Credit Risk Prediction

Data Analysis & Machine Learning

EBM805C05, Group 2

university of groningen

Anastasia Potsi (s5357071)
Freek Hobbenschot (s3428397)
Muhammet Yilmaz (s3514390)
Vasil Golemetsov (s3954390)

June 9, 2023

Abstract

In this report we predict credit risk using the German Credit data set from Hofmann (1994), containing 1000 observations and 21 features. To predict credit risk, we implemented a Logistic Regression model as well as the machine learning techniques Decision Trees, Random Forests, Boosting (specifically XGBoost), and Feed Forward Neural Networks (FNN). We have used the metrics accuracy, recall, specificity, precision and F1-score to evaluate and compare the different models. Our findings indicate that the Logistic Regression is outperformed by the more sophisticated machine learning techniques. Moreover, Decision Trees showed inferior performance compared to Random Forests and XGBoost. Random Forests demonstrated stability across different scenarios and scores highest on precision (0.732) and specificity (0.915). XGBoost is the best model in terms of accuracy (0.800) and F1-score (0.682). FNN showed promising results in capturing high-risk individuals. FNN in combination with SMOTE yielded the highest recall (0.847). It is argued that recall is the most important metric to evaluate credit risk and therefore FNN is recommended as best model for credit risk prediction. A case study illustrates the potential of the models in a real-world banking scenario.

Keywords: Credit Risk Prediction, Machine Learning, Decision Trees, Random Forest, XGBoost, Feed Forward Neural Networks

1 Introduction

In this report, we focus on a specific issue that has a significant impact on the financial sector: predicting credit risk for customers using machine learning techniques. By identifying risky customers, financial institutions can make informed decisions about granting loans or extending credit, and as a result, reduce the risk of default. We chose this topic because of its potential to benefit the financial sector by providing a tool to aid in decision-making. This project is feasible due to the availability of relevant data and similar studies on the usage of machine learning techniques to predict credit risk, see e.g Addo et al. (2018) and Tixier et al. (2016).

In this paper, we use the data set titled "German credit data" from Hofmann (1994) to predict whether a customer should be given credit or not. This data set consists of 1000 observations and 21 features. These features include income, employment status, credit history, outstanding debts, loan details, personal information, and living situation.

Our methodology will encompass a range of predictive modeling techniques, in order to accurately forecast the credit risk of customers using the chosen attributes. Our objective is to develop a model that effectively classifies customers as either high-risk or low-risk based on their unique characteristics. To achieve this, we will commence with a Logistic Regression model due to its proven efficiency in binary classification problems, see for example Lessmann et al. (2015). Subsequently, we incorporate the following sophisticated machine learning methodologies into our analysis: Decision Trees, Random Forests, XGBoost, and Feed Forward Neural Networks. By implementing and evaluating diverse classification methods, we strive to enhance the robustness and accuracy of our prediction model. This is in line with the approach suggested in Leo et al. (2019).

This report is structured in the following way; Firstly we have the section 'Data Description' in which we describe and summarize the data. Subsequently, we have the section 'Methods and Validation' in which we shortly explain the various methods that we have used, as well as the validation, evaluation methods, and the use of re-balancing methods to improve the imbalance in the data. Thereafter, we continue with the 'Analysis and Insights', in which we discuss the performance of the methods as well as the feature importance. Furthermore, this section contains a short case study that goes into the practical performance as well as the implications of using a prediction model to determine credit risk. Finally, the section 'Conclusion and Further Recommendations' contains our main findings, limitations, and suggestions for further research.

2 Data Description

In this section, we discuss and elaborate on the data set that is used in our study. The data set is a German Credit data set retrieved from Hofmann (1994). It consists of 1000 observations with 21 features (seventeen categorical and four numerical). The data set is clean in the sense that it does not contain any outliers or missing values. Therefore it is not needed to look into imputation, interpolation, or removal of observations.

The German Credit Data, as the title suggests, is particularly suited to our study given its focus on factors associated with credit risk. Comprising both categorical and numerical attributes such as checking account status, credit history, purpose of the credit, employment status, and others, it provides a holistic view of factors contributing to an individual's creditworthiness.

During our exploration, we identified four features that exhibited correlations. The feature *Duration* was found to be highly correlated with *Credit Amount*, and the feature *Property Magnitude* was found to be highly correlated with *Housing*. This high correlation could be problematic for the Logistic Regression, due to problems like multicollinearity. When using machine learning techniques it is not necessary -and sometimes not even recommended- to drop (highly) correlated features. Therefore, as our ultimate goal is to construct a model with the most accurate forecast, we have done the entire analysis of this report with and without dropping *property magnitude* and *duration*, and we found that including these two features significantly improved the accuracy metrics for the XGBoost and FNN methods, whereas it was not a problem for Logistic Regression. That is why we chose to not drop these two features.

We find a ratio of 1:2.33 between the classes 'good' and 'bad' for our target variable (700 good, and 300 bad). This indicates that the distribution of classes in the target variable is not heavily skewed. Therefore, we abstain from applying balancing techniques in this phase of the study. In the section Analysis and Insights, it is shortly discussed how several balancing techniques could help to obtain a more balanced data distribution and how this could help to enhance the performance of some of the (machine learning) models.
Continuing with the feature transformation, we proceeded to convert the categorical features into numerical representations. We also created several pie charts to visually represent and explore the data further, these bar charts can be found in Figure 5 in the Appendix. This helps in the interpretation of patterns and relationships within the data set and maybe helps to get more familiar with the data set.
Table 1, shows the information for the first five rows of the data set and how this is structured.
Observe that the data set contains 21 attributes such as *Checking Status*, *Credit History*, *Purpose*, *Credit Amount*, and others that provide information about different aspects that help in the classification of credit risk.

checking_status	duration	credit_history	purpose	credit_amount	savings_status	employment
<0	6	critical/other existing credit	radio/tv	1169	no known savings	≥7
0≤X<200	48	existing paid	radio/tv	5951	<100	1≤X<4
no checking	12	critical/other existing credit	education	2096	<100	4≤X<7
<0	42	existing paid	furniture/equipment	7882	<100	4≤X<7
<0	24	delayed previously	new car	4870	<100	1≤X<4

Table 1: Data set Information (Part 1)

If interested, please consult the information in Table 4 located in the appendix, which provides details about the data set including a brief description and an explanation of the transformation.
Figure 4, given in the appendix, shows correlations among the features in the data set. It is apparent, as we mentioned before, that there is a strong correlation between *Duration* and *Credit Amount* and between *Property Magnitude* and *Housing*.

3 Methods and Validation

In this section, we outline the approaches employed to develop credit risk predictions and we outline the validation process undertaken to assess their efficacy. We recognize the potential of machine learning techniques, particularly supervised machine learning, in enhancing credit risk assessment and prediction. Therefore, our emphasis lies in utilizing well-established supervised machine learning models and methodologies to forecast whether a borrower is classified as good (low-risk) or bad (high-risk) within the data set.

3.1 Methods

We are deploying a selection of machine learning techniques to ensure a comprehensive and accurate approach in order to accurately assess the task of predicting high and low credit risk. We will include various machine learning techniques, such as Logistic Regression, Decision Trees, Random Forests, Boosting (specifically XGBoost), and Feed Forward Neural Networks. We've chosen these particular techniques due to their inherent properties that make them suitable for this task, yet each method has its strengths and limitations which need to be recognized. We will now shortly discuss these models and explain why they are suited for our tasks and provide their strengths and weakness, for our problem.
Firstly, we will implement Logistic Regression, which is a robust and intuitive technique for tackling binary classification problems. Logistic Regression excels in handling linearly separable scenarios by generating probability scores for observations. This makes it ideal for our use case of determining credit risk since the output can be interpreted as the probability of an individual falling into high or low-risk categories. However, due to its simplicity, Logistic Regression tends to struggle with complex relationships and interactions between variables, but it is a good initial model that can act as a baseline.

Next, we will consider Decision Trees, where we will follow Kroese et al. (2019). A Decision Tree is a graphical representation of a flowchart-like structure that models decisions and their possible consequences. It consists of nodes, which represent decision points based on different features or variables, and branches, which represent possible outcomes or paths that can be taken based on the decisions. Each internal node splits the data based on a specific feature, while each leaf node represents a final outcome or prediction. Decision trees are commonly used for classification and provide a transparent way to interpret the decision-making process within the model.
Related to Decision Trees, we also include Random Forests, where we again will follow the approach outlined in Kroese et al. (2019). Random forest is an ensemble learning method that can be used for classification and regression problems. The idea behind this method is to create multiple Decision Trees, each based on its own random subset of the training data. Each of these Decision Trees consists of its own set of branches with decision outcomes and leaf nodes that represent class labels. These Decision Trees are the same as the Decision Trees that were discussed before. In the end, we take the majority vote of the ensemble of trees.
Also related to Decision Trees and Random Forests is Boosting, specifically, XGBoost of Chen et al. (2015). Boosting is an ensemble technique that iteratively constructs Decision Trees and as a prediction takes the weighted sum of all the predictions of all the trees. The idea is that by combining many weak learners (individual predictions of trees), the final prediction will be accurate. This idea comes from the fact Decision Trees are greedy in nature. That is, they decide to make a "split" of the data set, based on only looking at the reduction it gives to the sum of squared errors, in the next level of the tree (and not further down the tree). One weakness of Gradient Boosted Regression Trees (GBRT) is their potential for overfitting, which we will address by ensuring proper k-fold cross-validation on the hyperparameters.
The last model that we have used in our analysis is a Feed Forward Neural Network, utilizing the TensorFlow API from Abadi et al. (2015) to construct Neural Networks. An FNN is a powerful model that can be used for capturing non-linear patterns within the data set. The idea is that the input data gets mapped to the next layer of nodes and each node thus takes in the previous layers' data. There are weights and biases that need to be learned and each node has an activation function. These weights and biases get learned through stochastic gradient descent, where the training data set in batches is put through the FNN, and for each batch we calculate the gradient, doing so we train the model using gradient descent. However, one drawback of this model is its challenging interpretability, which is of course important in a practical problem such as this one.

3.2 Validation & Rebalancing

This combination of machine learning techniques will provide us with a broad and robust mechanism for predicting credit risk, which we will fine-tune and validate using cross-validation. By leveraging the strengths of each method and understanding its limitations, we strive for an optimal balance between predictive power and model interpretability. In order to optimize the performance of the various models, we will employ five-fold cross-validation to tune our hyperparameters as needed. That is, we first split 80% of the data set into a training set and the remaining 20% will act as a test set. Then for 5-fold cross-validation, we fold the training set five times and in each iteration, one of the folds will act as a validation set whereas the other four folds will act as a training set. Some examples of hyperparameters that needed to be optimized were the maximum tree depth, the maximum number of trees, and the hidden layers in the FNN. For the XGBoost we deployed a gridsearch algorithm that checked all the combinations of all the hyperparameters that needed to be optimized. We refer to Table 5 in the appendix for a list of all the hyperparameters that are optimized, of each method separately.
A major issue in our data set was the relative imbalance of the high-risk class versus the low-risk class. While in general, the data is relatively balanced in the sense that high-risk was 30% and low-risk was 70% of the data, which is not significantly unbalanced. However, since we also only have 1000 observations, we hypothesized that the models might have difficulty learning patterns that are associated with high-risk individuals. For a comprehensive understanding of the model's behavior, we conducted the analysis both with and without utilizing the SMOTE technique [1]. This approach allows us to examine and compare the

[1] We also conducted the analysis with the random oversampling and random undersampling techniques, however, these techniques

performance of the models in both scenarios.
The Synthetic Minority Over-sampling Technique (SMOTE) by Chawla et al. (2002) is a statistical approach used to address the imbalance in machine learning data sets by generating synthetic examples of the minority class. Instead of simply replicating minority class instances, SMOTE synthesizes new instances from the feature space of the minority class. The algorithm does this by taking each minority class sample and introducing synthetic examples along the line segments joining all of the k minority class nearest neighbors. We set k equal to 5 in this report.

3.3 Model Evaluation

Predictive models are often assessed via performance indicators such as accuracy, sensitivity, specificity, precision, and F-measure were we follow the approach of Fawcett and Provost (2013). These metrics are derived from values in a confusion matrix: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN} \qquad \text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

A confusion matrix visualizes these values, depicting actual versus predicted class outcomes. For our particular task, all metrics are vital, as they provide a comprehensive evaluation of our predictive model's performance. However, Recall takes precedence due to the nature of our problem.
Recall is the most crucial metric in the context of this problem, because it measures the proportion of actual high risks that are correctly identified. In other words, a high recall ensures that we minimize False Negatives, this is a situation where we mistakenly classify someone as a 'low' risk when they are in fact 'high'. This is particularly important as such misclassifications could lead to severe consequences, due to capital being lost. Hence, while we strive for a balanced performance in all metrics, our primary focus is to maximize Recall.

4 Analysis and Insights

4.1 General comments

Table 2 and Table 3 show the results of all the different methods that we have used and the different metrics we have tracked to assess which one yielded the most accurate predictions. We see that the more advanced methods outperform the relatively simple Logistic Regression and Decision Tree, both when we used the SMOTE technique and not. A possible reason why Logistic Regression performs worse than the other models is that Decision Tree-based algorithms like Random Forest and XGBoost automatically perform feature selection, making them more robust to irrelevant features. Whereas Logistic Regression uses all features to come up with a prediction, which can potentially degrade performance if some features are irrelevant or noisy. Another reason would be that Logistic Regression is a linear model and if the relationship between the features and the target variable is non-linear, then Logistic Regression might not perform as desired, as is for example discussed in Hastie et al. (2001).
We infer that the Decision Tree is outperformed by the ensemble tree techniques, Random Forest, and XGBoost. This is intuitive as Random Forest randomly drops some features to come up with a prediction and in that way reduces the variance of the prediction. Additionally, boosting uses an ensemble of trees to come up with a prediction, in that way combines many weak learners (Decision Trees) to iteratively come up with a good prediction. Each tree then would correct the errors that are generated by the previous trees.

resulted in worse performance of the prediction models.

Regarding the Feed Forward Neural Network, we observe that the FNN model seems to perform poorly overall, particularly in terms of precision and F1-score. Its performance is noticeably unstable between using and not using SMOTE, this is potentially due to the fact that our training set is very small when compared to typical machine-learning problems and Neural Networks need more data to provide accurate solutions. While it can identify positive cases very well when SMOTE is used, it struggles to do so when SMOTE isn't used. Its ability to correctly identify negative cases (specificity), however, improves significantly without SMOTE.

In general, the impact of using SMOTE seems to significantly improve the recall rate for all models, which indicates that these models can identify more actual positive cases when the minority class is oversampled. This is especially apparent with the FNN model where the recall score improves drastically. It is a bit unfortunate that while the recall increased after using SMOTE, SMOTE also significantly reduces the precision in all models except the Random Forest. This could be because while SMOTE helps identify more positive cases, it also increases the likelihood of falsely predicting negatives as positives.

We want to note that the Random Forest method yielded the most stable results, as all the other methods provided very different results when we used SMOTE or not. This is most likely due to the fact that our data set again was relatively small (1000 observations), so sampling the minority class has a major impact on the different accuracy metrics of almost all models. Figure 1 is a radar figure that displays the different metrics across a unit circle, the figure is especially nice to quickly get an idea of how the metrics compare against each other, across the different models.

Overall, XGBoost yielded the most accurate forecasts and scored relatively high across all metrics, however, for our specific problem, we put more emphasis on the recall of the model, which then we should choose the FNN with SMOTE. We have to note however that an FNN model is highly difficult to interpret, therefore we also observe that the Decision Tree yielded similar results as the FNN, but a Decision Tree is more intuitive, so that might be a good model to use in practice.

We conclude this section by evaluating the logistic regression, XGBoost and FNN models using various accuracy metrics at different threshold values, denoted as p. The threshold p is defined as the probability where an individual x is considered high risk if the output probability of high-risk, $P(r|x)$, exceeds p. Adjusting this threshold has an impact on the classification results: lowering the threshold predicts more people as low risk while raising the threshold predicts fewer people as high risk. A summary of these effects is provided in Table 6 (in the Appendix). It is particularly interesting that the FNN is able to capture 100% of the high risk in the test-data set when we lower the treshold[2]. More data is needed to come to a firm conclusion, but it seems that the FNN is very capable of capturing high-risk individuals.

Table 2: Model Comparisons with SMOTE

Metric	Logistic Regression	Decision Tree	Random Forest	XGBoost	FNN
Accuracy	0.715	0.680	0.745	**0.800**	0.685
Precision	0.511	0.528	**0.732**	0.642	0.481
Recall	0.763	0.800	0.429	0.729	**0.847**
Specificity	0.695	0.615	**0.915**	0.830	0.617
F1-score	0.612	0.636	0.541	**0.682**	0.356

Table 3: Model Comparisons without SMOTE

Metric	Logistic Regression	Decision Tree	Random Forest	XGBoost	FNN
Accuracy	0.785	0.750	0.750	**0.800**	0.735
Precision	0.690	0.727	**0.813**	0.711	0.688
Recall	0.492	0.457	0.371	**0.542**	0.186
Specificity	0.908	0.908	0.954	0.901	**0.965**
F1-score	0.574	0.561	0.510	**0.616**	0.114

[2]Due to the random initialization of the weights(and the low number of data points), the metrics differ slightly after each training, we took the average of five different FNN models

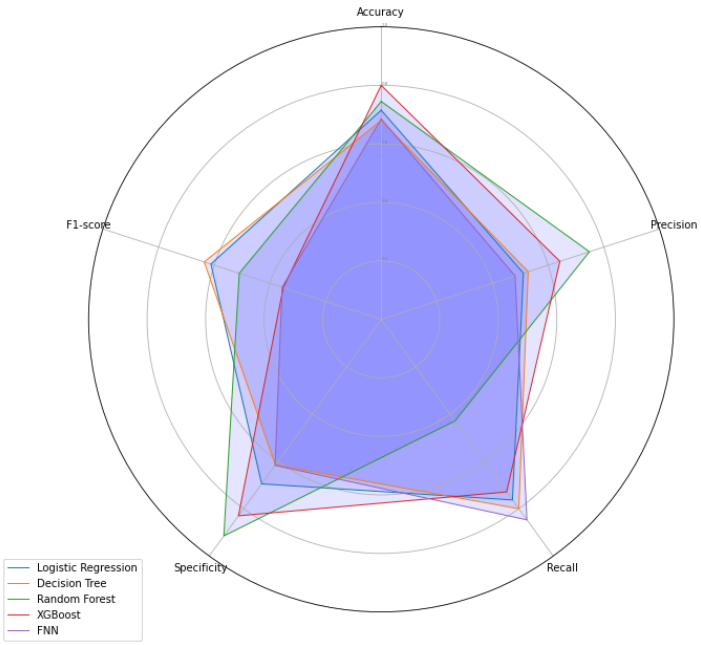

Figure 1: Comparison of the different accuracy metrics of all the methods, with SMOTE

4.2 Case study

To give a concrete example of this model can be used in practice and what the implications will be, we will provide this case study. We will analyze the reduction in cost(if any) in a bank/credit provider that will deploy this model practice. Therefore we initially choose the FNN model with SMOTE that is given in 2. So we assume that 2000 people ask the bank for a loan each week, where 90% percent of the people that ask for credit are good and 10% are bad. Moreover, we assume that the hourly wage of an employee that needs to assess each claim individually is 25$. The average time to determine whether someone is good is 1/4th and 2/4th of an hour. Individuals classified as high-risk often require more time on average to determine their high-risk status because they may possess profiles that are more challenging to analyze. Then, the average total cost for the bank without using the machine learning model is:

$$\begin{aligned}\text{Totalcost} &= n * n_l * t_l * W + n * n_h * t_h * W \\ \text{Totalcost} &= 2000 * 0.9 * 0.25 * 25 + 2000 * 0.1 * 0.5 * 25 = 13750.\end{aligned}$$

However, when we use a machine learning model, the equation gets a little bit more involved and we need to make some additional assumptions. When someone is classified as a good risk, then we do not look into that profile anymore and let that person take the loan. This comes however with the risk that of course someone will be classified as good, but is a bad risk (False negative). So we do have to take into account the potential loss that comes with this. The following equation provides the total expected loss that takes into account the false negatives:

$$\begin{aligned}\text{EL} =&(1 - \text{recall}) * n * p_l * \text{avgcredit} * (1 - RGD), \\ \text{EL} =&(1 - 0.847) * 2000 * 0.1 * 3983 * (1 - 0.9) = 12050.28,\end{aligned}$$

where $1 - recall$ is the proportion of bad risk that gets through, by getting classified as a good risk. On average bad risks take a loan of 3983 units and we assume that we can retrieve 90%(RGD = Recovery Given Default) of that amount when the high-risk defaults. (It becomes evident in this context why falsely classifying someone as low risk when they are actually high risk is more detrimental compared to the reverse situation. The potential financial loss incurred by misclassifying individuals as low risk when they are actually high risk is significantly higher.). Now, we also argue that, since our model is tuned such that

a significant amount of people get falsely classified as a bad risk, the people that get classified as good risk can take the loan without human assessment, this reduces the number of people that needs to be checked manually. So the total expected cost when using the machine learning model would then be:

$$\text{TC} = (n * p_h * (1 - \text{specificity}) * t_h^{ml}) * W + (n * p_l * \text{recall}) * t_l^{ml}) * W + EL \quad (1)$$

$$\text{TC} = (2000 * 0.9 * (1 - 0.617) * 0.125) * 25 + ((2000 * 0.1 * 0.847) * 0.25) * 25 + 12050.28 = 15263.41 \quad (2)$$

Our rationale is as follows: Among those considered low-risk, the proportion classified as low-risk (no manual checks required) is equivalent to specificity. Hence, the proportion of low-risk individuals still needing checks is (1-specificity). Similarly, for the inherently high-risk group, recall represents the proportion predicted as high-risk (requiring manual checks). However, 1 - recall denotes the proportion incorrectly predicted as low-risk (potentially receiving credit without checks). Additionally, using the ML model for an initial prediction reduces average assessment time by 50% (tml for low-risk and tmlh for high-risk). The total cost reduction equals 13750 - 15263.41 = -1513.41. So in this example calculation, using XGBoost increased the total costs for the bank. That is why we would recommend using the Feed Forward Neural Network with the threshold set equal to 0.2, as can be found in Table 6 (Appendix). In that case, using the formula given in Equation 1 we see that the total reduction in cost would be: $13750 - 5997.5 = 7752.5$ units. It becomes evident that we can optimize this function by identifying the model that results in the lowest total cost[3].
Naturally, this simulation relies on many assumptions, some more debatable than others. However, the key takeaway is that even a simple simulation can help us evaluate the benefits of using machine learning models, which seem to be quite advantageous in certain scenarios.

4.3 Feature importance

In this section, we will shortly assess which features were important during training. Feature importance of XGBoost is calculated based on the number of times a feature is used to split the data across all trees in the model, during the training phase [4]. Feature importance of Random Forest is calculated based on the average decrease in (Gini) impurity caused by the feature across all trees. For Random Forest, the values are relative between 0 and 1. In both cases, a higher value indicates a greater contribution to the model's performance.

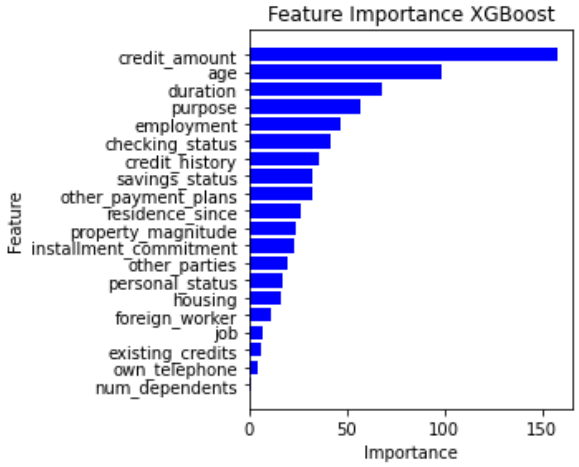

Figure 2: Feature Importance without SMOTE - XGBoost

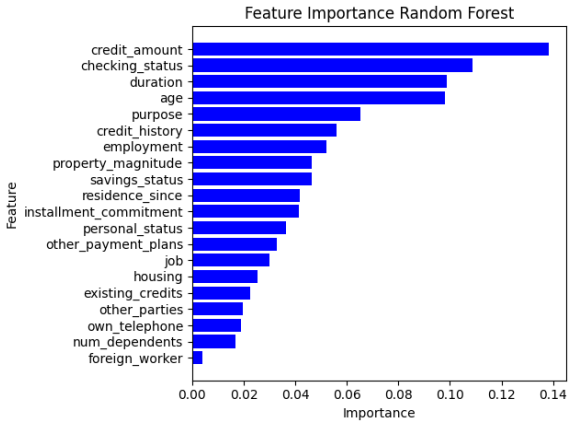

Figure 3: Feature Importance without SMOTE - Random Forest

[3] Actually for this set of parameters, this FNN setup yielded in the lowest total cost

[4] Note that there are several ways of doing this, one is not necessary better than the other one. Another method could be to calculate the gain in the loss function

We see for both models that the feature *Credit Amount* has the highest contribution. Therefore, the credit amount of a customer seems to play the most important role when predicting someone's credit risk. Other important aspects seem to be the age of the customer, the duration of the loan, and the purpose of the loan. Also notable, the number of dependence of the customer does play a minor role when using Random Forest, while it seems to play no significant role when using XGBoost. Other than that, there does not seem to be much difference between the feature importance for both models.

5 Conclusion and Further Recommendations

After a comprehensive analysis of various methodologies designed to predict credit risk, we've observed that the more advanced models generally outperform the Logistic Regression on all of the performance metrics. The reliability of these findings is supported by the use of several metrics such as accuracy, recall, and precision to evaluate the different models, with a special emphasis to recall in our case. We found that the XGBoost algorithm provided the most balanced model, as can be observed from the relatively high accuracy metrics (across all metrics). Moreover, we argue that recall is the most important metric to evaluate credit risk prediction and we observed that the FNN with SMOTE yielded the highest score on this (0.847). In addition, when examining alternative values for the threshold, we can infer that the FNN is able to 100% capture the high risks in the test data. More data is needed to affirm this conclusion, however, it seems that the FNN is highly capable of capturing the high risks in the data set. Since recall is perhaps the most important metric for the prediction of credit risk, we would argue that the FNN could be deployed in practice. However, the utilization of this model would rely on the client's preferences (and if the client agrees to use a difficult-to-interpret model as an FNN). Worth mentioning, is that the use of SMOTE significantly improves the recall rate for all models, indicating that these models became more effective at detecting customers who are more likely to default. This is very important for real-world scenarios, helping financial institutions to identify high-risk customers.
Further research could be done into other machine learning techniques such as Bagging or Support Vector Machines. Also, it is recommended to look into improvements of the current methods, such as the use of an elastic net for Logistic Regression, as it is a regularization technique that combines L1 and L2 norms preventing overfitting. Furthermore, the impact chances in the feature selection and the use of balancing techniques that will help to improve the balance between recall and precision are interesting topics for further research. Last but not least, as credit risk prediction is a sensitive area, ethical considerations are also important in terms of fairness (i.e. discrimination based on race, age, and sex), and privacy.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Peter Martey Addo, Dominique Guegan, and Bertrand K Hassani. Credit risk analysis using machine and deep learning models. *Risks*, 6(2):38, 2018.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

Tom Fawcett and Foster Provost. *Data Science for Business*. O'Reilly Media, Sebastopol, CA, 2013.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Hans Hofmann. German credit dataset. OpenML, 1994. Available at: `https://www.openml.org/search?type=data&sort=runs&id=31&status=active`.

D.P. Kroese, Z.I. Botev, T. Taimre, and R. Vaisman. *Data Science and Machine Learning: Mathematical and Statistical Methods*. Chapman & Hall/CRC machine learning & pattern recognition. CRC Press, Boca Raton, 2019. ISBN 9781138492530. URL `https://people.smp.uq.edu.au/DirkKroese/DSML/`.

Martin Leo, Suneel Sharma, and K Maddulety. Machine learning in banking risk management: A literature review. *Risks*, 7(1):29, 2019.

Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, and Lyn C Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124–136, 2015.

Antoine J-P Tixier, Matthew R Hallowell, Balaji Rajagopalan, and Dean Bowman. Application of machine learning to construction injury prediction. *Automation in Construction*, 69:102–114, 2016.

Appendix

Attribute	Description	Transformation
Checking Status	Reflects the amount of money in the checking account	1: <0 DM, 2: $0 \leq$ DM <200 DM, 3: $X \geq 200$ DM, 4: no checking
Duration in month	The length of the credit term in months	(numerical)
Credit History	The record of a borrower in terms of previous loans and repayment	0: no credits taken/all credits paid back duly, 1: all credits at this bank paid back duly, 2: existing credits paid back duly till now, 3: delay in paying off in the past, 4: critical account/other credits existing (not at this bank)
Purpose	The intended use of the loan	0: new car, 1: used car, 2: furniture/equipment, 3: radio/tv, 4: domestic appliance, 5: repairs, 6: education, 7: other, 8: retraining, 9: business
Credit Amount	The total amount of credit requested	(numerical)
Savings Status	Amount of savings held in savings account/bonds	1: <100 DM, 2: $100 \leq X <$ 500 DM, 3: $500 \leq X <$ 1000 DM, 4: $X \geq 1000$ DM, 5: unknown/ no savings account
Employment	Duration of current employment status	1: unemployed, 2: <1 year, 3: $1 \leq X < 4$ years, 4: $4 \leq X < 7$ years, 5: $X \geq 7$ years
Instalment rate in percentage of disposable income	The portion of disposable income that goes towards loan repayment	(numerical)
Personal Status	Personal status and gender of the applicant	1: male div/sep, 2: female div/dep/mar, 3: male single, 4: male mar/wid, 5: female single
Other Parties	Whether there are other debtors or guarantors	1: none, 2: co-applicant, 3: guarantor
Present residence since	The duration of residence at the current address	(numerical)
Property Magnitude	Type of property owned by the applicant	1: real estate, 2: life insurance, 3: car, 4: unknown / no property
Age in years	Age of the applicant in years	(numerical)
Other Payment Plans	Other existing installment plans	1: bank, 2: stores, 3: none
Housing	The housing status of the applicant	1: rent, 2: own, 3: for free
Number of existing credits at this bank	The number of existing credits at this bank	(numerical)
Job	Job status of the applicant	1: unemployed/unskilled - non-resident, 2: unskilled - resident, 3: skilled employee / official, 4: management/ self-employed/ highly qualified employee/ officer
Number of people being liable to provide maintenance for	The number of people the applicant is responsible for financially	(numerical)
Own Telephone	Whether the applicant has a telephone registered in their name	1: none, 2: yes, registered under the customer's name
Foreign Worker	Whether the applicant is a foreign worker	1: yes, 2: no

Table 4: Description of Attributes

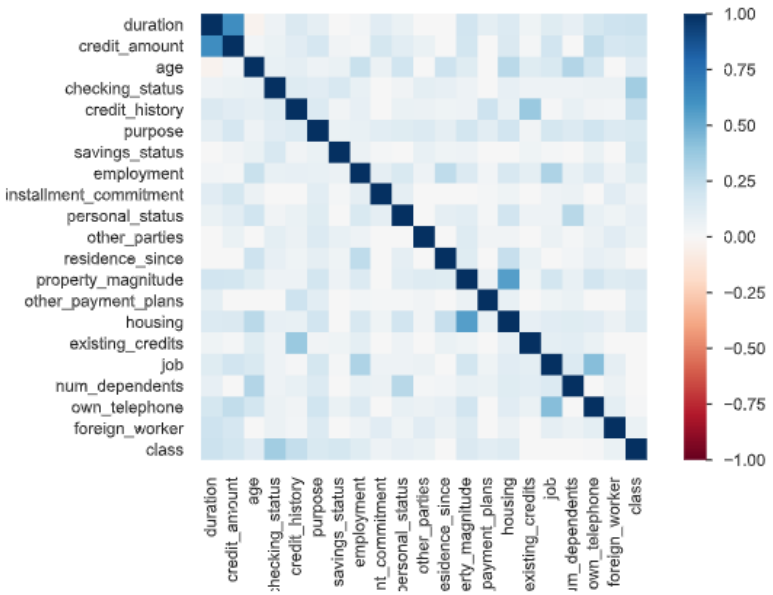

Figure 4: Correlations

Figure 5: Visualization of some features

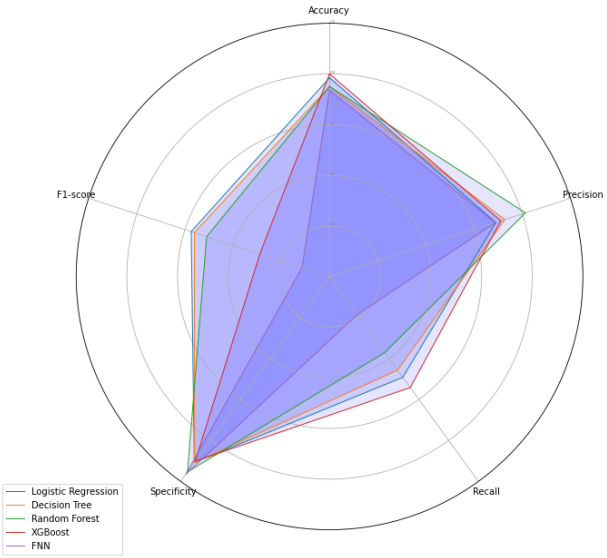

Figure 6: Method Comparisons without SMOTE using a Radar graph

Table 5: Hyperparameters needed to be tuned for different algorithms

Model	**Hyperparameters Needed to be Tuned**
Logistic Regression	-
Decision Trees	Tree-depth
Random Forest	Tree-depth
XGBoost	Tree-depth, Learning-rate, Gamma, subsample, colsample_bytree
FNN (Feed Forward Neural Networks)	Hiddenlayers, Activation function, number of neurons per hidden layer

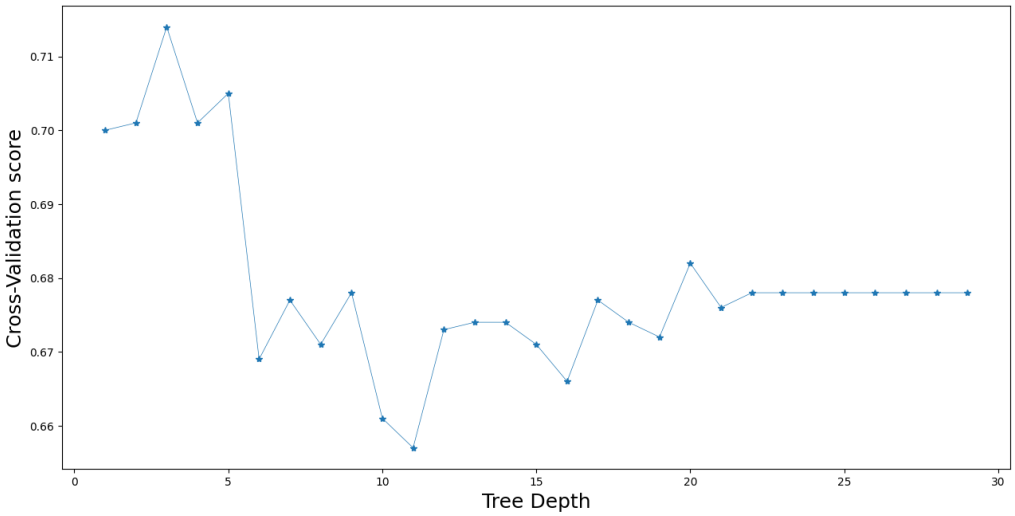

Figure 7: Validation score

Table 6: Model Comparisons with SMOTE

	Logistic Regression							
Threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Accuracy	0.375	0.495	0.580	0.670	0.720	0.755	0.785	0.770
Precision	0.319	0.362	0.403	0.466	0.518	0.574	0.690	0.810
Recall	0.983	0.932	0.881	0.814	0.746	0.661	0.492	0.288
Specificity	0.121	0.312	0.454	0.610	0.709	0.794	0.908	0.972
F1-score	0.481	0.521	0.553	0.593	0.611	0.614	0.574	0.425

	XGBoost							
Threshold	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80
Accuracy	0.540	0.690	0.755	0.785	0.800	0.790	0.765	0.735
Precision	0.385	0.485	0.569	0.643	0.711	0.774	0.833	0.875
Recall	0.932	0.814	0.695	0.610	0.542	0.407	0.254	0.119
Specificity	0.376	0.638	0.780	0.858	0.908	0.950	0.979	0.993
F1-score	0.344	0.348	0.329	0.309	0.288	0.231	0.154	0.076

	FNN							
Threshold	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80
Accuracy	0.325	0.405	0.520	0.620	0.685	0.760	0.790	0.760
Precision	0.304	0.331	0.376	0.431	0.481	0.566	0.644	0.667
Recall	1.000	1.000	0.949	0.898	0.847	0.797	0.644	0.373
Specificity	0.043	0.156	0.340	0.504	0.617	0.745	0.851	0.922
F1-score	0.318	0.332	0.344	0.353	0.356	0.362	0.322	0.210

Python Code

In this section we will provide snippets of the Python code that we have used for the 5 models, note that the code is not complete, because then it would be really long, so we chose to only include the most important/relevant parts.

Logistic regression

```
#Here we split the train and test set and fit a quick logistic regression and
    retrieve the predictions
X_train,X_test,y_train,y_test=
    train_test_split(X,y,test_size=0.20,random_state=42)
#we gonna oversample the train data
smote = SMOTE()
X_train, y_train = smote.fit_resample(X_train, y_train)
logit= LogisticRegression()
logit.fit(X_train, y_train)
# Predicting the model
pred_logit= logit.predict(X_test)
```

Decision Tree and Random Forest

The following snippet contains the code for both the Decision Tree and Random Forest methods. It also contains the code for the cross-validation check, feature importance, balancing techniques and corresponding visualisations.

```
# Setting the seed
np.random.seed(1234)

df = pd.read_csv("data2.csv")

# To test the models with the 'highly correlated' features
# df.drop(["duration", "property_magnitude"], inplace=True, axis=1)

# Splitting the target variable X from the set with features Y
X = df.drop(['class'], axis=1)  # Input features
y = df['class'].values  # True labels

# Initialize and train the classifier
clf = DecisionTreeClassifier(random_state=0)
xdepthlist = []
cvlist = []
tree_depth = range(1, 30)
for d in tree_depth:
    xdepthlist.append(d)
    clf.max_depth = d
    cv = np.mean(cross_val_score(clf, X, y, cv=5, scoring='accuracy'))
    cvlist.append(cv)

# Plot the Cross-Validation score against tree depth
plt.figure(figsize=(15, 10))
plt.xlabel('Tree Depth', fontsize=18, color='black')
plt.ylabel('Cross-Validation score', fontsize=18, color='black')
plt.plot(xdepthlist, cvlist, '-*', linewidth=0.5)
plt.show()
```

```
30
31
32 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
       random_state=None)
33
34 # To balance the data, use the following three balancing techniques
35
36 # Data balancing using undersampling
37 # rus = RandomUnderSampler(random_state=0)
38 # X_train, y_train = rus.fit_resample(X_train, y_train)
39
40 # Data balancing using oversampling
41 # ros = RandomOverSampler(random_state=0)
42 # X_train, y_train = ros.fit_resample(X_train, y_train)
43
44 # Data balancing using SMOTE
45 # smote = SMOTE()
46 # X_train, y_train = smote.fit_resample(X_train, y_train)
47
48
49 # DECISION TREE
50 new_depth = 3
51 clf_decisiontree = DecisionTreeClassifier(random_state=None, max_depth=new_depth)
52 model = clf_decisiontree.fit(X_train, y_train)
53 y_predict = clf_decisiontree.predict(X_test)
54 confusion = confusion_matrix(y_test, y_predict)
55 tree_accuracy = np.trace(confusion) / np.sum(confusion)
56 tn, fp, fn, tp = confusion_matrix(y_test, y_predict).ravel()
57 tree_specificity = tn / (tn+fp)
58
59 # Calculate the different scores. Pos_label 1is 2 (good)
60 precision_DT = precision_score(y_test, y_predict, pos_label='2')
61 recall_DT = recall_score(y_test, y_predict, pos_label='2')
62 f1_DT = f1_score(y_test, y_predict, pos_label='2')
63 predicted_probabilities = model.predict_proba(X_test)[:, 1]
64 true_labels = y_test
65 auc_roc_DT = roc_auc_score(true_labels, predicted_probabilities)
66
67 '''
68 # Figure for the Decision Tree
69 fig = plt.figure(figsize=(25, 20))
70 fig_tree = tree.plot_tree(clf_decisiontree,
71                           feature_names=['checking_status', 'credit_history',
                              'purpose', 'credit_amount',
72                                          'savings_status', 'employment',
                                              'installment_commitment',
                                              'personal_status',
73                                          'other_parties', 'residence_since',
                                              'age', 'other_payment_plans',
                                              'housing',
74                                          'existing_credits', 'job',
                                              'num_dependents', 'own_telephone',
75                                          'foreign_worker'],
76                           class_names=['good', 'bad'],
77                           filled=True)
78 plt.xlabel('Decision Tree', fontsize=18, color='blue')
79 plt.ylabel('Test', fontsize=18, color='blue')
80 plt.show()
```

```
'''

importance = model.feature_importances_
indices = np.argsort(importance)[::-1]

if np.sum(importance) == 0:
    print("Features are of no importance.")
else:
    print("The Feature Importance is:")
    for f in range(X_train.shape[1]):
        print("Feature %d (%f)" % (indices[f], importance[indices[f]]))

importance = importance[indices][::-1]
indices = indices[::-1]

plt.figure(figsize=(10, 6))
plt.barh(range(X_train.shape[1]), importance)
plt.yticks(range(X_train.shape[1]), X_train.columns[indices])
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Feature Importance Decision Tree')
plt.tight_layout()
plt.show()


# RANDOM FOREST MODEL
RF_model = RandomForestClassifier(n_estimators=100, max_features="sqrt",
    random_state=0)
RF_model.fit(X_train, y_train)
RF_model.score(X_train, y_train)
y_predict = RF_model.predict(X_test)

# Calculate scores
RF_confusion = confusion_matrix(y_test, y_predict)
RF_accuracy = np.trace(RF_confusion) / np.sum(RF_confusion)
tn, fp, fn, tp = confusion_matrix(y_test, y_predict).ravel()
RF_specificity = tn / (tn+fp)
precision_RF = precision_score(y_test, y_predict, pos_label='2')
recall_RF = recall_score(y_test, y_predict, pos_label='2')
f1_RF = f1_score(y_test, y_predict, pos_label='2')
predicted_probabilities = RF_model.predict_proba(X_test)[:, 1]
true_labels = y_test
auc_roc_RF = roc_auc_score(true_labels, predicted_probabilities)


print("Accuracy Decision tree:", tree_accuracy, "Accuracy Random Forest:",
    RF_accuracy)
print("Precision Decision tree:", precision_DT, "Precision Random Forest:",
    precision_RF)
print("Recall Decision Tree:", recall_DT, "Recall Random Forest:", recall_RF)
print("F1-Score Decision Tree:", f1_DT, "Recall Random Forest:", f1_RF)
print("Specificity Decision Tree:", tree_specificity, "Specificity Random
    Forest", RF_specificity)
print("AUC-ROC Decision Tree:", auc_roc_DT, "AUC-ROC Random Forest", auc_roc_RF)


importance = RF_model.feature_importances_
indices = np.argsort(importance)[::-1]
```

```
135
136 if np.sum(importance) == 0:
137     print("Features are of no importance.")
138 else:
139     print("The Feature Importance is:")
140     for f in range(X_train.shape[1]):
141         print("Feature %d (%f)" % (indices[f], importance[indices[f]]))
142
143 plt.figure(figsize=(10, 6))
144 plt.barh(range(X_train.shape[1]), importance)
145 plt.yticks(range(X_train.shape[1]), X_train.columns[indices])
146 plt.xlabel('Importance')
147 plt.ylabel('Features')
148 plt.title('Feature Importance Random Forest')
149 plt.tight_layout()
150 plt.show()
151
152 # confusion = confusion_matrix(y_test, y_predict)
153 # print("Confusion Matrix:")
154 # print(confusion)
155
156 # Calculate Mean Squared Error (MSE)
157 # mse = mean_squared_error(y_test, y_predict)
158 # print("Mean Squared Error:", mse)
159
160 # loss = zero_one_loss(y_test, y_predict)
161 # print(loss)
```

XGBoost

For the XGBoost there were 2 important pieces of code, first, we performed a grid search to determine the optimal hyperparameters. And after that, we fitted the test data to get predictions.

```
1 #define training and testing sets (should make a function out of this)
2 x_temp, x_test, y_temp, y_test = train_test_split(X, Y, test_size=0.2,
      random_state=42)
3
4  #we are going to oversample the train data
5 smote = SMOTE()
6 x_temp, y_temp = smote.fit_resample(x_temp, y_temp)
7
8
9 # define the hyperparameter grid for grid search
10 param_grid = {
11     'max_depth': [3, 4, 5],
12     'learning_rate': [0.1, 0.15, 0.2],
13     'gamma': [0, 0.1, 0.2],
14     'subsample': [0.8, 1.0],
15     'colsample_bytree': [0.8, 1.0]
16 }
17
18 #define the XGBoost model
19 xgb_model = xgb.XGBClassifier(objective='binary:logistic',
      eval_metric='logloss', n_jobs=-1)
20
21 #we will now perform the grid search
```

```
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5,
    scoring='accuracy')
grid_search.fit(x_temp, y_temp)

# get the best parameters and the average accuracy
best_params = grid_search.best_params_
best_accuracy = grid_search.best_score_
print("Best Parameters:", best_params)
print("Average Accuracy:", best_accuracy)
#these are the parameters that we are going to use in our final model

#and here we train the XGBoost one last time with the optimal hyperparameters
    determined in the grid search


##########################################
#specify the parameters
params_xgb = {
    'objective': 'binary:logistic',
    'eval_metric': 'logloss',
    'max_depth': 3,
    'learning_rate': 0.1,
    'gamma': 0,
    'subsample': 1,
    'colsample_bytree': 0.8,
    'n_jobs': -1
}

test_result = {}
dtest = xgb.DMatrix(x_test,label=y_test)
dtrain = xgb.DMatrix(x_temp, label=y_temp)
bst = xgb.train(params_xgb, dtrain, num_boost_round=200,evals=[(dtest,
    'testing')],
            early_stopping_rounds=20)
dtest = xgb.DMatrix(x_test, label= y_test)
preds_xgb = bst.predict(dtest)
preds_xgb = pd.DataFrame(preds_xgb, index=y_test.index)
preds_xgb_class = (preds_xgb >= 0.5).astype(int)
```

FNN

We will now provide some snippets of the code that we have used to train a FNN. We first show a snippet of the code that was used to validate the FNN model. Note that in the code below it is not displayed that we have changed the activation function, number of neurons or the number of hidden layers. We purposely did not do a grid search in case of a FNN, because we wanted more control over the results and each time we manually changed a hyperparameter of the FNN. The code below is the optimal configuration.

```
kf = KFold(n_splits=5, random_state=42, shuffle=True)

# Callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
model_checkpoint = tf.keras.callbacks.ModelCheckpoint('best_model.h5',
    monitor='val_loss', save_best_only=True)

accuracies = []
recalls = []

```

```
for train_index, val_index in kf.split(x_temp):
    x_train, y_train = x_temp.iloc[train_index],  y_temp.iloc[train_index]
    x_eval, y_eval = x_temp.iloc[val_index], y_temp.iloc[val_index]

    model = tf.keras.Sequential([
        tf.keras.layers.Dense(64, activation='relu',
            input_shape=(x_train.shape[1],)),
        tf.keras.layers.Dropout(0.1),
        tf.keras.layers.Dense(64, activation='relu',
            input_shape=(x_train.shape[1],)),
        tf.keras.layers.Dropout(0.1),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer='adam', loss='binary_crossentropy',
        metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=50, batch_size=32,
        validation_data=(x_eval, y_eval), callbacks=[early_stopping,
        model_checkpoint])
      # Load best model
    model.load_weights('best_model.h5')

    # Predict on the validation set
    preds_val = model.predict(x_eval)
    preds_val_class = (preds_val >= 0.5).astype(int)

    # Here we compute the accuracy
    accuracy = accuracy_score(y_eval, preds_val_class)
    accuracies.append(accuracy)
    # And we compute the recall for the current run
    recall = recall_score(y_eval, preds_val_class)
    recalls.append(recall)

# Compute average accuracy and recall over all runs
avg_accuracy = sum(accuracies) / len(accuracies)
avg_recall = sum(recalls) / len(recalls)


results_1 = {'Average accuracy': avg_accuracy, 'Average recall': avg_recall}
```

Below we used the optimal configuration determined by k-fold cross-validation to make a final prediction on the test set.

```

#then make one last data set for training and validation purposes:

x_train, x_eval, y_train, y_eval  = train_test_split(x_temp, y_temp,
    test_size=0.2, random_state=42)


# Callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
model_checkpoint = tf.keras.callbacks.ModelCheckpoint('best_model.h5',
    monitor='val_loss', save_best_only=True)

accuracies = []

```

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(x_eval.shape[1],)),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(x_train, y_train, epochs=50, batch_size=32, validation_data=(x_eval, y_eval),
          callbacks=[early_stopping, model_checkpoint])

# Load best model
# model.load_weights('best_model.h5')

preds_fnn = model.predict(x_test)
preds_fnn = pd.DataFrame(preds_fnn, index=y_test.index)

preds_fnn_class = (preds_fnn >= 0.5).astype(int)
```

Feedback draft report Group 2 by Group 1

Introduction

First of all, it would be helpful to include a more comprehensive and detailed discussion of the background and relevance of the problem in the introduction. You start with the claim that the specific issue has a significant impact on the financial sector but can you provide evidence or references for this? Starting the introduction with a paragraph that discusses the motivation for investigating the problem, using a real-life example of banks or other companies that have gone bankrupt as a result of issuing risky loans, could be a good approach. The problem definition is well-defined, specifying the type of problem to be analyzed and indicating the chosen approach for addressing it. Lastly, the citation '(Shi et al.)' misses the year, and 'Kag' as well.

Data Description

Regarding the data description, it is clearly stated why this data is relevant for the goal of this study and how you made the decisions while preparing the data. The table in the appendix with the attributes, description and applied transformations is a valuable addition. It might be a good idea to put Tables 1, 2 and 3 in the appendix and add a table with summary statistics. You also refer to some bar charts in this section but we could not find them in the report. Furthermore the caption of Figure 1 could be improved by including a short description. It is not clear what type of financial institution extends the credit in your data set. Last but not least, it is useful to include more information on the target variable. Please specify the criteria for classifying a risk as high or low in the context of credit risk.

Methods and Validation

Most of the methods used in this study are clearly explained, along with a discussion of why they are appropriate for handling the classification problem. Moreover the advantages and disadvantages of the models are discussed. Only for the random forest method the limitations are not mentioned. The report lacks clarity regarding the specific value of K employed in the K-fold cross-validation and a justification of the chosen value. Furthermore it is not clear which validation score is used when performing cross-validation. Try to define the concepts of true positives, true negatives, false positives and false negatives in terms of your specific research topic (true positive identifies that the model correctly predicts a high-risk). Additionally, it is important to clarify the relevance of the model evaluation metrics in relation to the research topic. The analysis section does discuss the relevance of certain evaluation metrics, although the explanations tend to be repetitive. n the analysis and insights section the threshold value (p) is used. It is better to provide the definition of p at the start of the section. One of the primary concerns of the analysis is the evaluation of the methods. It appears that, for certain methods, the quality of the fit is evaluated using K-fold cross-validation, whereas for other methods, a test set is employed. It could be preferable to utilize the same test approach for all models in order to ensure a fair and consistent evaluation across different models. In addition, it appears that the results for the grid search to determine the optimal hyperparameters of XGBoost are absent. Please also elaborate on why it is potentially important to address the imbalance in the data set. It is perhaps good to include a final section which combines the results of the used models. In the current lay-out metrics regarding the same method are displayed in one table for different values of parameters. For example, the metrics for the logistic regression are combined in a table for different values of the threshold p. But if we want to compare the metrics of the XGBoost method with the logistic regression method we have to shift through different pages to find the appropriate Tables. I think that creating one Table that presents the metrics for the different models enhances the readability of the results. One way to do this is by creating metrics as columns and by vertically moving through the different models.

Logistic Regression

Explanation is informative enough to capture your method. One thing that I would recommend is putting the citation in the consistent format like previously in the introduction. Example:

... classification problems and is well-documented in literature, such as 'The Elements of Statistical Learning' (Hastie, Tibshirani, and Friedman, 2009)...

Model Evaluation

Language is less consistent in this section, also there is a typo 'Decesion tree'. In addition, the Logistic Regression's threshold elaboration is unclear. It is clear that the threshold refers to bad/good credit. However, in what kind of sense? In addition, are the threshold imbalanced?

Conclusion and Further Recommendations

The Conclusion section can perhaps contain a better defined and precise conclusion. It is highlighted that some models perform better on some metrics for some thresholds but a practically relevant advice is not really given. Instead of highlighting that some metrics are important it is perhaps better to explain which metrics are important in which situations. In this way you can give a more precise conclusion by explaining that some models perform better in some situations by referring to the metric that is relevant for a specific situation/preference.

Conclusion Feedback

The report has a good structure and satisfies the page limit. The writing style of the report is mainly professional but there are a few spelling mistakes and punctuation errors that require attention. We strongly advise you to check the references because there are inconsistencies in the referencing style and even some missing references. Furthermore, to improve the presentation of the results, it is recommended to use figures such as a confusion matrix. Overall, from the perspective of a customer who has hired the authors for a project, our level of satisfaction could be considered moderate. The methods and data used in the study are suitable and well-explained. However the problem motivation, discussion of the outcome variable, model evaluation, presentation of the results and estimated gain require additional attention. Therefore we conclude that an appropriate grade for this report is a 5.5.

Feedback on Group 1 Report by Group 2

Problem and Method

The problem of predicting customer churn in the banking industry was clearly defined and well-motivated. The use of Support Vector Machines (SVM) and Synthetic Minority Over-sampling Technique (SMOTE) was justified and well explained. SVM is described in terms of its ability to handle non-linear boundaries using a kernel, specifically the radial basis function (RBF) kernel. The report also discusses the advantages and disadvantages of SVM, including its robustness to outliers and its lack of interpretability. Additionally, it is intriguing that you incorporated regularization into the logistic regression model. The use of SMOTE is presented as a means to improve the performance of the SVM model in predicting customer churn. However, it would be better if the report also talked about other methods like decision trees or neural networks, and why SVM was chosen over them.

Analysis

The analysis was thorough and insightful. The use of cross-validation for hyperparameter tuning in SVM was a strong point. It also compares the results with and without using SMOTE, which gives us a better understanding of how handling class imbalance in the data affects the results. More specifically, it discusses the performance of these models on both the original and SMOTE datasets, using metrics such as Accuracy, F1 Score, Precision, and Recall and provides a detailed analysis of how the performance of these models changes with varying values of C, the inverse of regularization strength, and l1 ratios. It would be nice to maybe explain the various features a little it more in-depth.

Style

The report was well-written and easy to understand. The use of tables and graphs was effective in visualizing the results and the impact of different hyperparameters. Also, you included a lot of references which is very nice. A nice addition would be to use heatmaps in the presentation of the SVM cross-validated F1-score in tables.There is also a drawback, the report lacks a conclusion where the main findings, the implications of the results, and suggest directions for future research should be summarized.That makes it difficult for the reader to understand the overall significance of the work.

Overall Grade: Good (7/10)

On the one hand, the report was well executed and provided valuable insights into customer churn prediction using SVM and SMOTE. On the other hand, there are areas where it could be better as we explained above. We would consider hiring them again given the quality of the work, but would expect these improvements to be addressed.