ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

MACHINE LEARNING ALGORITHM IN CREDIT SCORING TO PREVENT BAD DEBT IN COOPERATIVES

TAOFIK HIDAJAT*

Management Department, Sekolah Tinggi Ilmu Ekonomi Bank BPD Jawa Tengah, Semarang, Indonesia.
*Corresponding Author Email: inidotcom@yahoo.com

ANDY ISMAIL

Management Department, Darwan Ali University, Sampit, Indonesia.

Abstract

This research proposes a credit score model for cooperatives using machine learning. Until now, there is no standard credit score assessment in savings and loan cooperatives in Indonesia. There are still many savings and loan cooperatives that provide loans due to closeness to the management or manager of the cooperative. The purpose of this research is to obtain a credit scoring method through machine learning that is effective, efficient and high accuracy. To predict the chance of default, this research uses seven machine learning algorithms namely Logistic Regression Classifier, Support Vector Machine Classifier, K-Neighbors Classifier, Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, and Light Gradient Boosting Machine Classifier. The data taken from the loan data of 851 members of Bank BPD Jateng "Yakekar" Cooperative, Semarang, Indonesia. The results show that Logistic Regression, Support Vector Machine Classifier, and K-Neighbors Classifier are the models that have relatively better performance in identifying 'Current' collectibility. However, all models have difficulty in classifying other collectibility ('Bad' and 'Doubtful') with low precision and recall.

Keywords: Machine Learning; Credit Scoring; Cooperatives.

1. INTRODUCTION

Google and Temasek report (2019) states that the Indonesian population is 23% banked, 26% underbanked and 51% unbanked. Underbanked and unbanked are groups with low financial inclusion or unserved in financial services. Banks will only provide access to bankable groups that have a credit history through credit scoring. Thus, those with no credit history will not get access to loans because there is no data available for scoring (Niu et al., 2019).

This unserved group will then seek other financial access such as peer-to-peer lending (Hidajat, 2019) and cooperatives. Since potential customers or borrowers do not have a credit history, the credit risk in these institutions is greater. For P2P, which is usually owned by fintech companies with large capital, the utilization of credit score to handle default risk in assessing prospective borrowers has undergone tremendous development.

Currently, fintech is using big data technology to help process prospective borrowers (Campbell-Verduyn et al., 2017). However, cooperatives as the pillar of the Indonesian economy have not progressed in credit scoring. In fact, credit scoring requires detailed profile data of prospective borrowers so that lending is more accurate.

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

Cooperatives also have to maintain liquidity and solvency levels (da Silva Filho, 2002) so the profile of co-operative members should consist of people who do not have bad records. Thus, it is important to explore borrower characteristics that can minimise the potential for default in savings and loan cooperatives. The challenge that arises in credit scoring and the problem in this research is how to find potential borrowers with an effective and efficient process that has high accuracy.

In the financial industry, credit scoring is widely used to measure credit risk (Vasconcellos et al., 2019). Initially, a widely used technique was logistic regression. However, this method has weaknesses when variables exhibit complex nonlinear relationships (Bahnsen et al., 2016). Nonetheless, logistic regression models have strong advantages in terms of interpretability and variable stability so that they can be applied to the prediction of borrower default behaviour (Wang et al., 2020).

Credit scoring has evolved by utilising artificial intelligence (Thomas, 2000) such as machine learning and producing more accurate and efficient models (Dastile et al., 2020). Machine learning is an artificial intelligence application that uses statistical techniques to generate models from a set of data. Some examples of the implementation of machine learning methods in research include decision trees, AdaBoost, support vector machines (K. et al., 2020), neural networks (Malhotra & Malhotra, 2003), and k-nearest neighbors (West, 2000). The advantages of machine learning include being able to ignore assumptions in logistic regression and produce better classification (Vasconcellos et al., 2019).

Some uses of machine learning for credit scoring include the use of Random Forest, AdaBoost, and LightGBM to predict the chances of default in peer-to-peer lending using social network information in China (Niu et al., 2019), random forest to assess the effectiveness of the credit union lending process in Brazil (Vasconcellos et al., 2019), Naive Bayesian Model, Logistic Regression Analysis, Random Forest, Decision Tree, and K-Nearest Neighbor Classifier for bank loans in China (Wang et al., 2020).

This research proposes a credit score model for cooperatives using machine learning. Until now, there has been no standard credit score assessment in savings and loan cooperatives. There are still many savings and loan cooperatives that provide loans due to closeness to the management or manager of the cooperative (Sucipto, 2015). This research is important because cooperatives in Indonesia are currently not supervised by the Financial Services Authority but only through the cooperative supervisor. In addition, unlike financial service institutions such as banks and peer-to-peer lending, the development of credit scores, especially through machine learning, in cooperatives has received less attention. A reliable credit score will also minimise the risk of default and make cooperatives a more reliable and healthy institution.

2. METHODOLOGY

To predict the probability of default, this research uses seven machine learning algorithms namely Logistic Regression Classifier, Support Vector Machine Classifier, K-Neighbors Classifier, Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, and Light

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

Gradient Boosting Machine Classifier. The results of the seven methods will then be compared to see if there is a difference and if so, which method is best used to predict the chance of default.

The data source is taken from the database of BPD Jateng Bank Consumer Cooperative "Yakekar", Semarang, Indonesia. This cooperative has 851 members who are employees and retired employees of Bank Jateng.

3. RESULTS

3.1. Logistic Regression Classifier

Logistic Regression Classifier is a machine learning algorithm for predicting the probability of a target class or discrete dependent variable based on multiple independent variables or predictor variables that are continuous or discrete. In logistic regression, a logistic or sigmoid function is used to transform the independent variable into a value that lies between 0 and 1, which can be interpreted as the probability of the relevant target class. Then, these probabilities are compared with a certain threshold value, and a decision is made about the target class corresponding to these probabilities.

The Classification Report and AUC-ROC Score of the Logistic Regression Classifier in this study provided the following results.

(1) Class -1 and 0. These two classes have a precision and recall of around 1.00 for the positive class (class 1), which means that all predictions given for these two classes are correct with respect to class 1. However, the recall for class -1 and 0 to their own class (self-recall) is very low (0.00), which indicates that the model is almost unable to identify samples from these classes. This condition can be problematic, especially if these classes have important business or analytical significance.

(2) Class 1. This class has a precision of around 0.98 and a recall of around 1.00 against itself, which indicates that the model is very good at classifying class 1 samples. The F1-score is also high (0.99), indicating very good performance.

(3) Overall Accuracy. The overall accuracy of the model is 0.98. However, accuracy can be biased in cases of class imbalance like this, where the majority of samples are from class 1.

3.2. AUC-ROC Score

The AUC-ROC Score (Area Under the Receiver Operating Characteristic Curve) is 0.625. AUC-ROC measures the ability of the model to distinguish between positive and negative classes. The AUC-ROC score of 0.625 indicates that the model performs reasonably well in this regard, but there is still room for improvement.

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

The conclusions for the Logistic Regression Classifier are as follows:

(1) The Logistic Regression Classifier performs very well in identifying class 1, with high precision and recall and F1-score close to 1. This indicates that the model is effective in classifying class 1 samples. However, the model does not seem to be effective in identifying classes -1 and 0, as seen from the very low precision, recall, and F1-score for these classes.

(2) High overall accuracy may be misleading due to class imbalance. Therefore, the AUC-ROC Score provides a more holistic picture of the model's performance in distinguishing positive and negative classes. With an AUC-ROC Score of about 0.625, there is still room for improvement in the model's ability to distinguish the classes. It is worth considering strategies such as handling class imbalance or exploring other models if the -1 and 0 classes are also considered important in the analysis.

3.3. Support Vector Machine Classifier

Support Vector Machine (SVM) Classifier is a machine learning algorithm used to classify data by building a mathematical model that identifies the optimal decision boundary to distinguish two or more classes (binary classification). The SVM model works by finding the line that separates the two classes with the maximum distance (margins) so that the model is more general in classifying new data that has never been seen before. Classification Report Support Vector Machine Classifier in this study provides the following results.

(1) Class -1 and 0. These two classes have very low precision, recall, and F1-score (0.00), indicating that the SVM model is barely able to identify or predict samples from these two classes. This can be a serious problem if these classes have important business or analytical significance.

(2) Class 1. This class has a precision of about 0.98, a recall of about 1.00, and an F1-score of about 0.99, indicating that the SVM model is very good at classifying class 1 samples. This is a positive aspect of the model's performance.

(3) Overall Accuracy. The overall accuracy of the model was 0.98. However, it is important to note that accuracy can be biased in cases of class imbalance such as this, where the majority of samples are from class 1.

The conclusion for Support Vector Machine (SVM) is as follows.

(1) The SVM model performed very well in identifying class 1, with high precision, recall, and F1-score. This indicates that the model is effective in classifying class 1 samples. However, the model does not seem to be effective in identifying classes -1 and 0, which is evident from the very low precision, recall, and F1-score for these classes. This is a negative aspect of the model's performance.

(2) High overall accuracy may be misleading due to class imbalance. Therefore, it is necessary to consider strategies such as handling class imbalance or exploration of other models if the -1 and 0 classes are also important in the analysis.

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

(3) Overall, the performance of the SVM model in this context depends on the importance of certain classes in the analysis. If class 1 is the main focus and classes -1 and 0 are minority or less important, then the SVM model may be suitable. However, if all classes are important, further consideration needs to be given to improving the performance of the model for the minority classes.

3.4. K-Neighbours Classifier

K-Neighbors Classifier (KNN) is a machine learning algorithm used to classify data based on the majority category of the nearest neighbours (majority vote) of the given data. KNN works by storing all training data as a representation of the dataset and finding the majority category of the k nearest neighbours of the new data. The number k is predetermined and usually an odd number to ensure there is no draw in the majority voting process. KNN can also be used to perform regression by calculating the average value of the k nearest neighbours.

Classification Report K-Neighbors Classifier in this study provides the following results.

(1) Class -1 and 0. These two classes have a precision and recall of around 0.00, which shows that the K-Neighbors Classifier model is almost unable to identify or predict samples from these two classes. This indicates very poor performance in classifying samples from classes -1 and 0.

(2) Class 1. This class has a precision of around 0.98, a recall of around 0.99, and an F1-score of around 0.99, which indicates that the K-Neighbors Classifier model is very good at classifying class 1 samples. This is a positive aspect of the model's performance.

(3) Overall Accuracy. The overall accuracy of the model is 0.97. However, it is important to remember that accuracy can be biased in cases of class imbalance like this, where the majority of samples are from class 1.

The conclusion for the K-Neighbors Classifier is as follows.

(1) The K-Neighbors Classifier model has excellent performance in identifying class 1, with high precision, recall and F1-score. This indicates that the model is effective in classifying class 1 samples. However, the model does not seem to be effective in identifying classes -1 and 0, which can be seen from the very low precision and recall for these classes. This is a negative aspect of the model's performance.

(2) High overall accuracy may be misleading due to class imbalance. Therefore, it is necessary to consider strategies such as handling class imbalance or exploring other models if classes -1 and 0 are also important in the analysis.

(3) Overall, the performance of the K-Neighbors Classifier model in this context depends on the importance of certain classes in the analysis. If class 1 is the main focus and classes -1 and 0 are a minority or less important, then the K-Neighbors Classifier model may be suitable. However, if all classes are important, further consideration is needed in improving the model's performance against minority classes.

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

3.5. Decision Tree Classifier.

Decision Tree Classifier is a machine learning algorithm that is used to classify data by building a decision tree structure that can describe the relationship between several input and output variables. The Decision Tree Classifier model works by dividing the dataset into smaller subsets based on rules discovered from the input variables.

At each level, the algorithm will choose the best input variables (the ones with the most influence) to divide the dataset into increasingly smaller subsets. This process is carried out recursively until there are no more subsets that can be divided or the specified tree depth limit is reached.

The Classification Report results for the Decision Tree Classifier model are as follows.

(1) Class -1 and 0. These two classes have a precision and recall of around 0.00, which shows that the Decision Tree Classifier model is almost unable to identify or predict samples from these two classes. This indicates very poor performance in classifying samples from classes -1 and 0.

(2) Class 1. This class has a precision of around 0.98, a recall of around 0.99, and an F1-score of around 0.98, which indicates that the Decision Tree Classifier model is very good at classifying class 1 samples. This is a positive aspect of the model's performance.

(3) Overall Accuracy. The overall accuracy of the model is 0.96. However, it is important to remember that accuracy can be biased in cases of class imbalance like this, where the majority of samples are from class 1.

The conclusion for the K-Neighbors Classifier is as follows.

(1) The Decision Tree Classifier model has excellent performance in identifying class 1, with high precision, recall and F1-score. This indicates that the model is effective in classifying class 1 samples. However, the model does not seem to be effective in identifying classes -1 and 0, which can be seen from the very low precision and recall for these classes. This is a negative aspect of the model's performance.

(2) High overall accuracy may be misleading due to class imbalance. Therefore, it is necessary to consider strategies such as handling class imbalance or exploring other models if classes -1 and 0 are also important in the analysis.

(3) Overall, the performance of a Decision Tree Classifier model in this context depends on the importance of certain classes in your analysis. If class 1 is the main focus and classes -1 and 0 are a minority or less important, then a Decision Tree Classifier model may be suitable. However, if all classes are important, further consideration is needed in improving the model's performance against minority classes.

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

3.6. Random Forest Classifier

Random Forest Classifier is a machine learning algorithm which is a development of the Decision Tree Classifier and is used to classify data. The Random Forest Classifier model works by building many decision trees randomly on a subset of training data and conducting majority voting on the classification results provided by each tree.

The Classification Report results for the Random Forest Classifier model are as follows.

(1) Class -1 and 0. These two classes have a precision and recall of around 0.00, which shows that the Random Forest Classifier model is almost unable to identify or predict samples from these two classes. This indicates very poor performance in classifying samples from classes -1 and 0.

(2) Class 1. This class has a precision of around 0.98, a recall of around 0.99, and an F1-score of around 0.98, which indicates that the Random Forest Classifier model is very good at classifying class 1 samples. This is a positive aspect of the model's performance.

(3) Overall, the overall accuracy of the model is 0.96. However, it is important to remember that accuracy can be biased in cases of class imbalance like this, where the majority of samples are from class 1.

The conclusion for the Random Forest Classifier is as follows.

(1) The Random Forest Classifier model has excellent performance in identifying class 1, with high precision, recall and F1-score. This indicates that the model is effective in classifying class 1 samples. However, the model does not seem to be effective in identifying classes -1 and 0, which can be seen from the very low precision and recall for these classes. This is a negative aspect of the model's performance.

(2) High overall accuracy may be misleading due to class imbalance. Therefore, it is necessary to consider strategies such as handling class imbalance or exploring other models if classes -1 and 0 are also important in the analysis.

(3) Overall, the performance of a Random Forest Classifier model in this context depends on the importance of certain classes in your analysis. If class 1 is the main focus and classes -1 and 0 are a minority or less important, then a Random Forest Classifier model may be suitable. However, if all classes are important, further consideration is needed in improving the model's performance against minority classes.

3.7. XGBoost Classifier

XGBoost Classifier (Extreme Gradient Boosting Classifier) is a machine learning algorithm that is used to classify data using ensemble learning (combining models). XGBoost is a development of the Gradient Boosting algorithm which has better performance in overcoming overfitting. The XGBoost Classifier model works by building a number of decision trees in stages, with each tree adjusting the prediction error of the previous tree. At each iteration, the algorithm adjusts the weight of each sample to minimize the given loss function.

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

The results of the Classification Report for the XGBoost Classifier model provided are as follows:

(1) Class 0. This class has a precision of around 0.00, a recall of around 0.00, and an F1-score of around 0.00, which indicates that the XGBoost Classifier model is barely able to identify or predict samples from class 0. This indicates very poor performance in classifying samples from class 0.

(2) Class 1. This class has a precision of around 0.98, a recall of around 0.99, and an F1-score of around 0.98, which indicates that the XGBoost Classifier model is very good at classifying class 1 samples. This is a positive aspect of the model's performance.

(3) Class 2. This class also has a precision and recall of around 0.00, and an F1-score of around 0.00, which indicates that the model is barely able to identify or predict samples from class 2.

(4) Overall, the model accuracy is 0.96. However, it is important to remember that accuracy can be biased in cases of class imbalance like this, where the majority of samples are from class 1.

The conclusion for XGBoost Classifier is as follows.

(1) The XGBoost Classifier model has excellent performance in identifying class 1, with high precision, recall, and F1-score. This indicates that the model is effective in classifying class 1 samples. However, the model does not seem to be effective in identifying classes 0 and 2, which can be seen from the very low precision, recall, and F1-score for these classes. This is a negative aspect of the model's performance.

(2) High overall accuracy may be misleading due to class imbalance. Therefore, it is necessary to consider strategies such as handling class imbalance or exploring other models if classes 0 and 2 are also important in the analysis.

(3) Overall, the performance of the XGBoost Classifier model in this context depends on the importance of certain classes in the analysis. If class 1 is the main focus and classes 0 and 2 are a minority or less important, then the XGBoost Classifier model may be suitable. However, if all classes are important, further consideration is needed in improving the model's performance against minority classes.

3.8 Light Gradient Boosting Machine Classifier

Light Gradient Boosting Machine (LightGBM) Classifier is a machine learning algorithm that is used to classify data using ensemble learning (combining models). LightGBM was developed by Microsoft and designed to speed up the process of training models on large datasets. The LightGBM Classifier model works by building a number of decision trees in stages, with each tree adjusting the prediction error of the previous tree. At each iteration, the algorithm uses gradient-based One-Side Sampling (OSS) and Exclusive Feature Bundling (EFB) techniques to speed up model training.

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

The Classification Report results for the XGBoost Classifier model are as follows.

(1) Class -1 and 0. These two classes have a precision and recall of around 0.00, which shows that the XGBoost Classifier model is almost unable to identify or predict samples from these two classes. This indicates very poor performance in classifying samples from classes -1 and 0.

(2) Class 1. This class has a precision of around 0.98, a recall of around 0.99, and an F1-score of around 0.98, which indicates that the XGBoost Classifier model is very good at classifying class 1 samples. This is a positive aspect of the model's performance.

(3) Overall Accuracy. The overall accuracy of the model is 0.96. However, accuracy can be biased in cases of class imbalance like this, where the majority of samples are from class 1.

The conclusion for the Light Gradient Boosting Machine Classifier is as follows.

(1) The XGBoost Classifier model has excellent performance in identifying class 1, with high precision, recall, and F1-score. This indicates that the model is effective in classifying class 1 samples. However, the model does not seem to be effective in identifying classes -1 and 0, which can be seen from the very low precision and recall for these classes. This is a negative aspect of the model's performance.

(2) High overall accuracy may be misleading due to class imbalance. Therefore, it is necessary to consider strategies such as handling class imbalance or exploring other models if classes -1 and 0 are also important in the analysis.

(3) Overall, the performance of the XGBoost Classifier model in this context depends on the importance of certain classes in the analysis. If class 1 is your main focus and classes -1 and 0 are a minority or less important, then the XGBoost Classifier model may be suitable. However, if all classes are important, further consideration is needed in improving the model's performance against minority classes.

From the calculation of seven classification models based on Collectibility categorization ('Current', 'Congestion', 'Doubtful'), there are several main evaluation metrics, namely precision, recall, and f1-score, as well as general accuracy. The following are the results of the analysis and conclusions of the seven models.

Logistic Regression Classifier and Support Vector Machine (SVM) Classifier are able to classify Collectibility 'Current' with very high precision, recall and f1-score (0.98, 1.00, 0.99) as well as an accuracy of 0.98, but for the classification 'Congestion' and 'Doubtful' has low precision and recall, even close to 0. Thus, this model is good at identifying 'Current' Collectibility but is not effective in classifying other models.

The K-Neighbors Classifier method is able to classify 'Current' Collectibility with high precision, recall and f1-score (0.98, 0.99, 0.99), as well as an accuracy of 0.97. This method is effective in identifying 'Current' Collectibility and has better results than Logistic Regression and SVM, but provides low precision and recall, even close to 0 for 'Loss' and 'Doubtful' classifications. The Decision Tree Classifier method provides poor results compared to the Logistic Regression Classifier, Support Vector Machine (SVM) Classifier and K-Neighbors Classifier. This method is able to classify Collectibility 'Current' with high precision, recall and f1-score (0.98, 0.99, 0.98) with an accuracy of 0.96. For the Collectibility classification 'Loss' and 'Doubtful', this method provides low precision and recall.

Random Forest Classifier, XGBoost Classifier, and Light Gradient Boosting Machine Classifier provide similar results to Decision Tree Classifier. These three methods give poor results in classifying Collectibility other than 'Loss' and 'Doubtful'.

4. CONCLUSION

The aim of this research is to understand the extent to which each model is able to classify collectibility well. The calculation results show that Logistic Regression, SVM, and K-Neighbors Classifier are models that have relatively better performance in identifying 'Current' Collectibility. However, all models have difficulty in classifying Other Collectibility ('Loss' and 'Doubtful'), with low precision and recall.

Thus, in this case, no model is consistently good at classifying all Collectibility categories. Selection of the best model must consider business priorities and acceptable error rates in each Collectibility category.

Declaration of conflicting interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research was funded by Ministry of Education, Culture, Research, and Technology, Indonesia, contract number 004/SKP/P3M/STIEBPD/VI/2023.

Data availability

The data that support the findings of this study are openly available at dx.doi.org/10.6084/m9.figshare.24486157

References

1) Bahnsen, A. C., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, *51*, 134–142.

2) Campbell-Verduyn, M., Goguen, M., & Porter, T. (2017). Big Data and algorithmic governance: the case of financial practices. *New Political Economy*, *22*(2), 219–236.

3) da Silva Filho, G. T. (2002). Avaliação de desempenho em cooperativas de crédito: Uma aplicação do modelo de gestão econômico -GECON. *Organizações Rurais & Agroindustriais*, *4*(1).

4) Dastile, X., Celik, T., & Potsane, M. (2020). Statistical and machine learning models in credit scoring: A systematic literature survey. *Applied Soft Computing*, *91*, 106263. https://doi.org/https://doi.org/10.1016/j.asoc.2020.106263

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

5) Google, Temasek, B. & C. (2019). *Introducing e-Conomy SEA 2019*.

6) Hidajat, T. (2019). Unethical practices peer-to-peer lending in Indonesia. *Journal of Financial Crime*, *27*(1), 274–282. https://doi.org/10.1108/JFC-02-2019-0028

7) K., M. D., Kunal, M., & Rashmi, M. (2020). Evaluating Consumer Loans Using Machine Learning Techniques. In K. D. Lawrence & D. R. Pai (Eds.), *Applications of Management Science* (Vol. 20, pp. 59–69). Emerald Publishing Limited. https://doi.org/10.1108/S0276-897620200000020004

8) Malhotra, R., & Malhotra, D. K. (2003). Evaluating consumer loans using neural networks. *Omega*, *31*(2), 83–96.

9) Niu, B., Ren, J., & Li, X. (2019). Credit Scoring Using Machine Learning by Combing Social Network Information: Evidence from Peer-to-Peer Lending. *Information*, *10*(12), 397.

10) Sucipto, A. (2015). Prediksi Kredit Macet Melalui Perilaku Nasabah Pada Koperasi Simpan Pinjam Dengan Menggunakan Metode Algoritma Klasifikasi C4. 5. *Jurnal DISPROTEK*, *6*(1).

11) Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, *16*(2), 149–172.

12) Vasconcellos, de P. D. A., Rinaldo, A., Fabio, A., & Fonseca, M. A. M. A. (2019). Estimating credit and profit scoring of a Brazilian credit union with logistic regression and machine-learning techniques. *RAUSP Management Journal*, *54*(3), 321–336. https://doi.org/10.1108/RAUSP-03-2018-0003

13) Wang, Y., Zhang, Y., Lu, Y., & Yu, X. (2020). A Comparative Assessment of Credit Risk Model Based on Machine Learning ——a case study of bank loan data. *Procedia Computer Science*, *174*, 141–149. https://doi.org/https://doi.org/10.1016/j.procs.2020.06.069

14) West, D. (2000). Neural network credit scoring models. *Computers & Operations Research*, *27*(11–12), 1131–1152.

Appendix

1. Logistic Regression Classifier

```
### Logistic Regression Classifier
# Import library
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('111.csv')

# Separate features (X) and targets (y)
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']
```

ISSN 1533-9211
DOI: 10.5281/zenodo.10695859

```
# Divide the dataset into training data and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature normalization using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the Logistic Regression Classifier model
logistic_regression = LogisticRegression(random_state=42)

# Training the model
logistic_regression.fit(X_train, y_train)

# Predicting
y_pred = logistic_regression.predict(X_test)

# Displays classification reports
#classification_rep = classification_report(y_test, y_pred)
#print("Classification Report:\n", classification_rep)
classification_rep = classification_report(y_test, y_pred, zero_division=1)
print("Classification Report:\n", classification_rep)
Classification Report:
            Precision    recall  f1-score   support
        -1      1.00      0.00      0.00         2
         0      1.00      0.00      0.00         2
         1      0.98      1.00      0.99       167
    Accuracy                        0.98       171
   macro avg      0.99      0.33      0.33       171
weighted avg      0.98      0.98      0.97       171
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score, roc_curve, auc
import matplotlib.pyplot as plt

# Load dataset
data = pd.read_csv('111.csv')

# Separate features (X) and targets (y)
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']

# Divide the dataset into training data and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature normalization using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize model
logistic_regression = LogisticRegression(random_state=42)

# Training the model
logistic_regression.fit(X_train, y_train)

# Predicting
y_pred = logistic_regression.predict(X_test)

# Displays classification reports
classification_rep = classification_report(y_test, y_pred, zero_division=1)
print("Classification Report:\n", classification_rep)

# Calculating the predicted probability for the positive class (class 1)
y_pred_proba = logistic_regression.predict_proba(X_test)

# Calculating AUC-ROC Score
#roc_auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr', average='weighted')
try:
    roc_auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr', average='weighted')
except ValueError:
    roc_auc = 1.0  # Menganggap AUC-ROC sebagai 1 jika terjadi peringatan

# Displaying AUC-ROC Score
print("AUC-ROC Score (OvR - Weighted):", roc_auc)

# Get the ROC curve for each class
fpr = {}
tpr = {}
roc_auc_class = {}
```

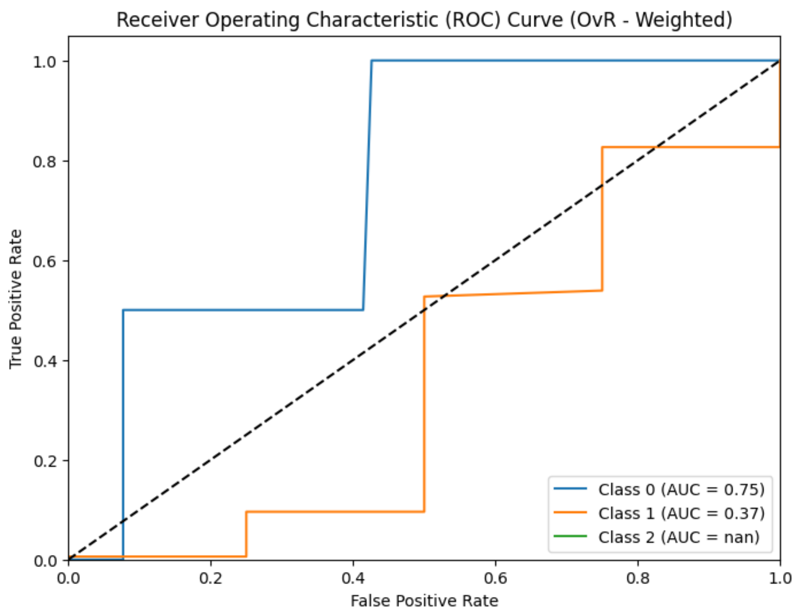
ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
for i in range(len(logistic_regression.classes_)):
   fpr[i], tpr[i], _ = roc_curve(y_test, y_pred_proba[:, i], pos_label=i)
   roc_auc_class[i] = auc(fpr[i], tpr[i])
# Displays the ROC curve for each class
plt.figure(figsize=(8, 6))
for i in range(len(logistic_regression.classes_)):
   plt.plot(fpr[i], tpr[i], label=f'Class {i} (AUC = {roc_auc_class[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--')  # Random lines
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve (OvR - Weighted)')
plt.legend(loc='lower right')
plt.show()
Classification Report:
            precision    recall  f1-score   support

        -1       1.00      0.00      0.00         2
         0       1.00      0.00      0.00         2
         1       0.98      1.00      0.99       167

  accuracy                           0.98       171
 macro avg       0.99      0.33      0.33       171
weighted avg     0.98      0.98      0.97       171

AUC-ROC Score (OvR - Weighted): 0.6249870237724489
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_ranking.py:1029:
UndefinedMetricWarning: No positive samples in y_true, true positive value should be
meaningless
  warnings.warn(
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859



2. Support Vector Machine Classifier

```
## Support Vector Machine Classifier

# Import library
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('111.csv')

# Separate features (X) and targets (y)
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']

# Divide the dataset into training data and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature normalization using StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
X_test = scaler.transform(X_test)

# Inisialisasi model Support Vector Machine (SVM) Classifier
svm_classifier = SVC(kernel='linear', random_state=42)

# Training the model
svm_classifier.fit(X_train, y_train)

# Predicting
y_pred = svm_classifier.predict(X_test)

# Displays classification reports
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00         2
           0       0.00      0.00      0.00         2
           1       0.98      1.00      0.99       167

    accuracy                           0.98       171
   macro avg       0.33      0.33      0.33       171
weighted avg       0.95      0.98      0.97       171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

3.K-Neighbors Classifier

```
# K-Neighbors Classifier
# Import library
import pandas as pd
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('111.csv')

# Separate features (X) and targets (y)
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']

# Divide the dataset into training data and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature normalization using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize model
knn = KNeighborsClassifier(n_neighbors=5)

# Training the model
knn.fit(X_train, y_train)

# Predicting
y_pred = knn.predict(X_test)

# Displays classification reports
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
Classification Report:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00         2
           0       0.00      0.00      0.00         2
           1       0.98      0.99      0.99       167
    accuracy                           0.97       171
   macro avg       0.33      0.33      0.33       171
weighted avg       0.95      0.97      0.96       171
```

The Seybold REPORT
ISSN 1533-9211
DOI: 10.5281/zenodo.10695859

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

1. Decision Tree Classifier

```
# Decision Tree Classifier
# Import library
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('111.csv')

# Separate features (X) and targets (y)
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']

# Divide the dataset into training data and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize model
decision_tree = DecisionTreeClassifier(random_state=42)

# Training the model
decision_tree.fit(X_train, y_train)

# Predicting
y_pred = decision_tree.predict(X_test)

# Displays classification reports
```

The Seybold Report
ISSN 1533-9211
DOI: 10.5281/zenodo.10695859

```
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
Classification Report:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00         2
           0       0.00      0.00      0.00         2
           1       0.98      0.99      0.98       167

    accuracy                           0.96       171
   macro avg       0.33      0.33      0.33       171
weighted avg       0.95      0.96      0.96       171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2. Random Forest Classifier

```
# Random Forest Classifier
# Import library
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('111.csv')

# Separate features (X) and targets (y)
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']
```

The Seybold REPORT
ISSN 1533-9211
DOI: 10.5281/zenodo.10695859

```
# Divide the dataset into training data and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize model
random_forest = RandomForestClassifier(random_state=42)

# Training the model
random_forest.fit(X_train, y_train)

# Predicting
y_pred = random_forest.predict(X_test)

# Displays classification reports
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
Classification Report:
            precision    recall  f1-score   support

      -1       0.00      0.00      0.00         2
       0       0.00      0.00      0.00         2
       1       0.98      0.99      0.98       167
  accuracy                         0.96       171
 macro avg       0.33      0.33      0.33       171
weighted avg     0.95      0.96      0.96       171
```

3. XGBoost Classifier

```
# XGBoost Classifier
# Import library
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('XGBoostClassifier.csv')

# Separate features (X) and targets (y)
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']

# Divide the dataset into training data and testing data
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model XGBoost Classifier
xgb_classifier = XGBClassifier(random_state=42)

# Initialize model
xgb_classifier.fit(X_train, y_train)

# Predicting
y_pred = xgb_classifier.predict(X_test)

# Displays classification reports
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.98      0.99      0.98       167
           2       0.00      0.00      0.00         2

    accuracy                           0.96       171
   macro avg       0.33      0.33      0.33       171
weighted avg       0.95      0.96      0.96       171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

7. Light Gradient Boosting Machine Classifier

```
# Light Gradient Boosting Machine Classifier
# Import library
import pandas as pd
import numpy as np
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
from sklearn.model_selection import train_test_split
import lightgbm as lgb
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('111.csv')

# Separate features (X) and targets (y)
X = data[['SEX', 'TIME_PERIOD', 'LOAN_AMOUNT', 'DEBET_BALANCE',
'GUARANTEE']]
y = data['COLLECTIBILITY']

# Divide the dataset into training data and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert dataset into LightGBM Dataset format
train_data = lgb.Dataset(X_train, label=y_train)

# Parameters for the LightGBM model
params = {
    'objective': 'binary',
    'metric': 'binary_error',
    'boosting_type': 'gbdt',
    'num_leaves': 31,
    'learning_rate': 0.05,
    'feature_fraction': 0.9
}

# Initialize model
lgb_classifier = lgb.train(params, train_data, num_boost_round=100)

# Predicting
y_pred_prob = lgb_classifier.predict(X_test, num_iteration=lgb_classifier.best_iteration)
y_pred = [1 if pred > 0.5 else 0 for pred in y_pred_prob]

# Displays classification reports
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
[LightGBM] [Info] Number of positive: 660, number of negative: 20
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was
0.000050 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 313
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
[LightGBM] [Info] Number of data points in the train set: 680, number of used features: 5
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.970588 -> initscore=3.496508
[LightGBM] [Info] Start training from score 3.496508
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Classification Report:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00         2
           0       0.00      0.00      0.00         2
           1       0.98      0.99      0.98       167

    accuracy                           0.96       171
   macro avg       0.33      0.33      0.33       171
weighted avg       0.95      0.96      0.96       171
```

ISSN 1533-9211

DOI: 10.5281/zenodo.10695859

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
 _warn_prf(average, modifier, msg_start, len(result))
```