Universidade de São Paulo [1]
Departamento de Matemática Aplicada e Estatística

Credit Default Modeling: Unbanked Customers

Benedito Faustinoni Neto[2]
Bruno F Bessa De Oliveira[3]
Connor Davis Sterrett[4]
Daniel Shinoda Pascoal[5]
Daniel C F Guzman[6]
Professor: Francisco Louzada Neto [7]

[1]Instituto de Ciências Matemáticas e de Computação
[2]E-mail: benedito.neto@usp.br
[3]E-mail: bruno.fernandes.oliveira@usp.br
[4]E-mail: connorsterrett@usp.br
[5]E-mail: daniel.shinoda@usp.br
[6]E-mail: camiguz89@usp.br
[7]E-mail: louzada@icmc.usp.br

Abstract

This report describes the development of a credit risk model based on the Vietnamese customer data provided by the international finance company Home Credit. Home Credit focuses its products on unbanked and underbanked customers, so the data used includes many atypical credit attributes, such as detailed housing information (e.g. number of floors in the customers' dwelling). The main purpose of this work is to apply concepts learned from the Probability and Statistics Course offered by University of São Paulo during the spring of 2020.

The dataset used in this study is provided by Home Credit on the Kaggle platform as part of a global challenge to produce the most accurate credit model.

Keywords: *machine learning; credit risk; financial services industry; consumer lending; banking; credit score; feature engineering*

1 Introduction

1.1 Subject

The subject of this work is credit risk, which Resti and Sironi [28] define as "the possibility that an unexpected change in a counterparty's creditworthiness may generate a corresponding unexpected change in the market value of the associated credit exposure". The main concept related to our use of credit risk is default risk, i.e., the risk of the borrower not being able to pay the debt.

Access to credit products is granted in most cases to customers that have prior history in the financial system. Meanwhile, 62% of the world's adult population are either unbanked or underbanked [1]. As a result, this population is generally not targeted or granted credit, preventing them from paying for education and investing in businesses. They also may become an easy target for illegal lenders.

1.2 The problem

The difficulty of granting credit to unbanked or underbanked customers is that lenders typically rely on information about customers' previous behavior with credit - thereby establishing an understanding of the customers' financial habits. Home Credit aims to address this issue by providing a collection of data from external sources such as telecom and housing data not typically used for credit modeling.

1.3 Methodology

The steps taken to create a predictive model for Home Credit are as follows:

- Business understanding;
- Data preparation;
- Exploratory data analysis;
- Feature engineering;
- Model building and model tuning;
- Gathering results and conclusions.

Home Credit has two main financial products: cash loans and revolving credit - which they identify in the dataset with the variable ”Contract Name Type”. This variable was later shown to be useful in predicting default.

Feature engineering was important to improve predictive capability of the model. Previous experience in the credit risk area and automatic tools were used to produce a dataset which was better suited for predicting default.

This dataset was originally provided on the Kaggle platform by the Home Credit team as a machine learning exercise. The evaluation criteria of a team’s model was solely the Area Under the Receiver Operating Characteristic Curve (AUC). The ROC curve measures the quality of a binary classifier and the area under this curve is a popular metric for measuring classification algorithms [16]. For our purposes we also wanted to verify the quality of the F-Measure (also known as the F1-Score) as it measures the precision and recall of a model.

2 Dataset Understanding, Exploratory Data Analysis and MVP

We began our analysis of our data by understanding our data structure, as in figure 1:

From the original data we had 219 variables distributed in 7 tables. We used Google Big Query to process this data and create an analytical table that aggregated important attributes from the original tables for each loan.

Exploratory Data Analysis (EDA) is the use of various tools and techniques to get a better understanding of the data, because as Hartwig & Dearing [14] put it in

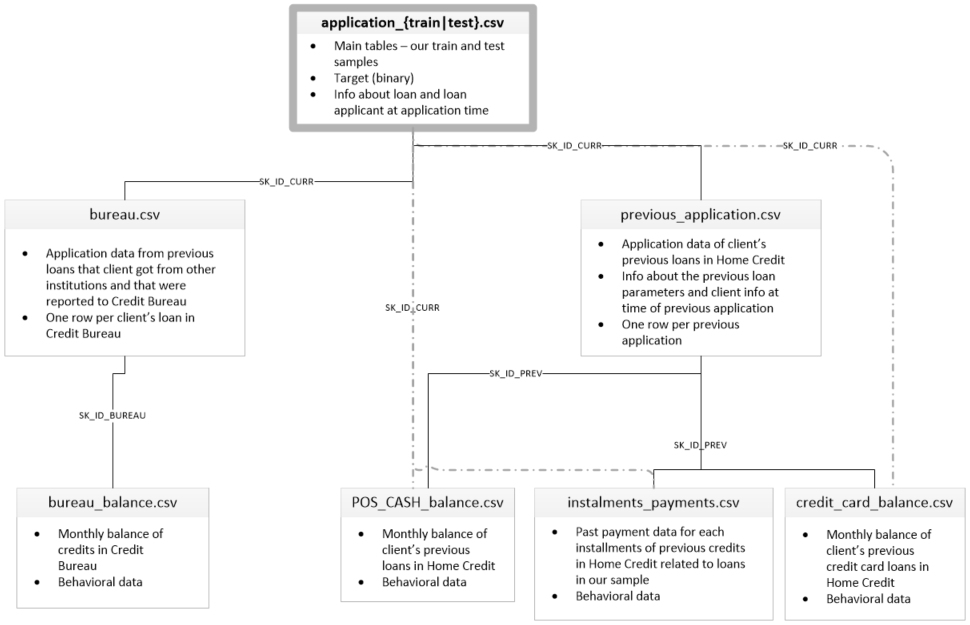

application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_BUREAU
SK_ID_PREV
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

Figure 1: Home Credit's database structure.

their book on the topic, "the more one knows about the data, the more effectively data can be used to develop, test, and refine theory".

Given that we had several hundreds of attributes, we used the Python library Pandas Profiling, which automates the process of generating helpful metadata about attributes such as the attribute datatype, missing/unique counts, quantile statistics, a histogram and more. This report along with the data dictionary provided by Home Credit served as a useful reference for understanding any attribute in our dataset.

We began our manual analysis by understanding our metric of interest, the average default rate, determined by equation 1:

$$\text{default rate} = \frac{n_d}{n} \tag{1}$$

Where n_d = the number of clients who were unable to pay and n = the total number of clients.

In figures 2, 3 and 4 below follows the default rate for some of the major demographic variables in the dataset.

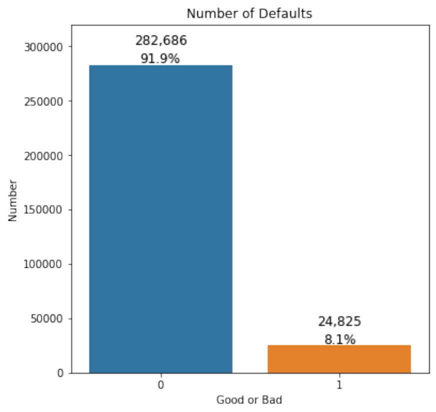
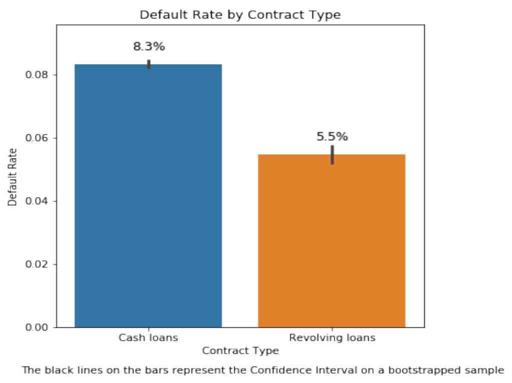
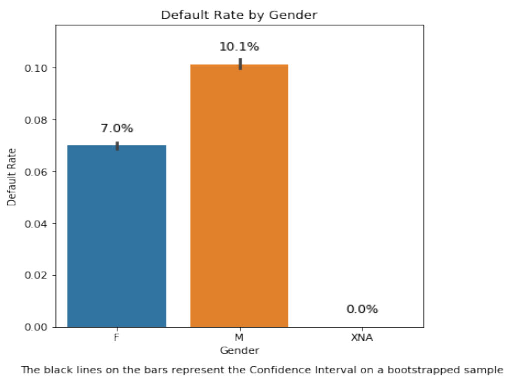


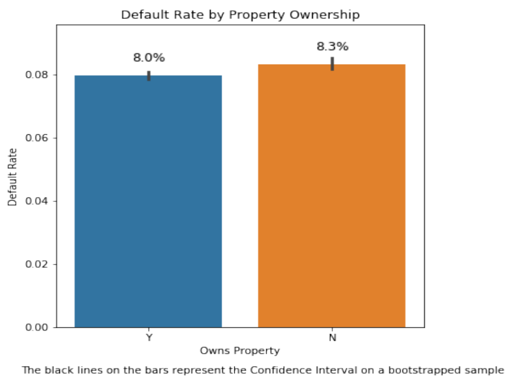
Figure 2: The average default rate for the Home Credit customer is 8.1%.



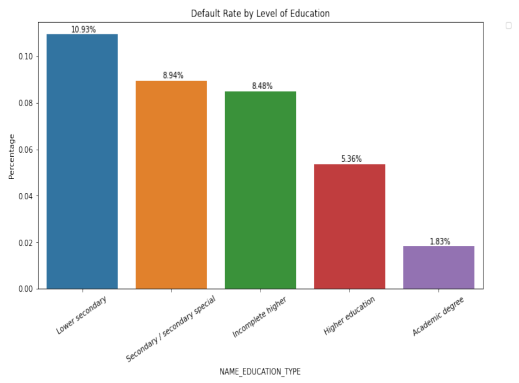
(a) Cash loans are higher risk.



(b) Women are lower risk.



(c) Property owners are lower risk.



(d) As education increases, risk decreases.

Figure 3: Our demographic information gives us a good initial idea of the customer's risk.

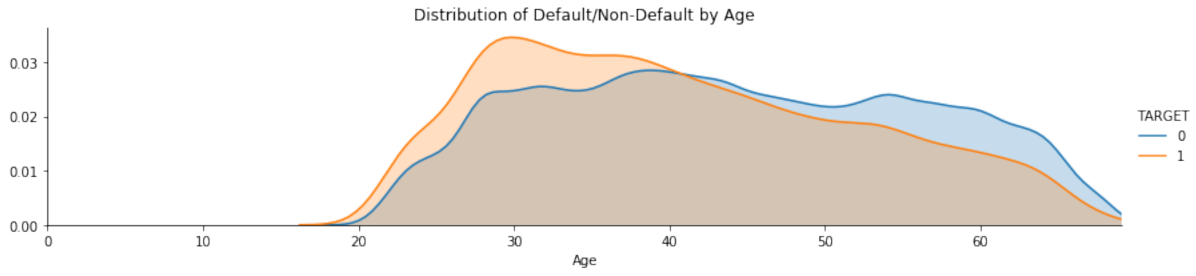


Figure 4: The population who defaults is concentrated in younger ages.

Using these valuable attributes, we were able to create a Minimum Viable Product (MVP).

2.1 MVP - Decision Tree

In order to have an initial understanding of the opportunities and difficulty of modeling the risk of clients in our dataset, we started with a decision tree model [6] using only demographic variables as inputs and setting aside all features related to behavior on previous loans. The performance of this algorithm evaluated on the full dataset is summarized on Table 1 below.

Metric	**Validation Dataset**	**Full Dataset**
Accuracy	**88%**	**88%**
Precision	**9%**	**9%**
Recall	**5%**	**6%**
F1-score	**6%**	**7%**
AUC	**61%**	**61%**

Table 1: Decision Tree Performance Metrics

Once we had verified the opportunity to obtain relevant performance results with the Home Credit dataset, we evolved our work through feature engineering, missing values imputation and development of more complex modelling strategies.

3 Feature Engineering, Missing Values Imputation and Transformations

To improve our model's performance, we first set out to improve the data which is used in the model. To achieve this, we used an approach known as feature engineering, which is described by Zheng & Casari as ”the act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model”. [30]

To that end, we took two approaches to creating new features: Manual Feature Engineering & Automated Feature Engineering.

3.1 Manual Feature Engineering

Manual feature engineering is the process of creating new features by calculating or transforming the dataset by hand. One approach is to apply prior knowledge of the domain to recreate common industry features for one's data. In this case, our team applied our previous experience working in credit risk to recreate common credit risk features.

We then evaluated our newly created features using the metric information value (IV). Information value in the context of credit risk is used to quantify the predictive power of a continuous feature by measuring the feature's ability to separate good and bad payers. To calculate IV, the continuous feature is first split into K bins, and the formula as described by Lund & Brotherton is applied:

$$IV = \sum_{k=1}^{K} (g_k - b_k) * \log \frac{g_k}{b_k} \tag{2}$$

Where $K =$ the number of bins, and g_k and b_k represent the percentage of good and bad payers within the k^{th} bin respectively. [22]

Below are a few of the most powerful predictive features created through manual feature engineering using our prior knowledge of credit risk.

1. Remaining Payments
   (a) **Definition**: The remaining payments left on the client's loan.
   (b) **IV**: 0.143 (9.8x more powerful than the average feature).
2. Client Debt to Limit Ratio
   (a) **Definition**: The ratio of the client's total outstanding debt with all creditors over the total credit available to the client from all creditors.
   (b) **IV**: 0.064 (4.4x)
3. Number of Loans Granted in the Last 12 Months
   (a) **Definition**: The number of loans the client has opened in the last 12 months.
   (b) **IV**: 0.047 (3.2x)
4. Amount Paid Down
   (a) **Definition**: The amount of the loan the customer paid as a down payment at the time the loan was granted.

(b) **IV**: 0.036 (2.4x)

In addition to creating features using previous experience, we created another feature using the K-Nearest Neighbors algorithm. The K-Nearest Neighbors algorithm identifies the closest K neighbors for a specific data point using Euclidean distance as described by Peterson, 2009 [25]. We constructed the algorithm using the four highest IV features which were available and calculated the mean default rate for the 500 nearest neighbors. This variable turned out to be the most predictive variable in our dataset with an IV of 0.901 (61.8 times more powerful than the average feature).

3.2 Automated Feature Engineering

The traditional feature engineering approach is based on both the domain of the knowledge of the analyst and his technical ability for data manipulation. Although the human factor plays a fundamental role in analyzing the data that goes into the model, the feature engineering process can be one of the most time consuming steps since the raw data is usually not in a single table, but rather distributed in several relational tables.

According to Kanter et al. in [18], "While recent developments in deep learning and automated processing of images, text, and signals have enabled significant automation in feature engineering for those data types, feature engineering for relational and human behavioral data remains iterative, human-intuition driven, and challenging, and hence, time consuming." This is illustrated in figure 5

Thinking about the automation of these tasks, Kanter et al. developed the "Deep Feature Synthesis" algorithm, which follows relationships between different datasets and applies mathematical functions to create the final feature. The algorithm stacks different mathematical functions sequentially, the max number of functions to stack defines the "depth" of the algorithm.

This deep feature synthesis algorithm was compiled in a python package called "Feature Tools". The package was applied on the home credit database with 3 datasets: application train, bureau and previous application. For computational purposes we defined the max depth of 2 (stacking only 2 functions), with this approach we were able to build 665 features in total. The process is described in figure 6

After the creation of our automatic and manual features, we found that our newly created features composed 80% of the top 50 most powerful predictive features in our dataset as seen in figure 7:

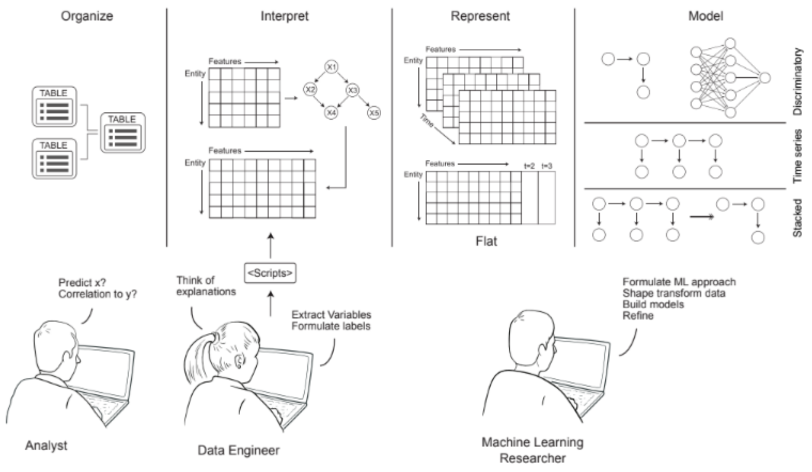
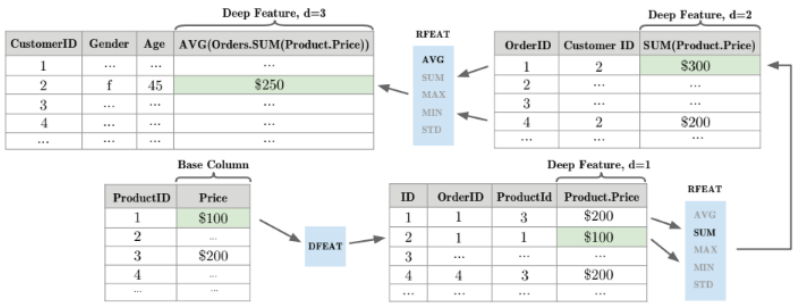


Figure 5: A typical flow of a machine learning project, where the analyst first asks if a particular variable is correlated with the target variable or not from previous domain knowledge. Second, the data engineer uses scripts to extract these features. Third, the variables are used as input for the modeling of the machine learning engineer.



CustomerID	Gender	Age	AVG(Orders.SUM(Product.Price))
1	...	...	...
2	f	45	$250
3	...	...	...
4	...	...	...
...	...	...	...

OrderID	Customer ID	SUM(Product.Price)
1	2	$300
2	...	...
3	...	...
4	2	$200
...	...	...

ProductID	Price
1	$100
2	...
3	$200
4	...
...	...

ID	OrderID	ProductId	Product.Price
1	1	3	$200
2	1	1	$100
3	...	...	...
4	4	3	$200
...	...	...	...

Figure 6: The illustration shows how the original data is aggregated according to the relationships between the tables and various aggregations are repeatedly performed to generate new features (e.g. the average order total per customer).

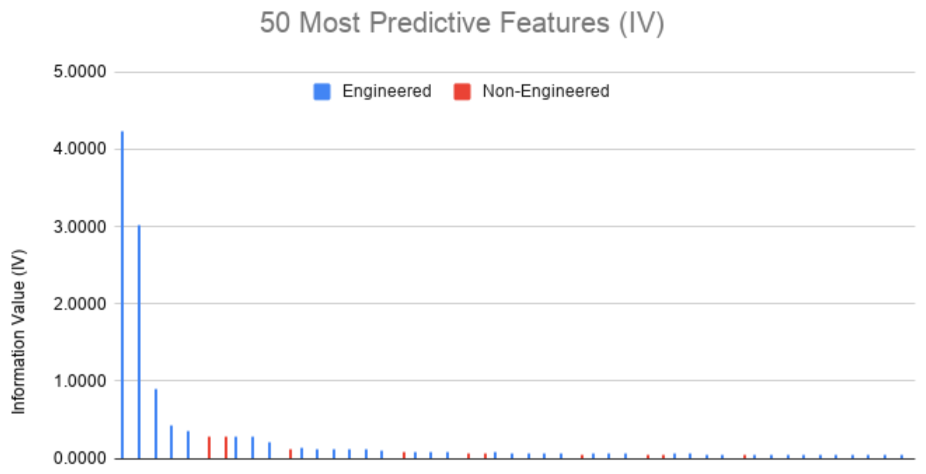


Figure 7: Information value for engineered and non-engineered features.

3.3 Missing values' imputation

The missing values imputation was divided in two main approaches. For the behavioral variables related to past contracts, the imputation was conducted by simply replacing the null values with zeroes, since the absence of information may indicate that the individual has no previous credit history or is a new Home Credit customer.

For the demographic features, we adapted the method proposed in [12] by using decision trees [26] instead of k-means clustering combined with k nearest neighbours [15]. For the feature selection, in this phase of the work we used the chi-square statistic [23] for categorical data, and the Mann-Whitney [10] statistic for continuous-domain variables.

We imputed the missing values in an iterative approach. First, we sorted the missing values in ascending order of missing rate. For each feature, we chose which features to use as the input for the imputation based on the aforementioned statistics. The trees were then trained using the selected input. We then repeated the process for each variable with the added benefit that our newly imputed features were available for future imputation. Figure 8 illustrates the approach.

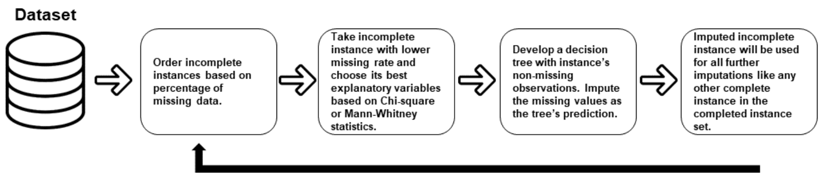

Dataset
Order incomplete instances based on percentage of missing data.
Take incomplete instance with lower missing rate and choose its best explanatory variables based on Chi-square or Mann-Whitney statistics.
Develop a decision tree with instance's non-missing observations. Impute the missing values as the tree's prediction.
Imputed incomplete instance will be used for all further imputations like any other complete instance in the completed instance set.

Figure 8: Missing values imputation methodology for demographic features.

3.4 Feature transformations

There were highly skewed and zero-inflated features in our dataset. In order to address such situations, we've employed some transformations on these instances as an attempt to achieve an improvement on the algorithms' performance.

a) Decision Tree Binning

In order to address the highly skewed features, a decision-tree binning strategy was adopted, as described in [11].

b) Inverse Hyperbolic Sine Transformation

The authors in [8] proposed this transformation as an alternative to the Box-Cox [5] transformation when the variable under study can assume either sign or could be affected by extreme observations. It can be described as:

$$g(y_t, \theta) = g_t = \frac{log(\theta y_t + (\theta^2 {y_t}^2 + 1)^{1/2})}{\theta} = \frac{sinh^{-1}(\theta y_t)}{\theta} \tag{3}$$

The feature selection adopted was the same employed for missing values imputation, that is, Chi-square and Mann-Whitney statistics. The optimization of parameter θ was conducted via Iterated Ordinary Least Squares as described in [29] and [9].

4 Modelling Strategies

We applied various classification algorithms and evaluated the performance metrics of each to identify the best model.

4.1 Neural Networks

Neural Networks are a very flexible class of machine learning algorithms that can be used for multiple types of problems and were proposed by McCulloch [24]. A neural network consists of a system of simple processing elements called neurons that acting together are able to map a certain behavior from input signals (data). Each neuron is configured to respond to the input signal according to a threshold called activation function. In our study we used the swish activation function [27].

4.2 LightGBM

Gradient boosting machines are a very popular class of machine learning algorithms that can be used for classification purposes. The technique combines the tree learning algorithm (weak learner) in series to achieve a strong learner from an ensemble of sequentially connected weak learners. Each learner attempts to minimize the errors of the previous one.

Light Gradient Boosting Machine is an open-source implementation of GBM by Microsoft [19] which reduces the computational complexity of training a model, while achieving almost the same accuracy.

4.3 Stacking

As a way to improve the precision and recall metrics in our dataset, the Neural Networks and LightGBM scores were combined in a linear stacking ensemble, as described in [7] and in [2]. Three different meta-learners were tested, and their particularities are discussed below.

a) Logistic Supervisor

With this approach we used the scores from the LightGBM and Neural Network models as input for a logistic regression [23]. For this regressor, the objective function is called sigmoid and is given by:

$$y = \frac{1}{1 + e^{-z}}, \tag{4}$$

where y is the event probability and z combines the sum of weighted evidences for each feature [20].

b) Naive Bayes Supervisor

With this technique we still wanted a probabilistic approach, but this time considering Bayes' theorem [15]. A difference from logistic regression is that Naive Bayes's Classifier assumes independence for the features.

c) F1-score optimization

We implemented the optimization method proposed in [17] in order to maximize the F1 score and therefore set an optimal balance between precision and recall metrics. The author proposes an approximation for the F-measure via expected utility maximization. The objective function is of the form:

$$\tilde{F}_{\alpha}(\theta) = \frac{\tilde{A}(\theta)}{\alpha n_{pos} + (1-\alpha)\tilde{m}_{pos}(\theta)}, \tag{5}$$

where, $\forall \alpha \in (0,1)$ and $\forall \gamma \in \mathbb{R}_{+}$:

$$\tilde{A}(\theta) = \sum_{\substack{i=1,\\ y=+1}}^{n} g(\gamma\theta x_i)$$

$$\tilde{m}_{pos}(\theta) = \sum_{i=1}^{n} g(\gamma\theta x_i)$$

$$n_{pos} = \text{number of true positive instances}$$

and g is the sigmoid function.

We treated α and γ as hyperparameters to be chosen via grid search on a hold-out set [15]. It is important to point out that the objective function is not concave, so the optimization was conducted several times with random initializations to avoid local maxima. The optimal parameter θ^* was used to combine both scores of the neural network and of the lightGBM ensemble as well through a sigmoid function, producing scores that maximized the F1-score under the traditional $\frac{1}{2}$-classifier threshold.

The F1-score optimization stacking method attained the best results and therefore is the only one we discuss in the following sections.

4.4 Feature Selection

We eliminated 78 categorical features because of their great cardinality, that caused sparsity after creating dummy variables. With the remaining 587 features we used the permutation feature importance technique for feature selection. The permutation feature importance is defined as the decrease in a model score when a single feature values are randomly shuffled. This technique benefits from being model agnostic and can be calculated many times with different permutations of the feature. [6] We used the implementation of the permutation feature importance algorithm within scikit learn to calculate the feature importance for our LightGBM and Neural Network models. We then narrowed our list of 587 features to 164 features by eliminating any features that had an importance of less than 0.01% on both models.

5 Results

In this section we'll discuss the results we've attained with our modelling strategies. We adopted the hold-out validation strategy [15] in order to train and evaluate our algorithms' performance.

The performance metrics selected for evaluation were Accuracy, Precision, Recall, F1-score and AUC [21].

5.1 A Comment on Feature Transformations

Although we expected the transformations mentioned in section 3.4 to result in an improvement in our algorithms' performance, we observed the contrary, as shown in figure 9.

Therefore, we'll present the results of the models developed with input features that were not submitted to the aforementioned transformations.

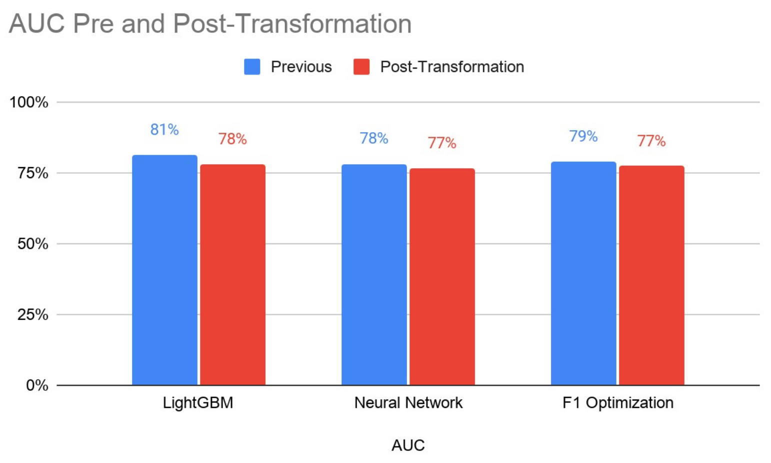


Figure 9: Transformations reduced AUC performance for all models.

5.2 Neural Networks

The chosen network architecture for our problem was a 3-layer ($32 \times 256 \times 32$) fully connected neural network regularized via dropout in the hidden layers ($0.5\times0.5\times0.5$) [13]. The hidden layers were connected through the Swish activation function, the latter being chosen because it outperformed the usual ReLU activation function in our dataset [27].

Metric	Validation Dataset	Full Dataset
Accuracy	**70%**	**70%**
Precision	**17%**	**18%**
Recall	**71%**	**72%**
F1-score	**28%**	**28%**
AUC	**77%**	**78%**

Table 2: Neural Network Performance metrics

5.3 LightGBM

LightGBM was the chosen tree-based ensemble method for our problem, due to the fact that it is the less prone to overfitting in this class of algorithms [19]. After grid search [15], the best parameters for our problem were a 4-depth, 300 estimators and minimum data number on a leaf equal to 300 observations.

Metric	Validation Dataset	Full Dataset
Accuracy	**92%**	**92%**
Precision	**56%**	**66%**
Recall	**4%**	**5%**
F1-score	**8%**	**9%**
AUC	**79%**	**81%**

Table 3: LightGBM Performance metrics

5.4 Stacking: F1-score optimization

As described in section 4.3, the stacking algorithm [7] already uses the scores on the validation dataset as an input to create the stacked model. Therefore, we test overfitting on the stacking model by testing on a third dataset, henceforth called the "extra-validation" dataset. In Table 4, we can see the results on the extra-validation dataset.

Metric	Extra-validation Dataset	Full Dataset
Accuracy	**86%**	**86%**
Precision	**26%**	**28%**
Recall	**41%**	**44%**
F1-score	**31%**	**34%**
AUC	**78%**	**79%**

Table 4: F1-optimization Performance metrics

5.5 An empirical solution: Model Blending

We also tried a simpler approach of combining both neural network and lightGBM scores, creating a weighted average of the models' scores based on their AUC on the validation set.

$$\text{blended score} = \frac{AUC_{valid}^{NN} * NN_{score} + AUC_{valid}^{LGBM} * LGBM_{score} + AUC_{valid}^{OptF1} * OptF1_{score}}{AUC_{valid}^{NN} + AUC_{valid}^{LGBM} + AUC_{valid}^{OptF1}}$$

where:

AUC_{valid}^{NN}: Area under the curve for the Neural Network algorithm on the validation set.

AUC_{valid}^{LGBM}: Area under the curve for the LightGBM algorithm on the validation set.

AUC_{valid}^{OptF1}: Area under the curve for the F1-score optimization algorithm on the validation set.

NN_{score}: Neural Network's predicted score in the full dataset.

$LGBM_{score}$: LightGBM's predicted score in the full dataset.

$OptF1_{score}$: F1-score optimization's predicted score in the full dataset.

Metric	**Full Dataset**
Accuracy	**88%**
Precision	**31%**
Recall	**36%**
F1-score	**33%**
AUC	**80%**

Table 5: Model Blending Performance metrics

6 Conclusion

6.1 Business Analysis

To evaluate the impact of our models, we set out to analyze how the models might be applied for the business of Home Credit. In order to do this, we first needed to know the interest rate of each loan. As this data was not provided by Home Credit, we needed to predict the interest rate based on previous Home Credit data. We calculated the interest rates on the loans previously granted by Home Credit and built a LightGBM model as described in section 4.2 to predict interest rates on new loans. We tuned the hyperparameters of the LightGBM model using a random search as described by Berstra & Bengio (2012). [4].

We found that the interest rates did not match our expectation. Normally one would expect that higher risk loans receive a higher interest rate. This practice is known as risk-based pricing. Our analysis found that the interest rates were uniform across our models' predicted risk classifications as seen in figure 10.

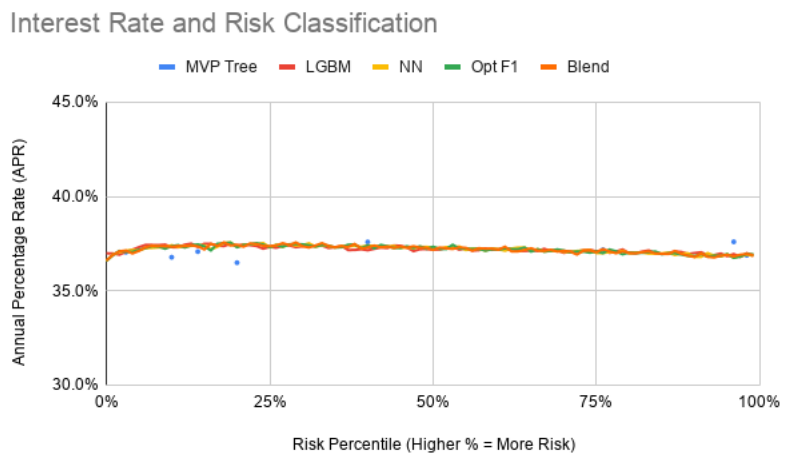


Figure 10: Interest rate is uniform across risk classifications.

This violates the risk-based pricing principle and is not optimizing profit because the higher risk loans have a higher default rate as shown in figure 11.

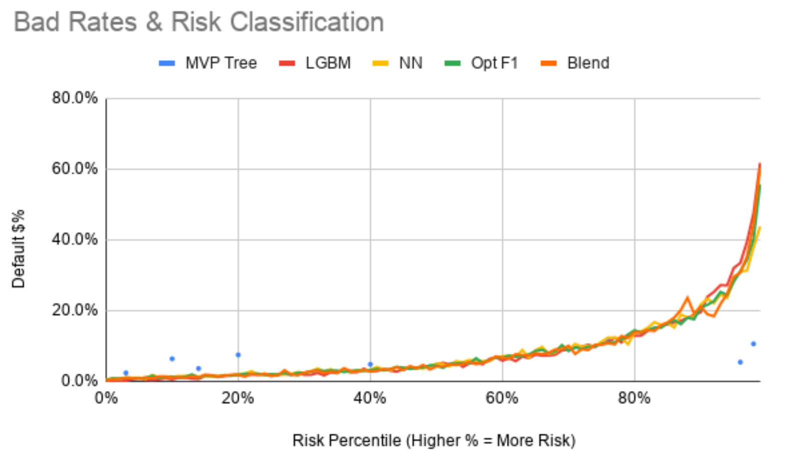


Figure 11: The highest risk classifications have the highest default rates.

We created a Profit and Loss (P&L) to estimate the Return on Assets (ROA) that would be expected at different decline thresholds. The results show that the LGBM model has the highest ROA at any given decline threshold. This is consistent with the finding that the AUC of the LGBM model was the highest (figure 12.)

Home Credit likely has some goals on the quantity of loans to grant, overall risk, ROA threshold, and total profits. Based on their individual situation, they would use this information to select a decline threshold that optimized for these goals.

6.2 Final Thoughts

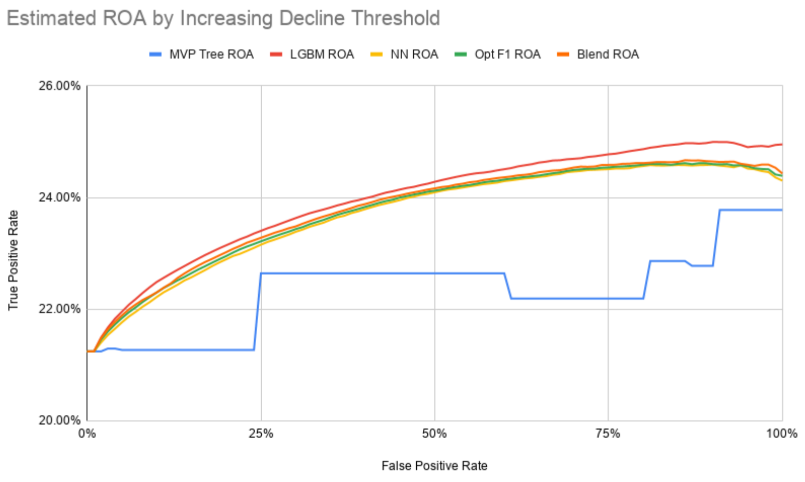


Figure 12: LGBM has the highest ROA at all decline thresholds..

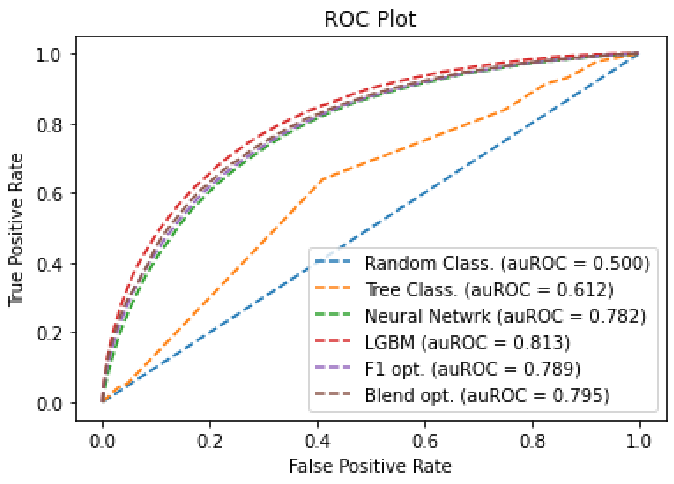


Figure 13: AUC Performance comparison among the different classification strategies.

As we can see the LightGBM model obtained the best performance among the tested models, with an AUC of 81% (figure 13) and the highest ROA at any given decline threshold in the business analysis. In addition to selecting the right algorithm, we found that feature engineering was also very important to our final performance, since the majority of the features with high information value were the engineered ones. Our study also demonstrated that the interest rates were not consistent with the borrowers' risk level, according to our risk rating.

Due to the challenges of granting credit to unbanked customers, we hope that this study will serve as a basis for future implementations of credit models that can serve this audience.

References

[1] R. Aitken. "'All data is credit data': Constituting the unbanked". In: *Competition & Change* 21.4 (2017), pp. 274–300.

[2] Sill et al. *Feature Weighted Linear Stacking*. https://arxiv.org/pdf/0911.0460.pdf. 2009.

[3] E. Baidoo. "A credit analysis of the unbanked and underbanked: an argument for alternative data". In: (2020).

[4] James Berstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13.1 (2012), pp. 281–305.

[5] G. E. P. Box and D. R. Cox. "An Analysis of Transformations". In: *Journal of the Royal Statistical Society* (1964), pp. 211–243.

[6] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[7] J. Brownlee. *Stacking Ensemble Machine Learning With Python*. https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/. 2020.

[8] John B. Burbidge et al. "Alternative Transformations to Handle Extreme Values of the Dependent Variable". In: *Journal of the American Statistical Association* 83.401 (1988), p. 4.

[9] L. Canchen. *Implementing Box-Cox Transformation for Linear Model*. https://frost-lee.github.io/box-cox-linear-model/. 2019.

[10] W. J. Conover. *Practical Nonparametric Statistics*. 3. Wiley, 1999.

[11] A. Dubey. *Discretisation Using Decision Trees*. https://towardsdatascience.com/discretisation-using-decision-trees-21910483fa4b. 2018.

[12] S. Gajawada and D. Toshniwal. "Missing value Imputation Method Based on Clustering and Nearest Neighbours". In: *International Journal of Future Computer and Communication* 1.2 (2012), p. 4.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. 2. The MIT Press, 2016.

[14] F. Hartwig and B. E. Dearing. *Exploratory data analysis*. 16. Sage, 1979.

[15] T. Hastie, R. Tibishirani, and J. Friedman. *The Elements of Statistical Learning*. 2. Springer, 2016.

[16] M. Hossin and M. Sulaiman. "A review on evaluation metrics for data classification evaluations". In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (2015), p. 1.

[17] M. Jansche. "Maximum Expected F-Measure Training of Logistic Regression Models". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, 2005, pp. 692–699. URL: https://doi.org/10.3115/1220575.1220662.

[18] James Max Kanter and K. Veeramachaneni. "Deep feature synthesis: Towards automating data science endeavors". In: *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (2015), pp. 1–10.

[19] G. Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 3146–3154. URL: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[20] Vlado Keselj. *Speech and Language Processing Daniel Jurafsky and James H. Martin (Stanford University and University of Colorado at Boulder) Pearson Prentice Hall, 2009, xxxi+ 988 pp; hardbound, ISBN 978-0-13-187321-6,* 115.00. 2009.

[21] D. K. P. Kroese et al. *Data Science and Machine Learning: Mathematical and Statistical Methods*. 1. Chapman&Hall, 2019.

[22] Bruce Lund and David Brotherton. "Information Value Statistic". In: Midwest SAS Users Group. Magnify Analytics Solutions, a Division of Marketing Associates. Columbus, OH, 2013. URL: https://www.mwsug.org/proceedings/2013/AA/MWSUG-2013-AA14.pdf.

[23] P. McCullag and A. Nelder. *Generalized Linear Models*. 2. Chapman&Hall, 1989.

[24] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[25] L. E. Peterson. "K-nearest neighbor". In: *Scholarpedia* 4.2 (2009). revision #137311, p. 1883. DOI: 10.4249/scholarpedia.1883.

[26] Md. G. Rahman and Md Z. Islam. "A Decision Tree-based Missing Value Imputation Technique for Data Pre-processing". In: *AusDM*. 2011.

[27] P. Ramachandran, B. Zoph, and Q. V. Le. *Searching for Activation Functions.* 2017. arXiv: 1710.05941 [cs.NE].

[28] A. Sironi and A. Resti. *Risk management and shareholders' value in banking: from risk measurement models to capital allocation policies.* Vol. 417. John Wiley & Sons, 2007.

[29] J. J. Spitzer. "A Primer on Box-Cox Estimation". In: *The Review of Economics and Statistics* 64.2 (1982), pp. 307–313.

[30] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning.* O'Reilly Media, Inc., 2018.