UNIVERSITY TUNKU ABDUL RAHMAN

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE

UECS3483/ UECS3213/ UECS3453 DATA MINING

JANUARY 2024 TRIMESTER

LECTURER: SU LEE SENG

No.	Student Name	Student ID	Programme	Practical Group
1.	Kim Bei Er	2103893	SE	P1
2.	Kuik Rui Luan	2103790	SE	P1
3.	Lim Juan Hong	2105435	SE	P1

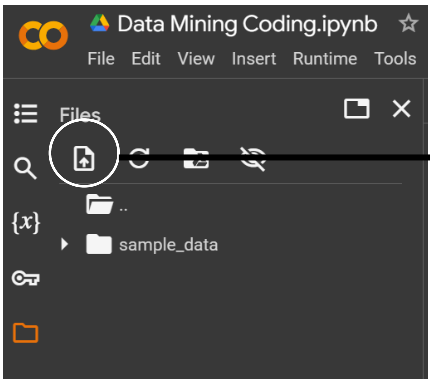
Table of Contents

README

Method 1 (Using Colab)

1. To run the code, kindly click the link below:
https://colab.research.google.com/drive/11tZIHH_TE18cNbZwuOjy7UDmwVv0deRe?usp=sharing

2. Upload 'BankLoanApproval.csv' and 'NewApplicants.csv' before running the code.



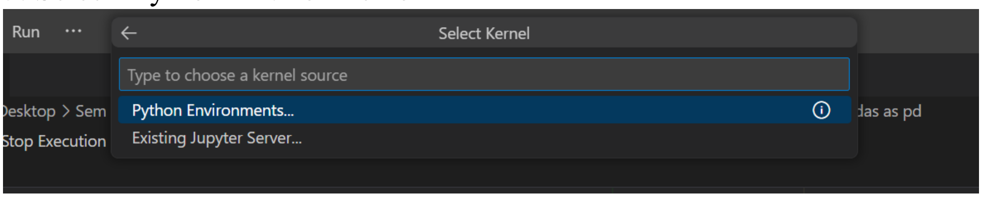
Click on the upload file icon to upload 'BankLoanApproval.csv' and 'NewApplicants.csv'

Method 2 (Using Visual Studio Code)

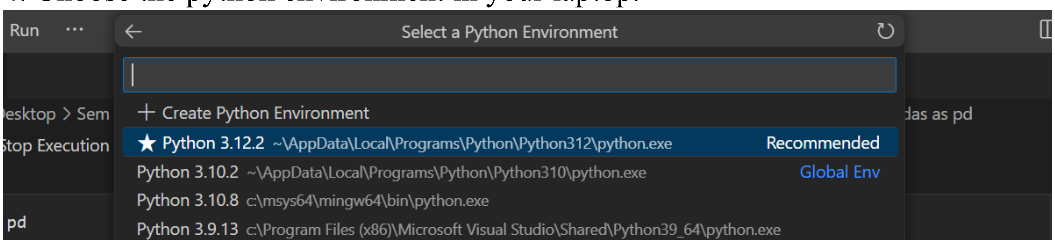
1. Open the 'Data Mining Coding.ipynb' in the submission folder using Visual Studio Code.

2. Click the 'Run' button

3. Select 'Python Environments'



4. Choose the python environment in your laptop.



1. Data Exploration

1.1 Variable Summarization

The "Bank Loan Approval" dataset provides information about individuals and their loans, with the objective to predict those at high risk of defaulting on their loans. This dataset comprises 18 key features that delve into various aspects of the applicant's background and loan specifics:

1. **LoanID**: A unique identifier for each loan, used for tracking purposes.
2. **Age**: Age of the borrower at the time of loan application, indicating their level of financial responsibility and stability.
3. **Income**: Annual income of the borrower, help in assess their ability to repay the loan.
4. **LoanAmount**: Amount of money being borrowed.
5. **CreditScore**: Credit score of the borrower, indicating their creditworthiness.
6. **MonthsEmployed**: Number of months the borrower has been employed, indicating their stability in their job.
7. **NumCreditLines**: Number of credit lines the borrower has open, reflecting their credit behavior.
8. **InterestRate**: Interest rate charged on the loan amount.
9. **LoanTerm**: Term length of the loan in months.
10. **DTIRatio**: Debt-to-Income ratio, indicating the borrower's debt compared to their income.
11. **Education**: Highest level of education attained by borrower (PhD, Master's, Bachelor's, High School), indicating their earning potential.
12. **EmployementType**: Type of employment status of the borrower (Full-time, Part-time, Self-employed, Unemployed), which affect their ability to repay the loan.
13. **MaritalStatus**: Marital status of the borrower (Single, Married, Divorced), which reflect their financial burden and responsibility.
14. **HasMortgage**: Whether the borrower has a mortgage (Yes or No), which can affect their financial obligations.
15. **HasDependents**: Whether the borrower has dependents (Yes or No), which can affect their financial responsibilities.
16. **LoanPurpose**: Purpose of loan (Home, Auto, Education, Business, Other), indicating the borrower's financial goals and intentions.

17. **HasCoSigner**: Whether the loan has a co-signer (Yes or No), which can affect the qualification for the loan.
18. **Default**: Binary target variable indicating whether the loan defaulted (1) or not (0), which is the variable to be predicted in the prediction model.

1.2 Objective, Potential Issue and Limitation

The objective of the "Bank Loan Approval" dataset is to develop a model to accurately classify the loan applications as 'Default' or 'Non-default' based on the application information. This prediction can help financial institutions or lenders make informed decisions about loan approvals to **minimize the financial losses** while **maximize the approval rate for deserving applicants**. One potential issue with this dataset in developing the machine learning model could be imbalanced classes, where the number of defaulters (1) is significantly lower than the number of non-defaulters. This imbalance could lead to biased models that are better at predicting non-defaulters than defaulters. However, there could be ethical considerations related to using certain features, such as age, education, or marital status, in predicting loan default risk, as they could lead to biased decisions.

2. Data Description

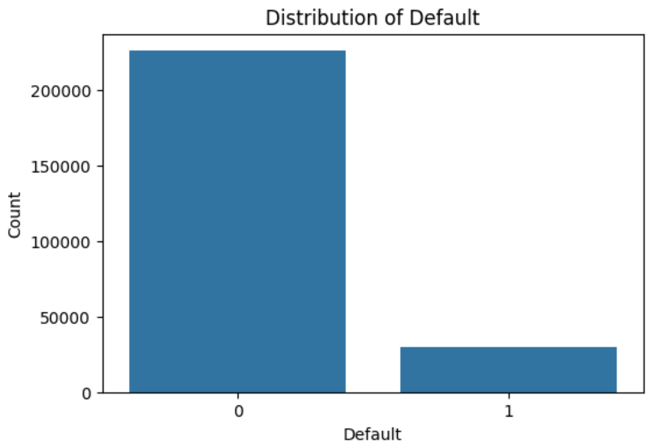
2.1 Descriptive Statistics

```
#Descriptive statistics
df.describe()
```

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Default
count	255327.000000	255327.000000	255327.000000	255327.000000	255327.000000	255327.000000	255327.000000	255327.000000	255327.000000	255327.000000
mean	43.498059	82500.225585	127579.236559	574.266125	59.542516	2.501036	13.492848	36.025896	0.500222	0.116118
std	14.990304	38963.150663	70841.308245	158.904496	34.643129	1.117021	6.636456	16.969297	0.230917	0.320367
min	18.000000	15000.000000	5000.000000	300.000000	0.000000	1.000000	2.000000	12.000000	0.100000	0.000000
25%	31.000000	48826.000000	66156.000000	437.000000	30.000000	2.000000	7.770000	24.000000	0.300000	0.000000
50%	43.000000	82467.000000	127557.000000	574.000000	60.000000	2.000000	13.460000	36.000000	0.500000	0.000000
75%	56.000000	116219.000000	188986.500000	712.000000	90.000000	3.000000	19.250000	48.000000	0.700000	0.000000
max	69.000000	149999.000000	249999.000000	849.000000	119.000000	4.000000	25.000000	60.000000	0.900000	1.000000

Descriptive Statistics

The descriptive statistics reveal valuable insights into the characteristics of the loan applicant dataset. On average, applicants are approximately 43.5 years old, with ages ranging from 18 to 69 years. The average income stands at $82,500, indicating a moderate-income level. However, the income distribution may exhibit a slight right skew, as the mean income slightly surpasses the median. Applicants request loans with varying amounts, as reflected by the average loan amount of $127,579 and a considerable standard deviation of $70,841. This wide variability suggests diverse financial needs among applicants. The creditworthiness of applicants, assessed by credit scores, with an average score of 573, indicating a good credit standing. Employment history among applicants varies, with an average tenure of approximately 59.5 months. Despite most applicants work for more than 30 months, there are also applicants who have never been employed and they might be university students or unemployed fresh graduate. The number of credit lines held by applicants ranges from 1 to 4, with the majority possessing 2 credit lines. Interest rates offered to applicants vary widely, ranging from 2% to 25%, with an average rate of approximately 13.49%. Loan terms span from 12 to 60 months, accommodating diverse preferences and financial circumstances. Applicants exhibit a diverse range of debt-to-income (DTI) ratios, with an average ratio of 0.5. The DTI ratio serves as a crucial indicator of an applicant's financial ability to manage debt obligations. Regarding loan defaults, approximately 11.6% of applicants defaulted on their loans. This proportion suggests potential imbalances in the dataset's target variable, which warrant careful consideration during model development and evaluation.

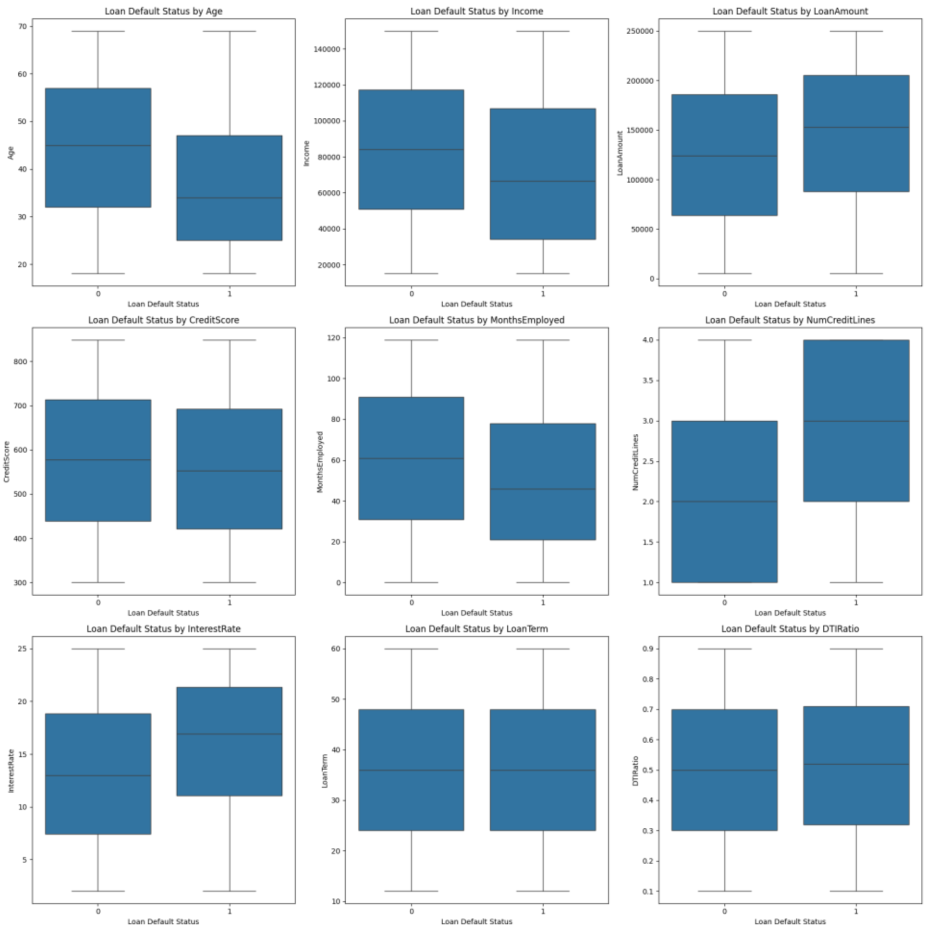



```
print("Number of class 0 (Non-default) instance: ", df[df["Default"] == 0].shape[0])
print("Number of class 1 (Default) instance: ", df[df["Default"] == 1].shape[0])
```

```
Number of class 0 (Non-default) instance:  225679
Number of class 1 (Default) instance:  29648
```

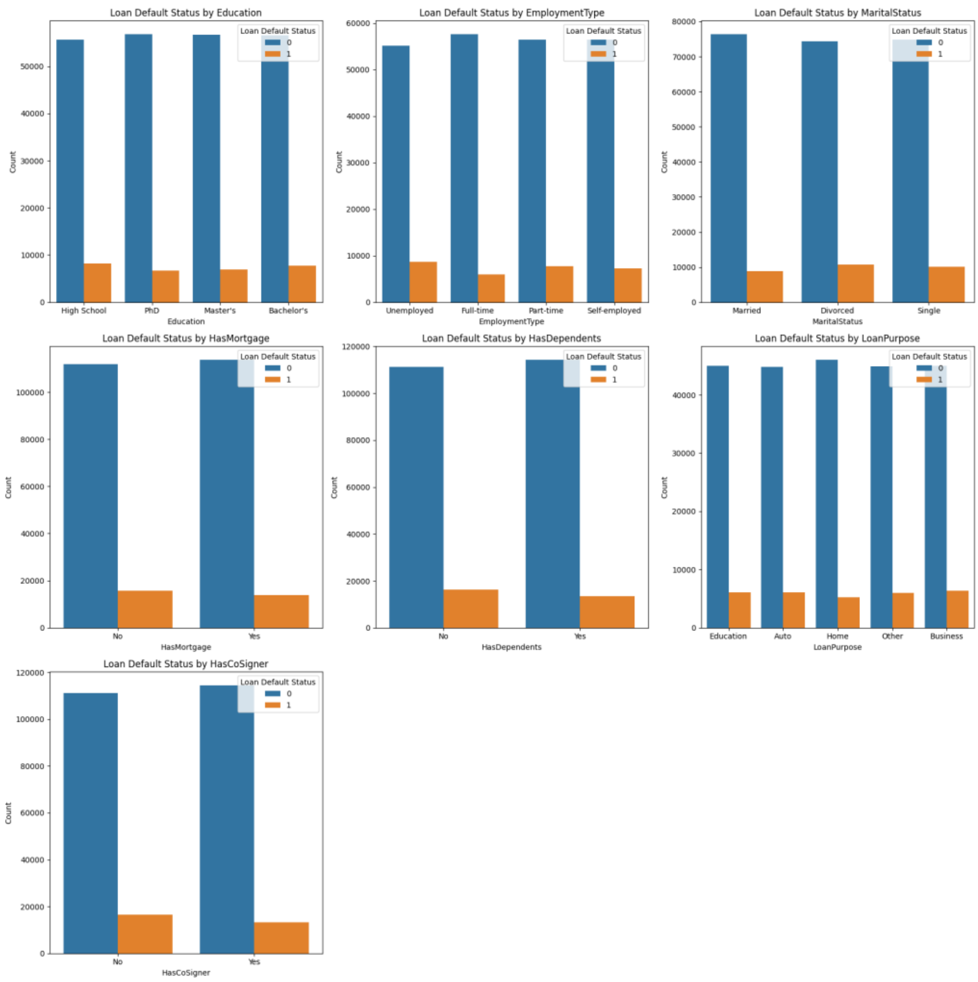
Count Plot for Default Variable

The count plot reveals that out of the total applicants, 225,679 individuals who did not default on their loans were approved for bank loans. Conversely, 29,648 applicants who defaulted on their loans had their bank loan requests rejected. This indicates the loan approval rate is relatively high. Moreover, the **ratio of class 0 (‘non-default’) to class 1 (‘default’)** is **<u>7.6:1</u>**. The significant disparity in the number of instances between the two classes indicates a **class imbalance**. Class imbalance can potentially impact the performance of machine learning models, particularly those trained on imbalanced datasets, as they may become biased towards the majority class.



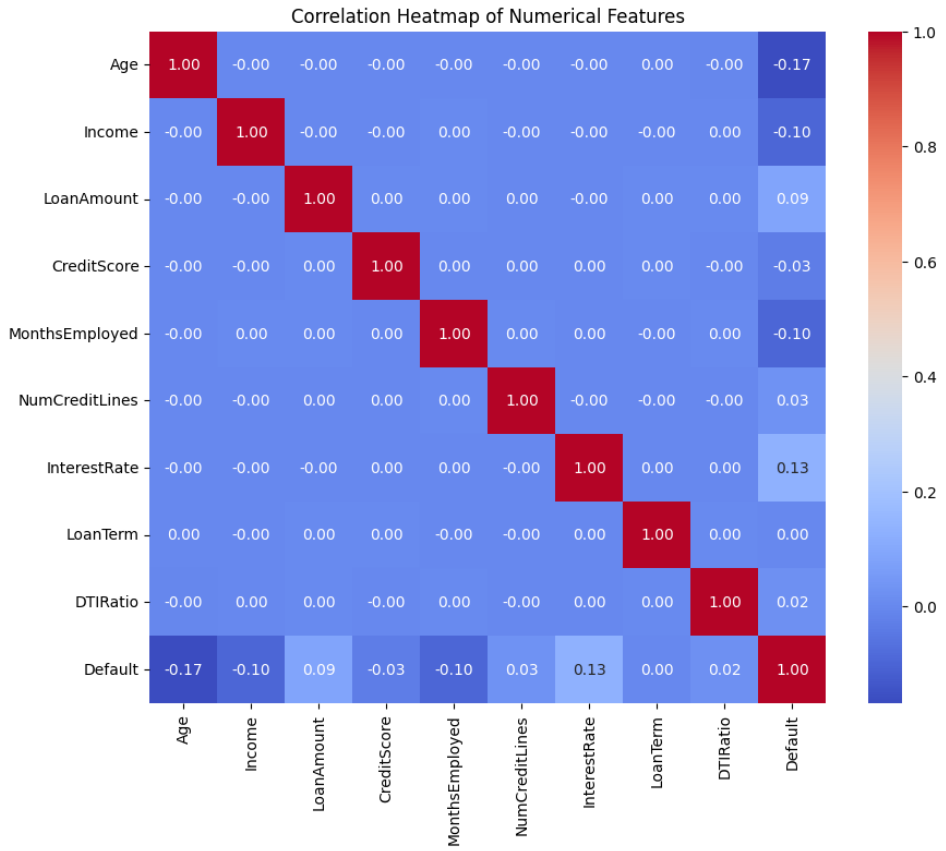
Boxplots for Numerical Variables

The boxplot reveals that **most younger applicants are more likely to default on their loans**. This may because young applicants may not have ability to pay the loan. Applicants with **higher income get to repay for their bank loans** as this may because of higher income individuals may have higher financial ability to pay back their loans. Applicants who **request for higher loan amount have higher probabilities on defaulting their loans** and this may pose higher risk to the bank. The credit score for both loan approval status is approximately the same but **non-defaulters have slightly higher credit scores** compared to defaulters, showcasing their higher creditworthiness. Applicants who have **worked for longer months are less likely to default on their loans.** This may be due to the applicants who have more working experience may have more income and more able to pay back the loan. **Non-defaulters mostly have less than 3 credit lines**, indicating they have less debt to pay. **Applicants who received lower interest rates** may have higher credit scores or better credit histories. Bank authorities may be more willing to approve loans for these applicants because they have demonstrated a lower likelihood of defaulting on their loans. The boxplots show similar distribution of loan terms for both 'Default' and 'Non-default' classes. This indicates that the **length of the loan term may not influence the likelihood of default**. Thus, it may not be a significant factor in determining whether an applicant defaults on their loan. Applicants with **lower Debt-to-Income (DTI) Ratio tends to pay back for their loans** compared to applicants with higher DTI Ratio. This is because lower DTI Ratio indicates that the applicant has a higher capability to manage debt payment. In summary, the boxplots of loan default status by nine of the numerical variables show the distribution of data within loan default status of 0 ('non-default') and 1 ('default') and there is no outlier observed.



Bar Plots for Categorical Variables

The bar plots depicting loan default status across seven categorical variables reveal a consistent distribution of unique values within each category, regardless of the loan default outcome (0 for 'non-default' and 1 for 'default'). This uniform distribution indicates that the presence of **these categorical variables does not significantly influence the likelihood of loan default**. This observation suggests that the variables represented by these categories may not have a strong predictive power in determining loan default. However, it is important to further investigate the relationship between these categorical variables and 'Default' variable through statistical tests or feature importance analysis to confirm their impact on the outcome.



Correlation Heatmap of Numerical Features

According to the correlation heatmap, there appears to be no significant correlation among the numerical features within the 'Bank Loan Approval' dataset, as indicated by null correlation coefficients. However, the Default variable shows some degree of correlation with other numerical variables. The **Default variable exhibits the strongest positive correlation (0.13) with the Interest Rate variable**, indicating that higher interest rates are associated with a higher likelihood of defaulting on loans. Additionally, there is a low positive correlation (0.09) between the Loan Amount variable and the Default variable, suggesting that larger loan amounts may receive rejection on bank loans due to the higher risk of default. Furthermore, the Default variable shows slight positive correlations with the Number of Credit Lines and DTI Ratio variables, implying that applicants with more credit lines or higher debt-to-income ratios may have a slightly increased possibility of defaulting.

Conversely, the **Age variable exhibits the strongest negative correlation (-0.17) with the Default variable**, indicating that older applicants may have a lower chance of defaulting on loans due to their financial stability. Similarly, the Income and Months Employed variables demonstrate negative correlations (-0.10) with the Default variable, suggesting that higher incomes and longer employment durations may be associated with a reduced risk of default. The Credit Score variable shows a weak negative correlation (-0.03) with the Default variable, implying that higher credit scores may be associated with a lower likelihood of default. Interestingly, the Loan Term variable does not appear to have a significant correlation with the Default variable, suggesting that the length of the **loan term may not strongly influence the loan default status.**

3. Data Preprocessing

3.1 Data Cleaning

```
[6] #Data preprocessing for training set
    #Check missing values for training set data
    print(x_train.isnull().sum())

    Age               0
    Income            0
    LoanAmount        0
    CreditScore       0
    MonthsEmployed    0
    NumCreditLines    0
    InterestRate      0
    LoanTerm          0
    DTIRatio          0
    Education         0
    EmploymentType    0
    MaritalStatus     0
    HasMortgage       0
    HasDependents     0
    LoanPurpose       0
    HasCoSigner       0
    dtype: int64
```

Check Missing Value

```
#Check duplicate rows for training set data
x_train.duplicated().any()

False
```

Check Duplicate Row

In this stage, we will handle the missing value and duplicate observation. In this case, the dataset does not have any missing value or duplicate observation. Thus, no action is needed for handling the missing value or duplicate row.

3.2 Feature Selection

```
Number of unique value in a variable
'LoanID':255327
'Age':52
'Income':114619
'LoanAmount':158725
'CreditScore':550
'MonthsEmployed':120
'NumCreditLines':4
'InterestRate':2301
'LoanTerm':5
'DTIRatio':81
'Education':4
'EmploymentType':4
'MaritalStatus':3
'HasMortgage':2
'HasDependents':2
'LoanPurpose':5
'HasCoSigner':2
'Default':2
```

Unique value in each variable

The 'LoanID' from the dataset appeared to be the unique value and it is useless in the analysis of data. In order to avoid the consumption of resource by irrelevant information that could impact the performance and accuracy, thus the **LoanID is dropped from dataset**.

```
#Define feature vector and target variable
x = df.drop(['LoanID', 'Default'], axis=1)
y = df['Default']
```

Define 'x' and 'y' Variables

'Default' variable is assigned to y as the target variable, while other features in DataFrame 'df' (except 'LoanID' and 'Default' variables) are assigned to x as independent variables.

3.3 Data Splitting

```
Training set size:  178728
Validation set size:  38299
Testing set size:  38300
```

Size of Each Data Set

The dataset is then split into training (70%), validation (15%), and testing (15%) sets. This split ensures the model is trained on a subset of data and evaluate at validation and testing dataset. **Splitting the dataset before preprocessing** helps in **avoiding the overfitting** and **data leakage** (Patil, 2024). If model learns patterns from the testing set or the entire dataset, it will result in overfitting. Data leakage can occur when preprocessing steps, such as imputing missing values are applied to the entire dataset before splitting, which will inadvertently introduce information from the testing set into the training process. Therefore, Patil (2024) suggested that splitting dataset before preprocessing will ensure the model is trained solely on patterns and characteristics inherent in the training data, rather than being influenced by specific patterns in the testing set. With that, the model can better generalize to unseen data and provide more reliable performance evaluations.

3.4 Detect and Remove Outlier

```
#Remove outliers in numerical variables using Z-score
from scipy import stats
z = np.abs(stats.zscore(x_train_numerical))
x_train2 = x_train_numerical[(z<3).all(axis=1)]
x_train2.shape
✓ 0.0s

(178728, 9)
```

Check and Remove Outlier

To handle the outlier, we split the 'x_train' variables into numerical and categorical variables. Then, we detect and remove the outlier by using the Z-score method with the threshold of 3. This is because Z-score is sensitive to the spread of data and identifies outlier based on their deviation from mean. However, the result shows that there is no outlier in the dataset.

3.5 Data Transformation and Standardization

Data transformation and standardization are important in preparing data for efficient analysis and modelling. It performs **one-hot encoding** and standardization on dataset to prepare it for machine learning modelling. The categorical variable is encoded by using pd.get_dummies() to convert it into binary variables. Dropping the first category in each categorical variable can avoid multicollinearity issues. This encoding ensures the variable can be used as input for machine learning algorithm. The numerical variable is also **standardized by using StandardScaler()** to avoid variable with larger magnitudes from dominating the model. This may standardize the range of features and make the training process more stable. Lastly, all the transformed variable is concatenated for model training.

```
             Age    Income  LoanAmount  CreditScore  MonthsEmployed  \
34705  -0.034294 -0.815934    0.637581    -0.372791       -0.072250
91534   1.632722 -1.392594   -0.768511     0.830369       -0.995964
135742  0.565831  0.140702   -0.573785    -0.139718        1.601982
38965   1.566041 -0.193464    1.568559    -0.026331        0.995795
152605 -1.567949  1.032511   -0.639786     0.024063        1.226723

        NumCreditLines  InterestRate  LoanTerm  DTIRatio  \
34705         0.445688     -1.406692 -0.707583  1.688273
91534         1.341111     -0.954764  1.411479  1.688273
135742        0.445688     -0.475720 -1.413937  1.255511
38965        -1.345159      0.628490  0.705125 -1.211232
152605        1.341111     -0.557068 -0.707583  0.043777

        Education_High School  ...  EmploymentType_Unemployed  \
34705                   False  ...                      False
91534                    True  ...                      False
135742                  False  ...                       True
38965                    True  ...                       True
152605                  False  ...                      False

        MaritalStatus_Married  MaritalStatus_Single  HasMortgage_Yes  \
34705                    True                 False            False
91534                    True                 False             True
135742                  False                  True            False
...
38965              False              False             True
152605             False              False             True

[5 rows x 24 columns]
```

Training dataset after encoding and standardization

```
             Age    Income  LoanAmount  CreditScore  MonthsEmployed  \
249374  1.242256  0.036544   -0.881548    -0.302496        0.731129
209752  0.974809  0.364488    1.235472    -0.672763       -1.549465
151150  0.172469 -1.592574    0.746812    -1.538809        0.990943
215987 -1.499072  1.300999   -1.401759    -0.227188       -0.336997
162055 -1.164764  0.495491   -0.087641     1.354287       -0.914363

        NumCreditLines  InterestRate  LoanTerm  DTIRatio  \
249374        1.348297     -0.275680 -0.707070  1.555027
209752       -0.443180      0.593941 -1.417833 -0.702950
151150       -0.443180     -0.841385  0.714456  0.252348
215987       -0.443180      0.794044 -0.707070  0.686574
162055       -0.443180      1.510203  1.425219  0.947110

        Education_High School  ...  EmploymentType_Unemployed  \
249374                  False  ...                      False
209752                  False  ...                      False
151150                  False  ...                      False
215987                  False  ...                       True
162055                  False  ...                      False

        MaritalStatus_Married  MaritalStatus_Single  HasMortgage_Yes  \
249374                  False                  True            False
209752                   True                 False             True
151150                  False                  True            False
...
215987              True              False            False
162055             False               True             True

[5 rows x 24 columns]
```

Validation dataset after encoding and standardization

```
             Age    Income  LoanAmount  CreditScore  MonthsEmployed  \
65844   0.233452  0.619176   -0.553485    -1.304957       -0.738768
126589 -1.100479  0.482542   -0.878464     0.899993       -0.854218
229083 -0.033334  0.063220   -1.188118    -1.179319        1.310466
148419 -0.233424 -0.265123   -1.592375    -0.595101       -1.431467
25019   1.567382  0.675098   -1.720407     1.660104        1.714541

        NumCreditLines  InterestRate  LoanTerm  DTIRatio  \
65844         1.339434     -0.328848  0.699276  1.645913
126589        0.445529     -0.242718 -0.715492 -1.560416
229083        0.445529      0.876979  1.406660 -0.390539
148419        0.445529     -1.295928 -0.008108  0.779337
25019        -0.448376     -0.307693  1.406660 -1.127128

        Education_High School  ...  EmploymentType_Unemployed  \
65844                   False  ...                       True
126589                  False  ...                       True
229083                  False  ...                       True
148419                  False  ...                       True
25019                   False  ...                      False

        MaritalStatus_Married  MaritalStatus_Single  HasMortgage_Yes  \
65844                    True                 False            False
126589                  False                 False            False
229083                  False                 False            False
...
148419             False              False             True
25019               True              False            False

[5 rows x 24 columns]
```

Testing dataset after encoding and standardization

4. Data Mining Model

4.1 Model Training (Without Resampling)

After data preprocessing, we develop seven data mining models to evaluate model and perform model comparison and selection. Below are the seven models with their corresponding performance metrics, confusion matrix and classification report.

```
Validation Set Metrics:
Accuracy: 0.8856889213817594
Precision: 0.6175298804780877
Recall: 0.03493351363533919
F1-score: 0.06612627986348123

Confusion Matrix for validation dataset:
[[33766    96]
 [ 4282   155]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     33862
           1       0.62      0.03      0.07      4437

    accuracy                           0.89     38299
   macro avg       0.75      0.52      0.50     38299
weighted avg       0.86      0.89      0.84     38299
```

```
Testing Set Metrics:
Accuracy: 0.8838642297650131
Precision: 0.610236220472441
Recall: 0.03441385435168739
F1-score: 0.06515342580916351

Confusion Matrix:
[[33697    99]
 [ 4349   155]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     33796
           1       0.61      0.03      0.07      4504

    accuracy                           0.88     38300
   macro avg       0.75      0.52      0.50     38300
weighted avg       0.85      0.88      0.84     38300
```

Logistic Regression

```
Validation Set Metrics:
Accuracy: 0.8853494869317736
Precision: 0.6197916666666666
Recall: 0.02681992337164751
F1-score: 0.0514149924389717

Confusion Matrix for validation dataset:
[[33789    73]
 [ 4318   119]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     33862
           1       0.62      0.03      0.05      4437

    accuracy                           0.89     38299
   macro avg       0.75      0.51      0.50     38299
weighted avg       0.86      0.89      0.84     38299
```

```
Testing Set Metrics:
Accuracy: 0.8838903394255875
Precision: 0.6446700507614214
Recall: 0.02819715808170515
F1-score: 0.054031057221867684

Confusion Matrix:
[[33726    70]
 [ 4377   127]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     33796
           1       0.64      0.03      0.05      4504

    accuracy                           0.88     38300
   macro avg       0.76      0.51      0.50     38300
weighted avg       0.86      0.88      0.83     38300
```

Naïve Bayes

```
Validation Set Metrics:
Accuracy: 0.8025535914775843
Precision: 0.1987661461345672
Recall: 0.23236421005183683
F1-score: 0.21425602660016624

Confusion Matrix for validation dataset:
[[29706  4156]
 [ 3406  1031]]
              precision    recall  f1-score   support

           0       0.90      0.88      0.89     33862
           1       0.20      0.23      0.21      4437

    accuracy                           0.80     38299
   macro avg       0.55      0.55      0.55     38299
weighted avg       0.82      0.80      0.81     38299
```

```
Testing Set Metrics:
Accuracy: 0.8001044386422976
Precision: 0.1977368622938243
Recall: 0.2289076376554174
F1-score: 0.21218357686766826

Confusion Matrix:
[[29613  4183]
 [ 3473  1031]]
              precision    recall  f1-score   support

           0       0.90      0.88      0.89     33796
           1       0.20      0.23      0.21      4504

    accuracy                           0.80     38300
   macro avg       0.55      0.55      0.55     38300
weighted avg       0.81      0.80      0.81     38300
```

Decision Tree

```
Validation Set Metrics:
Accuracy: 0.8861589075432779
Precision: 0.6842105263157895
Recall: 0.03222898354744196
F1-score: 0.06155832974601808

Confusion Matrix for validation dataset:
[[33796    66]
 [ 4294   143]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     33862
           1       0.68      0.03      0.06      4437

    accuracy                           0.89     38299
   macro avg       0.79      0.52      0.50     38299
weighted avg       0.86      0.89      0.84     38299
```

```
Testing Set Metrics:
Accuracy: 0.8839164490861618
Precision: 0.6559139784946236
Recall: 0.02708703374777975
F1-score: 0.05202558635394456

Confusion Matrix:
[[33732    64]
 [ 4382   122]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     33796
           1       0.66      0.03      0.05      4504

    accuracy                           0.88     38300
   macro avg       0.77      0.51      0.50     38300
weighted avg       0.86      0.88      0.83     38300
```

Random Forest

```
Validation Set Metrics:
Accuracy: 0.8744092535053134
Precision: 0.3184031158714703
Recall: 0.07369844489519946
F1-score: 0.11969253294289897

Confusion Matrix for validation dataset:
[[33162   700]
 [ 4110   327]]
              precision    recall  f1-score   support

           0       0.89      0.98      0.93     33862
           1       0.32      0.07      0.12      4437

    accuracy                           0.87     38299
   macro avg       0.60      0.53      0.53     38299
weighted avg       0.82      0.87      0.84     38299
```

```
Testing Set Metrics:
Accuracy: 0.8742558746736292
Precision: 0.34462151394422313
Recall: 0.07682060390763766
F1-score: 0.12563543936092955

Confusion Matrix:
[[33138   658]
 [ 4158   346]]
              precision    recall  f1-score   support

           0       0.89      0.98      0.93     33796
           1       0.34      0.08      0.13      4504

    accuracy                           0.87     38300
   macro avg       0.62      0.53      0.53     38300
weighted avg       0.82      0.87      0.84     38300
```

K-Nearest Neighbors

```
Validation Set Metrics:
Accuracy: 0.8841484111856707
Precision: 0.0
Recall: 0.0
F1-score: 0.0

Confusion Matrix for validation dataset:
[[33862     0]
 [ 4437     0]]
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     33862
           1       0.00      0.00      0.00      4437

    accuracy                           0.88     38299
   macro avg       0.44      0.50      0.47     38299
weighted avg       0.78      0.88      0.83     38299
```

```
Testing Set Metrics:
Accuracy: 0.8824281984334204
Precision: 1.0
Recall: 0.00022202486678507994
F1-score: 0.0004439511653718091

Confusion Matrix:
[[33796     0]
 [ 4503     1]]
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     33796
           1       1.00      0.00      0.00      4504

    accuracy                           0.88     38300
   macro avg       0.94      0.50      0.47     38300
weighted avg       0.90      0.88      0.83     38300
```

Support Vector Machine

```
Validation Set Metrics:
Accuracy: 0.8861327972009713
Precision: 0.5501319261213721
Recall: 0.09398242055442867
F1-score: 0.16053897978825793

Confusion Matrix for validation dataset:
[[33521   341]
 [ 4020   417]]
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     33862
           1       0.55      0.09      0.16      4437

    accuracy                           0.89     38299
   macro avg       0.72      0.54      0.55     38299
weighted avg       0.85      0.89      0.85     38299
```

```
Testing Set Metrics:
Accuracy: 0.8833681462140992
Precision: 0.5257301808066759
Recall: 0.08392539964476022
F1-score: 0.14474439977024697

Confusion Matrix:
[[33455   341]
 [ 4126   378]]
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     33796
           1       0.53      0.08      0.14      4504

    accuracy                           0.88     38300
   macro avg       0.71      0.54      0.54     38300
weighted avg       0.85      0.88      0.84     38300
```

Neural Network

Based on the results of the seven models, it is evident that there is a significant **class imbalance issue**. Firstly, the **recall scores for class 1 ('default') which is the minority class are considerably low**. For class 1, most of the models achieved low recall of 0.03 in validation and testing set, and Support Vector Machine even achieved recall score of 0 in both validation and testing set. Low recall means high false negative, indicating that the models have high chance to miss a large number of actual default applicants. Failing to identify actual default applicants may lead to loss of revenue of the bank authorities and increases the risk of loan defaulted by the applicants. Hence, it is important to have higher recall score in this scenario. Besides, the **confusion matrices for both validation and testing sets of every model appeared to be imbalance**. The training dataset has an uneven distribution of classes, where the number of proportion of instances for class 0 ('non-default') significantly outweighs the class 1 ('default'). The **F1-score for minority class ('default') is observed to be low** across the seven models. Low F1-score means the model has low ability in classifying the applicants correctly to default and non-default categories. Given these reasons, we decided to perform **resampling** on the dataset to address the class imbalance issue and reduce bias towards the majority class ('non-default').

4.2 Data Resampling Technique

To address the issue of imbalanced class, we performed oversampling on the minority class ('default') using **Synthetic Minority Over-sampling Technique (SMOTE)** which generates synthetic samples of minority class (Satpathy, 2020).

```
(array([0, 1], dtype=int64), array([158021,  20707], dtype=int64))
(array([0, 1], dtype=int64), array([158021, 158021], dtype=int64))
```

Size of 'Default' variable in training data set before and after resampling

After resampling, both classes now have 158021 instances, and we can repeat the previous model training process on this resampled dataset.

4.3 Model after Resampling and Hyperparameter Tuning

4.3.1 Logistic Regression

Logistic Regression is widely used for binary classification problems, where the goal is to predict the probability of an observation belongs to either one of two classes for binary classification (Wolff, 2020).

The Logistic Regression is initiated and fitted on training data. We made prediction on validation set and testing set using the trained model to calculate the evaluation metric and confusion matrix. The report for validation set and testing set are shown below.

```
Validation Set Metrics:
Accuracy: 0.7028643045510327
Precision: 0.2049800288943656
Recall: 0.5436105476673428
F1-score: 0.2977042705504814

Confusion Matrix for validation dataset:
[[24507  9355]
 [ 2025  2412]]
              precision    recall  f1-score   support

           0       0.92      0.72      0.81     33862
           1       0.20      0.54      0.30      4437

    accuracy                           0.70     38299
   macro avg       0.56      0.63      0.55     38299
weighted avg       0.84      0.70      0.75     38299
```

```
Testing Set Metrics:
Accuracy: 0.7066579634464752
Precision: 0.21163567817667722
Recall: 0.5484014209591475
F1-score: 0.3054095826893354

Confusion Matrix:
[[24595  9201]
 [ 2034  2470]]
              precision    recall  f1-score   support

           0       0.92      0.73      0.81     33796
           1       0.21      0.55      0.31      4504

    accuracy                           0.71     38300
   macro avg       0.57      0.64      0.56     38300
weighted avg       0.84      0.71      0.75     38300
```

Based on the report, the accuracy for the validation set metrics is 0.70 and for testing set is 0.71 which are moderate. The precision for class 0 ('non-default') is 0.92 for validation and testing set. The precision is 0.20 and 0.21 for class 1 ('default') in validation and testing set. This show that there is a high number of false positives. Besides, the recall for the report is 0.54 and 0.55 for class 1 in validation and testing set while for class 0 is 0.72 and 0.73 respectively. This indicates that the model can capture a good portion of actual positive. However, the F1-score for class 1 in validation and testing set is 0.30 and 0.31 which are considered low while for class 0 is 0.81 for both dataset which is considered good.

```
Best Hyperparameters: {'C': 0.001, 'penalty': 'l2', 'solver': 'liblinear'}
Best Cross-Validation F1 Score: 0.7358528306859768
```

To enhance model simplicity and interpretability, we utilize Grid Search Cross Validation for hyperparameter tuning. The best hyperparameter tuning is 0.001 of regularization strength, L2 penalty and 'liblinear' solver. The corresponding best cross-validation F1-score is 0.74, which is moderate. Then, the Logistic Regression model is trained again using the best hyperparameters and evaluate the model performance using validation and testing set.

```
Validation Set Metrics:
Accuracy: 0.695814512128254
Precision: 0.21196390064691317
Recall: 0.5981519044399369
F1-score: 0.313008609505838

Confusion Matrix for validation dataset:
[[23995  9867]
 [ 1783  2654]]
              precision    recall  f1-score   support

           0       0.93      0.71      0.80     33862
           1       0.21      0.60      0.31      4437

    accuracy                           0.70     38299
   macro avg       0.57      0.65      0.56     38299
weighted avg       0.85      0.70      0.75     38299
```

```
Testing Set Metrics:
Accuracy: 0.6984856396866841
Precision: 0.21895946377274178
Recall: 0.6092362344582594
F1-score: 0.3221413477342099

Confusion Matrix:
[[24008  9788]
 [ 1760  2744]]
              precision    recall  f1-score   support

           0       0.93      0.71      0.81     33796
           1       0.22      0.61      0.32      4504

    accuracy                           0.70     38300
   macro avg       0.58      0.66      0.56     38300
weighted avg       0.85      0.70      0.75     38300
```

According to the result after tuning, the **performance of the precision, recall and F1-score increase.** Although, there is a slight decrease in the accuracy in validation and testing set which is 0.70 for both sets. The precision increase to 0.21 and 0.22 for class 1 (‘default’) in validation and testing set while class 0 (‘non-default’) precision increase from 0.92 to 0.93 for both sets. This indicate that there is a decrease in number of false positive. The recall for class 1 (‘default’) increases significantly for both set, which are 0.60 for validation set and 0.61 for testing set. The increase of number show that the model can capture more numbers of actual positive. However, the recall for class 0 (‘non-default’) in both dataset decreases slightly to 0.71. F1-score for class 1 (‘default’) is increased to 0.31 and 0.32 for validation and testing set. This indicates a better balance between the precision and recall which is important in classification task especially dealing with imbalance datasets. The improved of performance on both validation and testing set show that the model with tuning is more suitable to make accurate predictions on unseen data. So, we can conclude that the **model after tuning might be more suitable to perform model comparison.**

4.3.2 Naïve Bayes

Naïve Bayes classifier works on Bayes’ theorem, where a feature is assumed to be independent from other features. Particularly, Gaussian naive Bayes is used in our model training for classification purposes (Shetty, 2023).

The Naïve Bayes model is instantiated and trained on training data. Prediction is made on validation set and testing set using the trained model to calculate the evaluation metric and confusion matrix. The report for validation set and testing set are shown below.

```
Validation Set Metrics:
Accuracy: 0.6955272983628815
Precision: 0.2126034373010821
Recall: 0.6022086995717827
F1-score: 0.31426051161423113

Confusion Matrix for validation dataset:
[[23966  9896]
 [ 1765  2672]]
              precision    recall  f1-score   support

           0       0.93      0.71      0.80     33862
           1       0.21      0.60      0.31      4437

    accuracy                           0.70     38299
   macro avg       0.57      0.65      0.56     38299
weighted avg       0.85      0.70      0.75     38299
```

```
Testing Set Metrics:
Accuracy: 0.6981984334203656
Precision: 0.2186328467735503
Recall: 0.6085701598579041
F1-score: 0.3216947362244

Confusion Matrix:
[[24000  9796]
 [ 1763  2741]]
              precision    recall  f1-score   support

           0       0.93      0.71      0.81     33796
           1       0.22      0.61      0.32      4504

    accuracy                           0.70     38300
   macro avg       0.58      0.66      0.56     38300
weighted avg       0.85      0.70      0.75     38300
```

From the evaluation metrics, we can observe that the classifier achieved an accuracy of 0.70 on both the validation set and testing set. This indicates that the model correctly predicted the class label for about 69.5% - 69.8% of the instances in both sets. By looking at the precision of class 1 in both sets, it shows that all the instances predicted as default only about 21% - 22% were actual default loans. Whereas for class 0 (‘non-default’), the precision achieves 0.93 for both sets. The recall score for class 0 (‘non-default’) is 0.71 in both sets while for class 1 (‘default’) is 0.60 for validation set and 0.61 for testing set. The F1-score for class 1 (‘default’) is approximately 0.31 on the validation set and 0.32 on the testing set while for class 0 (‘non-default’) is 0.80 for validation set and 0.81 for testing set. This metric provides a balance between precision and recall. In both sets, the classifier performed better at predicting non-default loans (class 0) than default loans (class 1). Hence, the SMOTE resampling improves the recall and F1-score for class 1 compared to a non-resampled scenario.

4.3.3 Decision Tree

Decision Tree Model which consists of root node, branches, internal and lead nodes, forming a hierarchical tree structure, serves as a non-parametric learning algorithm used for classification and regression tasks (IBM, 2023b).

We instantiated the Decision Tree Classifier and train the model on the training data. We made predictions on the validation set and testing set using the trained model and calculate the evaluation metrics and confusion matrix. The classification reports for validation set and training set are shown below.

```
Validation Set Metrics:
Accuracy: 0.39951434763309746
Precision: 0.13721121144599507
Recall: 0.7910750507099391
F1-score: 0.2338596841894863

Confusion Matrix for validation dataset:
[[11791 22071]
 [  927  3510]]
              precision    recall  f1-score   support

           0       0.93      0.35      0.51     33862
           1       0.14      0.79      0.23      4437

    accuracy                           0.40     38299
   macro avg       0.53      0.57      0.37     38299
weighted avg       0.84      0.40      0.47     38299
```

```
Testing Set Metrics:
Accuracy: 0.4097389033942559
Precision: 0.13908050560229673
Recall: 0.7744227353463587
F1-score: 0.23581110773079134

Confusion Matrix:
[[12205 21591]
 [ 1016  3488]]
              precision    recall  f1-score   support

           0       0.92      0.36      0.52     33796
           1       0.14      0.77      0.24      4504

    accuracy                           0.41     38300
   macro avg       0.53      0.57      0.38     38300
weighted avg       0.83      0.41      0.49     38300
```

Based on the classification report, the accuracy of validation set is 0.40 while for testing set is 0.41 which are quite low. The precision for both classes is low, which is around 0.14 for both datasets, indicating among all the applicants only small proportion were correctly predicted and many non-default applicants were predicted as default applicants. In contrast, the precision for class 0 (‘non-default’) is high, which is 0.93 for validation set and 0.92 for testing set. Besides, the recall for class 1 (‘default’) is 0.79 for validation set while 0.77 for testing set, indicating that the model can capture a good portion of actual default applicants. Whereas recall for class 0 (‘non-default’) is 0.35 for validation set and 0.36 for testing set. The f1-score for class 1 (‘default’) in validation set and testing set are low which are 0.23 and 0.24 respectively, while the f1-score for class 0 (‘non-default’) in validation set is 0.51 and testing set is 0.52. This indicates that the model ability to identify applicants who will default on their loans is limited and there might be potential risk to miss true default applicants.

```
Best Hyperparameters: {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best Cross-Validation F1 Score: 0.8356757452441291
```

To improve the performance of the model, we perform hyperparameter tuning using Grid Search Cross Validation as Grid Search CV perform exhaustive search across all possible combinations of specified hyperparameters to find the best hyperparameters. The best hyperparameters are where entropy is selected for criterion, maximum depth is set to none, minimum samples leaf is 1 and minimum samples split is 2. The corresponding best cross-validation F1-score is 0.84, which is relatively high. Then, the Decision Tree Model is trained again using the best hyperparameters to evaluate newly trained model’s performance using validation set and testing set.

```
Validation Set Metrics:
Accuracy: 0.31828507271730333
Precision: 0.12590623489608507
Recall: 0.8219517692134325
F1-score: 0.21836362003412868

Confusion Matrix for validation dataset:
[[ 8543 25319]
 [  790  3647]]
              precision    recall  f1-score   support

           0       0.92      0.25      0.40     33862
           1       0.13      0.82      0.22      4437

    accuracy                           0.32     38299
   macro avg       0.52      0.54      0.31     38299
weighted avg       0.82      0.32      0.38     38299
```

```
Testing Set Metrics:
Accuracy: 0.33467362924281985
Precision: 0.1276799659236121
Recall: 0.7986234458259325
F1-score: 0.22016158648549394

Confusion Matrix:
[[ 9221 24575]
 [  907  3597]]
              precision    recall  f1-score   support

           0       0.91      0.27      0.42     33796
           1       0.13      0.80      0.22      4504

    accuracy                           0.33     38300
   macro avg       0.52      0.54      0.32     38300
weighted avg       0.82      0.33      0.40     38300
```

According to the classification report, the accuracy of both validation and testing set decreases. The precision for class 1 (‘default’) and class 0 (‘non-default’) in validation and testing set also decreases by 1% respectively. The recall for class 1 (‘default’) in validation set and testing set increases and but decreases for class 0 (‘non-default’). The f1-score for both classes decreases in both datasets. This indicates that the Decision Tree **model before tuning has better performance**, thus the previous trained model will be used to in comparison later.

4.3.4 Random Forest

A Random Forest Model is a supervised machine learning algorithm. It is based on the combination of multiple decision trees to reach a single result and it is effective in handling the classification task in this assignment (IBM, 2023c).

We instantiated the Random Forest Classifier with a random state of 42 to produce same results across different runs. After training the model on the training data, we made predictions on the validation set and testing set using the trained model and calculate the performance metrics and confusion matrix. The classification reports for validation set and training set are shown below.

```
Validation Set Metrics:
Accuracy: 0.7072508420585394
Precision: 0.20172580787179714
Recall: 0.5163398692810458
F1-score: 0.2901101684183867

Confusion Matrix for validation dataset:
[[24796  9066]
 [ 2146  2291]]
              precision    recall  f1-score   support

           0       0.92      0.73      0.82     33862
           1       0.20      0.52      0.29      4437

    accuracy                           0.71     38299
   macro avg       0.56      0.62      0.55     38299
weighted avg       0.84      0.71      0.75     38299
```

```
Testing Set Metrics:
Accuracy: 0.7019321148825065
Precision: 0.2059223961878829
Recall: 0.5373001776198935
F1-score: 0.29773622047244097

Confusion Matrix:
[[24464  9332]
 [ 2084  2420]]
              precision    recall  f1-score   support

           0       0.92      0.72      0.81     33796
           1       0.21      0.54      0.30      4504

    accuracy                           0.70     38300
   macro avg       0.56      0.63      0.55     38300
weighted avg       0.84      0.70      0.75     38300
```

Based on the classification report, the accuracy of validation set is 0.71 while for testing set is 0.70 which are moderate. The precision for class 1 ('default') is low, which is around 0.2 for validation set while 0.21 for testing set, indicating a high number of false positives in which many non-default applicants are predicted as default applicants. For class 0 ('non-default'), the precision for validation and testing set is 0.92. Besides, the recall for class 1 ('default') is 0.52 for validation set while 0.54 for testing set, indicating that the model can capture a good portion of actual positive instances where most default applicants are captured correctly. For class 0 ('non-default'), the recall for validation set is 0.73 and testing set is 0.72. However, the f1-score for class 1 ('default') in validation set and testing set are low which are 0.29 and 0.3 respectively. The f1-score for class 0 ('non-default') in validation set is 0.82 while testing set is 0.81.

```
Best Hyperparameters: {'n_estimators': 200, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_depth': None, 'bootstrap': False}
Best Cross-Validation F1 Score: 0.9301695872977236
```

After that, to explore optimal set of hyperparameters, we performed hyperparameter tuning with Randomized Search Cross Validation as it is scalable to large hyperparameter search spaces with high-dimensional data. The best hyperparameters are 200 n_estimators, 5 minimum number of samples to be split, 2 min_samples_leaf, unlimited depth, and bootstrap is set to false. The corresponding best cross-validation F1-score is 0.93, which is relatively high. Then, we used the best hyperparameters found to instantiate a new Random Forest Classifier and make predictions on the validation set and testing set using the tuned model.

```
Validation Set Metrics:
Accuracy: 0.7009582495626517
Precision: 0.1985736380821447
Recall: 0.5208474194275411
F1-score: 0.28752721617418353

Confusion Matrix for validation dataset:
[[24535  9327]
 [ 2126  2311]]
              precision    recall  f1-score   support

           0       0.92      0.72      0.81     33862
           1       0.20      0.52      0.29      4437

    accuracy                           0.70     38299
   macro avg       0.56      0.62      0.55     38299
weighted avg       0.84      0.70      0.75     38299
```

```
Testing Set Metrics:
Accuracy: 0.6998955613577024
Precision: 0.2054113283884019
Recall: 0.5410746003552398
F1-score: 0.2977761485826002

Confusion Matrix:
[[24369  9427]
 [ 2067  2437]]
              precision    recall  f1-score   support

           0       0.92      0.72      0.81     33796
           1       0.21      0.54      0.30      4504

    accuracy                           0.70     38300
   macro avg       0.56      0.63      0.55     38300
weighted avg       0.84      0.70      0.75     38300
```

According to the result after tuning, the overall performance of the metrics remained unchanged except for recall score and f1-score which decreases by 1% for class 0 ('non-default') in validation set. This indicates that the default hyperparameters for the **model before tuning**

contribute to better performance. Thus, we will use the trained Random Forest Model before tuning to perform model comparison.

4.3.5 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) algorithm is a supervised learning which utilizes proximity to classify or predict the grouping of individual data points (IBM, 2023d).

We instantiated the K-Nearest Neighbors (KNN) Model and trained the model on the training data. Predictions are made on the validation set and testing set using the trained model and calculate the performance metrics and confusion matrix. The classification reports for validation set and training set are shown below.

```
Validation Set Metrics:
Accuracy: 0.6926029400245437
Precision: 0.18115438108484005
Recall: 0.4696867252648186
F1-score: 0.26146414904962045

Confusion Matrix for validation dataset:
[[24442  9420]
 [ 2353  2084]]
              precision    recall  f1-score   support

           0       0.91      0.72      0.81     33862
           1       0.18      0.47      0.26      4437

    accuracy                           0.69     38299
   macro avg       0.55      0.60      0.53     38299
weighted avg       0.83      0.69      0.74     38299
```

```
Testing Set Metrics:
Accuracy: 0.6914099216710182
Precision: 0.17868751647193182
Recall: 0.4515985790408526
F1-score: 0.25605841253855355

Confusion Matrix:
[[24447  9349]
 [ 2470  2034]]
              precision    recall  f1-score   support

           0       0.91      0.72      0.81     33796
           1       0.18      0.45      0.26      4504

    accuracy                           0.69     38300
   macro avg       0.54      0.59      0.53     38300
weighted avg       0.82      0.69      0.74     38300
```

After SMOTE resampling, the KNN model's performance improved compared to the previous result. The accuracy in both datasets achieved 0.69. However, the precision for class 1 ('default') in both datasets is low, which is 0.18. In contrast, the precision for class 0 ('non-default') is 0.91. In the validation set and testing set, the recall in class 1 ('default') increased significantly to 0.47 and 0.45 respectively, indicating that the model was better at identifying actual default cases. Whereas for class 0 ('non-default'), the recall for both datasets is 0.72. The F1-score also improved after resampling, reaching a value of 0.26 for class 1 ('default') in both datasets and 0.81 for class 0 ('non-default'). This indicates a more balanced performance between precision and recall compared to the initial model. However, the slight drop in accuracy compared with non-resampling may be attributed to the increase in false positives, which indicates that the model is less biased towards predicting non-default cases. Also, by increasing the number of minority class samples, SMOTE can also lead to more false positives, further contributing to the drop in precision.

```
Best Hyperparameters: {'metric': 'euclidean', 'n_neighbors': 4, 'weights': 'uniform'}
Best Cross-Validation F1 score: 0.9060810281750491
```

After that, we performed hyperparameter tuning with Randomized Search Cross validation as well. The best hyperparameters found through the randomized search were Euclidean metric, 4 n_neighbors, and uniform weights, which resulted in a cross-validation F1 score of approximately 0.91. Then, we use the best hyperparameters found to instantiate a new KNN Classifier and make predictions on the validation and testing set.

```
Validation Set Metrics:
Accuracy: 0.7488707276952401
Precision: 0.1890529348217501
Recall: 0.35496957403651114
F1-score: 0.24671052631578946

Confusion Matrix for validation dataset:
[[27106  6756]
 [ 2862  1575]]
              precision    recall  f1-score   support

           0       0.90      0.80      0.85     33862
           1       0.19      0.35      0.25      4437

    accuracy                           0.75     38299
   macro avg       0.55      0.58      0.55     38299
weighted avg       0.82      0.75      0.78     38299
```

```
Testing Set Metrics:
Accuracy: 0.7483550913838121
Precision: 0.18733252131546893
Recall: 0.34147424511545293
F1-score: 0.24193802107912538

Confusion Matrix:
[[27124  6672]
 [ 2966  1538]]
              precision    recall  f1-score   support

           0       0.90      0.80      0.85     33796
           1       0.19      0.34      0.24      4504

    accuracy                           0.75     38300
   macro avg       0.54      0.57      0.55     38300
weighted avg       0.82      0.75      0.78     38300
```

Based on the result after tuning, accuracy improved in both sets. Even though the recall and f1-score in class 0 ('non-default') and precision in class 1 ('default') are increased for both datasets, its overall performance shows the decrease of recall and F1-score in class 1 ('default') and precision in class 0 ('non-default') for both datasets as well. This indicates that the default hyperparameters for the **model before tuning contribute to better performance**. So, we can conclude that the model before tuning might be more suitable to perform model comparison.

4.3.6 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm commonly used in machine learning for binary classification and regression tasks by categorizing elements of a dataset into two distinct groups (Tabsharani, 2023).

Support Vector Machine (SVM) Model is instantiated with default kernel – Radial Basis Function (RBF) kernel and trained the model on the training data. Predictions are made on the validation set and testing set using the trained model and calculate the performance metrics and confusion matrix. The classification reports for validation set and training set are shown below.

```
Validation Set Metrics:
Accuracy: 0.7618475678216141
Precision: 0.23820701989716075
Recall: 0.48027946810908273
F1-score: 0.31846372263319134

Confusion Matrix for validation dataset:
[[27047  6815]
 [ 2306  2131]]
              precision    recall  f1-score   support

           0       0.92      0.80      0.86     33862
           1       0.24      0.48      0.32      4437

    accuracy                           0.76     38299
   macro avg       0.58      0.64      0.59     38299
weighted avg       0.84      0.76      0.79     38299
```

```
Testing Set Metrics:
Accuracy: 0.7613838120104439
Precision: 0.24224224224224225
Recall: 0.4835701598579041
F1-score: 0.3227862171174509

Confusion Matrix:
[[26983  6813]
 [ 2326  2178]]
              precision    recall  f1-score   support

           0       0.92      0.80      0.86     33796
           1       0.24      0.48      0.32      4504

    accuracy                           0.76     38300
   macro avg       0.58      0.64      0.59     38300
weighted avg       0.84      0.76      0.79     38300
```

Based on the classification report, the SVM model's performance improved after using SMOTE resampling technique. The accuracy is 0.76 on both sets. The precision, recall, and F1-score for the minority class improved significantly as before resampling three of them is 0.0. In this case, the precision for class 1 ('default') is 0.24 while for class 0 ('non-default') is 0.92 for both datasets. Similarly, for both datasets, the recall for class 1 ('default') is 0.48 and for class 0 ('non-default') is 0.80. Similar to f1-score, both datasets achieved 0.32 for class 1 ('default') and 0.86 for class 1 ('non-default'). This indicates that the resampled model is better at identifying default loans, which is a critical improvement for a loan default prediction model.

4.3.7 Neural Network

A neural network is a type of machine learning model inspired by the human brain which processes information by simulating the interactions between biological neurons to recognize patterns, assess choices, and make decisions (IBM, 2023a).

Neural Network Model is instantiated using MLPClassifier and trained the model on the training data. Predictions are made on the validation set and testing set using the trained model and calculate the performance metrics and confusion matrix. The classification reports for validation set and training set are shown below.

```
Validation Set Metrics:
Accuracy: 0.7403587561032925
Precision: 0.22228946041351488
Recall: 0.49673202614379086
F1-score: 0.3071348940914158

Confusion Matrix for validation dataset:
[[26151  7711]
 [ 2233  2204]]
              precision    recall  f1-score   support

           0       0.92      0.77      0.84     33862
           1       0.22      0.50      0.31      4437

    accuracy                           0.74     38299
   macro avg       0.57      0.63      0.57     38299
weighted avg       0.84      0.74      0.78     38299
```

```
Testing Set Metrics:
Accuracy: 0.7425848563968669
Precision: 0.22897054357728516
Recall: 0.5022202486678508
F1-score: 0.31453799624556766

Confusion Matrix:
[[26179  7617]
 [ 2242  2262]]
              precision    recall  f1-score   support

           0       0.92      0.77      0.84     33796
           1       0.23      0.50      0.31      4504

    accuracy                           0.74     38300
   macro avg       0.58      0.64      0.58     38300
weighted avg       0.84      0.74      0.78     38300
```

Based on the classification report, the model's accuracy is 0.74. The precision for both dataset in class 0 ('non-default') is 0.92, whereas for class 1 ('default') is 0.22 in validation set and 0.23 in testing set. For both datasets, the recall for class 0 ('non-default') is 0.77 while for class 1 ('default') is 0.50. Similarly, f1-score for both datasets is 0.84 for 'Non-default' class and 0.31 for 'Default' class.

```
Best Hyperparameters: {'solver': 'adam', 'learning_rate': 'adaptive', 'hidden_layer_sizes': (50, 100, 50), 'alpha': 0.0001, 'activation': 'relu'}
Best Cross-Validation F1 Score: 0.7988424624558816
```

After that, we performed hyperparameter tuning with Randomized Search Cross Validation as well. The best hyperparameters are 'adam' solver, an 'adaptive' learning rate schedule, a three-layer architecture for the hidden layers with 50 neurons in the first and third layers and 100 neurons in the second layer, an L2 penalty of 0.0001 and the 'relu' function. The best cross-validation F1 score is approximately 0.80. This indicates that the model trained with these settings perform well in terms of balancing precision and recall, which crucial for imbalanced classification tasks like loan default prediction.

```
Validation Set Metrics:
Accuracy: 0.7323428810151701
Precision: 0.20166256157635468
Recall: 0.44286680189317107
F1-score: 0.27713137296382484

Confusion Matrix for validation dataset:
[[26083  7779]
 [ 2472  1965]]
              precision    recall  f1-score   support

           0       0.91      0.77      0.84     33862
           1       0.20      0.44      0.28      4437

    accuracy                           0.73     38299
   macro avg       0.56      0.61      0.56     38299
weighted avg       0.83      0.73      0.77     38299
```

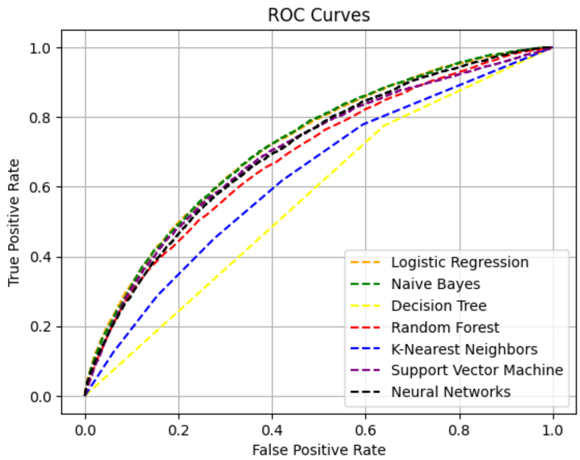
```
Testing Set Metrics:
Accuracy: 0.7365013054830287
Precision: 0.20950301517987108
Recall: 0.44738010657193605
F1-score: 0.28537034414388895

Confusion Matrix:
[[26193  7603]
 [ 2489  2015]]
              precision    recall  f1-score   support

           0       0.91      0.78      0.84     33796
           1       0.21      0.45      0.29      4504

    accuracy                           0.74     38300
   macro avg       0.56      0.61      0.56     38300
weighted avg       0.83      0.74      0.77     38300
```

Based on the result after tuning, the accuracy, precision, recall, and F1-score decrease on both validation and testing set. This indicates that the default hyperparameters for the model before tuning contribute to better performance. Thus, we will use the Neural Network Model before tuning to perform model comparison.

4.4 Comparison and Selection

```
AUC score:
Logistic Regression:  0.7211551489482292
Naive Bayes:  0.7212340625090002
Decision Tree:  0.5677800740289611
Random Forest:  0.686601967357378
K-Nearest Neighbors:  0.6269830579706428
Support Vector Machine:  0.7006866944799084
Neural Networks:  0.7037076247580563
```

AUC Score for 7 Models



ROC Curves for 7 Models

Based on the AUC scores and ROC curves, **Naïve Bayes has the highest AUC** (Area Under the Curve) score among all the seven models which is 0.72. ROC curve above also showed that the Naïve Bayes model performed better compared to other models. This indicates that Naïve Bayes has a **better trade-off between true positive rate and false positive rate**, which will maximize true positive rate while minimize false positive rate. Thus, Naïve Bayes may have better performance in classification of default and non-default loans.

Models	test - F1 score (0)	test - F1 score (1)	AUC Score
Naïve Bayes	0.81	0.32	0.72123
Logistic Regression	0.81	0.32	0.72116
Neural Networks	0.84	0.29	0.70371
Support Vector Machine	0.86	0.32	0.70069
Random Forest	0.77	0.3	0.68660
K-Nearest Neighbors	0.84	0.26	0.62698
Decision Tree	0.42	0.22	0.56778

Comparison for all models using F1-score and AUC score

The table arranges the models based on the descending order of AUC scores, with corresponding F1 scores evaluated on the testing dataset. Based on the table, Decision Tree model has lowest F1 score for both classes and AUC score. Naïve Bayes and Logistic Regression have similar performance, with a highest F1 score for class 1 ('default') and moderately high F1 score for class 0 ('non-default'). The AUC score also performs similar between these two models with Naïve Bayes achieves slightly higher AUC score than Logistic Regression. Even though F1 score for SVM is highest, but the recall is low (0.48) compared with Naïve Bayes and Logistic Regression (0.61). Consider from financial institution perspective, **minimizing false negatives** (incorrectly classifying actual defaults as non-defaults) **is critical to minimize the financial loss and risk**. As a result, we may prioritize models with higher recall for the default class, even if it comes at the cost of precision. Hence, we **focus on balanced F1 score for both classes and high AUC score** in order to select the best model.

Since Naïve Bayes achieves **relatively high F1 score** for both classes, this indicates a reasonable balance between precision and recall. Besides, Naïve Bayes has the **highest AUC score** among all models, suggesting a good discriminatory power in distinguishing between default and non-default loan applications. Therefore, we can conclude that **Naïve Bayes model is the best model**.

5. Interaction stage

After selecting the best model, we store the Naïve Bayes model using the Joblib library. This step is to ensure that we can retrieve and use the model without retrain it every time. After that, we load the model into the memory to store it for prediction on new data. Then, the new data which is the new applicants' information is evaluated by using the training model that we have loaded in memory. The new data undergoes similar data preprocessing before interacting with the trained model. This stage involves the separating features from the target variable ('Default' variable) and identifying the categorical and numerical variables. The encoding of categorical variables and standardization of numerical variables are carried out so that data representation is consistent with the trained model in order to get accurate prediction. After that, we combine all the dataset together for model to make prediction. Using the loaded model, we predict the result for the new dataset.

LoanID	Default
A01	0
A02	0
A03	1
A04	0
A05	0
A06	1
A07	1
A08	0
A09	0
A10	0
B01	0
B02	0
B03	0
B04	1
B05	0
B06	0
B07	1
B08	0
B09	1
B10	0

Prediction on Loan Default Status of 20 New Applicants

The prediction for the loan default status is shown as figure above. Out of 20 applicants, there are **14 applicants predicted as non-defaulters (class 0)** and **6 applicants are predicted to be defaulters (class 1)**. This means **14 of the non-defaulters will get loan approval** from bank while the **other 6 applicants will get rejected on their loan requests**. By identifying potential defaulters, our model can effectively mitigate financial risk for bank authorities. This proactive approach allows banks to allocate resources more efficiently and make informed decisions regarding loan approvals. Additionally, for loan applicants, the model increases the chances for non-defaulters to get approved for their loan requests. By accurately assessing risk, the model ensures that deserving applicants are more likely to receive loan approvals. Thus, utilizing this machine learning model on predicting the loan default status is useful to mitigate financial risk and facilitating the loan processes.

6. References

IBM (2023a). *What Are Neural Networks? | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/neural-networks.

IBM (2023b). *What is a Decision Tree | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/decision-trees.

IBM (2023c). *What is Random Forest? | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/random-forest.

IBM (2023d). *What is the k-nearest neighbors algorithm? | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/knn.

Patil, M. (2024). *The Roles of Data Sets in Machine Learning Projects: A Guide to Data Splitting*. [online] Medium. Available at: https://medium.com/@madhuri15/the-roles-of-data-sets-in-machine-learning-projects-a-guide-to-data-splitting-454beb468a49.

Satpathy, S. (2020). *SMOTE | Overcoming Class Imbalance Problem Using SMOTE*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/#:~:text=SMOTE%20is%20an%20oversampling%20technique.

Shetty, B. (2023). *An in-depth guide to supervised machine learning classification*. [online] Built In. Available at: https://builtin.com/data-science/supervised-machine-learning-classification.

Tabsharani, F. (2023). *What is a support vector machine? | Definition from WhatIs*. [online] WhatIs.com. Available at: https://www.techtarget.com/whatis/definition/support-vector-machine-SVM#:~:text=A%20support%20vector%20machine%20(SVM)%20is%20a%20type%20of%20supervised.

Wolff, R. (2020). *Classification Algorithms in Machine Learning: How They Work*. [online] MonkeyLearn Blog. Available at: https://monkeylearn.com/blog/classification-algorithms/.