Specialist Diploma in Applied Artificial Intelligence

Post Diploma Certificate in
AI Foundation & Machine Learning

ITI105
Machine Learning Project

Group Final Project Report

Group 10

1944757V | Babu Logiah Arun Kumar
1995149W | Karupaiyan Kalaivani
6422706H | Nursharinah Binte Sohaimi

Table of Contents

1. Machine Learning Formulation

This section briefly describes the Machine Learning (ML) framing of the problem. It states what the input(s) and output(s) are, and the algorithms used for the models.

Loan processing and approval in financial institutions is a major issue and bottleneck problem. Integrating ML technologies can make a significant change in this important business activity.

We want to develop an efficient loan default predicting system that will become a game-changer in loan processing and approval. This system can be used by all lending institutions to have a fair and successful loan approval process with the least minimal ratio value of lean defaulters. Thus, integrating the system with emerging technologies like ML can help perform time-intensive tasks and make problem-solving more efficient.

1.1 ML Problem Framing

This project aims to build a predictive model using a supervised method to categorize a loan application as a loan defaulter or not. Relevant personal data and transactions are collected from the historical loan default system. This information will be used to track when the lending is processed and determine the criteria for default loans. It then shall determine the criteria for default loans and forecast possible loan defaults with accuracy.

Figure 1: Model representation of balancing risks & profitability

To avoid financial losses from defaults, the ML solution shall:

- Identify customers who will successfully repay their loans.
- Minimize the default risk of borrowers ending up not repaying the loans.
- Quickly determine loan risks without high costs.

1.2 Input & Output

This project uses a financial dataset specially focused on loan defaults which has detailed information about borrowers and their loan status. The data was likely collected from various financial institutions. It includes variables such as borrower demographics, loan characteristics, and repayment history. This comprehensive dataset is highly relevant to our problem of predicting loan defaults, as it contains the necessary features to train and evaluate our ML model. By using this dataset, we can develop a robust solution to identify high-risk borrowers and mitigate potential financial losses.

The output of the ML solution should be self-explainable and be able to balance between risks and profits. The model will be a label that:

- Assess borrower risk.
- Predict customers that either would default on their loan or not.
- Find high-risk borrowers.
- Justifies the adjustment of loan terms and conditions.

1.3 Model Used

Collectively, the team has explored the various ML algorithms as listed below:

1. K-Nearest Neighbors
2. Logistic Regression
3. Decision Tree
4. Random Forest
5. Support Vector Machine (SVM)
6. Naïve Bayes
7. Stacking Classifier

2. Initial Dataset Review

The raw dataset used in this project is called Loan_Default.csv which is provided by M Yasser on Kaggle. It has 148,670 entries of financial transactions with 34 features. Note that the target label shall be the "Status" column which has a binary value.

Annex A shows the initial conditions of the dataset before it goes through the necessary preprocessing steps for data analysis. This information gave a sense of preliminary understanding of the dataset. Figure 2 shows a glossary of the features found in the dataset. From this general data understanding, each team member took their respective processing steps in attempts to clean the data for ML.

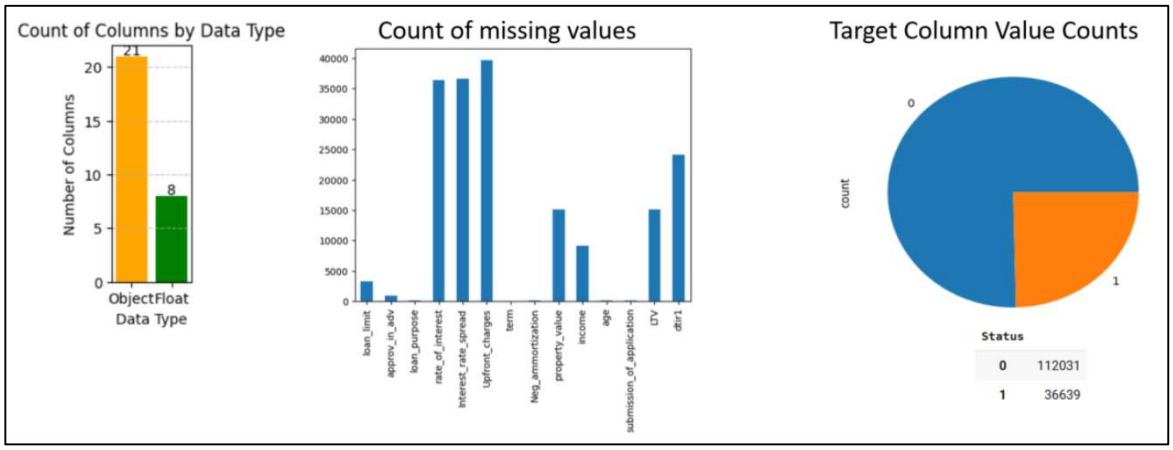


Figure 2: Summary information on the Loan Default dataset

S/N	Features	Description
1	ID	client loan application id
2	year	year of loan application
3	loan_limit	indicates whether the loan is conforming (cf) or non conforming (ncf)
4	Gender	gender of the applicant (male, female, joint, sex not available)
5	approv_in_adv	indicates whether the loan was approved in advance (pre, nopre)
6	loan_type	type of loan (type1, type2, type3)
7	loan_purpose	purpose of the loan (p1, p2, p3, p4)
8	Credit_Worthiness	credit worthiness (l1, l2)
9	open_credit	indicates whether the applicant has any open credit accounts (opc, nopc)
10	business_or_commercial	indicates whether the loan is for business/commercial purposes (ob/c - business/commercial, nob/c - personal)
11	loan_amount	amount of money being borrowed
12	rate_of_interest	interest rate charged on the loan
13	Interest_rate_spread	difference between the interest rate on the loan and a benchmark interest rate
14	Upfront_charges	initial charges associated with securing the loan
15	term	duration of the loan in months
16	Neg_ammortization	indicates whether the loan allows for negative ammortization (neg_amm, not_neg)
17	interest_only	indicates whether the loan has an interest-only payment option (int_only, not_int)

S/N	Features	Description
18	lump_sum_payment	indicates if a lump sum payment is required at the end of the loan term (lpsm, not_lpsm)
19	property_value	value of the property being financed
20	construction_type	type of construction (sb - site built, mh - manufactured home)
21	occupancy_type	occupancy type (pr - primary residence, sr- secondary residence, ir - investment property)
22	Secured_by	specifies the type of collateral securing the loan (home, land)
23	total_units	number of units in the property being financed (1U, 2U, 3U, 4U)
24	income	applicant's annual income
25	credit_type	applicant's type of credit (CIB - credit information bureau , CRIF - CIRF credit information bureau, EXP - experian , EQUI - equifax)
26	Credit_Score	applicant's credit score
27	co-applicant _credit_type	co-applicant's type of credit (CIB - credit information bureau EXP - experian)
28	age	the age of the applicant
29	submission_of_application	indicates how the application was submitted (to_inst - to institution, not_inst - not to institution)
30	LTV	loan-to-value ratio, calculated as the loan amount divided by the property value
31	Region	geographic region where the property is located (North, south, central, North-East)
32	Security_Type	type of security or collateral backing the loan (direct, indirect)
33	Status	indicates whether the loan has been defaulted (1) or not (0)
34	dtir1	debt-to-income ratio

Figure 3: Glossary of the Loan Default dataset

Note that upon completion of the data preprocessing steps, each team member has variations regarding their cleaned datasets. From here onwards, we assume that there are three versions of cleaned datasets. Each team member then separately worked on their modeling and experimentations. At the end, the members nominated their best model for general comparison. The final chosen model with the best performance will then be chosen for deployment.

3. Data Preparation

This section will discuss a compilation of the common and best techniques used for the preprocessing steps, Exploratory Data Analysis (EDA), and feature engineering done by the team. Details for individual EDA & feature engineering parts can be referred to at the annexes.

3.1 Preprocessing Steps

Below are the general steps that the group has taken to clean and prepare the dataset:

1. Numerical missing value imputation using k-NN/Simple imputer
2. Categorical missing value imputation using most frequent observations or constant
3. Drop columns for non-useful features (column: ID, year).
4. Detects & removes outliers more than value 100 (column: LTV).
5. General detection & removal of outliers for whole dataset using Z-score/IQR method
6. Drop columns with high correlation (using V Cramer's) for Categorical Features.
7. Drop columns with high correlation for Numerical Features.
8. Combine low frequency categorical values for affected categorical.

9. Use LabelEncoder for columns assumed to be of Ordinal Data.
10. Use One-hot Encoder for columns assumed to be of Nominal Data.
11. Apply MinMaxScaler to standardize range from 0 to 1.

3.2 Exploratory Data Analysis

Univariate Analysis

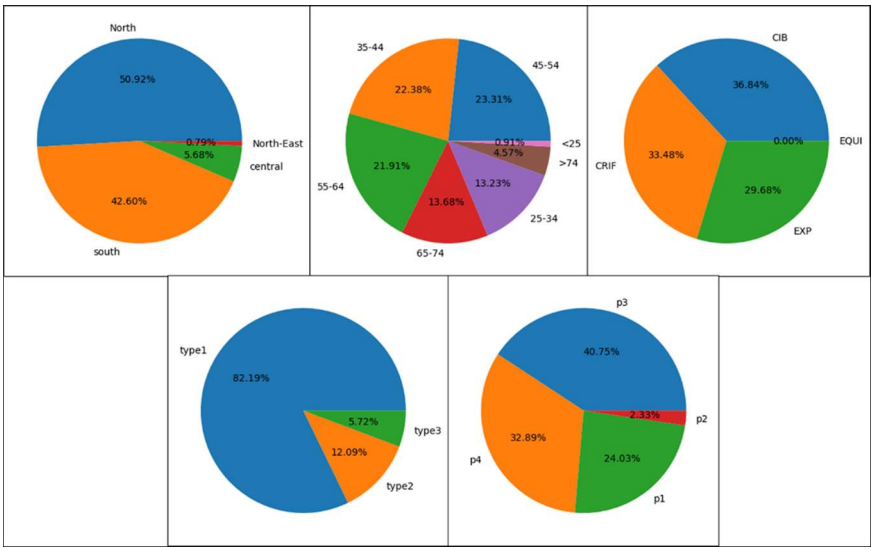


Figure 4: Distribution of some categorical features

Below is a list of descriptives of insights based on figure 4:

- Region: Most clients reside in North & South.
- Age: High percentage of clients taking loans are between 35-64 years old.
- Credit Type: Most clients are assessed based on the Credit Information Bureau, CRIF Credit Bureau or Experian standards.
- Loan Type: Most clients took up type1 loans.
- Loan Purpose: Less clients have p2 purpose while rest of the purpose are generally well distributed.

Bivariate Analysis

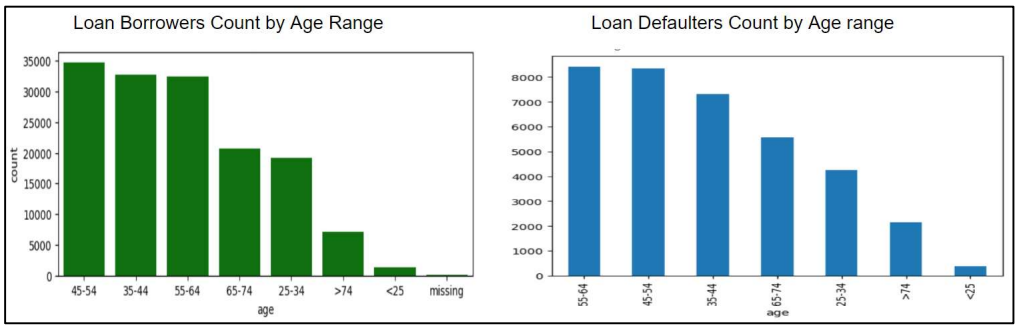


Figure 5: Age range by Loan Borrowers & Loan Defaulters

Most borrowers are between the ages of 45 and 64, and a significant proportion of loan defaulters are also within this age group.

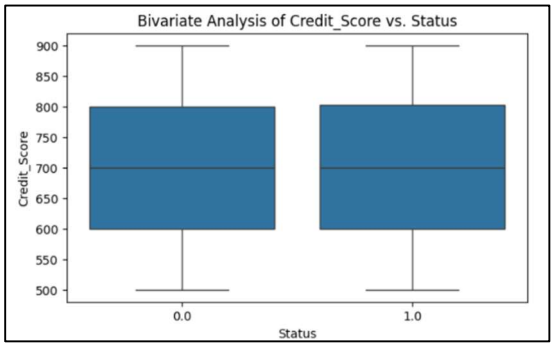


Figure 6: Analysis on Credit Score vs Status

Credit scores are equally distributed with the target classes.

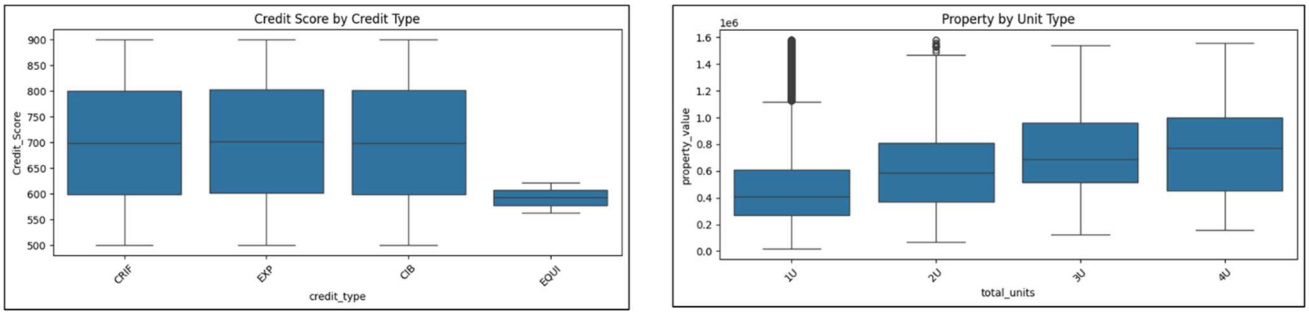


Figure 7: Credit Score by Credit Type & Property by Unit Type

Those with Equifax credit type have better scoring than others. While credit type from credit information bureau, CRIF credit information bureau, Experian generally has similar credit score. Also, the bigger the property (more units), the higher the property value.

Correlation Matrix

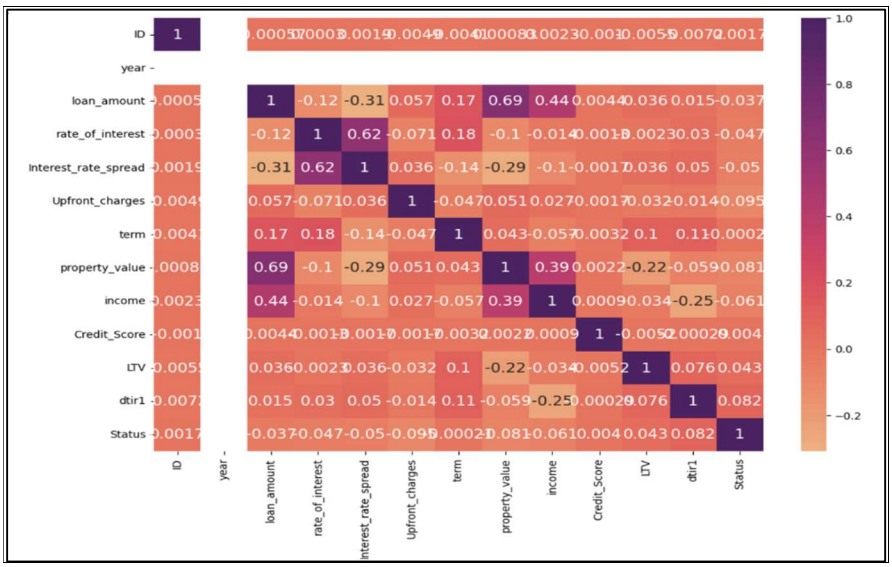


Figure 8: Correlation matrix of numerical features

Drop high correlated features for example rate_of_interest, Upfront_charges.

3.3 Train & Test Dataset Splitting

The dataset has an imbalanced characteristic, where one class is much more frequent than the other. It has been handled by using the stratification technique to ensure that both classes are proportionally represented in training and test datasets. This would lead to more reliable evaluation metrics during modeling and experimentations.

```
Training set distribution:          Test set distribution:
Status                              Status
0    74411                          0    31891
1    24475                          1    10490
Name: count, dtype: int64           Name: count, dtype: int64
```

Figure 9: Row counts of train & test dataset after stratification

4. Modeling & Experiments

This section briefly describes the main highlights of modeling and experimentations that was accomplished by each member.

4.1 Arun

1. Logistic Regression

   While a ROC score of 0.75 is good but not excellent. To improve the model further feature engineering, parameter tuning is tried. The model's performance, as measured by accuracy, is stable and does not change with different hyperparameter settings. This implies that the model is already performing at its best possible level with the given feature set and data.

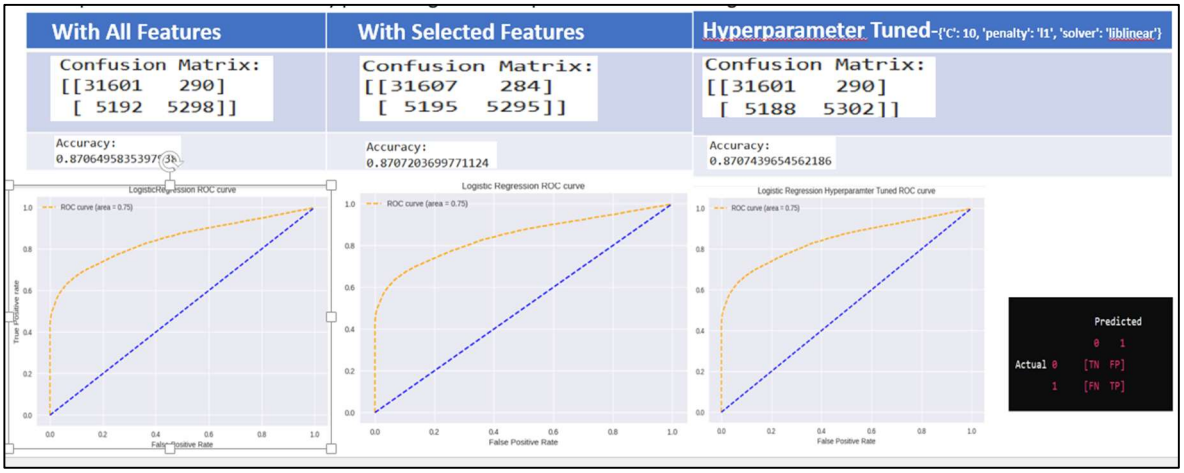


Figure 10: Logistic Regression experiments

2. SVM Hyperparameter Tuned

   With a low gamma, the model does not capture the complexities in the data. Thus, potentially leading to underfitting. With high C values, the model will classify all training examples correctly and low regularization.

Hyperparamters	C:0.1,gamma:0.1	C:0.1,gamma:0.0001	C:10,gamma:0.1	C:10,gamma:0.0001
Accuracy	0.95325	0.75248	0.98461	0.8631
Precision	0.9556	0.376	0.97478	0.9056
Recall	0.9174	0.5	0.98459	0.7297
F1-Score	0.9345	0.42	0.97956	0.7717
	Confusion Matrix: [[31519 372] [1609 8881]]	Confusion Matrix: [[31891 0] [10490 0]]	Confusion Matrix: [[31401 490] [162 10328]]	Confusion Matrix: [[31696 195] [5606 4884]]

Figure 11: SVM experiments

3. Stacking Classifier

   High Accuracy and F1-Score: This model performs exceptionally well, with very high accuracy and F1-score, suggesting a balanced performance across classes.

   Precision and Recall: Both are high, indicating that the model is both good at finding positive instances and minimizing false positives.

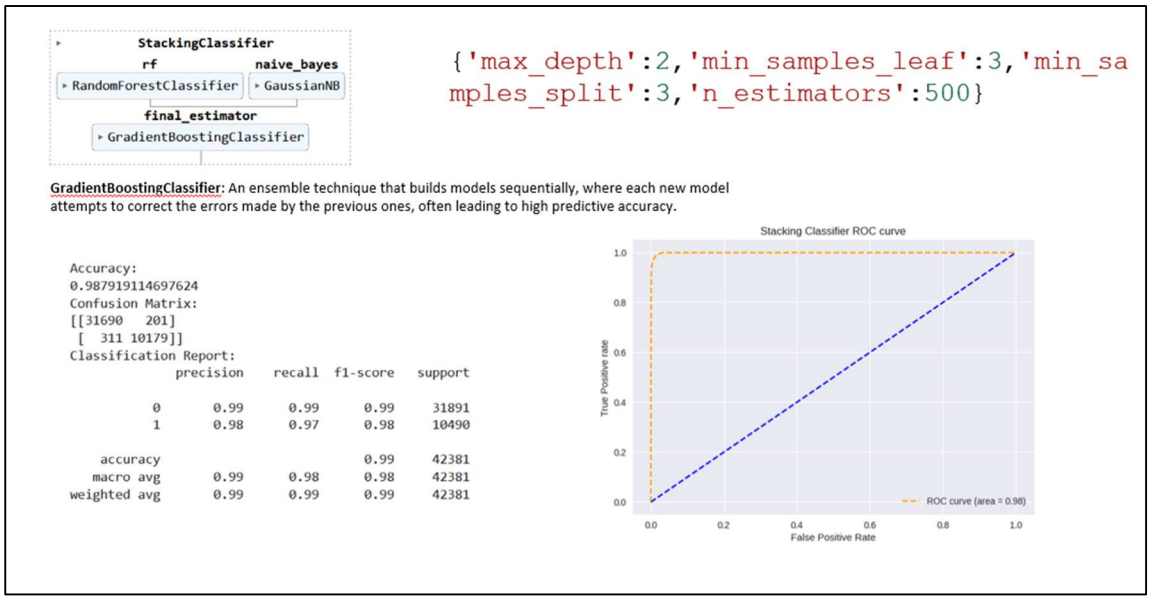


Figure 12: Stacking experiment

4.2 Kalai

Figure below shows a compilation of modeling and experiments with the respective metrics for comparison.

Method	Accuracy	precision	recall	f1-score	Confusion Matrix
GaussianNB()	0.87	0.52	1.00	0.68	[[24507 4426] [20 4713]]
KNeighborsClassifier(K=10)	0.88	0.65	0.89	0.75	[[23786 3180]
KNeighborsClassifier(K=5) (pca=0.95)	0.89	0.81	0.76	0.78	[741 5959]]
LogisticRegression	0.86	0.75	0.99	0.65	[[24461 4755] [66 4384]]
DecisionTreeClassifier	0.99	0.99	0.99	0.99	[[24429 135]
DecisionTreeClassifier (pca=0.95)	0.88	0.77	0.78	0.78	[98 9004]]
RandomForestClassifier	1	1	1	1	[[24425 0]
RandomForestClassifier (pca=0.95)	0.91	0.95	0.72	0.81	[102 9139]]
StackingClassifier	0.92	0.88	0.83	0.86	[[23535 992] [1535 7604]]

Figure 13: Comparisons of different ML algorithms

As it's a binary classification problem, ROC curves are used to focus on the performance of the classifier across all thresholds.

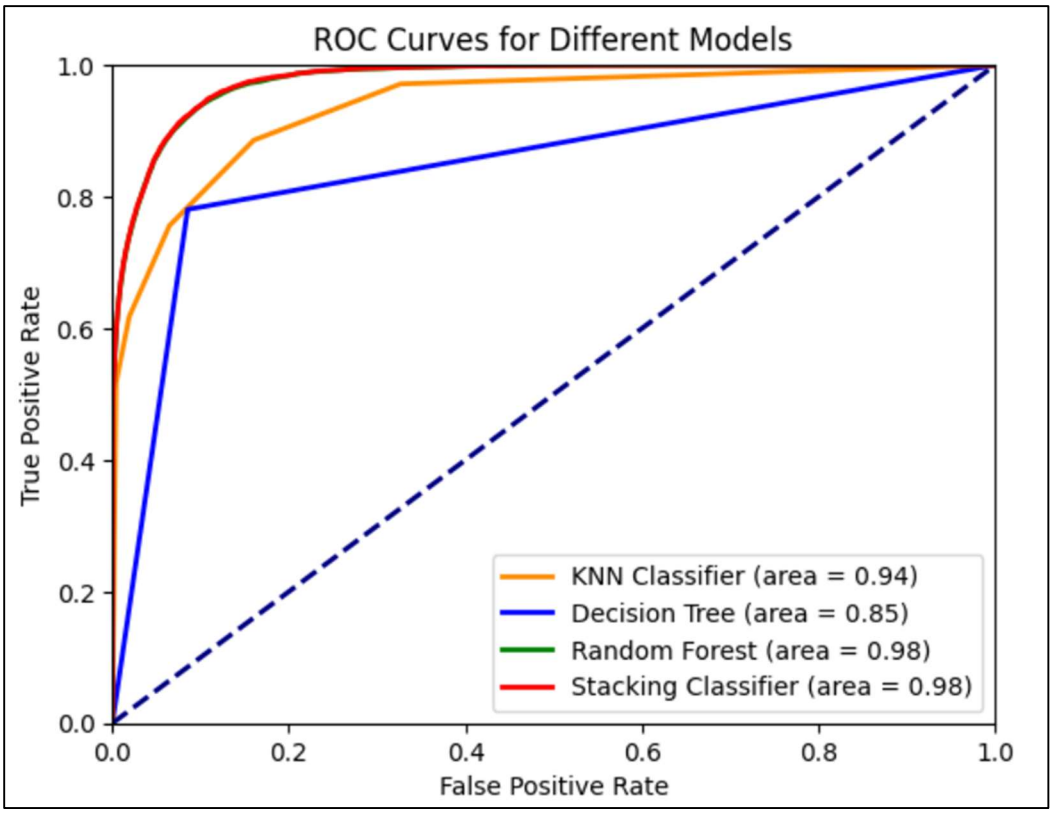


Figure 14: ROC curves for experimented models

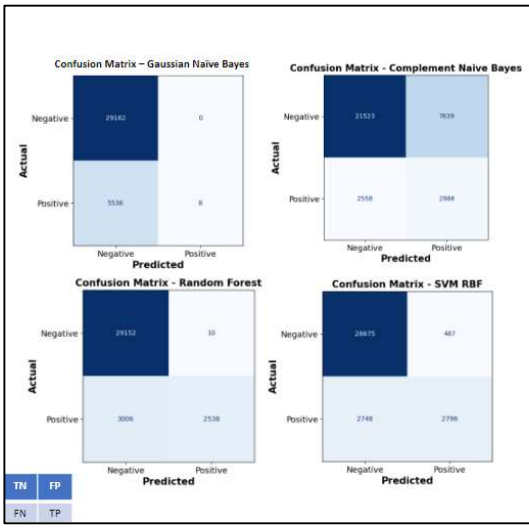
4.3 Sha

An initial base model was built, and further experimentation was explored to try determining the best performing model. Various strategies were implemented to overcome misclassifications faced during modeling. Steps for error analysis and subsequent model improvements (specifically, to increase True Positive and True Negative counts):

- Error analysis with Confusion Matrix
  - Indicates number of False Positive and False Negative for reduction to improve the model.
- Cross-Validation
  - Ensured all models generalize well and is not overfitted to particular subset of data.
- Alternative Models
  - Experiment with other models that might capture different relationships in the data to provide better performance.
- Class Imbalance Handling
  - Consider oversampling minority class or undersampling majority class (can improve recall for minority class).
- Hyperparameter Tuning
  - Consider Grid Search/Random Search to find optimal parameters for model (improve performance for complex models).
- Inspect Misclassified Cases (preliminary only)
  - Review those close to decision boundary. Understand why these cases were incorrectly classified for improvement insights.

1. Initial Base Models Performance

   Figure below shows Complement Naïve Bayes performs better than Gaussian as the algorithm can handle imbalance data well. Generally, all four models have the same Confusion Matrix profile where there's high misclassification for False Negative, and True Positive are not satisfactory high. Here, Random Forest does better than SVM.



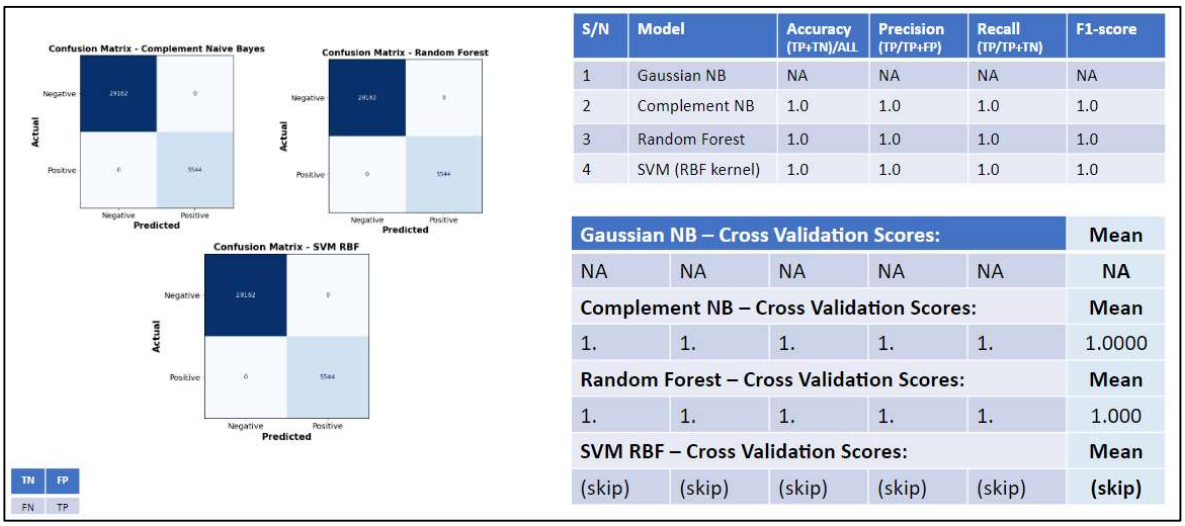
S/N	Model	Accuracy (TP+TN)/ALL	Precision (TP/TP+FP)	Recall (TP/TP+TN)	F1-score
1	Gaussian NB	0.840489	1.0	0.001443	0.002882
2	Complement NB	0.706189	0.281035	0.5386	0.369349
3	Random Forest	0.913099	0.996075	0.457792	0.627286
4	SVM (RBF kernel)	0.906788	0.85166	0.504329	0.633511

Gaussian NB – Cross Validation Scores:					Mean
0.8515	0.8545	0.8527	0.8534	0.8527	**0.8530**
Complement NB – Cross Validation Scores:					**Mean**
0.7070	0.7068	0.7101	0.7067	0.7072	**0.7076**
Random Forest – Cross Validation Scores:					**Mean**
0.9989	0.9978	0.9980	0.9984	0.9984	**0.9983**
SVM RBF – Cross Validation Scores:					**Mean**
(skip)	(skip)	(skip)	(skip)	(skip)	**(skip)**

Figure 15: Base model performance

2. Feature Engineering – Feature Selection

   Tried reducing from initial 32 features to 18 based on business context research. However, it resulted in overfitting characteristics for all models.



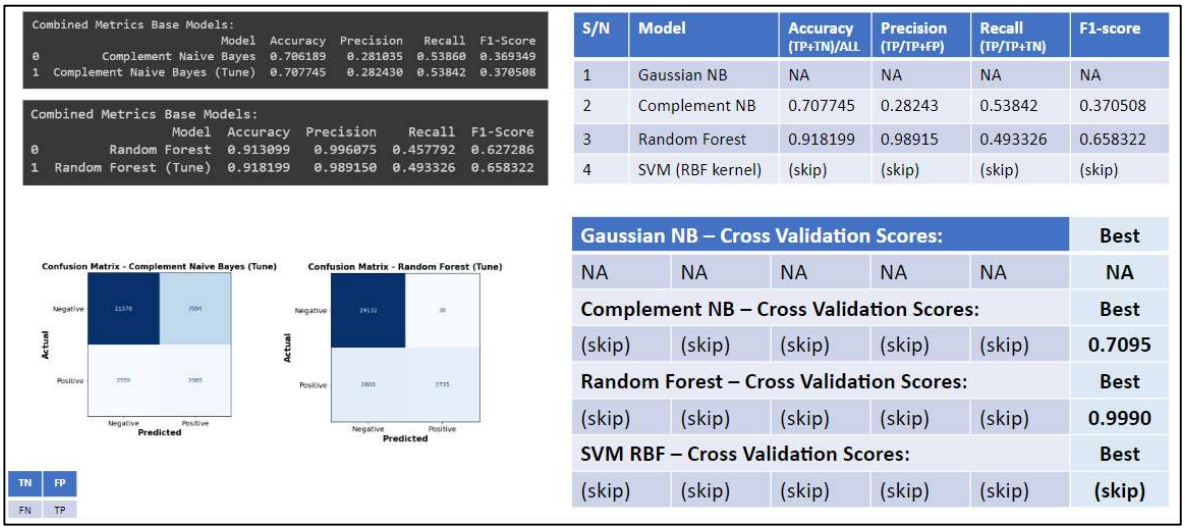
S/N	Model	Accuracy (TP+TN)/ALL	Precision (TP/TP+FP)	Recall (TP/TP+TN)	F1-score
1	Gaussian NB	NA	NA	NA	NA
2	Complement NB	1.0	1.0	1.0	1.0
3	Random Forest	1.0	1.0	1.0	1.0
4	SVM (RBF kernel)	1.0	1.0	1.0	1.0

Gaussian NB – Cross Validation Scores:					Mean
NA	NA	NA	NA	NA	NA
Complement NB – Cross Validation Scores:					**Mean**
1.	1.	1.	1.	1.	1.0000
Random Forest – Cross Validation Scores:					**Mean**
1.	1.	1.	1.	1.	1.000
SVM RBF – Cross Validation Scores:					**Mean**
(skip)	(skip)	(skip)	(skip)	(skip)	**(skip)**

Figure 16: Feature reduction experimentation results

3. Hyperparameter tuning

   Due to some models being computationally extensive, only Complement Naïve Bayes and Random Forest models were tuned. Found that there's not many improvements in performance as values change by the third decimal place. Still Random Forest is better than Complement Naïve Bayes.



Combined Metrics Base Models:

	Model	Accuracy	Precision	Recall	F1-Score
0	Complement Naive Bayes	0.706189	0.281035	0.53860	0.369349
1	Complement Naive Bayes (Tune)	0.707745	0.282430	0.53842	0.370508

Combined Metrics Base Models:

	Model	Accuracy	Precision	Recall	F1-Score
0	Random Forest	0.913099	0.996075	0.457792	0.627286
1	Random Forest (Tune)	0.918199	0.989150	0.493326	0.658322

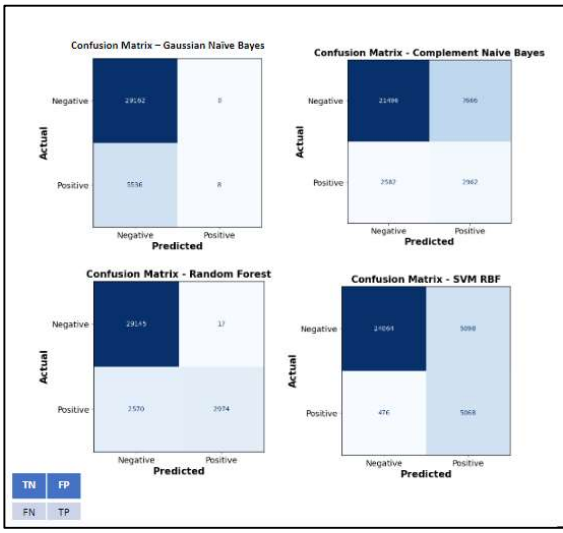
S/N	Model	Accuracy (TP+TN)/ALL	Precision (TP/TP+FP)	Recall (TP/TP+TN)	F1-score
1	Gaussian NB	NA	NA	NA	NA
2	Complement NB	0.707745	0.28243	0.53842	0.370508
3	Random Forest	0.918199	0.98915	0.493326	0.658322
4	SVM (RBF kernel)	(skip)	(skip)	(skip)	(skip)

Gaussian NB – Cross Validation Scores:					Best
NA	NA	NA	NA	NA	NA
Complement NB – Cross Validation Scores:					**Best**
(skip)	(skip)	(skip)	(skip)	(skip)	**0.7095**
Random Forest – Cross Validation Scores:					**Best**
(skip)	(skip)	(skip)	(skip)	(skip)	**0.9990**
SVM RBF – Cross Validation Scores:					**Best**
(skip)	(skip)	(skip)	(skip)	(skip)	**(skip)**

Figure 17: Tuning experimentation results

4. Oversampling – SMOTE

   SMOTE was only applied to train dataset while test dataset remains unchanged. Through experimentation, it was able to prove the theoretical hypothesis that SMOTE generally helps to improve recall for minority class. Amongst all techniques implemented, Random Forest with SMOTE is the best model produced.



S/N	Model	Accuracy (TP+TN)/ALL	Precision (TP/TP+FP)	Recall (TP/TP+TN)	F1-score
1	Gaussian NB	0.840489	1.000000	0.001443	0.002882
2	Complement NB	0.704720	0.278698	0.534271	0.366312
3	Random Forest	0.925460	0.994316	0.536436	0.696895
4	SVM (RBF kernel)	0.839394	0.498524	0.914141	0.645194
5	Voting (Hard)	0.924393	0.994580	0.529582	0.691149

Gaussian NB – Cross Validation Scores:					Mean
0.5986	0.5955	0.5952	0.5956	0.5857	0.5942
Complement NB – Cross Validation Scores:					**Mean**
0.6515	0.65730	0.6511	0.6567	0.6497	0.6533
Random Forest – Cross Validation Scores:					**Mean**
0.9990	0.9983	0.9985	0.9987	0.9158	0.9821
SVM RBF – Cross Validation Scores:					**Mean**
(skip)	(skip)	(skip)	(skip)	(skip)	(skip)

Figure 18: SMOTE experimentation results

5. Error Analysis (Test vs Predicted)

   To improve Random Forest model, can use interest_rate_spread feature to create new feature with given range (green line highlights) to better separate default and non-default. To improve Naïve Bayes and SVM models, might have to explore new features with higher e.g. higher polynomial degree.

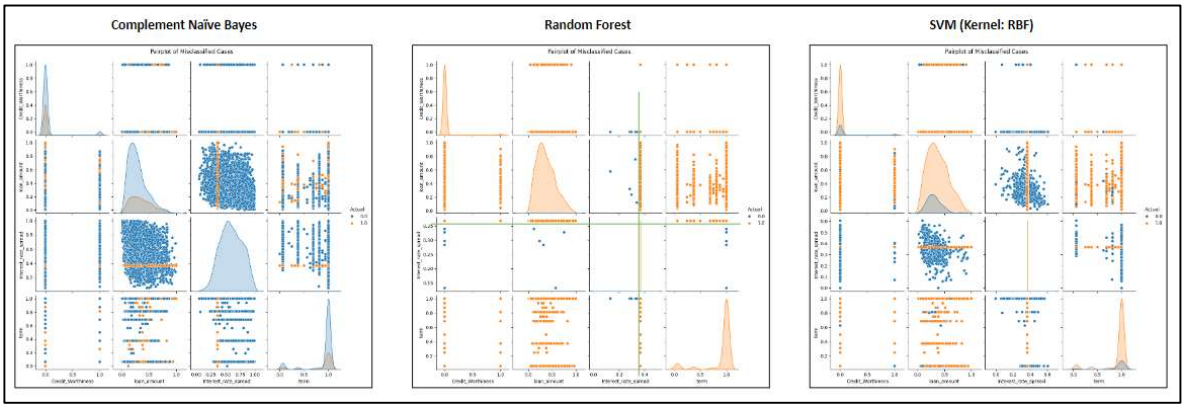


Figure 19: Pairplot of misclassified cases

From the figure below, Naïve Bayes has actual non-default (0) highly predicted wrongly by 80%. Random Forest has actual default (1) highly predicted wrongly by majority. SVM has actual default (1) highly predicted wrongly by 83%.

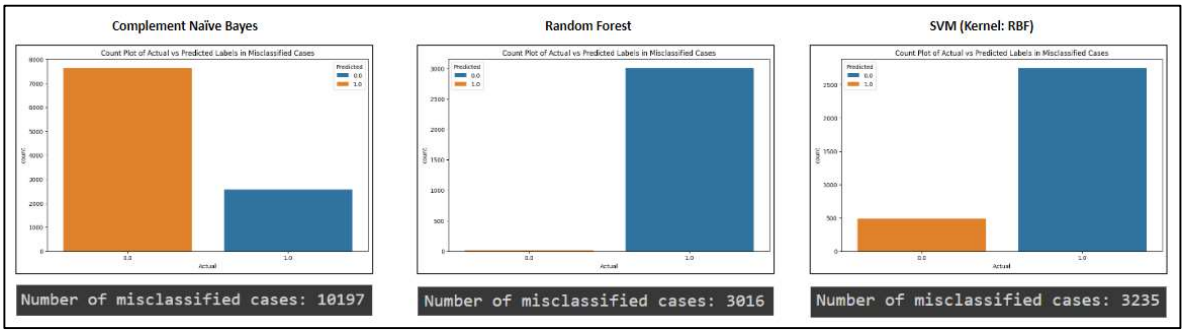


Figure 20: Count of Actual vs Predicted labels misclassified cases

6. Deployment/Application

This section describes the deployment of the team's model either as part of an application, or a standalone web service, or a demo app (e.g. using Streamlit). It includes the application architecture, technology stack used, and model experiments implemented.

6.1 Application Architecture

From the figure below, it shows the end-to-end process of model building to deployment.

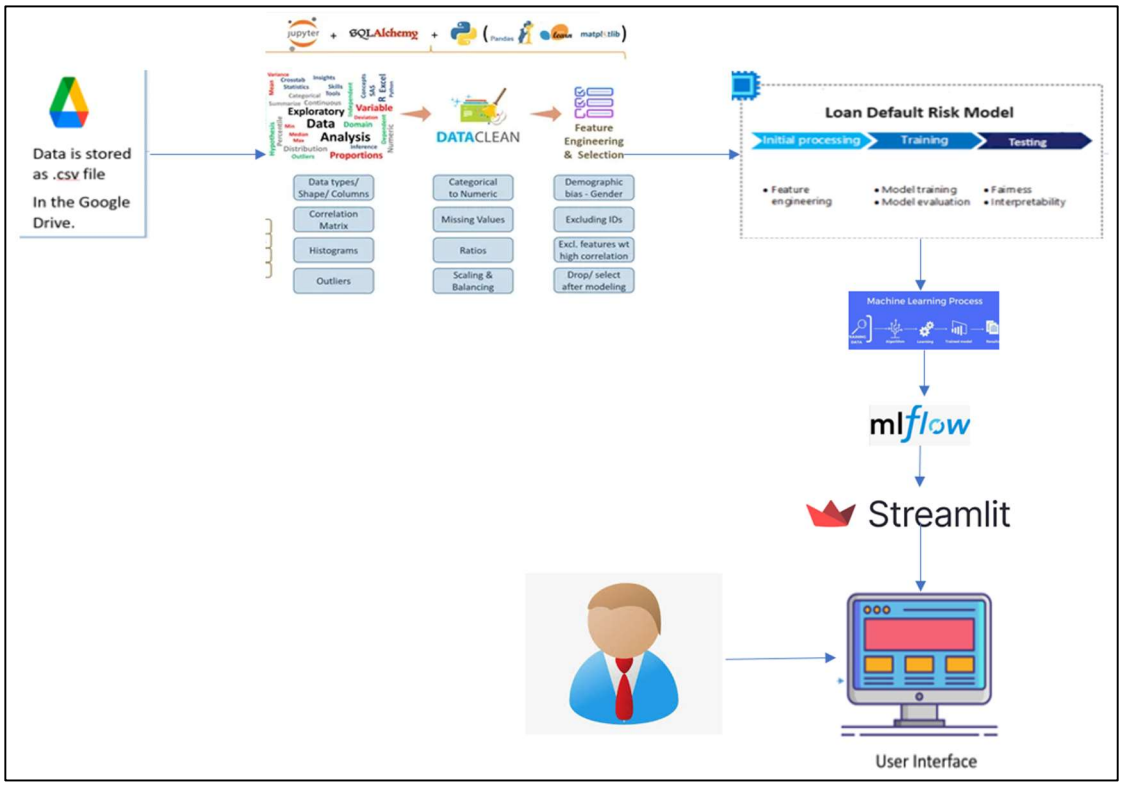


Figure 21: Backend architecture for deployment

6.2 Technology Stacked Used

Below shows the list of technologies utilized from modeling to deployment:

1. Python
2. Scikit Learn
3. GitHub
4. mlFlow
5. Streamlit

6.3 Experiment Log

Each member pushed their models and experimentations into mlFlow for experimental logging and comparison before deciding on the best model for deployment.

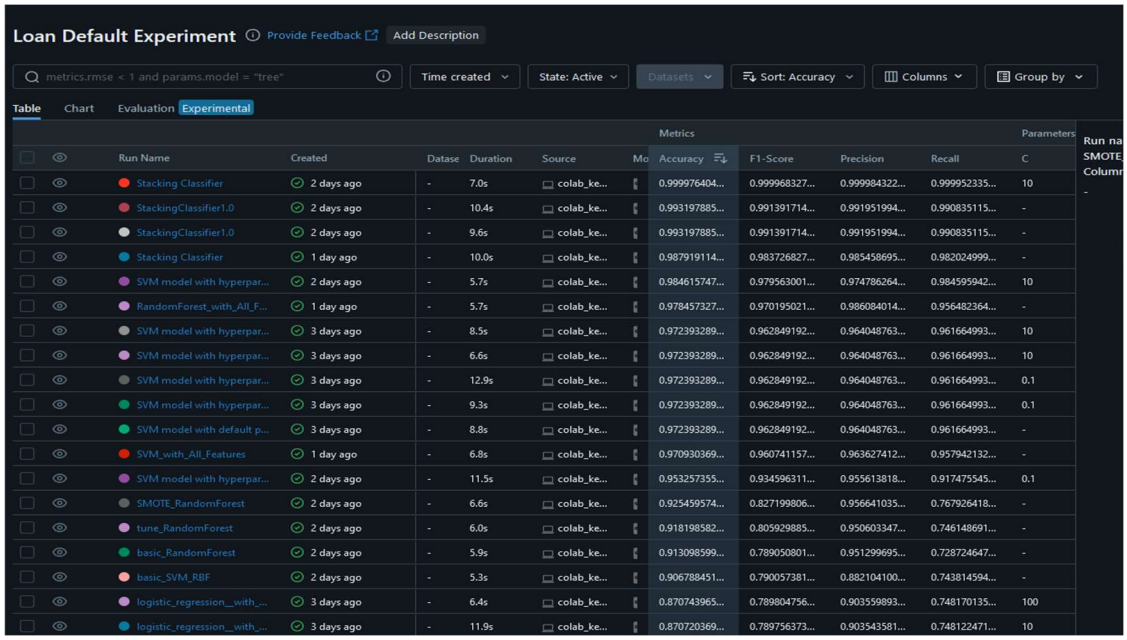


Figure 22: Sample of experiments implemented

6.4 Deployed Application

Upon finalizing the chosen model, Streamlit was used to create and deploy the model as a basic application for demonstration purposes. From the figure below, the user would key in the required fields and a result would show the prediction if a particular profile will be defaulted or not.

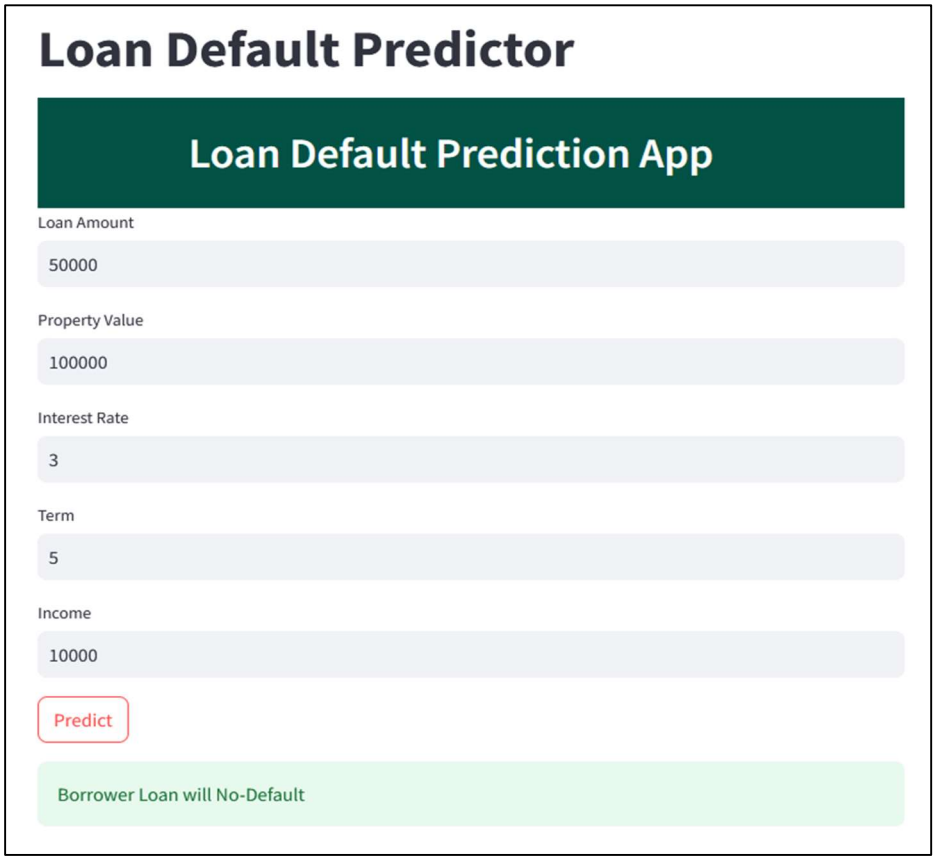


Figure 23: Demo application for Loan Default

7. Conclusion

This section describes the discussion of how the team deduced to the final model through analysis and reasoning for it being chosen. The figure below shows only the best models from each member for general comparison.

Best Models (amongst members)	Accuracy	Precision	Recall	F1-score	Confusion Matrix
Stacking Classifier (Random Forest/Gaussian NB + Gradient Boosting)	0.9879	0.98	0.97	0.98	[31690 201] 311 10179]
KNeighbors Classifier (k=10)	0.8835	0.81	0.89	0.84	[23786 3180] 741 5959]
Random Forest (SMOTE)	0.9254	0.99	0.54	0.70	[29145 17] 2570 2974]

Figure 24: Comparison of best 3 models

7.1 Analysis

1. Stacking (Random Forest & Gaussian Naïve Bayes):
    - High Accuracy and F1-Score: This model performs exceptionally well, with very high accuracy and F1-score, suggesting a balanced performance across classes.
    - Precision and Recall: Both are high, indicating that the model is both good at finding positive instances and minimizing false positives.
    - Best Overall Performance: This combination of Random Forest/Gaussian NB with Gradient Boosting as the meta-model is the best performer overall.
2. K-Nearest Neighbors:
    - Decent Recall: This model has a higher recall, indicating it's good at identifying positive instances but has lower precision, leading to more false positives.
    - Lower Accuracy and F1-Score: KNN has lower overall accuracy and F1-score, making it less reliable for this specific task, compared to stacking classifier.
3. Random Forest:
    - High Precision, Low Recall: The Random Forest model is highly precise but suffers from low recall, meaning it misses many positive instances.
    - Imbalanced Performance: This imbalance suggests that the model is too conservative, potentially due to overfitting or an inappropriate handling of class imbalance, even with SMOTE.

7.2 Chosen Model

The Stacking Classifier is the best choice among the models evaluated. It provides the best balance between accuracy, precision, recall, and F1-score, and its confusion matrix shows a strong performance in both correctly identifying true positives and true negatives. The combination of Random Forest/Gaussian NB as base models and Gradient Boosting as the meta-model seems to work well in this context.

8. Annexes

Annex A – Initial raw dataset details

```
Rows     :  148670
Columns  :  34
Missing values :  181135
```

Status	Count	Percentage
0	112031	75.355485
1	36639	24.644515

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148670 entries, 0 to 148669
Data columns (total 34 columns):
```

#	Column	Non-Null Count	Dtype
0	ID	148670 non-null	int64
1	year	148670 non-null	int64
2	loan_limit	145326 non-null	object
3	Gender	148670 non-null	object
4	approv_in_adv	147762 non-null	object
5	loan_type	148670 non-null	object
6	loan_purpose	148536 non-null	object
7	Credit_Worthiness	148670 non-null	object
8	open_credit	148670 non-null	object
9	business_or_commercial	148670 non-null	object
10	loan_amount	148670 non-null	int64
11	rate_of_interest	112231 non-null	float64
12	Interest_rate_spread	112031 non-null	float64
13	Upfront_charges	109028 non-null	float64
14	term	148629 non-null	float64
15	Neg_ammortization	148549 non-null	object
16	interest_only	148670 non-null	object
17	lump_sum_payment	148670 non-null	object
18	property_value	133572 non-null	float64
19	construction_type	148670 non-null	object
20	occupancy_type	148670 non-null	object
21	Secured_by	148670 non-null	object
22	total_units	148670 non-null	object
23	income	139520 non-null	float64
24	credit_type	148670 non-null	object
25	Credit_Score	148670 non-null	int64
26	co-applicant_credit_type	148670 non-null	object
27	age	148470 non-null	object
28	submission_of_application	148470 non-null	object
29	LTV	133572 non-null	float64
30	Region	148670 non-null	object
31	Security_Type	148670 non-null	object
32	Status	148670 non-null	int64
33	dtir1	124549 non-null	float64

```
dtypes: float64(8), int64(5), object(21)
memory usage: 38.6+ MB
```

Annex B – Arun's Preprocessing

In this data processing steps the null values are filled and the columns that don't contribute to the target value prediction is removed.

Data Cleaning:

(1) The security type column has typo "**Indriect**" have to change it to "**Indirect**".
(2) Some numerical columns like "property_value", income have missing values which are imputed with the median values. Since mean value imputation will not fit these columns as the min and max value ranges are more.
(3) Categorical columns which have missing columns are imputed with the constant value 'Missing' to preserve the information in the dataset.

Data Reduction:

(1) Removed ID column as it is a primary key value.
(2) Removed year column as it has only one value.

Annex C – Arun's EDA

For data preparation, full details on the work accomplished can be found on a separate Python programming file provided in the link here: Arun's Python Programming Hyperlink .

Insights gain:

(1) Loan limit type CF has more records.

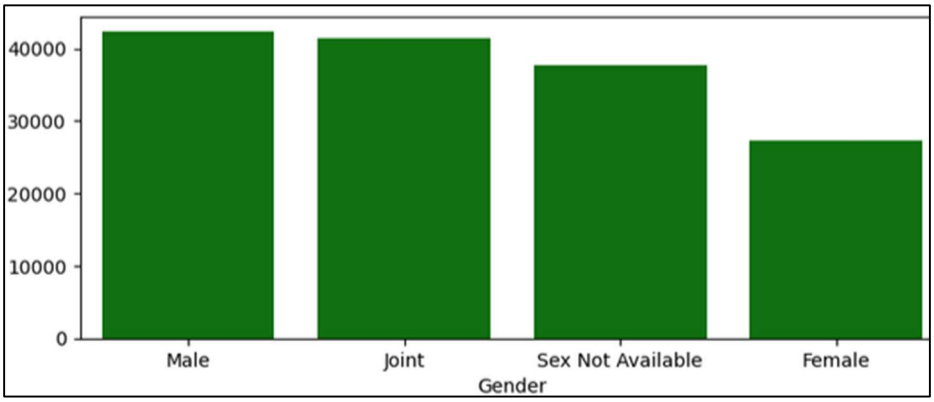


Figure 25: Count by gender

(2) Male and Joint borrowers are more. Also gender column has Sex Not Available value. Need to remove this column has no impact on the predictor column as any gender will default on the loan and not one specific gender.

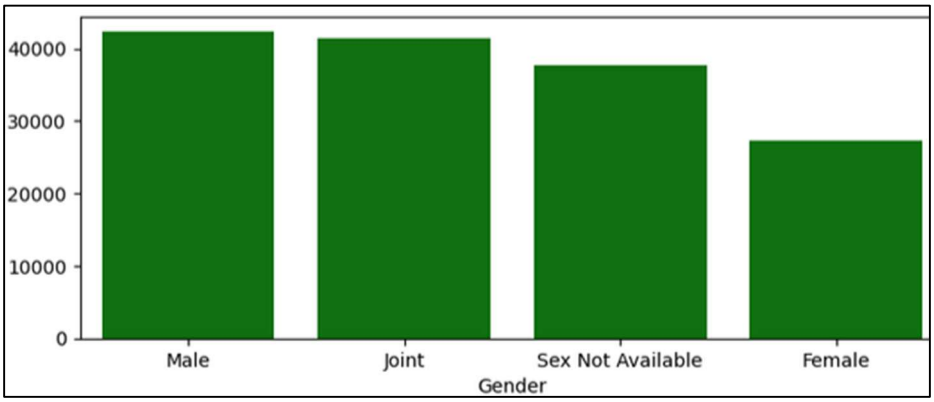


Figure 26: Count by gender

(3) More borrowers are in the North and South region, maybe North and South has more property developments.

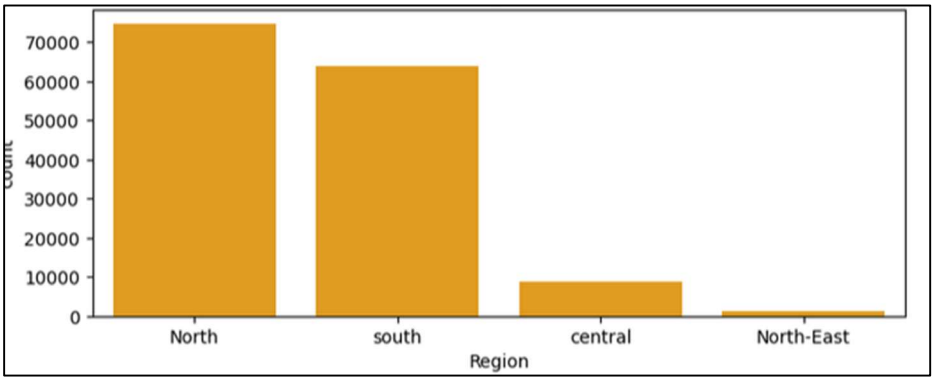


Figure 27: Count by region

(4) All age groups have borrowers but less borrowers in the age of 20's and 70's.Most of the borrowers are of working age.

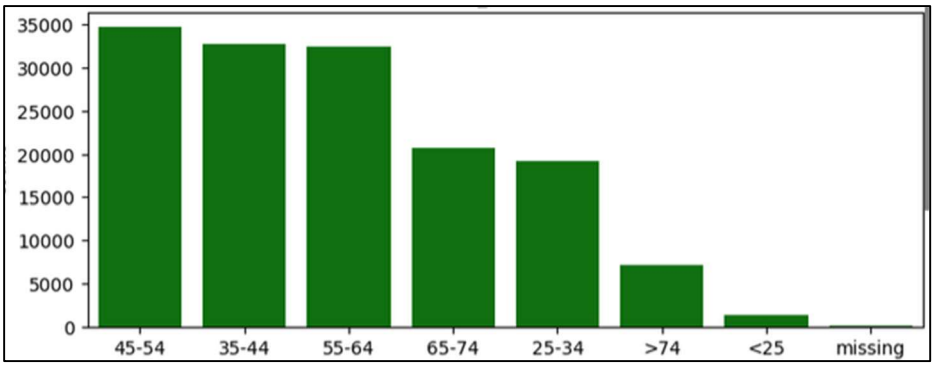


Figure 28: Count by age range in descending order

(5) Most of the loan borrowed is for the 1U property.

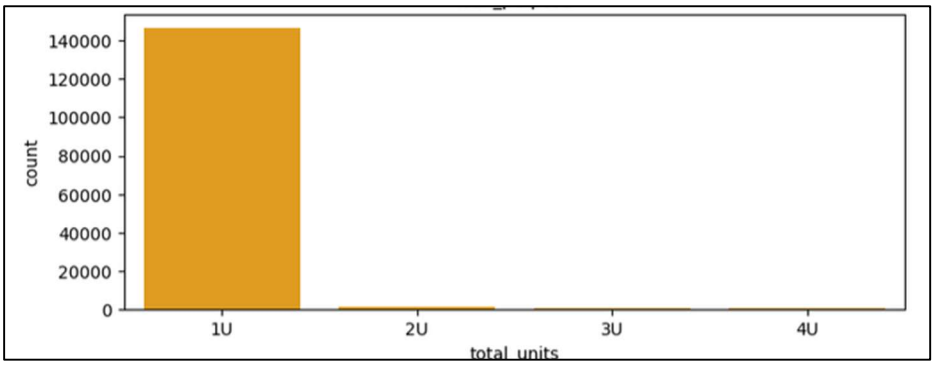


Figure 29: Count by total units

(6) Property value and loan amount columns are positively correlated.

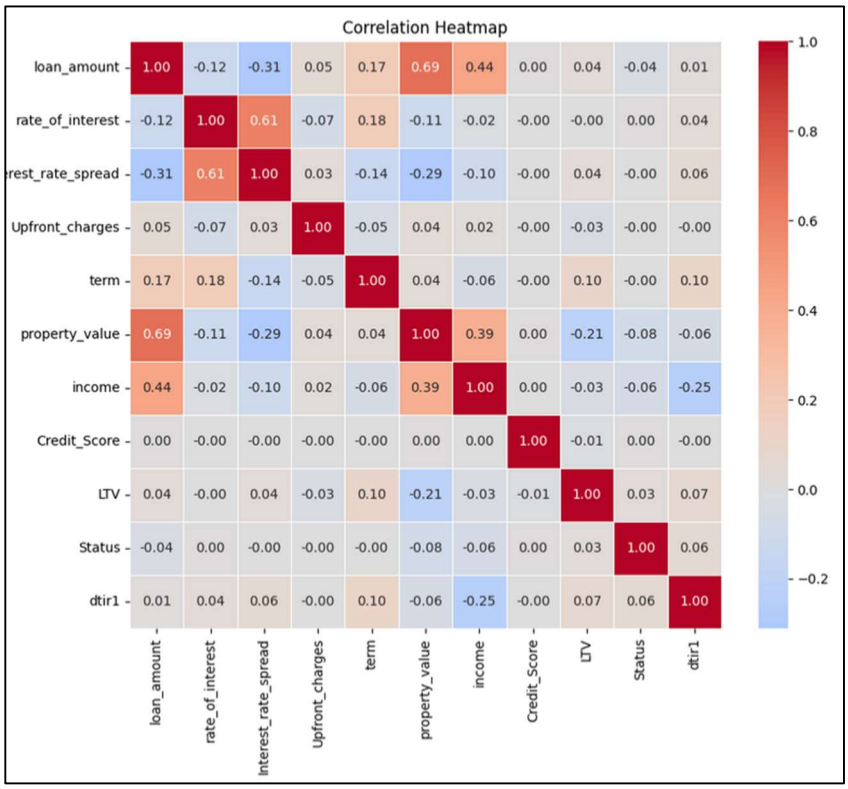


Figure 30: Correlation matrix in heatmap

Annex D – Arun's Feature Engineering

Several techniques were explored:

(1) Ordinal encoding

The age and loan type are the ordinal categories. The age feature has categories like age range instead of the numbers, similarly the loan type column.

(2) Handling outliers

Outliers' removal from the important features like property value and loan amount. Since these are left skewed it will have a negative impact on the model's performance.

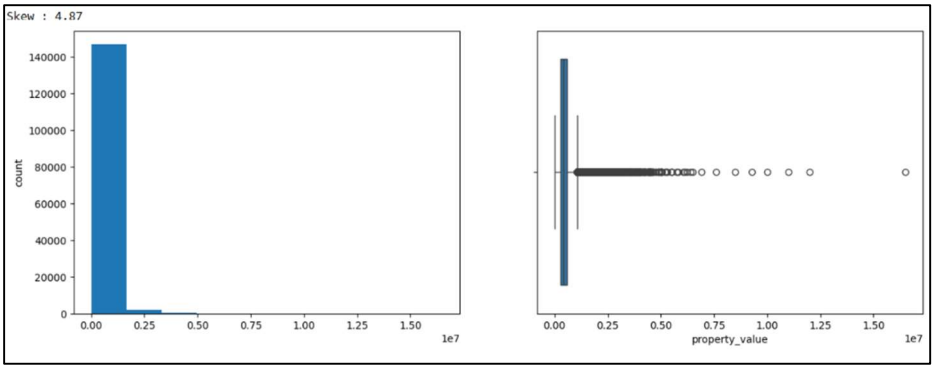


Figure 31: Outliers in property value.

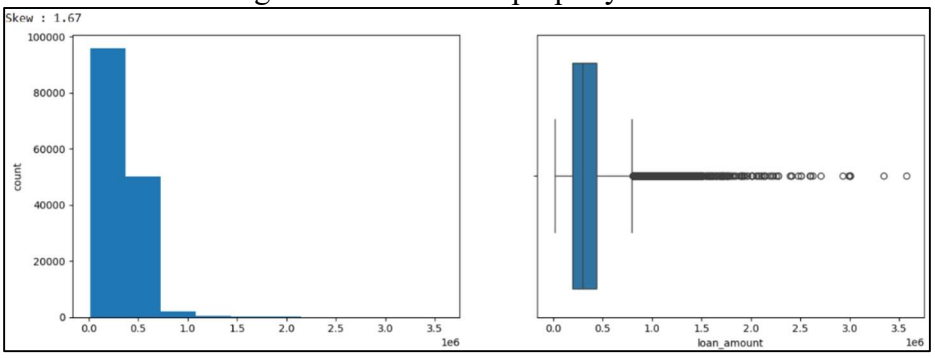


Figure 32: Outliers in loan amount.

After outlier removal with IQR, the data looks less skewed.

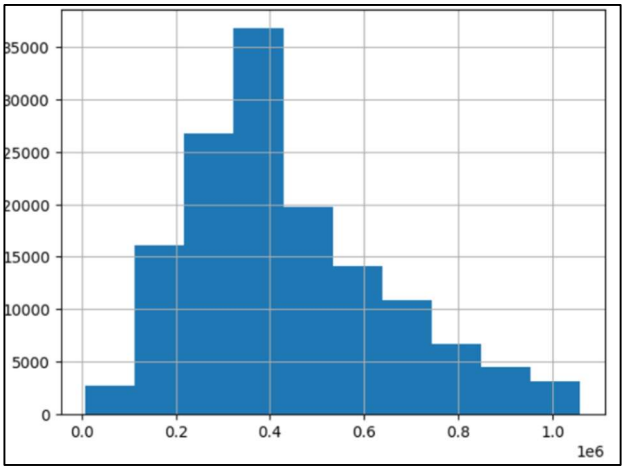


Figure 33: After outlier removal in property value.

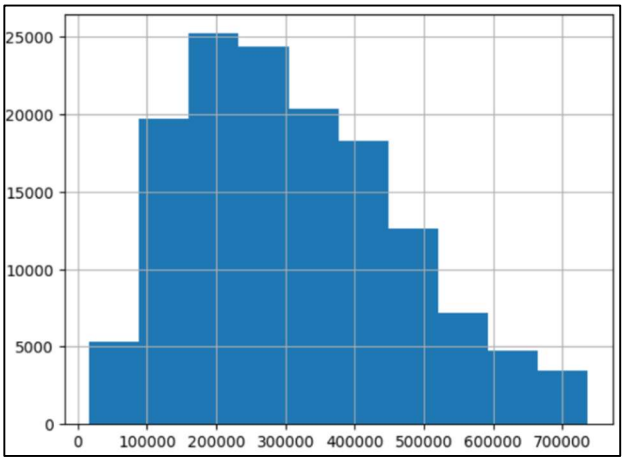


Figure 34: After outlier removal in loan amount.

(3) Scaling

Within the dataset, the numerical ranges between columns may differ quite largely. To standardize them, StandardScaler was applied so that the values will interpolate accordingly.

Annex E – Kalai's Preprocessing

Our dataset is extracted from Kaggle and analyzed. It contains both numerical and categorical data type with null values and outliers. Data analysis and preprocessing is the fast track to improving data quality using data cleaning, wrangling and data transformation.

Data Cleaning:

(1) Missing values for the numerical features are analyzed and can be filled with median value as there are 0 minimum values and outliers.
(2) Missing values for categorical features are analyzed and can be filled with median or more-frequent values.

Data Transformation:

(1) Missing value handling is done.
(2) Removed column ID and year.
(3) Outliers are removed based on the IQR method

Annex F – Kalai's EDA

For data preparation, full details on the work accomplished can be found on a separate Python programming file provided in the link here: Kalai's Python Programming Hyperlink .

Correlation matrix for numerical features:

(1) ID and year are not required for the prediction.

(2) loan_amount is correlated with income feature in .44 and the loan_amount is highly correlated with property_value as .69

(3) rate_of_interest is highly correlated with interest_rate_spread in .62

Figure 35: Correlation matrix for numerical features

Univariate Analysis:

(1) Credit_Score does not have any outlier and is equally distributed.

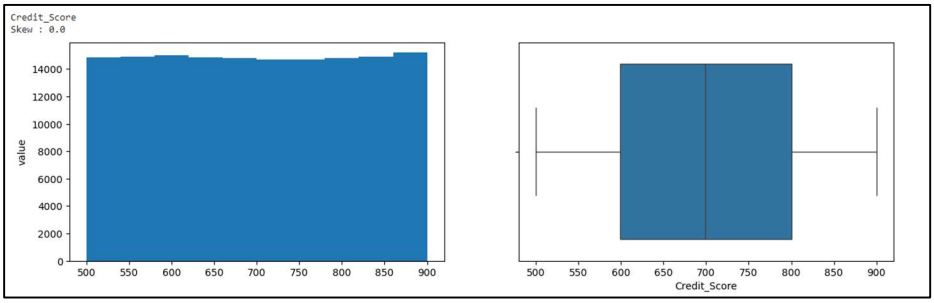


Figure 36: Credit score feature analysis

(2) Income has zero values and outliers.

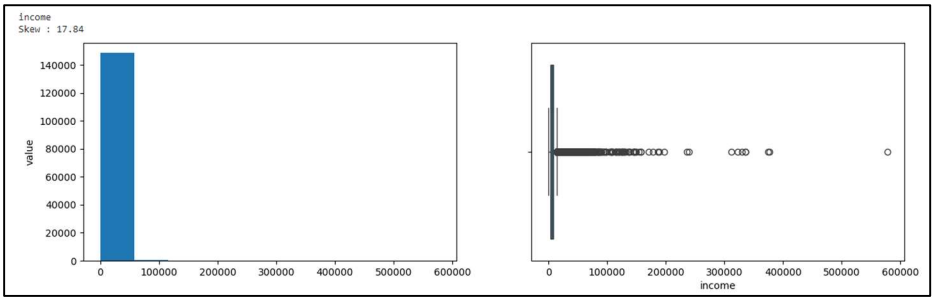


Figure 37: Income feature analysis

(3) Loan_Limit has outliers.

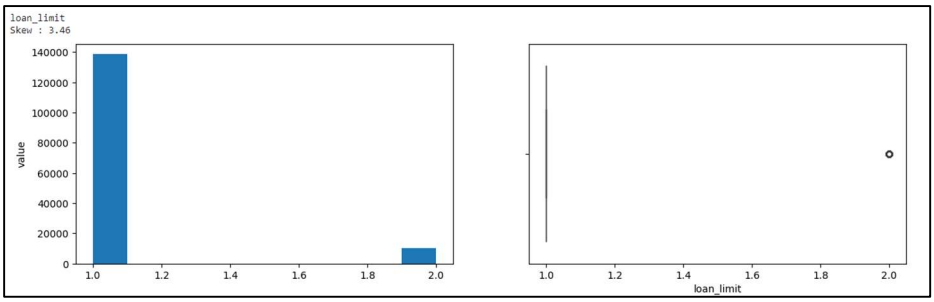


Figure 38: Loan limit feature analysis

(4) Loan_Amount has right skewed and outliers.

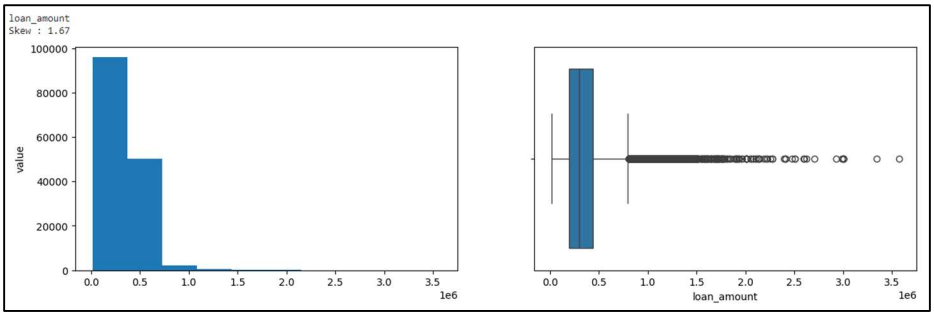


Figure 39: Loan amount feature analysis

(5) 140K units are with IU property.

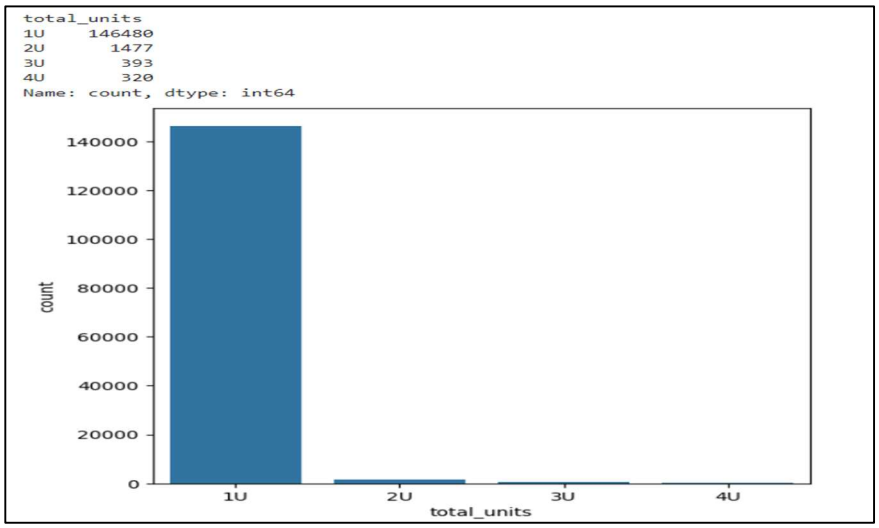


Figure 40: Total units feature analysis

(6) Secured_by landed property very low based on the prediction performance we can remove these data.

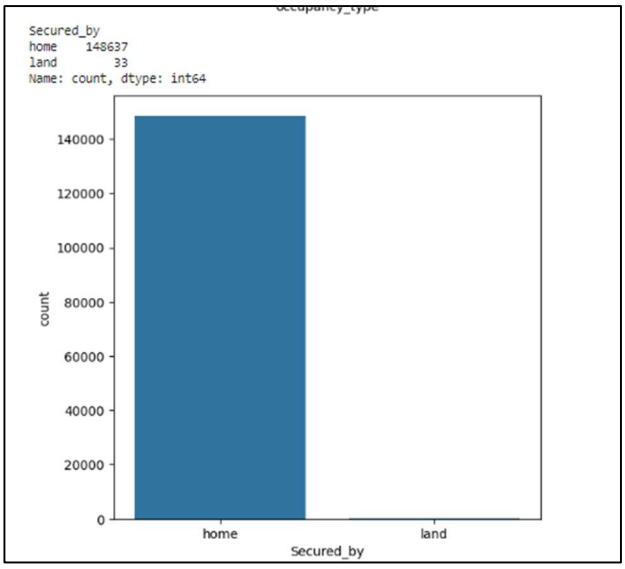


Figure 41: Secured by feature analysis

Bivariate Analysis:

(1) The variable term has a positive correlation with dtir1 and loan_amount.

(2) The variable Credit_Score has a positive correlation with dtir1 and loan_amount.

(3) The variable LTV in has a positive correlation with the loan_amount as LTV increases loan_amount increases.

(4) Dtir1 and LTV are negatively correlated with Income.

(5) Credit_Score is negatively correlated with LTV as Credit_Score increases LTV decreases.

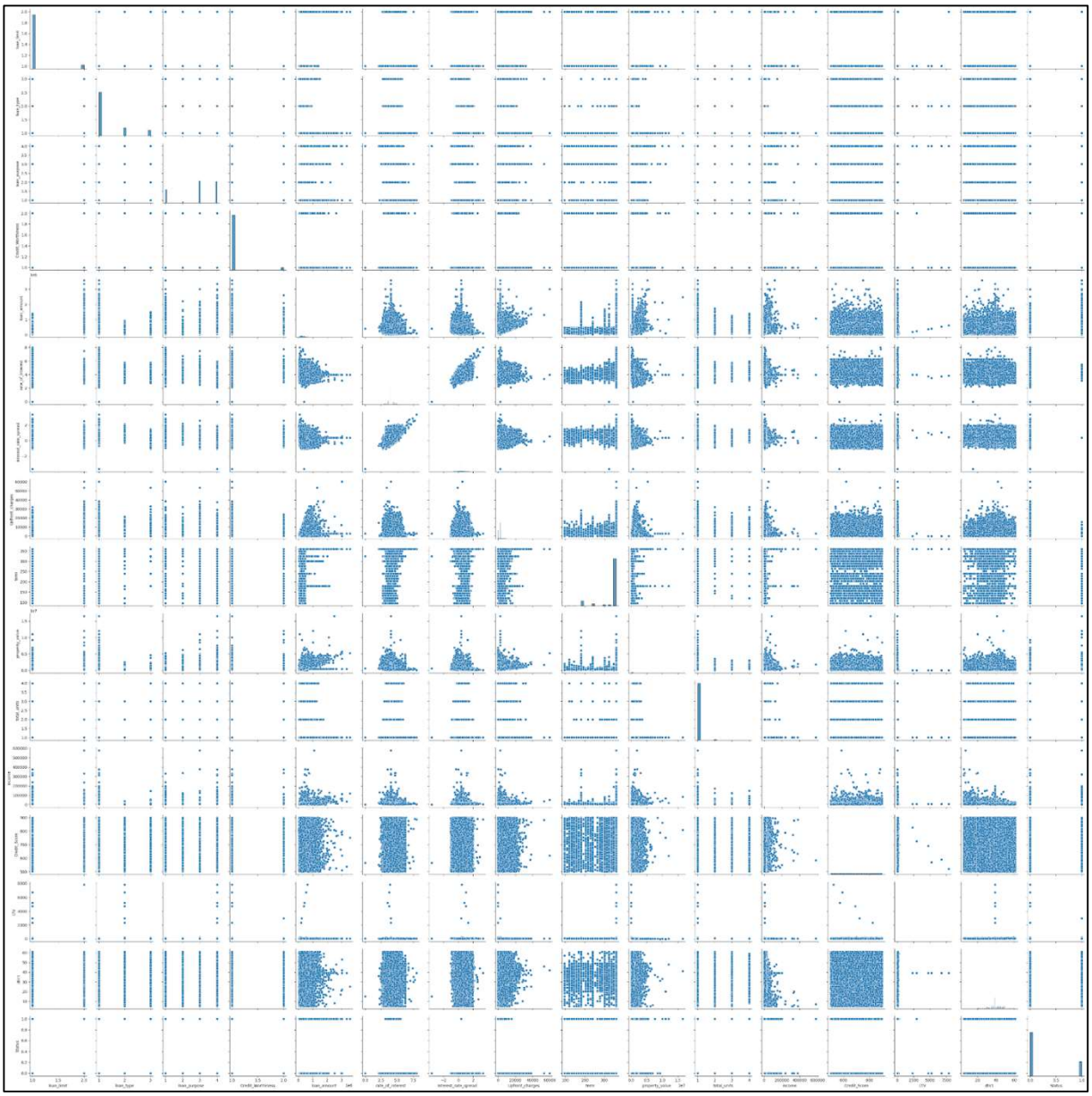


Figure 42: Pair plot analysis for numerical features

Annex G – Kalai's Feature Engineering

Several techniques were explored:

(1) Encoding

Using LabeEncoder, age feature is encoded from the age range into the numerical value. Using OneHotEncoder all other categorical features are encoded into numerical values which create new columns based on the value.

	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness
0	1	Sex Not Available	nopre	1	1	1
1	1	Male	nopre	2	1	1
2	1	Male	pre	1	1	1
3	1	Male	nopre	1	4	1
4	1	Joint	pre	1	1	1

5 rows × 32 columns

Figure 43: Before encoding

Gender_Joint	Gender_Male	Gender_Sex Not Available	approv_in_adv_pre	open_credit_opc	business_or_commercial_nob/c	Neg_ammortization_not_neg
0.0	0.0	1.0	0.0	0.0	1.0	1.0
0.0	1.0	0.0	0.0	0.0	0.0	1.0
0.0	1.0	0.0	1.0	0.0	1.0	0.0
0.0	1.0	0.0	0.0	0.0	1.0	1.0
1.0	0.0	0.0	1.0	0.0	1.0	1.0

Figure 44: After encoding

(2) Outlier Handling

Outliers are removed based on the IQR method.

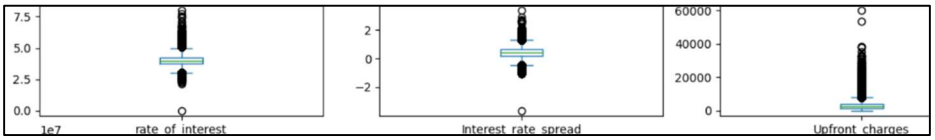


Figure 45: Before outlier handling

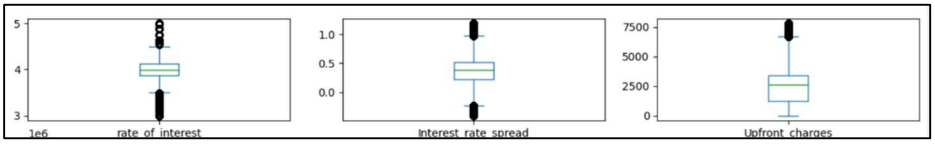


Figure 46: After outlier handling

(3) Feature Scaling

Standardization (Z-Score Normalization) technique is used for feature scaling.

Annex H – Sha's Preprocessing

Before any preprocessing steps are taken in attempts to prepare the dataset for exploratory data analysis, the dataset was initially split into train and test dataset. This is accomplished with a ratio of 70/30 and a technique called stratified sampling based on the target label.

Due to time constraints, Data Pipeline method was not explored in this solution thus resulting in two Python programming files that are split into different versions, Train and Test. The preprocessing steps however are similar, just that each file exports out the respective dataset as outputs.

Data Cleaning:

(1) Replace null values for justifiable columns (column: rate of interest, interest rate spread, upfront charges, LTV, dtir1).
(2) Drop columns for non-useful features (column: ID, year).
(3) Detect and remove outliers more than value 100 (column: LTV).
(4) General detection and removal of outliers for whole dataset using Z-score method.

Data Transformation:

(1) Categorical missing value imputation using most frequent observations (column: loan limit, approv in adv, loan purpose, neg ammortization, age, submission of application).
(2) Numerical missing value imputation using k-NN imputer (column: term, property value, income).

Post-processing to prepare for ML:

(1) Drop columns with high correlation (using V Cramer's) for Categorical Features (column: co-applicant credit type, submission of application, security type).
(2) Drop columns with high correlation for Numerical Features (column: rate of interest, upfront charges).
(3) Combine low frequency categorical values for affected categorical columns (column: loan type, loan purpose, occupancy type, total units, age, region).
(4) Use LabelEncoder for columns assumed to be of Ordinal Data (column: credit worthiness, total units, age).
(5) Use One-hot Encoder for columns assumed to be of Nominal Data (column: the rest of object datatype columns)
(6) Apply MinMaxScaler to standardize range from 0 to 1 (column: the rest of columns except those that uses LabelEncoder method).

Annex I – Sha's EDA

For data preparation, full details on the work accomplished can be found on a separate Python programming file provided in the link here:

- Sha's Python Programming Hyperlink - Train
- Sha's Python Programming Hyperlink - Test

Univariate Analysis:

(1) Region: Most clients reside in the North & South.

(2) Age: High percentage of clients taking loans between 35-64 years old.

(3) Credit Type: Most clients are assessed based on Credit Information Bureau, CRIF Credit Bureau or Experian standards.

(4) Loan Type: Most clients took up type1 loans.

(5) Loan Purpose: Less clients have p2 purpose while rest of the purpose are generally well distributed.

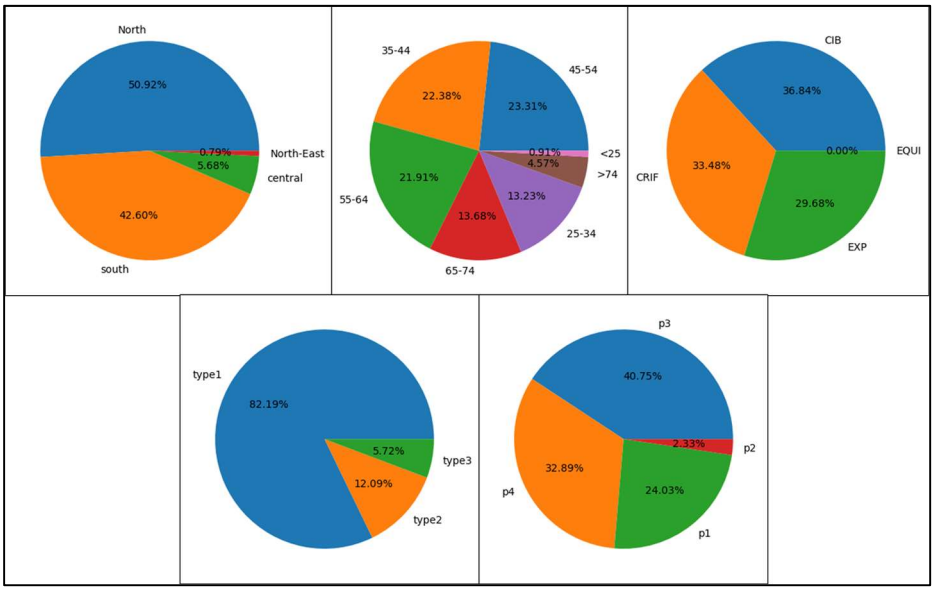


Figure 47: Distribution for some categorical features

Bivariate Analysis:

(1) Most transactions have not been defaulted on. But those who are male or undisclosed tend to default more.

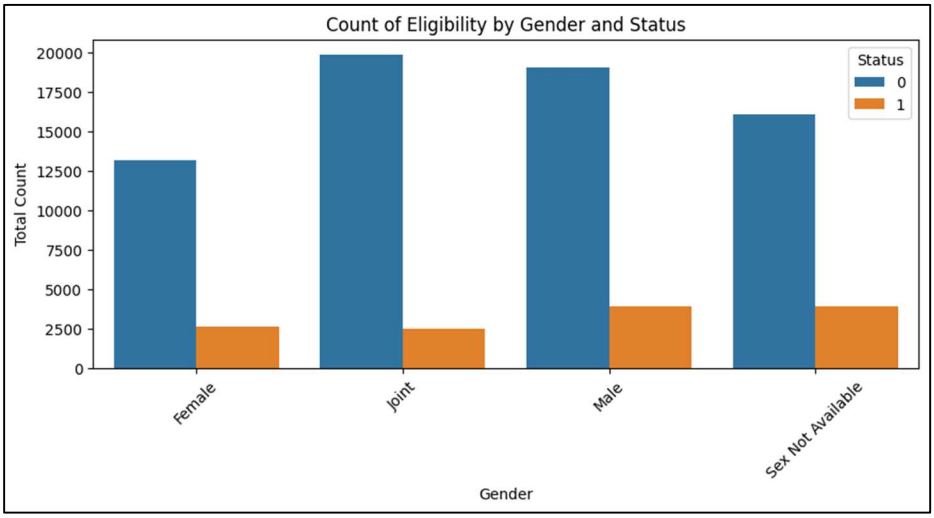


Figure 48: Eligibility by gender & status

(2) Peak income group are usually around the age range of 35 to 54.

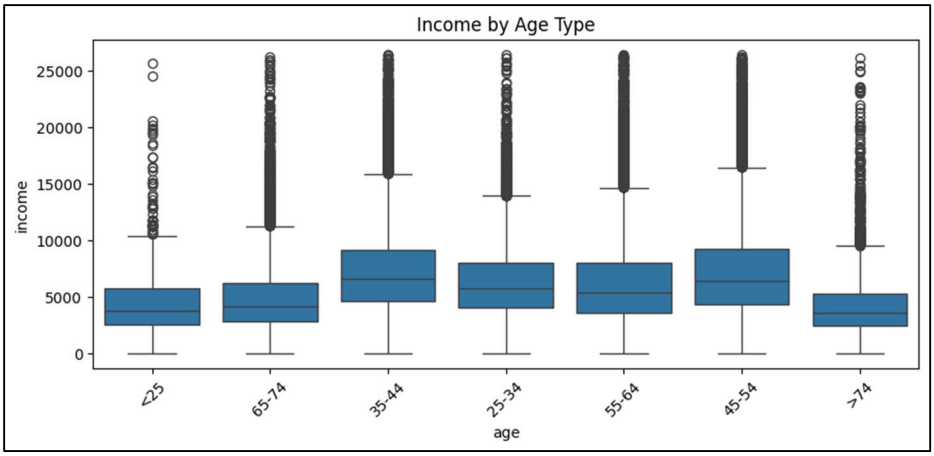


Figure 49: Income by age range

(3) Those with Equifax credit type have lower scoring than other credit score types. While credit types from credit information bureau, CRIF credit information bureau, Experian generally has similar range of credit score.

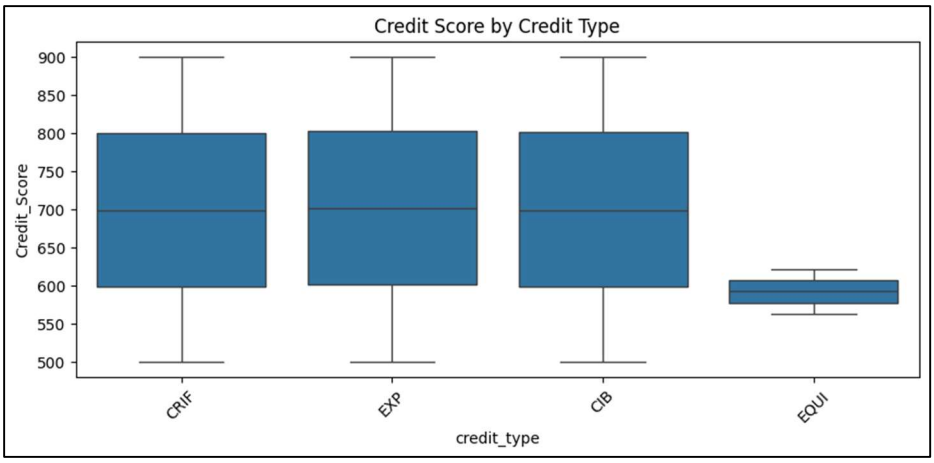


Figure 50: Credit score by credit type

(4) The bigger the property (more units), the higher the property value.

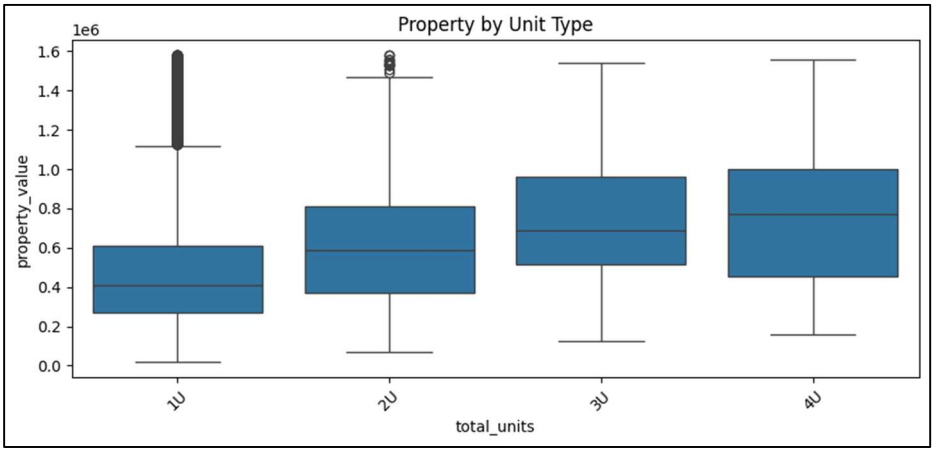


Figure 51: Property value by unit type

(5) The rate of interest gap doesn't differ much between interest types. And the interest rate spread is generally the same.

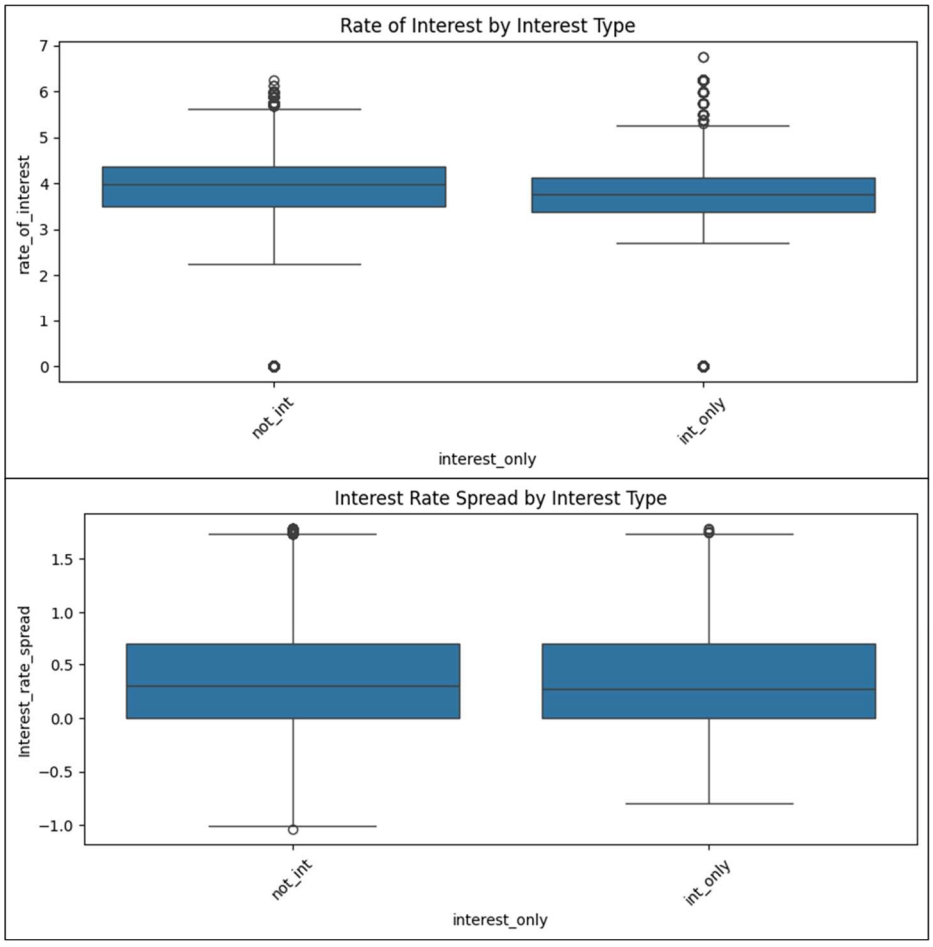


Figure 52: Comparison of rate of interest & spread by type

(6) Features with high correlation: co-applicant credit type, submission of application, security type, rate of interest, upfront charges.

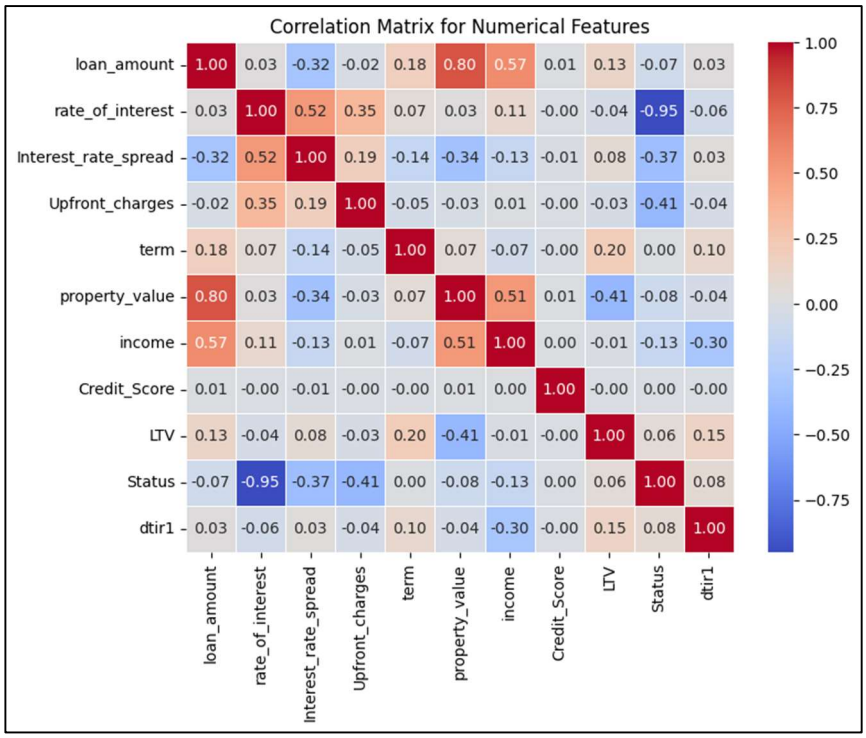


Figure 53: Numerical features correlation matrix

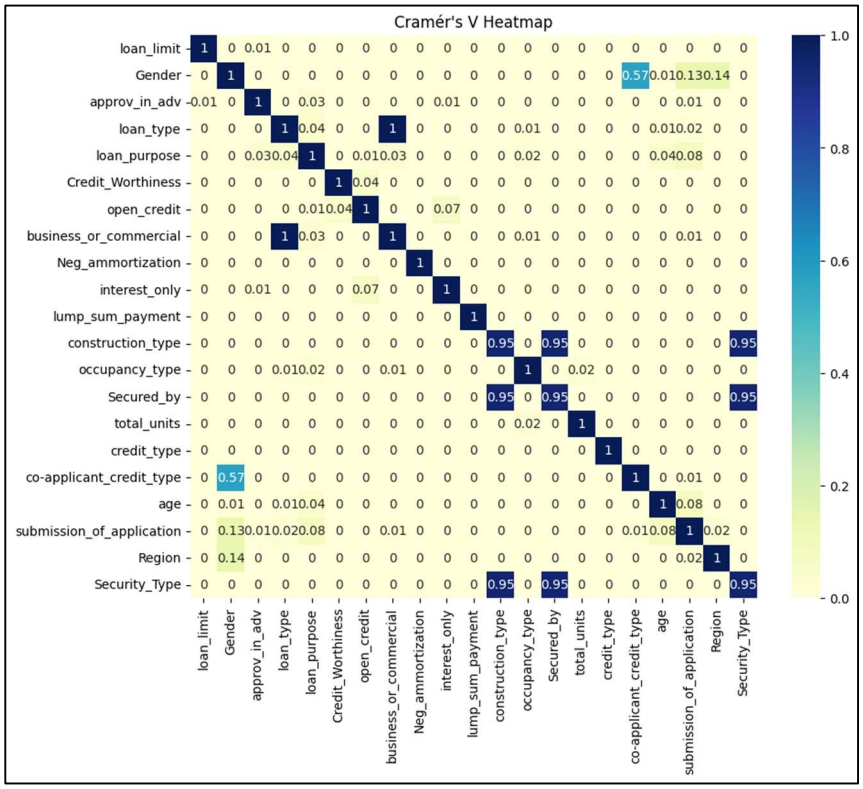


Figure 54: Categorical features correlation matrix

Annex J – Sha's Feature Engineering

Several techniques were explored:

(1) Imputation

For categorical columns, there are many null values found. These are replaced by looking for the highest frequency (i.e. mode) categorical values in each affected column. For numerical columns, there are many null values found. These are replaced by using k-NN imputer with k=5 in each affected column, since no data transformation was done to use the median or mode method.

(2) Handling outliers

For each numerical column, z-score was calculated for each value. Any values with z-score greater than threshold (usually 3) are considered outliers and are removed. Gaining some business knowledge, column LTV was flagged to have discrepancy or data issue. The value went over the standard range of 0-100% and those affected transactions are dropped.

(3) Feature selection

Bivariate analysis was explored through finding out the correlation between features in the dataset. For numerical columns, a standard correlation library was used and displayed on a heatmap to determine highly correlated features. For categorical columns, correlation was derived using V Cramer's method that makes use of statistical knowledge of chi contingency. The result is then displayed on a heatmap to determine highly correlated features. Several features are then chosen to be dropped.

(4) Data Transformation

After doing univariate analysis for each categorical feature, there are some categorical values that are of low frequency count. To reduce the total number of features after using one-hot encoding, some of the affected categorical values are combined. This also prevents them from being considered as non-useful as the variation gap is wide.

(5) Ordinal encoding

With use of business knowledge, certain columns are assumed to be Ordinal Data. This is by looking through the unique categorical values and seeing if the data can intrinsically be ranked or ordered. These are then converted using LabelEncoder to retain the predetermined or natural order characteristics.

(6) Nominal encoding

With use of business knowledge, certain columns are assumed to be Nominal Data. This is by looking through the unique categorical values and seeing if the data can be classified to fit into various groupings with no order or ranking. These are then converted using One-hot Encoder into binary like values.

(7) Scaling

Within the dataset, the numerical ranges between columns may differ quite largely. To standardize them, MinMaxScaler was applied so that the values will interpolate accordingly to the same range from 0 to 1. However, those columns that are LabelEncoded are not affected to retain its intrinsic ranking properties.