Loan Defaulter Prediction by kNN Classifier using Orange

Gloria Aprilia (D21230035)

Master of Management Program

School of Business and Management

Petra Christian University

This report is submitted for the final exam.

Table of Contents

Chapter 1: Introduction

1.1 Background

There are several income streams for commercial banks, including taking interest from loans such as mortgages, auto-loans, business loans, and personal loans (Kagan, 2023). One of the reasons a commercial bank can run into loss is when the borrowers fail to pay their loans. This defaulter behavior can negatively affect both the bank and the borrowers.

From the bank's point of view, the existence of loan defaulters can bring the following disadvantages:

- Disturbing the cash flow of the bank in a way the account payable is more than the account receivable
- Paying more for reminder system including reminder program and debt collector
- Paying more for administration system to handle documents for failed loan payment

In the end, the bank cannot get the money they need to return the interest rate of money the people put on them, putting their financial status in a critical condition.

From the borrower's point of view, being a defaulter can also bring them negative effects such as their account being marked as delinquent, reducing their credit score, and the borrowers can be sued in court (Chen, 2023).

The phenomenon of loan defaulters needs to be eliminated for the bank's and borrower's good. One of the solutions can be predicting the defaulters and preventing them from taking loans.

1.2 Problem

Loan defaulters can drive losses to a commercial bank. To prevent these losses, a bank needs to predict a defaulter before they actually delay their payment. However, it is hard to predict a defaulter because there is a lot of data to be considered.

1.3 Objective

The objective of this project is to develop a machine learning model that can predict a loan defaulter based on their behavior in certain aspects, so the bank can do more analysis before they approve the loan of the predicted defaulter.

Chapter 2: Literature Study

2.1 Imbalanced Data Handling

Imbalanced data happens when the data is not equally distributed among classes (Boyle, 2019). Thus, there are two categories of class, the minority classes that have a very low portion and the majority class that dominates the dataset. This situation happens frequently on classification problem datasets.

There are three degrees of imbalanced data (Google, 2023). The first is extreme degree, where the minority class is less than 1% of the total population. The second is the moderate degree, where the minority class ranges between 1% and 20% of the total population. Then the mild degree is where the minority class ranges between 20–40% of the total population.

There are several strategies to handle imbalanced data including resampling, weighted loss function, ensemble methods, anomaly detection, data augmentation, and cost-sensitive learning. Among those strategies, one important technical strategy is data resampling. There are two types of data resampling, which is undersampling and oversampling.

2.1.1 Undersampling

Undersampling is a technique used in handling imbalanced datasets in binary classification problems. In undersampling, the population of the majority class is reduced by randomly removing rows, bringing the class distribution closer to balance. This helps prevent the model from being biased towards the majority class.

The RandomUnderSampler library is a popular method in handling imbalanced data by undersampling. In this method, instances from the majority class are randomly selected and removed until a more balanced distribution is achieved. This is a straightforward approach, but it may lead to a loss of information, especially if the dataset is small to begin with.

2.1.1 Oversampling

Oversampling is a method to handle imbalanced data by randomly replicating the minority data until the dataset reach the equivalent number between majority and minority data. This helps machine learning models avoid being biased towards the majority class and improves their ability to generalize to the minority class.

Synthetic Minority Over-sampling Technique (SMOTE) is an oversampling method that gain its popularity because of its sophistication, rather than RandomOverSampler. SMOTE is an oversampling method that creates synthetic instances for the minority class. Instead of simply duplicating existing instances, SMOTE generates synthetic examples by interpolating between existing minority class instances. This helps in capturing the underlying patterns of the minority class more effectively.

2.2 Machine Learning Algorithms

A machine learning algorithm is a set of rules and statistical models that enable a computer system to learn patterns from data and make predictions or decisions without being explicitly programmed. Machine learning algorithms can be broadly categorized into two main types: classification and regression.

In classification problem, the goal of machine learning algorithm is to categorize input data into predefined classes or categories. Classification algorithms work by learning patterns from labeled training data. The training data consists of input features and corresponding class labels. The algorithm learns the relationship between the input features and the classes during the training phase. Once trained, the algorithm can take new, unseen data and predict the class label based on the learned patterns. Common classification algorithms include logistic regression, k-nearest neighbors (KNN), decision trees, random forests, and support vector machines (SVM).

2.2.1 Logistic Regression

Logistic regression is a type of statistical model used for binary classification problems. The logistic regression model uses the logistic function (also called the sigmoid function) to model the relationship between the input features and the probability of the target variable being in the positive class. The logistic regression model is trained using a method called Maximum Likelihood Estimation (MLE). The objective is to find the set of weights that maximizes the likelihood of the observed data given the model.

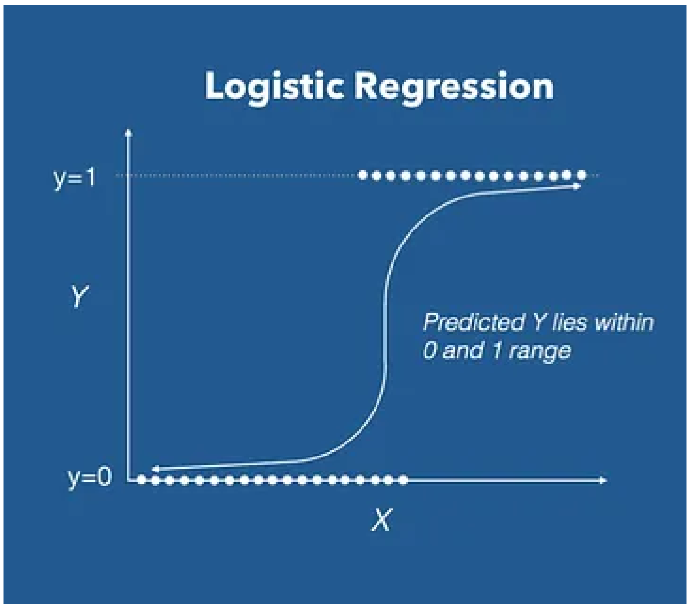
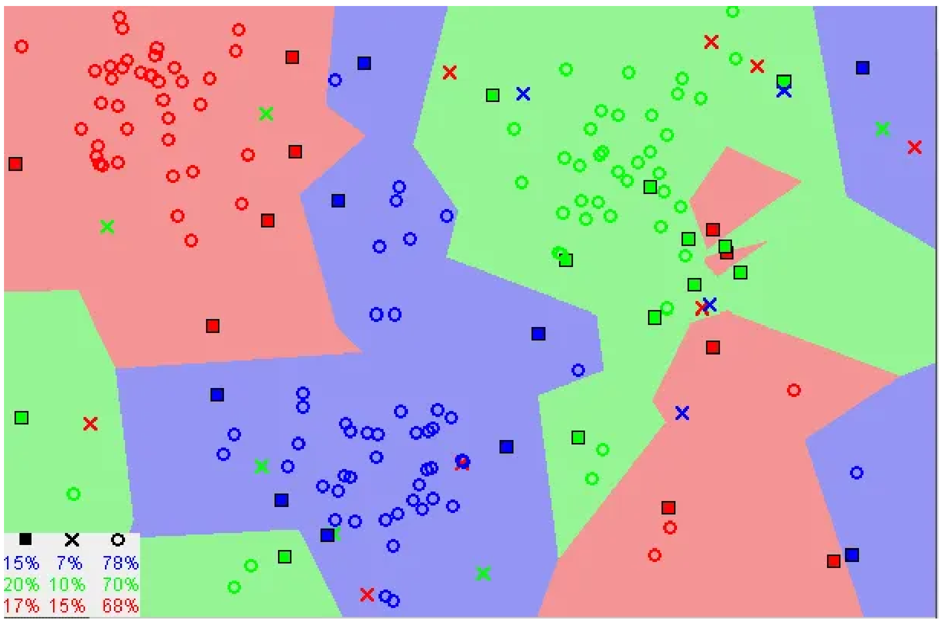


Figure 2.2.1.1 Logistic Regression Curve (Pant, 2019)

2.2.2 K-Nearest Neighbor (kNN) Classifier

K-Nearest Neighbors (KNN) is a simple and widely used classification algorithm in machine learning. During the training phase, the KNN algorithm stores all the available data points and their corresponding class labels. When a new, unseen instance needs to be classified, the algorithm calculates the distance between this instance and all the data points in the training set.

Common distance metrics include Euclidean distance, Manhattan distance, or other similarity measures. The algorithm selects the k data points (neighbors) from the training set that are closest to the new instance in terms of distance. The value of k is a user-defined parameter. For classification, a majority voting scheme is used among the k neighbors to determine the class of the new instance. The class that occurs most frequently among the neighbors is assigned to the new instance. The predicted class is assigned to the new instance based on the majority vote.



2.2.2.1 Data Neighborhood Area (Harrison, 2018)

2.2.3 Decision Tree

A Decision Tree Classifier is a popular machine learning algorithm that makes decisions by recursively partitioning the input space into regions, assigning a specific class label to each region. The process starts at the root of the tree, where the algorithm selects the most informative feature to split the data based on certain criteria. This split creates two child nodes representing subsets of the data. The algorithm then repeats the process for each child node, recursively expanding the tree until a stopping criterion is met, such as reaching a predefined depth or having a minimum number of samples in a node. At each decision node, the tree evaluates a feature's value for the input data, guiding it down the appropriate branch until it reaches a leaf node, where the final class label is assigned.

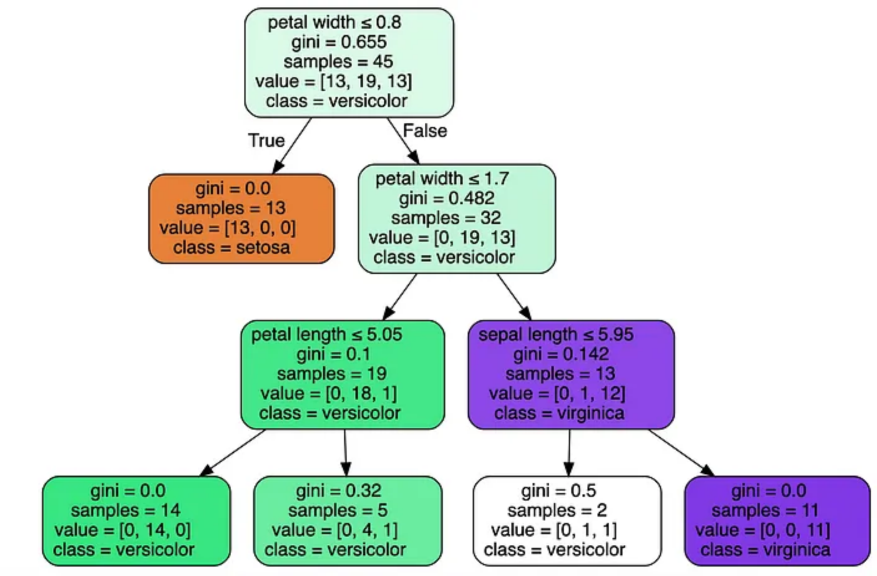


Figure 2.2.3.1 Decision Tree of Iris Dataset (Almog, 2018)

2.2.4 Random Forest

Random Forest is an ensemble learning algorithm that combines multiple decision trees to enhance predictive accuracy and generalization. It constructs a forest by training numerous decision trees on different subsets of the training data and considering random subsets of features at each split. The randomness introduced during training helps each tree capture diverse patterns in the data, reducing overfitting. During prediction, the Random Forest aggregates the output of individual trees through a majority vote (for classification) or averaging (for regression), producing a robust and accurate prediction. The ensemble nature of Random Forests makes them resilient to noise and outliers, and they are particularly effective in handling high-dimensional datasets and providing insights into feature importance.

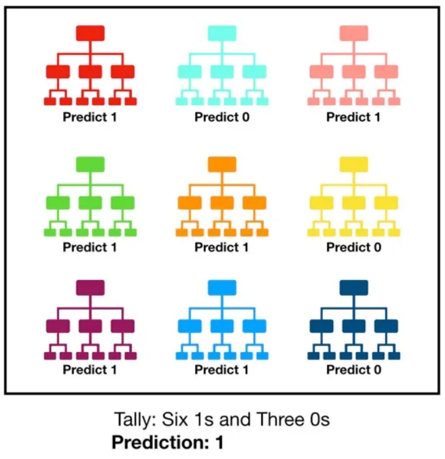
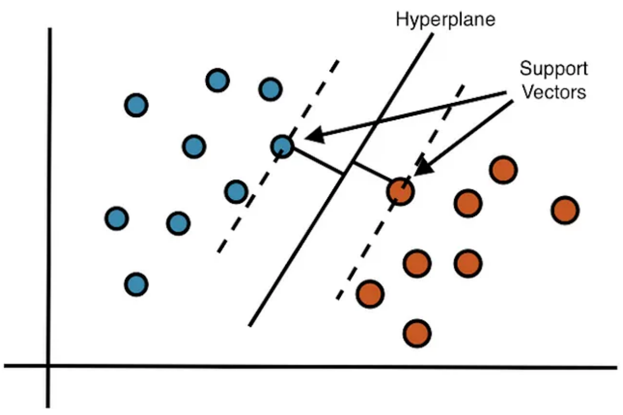


Figure 2.2.4.1 Decision Tree inside Random Forest (Yiu, 2019)

2.2.5 Support Vector Machine (SVM)

The primary objective of SVM in classification is to find the hyperplane that best separates the data into distinct classes with the maximum margin. The "margin" refers to the distance between the hyperplane and the nearest data points (support vectors) from each class. SVM aims to maximize this margin while minimizing classification errors. If a linear separation is not possible in the input space, SVM can use a kernel trick to map the data into a higher-dimensional space, allowing for a nonlinear decision boundary. SVM is particularly effective in scenarios with complex decision boundaries and can handle high-dimensional data efficiently. The optimization process involves finding the optimal weights and bias terms that define the hyperplane, providing a robust and accurate classification model.



2.2.5.1 Hyperplane in SVM (Vatsal, 2021)

2.3 Machine Learning Evaluation Matrix

2.3.1 AUC-ROC

The Area Under the Receiver Operating Characteristic (ROC) Curve is a metric used to evaluate the performance of a machine learning classification model. The ROC curve is a graphical representation of the model's ability to discriminate between positive and negative classes across different threshold values. The AUC-ROC quantifies the area under this curve, providing a single numerical value to assess the model's overall discriminatory power.

The ROC curve is created by plotting the True Positive Rate (TPR or Sensitivity) on the y-axis against the False Positive Rate (FPR) on the x-axis. TPR represents the proportion of true positive instances correctly classified as positive, while FPR represents the proportion of true negative instances incorrectly classified as positive. AUC-ROC is robust to class imbalance, making it a suitable metric for scenarios where one class significantly outnumbers the other.

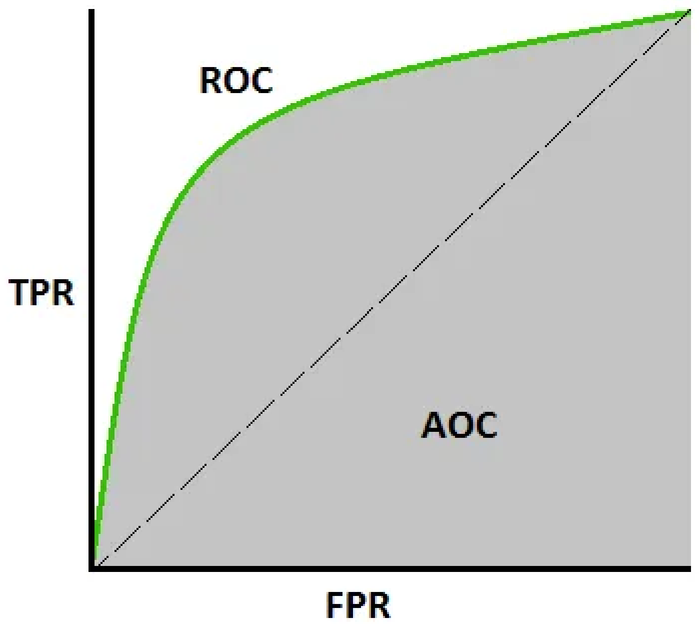


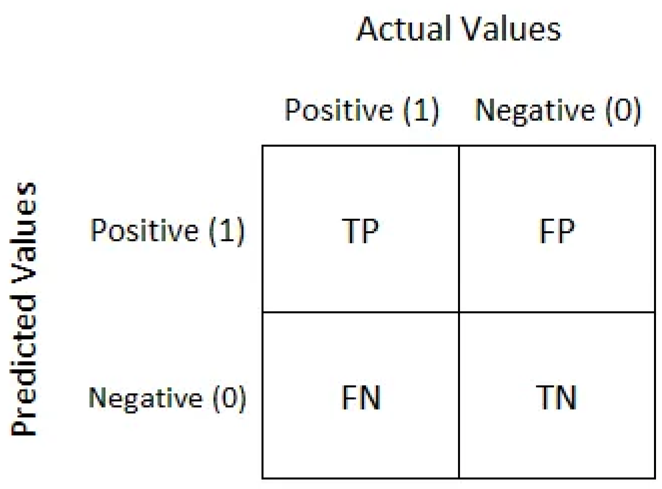
Figure 2.3.1.1 ROC Curve (Narkhede, 2018)

AUC-ROC calculates the area under the ROC curve. An AUC of 1.0 indicates perfect discrimination, while an AUC of 0.5 suggests that the model performs no better than random chance. Higher AUC values generally indicate better model performance. If the AUC is close to 1.0, the model has good discriminatory power across a range of thresholds. If the AUC is close to 0.5, the model's performance is comparable to random chance.

2.3.2 Confusion Matrix

A confusion Matrix is a fundamental matrix used to evaluate the performance of a classification machine learning model. It provides a summary of the predicted and actual class labels for a set of instances, allowing for a detailed analysis of the model's behavior. The confusion matrix consists of four essential metrics:

- True Positive (TP), instances that are correctly predicted as positive.
- True Negative (TN), instances that are correctly predicted as negative.
- False Positive (FP), instances that are incorrectly predicted as positive
- False Negative (FN), instances that are incorrectly predicted as negative



2.3.2.1 Confusion Matrix

From these metrics, several performance metrics can be derived, including accuracy, precision, recall, and F1 score. Each of those derived metrics are described in the following table:

Table 2.3.2.1 Metrics Derived from Confusion Matrix

No.	Derived Metrics	Formula
1	Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
2	Precision (Positive Predictive Value)	$\frac{TP}{TP + FP}$
3	Recall (Sensitivity or TP Rate)	$\frac{TP}{TP + FN}$
4	F1	$2\left[\frac{Precision \times Recall}{Precision + Recall}\right]$

Chapter 3: Methodology

3.1 Data

Data for this analysis is taken from Kaggle, under the title of Bank Loan Defaulter Prediction (Hackathon). This data originated from the Deloitte Touche Tohmatsu Limited Mini Hackathon. Data can be accessed online at the following link:
https://www.kaggle.com/datasets/ankitkalauni/bank-loan-defaulter-prediction-hackathon/data

The source provided some information on each column of the dataset. Those descriptions can be seen in the following table.

Table 3.1.1 Dataset Column Name and Description

No	Column Name	Description by Kaggle
1	ID	Unique ID of representative
2	Loan Amount	Loan amount applied
3	Funded Amount	Loan amount funded
4	Funded Amount Investor	Loan amount approved by the investors
5	Term	Term of loan (in months)
6	Batch Enrolled	Batch numbers to representatives
7	Interest Rate	Interest rate (%) on loan
8	Grade	Grade by the bank
9	Subgrade	Sub-grade by the bank
10	Employment Duration	Duration
11	Home Ownership	Owner ship of home
12	Verification Status	Income verification by the bank
13	Payment Plan	If any payment plan has started against loan
14	Loan Title	Loan title provided
15	Debit to Income	Ratio of representative's total monthly debt repayment divided by self reported monthly income excluding mortgage

16	Delinquency - two years	Number of 30+ days delinquency in past 2 years
17	Inquires - six months	Total number of inquiries in last 6 months
18	Open Account	Number of open credit line in representative's credit line
19	Public Record	Number of derogatory public records
20	Revolving Balance	Total credit revolving balance
21	Revolving Utilities	Amount of credit a representative is using relative to revolving_balance
22	Total Account	Total number of credit lines available in representatives credit line
23	Initial List Status	Unique listing status of the loan - W(Waiting), F(Forwarded)
24	Total Received Interest	Total interest received till date
25	Total Received Late Fee	Total late fee received till date
26	Recoveries	Post charge off gross recovery
27	Collection Recovery Fee	Post charge off collection fee
28	Collection 12 months medical	Total collections in last 12 months excluding medical collections
29	Application Type	Indicates when the representative is an individual or joint
30	Last Week Pay	Indicates how long (in weeks) a representative has paid EMI after batch enrolled
31	Accounts Delinquent	Number of accounts on which the representative is delinquent
32	Total Collection Amount	Total current balance from all accounts
33	Total Current Balance	Total current balance from all accounts
34	Total Revolving Credit Limit	Total revolving credit limit
35	Loan Status	1 = Defaulter, 0 = Non Defaulters

3.2 Data Preparation

The purpose of data preparation is to prepare the data before the next process, which is Exploratory Data Analysis and Machine Learning. The data preparation consists of two steps, data cleaning and data encoding. The process of data preparation is done using Python for Data Analysis with Jupyter Notebook as the code editor.

3.2.1 Data Cleansing

In the data cleansing, it is ensured that the dataset does not have missing values and unclear columns. When it is checked, there is no missing value on this data.

Unclear columns are the columns in the dataset that do not have a description. This data may cause bias in the process, so we need to drop these unclear columns. The remaining columns after this process are listed in the following table:

Table 3.2.1.1 Remaining Columns after Cleansing

Loan Amount	Total Delinquency
Funded Amount	Total Inquires
Investor Amount	Open Account
Payment Term	Total Bad Attitudes
Rate	Revolving Balance
Grade	Total Account
Subgrade	Initial Status
Home Type	Received Interest
Home Worth	Received Late Fee
Verification	Application Type
Debt to Income Ratio	Account Delinquent
Loan Status	

After cleaning unnecessary columns, the next data cleaning process is to ensure the datatype and simplifying the column name. Then, the dataset is ready for the data encoding process.

3.2.2 Data Encoding

Data encoding is purposed to change categorical data into numerical data that the processor can read. This data encoding is needed to support imbalanced data handling, which requires the data to be numerical.

To do data encoding, categorical data must be separated and classified first. The classification of categorical data can be seen in the following table:

Table 3.2.2.1 Categorical Data and Its Encoding

No	Column Name	Real Data	Encoded
1	grade (A is more trusted, G is less trusted)	A - G	7-1
2	subgrade (A1 is more trusted, G5 is less trusted)	A1 - A5 B1 - B5 C1 - C5 D1 - D5 E1 - E5 F1 - F5 G1 - G5	35 - 31 30 - 26 25 - 21 20 - 16 15 - 11 10 - 6 5 - 1
3	verification	Verified Not-Verified	1 0
4	home_type	OWN MORTGAGE RENT	2 1 0
5	application_type	JOINT INDIVIDUAL	1 0

3.3 Data Pre-Processing

Before training the data, this project did 2 data pre-processing steps, which are data test-train splitting and imbalance data handling.

In the data test-train splitting, this project divided the data by a ratio of 20:80, meaning 20% of the data is used for testing, and the remaining 80% is used for training. Besides the manual data splitting, this project also employs data cross-validation by 10 folds for ML training.

For the imbalance data handling, this project uses the imblearn library to do Random Under Sampler (RUS) and Synthetic Minority Over-sampling Technique (SMOTE).

Table 4.3.2.1 Result of Imbalanced Data Handling

Technique	# of Defaulter	# of Non-Defaulter	Total Data
Random Under Sampler (RUS)	4,993	4,993	9,986
Synthetic Minority Over-sampling Technique (SMOTE)	72,107	72,107	144,214

3.4 Machine Learning

To find the best machine learning model, this project tried the 5 most popular machine learning models for classification:

1. Logistic Regression
2. KNN Classifier
3. Decision Tree
4. Random Forest
5. Support Vector Machine (SVM)

After training the data, the model will be evaluated through the AUC and confusion matrix. Then, the model will be tested again using the testing dataset (entirely different from the training dataset). The Orange workflow is presented in the following image.

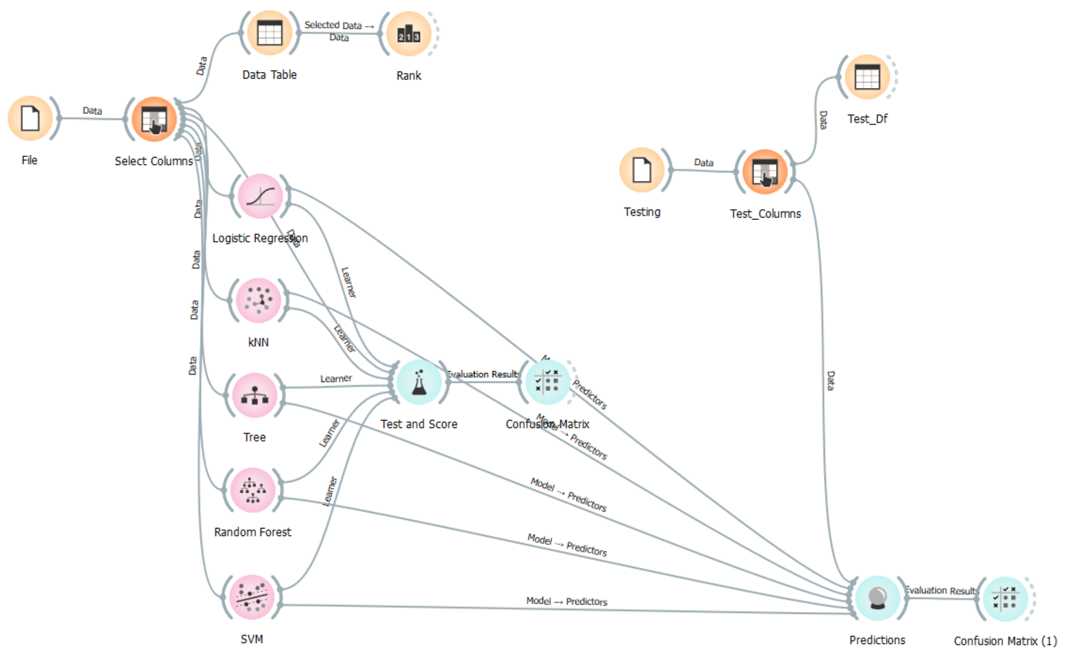


Figure 3.3.1 Orange Workflow

The idea is to train the undersampled and oversampled data to get the best AUC, and then we will choose the best machine-learning model. After that, we will do tuning and deploy the model.

Chapter 4: Result & Analysis

4.1 Exploratory Data Analysis (EDA)

In order to see the correlation between each column with the loan_status as the target column, we can use .corr method in Python.

Table 4.1.1 General correlation using Python

No	Column	Correlation Value
1	pmt_term	0.047019
2	received_late_fee	0.011345
3	debt_income_ratio	0.008053
4	total_delinquency	0.007695
5	received_interest	0.006856
6	home_worth	0.004035
7	funded_amt	0.002576
8	verification	0.002530
9	total_bad_attitude	0.001900
10	open_account	0.001827
11	application_type	0.001157
12	loan_amt	0.000511
13	home_type	0.000440
14	investor_amt	-0.001439
15	account_delinquent	-0.004486
16	rate	-0.004776
17	subgrade	-0.005179
18	grade	-0.009803
19	total_account	-0.015751
20	total_inquires	-0.018720

21	revolving_balance	-0.020543

Based on the correlation analysis, we can take columns with highest absolute correlation values to be processed further in the machine learning. Those variables are listed in the following table.

Table 4.1.2 Columns Used for ML Process

No	Columns	Correlation Value	Correlation
1	pmt_term	0.047019	Directly proportional
2	revolving_balance	0.020543	Inversely proportional
3	total_inquires	0.018720	Inversely proportional
4	total_account	0.015751	Inversely proportional
5	grade	0.009803	Inversely proportional
6	debt_income_ratio	0.008053	Directly proportional
7	total_delinquency	0.007695	Directly proportional
8	subgrade	0.005179	Inversely proportional
9	home_worth	0.004035	Directly proportional
10	application_type	0.001157	Directly proportional
11	loan_status	-	Target column

Payment term has the highest correlation to the target column, which is 0.047. It is directly proportional to the loan status, meaning that the higher the payment term, the borrowers are more likely to be defaulters. In the crosstab below, it shows that borrowers who take 59 months of payment are more likely to be defaulters. It is 7.48% of defaulters there, meaning in every 100 people who take 59 months loan, around 8 people are defaulters.

Table 4.1.3 Crosstab of Payment Term and Loan Status

Payment Term	Non-Defaulter	Defaulter
36 months	97.78%	2.22%

58 months	93.62%	6.38%
59 months	92.52%	7.48%
60 months	100.00%	0.00%

Longer payment terms mean that funds are tied up for an extended period before they are received. This delay in cash inflow can lead to difficulties in meeting immediate financial obligations, increasing the risk of default. Also, Those with longer payment terms may be managing higher levels of debt, either due to larger loan amounts or multiple financial commitments. High debt levels can strain financial resources, making it challenging to meet payment deadlines and increasing the likelihood of default.

A revolving balance is the portion of credit card spending that remains unpaid at the end of a billing cycle. In this dataset, borrowers with higher revolving balances are more likely to be defaulters. Using the mean value of the defaulter and non-defaulter class, it can be seen that the non-defaulter has 8.86% less revolving balance. A high revolving balance can be a sign that the individual is carrying a substantial amount of debt relative to their credit capacity. High levels of debt can strain financial resources and make it challenging for individuals to meet their repayment obligations, increasing the risk of default.

Table 4.1.4 Statistical Description of The Revolving Amount in Each Class.

Aspect	Non-Defaulter	Defaulter
Min	1.00	0.00
Mean	7673.00	8419.09
Max	116933.00	188657.00

4.2 Machine Learning (ML)

4.2.1 Undersampled Data

The performance of machine learning models that are trained using undersampled data is listed on the following table:

Table 4.2.1.1 Performance Matrix of ML Models using Undersampled Data

No	ML Model	AUC	CA	F1	Prec.	Recall
1	Logistic Regression	0.530	0.520	0.516	0.520	0.520
2	kNN Classifier	0.514	0.506	0.506	0.506	0.506
3	Decision Tree	0.524	0.513	0.511	0.513	0.513
4	Random Forest	0.548	0.528	0.519	0.530	0.528
5	SVM	0.505	0.502	0.497	0.502	0.502

From this performance matrix, if it is seen from the value of area under the curve (AUC) of receiver operating characteristic (ROC) curve, Random Forest model has the best performance among all models. However, with the AUC ranging from 0.5-0.6, this model cannot be considered a reliable model.

To analyze this model more, the following table presents the confusion matrix of each model:

Table 4.2.1.2 Confusion Matrix of Each Machine Learning Model using Undersampled Data

No	ML Model	TP	TN	FP	FN	Total
1	Logistic Regression	3012	2179	2814	1981	9986
2	kNN Classifier	2558	2492	2501	2435	9986
3	Decision Tree	2267	2857	2136	2726	9986
4	Random Forest	3326	1946	3047	1667	9986

5	SVM	2024	2987	2006	2969	9986

In the confusion matrix, the performance of the model can be seen from the true positive (TP) and true negative (TN) value. Based on this judgment, Random Forest still being the best model for predicting the loan defaulter.

4.2.2 Oversampled Data

The performance of machine learning models that are trained using oversampled data with cross-validation is listed in the following table:

Table 4.2.2.1 Performance Matrix of ML Models using Oversampled Data with Cross-Validation

No	ML Model	AUC	CA	F1	Prec.	Recall
1	Logistic Regression	0.533	0.523	0.520	0.524	0.523
2	kNN Classifier	0.843	0.772	0.770	0.780	0.772
3	Decision Tree	0.771	0.765	0.765	0.765	0.765
4	Random Forest	0.706	0.644	0.624	0.682	0.644
5	SVM	0.385	0.585	0.584	0.586	0.585

From this performance matrix, there are 2 best models based on their AUC, which is the Decision Tree and the kNN Classifier. Both of these models can also classified as more reliable models, due to the ROC range of 0.7-0.9. To determine which one is better, we can use the following confusion matrix:

Table 4.2.1.2 Confusion Matrix of Each Machine Learning Model using Undersampled Data with Cross-Validation

No	ML Model	TP	TN	FP	FN	Total
1	Logistic Regression	43184	32267	39840	28923	144214

2	kNN Classifier	61865	49470	22637	10242	144214
3	Decision Tree	55154	55126	16981	16953	144214
4	Random Forest	62897	29971	42136	9210	144214
5	SVM	46243	38099	34008	25864	144214

Based on the total value of true positive and true negative, the kNN Classifier has best prediction.

On the other hand, if machine learning is done by dividing the dataset into 80:20 ratios, the performance matrix is slightly different from the one with cross-validation.

Table 4.2.2.3 Performance Matrix of ML Models using Oversampled Data with Random Sampling 80:20

No	ML Model	AUC	CA	F1	Prec.	Recall
1	Logistic Regression	0.548	0.531	0.530	0.531	0.531
2	kNN Classifier	0.838	0.767	0.765	0.775	0.767
3	Decision Tree	0.764	0.757	0.757	0.757	0.757
4	Random Forest	0.708	0.647	0.631	0.679	0.647
5	SVM	0.372	0.594	0.591	0.596	0.594

The performance matrix of cross-validated and random-sampled data shows that cross-validated data has higher performance in AUC and Recall. Therefore, for the tuning process, this project uses the KNN Classifier with cross-validated data.

4.2.3 KNN Classifier Tuning

The kNN Classifier can be tuned to increase the performance metrics. The tune settings for the kNN are listed in the following table.

Table 4.2.3.1 KNN Classifier Tune Setting

No	Variable	Tune Setting
1	Number of Folds	20
2	Number of Neighors	11
3	Metric	Manhattan
4	Weighting	By Distance

By tuning the kNN Classifier using the setting listed in table 4.2.3.1, the classifier can reach AUC of 0.889. The improvement on the other metrics can be seen in the following table.

Table 4.2.3.2 KNN Classifier Performance Matrix Comparision: Regular and Tuned

No	ML Model	AUC	CA	F1	Prec.	Recall
1	kNN Classifier (regular)	0.843	0.772	0.770	0.780	0.772
2	kNN Classifier (tuned)	0.889	0.790	0.787	0.809	0.790

Chapter 5: Conclusion

To reach the optimum performance of the kNN Classifier to predict defaulters, we need to train oversampled data using SMOTE and tune the kNN Classifier as mentioned in chapter 3.4.3. When doing so, we can get an AUC of 0.889 which is considered reliable.

References

Almog, U. (2018, November 14). Decision Trees, Explained. *Towards Data Science.* https://towardsdatascience.com/decision-trees-explained-d7678c43a59e

Boyle, T. (2023, February 4). Dealing with Imbalanced Data. *Towards Data Science.* https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18

Chen, J. (2023, October 29). Default: What It Means, What Happens When You Default, Examples. *Investopedia.* https://www.investopedia.com/terms/d/default2.asp.

Google (2023, June 9). Imbalanced Data. *Google Machine Learning Fundamental Course.* https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

Harrison, O. (2018, September 11). Machine Learning Basics with the K-Nearest Neighbors Algorithm. *Towards Data Science.* https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Kagan, J. (2023, May 25). How Do Commercial Banks Work, and Why Do They Matter. *Investopedia.* https://www.investopedia.com/terms/c/commercialbank.asp#:~:text=Commercial%20banks%20make%20money%20by,capital%20to%20make%20these%20loans.

Narkhede, S. (2018, June 26). Understanding AUC - ROC Curve. *Towards Data Science.* https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Pant, A. (2019, January 22). Introduction to Logistic Regression. *Towards Data Science.*

https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Vatsal (2021, February 6). Support Vector Machine (SVM) Explained. *Towards Data Science*. https://towardsdatascience.com/support-vector-machine-svm-explained-58e59708cae3

Yiu, T. (2019, June 12). Understanding Random Forest. *Towards Data Science*. https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Attachments

Data Cleansing (Jupyter Notebook) Link:
https://github.com/gloria-aprilia/Final-Project-BDA-MM/blob/master/01_Data_Cleansing.ipynb

Exploratory Data Analysis (Jupyter Notebook) Link:
https://github.com/gloria-aprilia/Final-Project-BDA-MM/blob/master/02_EDA.ipynb

Machine Learning (Orange Workflow) Link:
https://github.com/gloria-aprilia/Final-Project-BDA-MM/blob/master/D21230035_Gloria_Ellysian_Aprilia.ows