
Performance assessment of credit risk models
with boosting algorithms and transfer learning
from Large Language Models

Evaluación del rendimiento de modelos de riesgo
crediticio con algoritmos de boosting y transfer
learning sobre modelos grandes de lenguaje



Trabajo de Fin de Grado
Curso 2022–2023

Autor

Mario Sanz Guerrero

Directores

Dr. Javier Arroyo Gallardo

Antonio Caparrini López

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Performance assessment of credit risk
models with boosting algorithms and
transfer learning from Large Language
Models

Evaluación del rendimiento de modelos de
riesgo crediticio con algoritmos de
boosting y transfer learning sobre modelos
grandes de lenguaje

Trabajo de Fin de Grado en Ingeniería Informática

Autor

Mario Sanz Guerrero

Directores

Dr. Javier Arroyo Gallardo

Antonio Caparrini López

Convocatoria: *Junio 2023*

Calificación: *10 - Matrícula de Honor*

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

29 de mayo de 2023

Agradecimientos

Me gustaría empezar agradeciendo el excelente trabajo de Javier y Antonio como directores de este trabajo. De vosotros me llevo grandes aprendizajes, y no solo en aspectos técnicos, sino también en lo personal. Gracias por vuestro tiempo y vuestra paciencia, pero sobre todo, gracias por vuestra pasión.

Gracias también a mis amigos de la Facultad. Sin vosotros, este precioso camino no hubiera sido lo mismo.

A mi padre y mi madre, por enseñarme a vivir siendo quien soy. Gracias por vuestros valores.

A Lucía, mi hermana, por tu apoyo incondicional en todo momento y tu alegría cuando más la he necesitado.

A Nadia, mi compañera de vida, por tu amor y paciencia. Gracias por ser y estar.

Por último, gracias a mis abuelos. Sé que estáis muy orgullosos de mi trabajo.

Resumen

Evaluación del rendimiento de modelos de riesgo crediticio con algoritmos de boosting y transfer learning sobre modelos grandes de lenguaje

El objetivo de este Trabajo de Fin de Grado es explorar el potencial de las técnicas de aprendizaje automático y aprendizaje por transferencia en el análisis del riesgo crediticio en préstamos entre particulares (P2P). La ausencia de un intermediario financiero tradicional en los préstamos P2P genera una significativa asimetría de información, lo que aumenta el riesgo. Una forma de mitigar este riesgo es predecir correctamente si el préstamo será o no devuelto. Sin embargo, al momento de otorgar el préstamo, la información disponible es limitada. En este estudio, se propone una aproximación novedosa mediante el uso de la descripción proporcionada por el prestatario, la cual consiste en texto libre sin estructura. Para aprovechar esta información e incorporarla en un algoritmo de clasificación tradicional, utilizaremos aprendizaje por transferencia con redes neuronales profundas, específicamente, el modelo BERT, un modelo grande de lenguaje (*Large Language Model*) desarrollado por Google en 2018 que se utiliza ampliamente en tareas de clasificación.

Este estudio se fundamenta en los resultados de investigaciones recientes que emplean técnicas avanzadas de aprendizaje automático, como algoritmos de *gradient boosting*, optimización de hiperparámetros mediante algoritmos genéticos y técnicas de IA explicables, para analizar el papel de las variables de entrada. A estos modelos se les añadirá como variable de entrada una puntuación generada por el modelo BERT, la cual indica la probabilidad de impago basándose en la descripción del préstamo.

Nuestro trabajo demuestra que las descripciones contienen información útil para predecir el impago, y que su inclusión mejora significativamente el rendimiento de los modelos de otorgamiento de créditos.

Palabras clave

Riesgo crediticio, préstamos P2P, aprendizaje automático, gradient boosting, IA explicable, PLN, Large Language Models, BERT, transfer learning.

Abstract

Performance assessment of credit risk models with boosting algorithms and transfer learning from Large Language Models

The objective of this Bachelor's Thesis is to explore the potential of machine learning techniques and transfer learning in the analysis of credit risk in peer-to-peer (P2P) lending. The absence of a traditional financial intermediary in P2P lending creates a significant information asymmetry, which increases the risk. One way to mitigate this risk is to accurately predict whether the loan will default or not. However, at the time of granting the loan, the available information is limited. In this study, a novel approach is proposed using the borrower's description, which consists of unstructured free text. To leverage this information and incorporate it into a traditional classification algorithm, we will use transfer learning with deep neural networks, specifically the BERT model, a Large Language Model developed by Google in 2018 that is widely used in classification tasks.

This study builds on the findings of recent research that employ advanced machine learning techniques, such as gradient boosting algorithms, hyperparameter optimization using genetic algorithms, and explainable AI techniques, to analyze the role of input variables. These models will be supplemented with an input variable representing a score generated by the BERT model, which indicates the probability of default based on the loan description.

Our work demonstrates that descriptions contain useful information for predicting default, and their inclusion significantly improves the performance of credit-granting models.

Keywords

Credit risk, P2P lending, machine learning, gradient boosting, explainable AI, NLP, Large Language Models, BERT, transfer learning.

Contents

1. Introduction	1
1.1. Motivation	1
1.2. State of the Art	2
1.3. Objectives	3
1.4. Work plan	5
2. Theoretical Foundations	7
2.1. A brief introduction to Machine Learning	7
2.2. Classification algorithms: from decision trees to gradient boosting . .	8
2.3. XGBoost and CatBoost	9
2.4. Addressing classification problems	12
2.5. Hyperparameter tuning using Genetic Algorithms	14
2.6. Natural Language Processing with Large Language Models and transfer learning	17
2.7. Explaining machine learning models using the SHAP values	24
3. Exploration of Data and Experimental Methodology	25
3.1. Dataset Description	25
3.2. Description of the experiment	39
3.3. Baseline classifiers	45
4. Analysis of the results of the boosting algorithms	49
4.1. Quantitative variables	49
4.1.1. XGBoost	49
4.1.2. CatBoost	51
4.2. Quantitative and categorical variables	53
4.2.1. XGBoost	53
4.2.2. CatBoost	55
4.3. Conclusions	58
5. Processing textual features with transfer learning from the BERT model	59

5.1.	Default prediction using BERT	60
5.1.1.	Training the BERT model	60
5.2.	Analysis of BERT predictions	63
5.3.	Explainability of textual information	67
5.3.1.	Exploring the text descriptions with BERTopic	67
5.3.2.	Relation between the quantitative and categorical variables and the BERT prediction	71
6.	Analysis of the results of the boosting algorithms with the BERT scores	75
6.1.	Quantitative, categorical and BERT variables	75
6.1.1.	XGBoost	75
6.1.2.	CatBoost	77
6.2.	Analysis of the impact of textual variables	79
6.3.	Conclusions	81
7.	Conclusions and Future Work	83
7.1.	Discussion of results	83
7.2.	Conclusions	84
7.3.	Future work	86
	Bibliography	89
	A. Genetic exploration analysis	93
	B. Assessment of elitism in the search for the best classifier	99

List of figures

2.1. Confusion matrix diagram.	13
2.2. Gene, chromosome, and population example.	14
2.3. One point crossover.	15
2.4. Two point crossover.	16
2.5. LSTM network diagram.	21
2.6. Transformer network diagram.	22
3.1. Default <i>vs</i> Non-Default distribution.	26
3.2. Loan amount distribution.	28
3.3. Debt-to-Income distribution.	28
3.4. FICO Score distribution.	29
3.5. Revenue distribution.	30
3.6. Revenue distribution without outliers.	30
3.7. Loan amount <i>vs</i> revenue scatter plot.	31
3.8. Employment length distribution.	32
3.9. Employment length default distribution.	32
3.10. Purpose distribution.	33
3.11. Purpose default distribution.	34
3.12. Home ownership distribution.	34
3.13. Home ownership default distribution.	35
3.14. Address state distribution.	35
3.15. Address state default distribution.	36
3.16. Description length distribution.	37
3.17. Stratified K -fold cross-validation diagram.	41
4.1. XGBoost feature importance (quantitative variables).	50
4.2. XGBoost SHAP values (quantitative variables).	51
4.3. CatBoost feature importances (quantitative variables).	52
4.4. CatBoost SHAP values (quantitative variables).	53
4.5. XGBoost feature importances (quantitative and categorical variables).	54
4.6. XGBoost SHAP values (quantitative and categorical variables).	55
4.7. CatBoost feature importances (quantitative and categorical variables).	56

4.8.	CatBoost SHAP values (quantitative and categorical variables).	57
5.1.	Diagram of the BERT neural network architecture.	61
5.2.	BERT training diagram.	62
5.3.	<i>BERT_pred</i> distribution.	64
5.4.	Default/Non-Default proportion by <i>BERT_pred</i> range.	64
5.5.	<i>BERT_pred</i> boxplot by purpose.	73
6.1.	XGBoost feature importances (quantitative, categorical and <i>BERT_pred</i> variables).	76
6.2.	XGBoost SHAP values (quantitative, categorical and <i>BERT_pred</i> variables).	77
6.3.	CatBoost feature importances (quantitative, categorical and <i>BERT_pred</i> variables).	78
6.4.	CatBoost SHAP values (quantitative, categorical and <i>BERT_pred</i> variables).	78
6.5.	BAC improvements by <i>purpose</i> after adding <i>BERT_pred</i> .	79
6.6.	Average description length by purpose.	81
A.1.	Fitness evolution.	94
A.2.	Hyperparameter exploration.	95
A.3.	Top-50 individuals hyperparameters.	96
A.4.	Top-50 individuals scores.	97
B.1.	Fitness evolution.	100
B.2.	Hyperparameter exploration.	101
B.3.	Top-50 individuals hyperparameters.	102
B.4.	Top-50 individuals scores.	103

List of tables

2.1.	XGBoost hyperparameter meaning.	11
2.2.	CatBoost hyperparameter meaning.	12
3.1.	Variables summary.	27
3.2.	Ranges of the XGBoost hyperparameters tuned.	42
3.3.	Ranges of the CatBoost hyperparameters tuned.	43
3.4.	Execution times of the boosting algorithms.	45
3.5.	XGBoost default parameters.	46
3.6.	XGBoost default performance.	47
3.7.	CatBoost default parameters.	47
3.8.	CatBoost default performance.	47
4.1.	XGBoost scoring metrics (quantitative variables).	50
4.2.	CatBoost scoring metrics (quantitative variables).	52
4.3.	XGBoost scoring metrics (quantitative and categorical variables).	54
4.4.	CatBoost scoring metrics (quantitative and categorical variables).	56
4.5.	Scoring metrics summary: XGBoost <i>vs</i> CatBoost.	58
5.1.	BERT optimal configuration by fold.	63
5.2.	Top-5 descriptions with lower <i>BERT_pred</i> values.	65
5.3.	Top-5 descriptions with higher <i>BERT_pred</i> values.	66
5.4.	<i>BERT_pred</i> metrics with 0.5 threshold.	66
5.5.	Top 3 topics' Pearson residuals by <i>purpose</i>	70
5.6.	Top 3 topics' Pearson residuals for <i>emp_length</i>	71
5.7.	Top 3 topics' Pearson residuals for <i>home_ownership</i>	71
5.8.	Top 3 topics' Pearson residuals for <i>addr_state</i>	71
5.9.	Correlation coefficients of quantitative variables with <i>BERT_pred</i>	72
5.10.	ANOVA test of categorical variables with <i>BERT_pred</i>	73
6.1.	XGBoost metrics for the different variable sets.	76
6.2.	CatBoost metrics for the different variable sets.	78
6.3.	Scoring metrics with quantitative, categorical and <i>BERT_pred</i>	81
7.1.	Experiments' results summary.	83

Introduction

1.1. Motivation

Credit risk is the potential loss incurred by an economic agent, the lender of the credit, as a result of total or partial failure on obligations by its counterpart, the borrower of the credit. This is a natural risk in the economic activity, which mainly consists of savings collection and credit supply by a financial institution. In the economic literature, whenever a credit does not meet its obligations correctly at the due date it is said to be delinquent or default¹.

Loans can also be granted between private individuals. This is called peer-to-peer (P2P) lending, and gained popularity in the decade of the 2000's with the rise of internet access and developing markets. One of the main advantages of P2P lending is the absence of the need for a traditional financial intermediary, thus reducing costs in the exchange (Guo et al., 2016). The main downside of this absence is that it implies a loss of information that financial experts have, which aggravates the severe information asymmetry² in the P2P market (Dorfleitner et al., 2016; Michels, 2012; Lin et al., 2013). As a result, it becomes crucial to employ accurate predictive models.

Predicting default in credit lending is a crucial aspect for assessing creditworthiness and credit risk. However, this is a very hard activity due to the high number of aspects that involve credit repayment.

Creditworthiness is evaluated based on various factors, which include income, loan amount, debt-to-income ratio, employment history, and credit scores like FICO, among others. These factors constitute structured data that is used by statistical and machine learning models to assess credit eligibility.

In addition, P2P loan applications often require borrowers to provide a free-text

¹In the context of credit risk, “*default*” refers to the failure of a borrower to repay their debt obligations on time or in full, as agreed upon in the loan agreement.

²Asymmetric information in P2P lending refers to a situation where one party in a transaction, typically the borrower, has more information about their creditworthiness than the other party, typically the lender. This information asymmetry can lead to adverse selection and moral hazard problems.

description explaining the intended purpose of the loan. Merrill Lynch³ affirmed in 1998 that up to 80% of the data in companies are unstructured. However, there is currently no benchmark study to assess the choice of models for processing text in credit scoring (Kriebel and Stitz, 2022), since most datasets used in credit scoring literature do not contain large amounts of textual data points (Kraus et al., 2020).

The motivation behind this work is to explore in credit risk the potential of combining state-of-the-art machine learning classification algorithms together with deep learning Large Language Models for Natural Language Processing, building upon the advancements made in recent research papers (see State of the Art). This will be done by prioritizing explainable AI techniques to understand what are the risk determinants and to incorporate as much transparency as possible in the decision process.

1.2. State of the Art

Many studies about credit risk in the P2P market have been published in recent years comparing different machine learning techniques. For example, Malekipirbazari and Aksakalli (2015) compare the performance of Random Forests against k-NN, SVM and logistic regression. Duan (2019) compares the use of neural networks against logistic regression, decision trees and SVM. The first author to propose a boosted tree in the P2P lending market was Xia et al. (2017). Ensemble techniques have also been studied. For instance, Li et al. (2018) propose an ensemble of XGBoost, neural networks and logistic regression to create an accurate but poorly interpretable model.

Several papers have also studied the treatment of imbalanced classes in the P2P lending market (Boiko Ferreira et al., 2017; Namvar et al., 2018) and hyperparameter optimization via genetic algorithms (Yang et al., 2019).

On the importance of variables, Lin et al. (2016) proved that monthly income, loan amount, debt-to-income ratio, working years and FICO are some of the most relevant factors that impact loan default.

The work by Ariza-Garzón et al. (2020) demonstrates the utilization of machine learning models and incorporates explainable AI techniques, specifically SHAP values (Lundberg and Lee, 2017), for comprehensive variable analysis. Additionally, it addresses the challenges associated with unbalanced classes and enhances model performance through the application of genetic algorithms. We will consider this work as a valuable reference for our research, serving as the foundational baseline for this Bachelor's thesis.

However, most works on credit risk assessment for P2P lending only consider hard information⁴. Recent studies have explored the use of soft information coming from loan description factors (Wang et al., 2019). The loan description typically includes some background information about the borrower, its debt status, the purpose of the loan, etc. (Jiang et al., 2017; Wang et al., 2016), so by processing it, the information asymmetry can be smoothed. The work by Dorfleitner et al. (2016) concluded that

³Investment management company from USA.

⁴Objective, verifiable and reliable data. Refers to quantitative and categorical variables.

information obtained from processing the description becomes especially useful when some hard information is missing, which entails that the knowledge gained from the natural language is redundant with other variables.

Regarding textual feature extraction in credit risk, the work of Kim and Cho (2018) propose the use of deep dense convolutional neural networks (CNN), while the later work by Zhang et al. (2020) explores the use of the Transformer architecture to process textual information, and compares it against the use of convolutional neural networks. This work concludes that the Transformer slightly outperforms CNN. However, the model proposed in this study was based on a neural network that takes as inputs both the hard features and the processed textual features, resulting in poorly explainable results.

Recent studies have compared state-of-the-art deep learning methods. Kriebel and Stitz (2022) compared several deep learning models, such as convolutional, recurrent, convolutional recurrent, average embedding, BERT⁵, and RoBERTa⁶ against traditional machine learning methods and rule-based approaches. They concluded that deep learning techniques are the ones with higher performance, especially those based on Transformers.

Stevenson et al. (2021) investigated loan default classification and compared the effectiveness of the BERT Large Language Model with traditional methods such as Logistic Regression (LR) and Random Forests (RF) for text processing. The BERT approach involved inputting the text representation from the BERT encoder, along with quantitative and categorical variables, into a neural network. Conversely, LR and RF utilized TF-IDF techniques to generate representation vectors of the text as input for the neural network. The results revealed that text alone exhibits surprising efficacy in predicting defaults. However, when combined with traditional hard data, its additional effectiveness is dependent on the length of the text, since longer texts offer more detailed information that is non-redundant with other variables.

Our work differs from the latter in terms of explainability and model combination. In the work of Stevenson et al. (2021), the authors employed a neural network to classify loan default by studying different ways of preprocessing the textual description. In our study, we apply the preprocessing method that yielded the highest performance (BERT). With BERT, we predict the default probability based on the description and incorporate it as a quantitative variable in the boosting algorithms. This approach allows us to obtain interpretable results, and we hypothesize that model combination will further improve our results.

1.3. Objectives

The present work is intended to improve the prediction of credit default by using natural language data. The final goal is to classify new, unseen credit loans in default or non-default by training boosting algorithms with data about previous and finished loans. We will use evolutionary algorithms to achieve the best possible

⁵Bidirectional Encoder Representations from Transformers. BERT is explained in more detail in section BERT model.

⁶Robustly optimized BERT pretraining approach.

classifiers by tuning their hyperparameters.

Firstly, we will compare the XGBoost and CatBoost algorithms. The native handling of categorical variables of CatBoost will be assessed since our dataset contains a high number of categorical variables and we believe that XGBoost could underperform due to the one-hot encoding. Furthermore, these algorithms will be fine-tuned using genetic algorithms to find the best hyperparameters. This way, we will evaluate the importance of hyperparameter optimization in the performance of the algorithms. The top-performing classifiers found in the genetic search will be compared with each other to determine whether different learning patterns achieve similar results.

We will also delve into Natural Language Processing (NLP) techniques to gain information from the loan descriptions that borrowers provide when applying for a loan. This way, we will be able to make more accurate predictions of credit return. Based on the conclusions of Stevenson et al. (2021) and Kriebel and Stitz (2022), we will be using the pre-trained Large Language Model BERT, which requires transfer learning to be fine-tuned for our specific problem. In addition, currently there is a lack of studies concerning the neural network configuration of BERT. Therefore, we will be assessing the effect of an optimal network architecture.

Finally, since current NLP techniques are considered black-box models, we will try to explain the knowledge that the loan descriptions are providing by thoroughly analyzing its results. Moreover, we will apply clustering techniques to the descriptions to group them into topics and study their relationships with other variables and with the delinquency of the loan.

We will be using explainable AI (XAI) techniques in the analysis of the results to gain a comprehensive and insightful understanding of the way these algorithms make their decisions. In this manner, we will be able to analyze how the best classifiers are working and try to open to some extent the black-boxes of NLP.

Therefore, this work aims to build a loan granting model that assesses the eligibility for loan approval based on the information available at the time of the loan request. Building upon the previous study by Ariza-Garzón et al. (2020) focused on AI explainability in P2P credit risk models, we will expand the research by incorporating the following characteristics:

- Comparison of the boosting algorithms XGBoost and CatBoost. Since the latter one is specifically developed for handling natively categorical variables, we will be comparing their performance when dealing with categorical data. Moreover, we will assess their performance improvement when optimizing their hyperparameters using genetic algorithms.
- Study of the effect of hyperparameter tuning using genetic algorithms: Comparison with the default algorithm configurations and comparison of the parameters of the top performing classifiers. This way, we will be able to analyze if there is any difference between their behavior and thus, in their learning process.
 - Exploration of the balanced accuracy scoring metric as an alternative to the AUC in the genetic algorithm's fitness function due to the problem

explained in section *On the use of AUC as fitness measure*.

- Addition of natural language data to assess the performance of NLP techniques in P2P lending problems. This will be done by using transfer learning from the BERT Large Language Model.
- Assessment of the effectiveness of parameter tuning in the last layers of the BERT neural network.
- Explainability of the knowledge gained from the NLP techniques by a thorough analysis of the results. Study of the redundancy of this information gained with other variables.

1.4. Work plan

The present work was started in July 2022 and has undergone continuous development until its submission date in May 2023. From the outermost point of view, the workload has been divided into the following main stages.

1. Summer months (July 2022 - September 2022)

During this stage, the workload was relatively light, averaging about 7 hours per week. The main tasks of this stage include the study of the algorithms used, from basic tree-like algorithms to gradient boosting algorithms. As I was new to the world of financial credits, I familiarized myself with the LendingClub dataset and the meanings of different variables. This also required reading and studying recent papers on P2P lending.

2. First course semester (September 2022 - January 2023)

During these months, the working time increased to an average of 10 hours per week. This stage focused on implementing the boosting algorithms pipeline and analyzing their results using accurate plots and statistical tests. Also, the experiments of this work were conducted on three different remote machines, which required appropriate configuration to run the algorithms on GPUs.

3. Second course semester (January 2023 - May 2023)

This stage was the most productive, with approximately 20 hours of work per week. During these months, I delved into studying NLP techniques employed in this work, such as the Transformer model, Large Language Models, and transfer learning methods. Additionally, I implemented the BERT pipeline and analyzed the results obtained by adding the processing of textual information to the boosting algorithms. I also applied explainability techniques to the insights derived from textual information processing. Finally, this thesis report was written by selecting the most significant results and analyzing the conclusions drawn from each experiment.

Theoretical Foundations

2.1. A brief introduction to Machine Learning

Machine Learning is a field of Artificial Intelligence that focuses on the development of algorithms and statistical models that can perform tasks without explicit instructions. These algorithms automatically improve their performance based on experience, allowing them to learn from data inputs and make predictions or decisions without being specifically programmed for a specific task. Machine learning can be applied in various domains such as image recognition, natural language processing, recommender systems, and predictive modeling, among others.

Unsupervised learning is a type of machine learning where the algorithm is trained on an unlabeled dataset and the goal is to find patterns or relationships in the data without the use of pre-existing labels. Unsupervised learning tasks include, mainly, clustering, dimensionality reduction, and anomaly detection.

On the other hand, supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning that the correct output is provided for each input data sample. The goal of supervised learning is to learn a mapping function that can generalize to unseen data and accurately predict the output. This approach is useful when the desired outcome is known and there is a clear relationship between input and output data. There are two main types of supervised learning: classification and regression. In classification, the algorithm is trained to predict a categorical output, such as classifying an email as spam or not spam. In regression, the algorithm is trained to predict a continuous output, such as predicting the price of a house based on its features.

In our project, we will be using supervised learning with a classification approach because our goal is to predict the default of credit loans. We will be provided with a labeled dataset that includes information about credit loans and whether they defaulted or not. By training a machine learning algorithm on this labeled data, we will be able to learn the relationship between the input variables (such as loan amount, borrower's annual income, etc.) and the binary output variable (default or not default). The resulting model can then be used to predict the default status of new credit loans based on the input features.

However, it is important to note that the approach we will be taking for this

classification problem can easily be adapted to a regression problem. If our goal was to predict the amount of money that a credit loan would default, instead of just predicting if it would default or not, we could simply modify our algorithm to use a regression approach instead of a classification approach.

2.2. Classification algorithms: from decision trees to gradient boosting

Decision Tree

First proposed by Quinlan (1986), a decision tree is a type of algorithm commonly used for classification in machine learning. It is a tree-like model that makes predictions by following a series of decisions, with each internal node representing a test of an input feature and each leaf node representing a prediction.

The algorithm builds the tree by dividing the input features into nodes and then using an information gain metric to determine the optimal splits. At each internal node, the algorithm selects the feature and split point that results in the greatest information gain, reducing uncertainty about the class label of the instances in the region. This process is repeated recursively until some stopping criteria are met, such as a minimum number of samples in a region or a maximum tree depth. The final decision tree can be used to make predictions for new instances by following the path from the root to a leaf node and returning the class label associated with that leaf node.

Decision trees are simple to understand, interpret, and visualize, making them a popular choice for many classification problems. Moreover, they are fast both to train and to make predictions, can handle continuous or categorical data, and are an explainable model. However, this technique could easily lead to an overfitting problem.

Ensemble learning decision-tree approaches

Ensemble learning is a machine learning technique that combines multiple models to create a more powerful and accurate model. The idea behind it is to combine the strengths of multiple models to reduce the weaknesses of each of them and improve the overall performance.

Bagging, or bootstrap aggregating, is a popular ensemble learning technique introduced by Breiman (1996) that uses bootstrap sampling¹ to create multiple versions of the same model. Each of these versions is trained in parallel on a different bootstrapped dataset, and the results of all the models are combined to produce a single prediction. The aim of bagging is to reduce the variance of a model by

¹Sampling is the process of selecting a subset of a population to estimate population parameters and make inferences, as collecting data from the entire population may not be feasible. Bootstrapping involves generating multiple samples from a dataset to estimate the distribution of a statistic or creating multiple training sets in machine learning.

averaging the predictions of multiple models trained on different samples of the data.

Boosting is another popular ensemble learning technique proposed by Freund and Schapire (1999) that focuses on improving the performance of weak models. Unlike bagging, boosting trains multiple models sequentially, where each model is trained to correct the mistakes of the previous model. The aim of boosting is to reduce the bias of a model by combining multiple models trained on different samples of the data.

Random Forest

Initially proposed by Breiman (2001), Random Forest is an ensemble learning method for classification and regression that uses bagging as the ensemble method and decision trees as the individual models trained in parallel.

The main idea behind random forest is to combine multiple decision trees to create a more robust and accurate model. In a random forest each tree is trained on a different random subset of the training data, and the final prediction is made by aggregating the predictions of all the trees, typically through a majority vote in the case of classification or by averaging in the case of regression. The use of multiple trees helps to mitigate the overfitting problem that can occur when using a single decision tree, as the errors made by individual trees are averaged out. The randomization in the training data and features also helps to reduce overfitting and improve the generalization performance of the model.

Gradient Boosting

On the other hand, Gradient Boosting, introduced by Friedman (2001), is a type of boosting algorithm, which implies that it uses a sequential ensemble learning method, where each model attempts to correct the mistakes of the previous model. It is often implemented using decision trees as the base models, although other base models can also be used, such as linear models or neural networks.

Gradient Boosting works by using a loss function to evaluate the performance of the model on the training data, and then using the gradient of this loss function to update the model in a direction that reduces the loss. This process is repeated iteratively, adding a new model to the ensemble on each iteration. The final prediction is made by combining the predictions of all the weak models through a weighted majority vote for classification or by a weighted sum in the case of regression.

Gradient Boosting has several implementations, each with its own strengths and weaknesses. We will now cover the most relevant ones for this Bachelor's thesis, which are XGBoost and CatBoost.

2.3. XGBoost and CatBoost

In our project, we will be using these two gradient boosting algorithms to predict loan default. Our initial intention was to use XGBoost as in the work of Ariza-Garzón et al. (2020). However, the high amount of categories in some of the

categorical variables make us suspect that XGBoost can underperform. To test this, we will compare the performance of XGBoost against CatBoost, and we will analyze the impact of the addition of categorical variables on each of them.

XGBoost

Extreme Gradient Boosting (XGBoost) is an implementation of gradient boosting that has become popular due to its strong performance and efficiency. It was initially introduced by Chen and Guestrin (2016), and it is maintained by the DMLC community as an open source project under the Apache 2.0 software license.

Regarding the tree construction, XGBoost builds an asymmetric tree that is grown level-wise and depth-wise. This means that the tree is grown by adding one level at a time and growing the nodes within that level to their full depth before moving on to the next level. This method is efficient and allows XGBoost to handle large datasets and to scale well.

While numerical variables are supported natively in XGBoost, this is not the case for categorical variables. For XGBoost to be able to handle them we need to apply one-hot encoding.

One-hot encoding is a variable preprocessing technique that assigns a binary value to each category of each categorical variable, where the column has a value of 1 if the sample belongs to that category, and 0 otherwise. The downside of this is that it greatly increases the total number of variables. This number will be given by:

$$\#Total\ var. = \#Quant.\ var. + \sum_{i=1}^{\#Categ.\ var.} \#Categories_i$$

One-hot encoding has another drawback: it fails to capture the mutual exclusivity of the encoded categories, resulting in a loss of information in the binary variables.

Moreover, XGBoost does not support missing values, which require manual imputation. A very interesting feature is that it supports running in GPU, which considerably accelerates the training process.

Its most relevant hyperparameters can be seen in Table 2.1.

CatBoost

CatBoost is another gradient boosting algorithm, similar to XGBoost, that is specifically designed to handle natively categorical variables and missing values. This makes CatBoost a suitable choice for loan default prediction problems as ours, where the loan applicants' information contains categorical variables. It was first presented by Prokhorenkova et al. (2017), and it is maintained by Yandex as an open source project under the Apache 2.0 software license.

As for the tree construction, CatBoost builds a symmetric tree. The tree is grown in a balanced way, with nodes being split into two parts at each level. In addition, the splitting condition at a certain level is the same for all the nodes in that level, resulting in a more balanced tree structure.

Parameter	Meaning
ETA (learning rate)	Step size shrinkage. Makes the model more robust by shrinking the weights on each boosting step.
<i>max_depth</i>	Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit.
<i>gamma</i>	Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger <i>gamma</i> is, the more conservative the algorithm will be.
<i>min_child_weight</i>	Minimum sum of instance weight needed in a child to keep partitioning the tree. The larger <i>min_child_weight</i> is, the more it will contribute to overfitting avoidance.
<i>alpha</i>	L1 regularization term on weights. Increasing this value will make the model more conservative.
<i>lambda</i>	L2 regularization term on weights. Increasing this value will make the model more conservative.
<i>subsample</i>	Fraction of observations (rows) to be randomly sampled for each tree. Lower values prevent overfitting, but too small values might lead to underfitting.
<i>colsample_bytree</i>	Fraction of columns (features) to be randomly sampled for each tree.
<i>scale_pos_weight</i>	Control the balance of positive and negative weights, useful for unbalanced classes.
<i>n_estimators</i>	Number of boosting rounds.

Table 2.1: XGBoost hyperparameter meaning.

It supports numerical variables and categorical variables natively without the need for preprocessing. It internally performs a one-hot encoding, which allows it to be more efficient than the one needed in XGBoost. According to the official documentation, this internal handling of categorical variables enhances training speed and resulting quality.

CatBoost does support missing values management for numerical features. By default, CatBoost will assign the minimum value of that feature in the dataset, although the maximum value assignment can be configured. As well as XGBoost, CatBoost also supports GPU training to accelerate this process.

We will now see that some hyperparameters of CatBoost differ from the ones in XGBoost. For instance, CatBoost does not have a *gamma* nor a *min_child_weight* equivalent parameter since CatBoost builds symmetric trees. Moreover, CatBoost does not support L1 regularization nor *colsample_bytree*, but it includes other parameters regarding feature selection such as *random_strength*. Its most relevant hyperparameters can be seen in Table 2.2.

Parameter	Meaning
<i>learning_rate</i>	Used for reducing the gradient step. Similar to <i>eta</i> in XGBoost.
<i>depth</i>	Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. Similar to <i>max_depth</i> in XGBoost.
<i>l2_leaf_reg</i>	Coefficient at the L2 regularization term of the cost function. Similar to <i>lambda</i> in XGBoost.
<i>bagging_temperature</i>	Defines the settings of the Bayesian bootstrap. The higher the value the more aggressive the bagging is.
<i>random_strength</i>	Amount of randomness to use for scoring splits when the tree structure is selected. Its use can avoid overfitting.
<i>scale_pos_weight</i>	Controls the balance of positive and negative weights, useful for unbalanced classes. Similar to <i>scale_pos_weight</i> in XGBoost.
<i>iterations</i>	The maximum number of trees that can be built when solving machine learning problems. Similar to <i>n_estimators</i> in XGBoost.

Table 2.2: CatBoost hyperparameter meaning.

2.4. Addressing classification problems

In this section, we will delve into the final machine learning concepts necessary for a comprehensive understanding of the experiment’s hands-on explanation.

Scoring Metrics

In order to understand scoring metrics, we should first delve into the concept of confusion matrix. A confusion matrix is a performance measurement tool used for evaluating classification algorithms. It summarizes correct and incorrect predictions in a table, and helps to identify the type of errors that the algorithm is making. The matrix has size $n \times n$, where n is the number of classes, and cells represent the following values:

- True Positives (TP): Number of instances that are correctly classified as positive.
- False Positives (FP): Number of instances that are incorrectly classified as positive.
- False Negatives (FN): Number of instances that are incorrectly classified as negative.
- True Negatives (TN): Number of instances that are correctly classified as negative.

A perfect classification algorithm would have all its instances in the main diagonal, that is, distributed among TP and TN. Figure 2.1 shows a diagram of a confusion matrix.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 2.1: Confusion matrix diagram.

A scoring metric in machine learning is a method of evaluating the performance of a model by comparing its predictions with the actual target values. These metrics provide a numerical value that quantifies the quality of the model's predictions, allowing for a more comprehensive evaluation and comparison of different models. Of all the existing scores, we will use those defined as follows:

- Accuracy: Measures the proportion of correct predictions made by the model out of all predictions made.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: Measures the proportion of true positive predictions among all positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: Measures the proportion of true positive predictions made by the model to all actual positive instances in the dataset. It is usually also called Sensitivity or True Positive Rate (TPR).

$$Recall = \frac{TP}{TP + FN}$$

- F1 Score: It's the harmonic mean of precision and recall, and is a good metric to use when there is an imbalance in the classes. The higher the value, the better the balance between both metrics.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Balanced Accuracy (BAC): It takes into account the imbalance of the classes and calculates the average accuracy of each class. A great feature of BAC is that it captures how well a model performs in both target classes.

$$BAC = \frac{Recall + Specificity}{2} = \frac{1}{2} \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right)$$

These are scoring metrics that measure the performance of a classifier in terms of the decisions it makes (final scores). However, other metrics rely on probabilities rather than binary predictions, such as the ROC-AUC.

The Area Under the Receiver Operating Characteristic Curve (ROC-AUC) is an evaluation metric that indicates the degree of separability (distinction) or intermingling (crossover) between the predictions of the two classes. It is calculated by plotting the False Positive Rate (FPR) against the True Positive Rate (TPR) at various thresholds and then computing the area under this curve. A higher ROC-AUC score indicates a greater distinction and lower crossover between the predictions of the two classes.

The combined analysis of multiple scoring metrics allows for a comprehensive understanding of the performance of a model. By using different metrics, the strengths and weaknesses of a classifier can be better understood, and the model's results can be better explained because each metric provides a different perspective on the model's performance. Also, this can help to detect classifiers that are learning in different ways, so that we can make more informed decisions about which models to use in our project with confidence in their explainability.

2.5. Hyperparameter tuning using Genetic Algorithms

For the hyperparameter optimization of boosting algorithms we will employ genetic algorithms. In this section, we will provide a comprehensive explanation of the theoretical foundations necessary for a clear understanding of genetic algorithms and their application in this context.

A genetic algorithm is an optimization technique that is inspired by the process of natural selection. The goal of a genetic algorithm is to find a suboptimal solution to a complex problem for which finding the optimal solution is very costly in terms of computation time and power. Genetic algorithms are inspired by the process of evolution and natural selection.

We will now explain the key concepts needed to understand a genetic algorithm.

- **Individual:** In a genetic algorithm, an individual is a candidate solution to the problem that is being optimized. It is also called a chromosome (Figure 2.2), since it consists of a sequence of components or characteristics that are called genes.

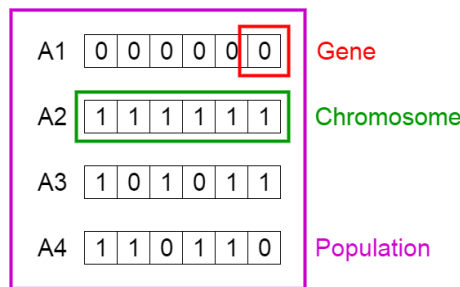


Figure 2.2: Gene, chromosome, and population example.

- Gene: Genes (Figure 2.2) represent the different aspects or parameters of an individual.
- Generation: It is the set containing all the individuals at a certain point. It is also called population (Figure 2.2).
- Fitness: It is a function to assign a numerical value to the goodness of an individual. It should be predefined by the user. Finding an accurate fitness function is crucial in a genetic algorithm, since it is the value that will determine the optimization path.
- Selection: It is the phase in the genetic algorithm in which individuals are selected and pass their genes to the next generation. There are many different selection methods. Some of them are:
 - Roulette wheel selection: The probability of selecting an individual is proportional to its fitness. The higher the fitness is, the more probability to select it.
 - Elitist selection: The individuals to pass their genes to the next generation are taken from the top individuals, ordered from best to worst.
 - Tournament selection: Introduces elitism in a softer way. A subset of k (tournament size) individuals is randomly selected from the population, and a direct comparison is made between them. The individual with the best fitness value within the subset is selected as a parent of the next generation.
- Crossover: It is the phase in the genetic algorithm in which the genes of two parents are recombined to generate offspring. There are many different crossover methods. Some of the most popular ones are:
 - One-point crossover: A point is chosen randomly at both parents' chromosomes. Children will take the left part from one parent and the right part from the other, and vice versa. It can be represented as in Figure 2.3.

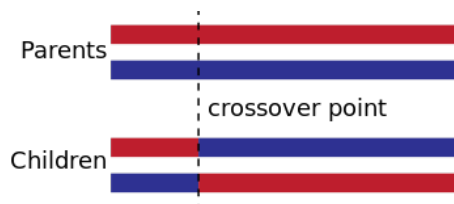


Figure 2.3: One point crossover.

- Two-point crossover: In a similar manner, two points are chosen randomly at both parent's chromosomes, so that a region is left in the middle of them. Children will take genes in this region from one parent, and genes in both edges from the other, and vice versa. It can be represented as in Figure 2.4.

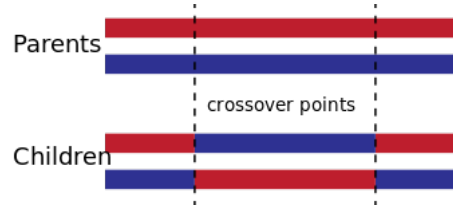


Figure 2.4: Two point crossover.

- Uniform crossover: When creating the children, each gene is taken from either parent with equal probability.

The crossover phase has an associated probability, which refers to the likelihood that two parent solutions will exchange genetic material to create new offspring solutions.

However, making crossovers over the same sets of genes might end up giving similar results after some iterations, so we would be stuck with the same kind of solutions. This is why the mutation phase is introduced.

- Mutation: Mutation is a crucial phase in a genetic algorithm, since it jumps in the solutions space to explore regions that would not be explored if it just depended on the parents.

Imagine that in the previous classification problem we have a model that has a parameter, say, *alpha*, that accepts continuous values between 0 and 1. If the parents that define the next generation had, say, values just of 0.2 and 0.3, larger values of *alpha* closer to 1 could only be explored by a mutation jump.

There are many different mutation methods. Some of them might not make sense depending on the problem, but here we list some of them:

- Random resetting mutation: All the genes are considered, and according to the mutation probability, the values of the selected ones are reset by generating new values. It can be done with or without restriction of the possible values for each gene, depending on the problem.
- Rotation to the right: All the genes are considered, and according to the mutation probability, the values of the selected ones are rotated to the right.
- Inversion: Similar to rotation to the right, but inverting the value of the selected genes. The definition of inverse should be predefined.

Mutation also has an associated probability. The mutation probability refers to the likelihood that a randomly selected gene in a chromosome will be modified or changed to create a new solution.

- Termination: The termination is the criterion needed to be reached to stop the algorithm. Some of the most common terminating conditions are:
 - Acceptance threshold reached: A minimum value for the output function has been achieved.

- Fixed number of generations reached.
- Maximum running time reached.

Genetic algorithm steps

A genetic algorithm consists of several steps:

1. Preconfiguration: Several parameters should be defined before running the genetic algorithm. These include: Generation size, fitness function, selection method, crossover method, crossover probability, mutation method, mutation probability and termination criterion.
2. Initialization of the random initial population.
3. Calculate the fitness of the population.
4. Perform selection over the current population.
5. Perform crossover over the current population, according to the crossover probability.
6. Perform mutation over the current population, according to the mutation probability.
7. Repeat steps 3-7 until the termination criterion is met.

One of the key features of a genetic algorithm is its ability to explore a large search space and find good solutions even when the problem is complex and the solution is not apparently easy. This is why genetic algorithms are widely used in hyperparameter tuning of machine learning models, in which it is common to have many hyperparameters that need to be set in order to optimize the performance of the model.

2.6. Natural Language Processing with Large Language Models and transfer learning

Introduction to NLP and its applications

Natural Language Processing (NLP) is a subfield of deep learning that focuses on enabling computers to process, analyze, and understand human language. The aim of NLP is to bridge the gap between human communication and computer systems, allowing computers to interpret and generate human language.

One of the main challenges of NLP is dealing with the complexity and ambiguity of human language, which is constantly evolving and can vary significantly depending on context and cultural background. NLP systems must be able to adapt to these variations and understand the nuances of human language.

NLP has a wide range of applications, from language translation to sentiment analysis, and has become increasingly important in the development of virtual assistants and chatbots.

In recent years, advances in machine learning and deep learning have led to significant breakthroughs in NLP. Techniques such as neural networks and transformers have enabled computers to generate human-like language, leading to complex systems such as the recent ChatGPT, which is able to respond to user queries in a natural and conversational manner. In addition, some transformer architectures allow the neural network to be used for other tasks such as classification, for which transfer learning is applied.

Transfer learning in NLP

Transfer learning is a powerful AI technique that allows models to reuse the knowledge learned from one task and apply it to another. In NLP, transfer learning is usually applied through the processes of pretraining and fine-tuning.

Pretraining is the process of training a language model on a large corpus of text data, such as Wikipedia or a collection of books. The goal of pretraining is to enable the model to learn the underlying patterns and structures of language. This is achieved through unsupervised learning, where the model is trained to predict missing words or sentences in the text corpus. By pretraining on a large corpus, the model can learn to recognize and generate language patterns, which can be fine-tuned for specific NLP tasks.

Fine-tuning is the process of adapting a pre-trained language model to a specific NLP task, such as sentiment analysis or, in our case, default prediction. It involves retraining the pre-trained model on a smaller task-specific dataset, with the goal of optimizing the model for the specific task, improving its performance.

In recent years, pre-trained models such as BERT and GPT have become popular in the NLP community due to their high performance on various tasks. BERT, for example, has been fine-tuned on tasks such as sentiment analysis, named entity recognition, and question answering, achieving top performance on benchmark datasets (Devlin et al., 2019).

Furthermore, transfer learning is very interesting from the energetic efficiency point of view. By transferring knowledge from already fitted models, the need for training new models from scratch is reduced, which can save a significant amount of energy. Thus, transfer learning can play a crucial role in achieving energetic efficiency in machine learning applications, contributing to the development of sustainable and environmentally friendly technologies.

NLP and neural networks

In this section, we will present the main concepts that allow language processing with neural networks.

Word Embeddings

Word embeddings are a representation technique that transforms words into dense vectors of numerical values.

Traditional NLP models represent words as one-hot vectors, where each word is represented by a vector with a 1 in the position corresponding to its index in the vocabulary and zeros elsewhere. However, one-hot vectors are very sparse, leading to low efficiency, and do not capture the semantic relationships between words. Word embeddings address this limitation by representing words as dense vectors with continuous values that capture the meaning and context of the word.

Word embeddings are typically learned using unsupervised learning. Some of the most common algorithms are word2vec and GloVe. They analyze large text corpora and learn to assign words to vectors in a high-dimensional space, in which the distance between vectors corresponds to the semantic similarity between words.

While in algorithms such as word2vec each word has a fixed representation regardless of the context within which the word appears, Large Language Models, such as BERT, produce word representations that are dynamically informed by the surrounding words.

Neural Networks

Neural Networks are a type of supervised machine learning algorithm that are designed to model complex relationships between inputs and outputs. They are inspired by the structure of the human brain, which consists of interconnected neurons that process information and communicate with each other.

A Neural Network consists of an input layer, where input data is supplied, hidden layers, consisting of several layers of artificial neurons, and an output layer, in which the solution to the problem is provided.

Each neuron in the hidden layers receives input from the neurons in the previous layer, and computes a weighted sum of the inputs, applies an activation function to this sum, and passes the result to the next layer. These weights are adjusted in the process of backpropagation, which aims to minimize the error between the predicted output and the actual output. This allows the network to learn to accurately map inputs to outputs, even for complex and nonlinear relationships.

The main downside of neural networks is that they are a black-box model, meaning that their internal workings are not explainable, and it may lead to results that are hard to interpret.

Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a type of Neural Network whose main advantage is their performance with sequential data, such as text, which can be seen as a sequence of words.

We as humans also have a certain degree of sequential memory. For example, it is very easy for us to say the alphabet in its natural order. However, it is very hard for us to say it backwards. We learn the alphabet following a sequential pattern, and this is the main idea of RNNs.

To this end, RNNs add a new concept to traditional neural networks called the hidden state, which is a representation of previous inputs that loops over the hidden layers. While traditional Neural Networks process input data in a single pass, RNNs process data sequentially. Each element of the sequence is represented by a vector, and the hidden state is updated after an element is processed.

RNNs are very useful in NLP because thanks to this hidden state they take into account the entire previous text to predict the following word. The longer the distance to a certain word in the text, the lower its importance to predict the following word.

The main disadvantage of RNNs is that they may store more recent information that is less relevant than older text. This issue is known as short-term memory, which also leads to the vanishing gradient problem² (Hochreiter, 1998). With the aim of solving this problem, Hochreiter and Schmidhuber came up with the idea of Long Short-Term Memory networks in 1997.

Long Short-Term Memory Networks

First introduced by Hochreiter and Schmidhuber (1997), Long Short-Term Memory (LSTM) networks are a type of RNN architecture designed to overcome the vanishing gradient problem of traditional RNNs.

LSTMs have a long and complex mathematical explanation behind, but its basic idea is to have a memory cell (or cell state) that can retain information for long periods of time, while also being able to selectively forget or update this information. The memory cell is controlled by three gates (Figure 2.5), which are independent neural networks that decide which information is maintained and which information is forgiven or updated:

1. Forget gate: Determines how much old information should be forgotten. It contains a sigmoid³ activation which normalizes its output to a value between 0 and 1. This output is then multiplied by the memory cell, so that values close to 0 are almost forgiven, and values closer to 1 are kept.
2. Input gate: Determines how much new information should be stored in the memory cell. This is done by a combination of sigmoid function (between 0 and 1, determining the importance of the current value) and a *tahn*⁴ function (between -1 and 1, to help to regulate the network).
3. Output gate: Determines how much information should be outputted from the memory cell. Done by a sigmoid function that provides the percentage of data that should be kept in the hidden state of the following network.

²Phenomenon where the gradients (derivatives of the loss with respect to the weights) become very small as they are propagated back through time during training, which leads to the network being unable to learn long-term dependencies in the data.

³Sigmoid function: Matches any value in the X domain to a Y value between 0 and 1. Represented by red circles in Figure 2.5.

⁴Hyperbolic tangent function: Matches any value in the X domain to a Y value between -1 and 1. Represented by blue circles in Figure 2.5.

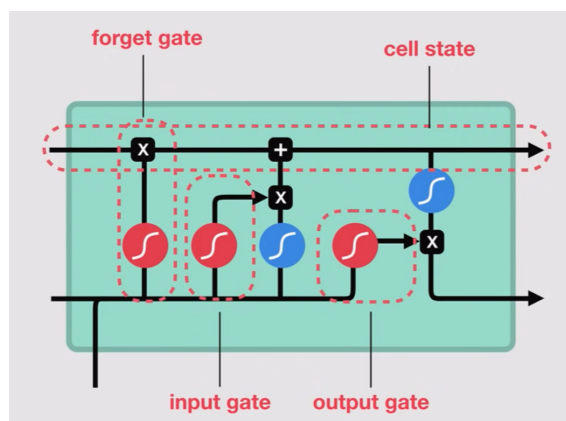


Figure 2.5: LSTM network diagram. Source: <https://www.youtube.com/watch?v=8HyCNIVRbSU>

LSTM networks have proven to be very effective in tasks that require modeling long-term dependencies, and make a more efficient use of relevant information in sequential data than RNNs.

Attention Mechanisms

LSTMs implied a much larger memory window in sequential data than RNNs. However, for very long texts, it is still not enough. Attention mechanisms take this memory window to infinity, allowing the processing of text of any length.

The basic idea behind attention mechanisms is to enable the network to selectively focus on different parts of the input data, depending on the context and the task at hand. This is done by assigning different weights to different parts of the input data, called attention weights, allowing the network to selectively attend to the most relevant information based on its current state.

One of the most common attention mechanisms used in NLP is self-attention, in which a neural network attends to different parts of the same input sequence. Self-attention is often implemented using a multi-head attention mechanism, which computes attention weights for different subspaces of the input sequence in parallel. This allows the network to capture different types of relationships between the input elements, leading to more robust and accurate predictions.

Transformer Architecture

Transformers are a type of neural network architecture introduced by Vaswani et al. (2017), from the Google Brain⁵ and Google AI Language⁶ research groups. Since its presentation, it has revolutionized the field of NLP, leading to state-of-the-art developments.

Transformers are an attention-based encoder-decoder architecture. On a high level, the encoder maps an input sequence into an abstract continuous representation

⁵More information about Google Brain: <https://research.google/teams/brain/>

⁶More information about Google AI Language: <https://research.google/teams/language/>

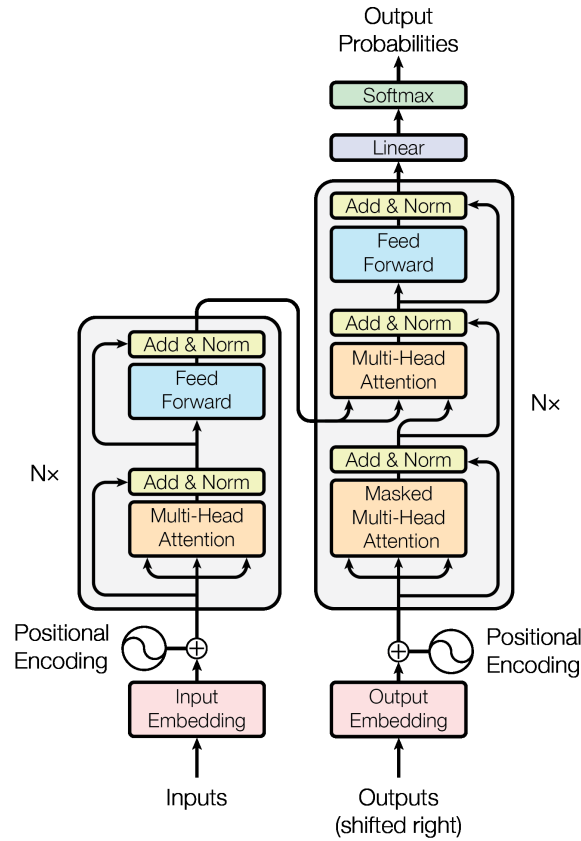


Figure 2.6: Transformer network diagram. Source: Vaswani et al. (2017)

that holds all the learned information of that input. Then, the decoder takes that continuous representation and generates a single output, while also feeding previous outputs. Both the encoder and the decoder are composed of multiple layers of self-attention and feed-forward neural networks.

The self-attention mechanism described in the original paper relies on three types of vectors:

1. **Query vector:** Represents the word currently being attended to in the self-attention mechanism. It is used to compute the attention scores that determine the importance of the other words in the sequence.
2. **Key vector:** Represents the other words in the sequence when computing the attention scores for the current word. It is used in combination with the query vector to compute the attention scores.
3. **Value vector:** Represents the information associated with the word. It is used to compute the output of the self-attention mechanism, which is a weighted sum of the value vectors of all the words in the sequence.

Each word in the sequence has these three vectors, which are used in combination to compute attention scores, which determine the importance of each word in the sequence with respect to the current word. The attention scores are then used to

compute a weighted sum of the value vectors of all the words in the sequence, which is used as the output of the self-attention mechanism.

Figure 2.6 depicts the transformer diagram presented in the original paper.

In the paper in which transformers were proposed, called “*Attention is all you need*” (Vaswani et al., 2017), researchers argued that RNNs are not needed for this kind of model to have a good performance. They proved that an architecture just based on attention mechanisms performs better and with better scalability to longer texts.

Another key advantage is that the computation of each output in a transformer can be done in parallel, unlike RNNs, which require sequential computation due to their inherent recurrence. Additionally, transformers typically involve less data movement than RNNs, which further reduces the computational cost of parallelizing the network. This makes transformers much more efficient to compute on GPUs, leading to lower computation times.

Large Language Models: BERT

Large Language Models (LLMs) are advanced NLP systems that understand and generate human-like text using a Transformer architecture. Their training process involves exposing them to extensive text data and teaching them to predict subsequent words or phrases based on context. By using large-scale computing resources and specialized architectures, LLMs learn from billions or trillions of parameters, enabling them to generate coherent text. Due to the high computational cost, LLMs are trained once and then used through transfer learning, where they are fine-tuned for specific NLP tasks. This approach leads to better results even with less labeled data than traditional supervised learning NLP methods.

BERT (Bidirectional Encoder Representations from Transformers) is a Large Language Model presented by Devlin et al. (2019), from the Google AI Language research group. It is a pre-trained transformer model that is trained on massive amounts of textual data, such as Wikipedia and web pages, to create a language representation model. Its pre-training phase consists of two main tasks:

1. Masked language modeling (MLM): Masking out certain words in a sentence and predicting the masked words based on the context of the sentence.
2. Next sentence prediction (NSP): Predicting whether two sentences are adjacent in a document or not.

Once pre-trained, BERT can be fine-tuned on the desired specific NLP task. This fine-tuning process involves training the model on a specific dataset with a labeled task-specific objective so that the model’s weights are adjusted to it.

BERT has demonstrated to be particularly successful in NLP tasks due to its ability to generate contextualized word embeddings. Compared to other pre-trained transformer models, it is able to successfully learn the nuances of the language by identifying the different meanings of a word depending on its context in a sentence (Devlin et al., 2019; Kriebel and Stitz, 2022).

2.7. Explaining machine learning models using the SHAP values

SHapley Additive exPlanations (SHAP) values, introduced by Lundberg and Lee (2017), are a useful tool for interpreting the predictions made by a machine learning model. They help to explain which features are driving a prediction and how they are contributing to it. They are part of what is known as Explainable AI (XAI) tools, which is a crucial concept in this Bachelor's thesis.

SHAP values are specifically designed to provide post hoc explainability for models that lack inherent explainability. They accomplish this by focusing on the input and output of the model, rather than the inner workings of the model itself. By not considering the model's internal mechanisms, SHAP values offer insights into the relationship between input variables and the corresponding output, regardless of the potential interdependence among input variables.

In our case, SHAP values will help us understand how each feature is contributing to the prediction of default or non-default. The SHAP values for a feature can be positive or negative, representing the contribution of that feature to the prediction of default or non-default. A positive SHAP value means that the feature increases the prediction of default, while a negative SHAP value means that the feature decreases the prediction of default (which is the positive class).

To interpret SHAP values, the Python library⁷ counts with a set of plots that enable us to understand how the input variable range correlates with its positive or negative contribution in predicting default. With these graphs it is possible to analyze the presence or absence of a relationship, the nature of the relationship (direct or inverse), and even more complex relationships beyond linearity, also taking into account the variability in the response.

⁷Source: <https://github.com/slundberg/shap>

Exploration of Data and Experimental Methodology

In this chapter, we are going to delve into the design of the experiment we conducted. We will describe all the steps involved, including: data preprocessing and feature selection, the tuning of the XGBoost and CatBoost models using the genetic algorithm, and the evaluation of the results. We detail each step to provide transparency, that is, a comprehensive account of the entire process and the choices we made to allow the reproducibility of the experiment. Additionally, we will discuss the challenges we faced and how we overcame them to arrive at the final results.

3.1. Dataset Description

For the purpose of this Bachelor's thesis, we have used a public dataset of the P2P lending company LendingClub, which is widely used in credit risk publications and the most widely used when dealing with the P2P market. It includes loans granted by the company between 2007 and 2018.

The original dataset can be found at Kaggle¹. The Kaggle description says:

LendingClub is an American peer-to-peer lending company, headquartered in San Francisco, California. It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC), and to offer loan trading on a secondary market. LendingClub is the world's largest peer-to-peer lending platform. The company claims that \$15.98 billion in loans had been originated through its platform up to December 31, 2015.

However, for this project we have used a modified version of the LendingClub dataset processed by Miller Janny Ariza-Garzón in his Ph.D. Thesis and that is very similar to the one used in the Ariza-Garzón et al. (2020) paper. In this work, the authors propose a model for the granting process, that is, it is used for deciding whether to grant or not a loan, and that is exactly our purpose.

¹Source: <https://www.kaggle.com/datasets/wordsforthewise/lending-club>

The dataset only considers loans whose final status is known for sure (that is, repaid or defaulted). Thus, it eliminates loans in transitory states (in grace period, late, etc.) and excludes loans with no information on income and indebtedness. As a result, the dataset contains a variable called *Default*, which is our target variable to predict. It has a binary value, being set to 1 if the obligation was defaulted, and 0 otherwise. The variables are distributed as follows:

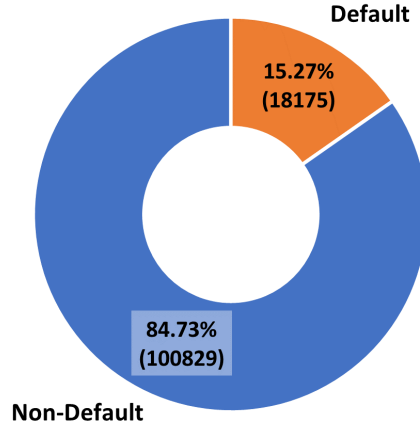


Figure 3.1: Default *vs* Non-Default distribution.

Regarding feature selection, it is worth noting that our aim is to build a loan-granting model capable of making decisions on loan approvals. Therefore, we will just consider those variables that are known at the time of the loan request, as financial entities would do. These variables are taken from the work of Ariza-Garzón et al. (2020), and backed by the work of Lin et al. (2016).

Miller carried out some other modifications of the original dataset to aggregate or simplify the information described by several variables. The credit experience with LendingClub (*experience_c*) was constructed as a binary variable indicating if the borrower is new to the entity. The *dtn* variable in the original dataset excluded the borrower's mortgage and the requested LC loan. However, Miller considered these two factors since they represent the situation of the borrower if the loan is granted. In the original dataset, information about the Fair Isaac Corporation credit bureau (FICO) is given by two values that define the credit scoring interval of the applicant to the bureau. Miller built the *fico_n* variable, which is computed as the average of these two values.

Since the use of the text description of the purpose of the loan is a fundamental part of this project, we will use the variable *desc* that was omitted in Miller's dataset. For unknown reasons, we noticed that the descriptions were no longer available as of March 2014, so we further filter Miller's dataset considering only the loans from the beginning (June 2007) until that month.

As a result, our dataset consists of 119,004 records or obligations, all of them including a description and no missing data.

The meaning and distribution of each variable are explained below. Table 3.1

shows a summary of them.

Quantitative variables		
Variable name	Meaning	Observed range
<i>loan_amnt</i>	The amount of the loan requested by the borrower.	1,000 - 35,000
<i>dti_n</i>	Indebtedness ratio. $dti = \frac{\text{Monthly debt payments}}{\text{Monthly income}}$	0.0 - 34.99
<i>fico_n</i>	Credit score that measures financial performance on previous obligations.	662.0 - 847.5
<i>revenue</i>	Borrower's annual income.	4,000 - 7,141,778
Categorical variables		
Variable name	Meaning	Categories
<i>emp_length</i>	Employment length of the borrower in years.	NI, <1 year, 1 year, 2 years, 3 years, 4 years, 5 years, 6 years, 7 years, 8 years, 9 years, 10+ years
<i>purpose</i>	Loan purpose category for the loan request.	Car, credit card, debt consolidation, educational, home improvement, house, major purchase, medical, moving, other, renewable energy, small business, vacation, wedding
<i>home_ownership</i>	The home ownership status provided by the borrower.	Mortgage, rent, own, other
<i>addr_state</i>	The United States state of residence provided by the borrower.	AK, AL, AR, AZ, CA, CO, CT, DC, DE, FL, GA, HI, IA, ID, IL, IN, KS, KY, LA, MA, MD, ME, MI, MN, MO, MS, MT, NC, NE, NH, NJ, NM, NV, NY, OH, OK, OR, PA, RI, SC, SD, TN, TX, UT, VA, VT, WA, WI, WV, WY
Text variables		
Variable name	Meaning	Character length range
<i>title</i>	Title of the loan provided by the borrower.	2 - 85
<i>desc</i>	Description of the loan provided by the borrower	3 - 3,966

Table 3.1: Variables summary.

Quantitative variables

The following variables are quantitative, and as such, they will have a continuous numerical value.

Loan amount (*loan_amnt*)

The amount of the loan requested by the borrower follows the distribution depicted in Figure 3.2.

The mean of the loan amount requested is \$13,999.19, with a standard deviation of 8,012.17. The distribution plot also exhibits noticeable peaks, indicating a

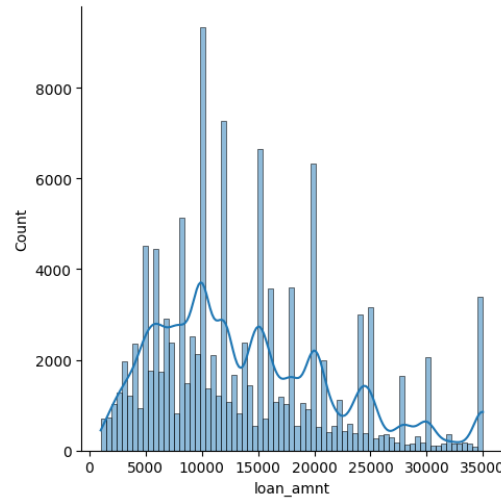


Figure 3.2: Loan amount distribution.

higher frequency of round numbers, which is expected and logical. Most repeated values are \$10,000 (8,631 occurrences), \$12,000 (6,851 occurrences), \$15,000 (6,140 occurrences) and \$20,000 (5,808 occurrences).

Debt-to-Income (dti_n)²

The indebtedness percentage is calculated from the borrower's total monthly debt payments on total debt obligations divided by the borrower's self-reported monthly income. Hence, a lower DTI implies a reduced monthly debt burden for the borrower in relation to their income, which seems less risky. Figure 3.3 shows its distribution.

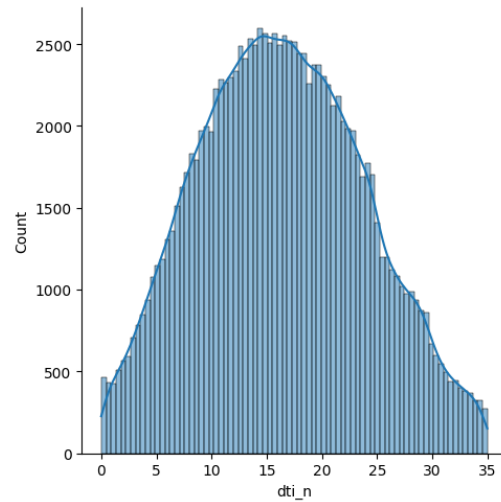


Figure 3.3: Debt-to-Income distribution.

This debt variable has a mean of 16.38 and a standard deviation of 7.53. More-

²The variable is called dti_n , where dti stands for “Debt-to-Income”.

over, we can appreciate a Gaussian-like distribution with no tails.

FICO Score (*fico_n*)

The FICO³ Score originates in the USA and measures the financial performance of the lender on previous obligations. It ranges from 300 to 850, with higher scores indicating lower credit risk. It is calculated based on the individual’s payment history (35%), debt burden (30%), length of credit history (15%), types of credit used (10%), and recent searches for credit (10%).

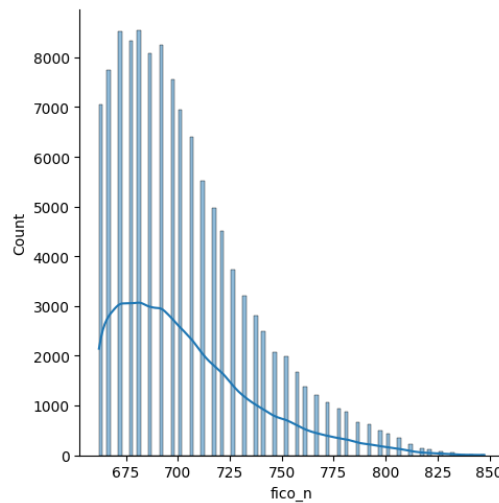


Figure 3.4: FICO Score distribution.

The histogram (Figure 3.4) shows that the distribution of the FICO score is right-skewed with a mean of 703.54 and a standard deviation of 32.74.

The lowest observed value is 662.0, which according to the VantageScore 4.0⁴ is a “fair” score. Considering that the poorest FICO value is 350, this means that we have a bias towards good FICO scores in our dataset. We do not have any instances from the “very poor” range (350-549) nor from the “poor” range (550-649). This is probably due to the fact that the LendingClub dataset consists of already granted loans, and applicants with low FICO scores were ruled out by LendingClub, or may not get a loan offer, or may not have accepted the loan offer due to a high interest rate.

The “fair” FICO score category ranges from 650 to 699. In this range we have the bulk of the dataset with a total of 64,062 borrowers, representing 53.83% of the total loans. The range of scores 700-749 is considered as a “good” FICO score. In this range we have 42,653 borrowers, representing 35.84% of the total loans.

The highest FICO score in our dataset is 847.5. According to the VantageScore 4.0, scores in the range 750-850 are “excellent” scores. 12,289 borrowers out of the

³Fair Isaac Corporation: Data analytics company focused on credit scoring services.

⁴Source:https://en.wikipedia.org/wiki/Credit_score_in_the_United_States#VantageScore

119,004 considered loans are in this range, which implies a 10.33% of the instances in our dataset.

Revenue (*revenue*)

The revenue represents the borrower's self-declared annual income during registration. Its distribution is as in Figure 3.5.

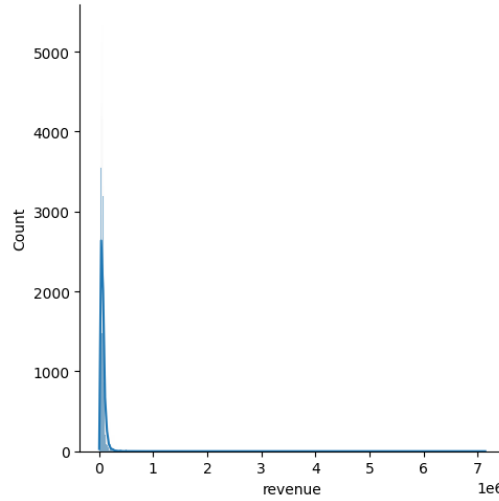


Figure 3.5: Revenue distribution.

The mean for the revenue of the borrowers is \$72,441.18, with a standard deviation of 53,089.98. Given that income is a very broad variable, we will take a close look at it without considering the outliers. For this, we will consider as an outlier every borrower whose income is above \$300,000. With this, we leave out 578 borrowers, which are 0.48% of the total number of borrowers, so Figure 3.6 is still very representative.

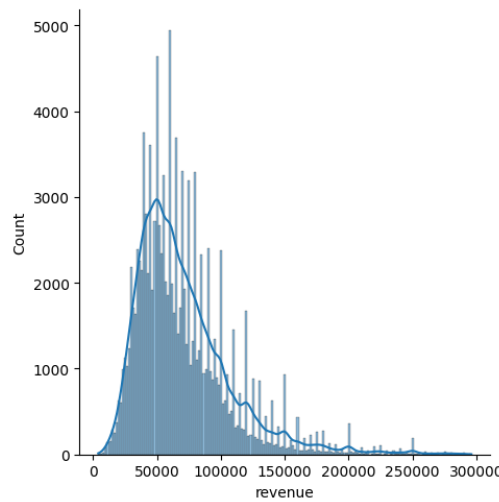


Figure 3.6: Revenue distribution without outliers.

Similar to the loan amount requested, there are instances where the reported income exhibits peaks corresponding to round numbers.

Taking out the outliers (revenue $>$ \$300,000, Figure 3.6), we have an average of \$70,555.85, with a standard deviation of 37,289.99, which is very reduced compared to that in the original data. Despite the borrowers taken out, the median keeps the same (\$62,000), and of course the mode is not modified (\$60,000).

Regarding the quantitative variables, an interesting aspect to investigate is the correlation between revenue and loan amount. By conducting this analysis, we will be able to confirm whether our dataset contains a diverse range of data points concerning the relationship between these two variables. Below we can see a scatter plot (Figure 3.7) that relates these two variables. As in the previous plot, we will take out the outliers (revenue $>$ \$300,000) for a more readable and enlightening representation.

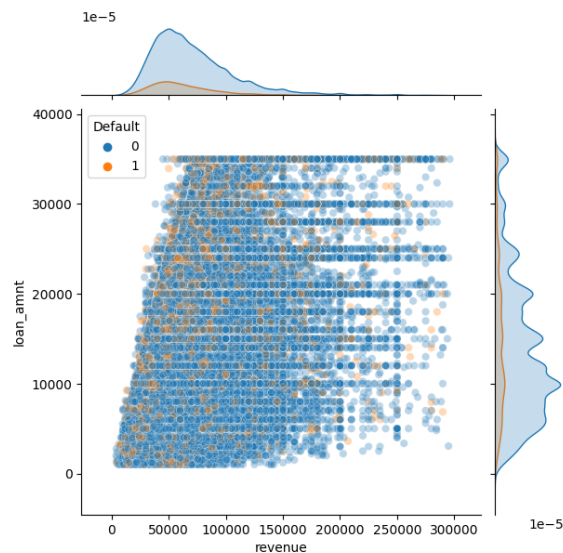


Figure 3.7: Loan amount *vs* revenue scatter plot.

We can see in the distributions at the margins that the loan amount data is much more uniform than the revenue data. In addition, we see that default (blue dots) and non-default (orange dots) loans are distributed along both variable ranges, but as expected, it is not easy to find regions where both classes can be separated.

The scatter plot also reveals that it is very infrequent to find small-amount loans coming from people with high revenues, most likely because they do not apply for them. The opposite holds also true, as we can see that for the lower revenues there are no loans over a certain amount (we do not know whether the limitation was imposed by the LendingClub platform, or that the loan with the interest rate was not accepted by the applicant, or that no offer arrived).

Categorical variables

Categorical variables are those that take their values from a finite set of predefined possible values. Below, it is indicated the set of discrete possible values for

each of them.

Employment length (*emp_length*)

Length of current employment in years. Possible values are between 0 and 10, where 0 means less than one year and 10 means ten or more years. There is also an extra category called NI (“no information”), covering unemployed borrowers. Note that, although its values are numerical, it is a categorical variable as it can only take values in a certain range. Its distribution is as in Figure 3.8.

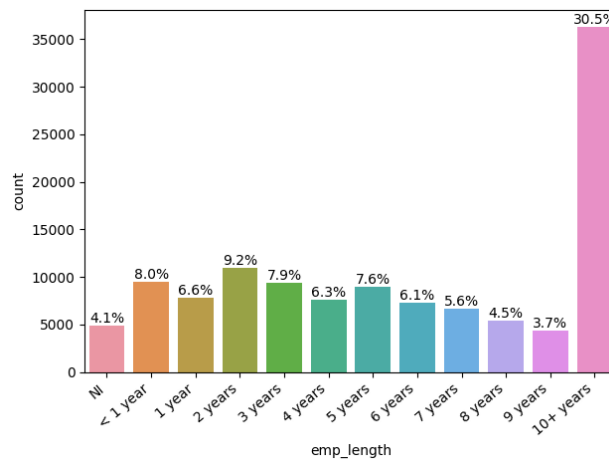


Figure 3.8: Employment length distribution.

There is a greater presence of people in the “10+ years” employment category, with 30.5% of the borrowers. This could indicate that the majority of individuals seeking credit loans possess job stability.

By looking at Figure 3.9, it seems that the category with the highest risk is that of individuals with no employment history. Moreover, we can see that loans from individuals with over 10 employment years have a default rate very close to the mean.

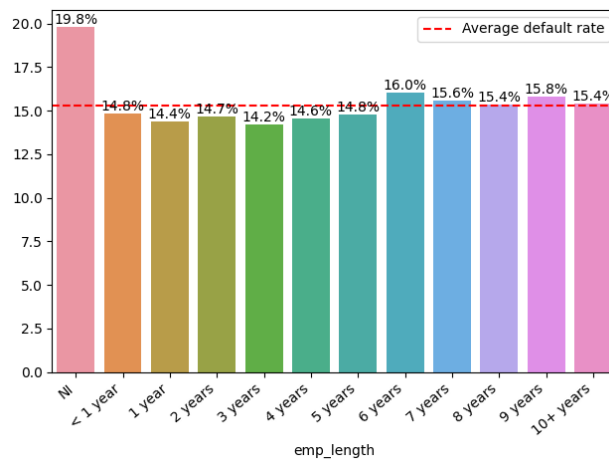


Figure 3.9: Employment length default distribution.

Purpose (*purpose*)

Category provided by the borrower for the loan application. It defines the reason for which the loan is requested. Possible values are: *car*, *credit_card*, *debt_consolidation*, *educational*, *home_improvement*, *house*, *major_purchase*, *medical*, *moving*, *other*, *renewable_energy*, *small_business*, *vacation*, *wedding*. Its distribution is as in Figure 3.10.

It is noteworthy that the most frequent purpose is “debt consolidation”, followed by “credit card” and “home improvement” purposes. In the context of “debt consolidation”, borrowers seek funds to pay off some or all of their existing credit obligations, resulting in a consolidated debt owed to a single lender. Therefore, it is reasonable to expect a higher frequency of “debt consolidation” loans, which can involve the repayment of loans obtained for other purposes.

Moreover, the “other” category of loan purposes is quite frequent, and this is very interesting for the goals of our project. By examining the descriptions of these loans, we can gain valuable insights into the financial behaviors and needs of individuals that may not be captured by common loan purposes.

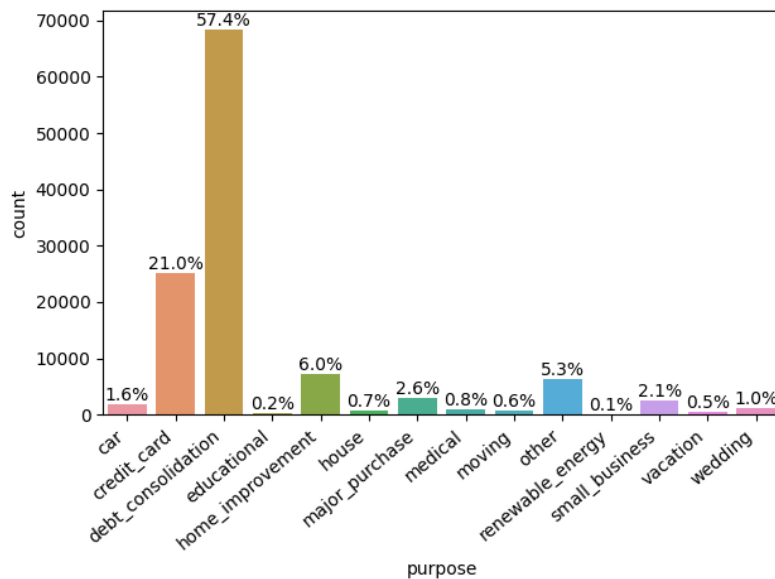


Figure 3.10: Purpose distribution.

We will now analyze the risk of each purpose by looking at its default rate in Figure 3.11. We can see that the “small business” purpose is by far the most risky one. On the contrary, “car” and “major purchase” are the purposes with lower default rates.

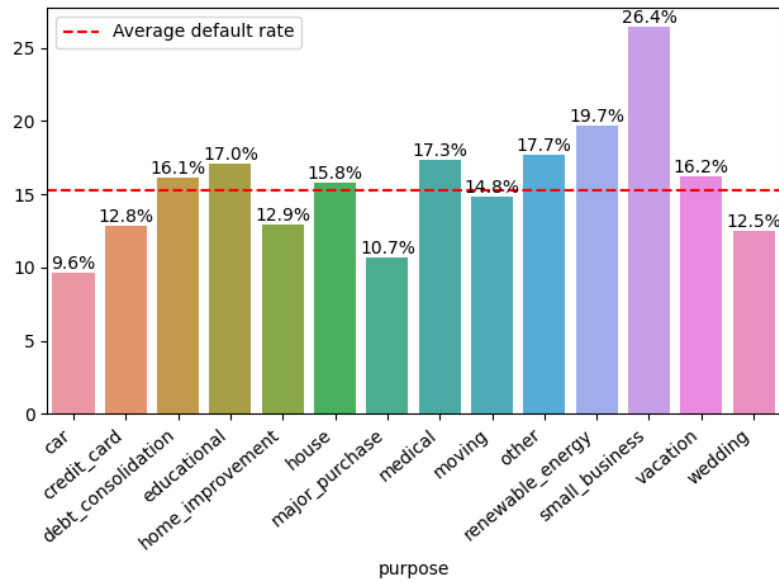


Figure 3.11: Purpose default distribution.

Home ownership (*home_ownership*)

The borrower's home ownership status. Possible values are: *MORTGAGE*, *RENT*, *OWN*, *OTHER*. Figure 3.12 represents its distribution.

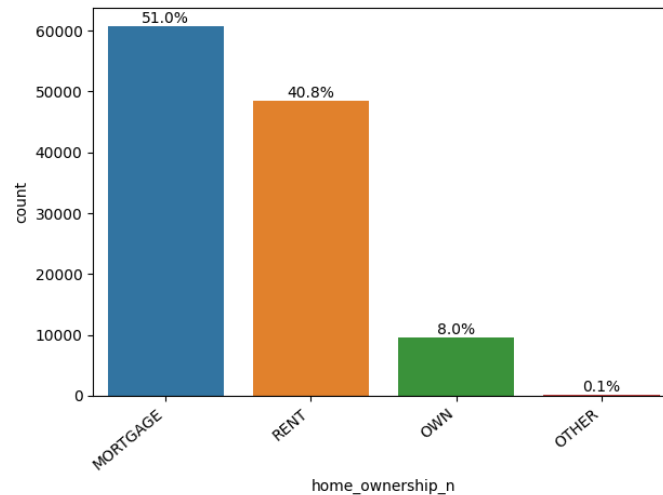


Figure 3.12: Home ownership distribution.

The most commonly reported home ownership status among individuals is “mortgage”, with more than half of the loans, closely followed by “rent”. A relatively low number of individuals report that they “own” their home. The least frequently reported home ownership status is the “other” category, which encompasses individuals who have other living arrangements.

Now we will focus on assessing the risk of each home ownership category, which will be done by examining its default rate (Figure 3.13).

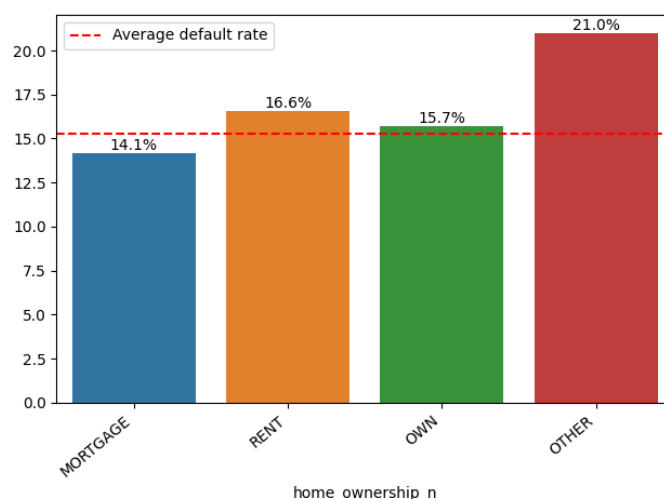


Figure 3.13: Home ownership default distribution.

We can analyze that the riskiest home ownership condition is “other”. This may be because a priori it seems to be the most unstable situation. The rest of the categories are quite evenly matched, with approximately 15% of default.

Address state (*addr_state*)

The State of residence provided by the borrower in the loan application. Possible values are each of the 50 states of the United States (Figure 3.14).

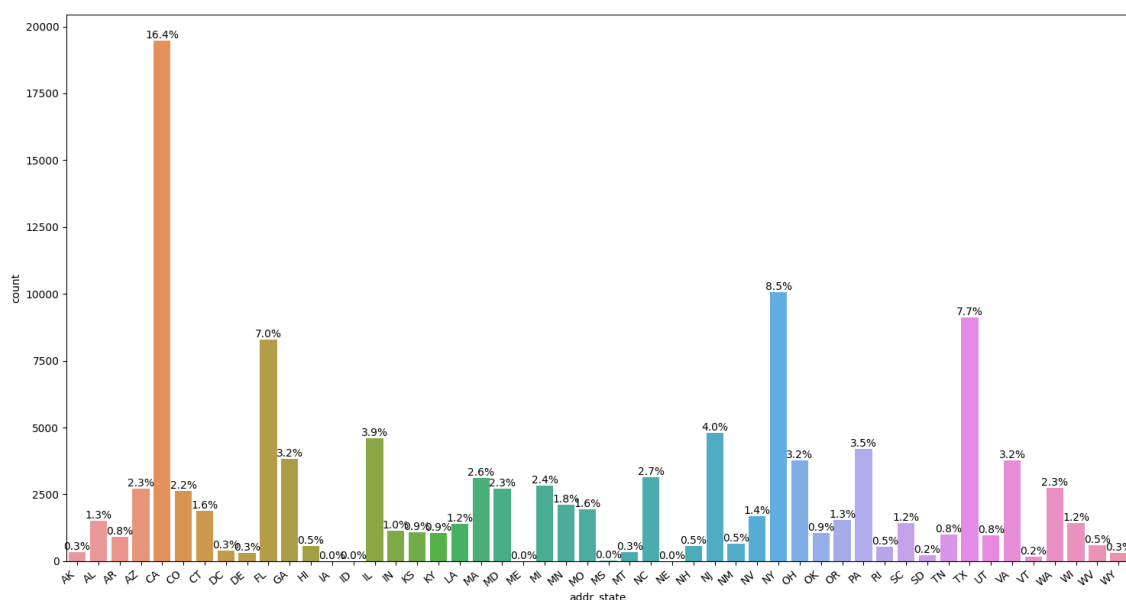


Figure 3.14: Address state distribution.

There is a notable spike in loans requested from California (16.4%), with approximately twice as many loans as the following states, which are New York (8.5%) and Texas (7.7%).

Note that although certain states have a 0.0% representation in the total count, it does not imply a complete absence of loans in those states. The states with minimal loan counts are as follows: Iowa (1 loan), Idaho (1 loan), Maine (1 loan), Nebraska (4 loans), and Mississippi (17 loans).

Now we will analyze the risk of a loan coming from each of the states by looking at their default rate in Figure 3.15.

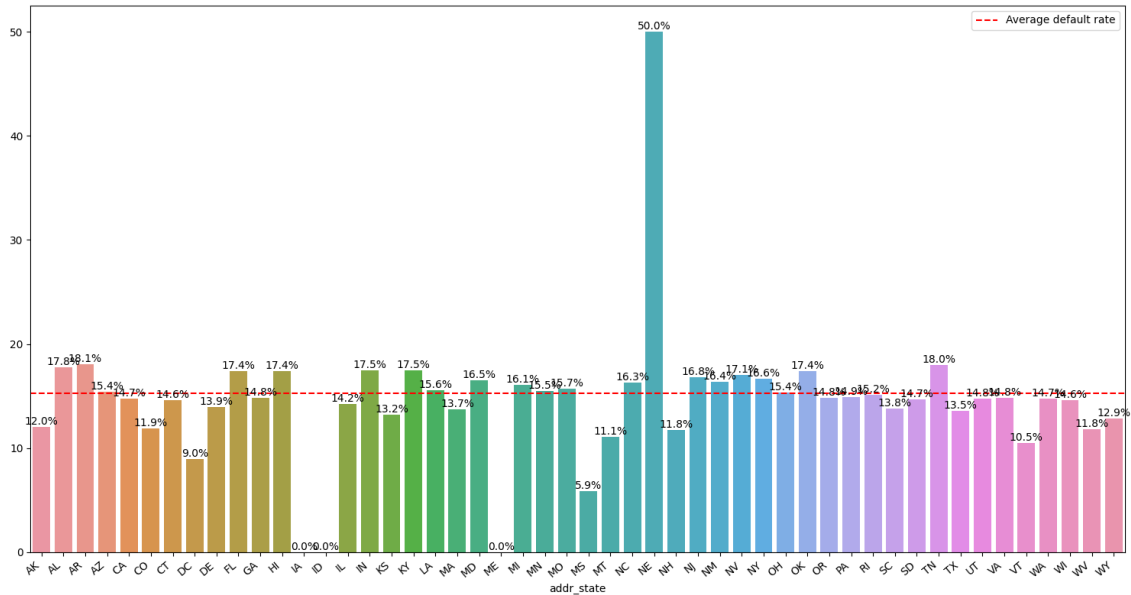


Figure 3.15: Address state default distribution.

We can see that the default rate is very close to the mean in all the states. Most outstanding outliers are those with a low quantity of requested loans. For instance, Nebraska has a 50% of default rate because just 4 loans come from there, and 2 of them were defaulted. Something similar happens with Iowa, Idaho, Maine and Mississippi, but with default rates much lower than the average.

Upon considering only the states with more than 100 requested loans, it is interesting to observe that the District of Columbia (DC) and Vermont (VT) exhibit lower default rates (9.0% and 10.5% respectively), while Arkansas (AR) and Tennessee (TN) showcase higher default rates (18.1% and 18.0% respectively). This finding is particularly surprising given the geographical proximity of these pairs of states.

Text variable

Now we will analyze text features, where the applicants introduce text freely and, as a result, they contain non-categorized data. The text will be processed using transfer learning over Large Language Models in order to transform them into input variables for the classification algorithms.

The dataset considered includes two textual variables: title and description. Upon analyzing the title, we concluded that it contains similar information to the

purpose variable due to its brevity. Therefore, we will just focus on processing and analyzing the description.

Description (*desc*)

Description of the loan provided by the borrower. Its length distribution is as in Figure 3.16.

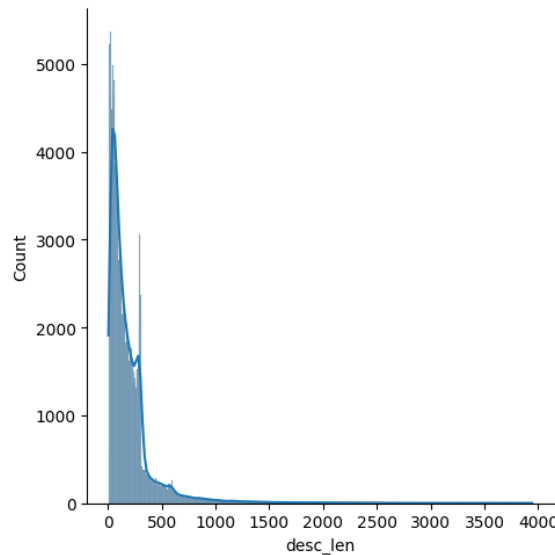


Figure 3.16: Description length distribution.

The description of the loan is a flexible field and consequently the distribution of its length is wide. Unsurprisingly, most of the loans have short descriptions. The distribution has a mean of 194 characters, with a standard deviation of 251 and a median of 125 characters. Moreover, the longest description contains 3,966 characters, while the shortest one contains just 3 of them.

The most frequent description length is that of 18 characters, which might look a bit strange. If we analyze the descriptions, at first sight we can see that there is a very repeated one, that is the title “Debt consolidation” (and all its variants: uppercase and lowercase, with space or underscore, etc), which has exactly 18 characters, and it matches with the most frequent purpose. For reference, we will now see some descriptions of different lengths.

First, we analyze those descriptions just containing 3 characters:

“Car”

“Spa”

As expected, they lack sufficient detail and resemble a title rather than a description. Now we will take a look at descriptions containing 50 characters:

“Payoff some credit cards and only have one payment”

“This is for a 95 Harley Davidson with 11,000 miles”

Here we have very interesting information that can be gained from processing the text. The first example has the purpose of “debt consolidation”, but by analyzing the description we know that the borrower is left with just one payment, which could be crucial for repaying the loan. In the second example, we gain even more information. As there is no purpose for buying a generic vehicle (there is just the “car” purpose), its purpose is set to “other”, and by analyzing the text we know that his purpose is to buy a motorcycle.

If we keep analyzing longer descriptions, the insights gained from the real purpose of the borrower and of his condition get more detailed. For example, here we have a 150-character-long description:

“I need to pay off two higher loan amounts and get my tractor fixed before summer so I can take care of my property. I also want to buy some new clothes.”

The following description contains a much more detailed condition of the borrower. In addition, we appreciate typos (e.g. “intrest”, “monthly”) in the description, which could also drive the classification towards a sharper result.

“I currently have a Lending Club Loan that I used for Debt Consolidation and have a 26% intrest rate on it. I will use this loan to pay that off to get a lower intrest rate and lower monthly payment. I have a very stable job and income for the last 14yrs”

Another critical use of the description is to gain more information about the most frequent purpose, which is “small business”. By processing the description, we can obtain the industry of the business, which might be a crucial aspect for granting the loan. See, for example:

“I am requesting this loan to use for start up capital for my company. I have 10+ years experience in the transportation industry and am launching a freight brokerage business. I have a niche marketing strategy in place.”

However, longer descriptions might also lead to repetitive text that does not provide any valuable information for the loan. For instance, see the following 1,000-character-long description:

“Hi, well winter is drawing in up here in Ohio and my family and I live in a 100 year old farm house with an Oil furnace. Last years heating bill was bad enough, but this year we’re looking at a 33% increase, so in an attempt to ‘stick it to the man’ we want to purchase and install a wood burning stove, they’re highly efficient now and produce very little polution from a renewable resource so we feel we’re doing fairly good for the environment too. The rest of the money is to pay off a higher interest credit card and for my application for Citizenship here in the USA. I’m from England, met my now wonderful wife and got married (obviously

*;-)), we now have a great 3 year old boy, but I'm feeling the odd one out in the family being the only alien here *grins*. Anyway, thank you in advance for any help you can give us, I've never missed a payment or bailed on a loan, I take my responsibilities very seriously. I just need a little help to keep my family warm this winter. Best wishes James + family"*

In conclusion, descriptions offer a different perspective of the purpose of the loan, much more deeply detailed. Thanks to it we can gain information about the specific purpose of the loan and about the status and condition of the borrower, reducing the severe information asymmetry present in P2P lending. In Chapters 5 and 6 we will analyze if a Large Language Model can use these descriptions to improve the classifier performance.

3.2. Description of the experiment

Having defined the dataset, the algorithms used, and the key concepts, we will outline our experiment. In this section we will discuss the design of the experiment, covering from the detailed implementation to the state that defines our starting point.

Modeling approaches to be compared: dataset and classification algorithms

We have designed three variants of our dataset to assess the impact of including categorical and textual variables in a granting model for loan applications:

1. Quantitative variables: The dataset only considering quantitative features (*loan_amnt*, *dti_n*, *fico_n*, *revenue*).
2. Quantitative and categorical variables: The dataset considering both quantitative and categorical features (*emp_length*, *purpose*, *home_ownership*, *addr_state*).
3. Quantitative, categorical and textual variables: The full dataset including the processing of the textual variable.

The classification algorithms to be compared will be XGBoost and CatBoost. With this, we expect to study the effect of the categorical and textual variables in the classifiers. We will also deeply analyze the native usage of the categorical variables by CatBoost, comparing it with their performance in XGBoost, externally preprocessed. This is discussed in Chapter 4.

For the dataset in the third point, the processing of the textual variable will be done with the BERT model. The neural network will be trained with the loan descriptions and will predict the probability of belonging to the default class. Therefore, the higher the BERT score, the more risky the loan is predicted to be. This BERT score will then be used as another quantitative variable in the classifiers

(XGBoost and CatBoost) and we will analyze the impact of adding the information gained from processing the texts. All this process is discussed in detail in Chapter 5.

As a result, we will run six main experiments regarding boosting algorithms, which involve the comparison of XGBoost and CatBoost performances on each of the three datasets previously mentioned. In addition, the processing of textual descriptions will have its own parallel experiment.

Pipeline of the experiment

The executions of this Bachelor's thesis have been run in pipelines. A pipeline is a sequence in which all the operations needed in the experiments are done, from data preprocessing to results printing and model evaluation. In a pipeline, the output of one stage becomes the input for the next one until the final stage produces the desired output, and it is commonly used to automate machine learning workflows.

Our pipeline has been implemented using Jupyter Notebooks, which required only the setting of initial parameters to make it work in any of the executions. The programming language used in these pipelines has been Python.

To ensure the reproducibility of our experiment, we set a random seed for all the algorithms in our pipeline that involved any kind of randomness. This way, we can obtain the same results every time we run the experiment with the same input data and parameters.

Data preprocessing

For the data preprocessing job we have used the Pandas⁵ library, a popular choice for data handling in data science projects. It allows the management of data in what they call “dataframes”, which is a table that is internally processed very efficiently.

This preprocessing phase included: Feature selection from the initial dataframe, separation of quantitative, categorical, textual and target data, and one-hot encoding of categorical variables for the XGBoost execution (this is not needed for CatBoost since it handles them natively).

Also, in order to obtain the BERT score representing the default percentage prediction, textual variables need to be preprocessed. Since this is a parallel task dealing with NLP, it will be explained in detail in Chapter 5.

K-fold Cross-Validation

Regarding data partitioning, we used a stratified k -fold cross-validation with k being equal to 5 (Figure 3.17). This means that each classifier is trained and tested 5 times, using 4 partitions for training and the missing one for testing. Since it is stratified, each fold has the same proportion of default/non-default as the original dataset, preventing any potential bias in our model evaluation and selection. Also, before partitioning into the 5 folds, data is shuffled to avoid any bias in the original

⁵Source: <https://github.com/pandas-dev/pandas>

order. The final performance of each model is measured as the average of the scores in the 5 test folds.

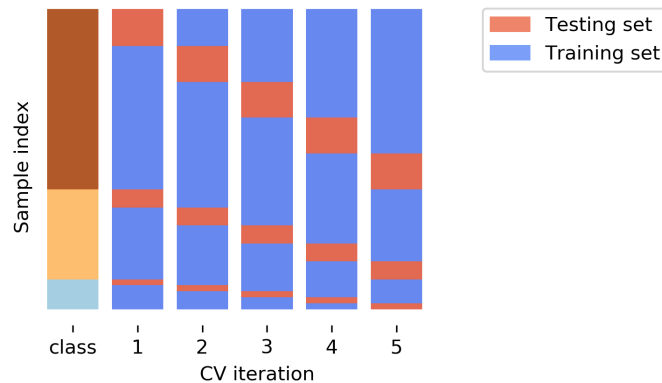


Figure 3.17: Stratified K -fold cross-validation diagram. Source: <https://amueller.github.io/aml/>

Hyperparameter tuning

One of the primary objectives of this work is to thoroughly investigate the effectiveness of hyperparameter tuning in boosting algorithms. To accomplish this, we will employ a genetic algorithm to fine-tune the hyperparameters and analyze their impact on the obtained results. The configuration of the genetic algorithm is described in this section.

Fitness measure

The fitness value for each individual will be the average balanced accuracy in the 5 validation sets of the cross-validation.

In addition to balanced accuracy, we will compute and store the following score metrics for each of the individuals: *accuracy*, *F1*, *precision*, *recall* and *AUC*. In this way, we will be able to analyze the behavior of the different classifiers, resulting in a better explainability of the model.

On the use of AUC as fitness measure

We initially optimized the classifiers by using the area under the curve (AUC) as the fitness measure, as suggested in several papers studying boosting classification with imbalanced datasets (Gong and Kim, 2017; Kriebel and Stitz, 2022). In some cases, it resulted in classifiers that had very good AUC scores but poor BAC. Those classifiers were in practice naïve or quasi-naïve because they classified all or most of the loans as non-default. The reason behind this fact delves into the nature of AUC, which does not measure the final prediction but the scores assigned by the classifier (values between 0 and 1) that represent a probability-like value of belonging to the default class. However, other classification metrics such as BAC (or accuracy, or the F1 score) measure the final predictions and not the scores. By default a classifier transforms the scores into final predictions, by binarizing the scores using a threshold

of 0.5. In our optimization process we found that some classifiers obtained the best AUC measures by assigning low scores (below 0.5) to all or almost all the examples, which eventually turned into naïve or quasi-naïve classifiers. In such cases, it was needed to set the optimum threshold value to maximize the performance of the final prediction of the classifier. In other words, such cases needed an extra step for fine-tuning the threshold value.

To avoid this problem, we decided to use the BAC as the fitness measure instead of AUC. By optimizing BAC, we force the classifier to learn scores that binarize well with a threshold of 0.5. In this way, we avoid the need to optimize the threshold. This is a major gain in the agility of the process because it eliminates one more dimension of the experiment that would otherwise need to be considered (along with other dimensions explored, such as the impact of categorical variables, the impact of textual variables, etc.).

By optimizing the parameters of XGBoost and CatBoost models with respect to the balanced accuracy, we can ensure that the resulting models are robust, generalizable, and capable of making accurate predictions for both positive and negative classes.

Chromosomes: hyperparameters considered

For the parameter tuning with genetic algorithms, we have defined the range of values that will be explored for each hyperparameter. In the following tables (3.2, 3.3), we see the explored hyperparameters for XGBoost and CatBoost, as well as their range of possible values.

Note that although the imbalance proportion of the target classes in the dataset is fixed (84.73% non-default), we decided to optimize the *scale_pos_weight* hyperparameter that controls the balance of positive and negative weights in both boosting algorithms. This way, we could explore if a different proportion of weights performs better than the default/non-default proportion.

XGBoost

Parameter	Min. value	Max. value
ETA (learning rate)	0.001	0.5
<i>max_depth</i>	2	12
<i>gamma</i>	0	10
<i>min_child_weight</i>	0	10
<i>alpha</i>	0.5	10
<i>lambda</i>	0.5	10
<i>subsample</i>	0.7	1
<i>colsample_bytree</i>	0.3	1
<i>scale_pos_weight</i>	0.1	10
<i>n_estimators</i>	2	500

Table 3.2: Ranges of the XGBoost hyperparameters tuned.

CatBoost

Parameter	Min. value	Max. value
<i>learning_rate</i>	0.001	0.5
<i>depth</i>	2	12
<i>l2_leaf_reg</i>	0.5	10
<i>bagging_temperature</i>	0.03	5
<i>random_strength</i>	0	10
<i>scale_pos_weight</i>	0.1	10
<i>iterations</i>	2	500

Table 3.3: Ranges of the CatBoost hyperparameters tuned.

Genetic algorithm configuration

The individuals of the genetic algorithm will be the set of hyperparameters of the models, with each hyperparameter being a different gene. That is, we will have individuals with 10 genes for XGBoost and 7 genes for CatBoost.

Our population will have a size of 150 individuals, and our termination condition will be reaching 20 generations. That is, we generate a total of 3,000 different individuals (hyperparameter configurations), which, taking into account the 5 folds for the cross-validation, means that 15,000 different classifiers will be trained and tested for every execution.

As previously stated, we use balanced accuracy (BAC) as the score metric to optimize the genetic search. More precisely, the fitness value for each individual will be the average balanced accuracy in the validation set of the cross-validation.

The chosen evolutionary strategy is ${}^6(\mu + \lambda)$, in which children and parents are added to produce the offspring. The notation of this kind of evolutionary strategies is as follows:

- μ (*mu*): Number of individuals to select for the next generation.
- λ (*lambda*): Number of offspring generated in each generation.

In our case, these values will be:

- μ (*mu*) = *Population size* = 150 *individuals*
- λ (*lambda*) = $2 * \text{Population size}$ = 300 *individuals*

With this, the computation done in each generation is as follows:

1. Parents selection

We will have a tournament selection with tournament size equal to 2. In other words, the algorithm will make subsets of two individuals and the best of

⁶“*Mu plus lambda*”. Its most common alternative is (μ, λ) or “*Mu comma lambda*”, in which children completely replace parents in the offspring.

each subset will be selected, repeating this tournament process until we have selected 150 individuals that will act as the parents (μ value).

Note that the same individual could be selected more than one time. However, this is not very likely since the tournament size is equal to 2. This means that for an individual to be selected twice, it should be randomly selected against another individual with a lower fitness value for two times.

2. Crossover + Mutation

To generate the 300 individuals in the offspring (λ value) we will apply crossover among the 150 selected parents, employing the two-point crossover method with an 80% probability. This means that for every pair of randomly selected parents, there is an 80% likelihood of them undergoing a crossover, while in the remaining 20% they will not be crossed. In the event of a crossover, two arbitrary points of the parents will be chosen, and the genetic segments (hyperparameters) between these points will be exchanged to create two new children. This process is repeated until we have 150 children (equivalent to 75 crossovers of two parents, each of which generates two new children).

These 150 children will then undergo the mutation process. The method used will be the random resetting mutation with a 20% probability. This implies that each gene (hyperparameter) of every child has a 20% chance of undergoing a mutation, getting a new random value within its range.

After being mutated, the children are combined with the parent individuals, resulting in a generation of 300 individuals (λ value).

3. Offspring selection

Out of the 300 individuals in the offspring candidates, only 150 of them (which is the population size) will progress to the next generation. To this end, we compute the fitness value of the 300 individuals and rank them in decreasing order, so that the top 150 individuals are the parents of the next generation.

In addition, as a side experiment, we will execute the genetic algorithm with the same configuration but replacing the tournament selection with a roulette selection. In this manner, we will be able to analyze the effect of this choice on the search for the best classifier, whether it helps to find a better final value or not. This experiment is reported in Appendix B.

Libraries used for genetic optimization

For the purpose of using a genetic algorithm, we have used the Sklearn-genetic-opt⁷ tool, which is a Python library that allows scikit-learn models hyperparameters tuning and feature selection using evolutionary algorithms. It is an open source library under the MIT software license, mainly maintained by its author, Rodrigo Arenas. The version used for this Bachelor's thesis is 0.10.0, which is the latest version released at the time of the project.

⁷Source: <https://github.com/rodrigo-arenas/Sklearn-genetic-opt>

The Sklearn-genetic-opt library acts as a frontend to make calls to the DEAP⁸ package, which is a Python framework for distributed evolutionary algorithms. Sklearn-genetic-opt also provides extra functionalities such as cross-validation and results visualization.

The tool works by creating a population of candidate solutions, evaluating their fitness using a cross-validation score, applying genetic operators such as selection, crossover and mutation, and repeating this process until a termination criterion is met.

GPU implementations of the classifiers

Both XGBoost and CatBoost support parallelized executions in GPU for lower computation times, so the training phase will be done in GPU for both of them.

For this, we have used a very powerful machine at the time of development of the project. Its specifications are the following:

- CPU: Intel Core i9-12900KS
- GPU: NVIDIA GeForce RTX 4090 24GB
- RAM: 128 GB
- SSD: 2 TB

By taking advantage of the GPU computation for the algorithms, the genetic exploration process achieved execution times stated in Table 3.4, which represents the fitting of 3,000 different classifiers. It is worth noting that the execution times for XGBoost and CatBoost are not directly comparable, as XGBoost made use of parallelization across all GPU cores, whereas CatBoost could not be parallelized due to hardware incompatibilities.

	XGBoost	CatBoost
Quant.	1h 53min 58s	7h 4min 28s
Quant. + Categ.	3h 48min 59s	7h 37min 44s
Quant. + Categ. + Textual	6h 14min 8s	16h 4min 17s

Table 3.4: Execution times of the boosting algorithms.

3.3. Baseline classifiers

Now that we have a good understanding of the dataset and the scoring metrics that we will be using, it is important to compare the performance of our machine learning models against a baseline. A baseline is a simple model that provides a starting point for comparison and can help us to assess the performance of more complex models. In our case, the simplest baseline would be a dummy classifier that assigns every loan to the majority class, that is, the non-default class. We will refer to it as the naïve classifier.

⁸Source: <https://github.com/DEAP/deap>

Naïve classifier

It is straightforward that the accuracy obtained by a dummy classifier is 0.8473, which is the proportion of default instances in our dataset. Since the target class in our dataset is imbalanced, with a much higher proportion of non-default loans, it makes sense that a simple classifier that always predicts the majority class will have a high accuracy. However, as we know, accuracy alone is not enough to measure the performance of imbalanced datasets, and it is important to consider other metrics to get a more complete picture of the model's performance.

Values of F1, precision and recall are 0.0, since the number of TP (true positives) is 0 for the default class.

BAC and AUC take a value of 0.5. Since they are balanced metrics, this same score would be obtained for a dummy classifier that always predicts default. That is, they reflect that the classifier does not discriminate between classes.

XGBoost default classifier

To address the importance of hyperparameter tuning in XGBoost and CatBoost, it is crucial to know the default performance of each of them, that is, the performance of both algorithms with their default parameterizations. The default values of the XGBoost hyperparameters are as in Table 3.5.

Parameter	Default value
ETA (learning rate)	0.3
<i>max_depth</i>	6
<i>gamma</i>	0
<i>min_child_weight</i>	1
<i>alpha</i>	0
<i>lambda</i>	1
<i>subsample</i>	1
<i>colsample_bytree</i>	1
<i>scale_pos_weight</i>	1
<i>n_estimators</i>	100

Table 3.5: XGBoost default parameters.

The performance obtained by XGBoost, just with quantitative and with quantitative and categorical variables, with the out-of-the-box parametrization is as in Table 3.6.

The default parametrization of XGBoost is having a similar behavior to that of the dummy classifier. This can be explained by the accuracy value (very close to the default rate), by the low F1 and recall values, and by the BAC value close to 0.5 (which means that the target classes are not distinguishable).

By adding categorical variables we obtain slightly better results, but still with a similar performance to that of the dummy classifier. This slight improvement can be appreciated, for example, in the AUC score, which improves by 1% just by adding the categorical features.

Metric	Quant.	Quant. + Categ.
BAC	0.5010	0.5019
AUC	0.6306	0.6406
F1	0.0071	0.0130
Precision	0.2789	0.2933
Recall	0.0036	0.0067
Accuracy	0.8464	0.8459

Table 3.6: XGBoost default performance.

CatBoost default classifier

Similarly, we will now analyze the performance obtained from the default parametrization of CatBoost for quantitative and quantitative plus categorical features). The default values of the CatBoost hyperparameters are as in Table 3.7.

Parameter	Default value
<i>learning_rate</i>	0.03
<i>depth</i>	6
<i>l2_leaf_reg</i>	3
<i>bagging_temperature</i>	1
<i>random_strength</i>	1
<i>scale_pos_weight</i>	1
<i>iterations</i>	1,000

Table 3.7: CatBoost default parameters.

Table 3.8 reveals that the results obtained from the default parametrization of CatBoost with quantitative variables are very similar to the ones obtained from XGBoost. We appreciate a similar performance to the dummy classifier one.

By adding the categorical features, CatBoost achieves mixed results. For example, the test AUC score increases from 0.637 to 0.652 by adding them, while the F1 score decreases from 0.006 to 0.005.

Metric	Quant.	Quant. + Categ.
BAC	0.5010	0.5009
AUC	0.6372	0.6519
F1	0.0063	0.0052
Precision	0.3466	0.3264
Recall	0.0032	0.0026
Accuracy	0.8468	0.8469

Table 3.8: CatBoost default performance.

In addition, if we compare XGBoost and CatBoost side by side, we appreciate a very similar behavior when it comes to the usage of categorical variables. Since CatBoost is specifically designed to deal with categorical variables, we expected

enhanced results compared to those of XGBoost, but this is not the case, at least with the default hyperparameter configuration.

Another important conclusion is the need for hyperparameter tuning. Looking at the balanced accuracy of the default classifiers we appreciate a very similar target class separation to that one of the dummy classifier which predicted non-default for all the loans.

Analysis of the results of the boosting algorithms

In this chapter, we analyze the results of the XGBoost and CatBoost classifications after tuning their hyperparameters with the genetic algorithm. Our objective is to examine the impact of parameter tuning on the performance of the algorithms, so that we can reach a conclusion on their importance. Additionally, we will also be exploring the impact of the addition of categorical variables on the results of the classifications to gain a deeper understanding of how this factor contributes to the accuracy of the classifications.

For the analysis of each execution, we have conducted an extensive examination employing a comprehensive set of four different plots. However, they focus on analyzing the evolution of the genetic algorithm (such as the fitness evolution over generations or the model's hyperparameter exploration) and they are quite repetitive over all the executions. Thus, we show an example of a complete analysis in Appendix A, and in this chapter we will focus on analyzing the performance of the classifiers together with their feature importances and SHAP values.

4.1. Quantitative variables

4.1.1. XGBoost

As stated in Appendix A, we see that already from the first generation there is at least one reasonably good individual. This indicates that if there were limitations on time and computational resources and it is still desirable to maintain a satisfactory classifier, the utilization of the genetic algorithm may not be necessary, and a simple generation of 150 individuals might be enough.

Also, the search spaces for the hyperparameters are biased due to the elitism factor. Having elitism through a tournament selection with a tournament size of 2 individuals out of 150 in each generation represents an optimal trade-off between the breadth and depth of exploration. While all parameters are explored across their full range, there are also pronounced peaks, indicating that certain values have produced superior results, which justifies deeper exploration.

Regarding the hyperparameters of the top 50 individuals, there are some hyperparameters that vary very slightly (such as *max_depth*, which only takes values of 4 or 5), while others have high variability (such as *learning_rate*, which ranges from 0.10 to 0.30). The reason for this is either similar performance with different parameter combinations, suggesting that the precise tuning of these parameters is not crucial, or that a similar BAC is produced by a similar combination of these parameters. Regarding the *scale_pos_weight* hyperparameter that controls the balance of positive and negative weights, it stabilizes in the value 5.56, which is similar to the default/non-default proportion of the dataset ($1 - \frac{1}{5.56} = 0.82$).

To elucidate this, we will analyze the scoring metrics of the best classifier achieved in the genetic search (see Table 4.1). The best classifier found has a BAC score of 0.6062 and an AUC of 0.6458, which implies an improvement of over 10% compared to the dummy and default-parameterized classifiers. However, both the accuracy and the precision have been reduced, which implies that the improvement in the balanced metrics comes from enhancing the prediction of the default class. This indicates that the tuned classifier makes a better distinction between the target classes, which is crucial when addressing credit risk. This can also be appreciated in the huge improvement in the F1 score, which also takes into account both target classes.

Metric	Naïve	Default XGBoost (quant.)	Best XGBoost (quant.)
BAC	0.5	0.5010	0.6062
AUC	0.5	0.6306	0.6457
F1	0.0	0.0071	0.3192
Precision	0.0	0.2789	0.2138
Recall	0.0	0.0036	0.6300
Accuracy	0.8473	0.8464	0.5896

Table 4.1: XGBoost scoring metrics (quantitative variables).

In terms of feature importance, Figure 4.1 shows that *fico_n* is by far the most significant feature for the top-performing loan classifier. In fact, its importance surpasses that of the subsequent features by more than double.

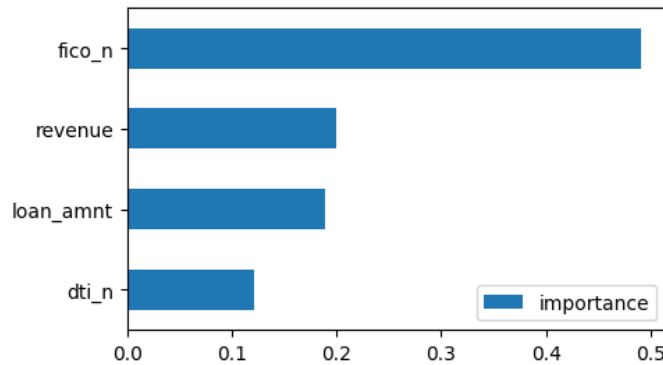


Figure 4.1: XGBoost feature importance (quantitative variables).

This makes sense if we look at the following plot showing the SHAP values (Figure 4.2), since *fico_n* is the feature with better separation of colors and with a higher impact of its values on the final prediction. In particular, according to the plot, the highest FICO values are very good indicators that the loan will be repaid.

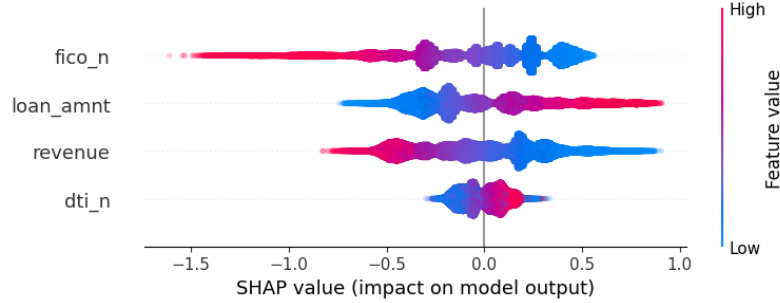


Figure 4.2: XGBoost SHAP values (quantitative variables).

The SHAP values for the variables *fico_n*, *loan_amnt*, and *revenue* are clearly defined, however, an unexpected outcome is observed with the *dti_n* variable.

In the case of *fico_n*, the red color indicates that a higher value of *fico_n* (which reflects an individual's credit experience), has a greater positive impact on the likelihood of loan repayment, while a lower value has a greater negative influence. The same happens for *revenue*, although its impact on the target variable is less pronounced. This observation is logical, as an individual with extensive credit experience and high income is more likely to repay a loan compared to the opposite scenario.

The opposite relationship holds true for *loan_amnt*. A high value of the requested loan amount has a greater negative impact on the likelihood of repayment, which is a reasonable outcome, since a larger loan amount makes it more challenging to repay the loan.

Finally, the impact of *dti_n* is much lower, as its values are closer to zero. In addition, the association of high or low values with impact is less clear, as we see the colors mixed in different parts of the figure. This indicates that, according to the SHAP values, it is not clear how the DTI values contribute to the default prediction.

4.1.2. CatBoost

This CatBoost execution should have similar results to the XGBoost one, as we are still not adding categorical variables, which is the differentiating factor favoring CatBoost.

The fitness evolution over generations is quite similar to the one of XGBoost. Again, from the first generation we obtain a very good individual, although it is far from being the best one taking into account that a 1% improvement of the BAC score can involve the correct prediction of several hundreds of loan requests.

The hyperparameter exploration is again biased due to the elitism factor. Nevertheless, we can also appreciate a complete exploration of all the hyperparameter ranges. These peaks in the exploration represent the driving of the search towards

the best individual.

The values of the best-found hyperparameter combination vary from that of XGBoost. For instance, the learning rate is increased from 0.10 to 0.25, while the tree depth is decreased from 5 to 2. Also, the number of iterations suffers from a substantial decrease from 500 to 300. All these values are correlated, since a higher learning rate may require lower tree depth and fewer iterations to achieve its best performance. As in the previous XGBoost execution, the *scale_pos_weight* hyperparameter stabilizes in the value 5.38, which is similar to the default/non-default proportion of the dataset ($1 - \frac{1}{5.38} = 0.81$).

Table 4.2 illustrates the scoring metrics of the best-found CatBoost classifier, and it is compared against the naïve classifier and the CatBoost default configuration. We can see that the BAC, which is the optimized metric, has improved by more than 10%. As it occurred in the previous XGBoost analysis, this improvement comes from the enhanced classification of the default class, which implies a recall increase and a precision decrease, resulting in a better F1.

Metric	Naïve	Default CatBoost (quant.)	Best CatBoost (quant.)
BAC	0.5	0.5010	0.6063
AUC	0.5	0.6378	0.6455
F1	0.0	0.0063	0.3198
Precision	0.0	0.3466	0.2162
Recall	0.0	0.0032	0.6136
Accuracy	0.8473	0.8468	0.6013

Table 4.2: CatBoost scoring metrics (quantitative variables).

Regarding the feature importance of the best classifier (Figure 4.3), we can see that the importance of *fico_n* is much lower than in XGBoost, being reduced from almost 50% to 35%. The *loan_amnt* and *revenue* have experienced significant increases in importance, rising from approximately 20% to around 30%, whereas the *dti_n* has been nearly halved from its previous value.

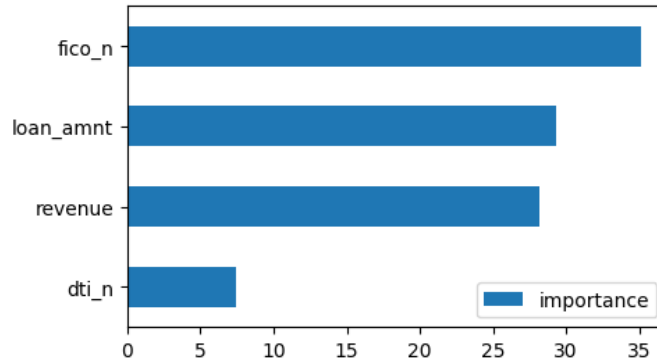


Figure 4.3: CatBoost feature importances (quantitative variables).

The only noticeable difference in the SHAP values (Figure 4.4) is the magnitude of the impact on the model output. For instance, the leftmost part of *fico_n* almost

reaches the -2.5 value, while for XGBoost it stopped at the -1.5 value. However, their proportions are equivalent to the XGBoost SHAP values (Figure 4.2).

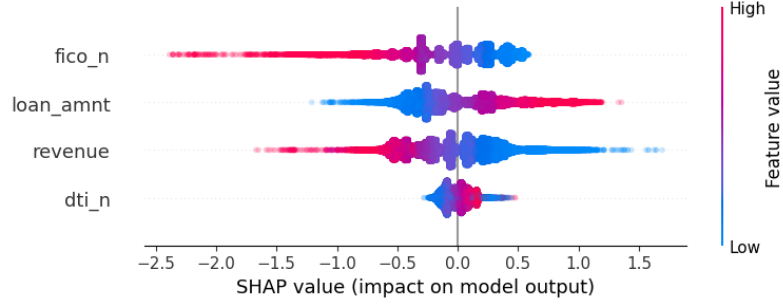


Figure 4.4: CatBoost SHAP values (quantitative variables).

By analyzing the results of the boosting algorithms, we have seen that both exhibit similar performance in terms of BAC and AUC, although their configuration to reach this performance is slightly different. These differences in performance also imply different learning patterns from the data. While XGBoost has a higher recall, CatBoost exhibits better precision, resulting in an overall higher F1 score.

4.2. Quantitative and categorical variables

4.2.1. XGBoost

Although the fitness evolution of this execution has a very similar shape to the previous ones, there is an overall quantitative improvement in the fitness value. This is the expected behavior, since the inclusion of categorical features should add new information that can be exploited by the classifiers.

Regarding the hyperparameter exploration, the conclusions are similar to the previous ones. We appreciate a full exploration of the hyperparameter ranges with significant peaks due to the elitism factor, which drives the exploration towards the best classifiers.

The best-found hyperparameter combination when adding the categorical features to XGBoost varies significantly from that of XGBoost without categorical features. For instance, the learning rate is stabilized at 0.25, while without categorical features it was in the 0.10 range. Also, the *max_depth* converges in the value of 2, while in the previous execution it ranged between 4 and 5. Additionally, the *gamma* value has shown a decrease, shifting from approximately 10 to around 6. As it occurred without the categorical features, the *scale_pos_weight* hyperparameter stabilizes in the value 5.82 for the top-50 individuals, which is similar to the default/non-default proportion of the dataset ($1 - \frac{1}{5.82} = 0.83$).

The last column of Table 4.3 illustrates the scoring metrics of the best XGBoost classifier found with quantitative and categorical variables. We can appreciate a quantitative improvement in all of them compared to the best XGBoost found without categorical variables, which indicates that the addition of these variables

enhances its predictive performance. Specifically, we can see that the BAC has improved by 0.92%, AUC has improved by 1.18% and F1 has improved by 0.74%. Both the precision and the recall have increased, which means that the classification of both classes is enhanced when adding categorical features.

Metric	Naïve	Default XGBoost (quant. + categ.)	Best XGBoost (quant.)	Best XGBoost (quant. + categ.)
BAC	0.5	0.5010	0.6062	0.6154
AUC	0.5	0.6306	0.6457	0.6575
F1	0.0	0.0071	0.3192	0.3266
Precision	0.0	0.2789	0.2138	0.2168
Recall	0.0	0.0036	0.6300	0.6614
Accuracy	0.8473	0.8464	0.5896	0.5835

Table 4.3: XGBoost scoring metrics (quantitative and categorical variables).

As for the feature importance, results in Figure 4.5 highlight that *fico_n* continues to be the most influential factor in determining the efficacy of the classifier by a large difference.

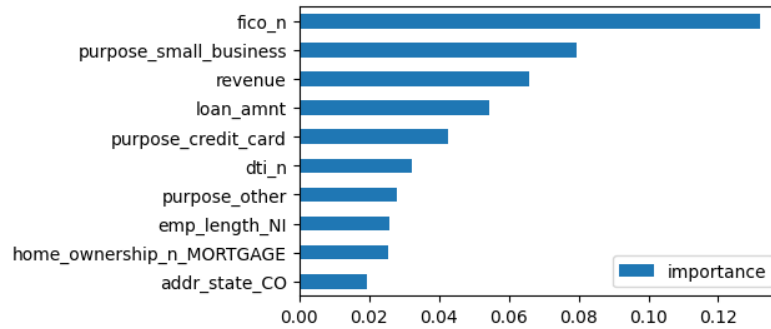


Figure 4.5: XGBoost feature importances (quantitative and categorical variables).

Surprisingly, two categorical features, “small business” purpose and “credit card” purpose, emerge as the second and fifth most important features respectively, both of them over some of the quantitative features. Both of these features are among the most frequently reported loan purposes and, according to the feature importance measure, they have a great impact on the performance of the classifier.

Furthermore, in the XGBoost execution excluding categorical features, the *fico_n* variable exhibited a feature importance of 50%. However, its importance has significantly decreased to 13.5%, and the overall importance is now more evenly distributed among various features.

Figure 4.6 depicts the SHAP values for XGBoost with the inclusion of categorical features, revealing comparable results to the SHAP values obtained without categorical features. However, we do see an improvement in the clarity of use of *dti_n*, since it now clearly separates the impact of high and low values.

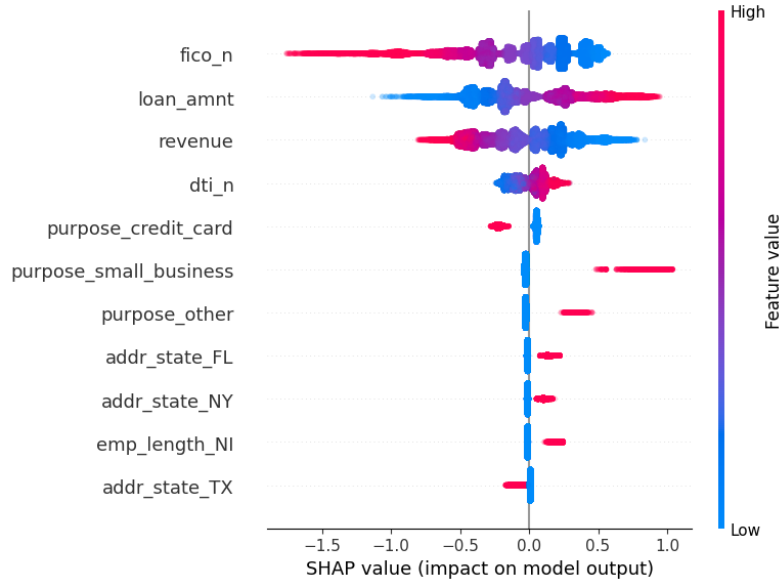


Figure 4.6: XGBoost SHAP values (quantitative and categorical variables).

The improvement in the classifier that was achieved by adding categorical variables has allowed for *dti_n* to provide more meaningful information in a consistent manner, even though its impact on the final prediction is still lower than other quantitative features like *fico_n*, *loan_amnt*, and *revenue*.

If we analyze the results for categorical features¹, we can see that the “credit card” purpose and the fact of living in TX slightly contributes to predicting that the loan will be repaid. On the other hand, “small business” and “other” purposes, “no information” of employment length and the fact of living in some states such as FL and NY contribute to predicting that the loan will be defaulted. Among these, the “small business” purpose is the one with a higher impact on the prediction.

4.2.2. CatBoost

Both the fitness evolution over generations and the hyperparameter exploration in this execution mirror the pattern observed in the previous XGBoost run. From the first generation we observe the emergence of a promising individual, and the hyperparameter exploration reveals peaks at the optimal values, while still conducting a full-range exploration.

The hyperparameters obtained for the top 50 individuals are very similar to the ones obtained in the first CatBoost execution (section 4.1.2). The hyperparameter that exhibits the most significant variation is the depth, which has been increased

¹Note that for categorical features we will just have a 1 or a 0, 1 meaning that the given variable is of the given category, and 0 meaning that the considered individual is not of the given category. This is why their SHAP value will only have intense blue and red colors. Also, note that all records in a dataset are of just one certain category for each variable (e.g. a loan cannot be of purpose “small business” and “credit card”), so just one category will be set to 1 and the rest to 0. This is why there is a high concentration of instances near the 0 impact value with a low feature value, which represents the 0.

from 2 to 3. This outcome is reasonable considering the addition of categorical features, as deeper trees may be necessary to enhance performance with more variables. Again, the *scale_pos_weight* hyperparameter stabilizes in the value 5.38 for the top-50 individuals, which is similar to the default/non-default proportion of the dataset ($1 - \frac{1}{5.38} = 0.81$).

The last column of Table 4.4 shows the scoring metrics of the best CatBoost classifier found with quantitative and categorical variables. All the metrics undergo a similar improvement to that of XGBoost when adding categorical features. Both the precision and the recall are improved, so the classification of both target classes is enhanced.

Metric	Naïve	Default CatBoost (quant. + categ.)	Best CatBoost (quant.)	Best CatBoost (quant. + categ.)
BAC	0.5	0.5010	0.6063	0.6153
AUC	0.5	0.6378	0.6455	0.6574
F1	0.0	0.0063	0.3198	0.3280
Precision	0.0	0.3466	0.2162	0.2229
Recall	0.0	0.0032	0.6136	0.6208
Accuracy	0.8473	0.8468	0.6013	0.6115

Table 4.4: CatBoost scoring metrics (quantitative and categorical variables).

This observation implies that even after incorporating the categorical features, the performance of XGBoost and CatBoost remains remarkably similar. This outcome is disappointing as we expected CatBoost, being specifically designed to handle categorical features, to leverage them more effectively, resulting in superior results.

Figure 4.7 displays the feature importance for CatBoost with the inclusion of categorical features. Comparing it to the feature importances without categorical features (Figure 4.3), we observe that the quantitative variables maintain similar proportions, but with lower magnitudes. Among them, the most significant decrease in importance is observed in the *revenue* feature (-10%). Turning to the categorical variables, *purpose* emerges as the most influential, with a 12% importance, surpassing the 6% importance of the quantitative variable *dti_n*. The rest of the categorical variables do not contribute much to the decision of the classifier.

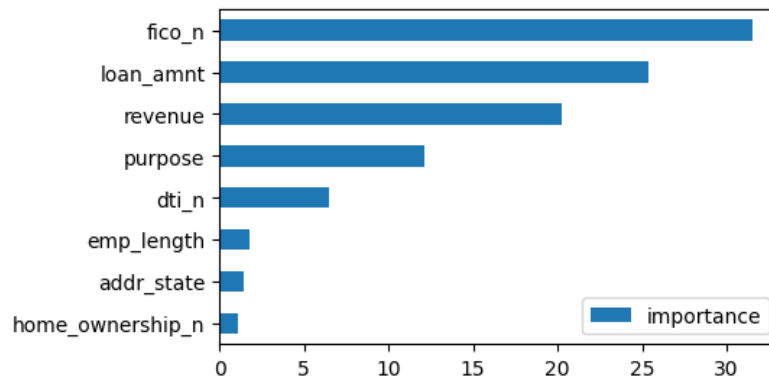


Figure 4.7: CatBoost feature importances (quantitative and categorical variables).

Lastly, Figure 4.8² shows similar SHAP values to the XGBoost with categorical variables results (Figure 4.6) in terms of the quantitative variables. As in the XGBoost SHAP values, the inclusion of categorical variables in CatBoost facilitates a more distinct utilization of the *dti_n* variable. This enables a clear differentiation between higher and lower values, enhancing interpretability.

With respect to the categorical variables, *purpose* is the one with the highest positive impact on the model output. In fact, *purpose* and *loan_amnt* are the variables with the highest positive impact on the model output, meaning that they are especially helpful when predicting default loans.

Regarding the categorical variables, *purpose* emerges as the feature with the most pronounced positive impact on the model output. In fact, *purpose* and *loan_amnt* are the variables that exhibit the highest positive influence, indicating their particular significance in predicting default loans, which is the positive class.

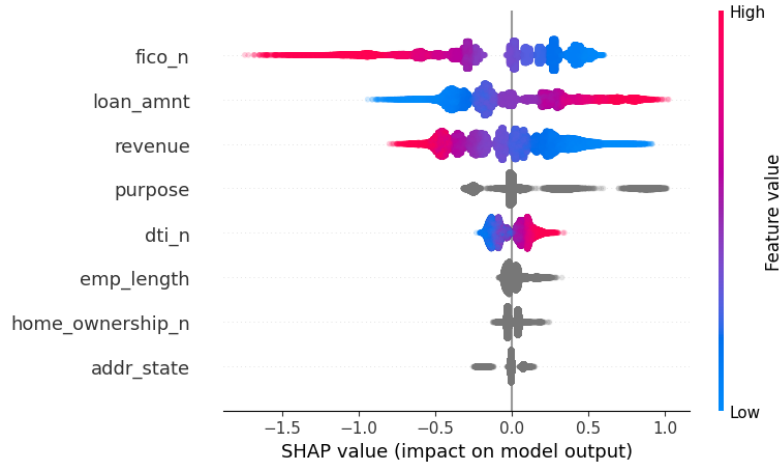


Figure 4.8: CatBoost SHAP values (quantitative and categorical variables).

Again, the results have shown almost identical performance of XGBoost and CatBoost when adding categorical variables. Although CatBoost is said to better handle categorical variables, it has not been the case in our study, as its performance is very similar to the XGBoost one in terms of BAC (our fitness score), which treated the categorical variables with one-hot encoding.

Moreover, we appreciate again different learning patterns from the data. As occurred previously with quantitative variables, XGBoost has a better recall, while CatBoost has superior precision. This discrepancy leads to a higher F1 score for CatBoost.

²Note that the SHAP values for CatBoost are aggregated by feature rather than by category, as seen in XGBoost. This distinction is the reason why CatBoost SHAP values do not exhibit high or low values for specific categories. Instead, they only reflect the overall impact on the model output and the data distribution within each range.

4.3. Conclusions

After conducting a comprehensive analysis of the XGBoost and CatBoost algorithms results, we have gained several important insights and come to several conclusions.

Firstly, we found that tuning the parameters of the boosting algorithms using a genetic algorithm significantly improved its performance. The optimized hyperparameters showed a great improvement in the results compared to the default parameters, demonstrating the effectiveness of hyperparameter tuning in this kind of algorithms.

Additionally, we found that incorporating categorical features, in addition to quantitative features, greatly improved the performance of the classifications. The BAC of the best classifiers found increased by an average of 0.91%³, which, although it may look like a small improvement, is significant considering the complexity of the classification problem and the large number of instances in the dataset. The inclusion of these categorical features not only provides a more thorough classification, but also leads to an enhanced use of the *dti_n* feature.

By looking at Table 4.5, which represents a summary of all the scoring metrics obtained, the main difference is that XGBoost has a higher recall but a lower precision than CatBoost, which implies their accuracy difference. This indicates that when CatBoost predicts a defaulted loan, it is more likely to be correct. On the other hand, XGBoost is better at identifying default instances overall.

Therefore, the choice on which model is better depends on the requirements of the specific task. If the main goal is to avoid default loans, then XGBoost might be a better choice, while CatBoost may be a better choice to maximize economic profit. However, a combination of both models could produce enhanced results.

Metric	Quantitative		Quantitative + categorical	
	Best XGBoost	Best CatBoost	Best XGBoost	Best CatBoost
BAC	0.6062	0.6063	0.6153	0.6154
AUC	0.6457	0.6455	0.6575	0.6574
F1	0.3192	0.3198	0.3266	0.3280
Precision	0.2138	0.2162	0.2168	0.2229
Recall	0.6300	0.6136	0.6614	0.6208
Accuracy	0.5896	0.6013	0.5835	0.6115

Table 4.5: Scoring metrics summary: XGBoost *vs* CatBoost.

Moreover, CatBoost is usually said to better handle categorical variables, but this has not been the case if we look at the balanced metrics used for guiding the model parametrization. Instead, it has almost identical results to XGBoost, which handles categorical variables through one-hot encoding. In fact, due to the native use of categorical variables by CatBoost, fine grain is lost in explainability both of feature importance and SHAP values, as they are aggregated by feature rather than by category.

³Average improvement for XGBoost and CatBoost.

Chapter 5

Processing textual features with transfer learning from the BERT model

In the previous chapter, we analyzed the positive impact of hyperparameter tuning on the XGBoost and CatBoost algorithms. The results obtained are much better than those of the baseline classifiers with the default hyperparameters. Also, the addition of categorical variables improved the outcome of the algorithms. With the aim of improving these results, now we turn our attention towards textual variables, specifically the loan description.

The description of the loan is an important piece of information that lenders consider when evaluating loan applications. It gives further information about the loan than the one obtained from continuous numbers or categories. However, automatically analyzing these descriptions can be a challenging task due to the unstructured nature of the text.

Our goal in this chapter is to apply transfer learning from the BERT model to predict whether a loan borrower will default or not. This task presents a significant challenge, as it differs substantially from traditional sentiment analysis tasks in which the objective is to predict whether a text is positive or negative. Instead, we must analyze the semantic content of the text to determine whether it contains any indications of potential default risk. This requires a deep understanding of the language used in loan descriptions and the ability to identify subtle linguistic patterns that may indicate financial risk.

In addition to the linguistic challenges of predicting loan defaults using NLP, there is also the inherent human factor of dishonesty or deception, as famously quoted by Dr. House, *“everybody lies”*. Loan applicants may not always provide accurate information in their loan descriptions, which can complicate the task of predicting default risk. This may hinder the importance of combining NLP techniques with other forms of data analysis, such as credit scores and financial records, to improve the accuracy of our predictions.

It is important to mention that BERT, as a LLM, is a black-box model. Consequently, we are unable to discern precisely what the model is learning from the text. It may be exploiting the semantic content, or it might also be extracting information from syntactic aspects, or other aspects such as text length, typos, or readability.

5.1. Default prediction using BERT

As previously mentioned, we will be using the BERT model for processing the text, which has proved to be the highest-performing large language model for classification based on the conclusions of several studies (Kriebel and Stitz, 2022; Stevenson et al., 2021).

The goal of the BERT neural network is to predict the probability of belonging to the default class, derived from processing the textual description of the loan. These default probabilities, ranging between 0 and 1, serve as indicators of how risky the loan is based on the text written in its description. By obtaining the default probability instead of relying on binary classification, we gain additional information that can be further exploited. Subsequently, these probabilities will be incorporated into the quantitative variables of the boosting algorithm and we will analyze the classification improvement by adding the descriptions processing by BERT.

5.1.1. Training the BERT model

As explained in Chapter 2, BERT is a pre-trained transformer-based Large Language Model. Therefore, transfer learning is needed to fine-tune the pre-trained model to our specific problem. For this, we will use TensorFlow, a popular open-source machine learning library developed by Google that provides a set of tools for implementing and optimizing deep learning algorithms.

The BERT model used¹ has the following characteristics: L=12 hidden layers (i.e., Transformer blocks), a hidden size of H=768, and A=12 attention heads.

The neural network was set up with a batch size of 32 and trained for 25 epochs. Its loss function was the weighted binary cross-entropy, which measures the difference between the predicted probabilities and the true binary labels of the input data taking into account class weights². All the internal hidden layers are dense layers using the ReLU³ activation function. Additionally, to obtain the probability of belonging to the default class, the last layer of the neural network was configured with a single neuron and a sigmoid⁴ activation function.

We have introduced additional layers to the BERT configuration to enhance its learning capability for our specific task. Moreover, we explored various configurations for the neural network after the BERT architecture to fine-tune its performance. Specifically, we experimented with the following configurations:

- Using a first extra dense layer of 128, 256 or 512 neurons.
- Using an optional second extra dense layer of 128 neurons.
- Adding a dropout layer before or after all the extra layers.

¹Source: https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4

²Considering class weights is crucial given the imbalance of our dataset.

³The Rectified Linear Unit (ReLU) activation function outputs the maximum of zero and the input value, “activating” the neuron if the input is positive.

⁴The sigmoid activation function introduces non-linearity and maps the input values to a range between 0 and 1, facilitating binary classification tasks.

- Considering a dropout percentage of 0%, 0.10%, 0.20% or 0.30% for all the dropout layers.
- Adding an early stopper to the neural network that stops the learning process if there is no loss improvement in three epochs in a row.

Therefore, the resulting diagram of the neural network architecture exploration is as in Figure 5.1.

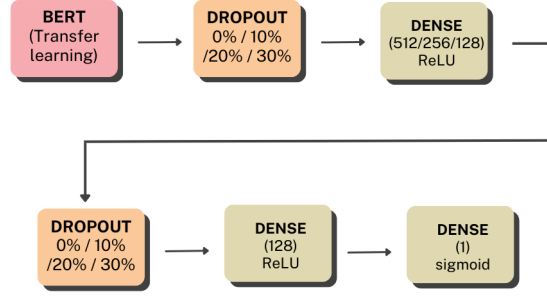


Figure 5.1: Diagram of the BERT neural network architecture.

Note that for making the BERT default prediction we have to reproduce the exact same folds as in the k -fold cross-validation of the boosting algorithms. The boosting algorithms will be trained in each iteration with the *BERT_pred* variable, and we should avoid BERT from training with a fold's validation data. This would not represent the real performance of the granting model, since validation data of the fold would include BERT predictions from its training phase, for which the real label is known.

To avoid this problem, we use the data as shown in Figure 5.2. This diagram consists of five steps:

1. Description extraction: Textual descriptions are extracted from the original dataset to be processed.
2. Folds generation: The exact same folds as in the boosting algorithms are generated. This is done by dividing the data into a train set (green color) and a test (purple color) set. Each fold gets a different test set.
3. Optimization of the neural network architecture: The train set obtained in the previous step is divided into a 70% train subset (light green color) and a 30% test subset (dark green color). The neural network is trained with the previously mentioned configurations on the train subset (light green color), and is tested by predicting the test subset (dark green color).
4. Default prediction: The optimal configuration obtained in step 3 (lower value of weighted binary cross-entropy in the test subset) is trained with the train and test subsets (light and dark green colors) and is used to predict the default probabilities of the test set obtained in step 2 (purple color). These are predictions of unseen data in the training phase, which build the *BERT_pred* values of the current fold.

5. *BERT_pred* integration: Once the *BERT_pred* values of all the folds are generated in step 4, they are grouped together and incorporated into the original dataset as a quantitative variable.

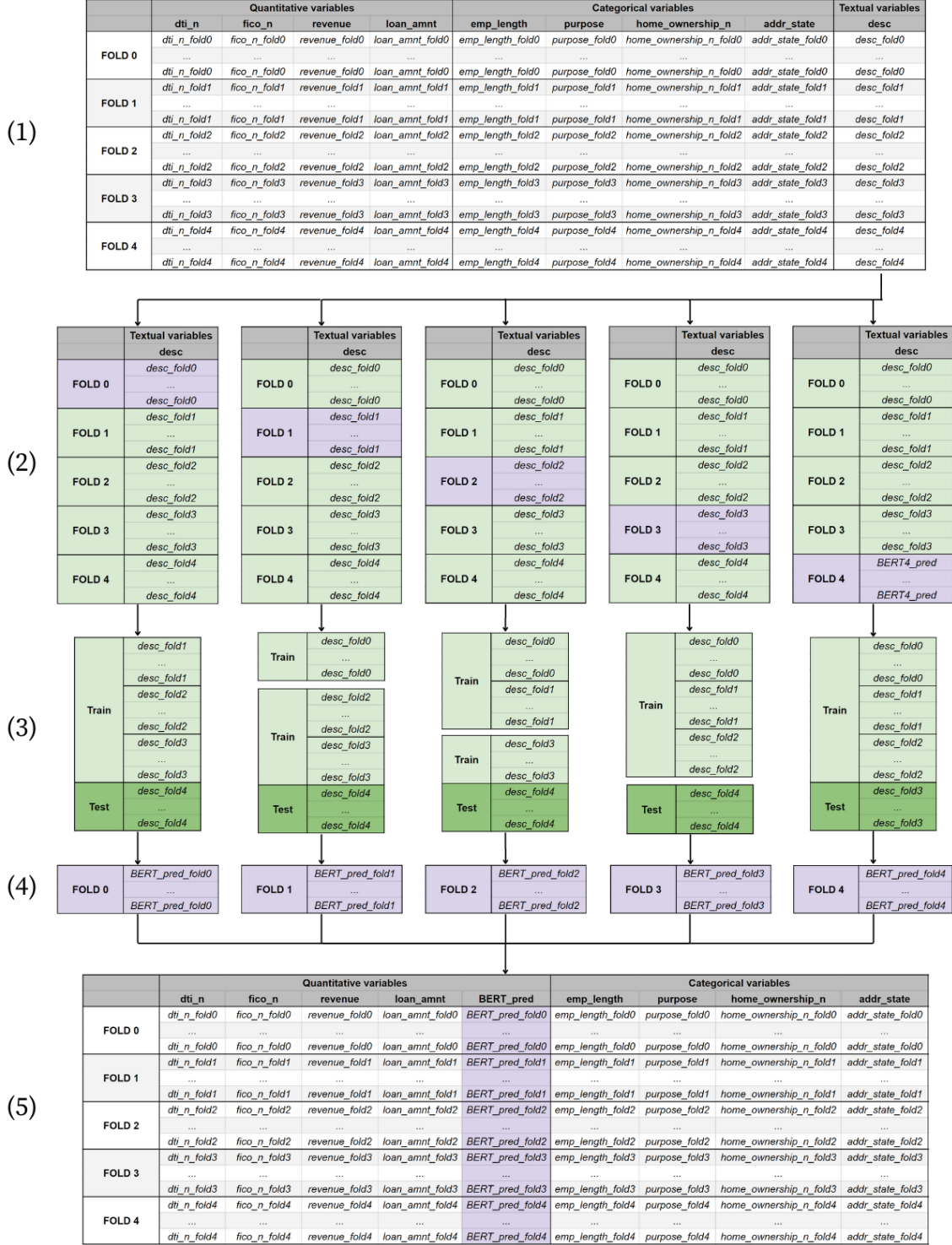


Figure 5.2: BERT training diagram.

As a result of the neural network architecture optimization, we explored 96

configurations of the neural network for each of the five folds. The resulting optimal configuration for each fold was as shown in Table 5.1⁵⁶.

	Neurons in 1st dense layer	Use 2nd dense layer (128 neurons)	Dropout %	Dropout position	Usage of Early Stopper
Fold 0	512	Yes	0%	Before	Yes
Fold 1	256	Yes	0%	Before	Yes
Fold 2	512	Yes	0%	Before	Yes
Fold 3	512	Yes	0%	Before	No
Fold 4	128	Yes	0%	Before	Yes

Table 5.1: BERT optimal configuration by fold.

While the optimal configurations of different folds use different number of neurons in the first dense layer, they all use the second dense layer which has a fixed number of 128 neurons. Regarding the dropout layers, the optimal configuration does not use dropout in any of the folds. Consequently, the parameter controlling the placement of dropout before or after the dense layers becomes irrelevant. Finally, all of them but fold number 3 made use of the early stopper.

By applying this workflow, we achieve a complete separation between the BERT processing and the boosting algorithms, effectively decoupling the two processes entirely. Therefore, they can be done in parallel. This reflects the practical application of the granting model in a production environment. When presented with a new loan request, the model would execute the following steps:

1. BERT prediction of the default probability based on the loan description.
2. Incorporation of this probability to the rest of the variables and classification of the loan into default or non-default with the boosting algorithms.

5.2. Analysis of BERT predictions

In this section, we aim to analyze the predictions made by BERT. As already stated, the *BERT_pred* variable indicates the probability assigned by BERT to a loan being classified as default, derived from the loan description processing.

First, we will take a look at the *BERT_pred* distribution density to get an idea of how well are the target classes separated based on this variable. Figure 5.3 shows a large peak around the 0.5 value for both non-default (blue color) and default (orange color) classes, meaning that BERT is not able to discern if those descriptions are likely to be default or not.

⁵Note: Cells with “Yes” or “No” mean that the optimal configuration did or did not make use of that layer, respectively.

⁶Note: The “Dropout position” column indicates whether the dropout layers were placed before or after the dense layers.

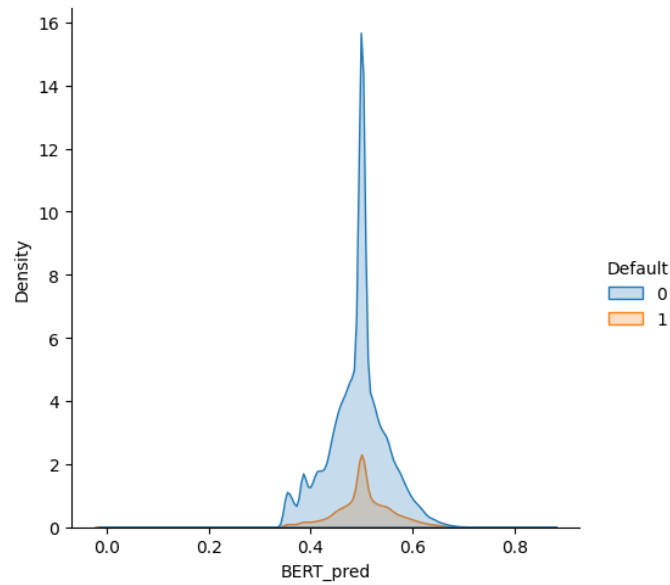


Figure 5.3: $BERT_pred$ distribution.

However, it is very hard to see the proportion of default/non-default loans for the different ranges of the $BERT_pred$ values. Therefore, Figure 5.4⁷ represents on its Y axis the proportion of default/non-default for the $BERT_pred$ ranges on the X axis, grouped every 1% of default probability (bins of size 0.01).

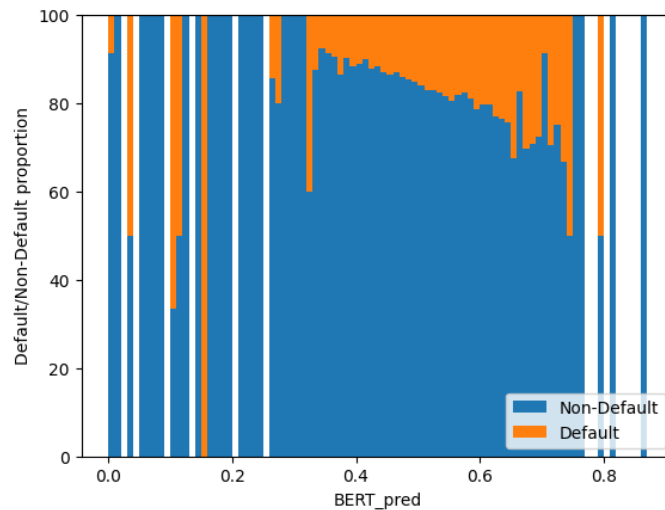


Figure 5.4: Default/Non-Default proportion by $BERT_pred$ range.

Now it becomes clearer that as the $BERT_pred$ value increases, the proportion of Defaults also increases. This is the expected behavior, as a higher $BERT_pred$ value indicates a higher probability of default. There are some ranges outside the central part (<0.3 and >0.7) that have few instances and therefore, their bins do

⁷White spaces mean that there are no descriptions with a default probability within that 1% range.

not provide much information. Nevertheless, upon initial observation, this is a very positive result considering the complexity of the problem, since we are trying to predict loan default based solely on its textual description.

To get an idea of how BERT is assigning these default probabilities, Tables 5.2 and 5.3 show the five descriptions with lower and higher *BERT_pred* values, respectively. That is, the descriptions that BERT identifies as least and most risky. At first sight, we can see that those that get a higher BERT score contain typos (e.g. “dept”, “consolidate”, “consilation”) and poorly descriptive text. Interestingly, these descriptions mainly describe debt consolidations without much detail. On the other hand, descriptions with lower BERT default predictions are much more detailed and contain numeric values that enable a deeper understanding of the purpose and the condition of the borrower. These purposes described include again debt consolidations, but also others such as credit card payment.

Moreover, we can appreciate that 2 out of the 5 descriptions with a higher *BERT_pred* value are correctly predicted as highly risky, while for the lower values, the five of them are correctly predicted with a low probability of default.

Description	<i>BERT_pred</i>	Real
“One of my credit card companies Washington Mutual that I had been paying faithfully, on time, not going over my limit, with online bill pay, sent a notice with a statement a few months ago notifying me that if I did not opt out in writing by a certain date my interest rate would be raised. [...] I owed \$13,208.97 on this card before I made my last payment of 405.00.”	4.5×10^{-15}	Non-default
“1. I plan to use the funds to consolidate debt. [...] 3. My monthly budget is \$3,080 (including the Lending Club loan): Rent \$800; Utilities \$150; Car loan \$345; Gas \$140; Insurance health and auto \$150; Lending Club loan \$655; Food \$600; Misc; \$200; Gym \$40.”	5.7×10^{-13}	Non-default
“Swapping a 17% APR for 6.03%”	9.5×10^{-10}	Non-default
“This loan is for the sole purpose of paying off all of my credit cards. The credit card interest charges are such that it’s difficult (feels like impossible) to ever get them payed off; This loan would make it affordable/possible for me to be out of consumer debt after the 3 years (possibly sooner);”	4.9×10^{-8}	Non-default
“In need of funds to pay off some bills as well as minor improvements to house and yard. I have an extremely secure career, and maintaining my credit worthiness is important to me.”	8.1×10^{-8}	Non-default

Table 5.2: Top-5 descriptions with lower *BERT_pred* values.

Description	<i>BERT_pred</i>	Real
<i>"I have been employeed with same company for the last 15 yrs. This loan will help me consalidate some dept."</i>	0.8657	Non-default
<i>"This loan is just to consolidate some misc bills that have accumulated with medical, holidays, etc."</i>	0.8103	Default
<i>"Loan being used to consolidate bills. My credit is solid and I always pay off my debits."</i>	0.7994	Non-default
<i>"My Loan is for consilation of all debt in to one"</i>	0.7495	Default
<i>"Help to pay off bills and home repairs. Help to pay off debt and home repairs"</i>	0.7479	Non-default

Table 5.3: Top-5 descriptions with higher *BERT_pred* values.

As a quantitative measure, the final predictions found in the *BERT_pred* variable have a weighted binary cross entropy of 0.1779, which was the loss used to train the neural network. This is a fairly low value, suggesting that, on average, the model is relatively accurate in predicting the probability of belonging to the default class.

However, without further context or comparison to other models, it is challenging to determine the significance of this loss value. Therefore, we will now assess the performance of the default probabilities (*BERT_pred* variable) in isolation, serving as a baseline for comparison. This evaluation will employ the scoring metrics seen in Chapter 4. Since we are dealing with probabilities, which range from 0 to 1, we will set a threshold of 0.5. Descriptions with a *BERT_pred* value higher than 0.5 will be categorized as default, while descriptions with a *BERT_pred* value lower than 0.5 will be classified as non-default. By employing this threshold-based approach, we can evaluate the effectiveness of the default probabilities alone. The results are presented in Table 5.4.

Metric	<i>BERT_pred</i> score
BAC	0.5411
AUC	0.5612
F1	0.2563
Precision	0.1819
Recall	0.4338
Accuracy	0.6156

Table 5.4: *BERT_pred* metrics with 0.5 threshold.

It can be seen that it does not perform very well in classifying the positive class (i.e., Default). This can be observed by looking at the low values of precision and recall. Therefore, taking into account that its accuracy is similar to the best one we had in Chapter 4, its strength relies in better classifying the majority class (i.e. Non-Default). However, just by binarizing the *BERT_pred* variable with a 0.5 threshold, BAC and AUC scores are over 0.5, indicating a fair class distinction. Nevertheless, these scores are worse than the ones obtained from the boosting algorithms only with quantitative variables, which achieved a BAC of 0.6062 and an AUC of 0.6456.

By combining the *BERT_pred* variable with other variables, we expect the boosting algorithms to find relationships among them, allowing for a richer use of the *BERT_pred*. For instance, we hypothesize that a higher *BERT_pred* value, particularly when the purpose variable is set to “other”, holds greater significance compared to other purposes due to the lack of concreteness of the “other” purpose.

5.3. Explainability of textual information

The BERT model that predicts the percentage of default from the description is a black box model, meaning that it is not explainable and we are not able to know the way it is learning to reach its outputs. To partially solve this problem, we will use two approaches to try to understand both the semantic content of the descriptions and how the BERT score relates with the rest of the variables in the dataset.

5.3.1. Exploring the text descriptions with BERTopic

First, we are going to analyze the content of the descriptions using topic modeling, which involves grouping them into the main topics. This analysis serves as an a priori examination of the textual variable. We are going to use a topic modeling tool based on the BERT model so that it extracts information in a similar way as the BERT model we use to predict the scores. This technique is called BERTopic.

BERTopic⁸ is a powerful library for topic modeling and clustering that uses the BERT language model presented by Grootendorst (2022). Introduced as an enhancement to traditional topic modeling methods, BERTopic leverages the contextualized word embeddings generated by BERT to capture the semantic meaning of words in a document. The BERTopic algorithm consists of the following steps:

1. Preprocessing text data by tokenizing the documents into words or subwords.
2. BERT is used to obtain contextualized word embeddings for each token, which capture semantic relationships between words.
3. Dimensionality is reduced through the UMAP⁹ technique.
4. Clustering is performed with the HDBSCAN¹⁰ technique. Each text is assigned a probability of belonging to each topic, and the highest one determines its assigned topic.

The result of running BERTopic is a set of clusters, each representing a different topic in the text data. By default, BERTopic is configured to create a cluster if it contains 10 or more descriptions. These topics are assigned numerical labels, ordered

⁸Source: <https://github.com/MaartenGr/BERTopic>

⁹Uniform Manifold Approximation and Projection. It preserves the local and global structure of the data, making it suitable for visualization and clustering tasks.

¹⁰Hierarchical Density-Based Spatial Clustering of Applications with Noise. It works well for topic modeling, as it can identify clusters of varying sizes and also detect outliers or noise.

according to their prevalence within the corpus. Additionally, each topic gets a set of keywords, which represent the most prevalent and distinctive words within their respective clusters.

The 119,004 total descriptions in our dataset are clustered into 940 topics by BERTopic after removing the stop words. It is noteworthy that 56,189 descriptions (47.2% of the total descriptions) are not clustered. This could be because they are so detailed that they cannot be grouped with other descriptions, or because they are so simple that they cannot be categorized into a topic.

Among the topics, the largest one has 2,328 descriptions, of which 2,248 have exactly the text *“Debt consolidation”* (96.6% of the descriptions in this topic). This explains that the two most representative keywords are *“consolidation”* and *“debt”*, while most of the other keywords are typos: *“consolidation”, “debt”, “consolidationnull”, “reorganization”, “yobrax”, “consolidationring”, “consolidationj”, “consolidationy”, “loadn”, “refief”*.

The second largest topic follows closely the first one, with 2,302 descriptions. This one does contain business-related words: *“business”, “equipment”, “inventory”, “marketing”, “expand”, “sales”, “capital”, “products”, “website”, “product”*. As an example, the following is a description from this topic:

“I am requesting this loan to use for start up capital for my company. I have 10+ years experience in the transportation industry and am launching a freight brokerage business. I have a niche marketing strategy in place.”

The third largest topic falls far behind the top two topics. It contains 1,836 descriptions, and its most representative words are related to the LendingClub platform (*“club”, “lending”, “existing”, “offered”, “second”, “previous”, “successfully”, “offers”, “clubs”, “offer”*). As an example, the following is a description from this topic:

“I currently have a Lending Club Loan that I used for Debt Consolidation and have a 26% intrest rate on it. I will use this loan to pay that off to get a lower intrest rate and lower monthly payment. I have a very stable job and income for the last 14yrs”

As previously mentioned, BERTopic is configured by default to create a cluster if it contains 10 or more descriptions. In our case, we get 22 clusters of the minimum size. These are very specific clusters. For instance, the most representative words of one of them are: *“partial”, “tlc”, “duplex”, “renovation”, “rentable”, “lakefront”, “consolidationhome”, “improvement”, “inprovement”, “needing”*. It looks like a home improvement-related topic. Let’s look at some descriptions of this topic:

“Purchased a Lakefront home one year ago needing complete renovation. Have consumer debt needing consolidation as a result of these capital improvements.”

“Partial debt consolidation and partial house remodel. House is a duplex that needs some TLC before I can put it on the market.”

In conclusion, BERTopic aims to address the challenge of high dimensionality in terms of instances and variables. However, the number of resulting topics remains excessively large, making them difficult to handle effectively. Nevertheless, in the next section, we will explore the association between the topics and the categories of the categorical variables.

Association between topics and categorical variables

In order to explain the topics generated by BERTopic, we will now calculate the standardized Pearson residuals between the topics and the purposes, so that we can gain information about what the topics talk about.

A standardized Pearson residual is a statistical measure that represents the difference between an observed value and the predicted value of that same observation, divided by the estimated standard deviation of the error term. A high standardized Pearson residual indicates a larger-than-expected deviation from the expected frequency. Residuals exceeding 2.0 indicate a strong relationship, while residuals over 3.0 indicate a very strong relationship.

In our case, we will compute the Pearson residuals to analyze between each categorical variable and the topics extracted from BERTopic. Furthermore, we will only consider positive residuals because they will reveal that the co-occurrence of a given topic and a given category is higher than expected, which means that the topic and the category are related.

Out of all the variables examined, we identified just four significant associations with a topic (Pearson residual > 2). In the following tables, we will delve into a detailed analysis of the most robust associations between variables and topics.

Table 5.5¹¹ shows the three topics with the highest Pearson residual for each category of the *purpose* variable. The highest Pearson residual is found between the “small business” purpose and topic number 1, with a value of 5.39. Such a high result makes sense if we analyze the most representative words of the first topic, which include “*business*”, “*equipment*”, “*inventory*”, “*marketing*” and “*sales*”, among others. There is also a strong association between the *debt_consolidation* purpose and the non-clustered topic (-1). This implies that this association co-occurs more than expected, meaning that these descriptions are more nondescriptive than the average of purposes. We can see in the table that the highest Pearson residual for all the purposes is a topic whose keywords can be easily related to the purpose.

Regarding the rest of the categorical variables, we have seen that they are much less correlated. Therefore, we will just show the top 3 Pearson residuals for each variable, indicating the category they belong to.

With respect to the employment length variable, Table 5.6 shows that its highest Pearson residual is found between the category “10+ years” and the topic 0, which was highly related to the “debt consolidation” purpose in Table 5.5. This indicates that people seeking loans with job stability tend to primarily focus on debt consolidation. The following two highest Pearson residuals are between the non-clustered topic (-1) and low employment lengths.

¹¹Note: Strong Pearson residuals (> 2) are marked with “(*)” in these tables.

Purpose	Pearson res.	Topic	Topic keywords
car	0.7327	24	"vehicle", "miles", "car", "ford", "purchase", "condition", "chevy", "reatta", "buying", "vw"
	0.3832	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"
	0.2823	80	"car", "vehicle", "camaro", "purchase", "mileage", "newer", "auction", "z28", "miles", "buy"
credit_card	0.7368	25	"refinance", "refinancing", "refi", "rate", "lower", "looking", "friendliest", "apr", "percent", "utilization"
	0.6349	6	"high", "rate", "rates", "lower", "balance", "cards", "paying", "minimum", "pay", "loan"
	0.6233	7	"apr", "aprs", "2999", "raised", "lower", "1999", "balance", "2499", "high", "higher"
debt_consolidation	4.1966 (*)	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"
	2.8088 (*)	0	"consolidation", "debt", "consolidationnull", "reorganization", "yobrax", "consolidationring", "consolidationj", "consolidationy", "loadn", "refief"
	0.5783	5	"purpose", "used", "consolidate", "balances", "nbsp", "card", "loan", "manageable", "consolidation", "higher"
educational	0.2292	68	"daughter", "school", "exam", "education", "tuition", "summer", "university", "aid", "semester", "june"
	0.0801	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"
	0.0248	3	"income", "questions", "phone", "insurance", "wife", "utilities", "rent", "mortgage", "balance", "year"
home_improvement	0.8765	27	"pool", "swimming", "inground", "backyard", "installation", "install", "ground", "liveing", "installing", "pools"
	0.8207	29	"pool", "ground", "swimming", "inground", "liner", "backyard", "installing", "yard", "install", "deck"
	0.7069	38	"bathroom", "remodel", "bath", "bathrooms", "kitchen", "master", "tub", "tile", "sink", "remodeling"
house	0.3911	62	"downpayment", "pmi", "renting", "home", "purchase", "house", "property", "buy", "dream", "boys"
	0.1156	85	"land", "property", "estate", "real", "properties", "investment", "buying", "sale", "india", "acres"
	0.0911	303	"buying", "buy", "2nd", "wealthiest", "auntie", "buyernull", "washingtonex", "upgade", "wanting", "florida"
major_purchase	0.3450	1	"business", "equipment", "inventory", "marketing", "expand", "sales", "capital", "products", "website", "product"
	0.2317	125	"boat", "motor", "sailboat", "fishing", "bass", "sailplane", "boats", "sail", "erie", "merlin"
	0.2145	43	"motorcycle", "bike", "harley", "davidson", "riding", "purchase", "bikes", "yamaha", "gas", "miles"
medical	0.4149	42	"surgery", "procedure", "medical", "insurance", "covered", "cover", "cosmetic", "doctors", "surgical", "expenses"
	0.2340	113	"dental", "implants", "root", "teeth", "canal", "canals", "dentist", "bridges", "smile", "extensive"
	0.1724	189	"covered", "insurance", "surgery", "testing", "claim", "medical", "procedure", "er", "cover", "deductible"
moving	0.7875	16	"moving", "relocation", "apartment", "relocating", "deposit", "relocate", "security", "rent", "closer", "expenses"
	0.5009	34	"moving", "deposit", "relocation", "apartment", "security", "apt", "deposits", "costs", "rent", "relocating"
	0.0219	271	"closing", "selling", "sale", "sold", "realtor", "offer", "costs", "sell", "auctioned", "cost"
other	0.3789	1	"business", "equipment", "inventory", "marketing", "expand", "sales", "capital", "products", "website", "product"
	0.2127	95	"transmission", "engine", "repair", "repairs", "sentra", "auto", "vehicle", "car", "fix", "kindly"
	0.2070	16	"moving", "relocation", "apartment", "relocating", "deposit", "relocate", "security", "rent", "closer", "expenses"
renewable_energy	0.0656	272	"solar", "panels", "energy", "electricity", "panel", "power", "pv", "array", "kw", "thermal"
	0.0149	1	"business", "equipment", "inventory", "marketing", "expand", "sales", "capital", "products", "website", "product"
	0.0065	823	"water", "purification", "filtration", "filter", "pump", "healthy", "terminix", "redrilled", "presure", "brittle"
small_business	5.3945 (*)	1	"business", "equipment", "inventory", "marketing", "expand", "sales", "capital", "products", "website", "product"
	0.0490	128	"property", "properties", "homes", "estate", "rental", "units", "foreclosure", "rehab", "real", "market"
	0.0411	549	"personnel", "sweets", "business", "agood", "coin", "busineses", "businesses", "bussines", "relations", "laundry"
vacation	0.3302	79	"vacation", "trip", "visit", "disney", "hawaii", "europe", "family", "anniversary", "cruise", "attend"
	0.2352	107	"vacation", "trip", "visit", "travel", "memories", "airfare", "paris", "seen", "family", "hawaii"
	0.1471	244	"vacation", "valdez", "cod", "bucket", "cape", "china", "summer", "alaska", "fishing", "lifetime"
wedding	0.7686	23	"wedding", "reception", "fiance", "honeymoon", "married", "fiancee", "photographer", "vendors", "venue", "upcoming"
	0.5875	41	"wedding", "married", "honeymoon", "fiance", "getting", "expenses", "dress", "weddings", "fiancee", "families"
	0.1567	81	"ring", "engagement", "girlfriend", "diamond", "propose", "dreams", "purchase", "proposing", "girl", "sungard"

Table 5.5: Top 3 topics' Pearson residuals by *purpose*.

Emp. length	Pearson res.	Topic	Topic keywords
10+ years	1.1829	0	"consolidation", "debt", "consolidationnull", "reorganization", "yobraz", "consolidationring", "consolidationj", "consolidationy", "loadn", "refief"
2 years	0.9714	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"
<1 year	0.8972	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"

Table 5.6: Top 3 topics' Pearson residuals for *emp_length*.

As for the home ownership variable, Table 5.7 shows that its highest Pearson residual is between the "Rent" category and the non-clustered topic (-1), with a large difference over the following Pearson residuals. Having such a high Pearson residual (1.1472) with a topic that is not clustered, means that there is a large number of loans with the "Rent" home ownership for which the fact of having the "Rent" home ownership is not a representative fact. Otherwise, they would be clustered into a "Rent" topic.

Home ownership	Pearson res.	Topic	Topic keywords
Rent	1.1472	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"
Rent	0.3321	3	"income", "questions", "phone", "insurance", "wife", "utilities", "rent", "mortgage", "balance", "year"
Rent	0.2883	16	"moving", "relocation", "apartment", "relocating", "deposit", "relocate", "security", "rent", "closer", "expenses"

Table 5.7: Top 3 topics' Pearson residuals for *home_ownership*.

Pearson residuals for the State variable are shown in Table 5.8. Again, the highest Pearson residual is found between the California State and the non-clustered topic (-1). For the same reason as for the home ownership variable, such a high Pearson residual (3.0199) with a topic that is not clustered indicates that this is not a representative factor. Otherwise, they would be clustered into a "California" topic.

State	Pearson res.	Topic	Topic keywords
CA	3.0199 (*)	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"
CA	1.3580	7	"apr", "aprs", "2999", "raised", "lower", "1999", "balance", "2499", "high", "higher"
WA	0.8794	-1	"free", "years", "payments", "paying", "payment", "help", "thank", "make", "loan", "monthly"

Table 5.8: Top 3 topics' Pearson residuals for *addr_state*.

We can conclude that the analysis of the topic modeling with BERTopic is not very conclusive as the number of topics is very high and the analysis made to describe the topics and relate them with other variables does not offer clear insights.

5.3.2. Relation between the quantitative and categorical variables and the BERT prediction

Our initial hypothesis was that the descriptions would consist primarily of information already captured by categorical variables, mainly by the loan purpose. To investigate this redundancy, we will examine the relationship between the BERT prediction and the categorical variables, but also the quantitative ones.

Quantitative variables

In order to measure the relation between the BERT default prediction and the quantitative variables (*loan_amnt*, *dti_n*, *fico_n* and *revenue*) we will use the Pearson coefficient that accounts for linear relationship and the more general Spearman coefficient that accounts for rank correlation.

By looking at Table 5.9, we can see that there is a weak but significant correlation between the default predictions of BERT and the quantitative variables (the *p-values* close to 0 mean that there is strong evidence that the observed correlation is statistically significant and not due to random chance, even if it is very weak). The highest correlation is found with the FICO variable, exhibiting a negative relationship. This finding suggests that as the FICO value increases, the default probability estimated by BERT decreases. That is, the higher the FICO score, the lower the BERT score, which means that reliable texts according to BERT are weakly related to reliable FICO values. Similarly, the revenues are also negatively correlated to the BERT scores. This means that reliable texts according to BERT are weakly related to higher revenues.

Variable	Pearson		Spearman	
	Correlation	<i>p-value</i>	Correlation	<i>p-value</i>
<i>loan_amnt</i>	-0.0119	4.28×10^{-5}	-0.0125	1.54×10^{-5}
<i>dti_n</i>	0.0560	2.18×10^{-83}	0.0506	2.49×10^{-68}
<i>fico_n</i>	-0.1182	0.0	-0.1152	0.0
<i>revenue</i>	-0.0775	5.89×10^{-158}	-0.1027	1.25×10^{-276}

Table 5.9: Correlation coefficients of quantitative variables with *BERT_pred*.

Categorical variables

To measure the level of redundancy between the BERT scores and the categorical variables, we will use the ANOVA test.

The Analysis of Variance (ANOVA) test is a statistical technique used to compare the means of three or more groups to determine if there are significant differences between them. It is commonly employed to assess the relationship or correlation between a categorical variable (with three or more levels) and a quantitative variable.

If the null hypothesis is rejected (*p-value* < 0.05), it implies that there is evidence of a significant correlation between the categorical variable and the quantitative variable.

Table 5.10 shows the ANOVA test results for the categorical variables. The most important observation is that the *p-value* of all the variables is very close to 0 and less than 0.05. This implies that the differences between the groups are unlikely to occur by chance, meaning that there is a statistically significant relationship between *BERT_pred* and all the categorical variables.

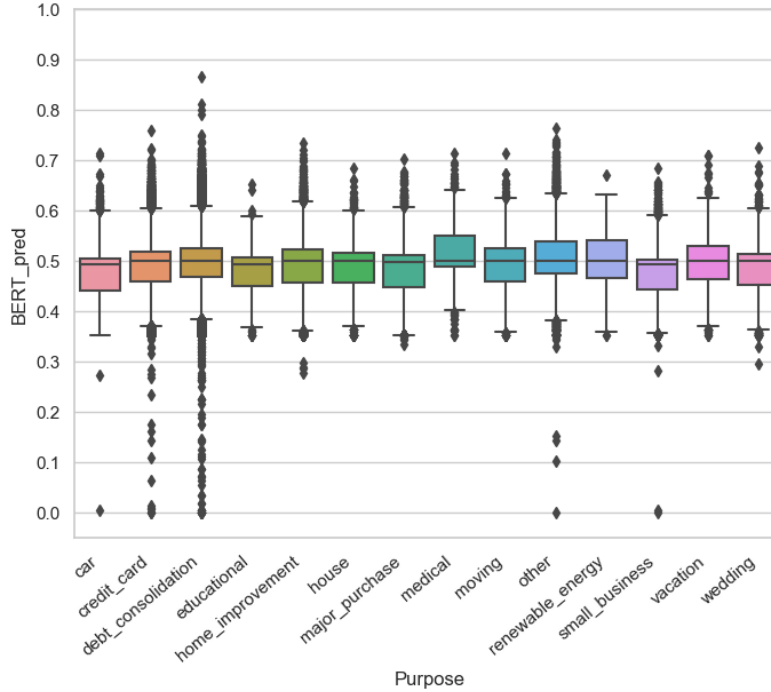
In particular, it can be said that the means of the BERT scores among the categories of each variable are significantly different. Thus, it means that there is some kind of relation between each categorical variable (or some of its categories) and the BERT score.

Variable	ANOVA test with <i>BERT_pred</i>	
	F-statistic	<i>p-value</i>
<i>emp_length</i>	112.00	4.534×10^{-256}
<i>purpose</i>	101.18	9.097×10^{-272}
<i>home_ownership</i>	64.71	9.247×10^{-55}
<i>addr_state</i>	6.32	1.845×10^{-39}

Table 5.10: ANOVA test of categorical variables with *BERT_pred*.

Since the description variable is typically used to describe the purpose of the loan, we would expect a relationship with the *purpose* variable. By looking at the *p-values* of Table 5.10, we can see that this hypothesis is true, as it has the largest significance. Due to this, we will now look at the boxplot representing the *BERT_pred* distribution by purpose (Figure 5.5).

In general, the default probabilities are maintained in the central part (between 40% and 60%). It is interesting to see that there are some purposes with a wider probability range, suggesting that BERT finds further evidence of default or non-default. For instance, “debt consolidation” is the only purpose that takes probabilities over 80%, and the “other” purpose has a high density on higher default probabilities (around 70%). Moreover, we can see that the *BERT_pred* variable is confident when predicting non-default for purposes such as “car”, “credit card”, “debt consolidation”, “other” and “small business”, since its lowest values are very close to 0% probability of default.

Figure 5.5: *BERT_pred* boxplot by purpose.

Chapter 6

Analysis of the results of the boosting algorithms with the BERT scores

In this chapter, we will build upon our previous exploration of textual information processing with BERT. We have observed how BERT effectively converts textual data into a probability of belonging to the default class, representing the reliability of the description. Now, our focus shifts to incorporating these percentages as input variables for the boosting algorithms that were discussed in Chapter 4. The objective is to analyze the extent to which these percentages can be leveraged to achieve improved results.

6.1. Quantitative, categorical and BERT variables

To validate the hypothesis discussed earlier, we will integrate the *BERT_pred* score alongside quantitative and categorical variables into the boosting algorithms. We will then analyze and compare the outcomes obtained from this integration with the results presented in Chapter 4. By doing so, we aim to demonstrate the potential impact and efficacy of combining these variables in enhancing the overall performance of the algorithms.

To achieve this, we will run the same pipeline as in Chapter 4, with the only difference that we will now also have the *BERT_pred* variable. The process of hyperparameter tuning using genetic algorithms and the subsequent analysis of results will remain unchanged. As a result, the outcomes obtained in both Chapter 4 and this chapter are directly comparable.

6.1.1. XGBoost

The fitness evolution over generations has a similar shape to previous experiments. Already from the first generation, we can see a reasonably good suboptimal individual. However, its fitness is surpassed in subsequent generations.

The addition of *BERT_pred* does not imply any change in the exploration pattern either. Similarly to previous executions, the full range of all the hyperparam-

ters is tested, while significant density peaks are observed in the optimal values due to the elitist factor of the tournament selection applied in the genetic algorithm.

It is curious to see that some hyperparameters in the best combinations have converged to values different from those found in section 4.2.1. For instance, the learning rate has been reduced from the 0.25 range to approximately 0.1, while the $n_estimators$ has been increased from around 150 to roughly 450. However, these adjustments are interconnected, as a lower learning rate may require a greater number of iterations to reach optimal performance.

With respect to performance, Table 6.1 shows the scoring metrics of the best XGBoost classifier found with quantitative, categorical and $BERT_pred$ variables. We can see that the use of $BERT_pred$ leads to optimal performance in all scoring metrics except for accuracy, which implies that in this model the non-default class is recognized slightly worse than in the other two models. However, this discrepancy is not a problem since accuracy is considered a less reliable metric due to its failure to account for data imbalance. In fact, a lower accuracy score may indicate superior performance, as demonstrated by the naïve classifier in subsection Naïve classifier.

Metric	Quant.	Quant. + Categ.	Quant. + Categ. + $BERT_pred$
BAC	0.6062	0.6154	0.6181
AUC	0.6457	0.6575	0.6641
F1	0.3192	0.3266	0.3291
Precision	0.2138	0.2168	0.2192
Recall	0.6300	0.6614	0.6683
Accuracy	0.5896	0.5835	0.5832

Table 6.1: XGBoost metrics for the different variable sets.

In terms of feature importance (Figure 6.1), $BERT_pred$ emerges as the fourth most significant variable in predicting loan default, with an approximate importance of 6%. This implies that $BERT_pred$ holds a considerable level of importance, comparable to other influential variables such as the “credit card” purpose and revenue variables, which were considered significant factors in section 4.2.1 (Figure 4.5).

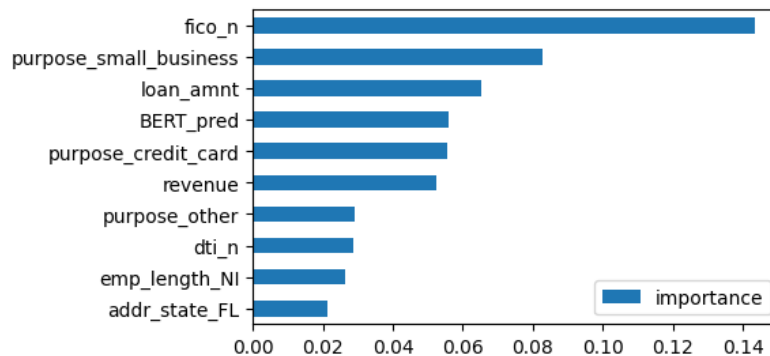


Figure 6.1: XGBoost feature importances (quantitative, categorical and $BERT_pred$ variables).

The SHAP values (Figure 6.2) reveal a similar utilization of variables compared to previous executions. Regarding the *BERT_pred* variable, its distribution aligns with expectations, displaying a dense concentration around the 0.5 value range. These values have a neutral impact of 0 on the model's output due to their inability to provide discriminatory information. However, higher values of *BERT_pred* contribute to predicting default, while lower values help in predicting non-default, both with a proportional impact on the model's output.

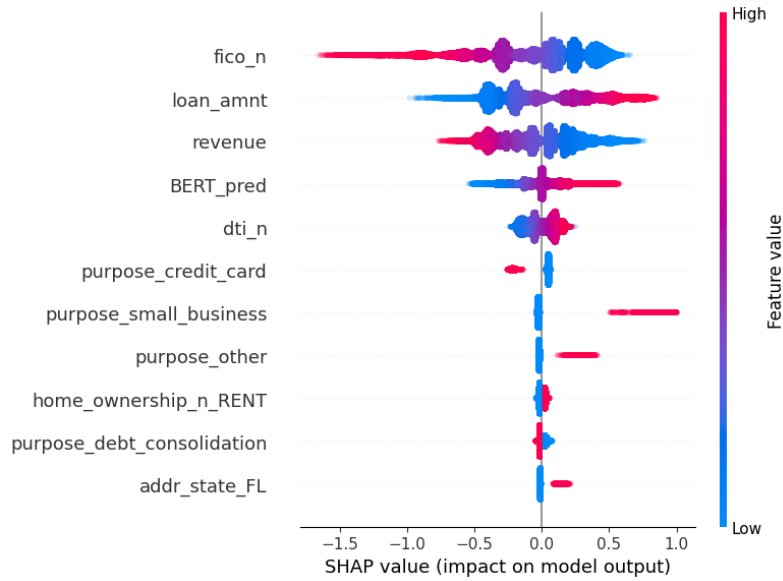


Figure 6.2: XGBoost SHAP values (quantitative, categorical and *BERT_pred* variables).

6.1.2. CatBoost

The evolution of fitness across generations and the exploration of hyperparameters in this execution follow a similar pattern to the previous XGBoost run. In both cases, we appreciate the emergence of a promising individual right from the initial generation, and the exploration of hyperparameters contains peaks at optimal values, while also performing a full range exploration.

Just as happened with the previous XGBoost experiment, the hyperparameter that exhibits a larger modification in the top 50 individuals is the learning rate and the iterations. The learning rate is decreased with convergence in the 0.7 range, while the number of iterations is increased around the 320 value. Again, these changes are correlated for the same reason as before.

Table 6.2 shows the scoring metrics of CatBoost with the different sets of variables. Again, the improvement comes from enhancing the classification of the default class, while losing some accuracy and precision due to slightly worse results in the non-default class.

Metric	Quant.	Quant. + Categ.	Quant. + Categ. + <i>BERT_pred</i>
BAC	0.6063	0.6153	0.6180
AUC	0.6455	0.6574	0.6641
F1	0.3198	0.3280	0.3297
Precision	0.2162	0.2229	0.2220
Recall	0.6136	0.6208	0.6483
Accuracy	0.6013	0.6115	0.5970

Table 6.2: CatBoost metrics for the different variable sets.

Regarding feature importance (Figure 6.3), CatBoost gives more importance to the *BERT_pred* variable (9%) than XGBoost (6%, Figure 6.1). All importances are reduced proportionally from those given by CatBoost without *BERT_pred* (Figure 4.7), and the importance attributed to *BERT_pred* is derived from this proportionally distributed reduction across other features.

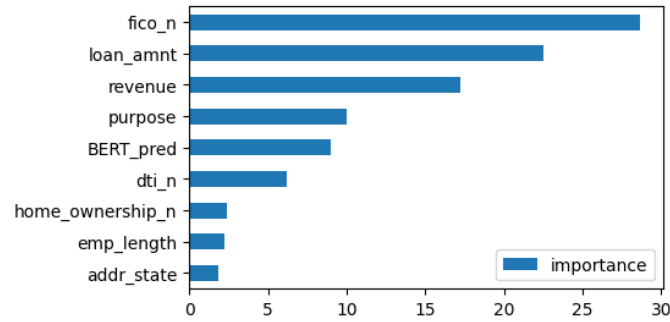


Figure 6.3: CatBoost feature importances (quantitative, categorical and *BERT_pred* variables).

Figure 6.4 represents the SHAP values, and reveals a very similar feature impact as that of XGBoost (Figure 6.2). Both models demonstrate comparable ranges of influence on the model's output and exhibit similar patterns in variable usage.

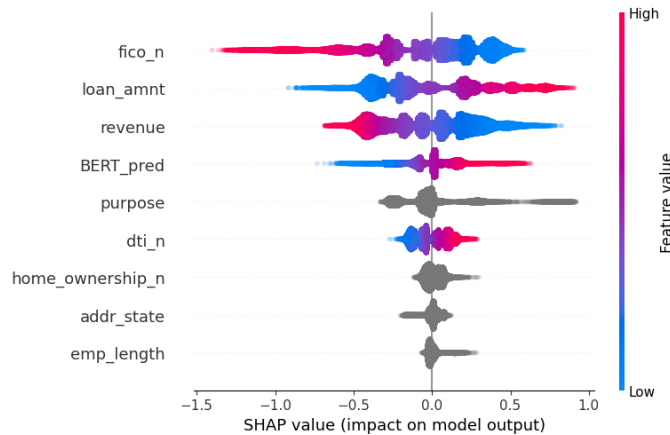


Figure 6.4: CatBoost SHAP values (quantitative, categorical and *BERT_pred* variables).

6.2. Analysis of the impact of textual variables

After a thorough analysis, we have found that the improvements resulting from the inclusion of textual variables are not as substantial as we had anticipated in predicting loan default. In this section, we focus on analyzing the classifications of the top-performing XGBoost model before and after the inclusion of textual variables.

One of the main hypotheses of this chapter was that employing NLP techniques on textual variables would mainly improve the prediction for the “other” category purpose variable, since we saw in section 5.3.1 that the borrower’s description is closely related to the purpose variables. The “other” category is inherently ambiguous, and we believed that further processing of its description would lead to a better prediction outcome.

To this end, we will conduct an analysis of the predictive performance for each loan purpose when using BERT’s predictions versus not using them.

Figure 6.5 depicts the BAC improvement for each loan purpose, with the darker color representing BAC without BERT and the lighter color representing BAC with BERT. The BAC improvement percentage for each loan purpose is also displayed above the bars.

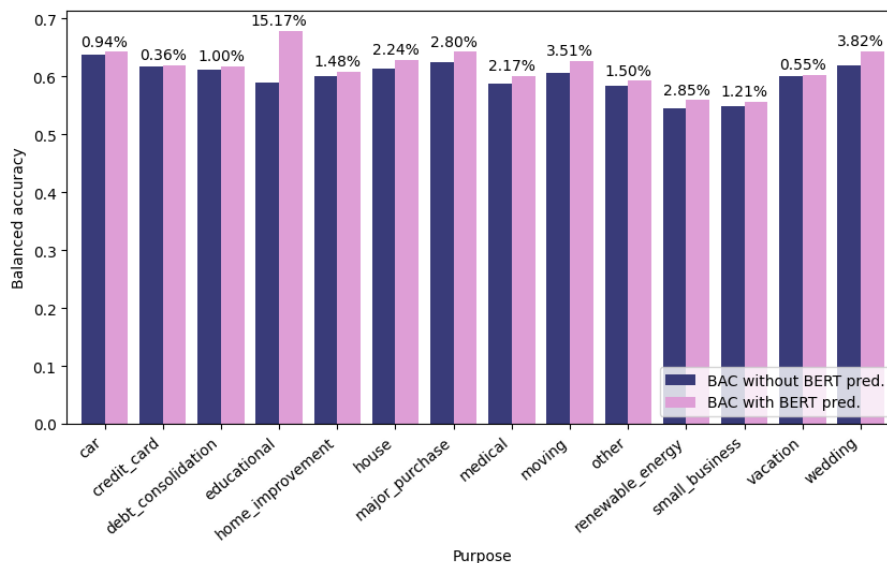


Figure 6.5: BAC improvements by *purpose* after adding *BERT_pred*.

Contrary to our hypothesis, the loan purpose that exhibits the most significant improvement with the inclusion of textual variables is by far the “educational” purpose, with a substantial BAC increase of over 15%. Our initial expectation was that the “other” purpose would be the most improved one, but this is not reflected in the analysis results. However, this category ranks eighth out of the 14 purposes in terms of improvement, with an increase of 1.5%.

This slight improvement is coherent with the information analyzed in Table 5.5, since the “other” purpose has very low standardized residuals with the topics. More-

over, the keywords of these topics are very diverse, dealing with topics such as business, vehicles or home relocation.

Nevertheless, it is not surprising that “educational” is the purpose that exhibits the most significant improvement after the inclusion of textual variables. This purpose typically entails a considerable amount of details that must be accurately assessed for a thorough understanding of the loan application.

For instance, the “educational” purpose might include details about the field of studies, educational institution, total cost breakdown, academic record, and future career plans, among others. These details require a precise understanding for accurate risk assessment, and the inclusion of textual variables through LLM processing techniques can aid in achieving this.

To illustrate these improvements, we will now see some examples of descriptions. First of all, the following is an example of a non-default loan request with “educational” purpose whose prediction was missed without BERT (predicted default) but the addition of BERT makes it hit, correctly predicting non-default:

“I’m currently pursuing a MBA in finance. [...] I don’t currently qualify for financial aid due to my GPA which dropped below a 3.0. [...] The reason for the drop in my GPA was primarily health related. I have Multiple Sclerosis; [...] Since then, I’ve started on a new medication (Tysabri) and I feel better than I’ve felt in a long time. [...] Overall, I’m in a great place right now to move forward toward the completion of my degree, I just need a little help getting there.”

We can see that this description contains details about the academic and personal background of the borrower. Although it was initially predicted as default, after the *BERT_pred* variable inclusion it was correctly predicted as non-default.

By examining descriptions with the “educational” purpose to make this analysis, we noticed that these descriptions are longer in general than the average descriptions seen in Chapter 3. To examine this, Figure 6.6 shows the distribution of the average description length grouped by purpose.

The length of descriptions for “educational” and “small business” purposes is significantly longer compared to the average. This disparity results in a substantial improvement in BAC for the “educational” purpose (+15.17%, Figure 6.5), whereas the improvement for the “small business” purpose is relatively minor (+1.21%, Figure 6.5). This discrepancy can be attributed to the fact that the “small business” purpose has a significantly higher default rate compared to the average (Figure 3.11). Consequently, the impact of processing these descriptions is relatively limited, leading to a smaller improvement.

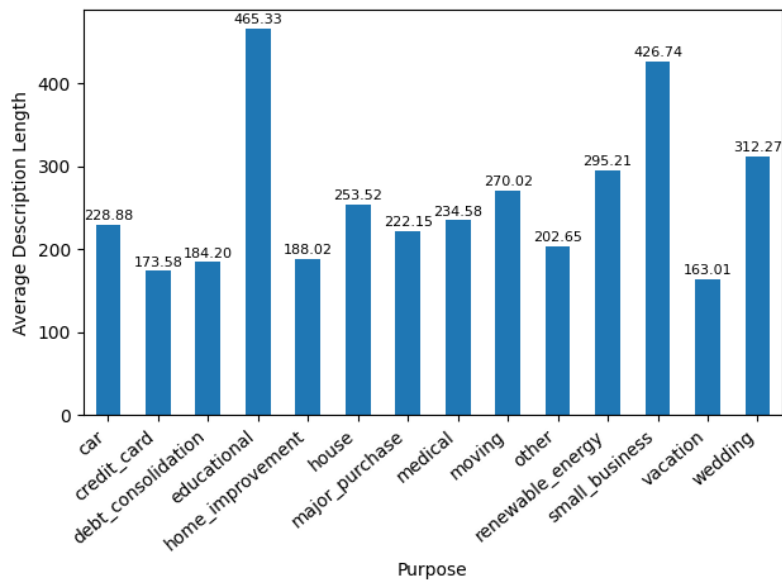


Figure 6.6: Average description length by purpose.

6.3. Conclusions

As the best-performing classifiers for XGBoost and CatBoost have been achieved with quantitative, categorical and *BERT_pred* variables, we will compare their performance with this set of variables. Table 6.3 shows this performance comparison.

Metric	Best XGBoost	Best CatBoost
BAC	0.6181	0.6180
AUC	0.6641	0.6641
F1	0.3291	0.3297
Precision	0.2192	0.2220
Recall	0.6683	0.6483
Accuracy	0.5832	0.5970

Table 6.3: Scoring metrics with quantitative, categorical and *BERT_pred* variables.

The balanced accuracy, which is the optimized metric, has undergone an improvement of 0.3%. It might look like a small improvement, but taking into account the complexity of the classification with BERT and the redundancy with other variables that the textual information has, it is a significant improvement that may lead to hundreds of new correct classifications.

As it occurred in the conclusions of Chapter 4, both XGBoost and CatBoost are achieving very similar performance. The main difference among them is that CatBoost has slightly better precision, while XGBoost has a slightly better recall. Again, the choice on which model is better depends on the requirements of the specific task.

Conclusions and Future Work

The purpose of this chapter is to present and compare the results obtained from the most significant studies discussed in Chapter 1 (Introduction) with our own findings. Moreover, we will conduct a detailed analysis of the outcomes derived from our experiments, drawing meaningful conclusions for this Bachelor’s thesis. Lastly, we will explore potential avenues for future research to further advance our study.

7.1. Discussion of results

To facilitate a meaningful comparison between previous studies and our own results, we provide a comprehensive summary in Table 7.1 of the outcomes obtained through our experiments. With this information at hand, we can now delve into the results of the relevant studies conducted in the field.

Metric	Quantitative		Quantitative + categ.		Quantitative + categ. + <i>BERT_pred</i>	
	Best XGBoost	Best CatBoost	Best XGBoost	Best CatBoost	Best XGBoost	Best CatBoost
BAC	0.6062	0.6063	0.6154	0.6153	0.6181	0.6180
AUC	0.6457	0.6455	0.6575	0.6574	0.6641	0.6641
F1	0.3192	0.3198	0.3266	0.3280	0.3291	0.3297
Precision	0.2138	0.2162	0.2168	0.2229	0.2192	0.2220
Recall	0.6300	0.6136	0.6614	0.6208	0.6683	0.6483
Accuracy	0.5896	0.6013	0.5835	0.6115	0.5832	0.5970

Table 7.1: Experiments’ results summary.

Regarding the boosting algorithms, our work builds upon the foundation laid by Ariza-Garzón et al. (2020), which served as a baseline for our research. In their study, the authors achieved a balanced accuracy (BAC) of 0.624 and an area under the receiver operating characteristic curve (AUC) of 0.674 using XGBoost. This represents a modest improvement of 0.6% in BAC and 1% in AUC compared to our own results. It is important to note, however, that direct comparisons between our findings and theirs are not feasible due to differences in dataset processing.

In our study, we utilized the same dataset as Ariza-Garzón et al. (2020) for consistency, but we implemented specific criteria during data selection. As explained in the *Dataset Description* section, we chose to focus solely on loans that had associated descriptions. Consequently, our final dataset consisted of 119,004 entries out of the original 1,347,681, amounting to only 8.83% of the data utilized by the authors. Furthermore, it is worth noting that loans with available descriptions were only accessible until March 2014, while in Ariza-Garzón et al. (2020) loans until 2018 are considered. Therefore, there is also a temporal differential factor.

By incorporating textual description processing using the BERT model, we were able to achieve results that were remarkably similar to those reported by Ariza-Garzón et al. (2020). Despite considering only a small fraction of the data used by the authors, our findings demonstrate the effectiveness of integrating advanced techniques for text analysis.

When it comes to employing Large Language Models for description processing, the most relevant papers are that of Kriebel and Stitz (2022) and Stevenson et al. (2021).

In Kriebel and Stitz (2022), the authors achieved an AUC of 0.712 by incorporating BERT-based processing of descriptions alongside structured data. This corresponds to a 4.8% improvement in AUC compared to our own results. Similarly, Stevenson et al. (2021) accomplished an AUC of 0.72 by combining structured data with BERT processing. This represents a 5.6% improvement in AUC relative to our findings.

However, these results are not directly comparable to ours either, as they use variables that are not available at the granting time (e.g., interest rate on the loan), enhancing their predictions. Moreover, both studies utilize a neural network as a single model, incorporating all variables together. By employing a neural network, these studies are able to leverage the textual representation provided by the BERT encoder, which offers more information than our default percentage (*BERT_pred* variable). While the neural network can exploit this additional information, our boosting algorithms are not designed to do so. In addition, both studies prioritize AUC as the scoring metric. In our case, our genetic algorithm was optimized for BAC, making our AUC potentially suboptimal and, therefore, not directly comparable to their optimized AUC values.

7.2. Conclusions

In this Bachelor’s thesis, we have had the opportunity to explore a wide range of experiments, both those initially planned and those that arose along the course of our research. Our focus on thoroughness and attention to detail guided us throughout this journey, leading us to the insightful conclusions presented in this section.

In our study, we initially demonstrated the effectiveness of hyperparameter optimization. To achieve this, we employed a genetic algorithm, which produced significantly improved results compared to the default configuration of the boosting algorithms. We observed remarkable enhancements, with the optimized metric, BAC, showing a notable increase of over 11%. This large improvement may be

because parameters such as *scale_pos_weight* were optimized, which control the balance of positive and negative weights. Therefore, the dataset imbalance is taken into account.

Subsequently, our investigation focused on evaluating the performance of the CatBoost model when handling categorical variables. Surprisingly, we discovered that the native treatment of categorical variables by CatBoost did not involve differential results. Its performance was very similar to that of XGBoost, which requires one-hot encoding to handle categorical variables. In fact, the use of one-hot encoding in XGBoost allowed us for a fine-grained analysis of these variables, enabling the study of the effect of each category.

Related to this topic, we also aimed to evaluate the incorporation of categorical variables. When we included these variables, we observed a 1% improvement in the BAC compared to considering only quantitative variables. While this improvement may appear small at first glance, it is important to bear in mind that the size of the dataset is 119k loans, so the improvement implies the correct classification of several hundreds of loans. Furthermore, upon examining feature importance and SHAP values, it becomes evident that the top variables contributing to the model’s classification are mainly quantitative. This observation helps to explain the relatively modest improvement attained through the inclusion of categorical variables.

However, note that the consideration of categorical variables allows for a more nuanced understanding of the data. It enables us to capture the unique characteristics and relationships within different categories, contributing to a more fine-grained analysis.

In relation to the utilization of Large Language Models for loan description processing, our findings indicate that the BERT model captures certain knowledge about the loans that is not encompassed by classical variables. As a result, the performance of classifiers is enhanced, with an improvement of 0.27% in BAC and 0.67% in AUC for the boosting classifiers. When the performance of these BERT predictions was evaluated independently with a binarization threshold, they achieved a BAC of 54.11% and an AUC of 56.12%. These results indicate that BERT is capable of discerning among the different classes to some extent. The relatively modest improvement observed in the classifiers can be attributed to the proven relation between the BERT score and other variables, particularly the loan purpose. It may also happen that the relationship between both aspects does not imply causality in either direction. However, both the text descriptions and the rest of the variables inform of the loan risk in a similar way.

Regarding the utilization of BERT, we explored various configurations of the neural network beyond the LLM, specifically focusing on the layers where fine-tuning of the pre-trained model takes place. This approach is novel within the context of BERT-based classification, as most existing articles merely append a final dense layer. Our findings revealed that an optimal network architecture can contribute to improving the default probabilities. However, further experimentation is needed to see whether the results can be improved, for example, by considering other architectures.

While deep learning proves to be effective in processing descriptions, one of its

main drawbacks lies in the lack of explainability. To address this limitation, we conducted an extensive analysis to see its relationship with other variables and found that it was significantly related to all of them. Furthermore, we analyzed the improvement in each loan purpose and discovered that the most significant improvement was observed in loans intended for educational purposes, which have lengthier and more detailed descriptions, allowing BERT to extract more information and perform better. It is worth noting that improvements in BAC were observed across all loan purposes, indicating the overall efficacy of incorporating text processing with BERT.

Hence, our study makes several distinct contributions to the current literature on machine learning-based P2P lending models. These contributions include:

- **Integration of BERT Default Probability:** To enhance explainability, we introduce the BERT default probability as a quantitative variable. This inclusion allows for a more transparent understanding of how BERT contributes to the prediction process, providing valuable insights into the decision-making mechanism.
- **Exhaustive BERT Fine-Tuning Configuration:** We extensively explore and fine-tune the architecture of BERT to optimize its performance in the context of P2P lending models by considering various configuration options.
- **Thorough Analysis for BERT Explainability:** In order to enhance explainability, we conduct a comprehensive analysis of the BERT prediction process, aiming to elucidate the factors and patterns that contribute to the BERT's predictions.

By introducing these contributions, our study makes significant advances in the understanding and application of machine learning-based P2P lending models with textual data, laying the foundation for further research and development in this field.

7.3. Future work

As for future work, we identify two main ways of approaching it.

On the one hand, from the scientific and research point of view, we can continue with the same research venue. Here are the main ideas:

- **Delve deeper into what information models the BERT default prediction.** This could be done by using NLP methods and statistical analysis.
- **Study whether the final prediction can be improved by training BERT and the boosting algorithms at the same time, instead of in parallel.** Another option would be to incorporate the structured variables in a post-BERT layer in the neural network.

- Throughout our analysis, we have seen that achieving better results using the available variables at the time of granting loans poses a challenge. However, when comparing the performance of XGBoost and CatBoost, both models achieved similar F1 scores, but their precision and recall values differed. Therefore, a combination of both might improve the results by leveraging their strengths.

On the other hand, we identify a more applied line of work, in which the knowledge obtained in this Bachelor's thesis could be applied to create a company dedicated to offering prediction, classification or generation services based on Large Language Models.

One potential service that the company could provide is the integration of numerical, categorical, and textual variables to offer personalized product recommendations to customers of a store, with the textual variables being derived from customer reviews. By analyzing the aspects that users value in a product, the company can offer products that stand out in those areas.

Furthermore, thanks to the semantic and contextual analysis capabilities of LLMs, the company can extend its services to other customer-centric domains. For instance, it can automate the analysis of hotel and restaurant reviews, enhance customer service experiences, or, of course, predict credit default.

Bibliography

- ARIZA-GARZÓN, M. J., ARROYO, J., CAPARRINI, A. and SEGOVIA-VARGAS, M.-J. Explainability of a Machine Learning Granting Scoring Model in Peer-to-Peer Lending. *IEEE Access*, Vol. 8, 64873–64890, 2020.
- BOIKO FERREIRA, L. E., BARDDAL, J. P., GOMES, H. M. and ENEMBRECK, F. Improving Credit Risk Prediction in Online Peer-to-Peer (P2P) Lending Using Imbalanced Learning Techniques. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 175–181. 2017.
- BREIMAN, L. Bagging predictors. *Machine Learning*, Vol. 24(2), 123–140, 1996.
- BREIMAN, L. Random Forests. *Machine Learning*, Vol. 45, 5–32, 2001.
- CHEN, T. and GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. *CoRR*, Vol. abs/1603.02754, 2016.
- DEVLIN, J., CHANG, M.-W., LEE, K. and TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019.
- DORFLEITNER, G., PRIBERNY, C., SCHUSTER, S., STOIBER, J., WEBER, M., DE CASTRO, I. and KAMMLER, J. Description-text related soft information in peer-to-peer lending – Evidence from two leading European platforms. *Journal of Banking & Finance*, Vol. 64, 169–187, 2016. ISSN 0378-4266.
- DUAN, J. Financial system modeling using deep neural networks (DNNs) for effective risk assessment and prediction. *Journal of the Franklin Institute*, Vol. 356(8), 4716–4731, 2019.
- FREUND, Y. and SCHAPIRE, R. E. A Short Introduction to Boosting. *Japanese Society for Artificial Intelligence*, Vol. 14(5), 771–780, 1999.
- FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, Vol. 29(5), 1189 – 1232, 2001.
- GONG, J. and KIM, H. RHSBoost: Improving classification performance in imbalance data. *Computational Statistics & Data Analysis*, Vol. 111, 1–13, 2017.

- GROOTENDORST, M. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. Tech. rep., 2022.
- GUO, Y., ZHOU, W., LUO, C., LIU, C. and XIONG, H. Instance-based credit risk assessment for investment decisions in P2P lending. *European Journal of Operational Research*, Vol. 249(2), 417–426, 2016. ISSN 0377-2217.
- HOCHREITER, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 06(02), 107–116, 1998.
- HOCHREITER, S. and SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, Vol. 9(8), 1735–1780, 1997.
- JIANG, C., WANG, Z., WANG, R. and DING, Y. Loan default prediction by combining soft information extracted from descriptive text in online peer-to-peer lending. *Annals of Operations Research*, Vol. 266(1-2), 511–529, 2017.
- KIM, J.-Y. and CHO, S.-B. Deep Dense Convolutional Networks for Repayment Prediction in Peer-to-Peer Lending. In *Advances in Intelligent Systems and Computing*, 134–144. Springer International Publishing, 2018.
- KRAUS, M., FEUERRIEGEL, S. and OZTEKIN, A. Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research*, Vol. 281(3), 628–641, 2020.
- KRIEBEL, J. and STITZ, L. Credit default prediction from user-generated text in peer-to-peer lending using deep learning. *European Journal of Operational Research*, Vol. 302(1), 309–323, 2022.
- LI, W., DING, S., CHEN, Y. and YANG, S. Heterogeneous Ensemble for Default Prediction of Peer-to-Peer Lending in China. *IEEE Access*, Vol. 6, 54396–54406, 2018.
- LIN, M., PRABHALA, N. R. and VISWANATHAN, S. Judging Borrowers by the Company They Keep: Friendship Networks and Information Asymmetry in Online Peer-to-Peer Lending. *Management Science*, Vol. 59(1), 17–35, 2013.
- LIN, X., LI, X. and ZHENG, Z. Evaluating borrower’s default risk in peer-to-peer lending: evidence from a lending platform in China. *Applied Economics*, Vol. 49(35), 3538–3545, 2016.
- LUNDBERG, S. M. and LEE, S. A unified approach to interpreting model predictions. *CoRR*, Vol. abs/1705.07874, 2017.
- MALEKIPIRBAZARI, M. and AKSAKALLI, V. Risk assessment in social lending via random forests. *Expert Systems with Applications*, Vol. 42(10), 4621–4631, 2015.
- MICHEL, J. Do Unverifiable Disclosures Matter? Evidence from Peer-to-Peer Lending. *The Accounting Review*, Vol. 87(4), 1385–1413, 2012.

- NAMVAR, A., SIAMI, M., RABHI, F. A. and NADERPOUR, M. Credit risk prediction in an imbalanced social lending environment. *CoRR*, Vol. abs/1805.00801, 2018.
- PROKHORENKOVA, L., GUSEV, G., VOROBEOV, A., DOROGUSH, A. V. and GULIN, A. CatBoost: Unbiased boosting with categorical features. 2017.
- QUINLAN, J. R. Induction of Decision Trees. *Machine Learning*, Vol. 1(1), 81–106, 1986.
- STEVENSON, M., MUES, C. and BRAVO, C. The value of text for small business default prediction: A Deep Learning approach. *European Journal of Operational Research*, Vol. 295(2), 758–771, 2021.
- VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L. and POLOSUKHIN, I. Attention Is All You Need. 2017.
- WANG, C., ZHANG, W., ZHAO, X. and WANG, J. Soft information in online peer-to-peer lending: Evidence from a leading platform in China. *Electronic Commerce Research and Applications*, Vol. 36, 100873, 2019.
- WANG, S., QI, Y., FU, B. and LIU, H. Credit Risk Evaluation Based on Text Analysis. *International Journal of Cognitive Informatics and Natural Intelligence*, Vol. 10(1), 1–11, 2016.
- XIA, Y., LIU, C. and LIU, N. Cost-sensitive boosted tree for loan evaluation in peer-to-peer lending. *Electronic Commerce Research and Applications*, Vol. 24, 30–49, 2017.
- YANG, X., FAN, W., WANG, L., YANG, S. and WANG, W. Risk Control of Online P2P Lending in China Based on Health Investment. *Ekoloji*, Vol. 49(35), 2013–2022, 2019.
- ZHANG, W., WANG, C., ZHANG, Y. and WANG, J. Credit risk evaluation model with textual features from loan descriptions for P2P lending. *Electronic Commerce Research and Applications*, Vol. 42, 100989, 2020.

Genetic exploration analysis

This appendix provides a detailed step-by-step analysis of the experiments conducted in this project. The analysis encompasses genetic exploration and evolution, and scoring metrics of the best individuals. The analysis is supported by a set of four plots for each execution, including:

- Fitness evolution over generations: Evolution of the maximum, minimum and average fitness of all the 150 individuals for each of the 20 generations.
- Hyperparameter exploration: Two-by-two hyperparameter scatter plots showing the hyperparameter values explored by the genetic algorithm.
- Hyperparameters of the top-50 individuals: For each hyperparameter, a graph is plotted with its value for the top-50 best individuals, from best individual (top-1) to worst (top-50).
- Scores of the top-50 individuals: For each score metric, a graph is plotted with its value for the top-50 best individuals, from best individual (top-1) to worst (top-50).

Due to repetitive results among the experiments, these plots have been omitted in Chapters 4 and 6. Instead, we have included the most significant findings from the analysis in those chapters. To provide an example of a complete analysis, we will now examine the results obtained from the execution of XGBoost with quantitative, categorical, and *BERT_pred* variables, which turned out to be the one with highest performance.

Fitness evolution over generations

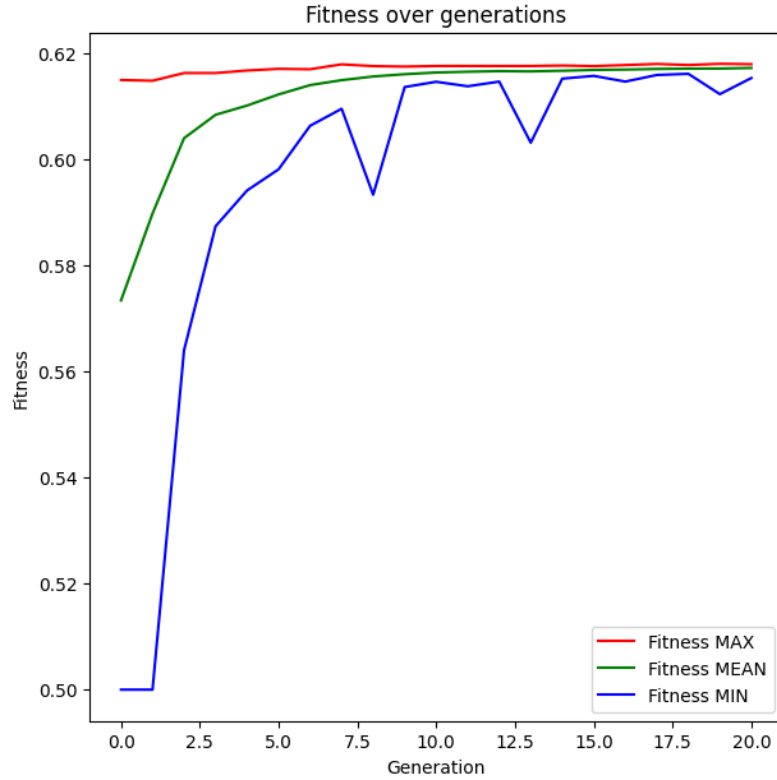


Figure A.1: Fitness evolution.

It can be seen in Figure A.1 that already from the first generation there is at least one reasonably good individual (maximum, in red). In addition, a steady improvement in the average fitness of the generations can be seen (green line) as a consequence of the elitism found in the tournament selection, although this improvement is much reduced from the sixth generation onwards.

This consideration leads me to believe that if there are limitations on time and computational resources and it is still desirable to maintain a satisfactory classifier, the utilization of the genetic algorithm may not be necessary. In this scenario, it would suffice to generate an initial population of 150 individuals, as is performed in the first generation of the genetic algorithm, and retain the best one, without much need for further iterations.

Hyperparameter exploration

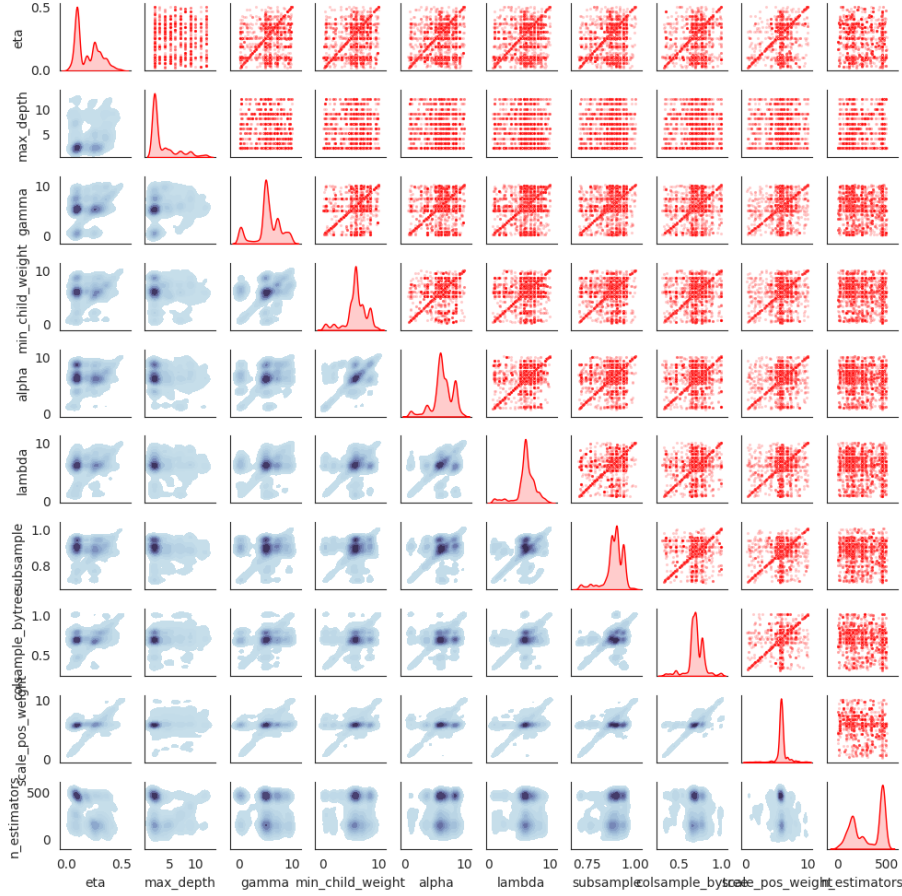


Figure A.2: Hyperparameter exploration.

The search spaces for the hyperparameters are biased due to the elitism factor. If this factor was removed, each hyperparameter would be explored with a more homogeneous distribution throughout its range. However, by incorporating elitism, we are able to more thoroughly explore each hyperparameter in regions that have previously produced favorable results.

Having elitism through a tournament selection with tournament size of 2 individuals out of 150 in each generation represents an optimal trade-off between the breadth and depth of exploration. This can be observed in the density of exploration along the main diagonal of Figure A.2. While all parameters are explored across their full range, there are also pronounced peaks, indicating that certain values have produced superior results and therefore justifies deeper exploration.

Top-50 individuals hyperparameters

Figure A.3 represents the hyperparameters of the 50 individuals with a higher BAC. It contains repeated values for some hyperparameters, again due to the elitist

factor of the tournament selection, which will select better individuals.

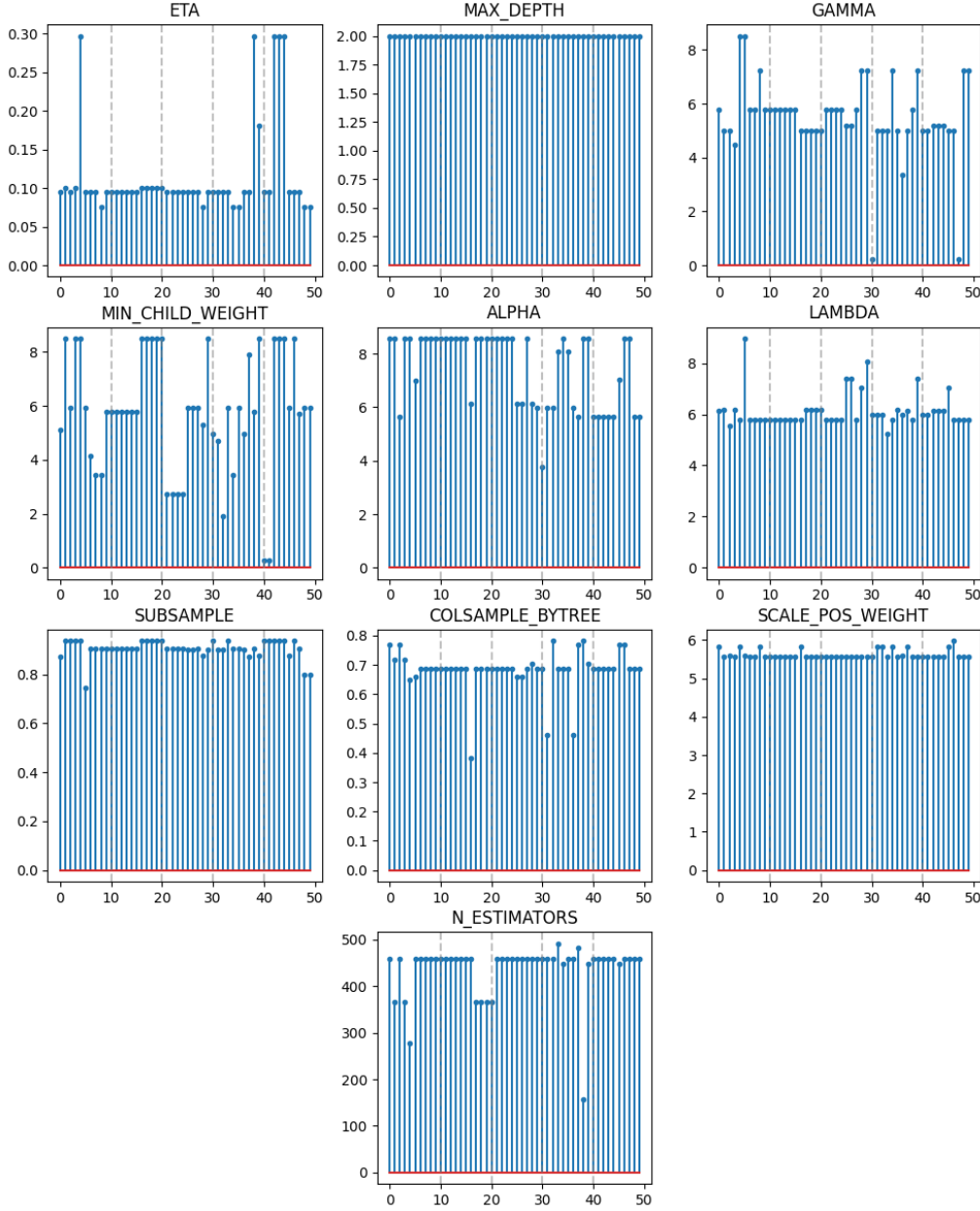


Figure A.3: Top-50 individuals hyperparameters.

However, there are also hyperparameters whose value varies widely in the top-50 of individuals, such as *gamma* or *min_child_weight*. 150 individuals and 20 generations are large enough to explore different combinations of parameters. The outcome is either similar performance with different parameter combinations, suggesting that the precise tuning of these parameters is not crucial, or that a similar BAC is produced by a similar combination of these parameters. To elucidate this, let's look at the following plot, representing the scoring metrics of these top-50 individuals.

Top-50 individuals scores

Figure A.4 shows the scoring metrics of the top-50 individuals. By looking at them, we can see that the BAC, which is the optimized metric, is getting similar results throughout the 50 individuals. However, metrics such as accuracy, precision and recall have a larger variation. This, in addition to the high variation in some hyperparameters in Figure A.3, indicates a different learning pattern in some of the classifiers, although they have a similar performance in terms of BAC.

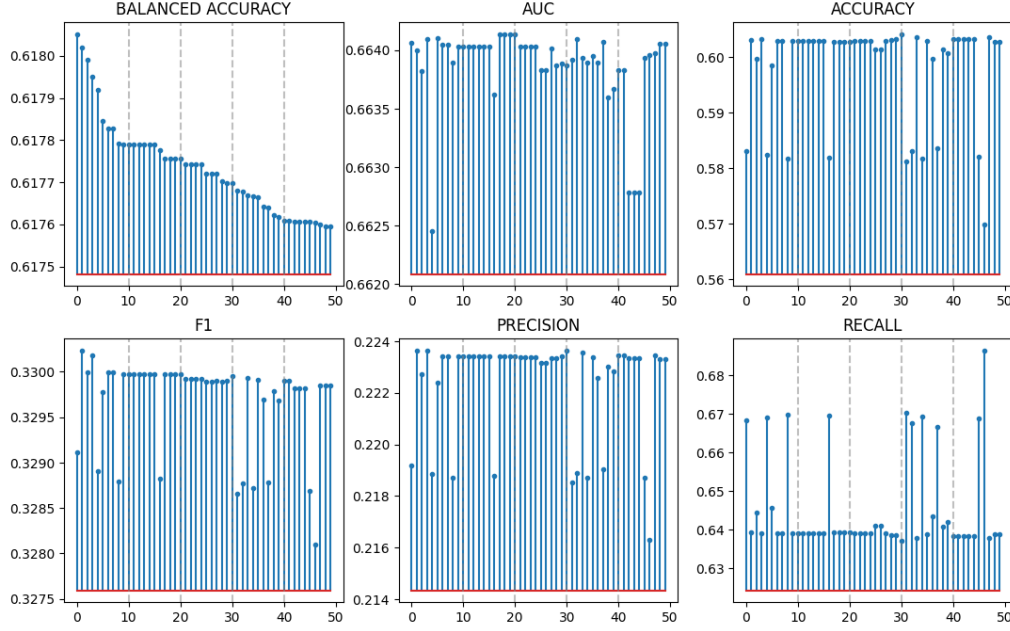


Figure A.4: Top-50 individuals scores.

Although the range of variation of BAC and AUC is small, at least two or three different “types” of classifiers can be seen judging by how they perform in terms of accuracy and recall of the default class (one of them would be the first one, another would be the second, and the last one would be the third one). The precision, recall and accuracy of these “types” of individuals is repeated throughout the top-50 individuals.

This indicates that although they are the same overall, they behave reasonably differently and one could think of combining their predictions to obtain a better classifier. These findings suggest that although these classifiers exhibit similar overall performance, they still exhibit notable behavior differences. Therefore, combining their predictions could potentially yield an improved classifier.

Assessment of elitism in the search for the best classifier

In Appendix A we have presented an evaluation of the impact of parameter tuning on boosting algorithms, and the results indicate that it can have a significant effect on the overall performance of these algorithms.

In this appendix, we will assess the importance of elitism in the genetic algorithm when trying to find the best classifier. By comparing the results of the genetic algorithm with elitism and without it, we will be able to reach a more forceful conclusion on whether hyperparameter tuning matters or not in boosting algorithms, and we will gain insights on the importance of driving the exploration of these hyperparameters towards the best solutions.

The genetic algorithm of this experiment will have the exact same configuration as the one explained in section 3.2, with the only difference being the selection method. We will now use a roulette selection, in which the selection probability of an individual will be proportional to its fitness. This way, we do not have a deterministic elitism as before, in which the best individual from the tournament was always selected.

By performing roulette selection, we give the worst individuals the possibility of being selected, although their probability of selection is lower. It makes sense that the probability of selection of the worst individuals is lower, since otherwise, the population could be worse and worse after each generation.

Utilizing the information provided in the previous appendix, where we presented a comprehensive analysis of XGBoost involving quantitative, categorical, and *BERT_pred* variables, we will now proceed to conduct a similar analysis while excluding the concept of elitism.

Fitness evolution over generations

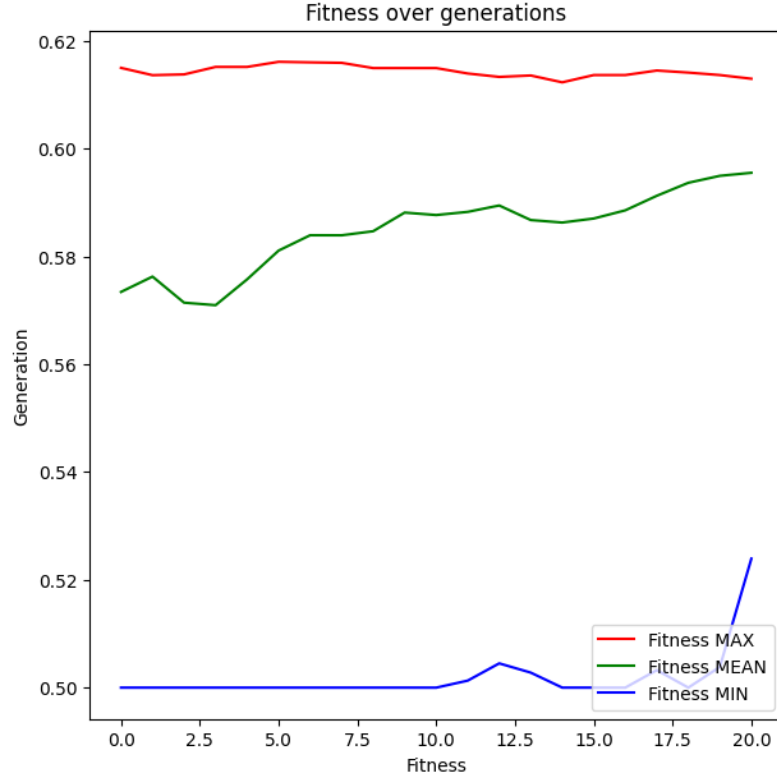


Figure B.1: Fitness evolution.

Due to the absence of elitism, it is expected that there will not be a significant overall improvement in fitness across generations, as illustrated in Figure B.1. In fact, the red line representing the best individual of each generation shows a slight decline over time.

Moreover, as the exploration is not driven towards better individuals, the blue line representing the minimum fitness value tends to remain consistently low in most generations, reaching the minimum value. It is important to note that fitness, in this context, refers to the balanced accuracy of an individual's predictions. A fitness value of 0.50 indicates a complete inability to differentiate between the two classes.

Hyperparameter exploration

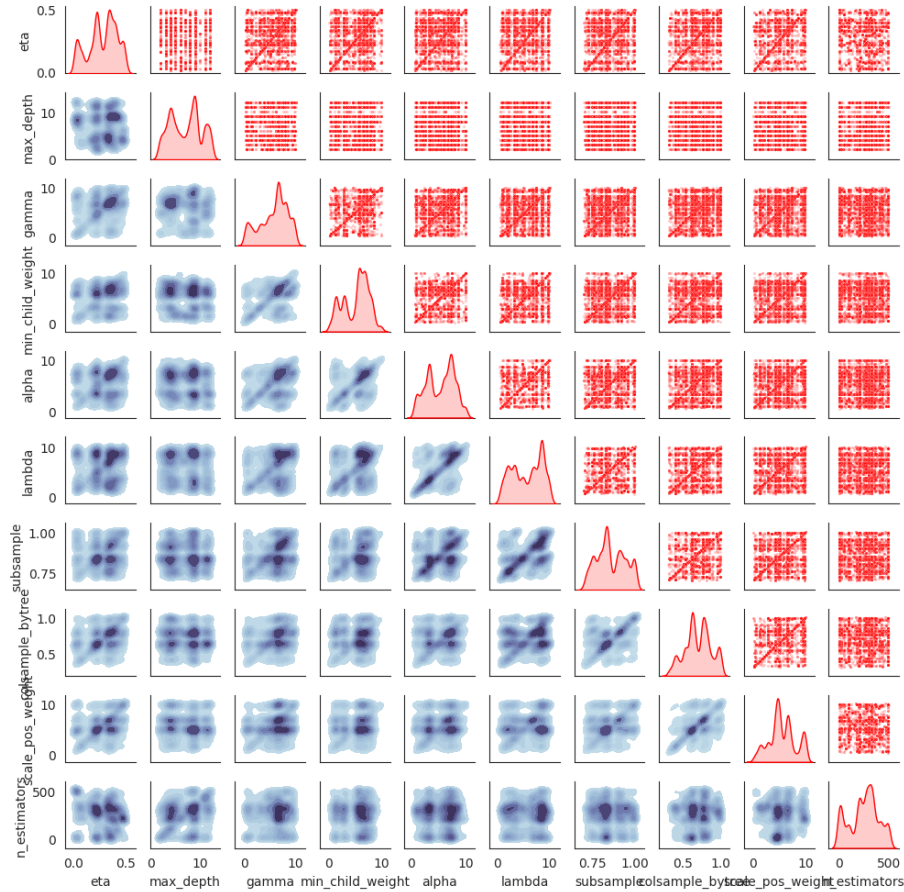


Figure B.2: Hyperparameter exploration.

As expected by not having elitism, we obtain a broader range of hyperparameter exploration that can be appreciated in the main diagonal of Figure B.2. When comparing the density of exploration along the main diagonal with the plots analyzed in Chapters 4 and 6 and presented in Appendix A, we can see that peaks are less pronounced and the density of exploration is more spread out, indicating a wider range of parameter combinations being explored.

Top-50 individuals hyperparameters

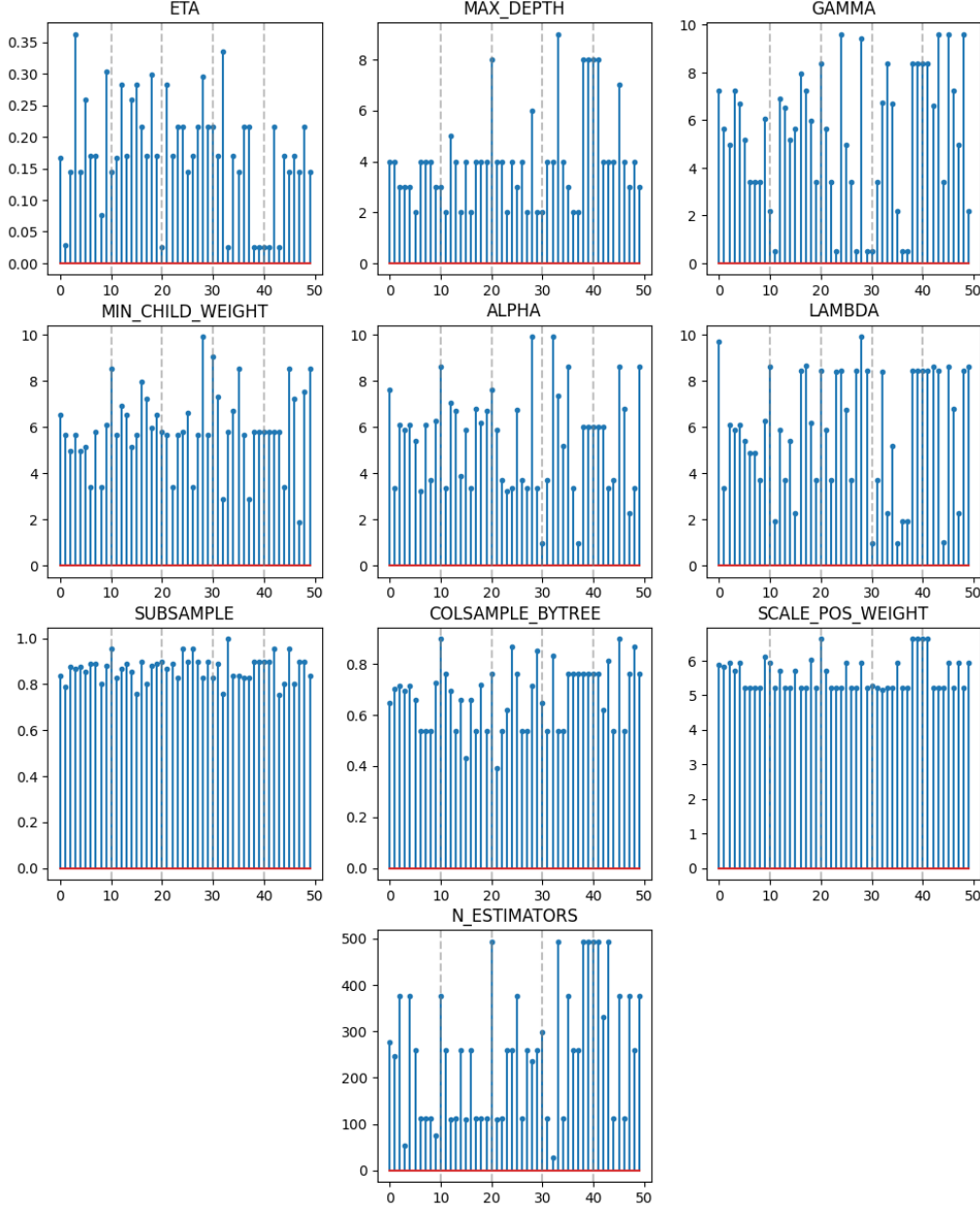


Figure B.3: Top-50 individuals hyperparameters.

Again, as expected, Figure B.3 depicts a high diversity in the hyperparameters values of the 50 best classifiers compared to the ones analyzed in Chapter 4 and presented in Appendix A. There we saw homogeneous hyperparameters that resulted in similar performances.

We will now analyze if this high variance of hyperparameters also implies a larger difference in performance.

Top-50 individuals scores

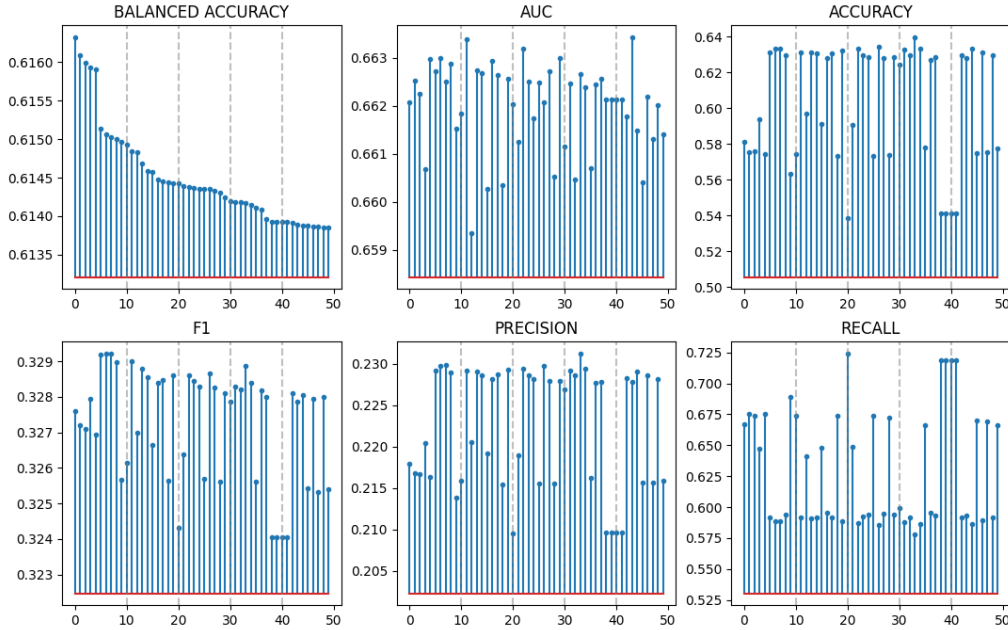


Figure B.4: Top-50 individuals scores.

Figure B.4 shows that not only are the BAC results worse, but also the rest of the metrics are much more varied. These two conclusions are reasonable.

First, by removing elitism, we stopped driving the search towards best solutions. However, the performance of these classifiers is not that far away since, as stated in section 3.2, we are exploring 3,000 different classifiers. By sheer probability, there will be a few that perform almost as well as the best one obtained with elitism.

We can see that the BAC, which is the optimized metric, has been reduced from 0.6181 to 0.6163 with the exclusion of elitism. It might seem like a small reduction (0.2%), but considering that the addition of *BERT_pred* in Chapter 6 had an improvement of 0.3%, this reduction is significant, and could lead to misclassifying several hundreds of loans.

