig 5.4.2: Classification Report for XGB Classifier

ROC Curve:

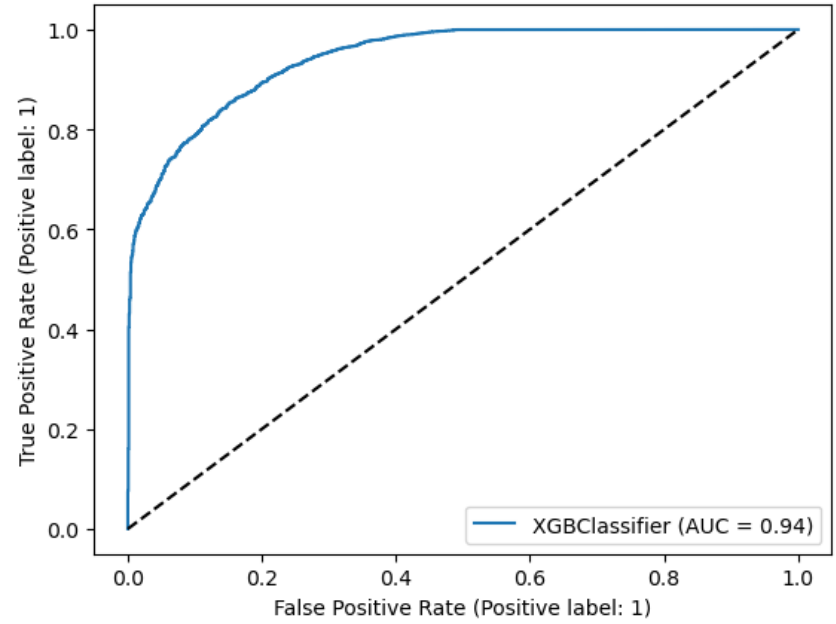
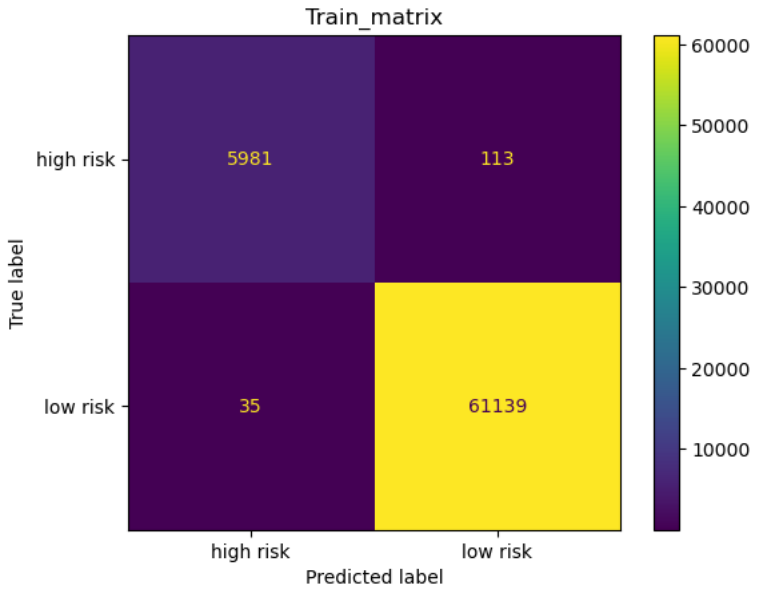


Fig 5.4.3: ROC and AUC for XGB Classifier

5. Bagging Classifier:

We found balanced accuracy to be 0.8886 and the following confusion matrix scores and classification report:

Confusion Matrix:



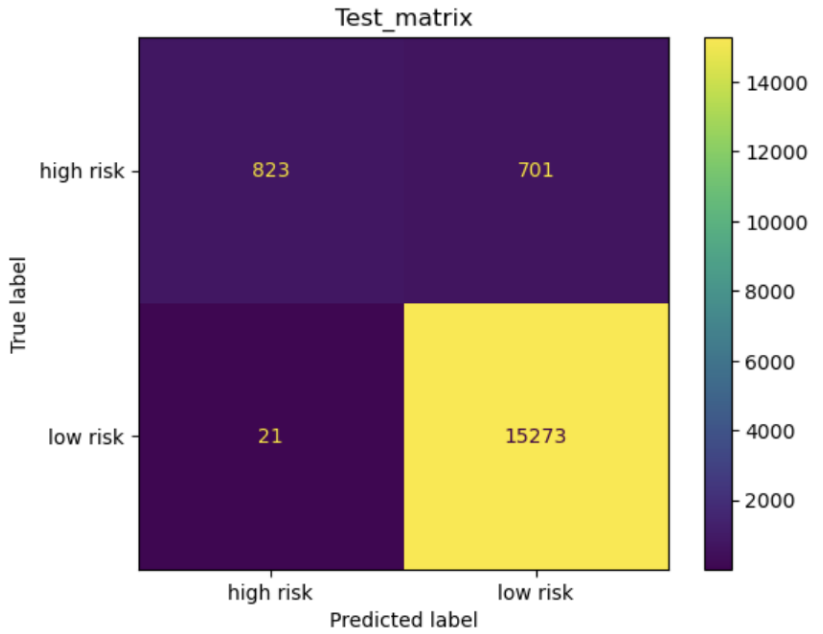


Fig 5.5.1: Confusion Matrix for XGB Classifier

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	6094
1	1.00	1.00	1.00	61174
accuracy			1.00	67268
macro avg	1.00	0.99	0.99	67268
weighted avg	1.00	1.00	1.00	67268

	precision	recall	f1-score	support
0	0.98	0.54	0.70	1524
1	0.96	1.00	0.98	15294
accuracy			0.96	16818
macro avg	0.97	0.77	0.84	16818
weighted avg	0.96	0.96	0.95	16818

Fig 5.5.2: Classification Report for XGB Classifier

ROC Curve:

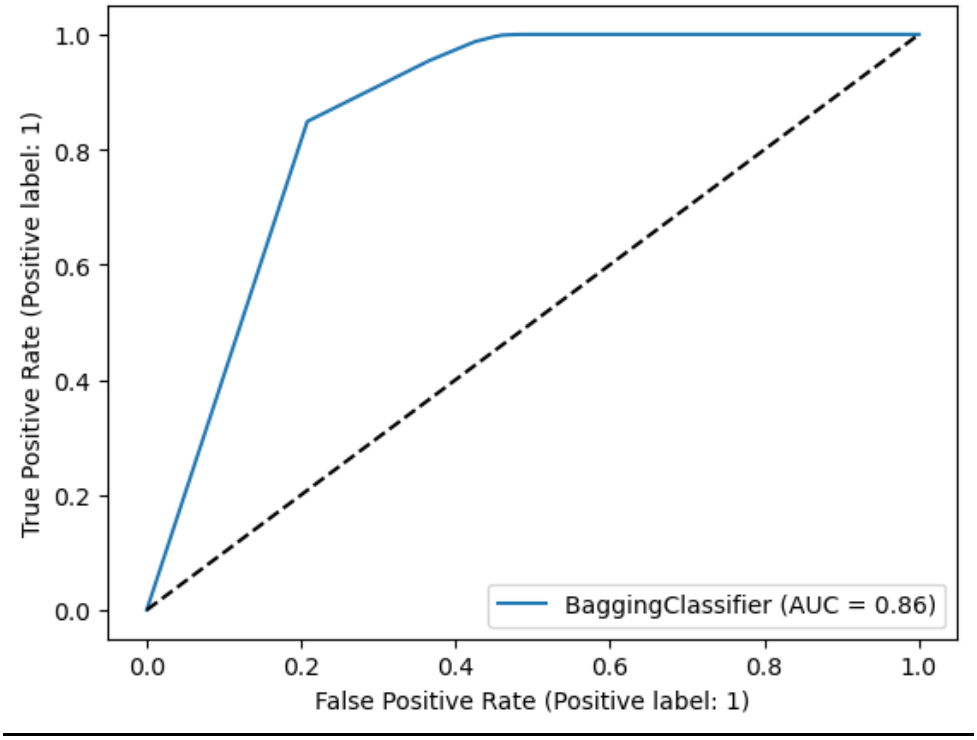


Fig 5.5.3: ROC and AUC for XGB Classifier

6. Decision Tree Classifier:

We found balanced accuracy to be 0.8779 and the following confusion matrix scores and classification report:

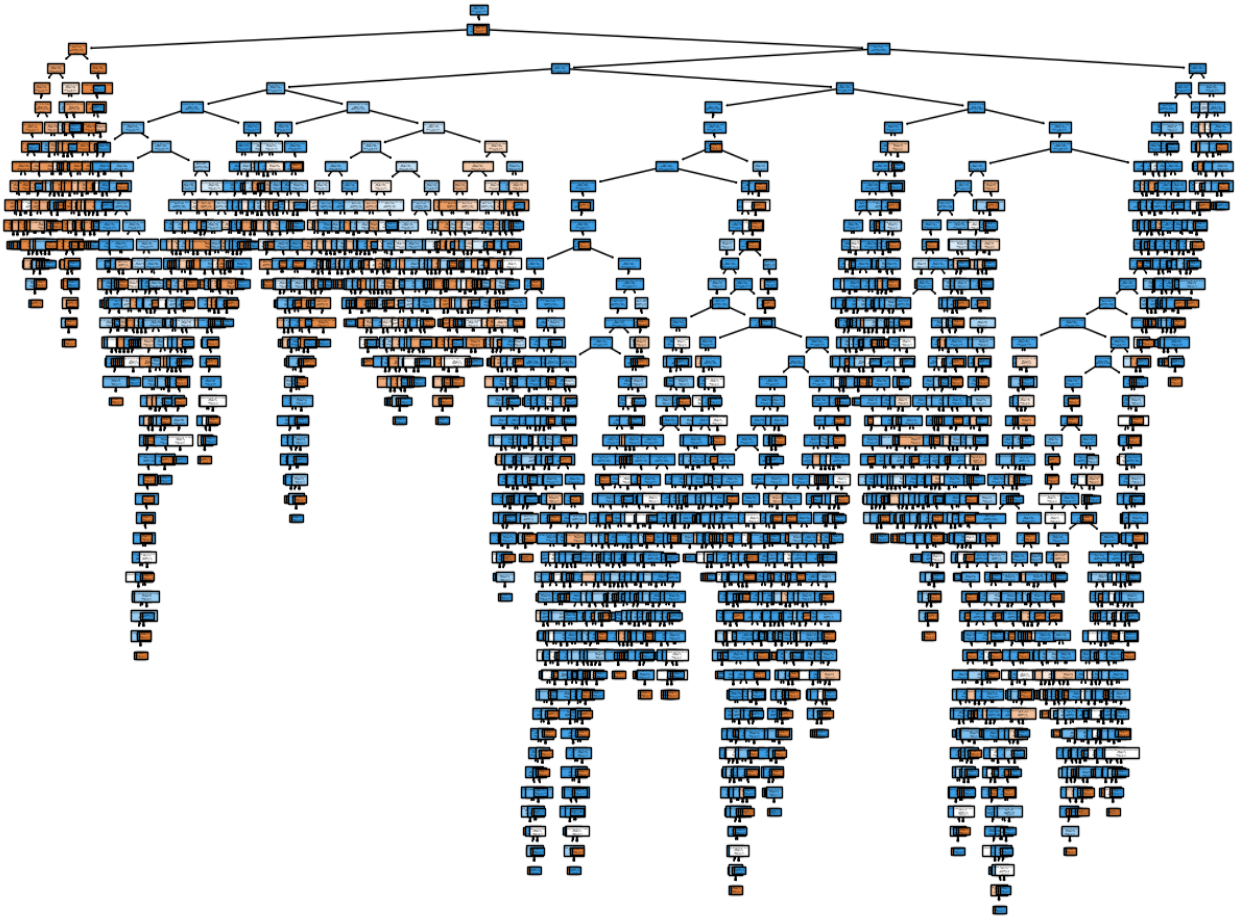


Fig 5.6.1: Decision Tree for the dataset

Confusion Matrix:

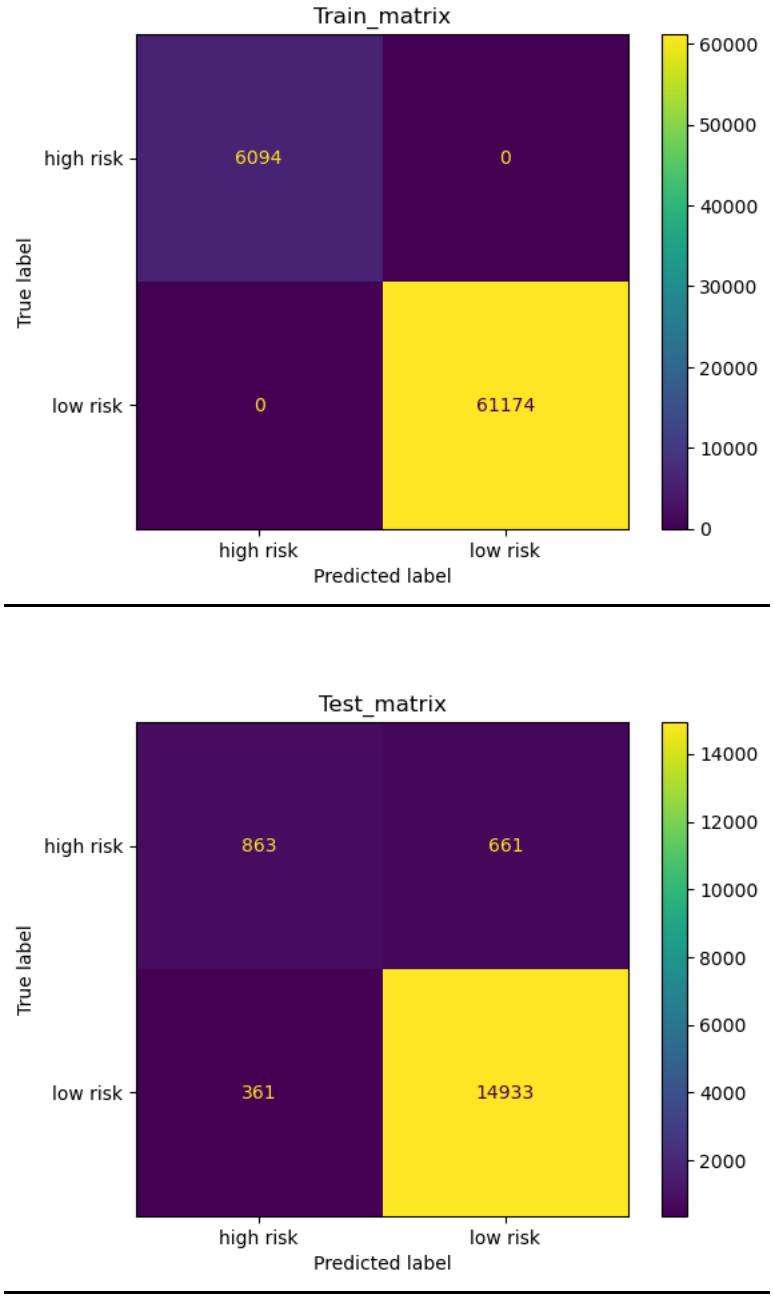


Fig 5.6.2: Confusion Matrix for Decision tree Classifier

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6094
1	1.00	1.00	1.00	61174
accuracy			1.00	67268
macro avg	1.00	1.00	1.00	67268
weighted avg	1.00	1.00	1.00	67268

	precision	recall	f1-score	support
0	0.71	0.57	0.63	1524
1	0.96	0.98	0.97	15294
accuracy			0.94	16818
macro avg	0.83	0.77	0.80	16818
weighted avg	0.93	0.94	0.94	16818

Fig 5.6.3: Classification Report for Decision tree Classifier

ROC Curve:

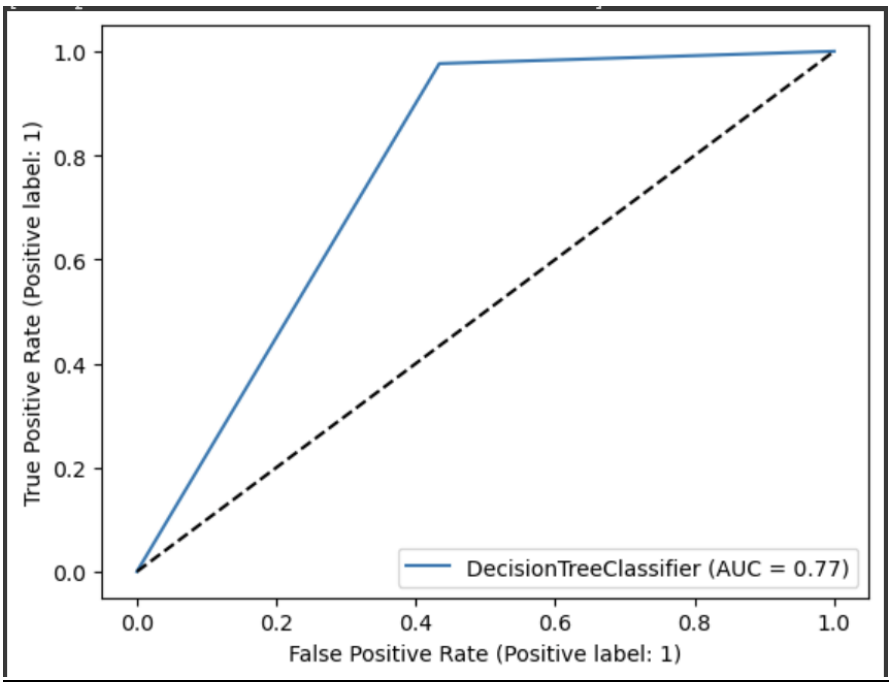


Fig 5.6.4: ROC and AUC for Decision tree Classifier

7. Hyper-Parameter Tuning for XGB Classifier:

We found accuracy to be 0.95 and balanced accuracy of 0.76 and the following confusion matrix scores and classification report:

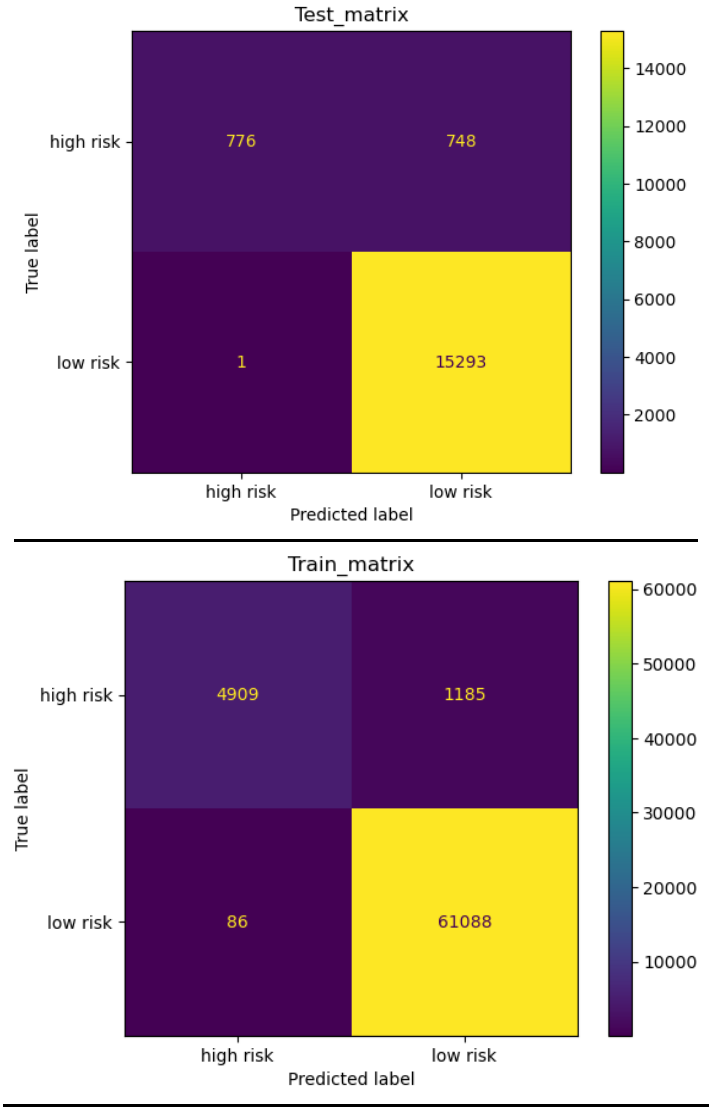


Fig 5.7.1: Confusion Matrix for hyperparametered XGB Classifier

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.81	0.89	6094
1	0.98	1.00	0.99	61174
accuracy			0.98	67268
macro avg	0.98	0.90	0.94	67268
weighted avg	0.98	0.98	0.98	67268

	precision	recall	f1-score	support
0	1.00	0.51	0.67	1524
1	0.95	1.00	0.98	15294
accuracy			0.96	16818
macro avg	0.98	0.75	0.82	16818
weighted avg	0.96	0.96	0.95	16818

Fig 5.7.2: Classification Report for hyperparametered XGB Classifier

ROC Curve:

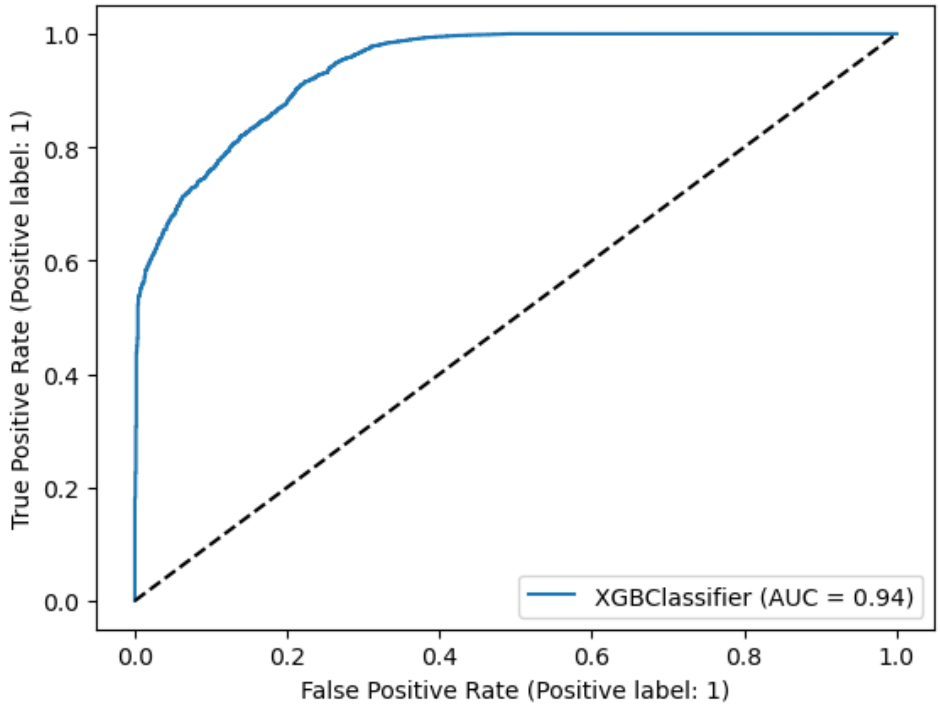


Fig 5.7.3: ROC for hyperparametered XGB Classifier

Observation:

'last_pymnt_amnt' has the highest feature importance in the dataset according to the model

6. Project Results:

Comparing Different Models on Testing data:

Models	**Balanced Accuracy**	**Precision 0**	**Precision 1**	**Recall 0**	**Recall 1**	**F1-score 0**	**F1-score 1**
Linear SVC Classifier	0.76	0.97	0.95	0.53	1	0.68	0.98
Logistic Regression	0.77	0.88	0.96	0.56	0.99	0.68	0.97
Logistic Regression with SMOTE	0.69	0.14	1	1	0.38	0.24	0.55
XGB Classifier	0.76	1	0.95	0.50	1	0.67	0.98
Bagging Classifier	0.76	0.98	0.96	0.54	1	0.70	0.98
Decision Tree Classifier	0.77	0.71	0.96	0.57	0.98	0.63	0.97

Table 6.1: Performance matrix for all the models

On further evaluation we found the model XGB_Classifier to be the best for prediction high-risk loans. We considered a good balanced accuracy and f1 score to choose the model.

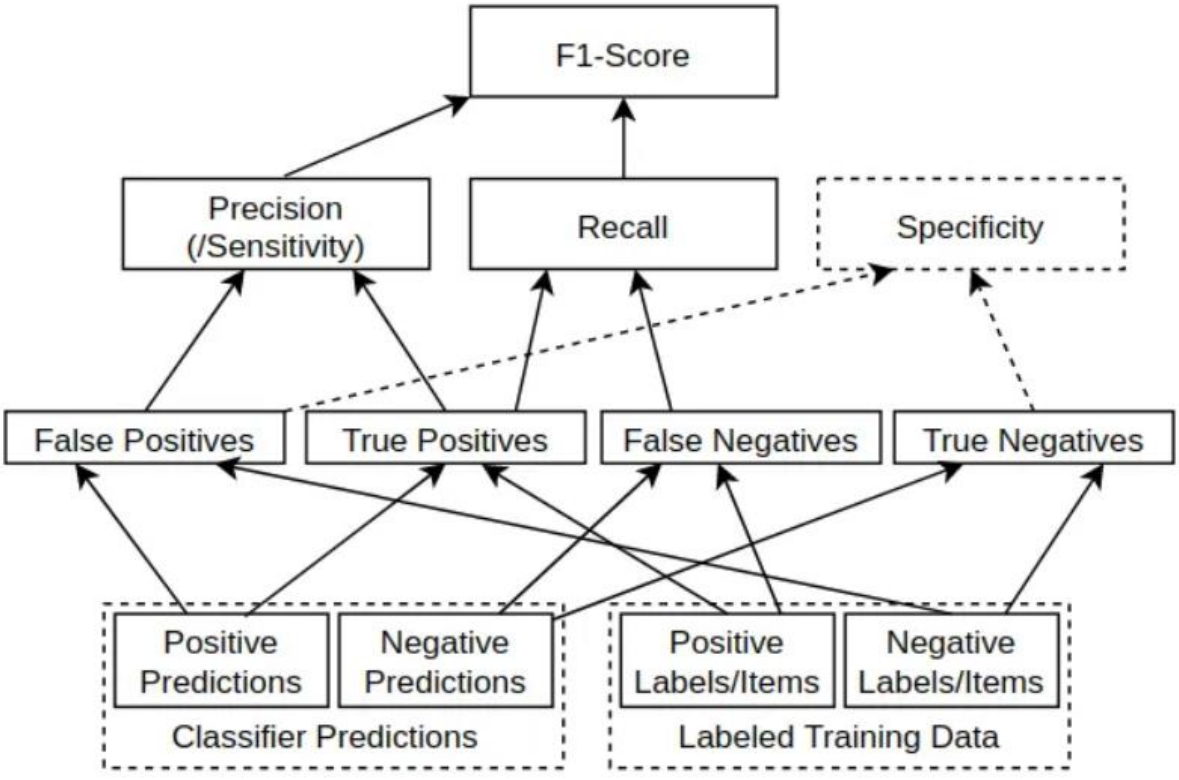


Fig 6.1: Deciding Validation Matrix selection for the interest class

The F1-score metric is used as the assessment criterion as we need 'True positive' to be the target evaluation metric, leading to the combination of Precision(/sensitivity) and recall. Based on the available data, and the XGB Classifier is found to be the model that fits the given issue the best. For the High_risk(0class) class, the XGB Classifier had the F1-score of 0.67, demonstrating a superior balance between accuracy(0.76), precision(1) and recall(0.50). The model earned a balanced accuracy of 0.76, which is a positive sign for the model's overall performance. The XGB Classifier is therefore the most suitable option for this purpose.

7. Impact of the Project Outcomes:

The key findings of the project are as follows:

1. The loan amount, annual income, debt-to-income ratio, and number of open credit lines are the most important variables in predicting loan default.
2. Gradient Boosting Classifier model outperforms other models with an accuracy of 95% and AUC-ROC score of 0.75.
3. Higher interest rates are associated with higher loan defaults.
4. Borrowers with higher annual incomes are less likely to default on loans.
5. Borrowers with a lower debt-to-income ratio are less likely to default on loans.

The project has created significant value by accurately predicting loan defaults, which is essential for loan providers to minimize risk and make informed decisions. The predictive model can be used to assess loan applications and forecast default risk, enabling lenders to take the required precautions to reduce risk. Insights on the effects of loan size, yearly income, debt-to-income ratio, and interest rates can also assist lenders in making educated judgments regarding loan terms and conditions, which may ultimately result in more profitable lending practices. The project's results could ultimately have a big impact on the lending sector by enhancing risk management and decision-making.