Vol: 05 Issue: 05

International Journal of Data Science and Advanced Analytics (ISSN: 2563-4429)

Credit Rating Prediction Using Different Machine Learning Techniques

Aiyegbeni Gifty[1], Yang Li[2], Joseph Annan[3], Funminiyi Adebayo[4]

[1, 2, 3, 4] University of East London, Department of Engineering and Computing, London, England, United Kingdom

aiyegbenigifty@gmail.com[1], Y.Li@uel.ac.uk[2], J.Annan@uel.ac.uk[3], N.Adebayo@uel.ac.uk[4]

Corresponding author email: aiyegbenigifty@gmail.com

Abstract — Credit rating prediction is a crucial task in the banking and financial industry. Financial firms want to identify the likelihood of customers repaying loans or credit. With the advent of machine learning algorithms and big data analytics, it is now possible to automate and improve the accuracy of credit rating prediction. In this research, we aim to develop a machine learning-based approach for customer credit rating prediction. Machine learning algorithms, including decision trees, random forests, support vector machines, and logistic regression, were evaluated and compared in terms of accuracy, precision, and AUC. Feature selection was also performed to analyze the importance of different features in predicting credit ratings. Findings suggested that status, duration, credit history, amount, savings, other debtors, property, and employment duration are the most important features in predicting credit ratings. Results showed that the support vector machine algorithm did best in predicting bad credits, achieving an accuracy of 79.7%, AUROC of 0.76, and a precision of 0.88. After optimization, an AUROC of 78% was obtained. This is a 78% accuracy for properly identifying bad credits. This research demonstrates the potential of machine learning algorithms for customer credit rating prediction and could have significant implications for the banking and financial industry by enabling more accurate and efficient credit rating predictions and reducing the risk of defaults and financial losses.

Keywords— *Credit rating; Credit default; Machine learning; Default prediction; Model optimization.*

1. Introduction

Credit rating is the assessment of an individual's or business's creditworthiness, which is the likelihood of repayment of a loan or credit [1]. Credit rating is usually determined by credit bureaus, which collect and analyze data related to an individual's or business's credit history, including payment history, credit utilization, length of credit history, and types of credit used [2].

Credit rating prediction is the process of using machine learning algorithms and statistical models to predict an individual's or business's credit rating based on their credit history and other relevant data [3]. Credit rating prediction is important for lenders and financial institutions as it helps them make informed decisions about lending money or extending credit to individuals or businesses. Accurately predicting credit ratings can minimize the risk of defaults and financial losses and help maintain a healthy lending portfolio.

The traditional approach to credit rating prediction relied on the subjective judgment of credit analysts, which was time-consuming and prone to errors. However, with the advent of machine learning algorithms and big data analytics, it is now possible to automate and improve the accuracy of credit rating prediction [4], [5]. Machine learning algorithms are particularly useful for tasks involving large datasets and complex patterns, like customer credit rating prediction [5]. Credit rating prediction is important for lenders and financial institutions as it helps them assess the possibility of defaults (bad credits) to reduce financial losses [6]. Machine learning algorithms and statistical models can be used to automate and improve the accuracy of credit rating prediction, which can have significant implications for the efficient functioning of the lending market [7].

In this research, the aim is to develop a machine learning-based approach for predicting credit defaults (bad credits). This was done by exploring various machine learning algorithms, including decision trees, random forests, support vector machines, and logistic regression, and evaluating their performance on a dataset of customer credit histories. The research question is: Which machine learning algorithm perform best in predicting bad credit ratings?

To answer this question, an analysis of the dataset and evaluation of the performance of each algorithm in terms of accuracy, precision, recall, and F1 score will be performed. Feature selection will also be done to analyze the importance of different features in predicting credit ratings. This research will contribute to existing literatures by providing a comparative evaluation of different machine-learning algorithms for credit rating prediction.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work in the field of credit rating prediction. Section 3 presents the machine learning algorithms, and describes the dataset and the pre-processing steps. Section 4 describes the model implementation. Section 5 discusses the results, Section 6 provides a discussion of findings and model optimization, and Section 7 concludes the paper and suggests future directions for research.

2. Literature Review

Several studies have been conducted in the area of customer credit rating prediction using machine learning algorithms. In this literature review, a survey of some studies that have used machine learning algorithms to predict customer credit ratings has been done.

One of the earliest studies in this field was by [2], who proposed a research project that utilizes Support Vector Machine (SVM) to classify and predict credit ratings in the Indian market. To maximize the usefulness of the model, it was integrated with compounding techniques and dimension reduction methods like Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). The evaluation of the model's performance demonstrated outstanding performance in terms of classification and prediction when combined with the aforementioned techniques, as compared to its performance alone.

Another study by [4] used machine learning algorithms, like decision trees, logistic regression, and support vector machines, to analyze credit data and predict the creditworthiness of borrowers. The study highlights the importance of accurate credit scoring in mitigating the risks associated with lending and improving the efficiency of credit decision-making processes.

[1] proposed a model based on Neural Network (NN) for predicting credit card payment defaulters using the Microsoft Azure Machine Learning Platform. This proposed model was compared with three machine learning algorithms built over Bayes Point Machine, Logistic Regression, and Decision Tree. The proposed model was found to be better in terms of prediction rate and other parameters.

In the research by [6], data mining techniques like logistic regression, decision trees, and random forest machine learning algorithms were utilized to precisely analyze customer data and differentiate between defaulters and non-defaulters. Among the classifiers, the random forest algorithm had the highest performance in terms of F1-score, accuracy, and AUROC with scores of 87%, 80%, and 69%, respectively.

In a more recent study, [8] applied a predictive model that employs a deep neural network in conjunction with a decision tree classifier to identify default and non-default customers in the finance industry. By utilizing the credit score dataset, the model achieves a good level of accuracy in predicting customer behavior, with an approximate 87% accuracy rate for identifying default and non-default customers. Similarly, [3] proposed a credit scoring model that utilizes Rough Set Theory (RST) and Multi-Layer Perceptron (MLP) Neural Network. To assess its effectiveness, the model was compared against six other machine learning models using both the Australian and German credit datasets. The results indicate that the proposed model achieves an accuracy rate of 90% and 87% for the Australian and German credit datasets, respectively.

Furthermore, several recent studies have explored the use of ensemble learning algorithms for credit rating prediction. For instance, [9] proposed to classify credit card applicants' credit status by analyzing their personal information and data submitted during the application process using machine learning algorithms. Logistic regression, decision trees, and neural network models were employed for this purpose. To improve accuracy, a group voting method was proposed to combine the predictions of the three models. By implementing this method, the accuracy of the ensemble algorithm was improved by 3-14% for the different single models. The resulting credit scores were then used to set borrowing limits and repayment rules for users.

The above studies demonstrate that machine learning algorithms can be used to accurately predict customer credit ratings. Random forest and SVM algorithms have also been shown to perform well in different contexts. Overall, machine learning algorithms have the potential to improve the efficiency and accuracy of credit rating prediction, which can have significant implications for the financial industry.

3. Methodology

3.1 Models

3.1.1 Decision Trees

Decision trees are a popular machine-learning technique for classification and regression tasks. They work by recursively splitting the data into subsets based on the most important features. The algorithm creates a tree-like model of decisions and their possible consequences, which is used for making predictions on new data.

The decision tree algorithm recursively splits the data by the feature that provides the highest information gain, until a stopping criterion is met (e.g. a maximum tree depth, a minimum number of instances per leaf, etc.). At each node of the tree, the algorithm selects the feature that best separates the data based on the information gained.

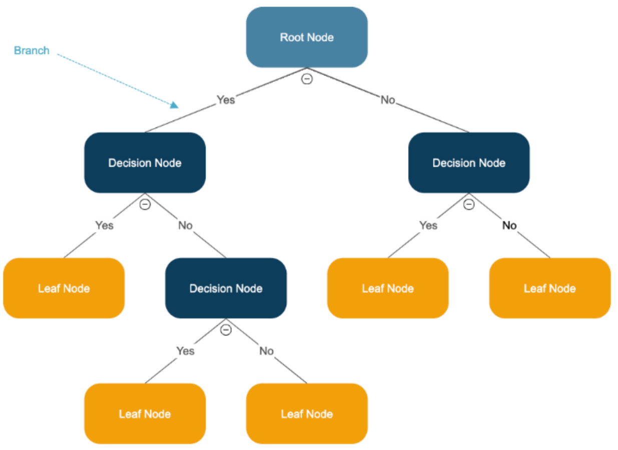


Fig 1 Diagrammatic representation of a decision tree model

3.1.2 Random Forest

Random Forest is a supervised machine learning algorithm. It is an ensemble method that combines multiple decision trees to create a more accurate and robust model.

The basic idea behind Random Forest is to build a large number of decision trees, each trained on a random subset of the data and a random subset of the features. Each tree is trained independently, using a different subset of the data and features, and the final prediction is the average (for regression) or the mode (for classification) of the predictions of all the trees. Random Forest can handle large datasets and noisy data and it is less prone to overfitting than a single decision tree, as the multiple trees reduce the variance of the model.

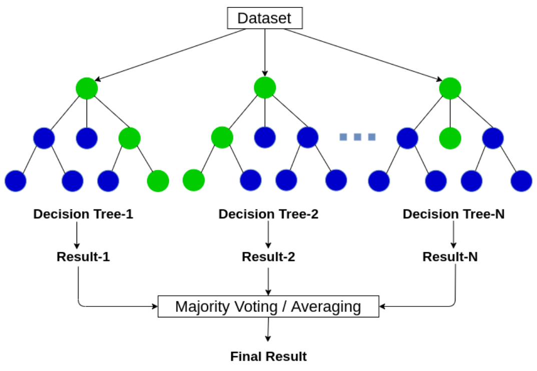


Fig 2 Diagrammatic representation of a random forest model

3.1.3 Support Vector Machines

SVMs are a powerful class of supervised machine-learning algorithms that can be used for both classification and regression tasks. SVMs are based on the idea of finding a hyperplane that separates the data into classes, such that the margin between the hyperplane and the closest points in each class is maximized.

In the case of linearly separable data, the hyperplane can be found by maximizing the margin between the two classes. However, if the data is not linearly separable, SVMs use the kernel trick to transform the data into a higher-dimensional space where it may be linearly separable.

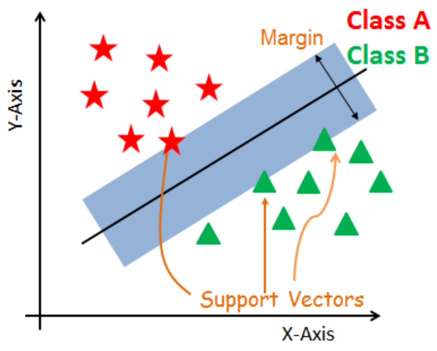


Fig 3 Diagrammatic representation of an SVM model

3.1.4 Logistic Regression

Logistic Regression is a type of supervised machine learning algorithm used for binary classification tasks. It models the probability of a binary response variable (i.e., a variable that takes on one of two possible outcomes) as a function of one or more predictor variables.

The output of logistic regression is a probability score between 0 and 1, which represents the likelihood of the binary outcome. Logistic regression assumes that the log-odds of the positive outcome is a linear function of the predictor variables. This relationship is modeled using the logistic function (also known as the sigmoid function), which maps any real-valued input to a value between 0 and 1.

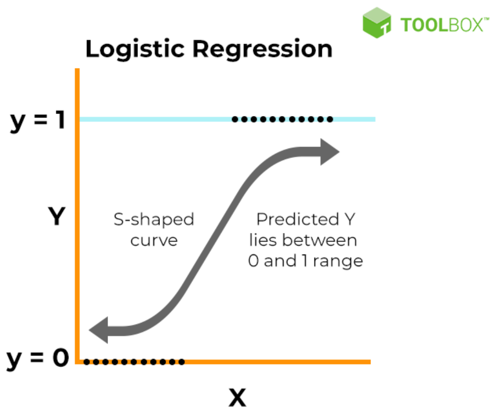


Fig 4 Diagrammatic representation of the logistic regression model

3.2 Evaluation Metrics

To evaluate the performance of the models, several metrics can be used. For this research, some of the commonly used evaluation metrics will be used as listed below:

1. *Accuracy:* This measures the proportion of correctly classified instances out of the total number of instances in the test set. It is calculated as:
   Accuracy = (Number of correct predictions) / (Total number of predictions)
2. *Precision:* This measures the proportion of true positives (correctly classified positive instances) out of all predicted positive instances. Precision is a useful metric when the cost of false positives is high. It is calculated as:
   Precision = (True positives) / (True positives + False positives)
3. *Recall:* This measures the proportion of true positives out of all actual positive instances. Recall is calculated as:
   Recall = (True positives) / (True positives + False negatives)
4. *F1-score:* This is a weighted average of precision and recall, with equal weight given to both. It is calculated as:
   F1 score = 2 * (Precision * Recall) / (Precision + Recall)
5. *ROC curve and AUC:* The ROC (Receiver Operating Characteristic) curve is a plot of the true positive rate (sensitivity) against the false positive rate (1 - specificity) for different threshold values. A higher AUC indicates better performance.

3.3 Dataset and Collection:

The dataset for this project was gotten from a UCI Machine Learning repository and can be found here. The dataset is named South German Credit (UPDATE) Data Set. The data was collected from the year 1973 to 1975 and it is a stratified sample from actual credits with bad credits heavily oversampled. The data contains 1000 instances of customer credit with 700 good credits, 300 bad credits, and 20 predictor variables. The dependent variable is Credit risk with 1 being food credit and 0 being bad credit. In Appendix B is a table showing the variables which contains a brief description for each attribute.

3.4 Data Preprocessing:

3.4.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) was done to explore and summarize the data to gain insights into its characteristics, features, and patterns, to identify important patterns and relationships in the data, as well as to identify potential issues, outliers, or anomalies that may need to be addressed. Some common graphical techniques like scatter plots and box plots, and some common statistical techniques like measures of central tendencies, like mean, median, and mode, and measures of dispersion, like variance and standard deviation were done using the *summary()* function.

From the values of the central tendency, it was deduced that the majority of the data set is normally distributed as the mean and median of a large number of the attributes are close. Nevertheless, Amount, age, and duration seem to have larger values compared to the rest of the variables. As part of EDA, the target variable was visualized to have an idea of the distribution. From the plot in Appendix C, it can be inferred that the dataset is imbalanced as the 0's (bad credit) classes are underrepresented in the dataset compared to the 1's (good credit) classes. Hence there will be a need to balance the dataset before model building to reduce bias and improve decision-making. From the box plot in Appendix D, it can be inferred that even when a lot of the variables may be normally distributed, but there are also variables with outliers. Additionally, some variables like amount are in significantly different scales so some variables need to be scaled.

3.4.2 Data Cleaning

Data cleaning involves preprocessing the data by removing duplicates, handling missing values, and transforming variables. After checking for duplicates, it was seen that the dataset has no duplicates. Additionally, the data types of the variables were checked to see if they were appropriate. The plot in Appendix E shows the percentage of missing values for each variable in the dataset. It can be inferred that there are no missing values.

3.4.3 Standardization

The reason for standardizing is to bring all variables to a similar scale, which can be useful for many purposes, like improving the performance of machine learning models, making visualizations more interpretable, and improving the accuracy of the analysis. To have all the variables in similar scale, age, duration, and amount variables were standardized. These variables have a much larger range of values than others, and scaling them would be important to prevent them from dominating the classification model. Also, other variables have a relatively small range of values that are already consistent, scaling them as well may not provide significant benefits to the model's performance.

The plot below shows the dependent and independent variables after standardization, and now there is not much variation between the variables compared to the previous boxplot before standardization though some outliers still exist.

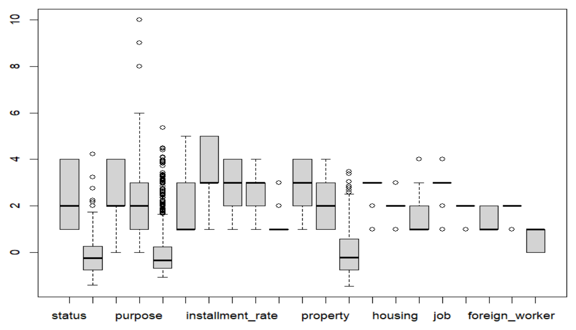


Fig 5 Boxplot of variables after standardization

3.4.4 Correlation Matrix/Plot

Due to the number of variables, the correlation plot does not give so much information, this can be seen in Appendix F. Hence, a correlation matrix was created between each variable and credit_risk. Features with high correlation (negative/positive) suggest that they are important features for credit risk prediction. The coefficients of correlation displayed in the correlation matrix in Appendix G range from 1 to -1. Where -1 depicts perfect negative correlation and +1 depicts perfect positive correlation. Where having a negative correlation with credit risk implies that when that variable decreases, credit risk increases and vice versa. Having a positive correlation with credit risk implies that when that variable increases, credit risk increases and vice versa. But when the correlation is 0, it implies that when that variable increases or decreases, it does not affect credit risk and vice versa.

From the matrix, the variables with the highest positive correlation with credit risk (both negative and positive) can be concluded that status, credit history, savings, and employment duration. While those with the highest negative correlation were duration and property.

3.5 Feature Selection:

Feature selection is the process of selecting a subset of relevant features that are most useful for predicting the target variable in a machine learning model. Feature selection was done to reduce the computational expensiveness of the models to be built and to make the models more efficient.

3.5.1 Boruta

Boruta is a feature selection algorithm used in machine learning for identifying the most relevant features in a dataset. The Boruta algorithm works by comparing the importance of each feature in the original dataset to the importance of features created by permuting the values of the original features. Boruta was used to perform the feature selection process here.

After Boruta feature selection, it was concluded that there are 14 important variables and 6 unimportant variables as shown in the figure above. Information about the attribute statistics can be found in Appendix H.

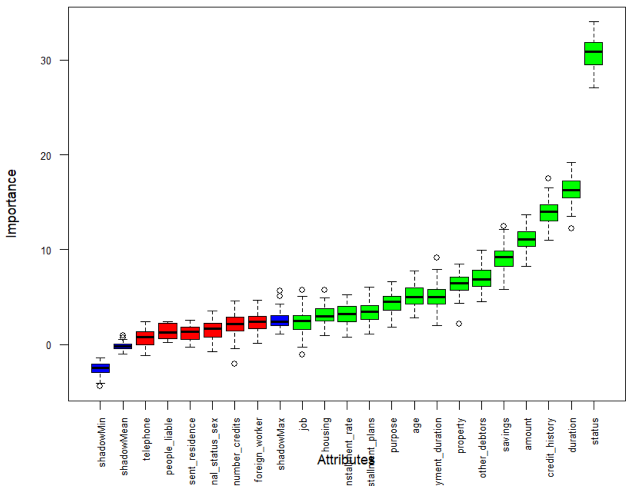


***Fig 6** Plot of the importance of the variables after Boruto feature selection*

The plot above shows the importance of each variable. Status, duration, credit history, amount, savings, other debtors, property, employment duration, age, purpose, installment plans, housing, and job were confirmed as the important features. While number of credits, personal status sex, present residence, people liable, and telephone were confirmed as unimportant.

Hence, the unimportant features were dropped and only important features proceeded for the machine learning implementation. This is because using only the important variables has been proven to lead to improved model performance, faster model training, and improved model interpretability.

4. Machine Learning Implementation

4.1 Data Randomization

The implementation of the machine learning algorithms started by setting the seed for the sake of reproducibility and replication. The data randomization was done to shuffle the data to limit bias and overfitting in the models.

4.2 Data Splitting

The dataset was divided into training and test sets by indexing 70% of the observations as the training dataset and 30% of the observations as the test dataset. The choice of the split ratio was because of the size of the dataset to properly train the model and have a good amount of data for testing.

4.3 Data Balancing

Since the data is imbalanced, the model may be biased towards the majority class (good credit), and it may not perform well on the minority class (bad credit), so the data was balanced to limit the chances of biases. This was done by oversampling by replicating the instances of the minority class. The bad credits were randomly duplicated to equal the good credits. Oversampling was chosen over undersampling for balancing because the bad credit is severely underrepresented in the dataset, and the dataset size is not too large. Undersampling can further reduce the size of the majority class, leading to an even smaller dataset so oversampling was a better choice. Before oversampling, the number of good credits in the training dataset was 479 and the number of bad credits was 221. After oversampling the minority class became as well represented as the majority class. The number of good credits remained 479 and the number of bad credits became 473 as seen in Appendix I.

4.4 Model Building

Four models were built using decision trees, random forests, support vector machines, and logistic regression algorithms. Each model used the same split ratio of 70/30 and the same variables from feature selection.

4.4.1 Logistic Regression:

The first logistic regression model was built with the 14 independent variables. The overview of the model showed that 4 variables (duration, purpose, employment_duration, and other_installment_plans) were not significant so they were dropped and another model was created using the 10 significant variables from model 1 to improve efficiency, interpretability, and use of less computational power. The figure in Appendix J shows the summary of Model 2. The model was fitted into the oversampled dataset. The results indicate that the variables status, credit_history, savings, property, housing, installment_rate, amount, and job are significant predictors of credit risk. The model's deviance residuals suggest that the model fits well, and the AIC value indicates that the model has good fit quality.

4.4.1.1 Variance Inflation Factor

The variance inflation factor was calculated to check for multicollinearity between the independent variables. The result in Appendix K showed that there was no multicollinearity as the square root of the VIF values for all variables was less than 2.

4.4.1.2 Odds Ratios (OR)

Then the odds ratio was calculated and its associated 95% confidence interval in the model using a two-tailed test, i.e. considering both the upper and lower tails of the distribution. From the figure in Appendix L, status, credit_history, savings, other_debtors, age, housing, and job all had an OR greater than 1. This indicates a positive association between these variables and credit_risk. This means the odds of the credit_risk will increase by a factor of 1*OR for a one-unit increase in these variables.

Installment_rate, property, and amount all had an OR less than 1. This indicates a negative association between these variables and credit_risk. This means the odds of the credit_risk will decrease by a factor of 1/OR for a one-unit increase in these variables. After training the model, credit risk was predicted on the test data set with model 2. As shown in the cross table in Appendix M, the model predicted 41 bad credits as good and 21 good credits as bad. Details on the accuracy assessment of this model can be found in Appendix N.

4.4.2 Decision Trees:

The *c50 library* was loaded and the dependent variable was converted to factors as the decision tree only works with outcomes that are factors. Then a decision tree model was built on the oversampled dataset with the selected features from feature selection. A short description of the decision tree showed 952 training data observations and 14 independent variables. The tree size is 105, which means that it has 105 nodes. The non-standard option attempt to group attributes suggests that the algorithm may have attempted to group some predictor variables to simplify the tree structure and reduce overfitting. After training the model, we predicted the credit risk on the test data set. The model predicted 36 bad credits as good and 41 good credits as bad. Details on the accuracy assessment of this model can be found in Appendix O.

4.4.3 Support Vector Machines (SVM)

An SVM model was created with all the significant independent variables from feature selection using the *ksvm( )* function on the oversampled dataset. The type was set to "*C-svc*" and the kernel to "*rbfdot*".

The summary of the SVM model it indicates that the SVM is a C-support vector classification (C-svc) model, where the hyperparameter cost C has been set to 1. The kernel function used is the Gaussian Radial Basis kernel function, and the number of support vectors is 638. The objective function value is -501.549 which is a measure of the quality of the SVM model. Finally, the training error is 0.184, which means that the model misclassifies about 18% of the training data. After training the model, credit risk was predicted on the test data set. The model predicted 25 bad credits as good and 36 good credits as bad. Details on the accuracy assessment of this model can be found in Appendix P.

4.4.4 Random Forests

The *RandomForest library* was loaded and a model was created with all the significant variables using the *randomforest( )* function on the oversampled dataset.

There was an out-of-bag (OOB) estimate of the error rate on the data, which is the average error rate across all trees in the random forest model with 500 trees. The OOB error rate is 10.19%, which means that the model can correctly predict credit risk about 89.81% of the time. The confusion matrix shows that the model has more difficulty predicting the good credit class. The model correctly classified 443 out of 473 samples of class bad credit (93.66%) and 412 out of 479 samples of class good credit (86.02%). After training the model, credit risk was predicted on the test data set. The model predicted 38 bad credits as good and 22 good credits as bad. Details on the accuracy assessment of this model can be found in Appendix Q.

5. Results/Model Evaluation

The performance of each model was evaluated using metrics of accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUROC or AUC).

5.1 Logistic Regression

From the table below, the model achieved an accuracy of 79%. The model has a precision of 83% in classifying bad credit making it a good model and recall of 90.5% means that the model has a 91% chance of not missing out on any of the good credit in the whole observation. The F1-score for the model is 86.6%.

Metric	Value
Accuracy	79.3%
Precision	82.9%
Recall	90.5%
F1-score	86.6%
AUC	0.69

Table 1 *Results of the Logistic regression model Prediction*

This model had an Area under the receiver operating characteristics (AUROC) curve of 0.69 which means that there is a 69% chance that the model will be able to distinguish between positive class (bad credit) and negative class (good credit). This is a fairly good model.

5.2 Decision Tree

From the table below, the model achieved an accuracy of 74.3%. The model has a good precision of 81.4% in classifying bad credit and a recall of 83% means that the model has an 83% chance of not missing out on the good credits. The F1-score for the model is 82.3%. The AUROC curve of 0.679 indicates there is a 68% chance that the model will be able to distinguish between Good credit and Bad credit. This is also a fairly good model.

Metric	Value
Accuracy	74.3%
Precision	81.4%
Recall	83%
F1-score	82.3%
AUC	0.679

Table 2 *Results of the Decision Tree model prediction*

5.3 Support Vector Machines

From the table below, the model achieved an accuracy of 79.7%. The model has a precision of 88.1% in classifying bad credit and an 83.7% chance of not missing out on Good credits (recall). The F1-score for the model is 85.8%. The AUROC curve of 0.76 represents that there is a good chance that the model will be able to distinguish between Good credit and Bad credit. This is a good model.

Metric	Value
Accuracy	79.7%
Precision	88.1%
Recall	83.7%
F1-score	85.8%
AUC	0.76

Table 3 Results of the SVM model

5.4 Random Forest

From the table below, the model achieved an overall accuracy of 80%. The model has a good precision of 83.9% in classifying bad credit and recall of 90.05% meaning the model has a 90% chance of not missing out on the Good credits. The F1-score for the model is 86.88%. The AUROC curve of 0.71 indicates there is a 71% probability of the model properly distinguishing between Good credit and Bad credit. This is a good model.

Metric	Value
Accuracy	80%
Precision	83.97%
Recall	90.05%
F1-score	86.88%
AUC	0.71

Table 4 Results of the Random Forest Model

6. Discussion

6.1 Comparison and Selection

The primary goal here is to identify as many high-risk credit applicants as possible. This is because the bank may suffer significant financial losses due to false positives, whereas false negatives may result in lost business and loan denials for customers. Therefore, reducing false positives is more crucial than reducing false negatives since the consequences of false positives are more severe. Overall, the frequency with which the models identified bad credit as good credit (false positives) is greater than their identification of good credit as bad credit (false negatives). The model performance will be compared based on accuracy, precision, and AUROC score where precision is given more priority because when false positives are particularly costly, precision is the most appropriate metric to use when selecting a model.

Model	Accuracy	Precision	AUROC
Logistic Regression	79.3%	82.9%	0.69
Decision Tree	74.3%	81.4%	0.68
Support Vector Machine	79.7%	88.1%	0.76
Random Forest	80%	83.97%	0.71

Table 5 Summary of model results

From the table above, for accuracy, the random forest model outperforms the other models, followed by the support vector machine model, the logistic regression model, and then the decision tree model. For precision SVM, random forest, logistic regression, and Decision tree respectively.

Lastly, for AUCROC, the SVM had 76% accuracy in identifying good and bad credit. The random forest model and logistic regression followed with 71% and 69% respectively. On the other hand, the decision tree model had the lowest accuracy of 68% in detecting good or bad credits. Considering all the evaluated metrics, the SVM Model was identified as the best model. This is because it showed the best ability in classifying bad credits properly (precision and AUC). Even with a slightly lower accuracy compared to the random forest, SVM showed the best performance since satisfies the aim of this research in identifying bad credits as the consequences of bad credits are more severe.

6.2 Model Optimization

Optimization of a model can be done by hyperparameter tuning, ensembling, and so on. Here the best model (SVM) was optimized using hyperparameter tuning. The hyperparameters were tuned using cross-validation with the *caret* and *e1071* packages.

This code snippet in Appendix R shows the optimization of the SVM algorithm hyperparameters using cross-validation. The model predicted only 17 bad credits as good and 50 good credits as bad. This shows that the model has improved and can identify more bad credits. Details on the accuracy assessment of this model can be found in Appendix S.

Metric	Value
Accuracy	78%
Precision	90.9%
Recall	77.3%
F1-score	83.6%
AUC	0.78

Table 6 Results for the optimized model

From the table above, the model's accuracy dropped to 78%. Nevertheless, the precision is the highest so far. Meaning this model identifies more bad credits. This is the best model because the AUROC shows a 78% probability of the model properly identifying Bad and good credits and the aim of this work is to reduce the risk of defaults and financial losses.

7. Conclusion

In conclusion, the use of machine learning-based approaches for customer credit rating prediction has shown great promise in the financial industry. The four popular algorithms: decision trees, random forests, support vector machines, and logistic regression, have been extensively used and compared for their accuracy and effectiveness in predicting credit risk. Overall, the SVM algorithm was the most effective model for bad credit rating prediction in this study. By leveraging the power of machine learning algorithms, financial institutions can improve their decision-making process and reduce the risk of granting loans to potentially high-risk customers. The results of this research could have significant implications for the banking and financial industry, by enabling more accurate and efficient credit rating predictions and reducing the risk of defaults and financial losses.

The study has some limitations. The dataset used may not be representative of all credit rating scenarios, and the performance of the algorithms may vary depending on the dataset. Additionally, the study did not consider the impact of demographic and economic factors on credit rating, which may influence the accuracy of the predictions.

Further research should consider the impact of demographic and economic factors on credit rating and investigate the performance of machine learning algorithms on a more diverse dataset. Additionally, research can be done to explore the integration of additional machine learning techniques, ensemble techniques, alternative algorithms, and improved feature selection for credit risk prediction to enhance the accuracy of credit scoring models using a more diverse dataset. With continued research and refinement, machine learning-based credit rating prediction can be a valuable tool for financial institutions to manage risk and improve business outcomes.

References

[1] A. Motwani, P. Chaurasiya, and G. Bajaj, ‘Predicting Credit Worthiness of Bank Customer with Machine Learning over Cloud’, *International Journal of Computer Sciences and Engineering*, vol. 6, no. 7, pp. 1471–1477, Jul. 2018, doi: 10.26438/ijcse/v6i7.14711477.

[2] S. Palit, ‘A SVM approach for classification and prediction of credit rating in indian market’, Aug. 2015, doi: 10.13140/RG.2.1.1593.3286.

[3] R. E. Ekong, K. G. Akintola, and B. M. Kuboye, ‘Development Of Credit Scoring Model For Borrowers Using Machine Learning Techniques’, *PERSPEKTIF*, vol. 11, no. 3, pp. 829–838, Jun. 2022, doi: 10.31289/perspektif.v11i3.7180.

[4] S. Bhatia, P. Sharma, R. Burman, S. Hazari, and R. Hande, ‘Credit Scoring using Machine Learning Techniques’, 2017.

[5] Ankit Karmakar, ‘Machine Learning Approach to Credit Risk Prediction: A Comparative Study Using Decision Tree, Random Forest, Support Vector Machine and Logistic Regression’, *FinancialMathematicsTermPaperofAnkitKarmakar*, Mar. 2023.

[6] Blessing Oluwatoyin Adelabu and Tom Carroll, ‘Credit Risk: Assessing Defaultability through Machine Learning Algorithms’, *African Institute for Mathematical Sciences (AIMS), Senegal*, Feb. 2021, doi: 10.13140/RG.2.2.31453.49124.

[7] M. Wallis, K. Kumar, and A. Gepp, ‘Credit Rating Forecasting Using Machine Learning Techniques’, Jan. 2019.

[8] A. Kumar, D. Shanthi, and P. Bhattacharya, ‘Credit Score Prediction System using Deep Learning and K-Means Algorithms’, in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Aug. 2021. doi: 10.1088/1742-6596/1998/1/012027.

[9] Chengyijing Wang, Haining Jiang, Xiaoyan Jin, and Ziyu Zhou, ‘Customer Credit Rating by Machine Learning’, Jan. 2023.

Appendix

A: Code

The R script of this project can be found here

B: Variable Attribute Description Table

S/n	Variable name	Variable class	Description
1.	Status	Categorical	Status of the debtor's checking account with the bank
2.	Duration	Quantitative	Credit duration in months
3.	Credit_history	Categorical	History of compliance with previous or concurrent credit contracts
4.	Purpose	Categorical	Purpose for which the credit is needed
5.	Amount	Quantitative	Credit amount in DM
6.	Savings	Categorical	Debtor's savings
7.	Employment_duration	Ordinal	Duration of debtor's employment with current employer
8.	Installment_rate	Ordinal	Credit installments as a percentage of debtor's disposable income
9.	Personal_status_sex	Categorical	Combined information on sex and marital status
10.	Other_debtors	Categorical	Is there another debtor or a guarantor for the credit?
11.	Present_residence	Ordinal	Length of time (in years) the debtor lives in the present residence
12.	Property	Ordinal	The debtor's most valuable property
13.	Age	Quantitative	Age in years
14.	Other_installment_plans	Categorical	Installment plans from providers other than the credit-giving bank
15.	Housing	Categorical	Type of housing the debtor lives in
16.	Number_credits	Ordinal	Number of credits including the current one the debtor has (or had) at this bank
17.	Job	Ordinal	Quality of debtor's job
18.	People_liable	Binary	Number of persons who financially depend on the debtor
19.	Telephone	Binary	Is there a telephone landline registered on the debtor's name?
20.	Foreign_worker	Binary	Is the debtor a foreign worker?
21.	Credit_risk	Binary	Has the credit contract been complied with (good) or not (bad)?

Table 7 *Table summarizing the dependent and independent variables*

C: Visualization of Target Variable

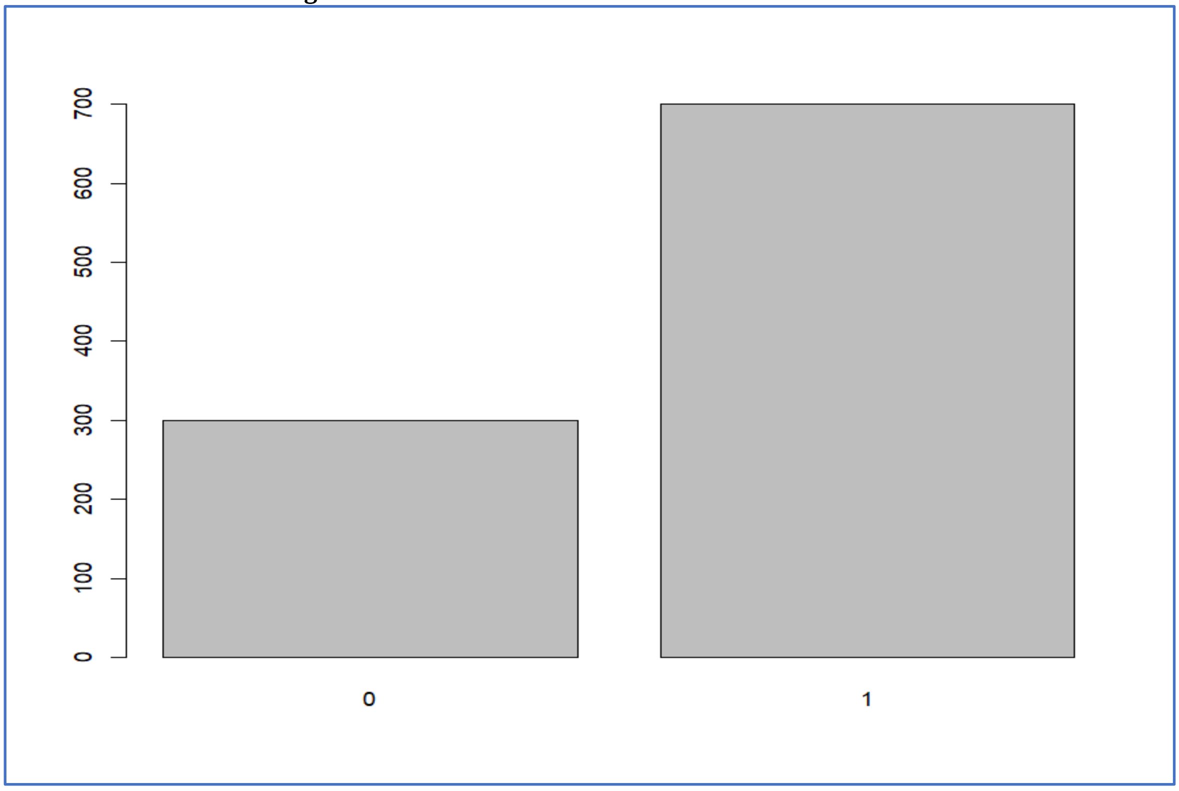


Fig 7 Visualization of the target variable

D: Boxplot of the data set

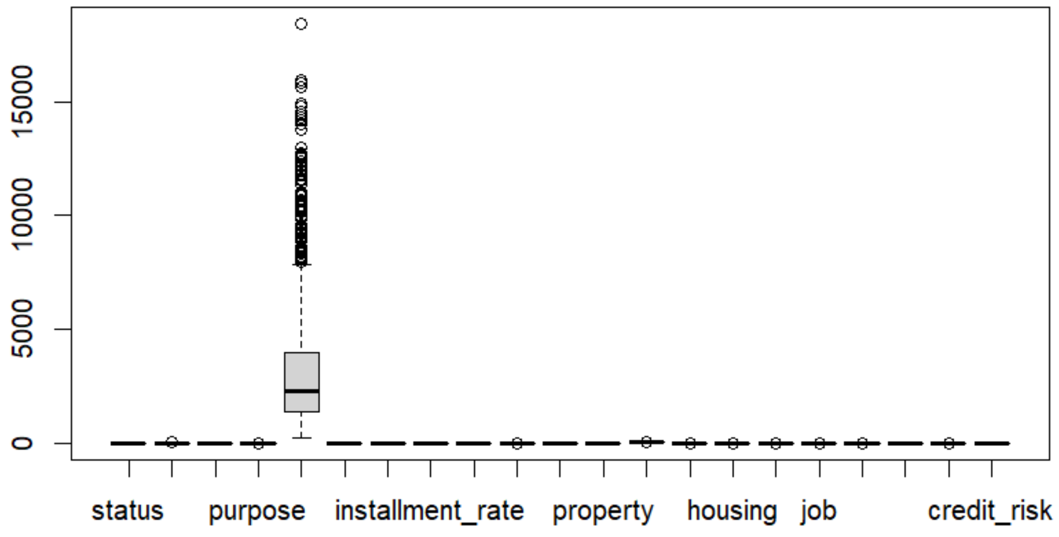


Fig 8 Boxplot of the data set

E: Missingness Map

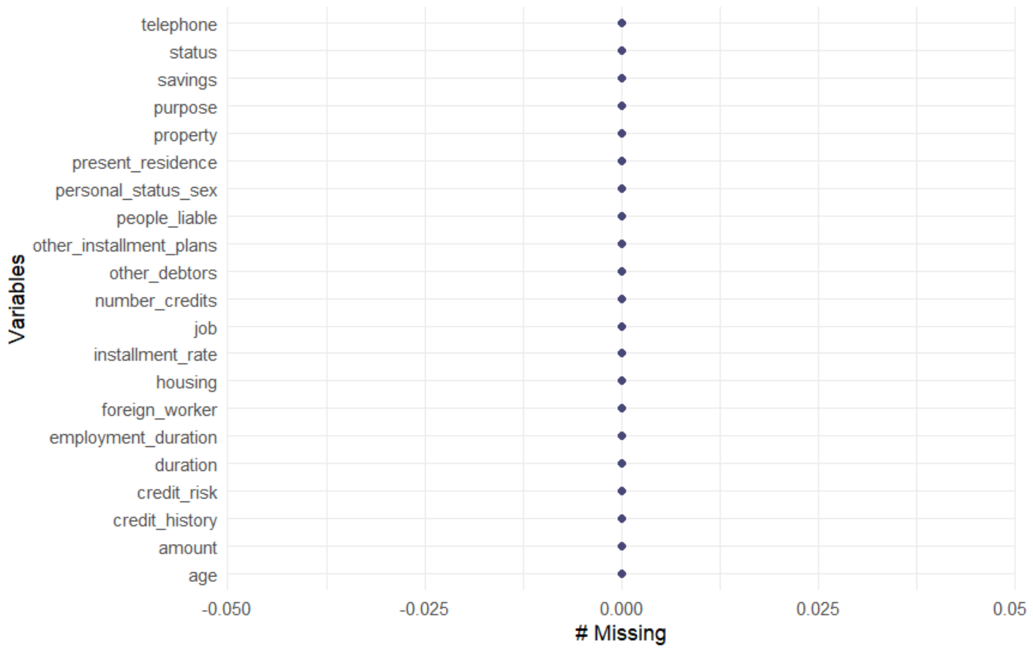


***Fig 9** Missingness map*

F: Correlation Plot

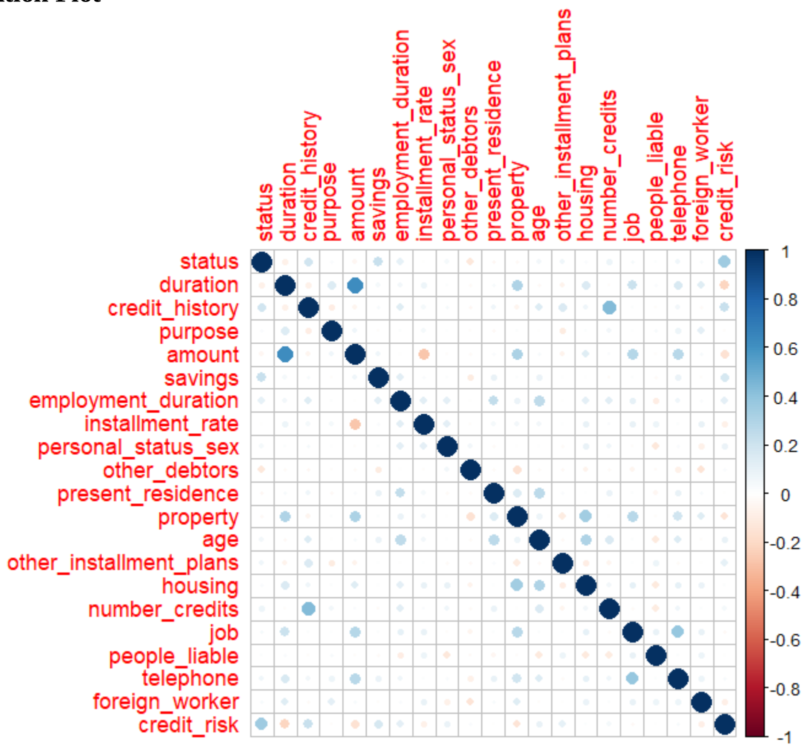


***Fig 10** Correlation plot*

G: Correlation matrix

```
                        status    duration     credit_history     purpose    amount     savings

credit_risk               0.35      -0.21             0.22           0.02     -0.09       0.18


               employment_duration installment_rate personal_status_sex  other_debtors

credit_risk                 0.12              -0.07                0.09              0.00


               present_residence    property    age   other_installment_plans    housing

credit_risk               0.00           -0.14      0.11             0.11                0.02

                 number_credits    job     people_liable    telephone       foreign_worker

credit_risk              0.05        -0.04        0.00           0.04                -0.08

                          credit_risk

credit_risk                    1.00
```

Fig 11 *Correlation matrix of the dependent variable and the independent variables*

H: Attribute Statistics

```
> attStats(fs)
                           meanImp  medianImp     minImp    maxImp  normHits  decision
status                  30.7928826 30.8435335 27.0553287 34.042819 1.0000000 Confirmed
duration                16.3325490 16.2956185 12.2301311 19.186763 1.0000000 Confirmed
credit_history          13.8811333 14.0019391 11.0388800 17.522648 1.0000000 Confirmed
purpose                  4.4128781  4.4976085  1.8755861  6.660119 0.9292929 Confirmed
amount                  11.0625336 11.1044981  8.2167537 13.722535 1.0000000 Confirmed
savings                  9.1727263  9.2319095  5.8488359 12.453703 1.0000000 Confirmed
employment_duration      5.0101308  5.0043219  2.0200551  9.147223 0.9494949 Confirmed
installment_rate         3.1839780  3.2333631  0.8127315  5.221719 0.6868687 Confirmed
personal_status_sex      1.5743712  1.6609850 -0.7872514  3.580118 0.1111111  Rejected
other_debtors            6.9904999  6.8520672  4.5023220  9.930232 1.0000000 Confirmed
present_residence        1.1943666  1.3693453 -0.2720930  2.555267 0.0000000  Rejected
property                 6.4659800  6.4911099  2.1912973  8.488900 0.9898990 Confirmed
age                      5.0945221  4.9707509  2.8300417  7.763452 0.9595960 Confirmed
other_installment_plans  3.4147788  3.4990248  1.1264067  6.063808 0.7171717 Confirmed
housing                  3.0755403  2.9653691  0.9422270  5.736803 0.6969697 Confirmed
number_credits           2.1030918  2.2055177 -2.0138224  4.617487 0.4040404  Rejected
job                      2.3917375  2.4579562 -1.0399376  5.719604 0.4545455 Confirmed
people_liable            1.4215216  1.2586682  0.2084772  2.442876 0.0000000  Rejected
telephone                0.6009418  0.7880006 -1.1231431  2.378248 0.0000000  Rejected
foreign_worker           2.3716277  2.3751855  0.1351165  4.700882 0.4646465  Rejected
```

Fig 12 *Attributes Statistics of the Boruto feature selection*

I: Data Balancing

```
#Model Balancing

library(ROSE)

library(smotefamily)


over = ovun.sample(credit_risk~., data = dftrain, method = "over")$data

> table(over$credit_risk)

  0   1 
473 479 

barplot(table(over$credit_risk))
```

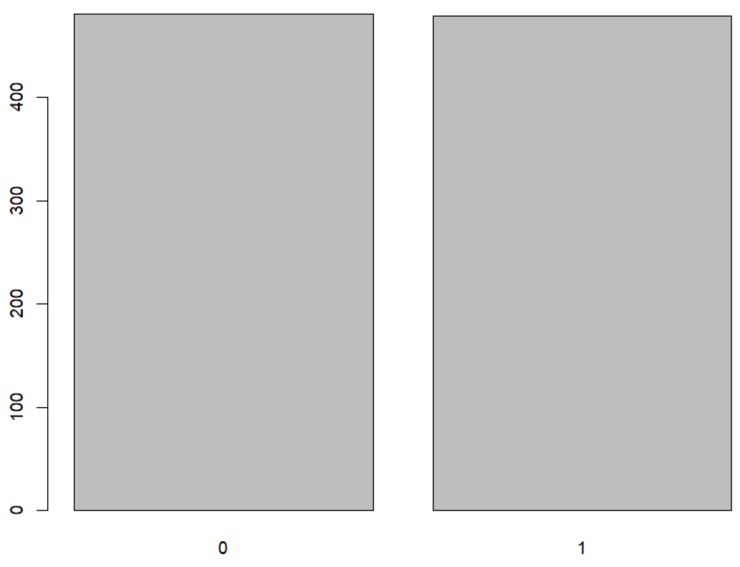


Fig 13 Data balancing process

J: Summary of Linear regression model 2

```
> # Second round excluding non-significant variables
> lgr2 = glm(credit_risk ~ status + credit_history +
+               amount + savings + installment_rate +
+               other_debtors + property + age + housing + job, data = over, family = "binomi
al")
> summary(lgr2)

Call:
glm(formula = credit_risk ~ status + credit_history + amount +
    savings + installment_rate + other_debtors + property + age +
    housing + job, family = "binomial", data = over)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1652  -0.9119   0.2693   0.9223   2.0663

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      -2.79374    0.58097  -4.809 1.52e-06 ***
status            0.51340    0.06512   7.884 3.17e-15 ***
credit_history    0.30803    0.07277   4.233 2.31e-05 ***
amount           -0.37899    0.08489  -4.465 8.02e-06 ***
savings           0.29892    0.05339   5.599 2.15e-08 ***
installment_rate -0.29116    0.07121  -4.089 4.33e-05 ***
other_debtors     0.46270    0.16646   2.780  0.00544 **
property         -0.33775    0.08622  -3.917 8.96e-05 ***
age               0.24444    0.08673   2.819  0.00482 **
housing           0.30520    0.15552   1.962  0.04971 *
job               0.31734    0.12486   2.542  0.01104 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1319.7  on 951  degrees of freedom
Residual deviance: 1064.2  on 941  degrees of freedom
AIC: 1086.2

Number of Fisher Scoring iterations: 4
```

Fig 14 *Summary of Linear regression model 2*

K: Variance Inflation Factor

```
#Variance Inflation factor

vif(lgr2)

sqrt(vif(lgr2)) > 2  # if > 2 vif too high


> sqrt(vif(lgr2)) > 2  # if > 2 vif too high
          status   credit_history           amount          savings installment_rate
other_debtors
           FALSE            FALSE            FALSE            FALSE            FALSE
FALSE
        property              age          housing              job
           FALSE            FALSE            FALSE            FALSE
```

Fig 15 *Variance Inflation Factor*

L: Odds ratio

```
# Calculate Odds Ratio - Exp(b) with 95% confidence intervals (2 tail)

exp(cbind(OR = coef(lgr2), confint(lgr2)))

Waiting for profiling to be done...
                         OR      2.5 %    97.5 %
(Intercept)      0.06119202 0.01934004 0.1890290
status           1.67096850 1.47247739 1.9010779
credit_history   1.36073782 1.18134846 1.5717858
amount           0.68455407 0.57768156 0.8060392
savings          1.34839509 1.21585237 1.4993033
installment_rate 0.74739660 0.64926337 0.8585318
other_debtors    1.58836090 1.15136184 2.2164296
property         0.71337609 0.60169643 0.8439089
age              1.27690371 1.07878614 1.5161837
housing          1.35689518 1.00123019 1.8432982
job              1.37347596 1.07705454 1.7582372
```

Fig 16 Odds ratio

M: Results of the logistic regression prediction model

```
> library(gmodels)
> CrossTable(x = c_h, y = dfpred, prop.chisq = FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  300


             | dfpred
         c_h |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |        38 |        41 |        79 |
             |     0.481 |     0.519 |     0.263 |
             |     0.644 |     0.170 |           |
             |     0.127 |     0.137 |           |
-------------|-----------|-----------|-----------|
           1 |        21 |       200 |       221 |
             |     0.095 |     0.905 |     0.737 |
             |     0.356 |     0.830 |           |
             |     0.070 |     0.667 |           |
-------------|-----------|-----------|-----------|
Column Total |        59 |       241 |       300 |
             |     0.197 |     0.803 |           |
-------------|-----------|-----------|-----------|
```

Fig 17 Results of the logistic regression prediction model

N: Logistics Regression Confusion Matrix

```
> library(caret)
> confusionMatrix(dfpred, c_h, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  38  21
         1  41 200

               Accuracy : 0.7933
                 95% CI : (0.743, 0.8377)
    No Information Rate : 0.7367
    P-Value [Acc > NIR] : 0.01368

                  Kappa : 0.4202

 Mcnemar's Test P-Value : 0.01582

            Sensitivity : 0.9050
            Specificity : 0.4810
         Pos Pred Value : 0.8299
         Neg Pred Value : 0.6441
             Prevalence : 0.7367
         Detection Rate : 0.6667
   Detection Prevalence : 0.8033
      Balanced Accuracy : 0.6930

       'Positive' Class : 1

> # Calculate F1 score
> #F1_score <- 2 * (precision * recall) / (precision + recall)
> F1_score <- 2 * (0.8299 * 0.9050) / (0.8299 + 0.9050)
> F1_score
[1] 0.8658245
```

Fig 18 *Logistic Regression: Confusion matrix and F1 score*

O: Decision Tree Confusion Matrix

```
> confusionMatrix(dtpred, c_h, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  43  41
         1  36 180

               Accuracy : 0.7433
                 95% CI : (0.69, 0.7918)
    No Information Rate : 0.7367
    P-Value [Acc > NIR] : 0.4260

                  Kappa : 0.3516

 Mcnemar's Test P-Value : 0.6485

            Sensitivity : 0.8145
            Specificity : 0.5443
         Pos Pred Value : 0.8333
         Neg Pred Value : 0.5119
             Prevalence : 0.7367
         Detection Rate : 0.6000
   Detection Prevalence : 0.7200
      Balanced Accuracy : 0.6794

       'Positive' Class : 1

> # Calculate F1 score
> #F1_score <- 2 * (precision * recall) / (precision + recall)
> F1_score <- 2 * (0.8364 * 0.8100) / (0.8364 + 0.8100)
> F1_score
[1] 0.8229883
```

Fig 19 *Decision Tree: Confusion matrix and F1 calculation*

P: Support Vector Machine Confusion Matrix

```
> confusionMatrix(svmpred, c_h, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  54  36
         1  25 185

               Accuracy : 0.7967
                 95% CI : (0.7466, 0.8407)
    No Information Rate : 0.7367
    P-Value [Acc > NIR] : 0.009526

                  Kappa : 0.4984

 Mcnemar's Test P-Value : 0.200415

            Sensitivity : 0.8371
            Specificity : 0.6835
         Pos Pred Value : 0.8810
         Neg Pred Value : 0.6000
             Prevalence : 0.7367
         Detection Rate : 0.6167
   Detection Prevalence : 0.7000
      Balanced Accuracy : 0.7603

       'Positive' Class : 1

> # Calculate F1 score
> #F1_score <- 2 * (precision * recall) / (precision + recall)
> F1_score <- 2 * (0.8810 * 0.8371) / (0.8810 + 0.8371)
> F1_score
[1] 0.8584891
```

Fig 20 Support Vector Machine: F1-score and confusion matrix

Q: Random Forest Confusion Matrix

```
> confusionMatrix(rfpred, c_h, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  41  22
         1  38 199

               Accuracy : 0.8
                 95% CI : (0.7502, 0.8438)
    No Information Rate : 0.7367
    P-Value [Acc > NIR] : 0.006507

                  Kappa : 0.4486

 Mcnemar's Test P-Value : 0.052808

            Sensitivity : 0.9005
            Specificity : 0.5190
         Pos Pred Value : 0.8397
         Neg Pred Value : 0.6508
             Prevalence : 0.7367
         Detection Rate : 0.6633
   Detection Prevalence : 0.7900
      Balanced Accuracy : 0.7097

       'Positive' Class : 1

> # Calculate F1 score
> #F1_score <- 2 * (precision * recall) / (precision + recall)
> F1_score <- 2 * (0.8397 * 0.90) / (0.8397 + 0.90)
> F1_score
[1] 0.868805
```

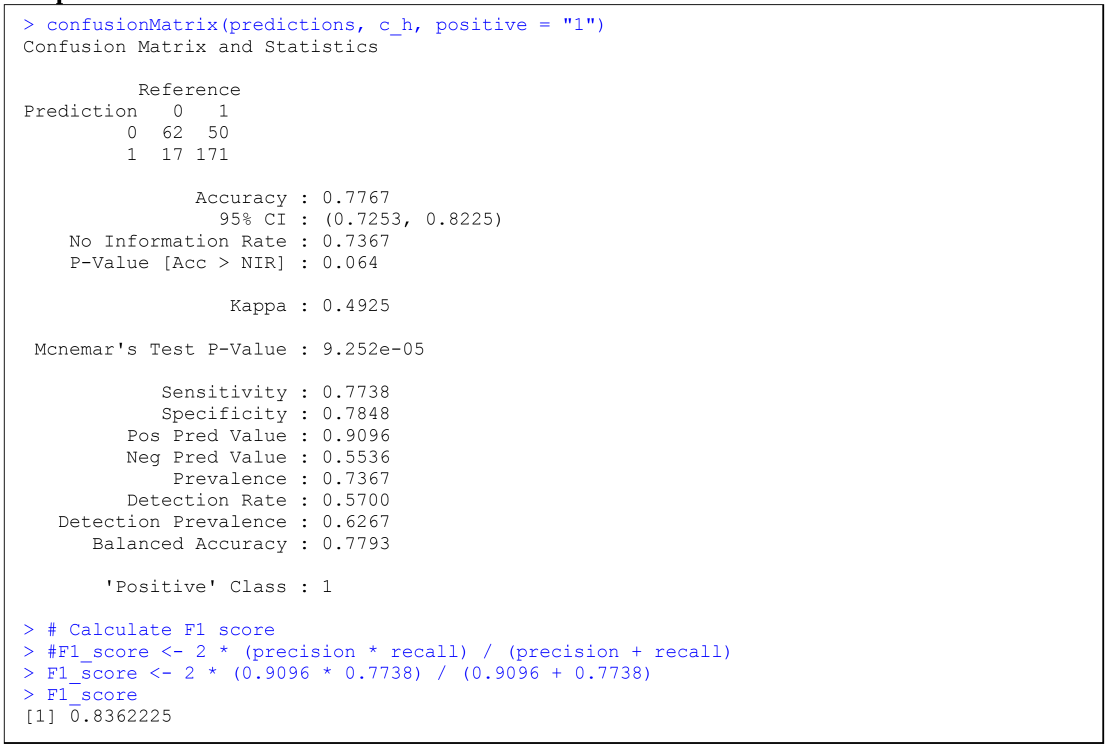
Fig 21 Random Forest: F1 score calculation and confusion matrix

R: Code snippet for model optimization

```
#MODEL OPTIMIZATION

#Create a grid of hyperparameters to search over

tuneGrid <- expand.grid(C = c(0.01, 0.1, 1, 10, 100),

                        sigma = c(0.01, 0.1, 1, 10))

set.seed(123)

#Perform cross-validation to tune the hyperparameters

svmModel <- train(credit_risk ~ status + duration + credit_history + purpose +

                    amount + savings + employment_duration + installment_rate +

                    other_debtors + property + age + other_installment_plans +

                    housing + job, data = over, method = "svmRadial",

                  tuneGrid = tuneGrid, trControl = trainControl(method = "cv", number = 5))


# Select the best hyperparameters

bestC <- svmModel$bestTune$C

bestSigma <- svmModel$bestTune$sigma


# Train the SVM model using the best hyperparameters

finalModel <- svm(credit_risk ~ status + duration + credit_history + purpose +

                    amount + savings + employment_duration + installment_rate +

                    other_debtors + property + age + other_installment_plans +

                    housing + job,

                  data = over,

                  method = "svmRadial",

                  cost = bestC,

                  gamma = 1/(2*bestSigma^2))


# Test the model on the testing set

predictions <- predict(finalModel, dftest)

confusionMatrix(predictions, c_h)
```

Fig 22 *Code snippet for model optimization*

S: Optimized SVM Confusion Matrix



```
> confusionMatrix(predictions, c_h, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  62  50
         1  17 171

               Accuracy : 0.7767
                 95% CI : (0.7253, 0.8225)
    No Information Rate : 0.7367
    P-Value [Acc > NIR] : 0.064

                  Kappa : 0.4925

 Mcnemar's Test P-Value : 9.252e-05

            Sensitivity : 0.7738
            Specificity : 0.7848
         Pos Pred Value : 0.9096
         Neg Pred Value : 0.5536
             Prevalence : 0.7367
         Detection Rate : 0.5700
   Detection Prevalence : 0.6267
      Balanced Accuracy : 0.7793

       'Positive' Class : 1

> # Calculate F1 score
> #F1_score <- 2 * (precision * recall) / (precision + recall)
> F1_score <- 2 * (0.9096 * 0.7738) / (0.9096 + 0.7738)
> F1_score
[1] 0.8362225
```

Fig 23 *SVM: F1 score calculation and confusion matrix after optimization.*