NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

EXPLAINING THE PREDICTIONS OF A BOOSTED TREE ALGORITHM

Application to Credit Scoring

Pierre Antony Jean Marie Salvaire

Dissertation presented as partial requirement for obtaining the Master' degree in Statistics and Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

EXPLAINING THE PRDICTIONS OF A BOOSTED TREE ALGORITHM

APPLICATION TO CREDIT SCORING

by

Pierre Antony Jean Marie Salvaire

Dissertation report presented as partial requirement for obtaining the Master's degree in Information Management, with a specialization in Business Intelligence and Knowledge Management

Supervisor: Rui Goncalves

February 2019

ABSTRACT

The main goal of this report is to contribute to the adoption of complex « Black Box » machine learning models in the field of credit scoring for retail credit.

Although numerous investigations have been showing the potential benefits of using complex models, we identified the lack of **interpretability** as one of the main vector **preventing from a full and trustworthy adoption** of these new modeling techniques. Intrinsically linked with recent data concerns such as individual **rights for explanation**, **fairness** (introduced in the GDPR[1]) or **model reliability,** we believe that this kind of research is crucial for easing its adoption among credit risk practitioners.

We build a **standard Linear Scorecard model** along with a more advanced algorithm called **Extreme Gradient Boosting** (XGBoost) on a **retail credit open source dataset**. The modeling scenario is a **binary classification** task consisting in identifying clients that will experienced 90 days past due delinquency state or worse.

The interpretation of the Scorecard model is performed using the raw output of the algorithm while more complex data perturbation technique, namely **Partial Dependence Plots** and **Shapley Additive Explanations** methods are computed for the XGBoost algorithm.

As a result, we observe that the **XGBoost algorithm is statistically more performant** at distinguishing "bad" from "good" clients. Additionally, we show that the **global interpretation of the XGBoost is not as accurate as the Scorecard algorithm**. At an individual level however (for each instance of the dataset), we show that the level of interpretability is very similar as they are **both able to quantify the contribution of each variable to the predicted risk of a specific application**.

KEYWORDS

Credit Scoring, XGBoost, Model Interpretation, Black Box

[1] General Data Protection and Regulation

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

PDP: Partial **D**ependence **P**lot

SHAP: SHapley **A**dditive ex**P**lanations

XGBoost: e**X**treme **G**radient **Boost**ing

WoE: Weight **O**f **E**vidence

Roc Auc: Area **U**nder the **C**urve

GDPR: General **D**ata **P**rotection **R**egulation

PD: Probability of **D**efault

1 Introduction and Motivations

1.1 Credit Scoring

Credit scoring can be defined as a quantitative method used to measure the probability that a loan applicant or existing borrower will default (Gup & Kolari, 2005). For each individual, a score is calculated according to the probability of default that was given by a statistical model. The final score is most commonly based on demographic characteristics and historical data on payments. During the modeling process, the algorithm identifies and learns how these characteristics are related to credit risk. Later on, the algorithm applies these patterns to a new population and assigns a score to a new customer. Low scores correspond to very high risk, and high scores indicate almost no risk.

After defining a score for each individual, the decision maker has to choose a cutoff score above which loan applications will be rejected. This decision is made according to the risk appetite and strategy that one wishes to put in place. "*These techniques decide who will get credit, how much credit they should get and what operational strategies will enhance the profitability of the borrowers to the lenders*" (Thomas et al., 2002).

Credit scoring applications in banking sector have expanded during the last 40 years (Banasik and Crook, 2010), especially due to the growing number of credit applications for different financial services, among which consumer loans is considered to be one of the most important and essential in the field (Sustersic et al., 2009).

In this growing environment, the need for an automated credit scoring process over the global financial system have been a key driver of the development of credit scoring techniques. As a result, banks developed industry standards for building, evaluating and regulating models and credit scoring processes. Scorecards models, based on Logistic Regression algorithm, became the most widely used tool for building scoring models (Abdou & Pointon, 2011 - Thomas, Edelman & Crook, 2002, p27 - Siddiqi, 2017).

According to many practitioners and as reported by Hasan (2016), there is three main advantages of using Logistic Regression based Scorecards in consumer loans credit scoring:

- The reduction of time for evaluating new applications. Applications can be scored instantly without complex computations.
- Its simplicity; "the scorecard is extremely easy to examine, understand, analyze and monitor".

- Finally, its building process being highly transparent, the scorecards can easily meet any regulatory requirement.

1.2 Challenger Models: Potential benefits

Over the last years, new computational capacities and the exponential growth of available data drastically transformed the industry of predictive analytics.
New techniques such as ensemble methods, optimization algorithms or neural networks brought new opportunities and challenges to the entire community of predictive modelling practitioners, and subsequently, to the field of credit scoring.

The outperforming results of algorithms such as Random Forest or Gradient Boosting over Logistic Regression are already well studied and documented. Results tend to show that those new algorithms are clearly better than Logistic regression in term of error rate.

These scientific advancements can be observed in the work from Marie-Laure Charpignon, Enguerrand Horel and Flora Tixier (2014) in which they built models on a consumer finance dataset (kaggle: give me some credit Post crisis data) using Logistic Regression, Random Forest and Gradient Boosting. They compared the results in term using different metrics and the results tend to show that although Gradient boosting and Random Forest tend to overfit the data, they are clearly outperforming Logistic regression on the aspect of performance.

In 2010, Khandani et al also contributed in showing the potential of using more advanced non-parametric methods in assessing credit risk. They show how this technology can be applied to parallel fields such as preventing systemic risks. "*we are able to construct out-of-sample forecasts that significantly improve the classification rates of credit-card-holder delinquencies and defaults, with linear regression R2's of forecasted/realized delinquencies of 85%*" (Khandani et al., 2010).

In a more practical way, Manuel A. Hernandez and Maximo Torero investigated this potential by using data from micro lending business in rural Peru. Their results tend to show that "*significant improvements on the accuracy of risk ranking are possible when relying on less structured, data-driven methods to construct scores based on default probabilities, particularly when the odds of defaulting or repaying are not necessarily linear with respect to all of the covariates"* (Hernandez & Maximo, 2014).

Showing the impact that the adoption of artificial intelligence could have on profit margins, a study conducted by McKinsey & Company (Columbus, 2017) and demonstrates that companies who fully supported AI initiatives have achieved 3 to 15% percentage point higher profit margin, around 12% in the financial industry as shown in the figure 1. Over the 3,000

business leaders who were interviewed for their survey, the majority expect margins to increase by up to 5% points in the next year.

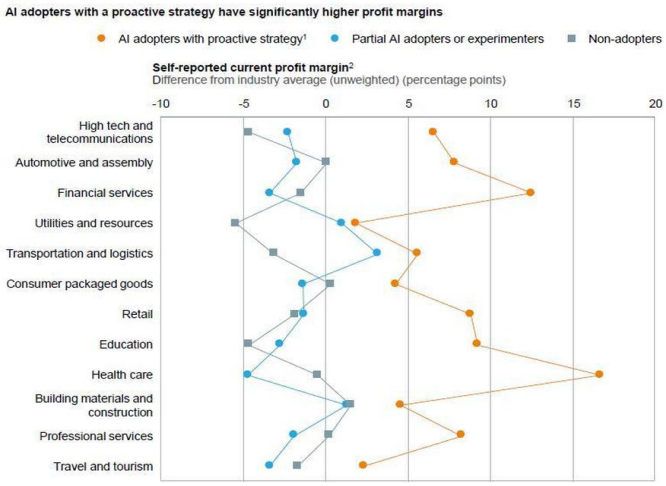


Figure 1: AI Adopters Segmentation (Columbus, 2017)

Corroborating the observations made by McKinsey, Knut Opdal, Rikard Bohm and Thomas Hill (Knut et al., 2017) conducted a research discussing "*whether machine learning and automatic hyperparameter optimization represent disruptive technologies for risk management*". Their experiment shows that using a Random Forest instead of a Logistic Regression could lead to an 8% rise of expected profit.

However, while more and more researchers are exploring the use of machine learning and its benefits for credit scoring, banks and credit institutions are making very cautious steps toward its adoption. Thus, the ongoing domination of Logistic Regression in these industries.

As reported in an article from American Banker called "*Is AI making credit scores better, or more confusing?*" (Crosman, 2017) and as shown in the figure 2 below, if new technology can potentially bring more statistical performance and new lending opportunities, they might also bring more opacity to the credit scoring process and negatively impact the entire credit financial cycle that was built around it.

Pros	
More precise scores, through more nuanced evaluations of data	Ability to consider consumers that linear models would reject
Cons	
Risk that regulators will consider the score a "black box"	Challenge of providing one clear reason for credit denial

Figure 2: The two faces of AI-based credit scores (Crosman, 2017)

1.3 "Black Box" Dilemma

One of the most attractive features of new machine learning tools is that it is intended to solve problems that simpler algorithm can't. The improvement on that aspect could open new market opportunities through the use of new data sources. However, the use of these techniques comes at cost inherent to their functioning, commonly defined as being a "*Black Box*" (Guidotti et al., 2018).

The term "Black Box" comes from the fact that models created through the process of ensemble models are very complex and their intricate functions cannot be understood by humans. Although practitioners may have a general understanding of the internal flow of an algorithm, the exact path to the output decisions, that may be based on thousands of rules, remain unexplained (Hara & Hayashi, 2016).

While this might not be a significant issue for other industries, the credit system is generally greatly "constrained" by validation, monitoring, reporting and regulations processes that should be considered when using a given modeling tool. As an example, credit scores must usually (at the very least) come with some sort of verbal or written explanation as mandated by the General Data Protection Regulation (GDPR), which theoretically grants citizens a "*right to an explanation*" in the event that an automated (machine learning based) decision could "significantly affect" them. Thus, the use of a scoring equation that is totally opaque and unexplainable could go against some basic citizens' Rights (Wachter et al., 2017; Doshi-Velez & Kim, 2017). In this context, a simpler, interpretable models[2] such as decision trees, rules (Letham et al., 2015) or linear models (Ustun & Rudin, 2015) will be preferred even if they don't have the most predictive performance (Ribeiro et al., 2016).

Recently, Zachary C. Lipton (2018) tried to identify the rationale behind the interest in studying models' interpretability. It is one of the rare papers to mention, among other field, the interest of interpretability in credit scoring. He points out the fact that "*According to their own technical report, FICO trains credit models using logistic regression, specifically citing interpretability as a motivation for the choice of model*" (Lipton, 2017).

While the prevailing solution to the issue of the Black Box is to use interpretable models at the cost of lower accuracy (Bastani et al., 2018), research on the topic has been growing rapidly

[2] Model that gives the possibility of easily inspecting its components (e.g., a path in a decision tree or the weight of a feature in a linear model)

in recent years[3] and new methods for complex model decision approximation have emerged (Gilpin et al., 2018).

Most of these alternative approaches are all based on the principle of being model-agnostic, so that they can be applied to any decision system. These techniques can involve the superposition of a simple model on top of a Black Box (Craven & Shavlik, 1996) or the perturbations of the original input and analysis of how it impacts the output (Fong & Vedaldi, 2017).

In August 2016, Marco Tulio Ribeiro et al (Ribeiro et al., 2016) heavily contributed to the development of surrogate[4] interpretation by introducing LIME (**L**ocal **I**nterpretable **M**odel-Agnostic **E**xplanation); "*a novel explanation technique that explains the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction*". By assuming that although a complex model might not be simple (or linear) at a global decision level, local region of the decision space might happen to be simple. Consequently, making it possible to fit a simpler model for a specific input region and being able to explain a set of predictions locally. The experiments they've made presented evidence that LIME technology can be used in a variety of models in the text and image domains.

In 1996, Mark W. Craven and Jude W. Shavlik (Craven & Shavlik, 1996) presented a method for extracting Tree-Structured representations of trained networks. Their method is similar to LIME technology, but it is specific to neural networks algorithm, as they are approximating simple decision trees to different vectors of the network.

Similarly, Sameer Singh (Singh et al., 2016) proposed to approximate the model locally by using what they call "programs". A "program" is a set of basic, human friendly set of syntax ("OR", "AND", "IF", "ABOVE", "BELOW", "EQUAL") that are combined with variable for explaining different region of the output space. The method they proposed is highly expressive and can be understood by any human.

Another conceptual framework for approximating a model decision output is through data perturbation. By perturbing the input data and analyze the impact it has on the final output, many practitioners were able to significantly increase their understanding of the model.

In the literature, one can find numerous examples of complex algorithm being better understood with the help of Partial Dependence Plots (PDP), a perturbation base interpretation method introduced by Friedman in 2001 (Friedman, 2001). By plotting the change in predictions of instances as a given feature(s) is perturbed, Green and Kern (2010) were able to understand

[3] Google Scholar finds more than 20,000 publications related to interpretability in ML in the last five years (Doshi-Velez & Kim, 2017).

[4] Simple model to superposed on top of a complex system (Black Box model in our case)

how the conditional average treatment effect of a voter mobilization is impacted by each variable. In a different context, Elith (Elith et al., 2008) used PDPs to understand how different environmental factors can influence the distribution of a particular freshwater eel while using a gradient boosting.

By inspecting the results of the PDP, Qingyuan Zhao and Trevor Hastie (Zhao & Hastie, 2017) were able to extract causal information from their model. Doing so, they insist on the fact that some domain specific knowledge (business knowledge) is a necessary condition to be applied with the use of PDP method.

In 2017, Ruth C. Fong and A. Vedaldi proposed a "*comprehensive, formal framework for learning explanations as meta-predictors*" (Fong & Vedaldi, 2017). By perturbing the input of an image classification algorithm, they managed to understand which part of the image contributed the most to the final prediction.

Another type of perturbation was introduced a few years later, this one based on coalitional game theory[5] (Minhai, 2017) and on the Shapley value. The main idea is to decompose the changes in prediction when a set of given value are "ignored", meaning they are not included in the model. Results tend to demonstrate that the method is efficient and that the explanations are "*intuitive and useful*" (Strumbelj & Kononenko, 2010). Using a very similar method, Jianbo Chen (Chen et al., 2018) demonstrate that either on language and image data this method compares favorably with other methods using both quantitative metrics and human evaluation.

In 2016, the Shapley Value was further studied by Lundberg (Lundberg, 2016) that created an open source framework (available on Python) for using this new tool. As for as we know, no literature explicitly tested the use of their implementation. In the methodology section (see Shap section) we'll investigate its current implementation and we will be discussing its outputs on our test use case in the result discussion section of this work.

1.4 Properties of Interpretations

While studying previous literature about model interpretability, we noticed that although many approaches were developed, there is no consensus about a clear definition or technical meaning of what an explanation is (Lipton, 2017).

As a very general ground rule, we could safely state that interpretability is most often associated with the capacity of a human to understand the cause of a decision (Gilpin et al., 2018 – Miller, 2019 - Kim et al., 2016), but the practical form it takes e.g., completeness, compactness

[5] A coalitional (or cooperative) game is a model of interacting decision-makers that focuses on the behavior of groups of players.

or comprehensibility, remain mostly subjective to each work (Guidotti et al., 2018). Even in the scope of GDPR, there is no information about the expected properties of required explanation (Rudin, 2018).

For each studied paper, it seems that authors are usually adapting their technical expectations of the interpretation according to the field of their work (e.g., image recognition, medicine …) or to the method they are implementing. On this matter, it seems legitimate to imagine that different technical representations of Black Box' explanations could be appropriate for different kinds of users and fields of application (Singh et al., 2016).

In that context, a distinction which we believe fundamental has nonetheless started to emerge. An interpretation method can be considered as being Local or Global level (Guidotti et al., 2018). Although most literature reviewed only implicitly specify if their proposal is global or local, we try to give an overview of the distinction of both as it strongly impact the methodology of our work.

- **Global Interpretation**

Interpreting a model globally means representing the internal functioning of a trained model in a human understandable way (Yang et al., 2018). It is usually done by an aggregation of instance explanations over many (training) instances (Robnik-Sikonja, 2017) or by simply reading the model's output (linear models). Global interpretation of complex algorithms is known to be either not possible, or too simplistic to represent the original model, thus, the recent development of tools (local surrogate models) to interpret the model locally (Ribeiro et al., 2016).

- **Local Explanation**

Interpreting a model locally means that you can focus on a single instance and examine what the model predicts for this specific input, and explain why (*Molnar, 2019*). It usually comes with providing the impact of input feature values on the prediction (Robnik-Sikonja, 2017)

Despite the soaring attention to the topic, the general conceptual framework of model interpretability has not been defined. Consequently, it is still hard for any business or practitioner to rely on such technologies since we can hardly set expectation regarding their uses, thus making difficult to understand its potential benefits.

1.5 Motivations and Methodology

In this work, we first intend to demonstrate the benefits of using a complex Black Box model over a standard Logistic Regression Scorecard by comparing it using state of the art statistical performance metrics. In a second time, we try to understand if the use of two

interpretation techniques, namely Partial Dependence Plots and Shapley Additive Explanation could help in understanding the functioning of a specific complex model.

By applying these methods in the specific field of Credit Scoring, we hope to contribute to a better understanding of what interpretation methods are and what to expect from them in this particular field. The growing interest of using Black Box models and the recent questions around data regulation are opening a new, promising, area of investigation. In that context, we hope that our work could open the way for further empirical studies that are necessary to back up the work being done by a raising community of researchers and practitioners.

Throughout the following paper, we build two statistical models, a Scorecard algorithm, already mentioned, along with a more complex one, considered as a Black Box tool usually referred to as Extreme Gradient Boosting (XGBoost).

The scorecard algorithm was built using James, the flagship product of James startup used by many financial institutions for assessing risks at the time of credit application. The product was embedded with state-of-the-art techniques and is the results of many years of research from the James' team and its partners within the financial industry.

The XGBoost algorithm, not yet available inside of James, was built in python, using building and optimization techniques that are as close as possible from James, in order to test both models in the same conditions. Given the very recent development of this tool and some of its components that will be used here, this work also opens the way for evaluating its potential applicability in the Credit Scoring context.

Before to analyze the results of each algorithm in the ***Results and Discussion*** section of this document, we will first provide the reader with contextual information about the data that is going to be used and its preprocessing. We will then cover the basic theoretical knowledge that we believe necessary to understand both the current implementation of the Scorecard in the Credit industry and the functioning of the XGBoost model. Some basic knowledge will be given about the way the model hyper parameters optimization is done within James (and reproduced in Python) without getting in much details since it is not the scope of our research.
Finally, we will define the evaluation and interpretation metrics that will be used for evaluating our two models, necessary step before to jump into the actual evaluation and interpretation of both algorithms.

2 Data Description and Cleansing

It is not uncommon to find the expression "Garbage in, garbage out" when reading about predictive modelling. This expression shows the importance of having relevant data in its best shape in order to be possible to construct a good model.

2.1 Data Discovery

The data used in this project comes from the competition "Give me some credit" launched on the website Kaggle. It consists of 150,000 consumers, each characterized by the 10 variables described in the table below.

Variable	Description	Type
Serious Delinquency 2 years	Person experienced 90 days past due delinquency or worse within 2 years	Y/N
Revolving Utilization of Unsecured Lines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits	percentage
Age	Age of borrower in years	integer
Number of Time 30-59 Days Past Due Not Worse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years.	integer
Debt Ratio	Monthly debt payments, alimony, living costs divided by monthly gross income	percentage
Monthly Income	Monthly income	real
Number of Open Credit Lines and Loans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g., credit cards)	integer
Number of Times 90 Days Late	Number of times borrower has been 90 days or more past due.	integer
Number Real Estate Loans or Lines	Number of mortgage and real estate loans including home equity lines of credit	integer
Number of Time 60-89 Days Past Due Not Worse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years.	integer
Number of Dependents	Number of dependents in family excluding themselves (spouse, children etc.)	integer

Table 1 Data Dictionary

As described in the table above, the event that we will be trying to predict is given by the Serious Delinquency 2 years variable. It indicates if the person that was granted a loan experienced 90 days past due delinquency or worse in the 2 years after receiving the loan. Equally said, it is an indication of 3 consecutive months without payment the installment. In the data, it is coded as "1" if the event occurred and "0" otherwise. By calculating some basic proportion statistics, we get the table below:

SeriousDlqin2yrs	Count	Rates
< 90 days past due	139974	93,32%
> 90 days past due	10026	6,68%

Table 2 : Default Rate Analysis

As one can see, the proportion of people that experience 90 days past due is 6.68%, which represent 10,026 cases.

Some basic descriptive statistics of the rest of the variable is given in the table below.

	mean	Standard Deviation	min	25% Quantile	Median	75% Quantile	max
Revolving Utilization of Unsecured Lines	5.90	257.04	0.00	0.04	0.18	0.58	50708
Age	51.29	14.43	0	40	51	61	103
Number of Time 30-59 Days Past Due Not Worse	0.38	3.50	0	0	0	0	98
Debt Ratio	26.60	424.45	0	0.14	0.30	0.48	61106
Monthly Income	6670	14384	0	3400	5400	8249	3008750
Number of Open Credit Lines and Loans	8.76	5.17	0	5	8	11	58
Number of Times 90 Days Late	0.21	3.47	0	0	0	0	98
Number Real Estate Loans or Lines	01.05	1.15	0	0	1	2	54
Number of Time 0-89 Days Past Due Not Worse	0.19	3.45	0	0	0	0	98
Number of Dependents	0.85	1.15	0	0	0	2	20

Table 3 : Descriptive Statistics

As one can observe, we can see some inconsistencies in the data by looking at this table. For example, we see that the minimum age is “0” (zero) which does not make sense in the context of credit application. In a later section, we’ll go through each variable, study their distribution and clean intervene in such inconsistencies as the one for *Age* variable.

2.2 Missing Values

Missing data arise in almost all serious statistical analyses. Since most statistical models and machine learning algorithms rely on a data sets that is free of missing values, it is of a major importance to handle the missing data appropriately. Simple methods may suffice sometime in some case (imputation with the mean or omission of the instances with missings). Some algorithm like CART naturally account for missing data and there is no need for imputation (Wagstaff, 2004). In many other situations, missing values should be imputed prior to running statistical analyses.

We can observe in the table below the number of missing and the percentage of instances it represents for each variable in our original dataset.

Variable	Number of Missings	% of Missings
Serious Dlqin 2 yrs	0	0,00%
Revolving Utilization of Unsecured Lines	0	0,00%
Age	0	0,00%
Number of Time 30-59 Days Past Due Not Worse	0	0,00%
Debt Ratio	0	0,00%
Monthly Income	29 731	19,82%
Number of Open Credit Lines and Loans	0	0,00%
Number of Times 90 Days Late	0	0,00%
Number Real Estate Loans or Lines	0	0,00%
Number of Time 60-89 Days Past Due Not Worse	0	0,00%
Number of Dependents	3 924	2,62%

Table 4 : Missing Values

With 29,731 missings (≅20% of missing value) for the *Monthly Income*, imputing with a standard technique such as the mean or the median might significantly affect the distribution of the variable. Doing so, there is a risk that the learning of an algorithm would end up being biased in some way (if not treated specifically for this algorithm). For this reason, the 29,731 instances with missings were removed from the data. Doing so, all the missings from the *Number of Dependents* variable were also removed in the process.
We finally ended up with a clean (no missing values) dataset consisting of 120, 269 instances.

2.3 Outliers

Before to go through each variable for which potential outliers were detected in our case, it is important to highlight the specificity of the 2 algorithms that we will explain and use. Each type of algorithm deals differently with outlier and may require special preprocessing.

For the scorecard algorithm, all the continuous variables will be bucketed, i.e. variables are categorized into logical intervals, for ease of interpretation and implementation. This truncation that is usually used in credit scoring is known for significantly reducing the effect and influence of outliers/ extreme values (Bolton, 2010).

Regarding the XGBoost algorithm, it is an ensemble model based on CART[6], which is known to be extremely resistant to outliers (up to a certain number) (Sutton, 2005).

[6] **C**lassification **A**nd **R**egression **T**ree

Due to the points above, our strategy for the treatment of outlier will be:

- Removal of outlier if they represent a too significant number of instances (limit of 1,000);
- Imputation with the median of the same variable if it represents a low number of instances and that the value is obviously not possible (a negative income for example);
- No action if the extreme nature of the value is doubtful and if it represents a low number of instances.

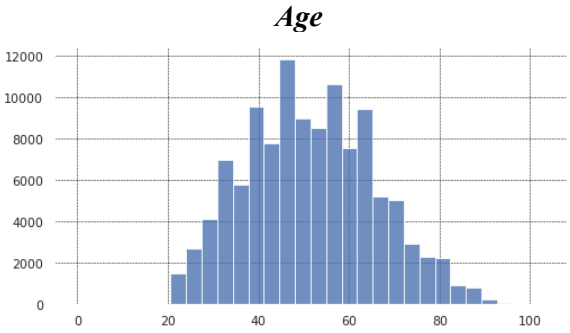


Figure 3: Age Distribution

As one can observe, the *Age* variable is ranging from 0 to 100. But to be qualified as a borrower, the person must be at least 18 years. There were one only record with a value of "0", obviously being an outlier. Hence, we imputed this instance with the median age (51 years old).

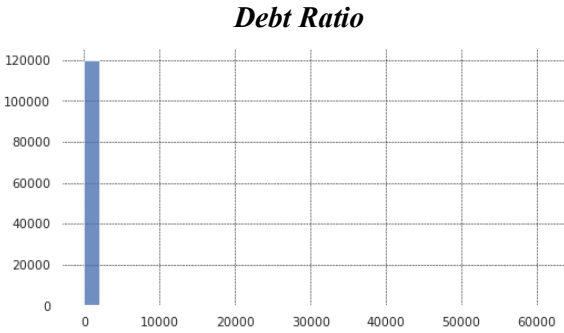


Figure 4: Debt Ratio Distribution

Although a Debt Ratio may exceptionally be higher than 1 (100%), we identified 2,106 cases that had a value over 10 (1000%). Such a significant number might result in affecting the learning of the algorithm, so we decided to remove those instances. After doing so, the new dataset consists in 118,163 instances.

Number of Dependents

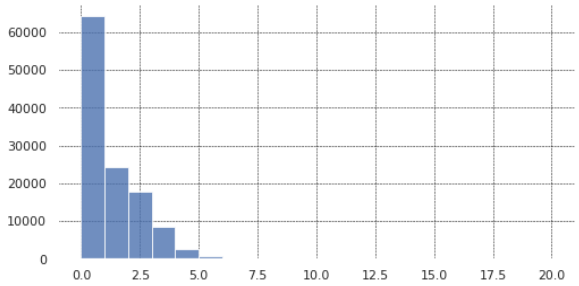


Figure 5: Number of Dependents Distribution

In that case, we identified 11 cases for which the *Number of Dependents* was above 8. Although it may be the result of an error in the application form or from other source, it is a hard statement to make because the maximum is not so far from the end of the distribution plot. Also, we considered that the algorithm should be robust to such a low number of concerned instances and that any kind of treatment was not necessary.

Number of Times 90 Days Late

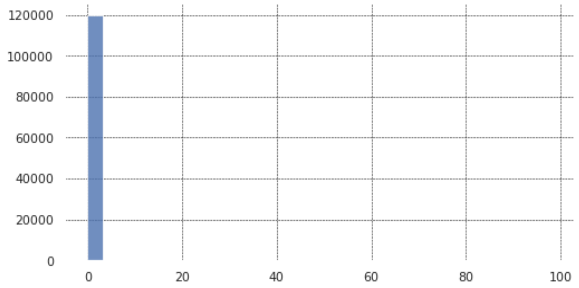


Figure 6: Number of Times 90 Days Late Distribution

For this variable, we identified 148 cases for which the Number of Time 90 Days Late was above 20. We considered that the algorithm should be robust to such a low number of concerned instances.

Monthly Income

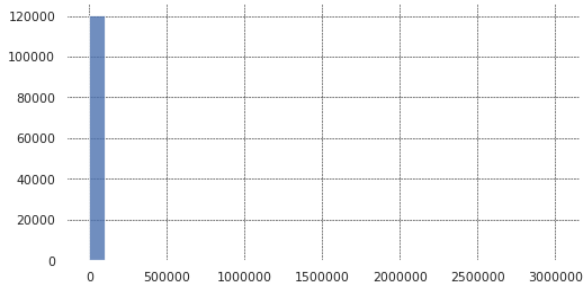


Figure 7: Monthly Income Distribution

For the *Monthly Income*, we identified 300 cases above 50,000. We considered that the algorithm should be robust to such a low number of concerned instances.

Revolving Utilization of Unsecured Lines

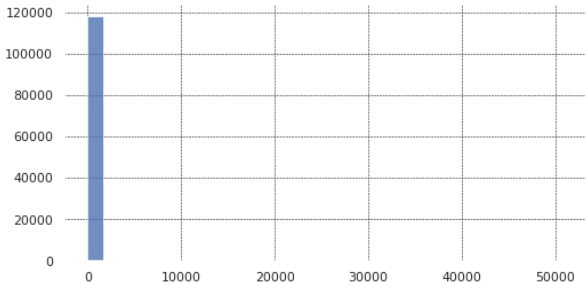


Figure 8: Revolving Utilization of Unsecured Lines Distribution

In this case, we found it weird to be above 1 (100%) because credit card usually have a limit that we cannot overpass (possible for some payments so extra fees can add up). On the graph above, we can observe an abrupt fall in the distribution after 1. We identified 2,773 cases above 100% and removed them from the analysis since it could affect the algorithm.

After treating all the outliers, the dataset ended up with 115,426 instances of credit application.

3 Variable Selection and Data Partitioning

The transformed variables were assessed in terms of their power in discriminating between "good" and "bad" clients. In order to do that, statistic indicator Univariate GINI was used.

3.1 Univariate GINI index

GINI index is a measure for quantifying the ability of a numeric feature to distinguish between classes (Zhao et al., 2010). It is used by many practitioners as a feature evaluation and selection tool. It has shown to be quite performant in many domains as it can significantly improve the learning performance compared to several existing feature selection criterions (Singh et al., 2010 - Liu et al., 2010)

Index	Variable	Univariate GINI
0	Revolving Utilization of Unsecured Lines	0,496
1	Number of Time 30-59 Days Past Due Not Worse	0,348
2	Number of Times 90 Days Late	0,282
3	Age	-0,227
4	Number of Time 60-89 Days Past Due Not Worse	0,216
5	Monthly Income	-0,152
6	Debt Ratio	0,14
7	Number of Dependents	0,093
8	Number of Open Credit Lines and Loans	-0,064
9	Number Real Estate Loans or Lines	-0,049

Table 5 : Univariate GINI

As shown in the table 5 above, *Revolving Utilization of Unsecured Lines* variable is showing the more importance in term of Univariate GINI. As a credit risk analyst, we would expect a high and frequent utilization of the credit limit to be an important vector of risk since it may be a symptom of a particular behavior of not managing well debts and monthly payments.

The importance of 30-59 and 90 days past due is easy to interpret as they describe a past state of delinquency, as the one we are trying to predict (90 days late). It is interesting to note that 60-89 days past due has a relatively lower importance while it represents a very similar behavior.

3.2 Correlation Analysis

Any of the independent variables (predictor) that would potentially be included in the model cannot be highly correlated with another variable. If so, the accuracy and stability model we will build might be compromised.

Therefore, a correlation assessment was performed using the Pearson coefficient which is a measure of the strength of a linear association between two variables. This correlation metric attempts to draw a line that best fit through the data points of two variables by reducing as much as possible the distance between the points and the line. The Pearson correlation coefficient is a generalization of these distance of all points from the line.

In case of having highly correlated variables (above 30% correlation), we removed one of the correlated variables based on two conditions:

1) The potential discriminatory power (Univariate GINI statistic) of the two variables
2) Business/human (intuitive) considerations

The figure 9 below gives a visual illustration of all the correlation of our dataset.

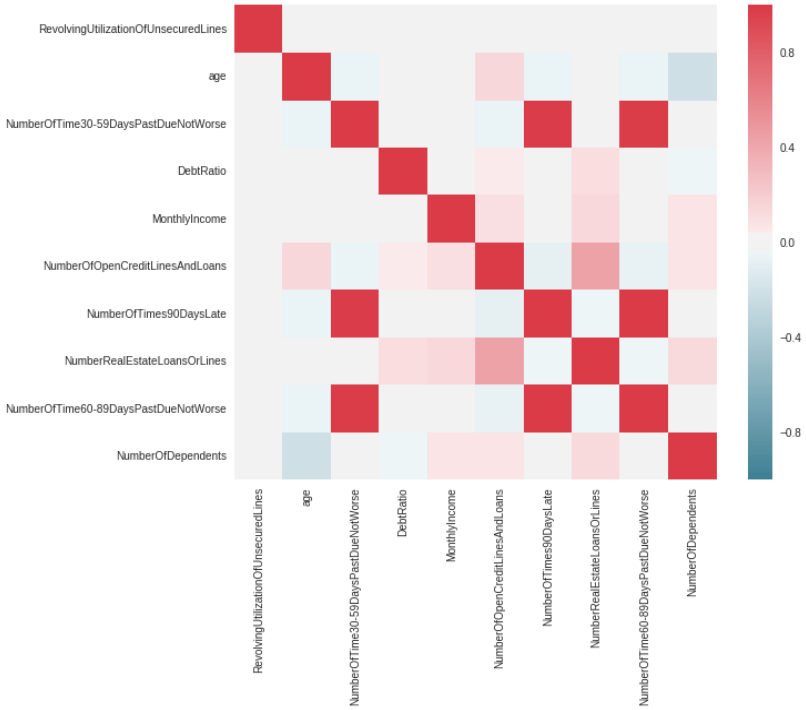


Figure 9: Correlation between 10 variables (Numeric table in annex n°2)

Having a look at the correlation table, we can identify 6 variables that have at least a correlation above 30%. Namely;

- *Number of Time 30-59 Days Past Due Not Worse,*
- *Number of Time 60-89 Days Past Due Not Worse,*
- *Number of Times 90 Days Late,*
- *Number of Dependents,*
- *Number Real Estate Loans or Lines,*

- *Number of Open Credit Lines and Loans,*
- *Age.*

In the list above, the variables that were finally removed were highlighted in red for better visualization purposes, for each one of them, the rationale is explained below.

Number Real Estate Loans or Lines and *Number of Open Credit Lines and Loans* variables are showing a correlation of 43%. For this pair, *Number Real Estate Loans or Lines* variable was removed because it has a lower Univariate GINI (-0.08 against -0.07). As one could expect, *Number of Time 30-59 Days Past Due Not Worse***,** *Number of Time 60-89 Days Past Due Not Worse* and *Number of Times 90 Days Late* are cross-correlated above 98%.

We chose to keep *Number of Times 90 Days Late* variable although it didn't have the highest Univariate GINI because it is the one that is the closest from the target in term of business definition. Compared with the two others, *Number of Time 60-89 Days Past Due Not Worse* variable the lowest Univariate GINI.

The new correlation table after removing the three aforementioned is presented in the figure 10 below. In this final set of variables, the maximum correlation is 21% (between *Age* and *Number of Dependents*)

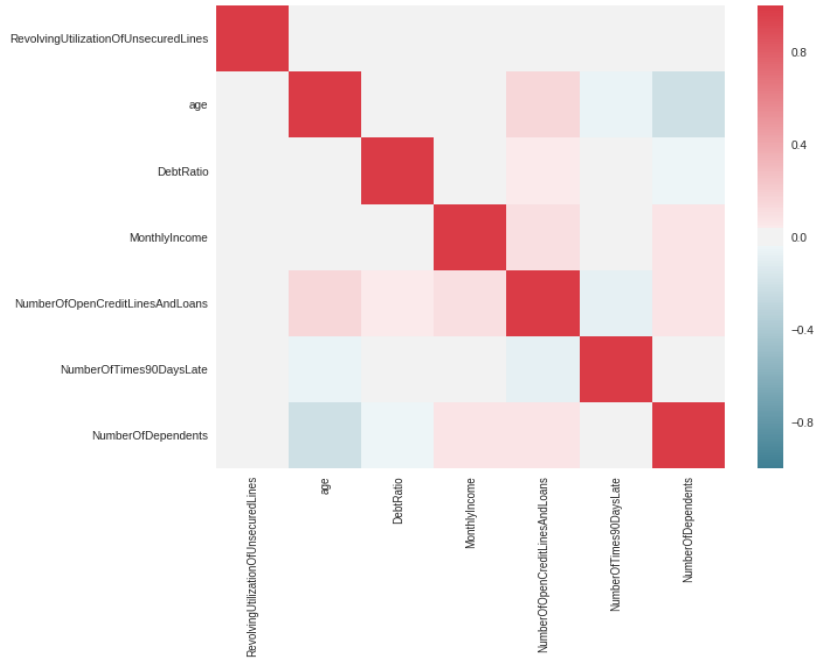


Figure 10: Correlation between 7 variables (Numeric table in annex n°3)

3.3 Sampling: Model Training and Testing

Model validation is primarily a way of measuring the predictive reliability of a statistical model and keep control of the learning procedure. The basic idea is to use part of the dataset to train the classifier (train set) and another part of the data to test the classifier (test set) as if it was "new data". The aim is to maximize the accuracy of the model (bias) while minimizing its complexity (variance).

The **holdout method** is a very simple kind of model validation method and is one of the most commonly used. It consists in separating the data into two sets usually referred as the training set and the test set (see figure 11 below). The algorithm will be ran looking only at the training set and we use the test set to evaluate its performance.

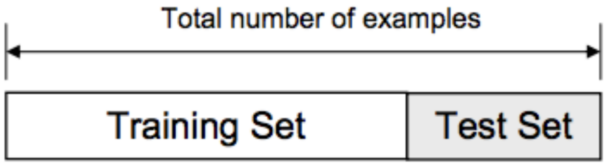


Figure 11: Holdout Train Test Split

This validation method is known to lead to a low accuracy (bias or overfitting) since the evaluation heavily depends on how the data points are distributed between the training and the test. Therefore, the final evaluation may differ depending on the data distribution, leading to a poor generalization with new "unseen" data.

According to many studies, the traditional approach for tackling this bias or overfitting problem is k-fold cross-validation (Kim, 2009). This method, embedded within James product, is a generalization of the hold out method in which meanwhile the data set is randomly partitioned into K subsets, each subset is used as a test set, considering the other K-1 subsets as training set (Witten et al., 2016). In this approach, the entire dataset is used for both training and testing the model. Typically, a value of K=10 is used in the literature (Mitchell, 2017).

In our specific case, we will follow the specification of the model validation made in James, detailed below and illustrated in the figure nº12 below.
James tool is using a combination of the two methods mentioned above, that is, after splitting the data set into train and test set, the training set is again split into 3 validation sets for the fitting of the algorithm.

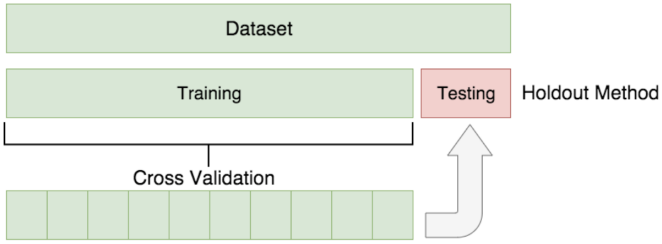


Figure 12: Validation and testing methodology

As no date variable was part of the data provided, we used a stratified random sampling method for partitioning the data between the train and the test set of the holdout method. The stratify method allow to keep the same default rate between the train and the test set. A splitting configuration of 20% of the data allocated to the test set was chosen.

In the end, the train set contains 92340 observation and the test set contains 23086 observations.

4 Modelling Techniques

4.1 Scorecard

4.1.1 Logistic Regression

While a lot of different models can come to mind when talking about linear models, we will focus our attention on linear and a derivation of it, Logistic Regression.
As for any predictive algorithm, the main goal of a Linear Regression is to generalize an event using characteristics that we assume to be a partial description of a wider phenomenon. In our case, we want our model to estimate a probability of a *90 Days Past Due Or Worse* happening or not during the credit lifetime.

In the case of linear regression, the interaction between different predictors and a target response or event is represented by a straight line that will become the estimated predictor for each data observation. The shape and the slope of this line is defined by a set of parameters minimizing the mean squared error (average distance between the estimated predictor and the actual value to be estimated). The parameters of the Linear Regression are defined by the formula:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

β_0 is the intercept (5 on the graphic below) and β_1 is the slope of the red line. The figure 13 below represents the observations (blue dots) and the regression line (red line) on a (X, Y) axis, X being a single predictor and Y a continuous Target.

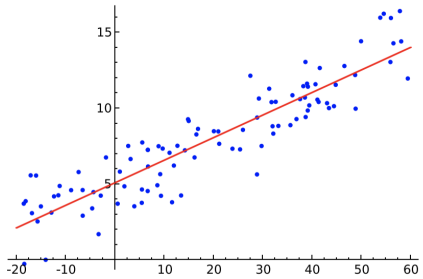


Figure 13: Linear Regression

In the case of credit scoring, the event we are trying to predict is binary. Drawing a line to represent its relationship with another characteristic would therefore not make much sense. In fact, using a Linear Regression to predict this type of dependent variable would be violating several mathematical assumptions inherent to the concepts that defines the Linear Regression. A

visual illustration of the problem is given by the figure 14 below, where 2 possible outcomes (1 or 0) of an event Y are associated with a characteristic X:

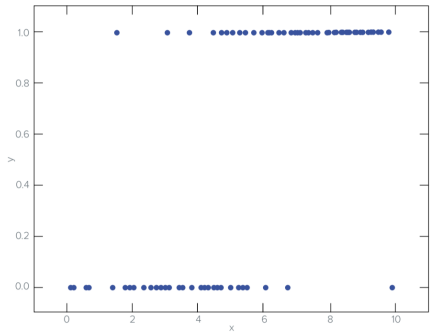


Figure 14: Binary Problem Representation

To resolve this dilemma, we should assume that the probability law that defines the probability of Y being 1 given the characteristics of an individual (otherwise written $P(Y = 1 \mid X = x)$) is Logistic. The Logistic Regression is therefore assumed to be following a logistic law defined by the cumulative distribution function:

$$F(x) = \frac{1}{1 + exp^{-x}}$$

By looking at the figure 15, one can observe that the Y axis is no longer defined by the dependent variable but by the probability of this dependent variable to be equal to 1 (it would be the probability that a customer defaults in credit scoring).

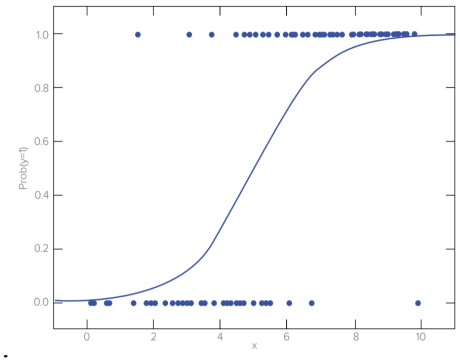


Figure 15: Logistic Regression

When fitting a Logistic Regression, we try to find the optimal parameters values that will be associated with one or several predictors. Those parameters are usually optimized toward the maximization of the likelihood[7] of the event Y happening or not.

For the sake of illustration, we will consider that we only have two predictors X_1 and X_2. The model is therefore:

[7] Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are set in a way that the produced output of the model is as close as possible from the event that is actually observed.

$$P(Y = 1 \mid X) = \frac{1}{1 + exp^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2)}}$$

With this function, we then estimate β_0, and β_1 and β_2 (with likelihood maximisation) and define the decision boundary of the model.

For model interpretation purposes, it is important to point out that the final values of each Beta is a constant measure of change of the logit transformed probability of Y for one unit change in the input variable. The equation for the logit transformation of a probability of an event is given by:

$$Logit\ (p_i) = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k$$

where p is the posterior probability of the event, X are input variables, β_0 is the intercept of the regression line and β_k are the parameters associated with each variable.

The logit transformation can be interpreted as the log of the odds, that is, $log(\frac{P(event)}{P(non\ event)})$. It is used to linearize posterior probability and limit outcome of estimated probabilities in the model between 0 and 1. By selecting a cutoff out of the logit transformed probabilities given by the Logistic Regression, we are then able to classify each observation into the class 1 or 0. This is why the Logistic Regression is often referred to as a linear classifier.

As a parametric method, the Logistic Regression is not free from several critics, mainly residing in the assumptions it takes such as:

- linear relationship;
- independent error terms;
- uncorrelated predictors;
- use of relevant variables.

These assumptions (like the linear relationship), when violated, naturally compromise the model accuracy. The limited complexity associated to the Logistic Regression is the main disadvantage in comparison to other (non-parametric) techniques. On the other hand, it has advantages compared to this later group of methods such as having results that are easy to interpret (due to the log constant scaling) and usually requiring less data.

4.1.2 Scorecard

A Credit Scorecard can be defined as a mathematical procedure that tries to estimate the likelihood of a customer to display a particular behavior (to be in default for example).
This predictive procedure can be based on any model but the most common way to go is to use a Logistic Regression (Anderson, 2007).

The output of the scorecard is a score that can take different range (e.g., 200 - 800). A customer having a low score would be considered at risk of displaying the event we're trying to predict (to be in default for example). On the other hand, a high score indicated a low chance of displaying the given event.

The overall score of an individual is calculated from a scorecard table in which a number of "points" are associated with different characteristics that were used in the model. All those "points" are aggregated for each individual in order to obtain their final score. (Anderson, 2007) An example of such a table is given below in the figure 16.

Characteristic	Attributes					Points
Years @ address	<3 years 30	3–6 years 36	>6 years 38		Blank 35	38
Years @ employer	<2 years 30	2–8 years 39	9–20 years 43	>20 years 64	Blank	43
Home phone	Given 47				Not Given 30	30
Accom. status	Own 41	Rent 30	Parents 39		Other 36	41
Bankers	Us 49	Them 42			Blank 42	49
Credit card	Bank or travel 75		Retail or garage 43		Blank 43	43
Judgments on bureau	Clear 20	1 –16	2 –30	3 –54		20
Past experience	None 3	New 13	Up-to-date 36	Arrears –1	Write-off Reject	3
					Final score	267

Figure 16: Example Scorecard (Anderson, 2007)

At first sight, it might be hard to understand the relationship between the table presented above and the Logistic Regression explained in the previous section of this work.

To get an accurate intuition about this, we need to go a step before the modelling phase and focus our attention on how the data was prepared before the training of the Logistic Regression. Indeed, the construction of a scorecard involved the careful binning[8] of every numerical variable. Doing this, each observation of a data set is no longer represented by the actual value of its attributes but by its affiliation to a range of value (for numerical variables). This method is well suited for credit scoring for its ease of interpretation and implementation. The selection of the ranges for each variable is usually performed using a combination of business knowledge and statistical insights.

The comprehension of the client portfolio might indeed impact the way the ranges should be created. One might want to create groups that respects operational and business considerations

[8] Also called Discrete binning or bucketing.

(using some strategy in grouping postal codes or choosing defined ranges to coincide with corporate policies for example) (Siddiqi, 2017).

To make sure that, during the binning process, each group we create is somehow differentiable from other groups (in term of default rate as an example), the weight of evidence (WOE) of the different categories are examined and grouped together if they have similar relative risk (if the WOE is similar).

The Weight of Evidence is a heavily used measure in credit scoring for assessing the "strength" of a grouping (Garla, 2013). It is defined by:

$$WOE_i = ln((\frac{N_i}{\Sigma_{i=1}^{n} N_i})/(\frac{P_i}{\Sigma_{i=1}^{n} P_i}))$$

where P is the number of occurrences (default), N is the number of non-occurrences (non-default) and i the index of the attribute being evaluated. A precondition for the calculation is non-zero values for all N_i and P_i .

The WOE does not consider the proportion of observations, so it measures a relative risk, given the overall event rate (default rate). A negative WoE indicates that the proportion of defaults is higher for that attribute than the overall proportion and indicates higher risk. By nature, the weight of Evidence is also commonly used for feature selection, since having a binned variable with very disparate WoE for each group indicate that this predictor is strong in differentiating goods from "Bads".

However, the statistical strength is not the only factor to take into account for considering a variable as a meaningful predictor.

In the figure nº 17 presented below, one can observe an example of a binning procedure on Age variable.

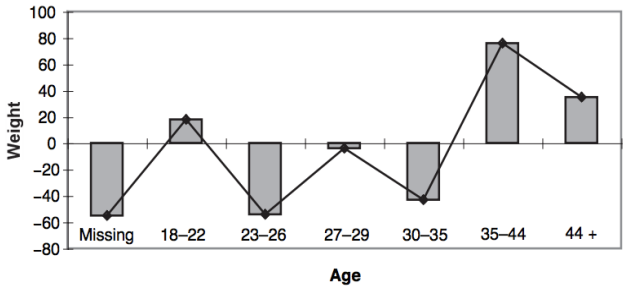


Figure 17: Binning of Age Variable (Anderson, 2007)

Missings information are always grouped together as a separate group as shown on the left part of the graph (Zeng, 2014).

By looking at this simple line, we can see that population ranging between 23-26 and 30-44 years old have a very strong relative rate of event (default rate). On the other hand, 35-44 years

old ranging population is relatively less exposed to the event occurrence (default). In between, we observe strong reversals that, in a lot of cases, might be going against business experience or operational considerations. In the example above, a lot of decision makers would be expecting a linear relationship between the Age and the risk.

Applying a Logistic Regression on the example above (without adjusting the bins) would potentially lead to the attribution of more "credit points" (less risk) to 18-22 years old population than for the 30-35 years old one. In between, the opposite trend is observed between younger and older (23-26 years old would be granted less points than 27-29 years old). If not backed up by strong evidence and acceptable explanation, this decision-making strategy would be very hard to be considered as logic and fair.

An example of logical trend is presented in the figure 18 below. In this case, the allocation of risk (and mechanically, of the "points" that will be attributed to each group) is having a linear and logical relationship between the attributes in Age and proportion of "Bads". This trend would much likely confirm the intuition and business experience of decision makers.

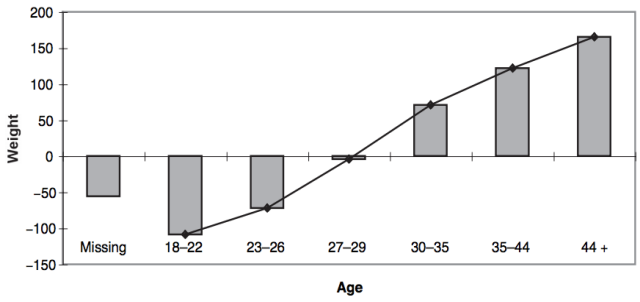


Figure 18: WoE and Logical Trends (Anderson, 2007)

In some cases, however, reversals may be reflecting actual behavior ("U" shaped pattern for example) and masking them may decrease the overall strength of the predictor. As already mentioned, this type of behavior should be investigated first, to see if there is a valid business explanation behind

Given the lack of business knowledge due to the external source of the data, the above process was purely based on a statistical approach, establishing relationships with the only objective of ensuring logical trend.

Due to obvious mathematical reasons, in some cases, reaching a logical trend might be mechanically impossible without drastically reducing the number of bins. This is why some variables had to be reduced to the minimum possible bucketing number of two. The full table with results of binning process for all the variable can be found in the annex nº1.

After the binning is performed, the next step to follow is to encode each bin with its associated WoE. This way the variable will shift from a categorical to a numerical type while conserving its linear trend (high WoE bins will be ordinally higher than lower ones).

Using the table nº 6 presented below for age, each data points of the variable will be encoded with the respective WoE of the bucket in which it falls into (-0.478 for a data point of 3 years old for example).

age	COUNT	WOE
[21, 35.5]	13771	-0,478
(35.5, 42.5]	12848	-0,271
(42.5, 50.5]	18739	-0,161
(50.5, 57.5]	15581	-0,021
(57.5, 63.5]	12369	0,369
(63.5, 103]	19032	0,898

Table 6 : Age Variable Final Encoding

This recoding of predictors is particularly well suited for subsequent modeling using Logistic Regression. By fitting a linear regression equation of predictors (encoded with WoE) to predict the logit-transformed binary Y variable, the predictors are all prepared and coded to the same (WoE) scale, and the parameters in the linear logistic regression equation can be fairly compared.

After fitting the logistic regression, the output of the algorithm can be scaled in many formats. In some cases (depending on implementation platforms, regulatory requirements or other factors) the scorecard has to be produced in a specific format. In this case, a scaling parameterization needs to be applied.

Scaling refers to the range and format of scores in a scorecard and the rate of change in odds for increases in score. The score is usually defined as the good/bad odd (e.g., score of 8 means a 8:1 odd -- equivalent to 8% chance of event or default) and it can have very specific shape defined for example by:

- the definition of a numerical minimum or maximum scale (e.g., 0–1000 or 300–800);
- the definition of a specific odds ratio at a given point (e.g., odds of 5:1 at 500);
- the definition of a specific rate of change of the odds (e.g., double every 50 points).

Since the choice of scaling does not impact the predictive power of the model, it is a decision that is only based on operational or regulatory considerations (Siddiqi, 2017).
The relationship between odds and scores can be presented as a linear transformation given by:

$$Score = Offset + Factor\, ln(odds)$$

4.2 XGBoost

Since 2015, a first to try, always winning Non-Parametric algorithm surged to the surface: XGBoost. This algorithm re-implements the tree boosting and gained popularity by winning Kaggle and other data science competition (Nielsen, 2016). XGBoost is able to solve real world scale problems using a minimal amount of resources (Chen & Guestrin, 2016).

In this section, we will introduce all the basic techniques that are implemented in the XGBoost that we are using in our work. We will take a try to understand the theoretical background needed for the understanding of the XGBoost, going through CART Decision Trees, Ensemble Models and boosting techniques.

4.2.1 CART - Decision Tree

Decision trees are another classification technique used for developing credit scoring models, also known as recursive partitioning (Hand & Henley, 1997), Additive Tree Model (ATMs) (Cui et al., 2015) or Classification and Regression Trees (CART).

These types of algorithm are important in the machine learning community since it have been around for decades and modern variations like Random Forest are among the most powerful techniques available.

Used within XGBoost (XGBoost Documentation), a CART decision tree is a simple model that tries to predict the value of a target event by inferring simple decision rules from the training data. In other words, a decision tree is a set of rules used to classify data into categories (Good vs Bad or default vs non-default).

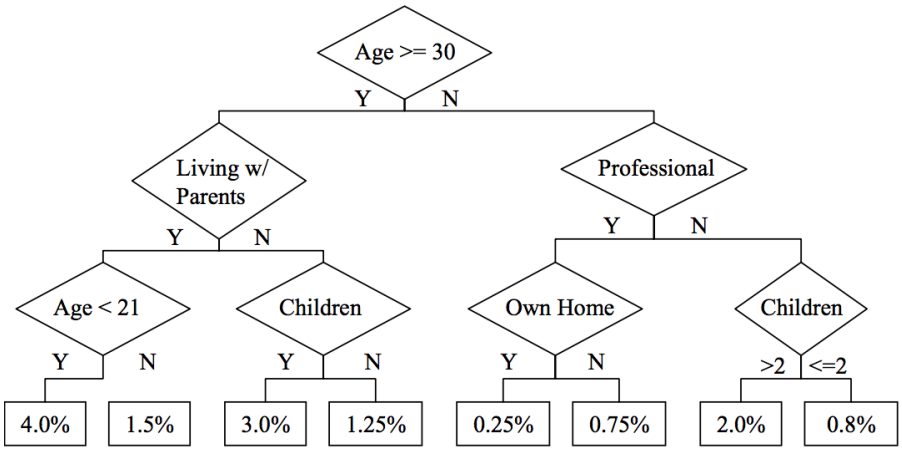


Figure 19: Decision Tree

The top of the tree as shown above contains the full training data and is referred as the *root node*. Each inferior stage of the tree level is referred as *child nodes*, and at the bottom, we have the *terminal nodes*. The depth of tree, (the number of levels it can build) are among the parameters that are defined by the user. If no limit was set, the algorithm would split the data until it can reach a terminal node that is completely pure (containing only "Bad" or only "Good"). At the end, the terminal node values can be used as estimates (scores), or as a grouping tool. In our case (binary outcome), the value given by the terminal node is a probability.

There are several possible splitting criterions in decision trees for building these rules. Entropy is one of them (used in XGBoost) and it is the one we will present here. Two kinds of entropy must be calculated at each branch level of the decision tree in order to perform the split:

1. **The Set entropy:** looking at the whole data set, it defines the number of individuals in each class and calculate p_i (percentage of total individuals in class i) and then the target entropy. It is defined by:

$$Entropy\ (Set)\ =\ -\sum_{i=1}^{k} P(value_i).log_2(P(value_i))$$

   where d is the number of classes (two in our case: safe and default).
2. **Each Feature entropy:** it is the same principle as for the set entropy but with a different method of calculation. For each feature N, we use its frequency table. The formula is defined by:

$$Entropy\ (Feature)\ =\sum_{i=1}^{n}(\frac{Nº\ of\ individuals\ in\ child\ node\ i}{Nº\ of\ total\ individuals\ on\ the\ parent\ node}).Entropy(child\ node\ i)$$

The information gain is then calculated for each splitting option of each feature. This metric measures the reduction in entropy that we get from the splitting of a given feature and splitting border. It is defined by:

$$Gain(Set, Feature)\ =\ Entropy\ (Set)\ -\ Entropy\ (Feature)$$

Thus, after assessing all the features, the one that maximizes the reduction in entropy (with the largest information gain) is the chosen one to perform the split. With new data, the algorithm will apply the previously defined set of rules to the new observations. At the end of the process, each instance will end up in a specific endpoint and assigned a predicted class.

In the literature and empirical evaluation studies, Decision trees are usually considered as a powerful analysis tools since it allows the discovering of feature interactions toward the

explanation of a specific event. However, it usually provides poor results and require an important amount of data in order to be significant (Anderson, 2007).

4.2.2 Ensemble Models

The ensemble method consists in the idea that a set of individually trained weak classifiers or learners (such as decision trees), performing weakly alone, can be combined to become better by reducing the prediction errors (Buja & Stuetzle, 2006). Previous research has shown that an ensemble is generally more accurate than any of the single classifiers in the ensemble, especially when it comes to decision trees (Opitz, 1999).

Two popular methods for creating accurate ensembles are Bagging (Breiman, 1996) and Boosting (Freund & Schapire, 1996). These methods usually rely on "resampling" techniques to obtain different training sets for each of the individual classifiers. Each tree is grown using CART methodology described earlier (Breiman et al., 1984).

Bagging (Breiman, 1996), also known as "bootstrap" (Efron & Tibshirani, 1993) ensemble method is based on the statistical concept of estimating quantities about a population by averaging estimates from multiple subsets of data samples. Breiman was the first to present empirical evidence that bagging can significantly reduce prediction error (or variance) of the algorithm. Applied to machine learning, the bagging consists in creating a random subset of instances and in a redistribution of the training set for each individual classifier that compose the ensemble.

In many ensemble approaches such as Random Forest, each tree is also built by selecting at random a small group of input coordinates (also called features or predictors) to be included in each individual learner. In the end, a chosen metric (e.g., Majority-Voting) is calculating the proportion of trees that classified one individual in each class and the class with the biggest importance defines the predicted class membership as shown in the illustration below (figure 20).

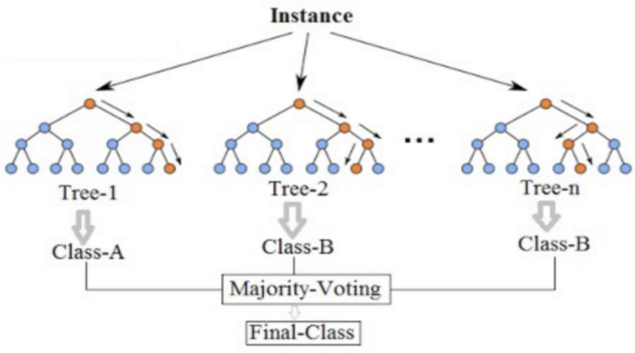


Figure 20: Ensemble of Decision Tree

As an ensemble decision tree model, XGBoost is based on the second ensemble prediction method mention earlier as boosting. Brought by Jérôme Friedman, this technique produces competitive, highly robust predictions for both regression and classification (Friedman, 2001)

The idea of the boosting methods is to produce individual classifiers sequentially and not randomly as it is done with bagging. To do so, the subset of training data used for each member of the ensemble is chosen based on the performance of the classifier(s) built previously during the construction of the ensemble. More specifically, examples that are incorrectly predicted by previous classifiers are chosen more often than examples that were correctly predicted. The problematic of ensemble modelling, with the perspective a boosting technique became about whether a weak learner can be modified to become better. Adaboost (Freund & Schapire, 1997), is usually considered to be the first successful example of a boosting algorithm.

In the end of the boosting process, there is M classifiers (M depending on the number of iterations defined by the user). After evaluating the weight of each of tree, based on their error rate, the final classification is made by combining the outputs of all classifier relatively to the weight associated to each one of them and to the chosen averaging method.

4.2.3 Gradient Boosting

Given the success of the Adaboost method, the statistics community developed a generalization of the boosting method applied to arbitrary loss functions: the Gradient Boosting (Friedman, 2001 - Sigrist, 2018).

This method is based on the concept of the gradient descent algorithm which is a first-order iterative optimization algorithm for finding the minimum of a function. This algorithm initiates the optimization of a formula $f(x)$ with a random value for x, and performs several iterations using the formula below to update the value.

$$x_{current\ iteration} = x_{previous\ iteration} - \eta \frac{\delta f\ (x^{previous\ iteration})}{\delta\ x^{previous\ iteration}}$$

With η being the magnitude step that will determine the size of the following one. The term that follows is the gradient that gives the direction to be taken for minimizing the loss function.

After initializing a model with a constant value, at each iteration the algorithm computes the pseudo residuals (gradient of the error with respect to the loss predictions of the previous model) according to the following formula:

$$r_{im} = -\left[\frac{\delta\, L\,(Y_i, (x_i)}{\delta\, F(x_i)}\right], for\ i = 1, 2, \ldots, n$$

Then, it trains the following tree using this pseudo residuals as a new target to predict. Doing so, each new tree specializes into predicting well the previous instances that were wrongly predicted by trying to minimize the loss function.

In the case of the XGBoost, it actually uses a minimization technique derived from the gradient boosting method known as Newton boosting (Nielsen, 2016 - Chen & Guestrin, 2016 - Sigrist, 2018). As we saw, the Gradient boosting is based on a first-order gradient descent updates while the other is based on a second-order gradient descent update. Recent research tends to show that Newton boosting performs significantly better than the other boosting variants for classification (Sigrist, 2018) as it gives a more accurate view of the direction to take for minimizing the loss function.

5 Hyper-Parameters Optimization

As it is done in James and in many Credit Risk related machine learning task, our study will also include the careful tuning of learning parameters and model hyperparameters. hyperparameters usually refers to model properties that cannot be directly learned from the regular training process. It can be the complexity of the model or how fast it should learn, which are usually fixed before the training or the boosting optimization. The following section aims at giving a brief introduction to hyperparameters optimization problematic and one of its solution, used within James software: Gaussian Process or Bayesian optimization.

5.1 Bayesian optimization

Bayesian optimization has proved to be a good choice in many contexts by yielding to better performance when compared with other state of the art optimization techniques. In theory, it works by assuming that the function to optimize follow a Gaussian process[9] and keep a posterior distribution of this function while results with different hyperparameters are observed (Mockus et al. 2014).

As the number of iterations grows, the posterior distribution of the loss function becomes more accurate and it becomes more clear which regions of the parameter space are should be explored further and which one should not, as shown in the figure 21 below.

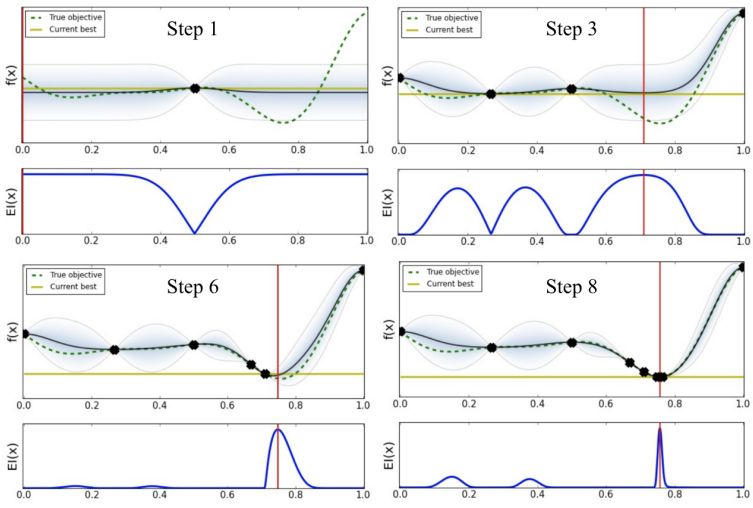


Figure 21: Illustration of a Bayesian optimization Process

[9] Collection of random variables indexed by time or space, such that every finite collection of those random variables has a multivariate normal distribution

At each step of the process, a Gaussian distribution is contoured to the known samples (points previously explored), and the posterior distribution determines the next point that should be explored.

5.2 Practical Application

For the purpose of optimizing the parameters of our algorithms as it is done in James, we used Scikit-Optimize (or Skopt), which is a simple and efficient library to minimize (very) expensive and noisy black-box functions. The optimization was performed toward the maximization of the Roc Auc[10] metric.

In our study, the following list of parameters will be optimized for the XGBoost:

Hyper Parameter	Definition	Ranges of Optimization
Learning rate	Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.	[0.01; 0.5]
max depth	Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 indicates no limit. Note that limit is required when grow policy is set of depth wise.	[2; 10]
minimum child weight	Minimum sum of instance weight needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than *min_child_weight*, then the building process will give up further partitioning.	[1; 7]
lambda	L2 regularization term on weights. Increasing this value will make model more conservative.	[0.0003; 100]
subsample	Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration.	[0.6; 1]
scale_pos_weight	Control the balance of positive and negative weights, useful for unbalanced classes. A typical value to consider: sum(negative instances) / sum(positive instances).	[1; 10]
colsample_bytree	Subsample ratio of columns when constructing each tree. Subsampling will occur once in every boosting iteration.	[0.05;1]

Table 7 : XGBoost Hyper Parameters

[10] Refer to “Performance Evaluation” section of this document

Within James, the following list of parameters will be optimized for the Scorecard:

Hyper Parameter	Definition	Ranges of Optimization
penalty	Used to specify the norm used in the penalization.	['l1', 'l2']
C	Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.	[0.001; 10]
class weight	Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one.	[null, 'balanced']
fit intercept	Specifies if a constant (a.k.a. bias or intercept) should be added to the decision function.	[true, false]
intercept scaling	Useful only when the solver 'liblinear' is used and self.fit_intercept is set to True. In this case, x becomes [x, self.intercept_scaling], i.e. a "synthetic" feature with constant value equal to intercept_scaling is appended to the instance vector.	[0.01; 10]

Table 8 : Scorecard Hyper Parameters

As both models are completely different by nature (one is linear and the other is based on decision tree boosting), there is no practical way to ensure that the parameters selections allowed for a fair comparison or not. In James, the choice of parameters was based on practical use cases and feedback from clients while the XGBoost parameters were selected according to the modelers experience and judgment.

6 Models Evaluation

The performance of each model is assessed in the Results chapter through the analysis of two indicators that aim at measuring the relative quality of each technique. In this section, we will focus our attention on these two indicators: The Receiver Operating Characteristic Curve and the Log-Loss statistic.

The Receiver Operating Characteristic metric, one of the most widely used evaluation method for predictive modeling (Figini & Maggi, 2014) will inform us on how well the model ranks the examples while the Log Loss is based on an understanding of error as it is measuring the deviation from the true probability (Ferri et al., 2009).

6.1 Receiver Operating Characteristic Curve

The Receiver Operating Characteristics (ROC) curve, or sometimes called "Lorentz diagram" represents the true positive rate or "hit rate" (proportion of bad cases predicted as bad - plotted on the vertical axis) against the proportion of false positive rate or "false alarm rate" (good cases predicted as bad - plotted on the horizontal axis) at all cut-off score values (Satchell & Xia, 2007). Once the plot is done, we can easily compare the performance of our model with the one of a hypothesis random model ("coin toss" model) or a perfect one. The graph below is an example of this representation.

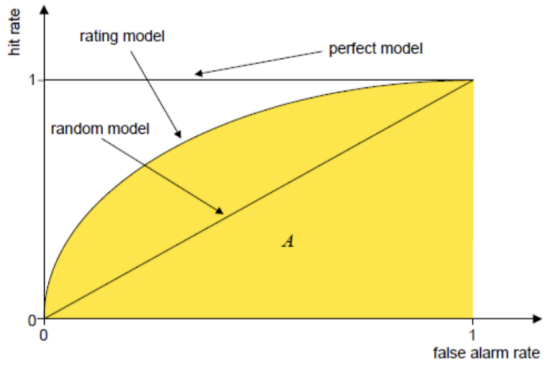


Figure 22: Receiver Operating Characteristic Curve

More than just a visual representation, the ROC curve allows the calculation of two state of the art coefficients that are the Auc (Area Under the Curve) and the GINI (originally a measurement of income inequality frequently used by economists (Macedo, 2013) .

The AUC indicator ranges from 0 to 1 and is basically represented in the figure 22 above as *A* (area highlighted in yellow). An AUC of "1" would be the expression of a perfect model while "0.5" would be the equivalent of randomness. The GINI index, also used very frequently,

is directly calculated through the AUC value according to the formula: $GINI = 2 \times Auc - 1$ (Paulo Macedo, 2013).

In our study, these metrics are particularly well suited since they give one number that summarizes the performance of the model over all cut-off scores and allow therefore to compare the models at a global level. Furthermore, the ROC (AUC) is known to be a representation of choice when the costs assigned to false positives and false negatives are not known at the time of training (Bach et al., 2006), as it is the case in our experiment.

6.2 Log-Loss (Logarithmic Loss)

Logarithmic Loss, or simply Log-Loss, is a classification loss function often used as an evaluation metric in predictive modeling tasks. It quantifies the accuracy of a classifier by penalizing false classifications, giving more weight to error for which the classifier is confident about an incorrect classification.

For binary classification with a true label $y \in \{0,1\}$ and a probability estimate $p = \Pr(y = 1)$, the log loss per sample is the negative log-likelihood of the classifier given following formula:

$$\text{Logarithmic Loss} = -\frac{1}{N}\sum_{i=1}^{N} [(y_i log(p_i) + (1 - y_i)\log(1 - p_i))]$$

where p_i is the probability that the i^{th} data point belongs to class 1 (default in our case), as judged by the classifier. y_i is the true label and is either 0 or 1.

The plot below shows the Log Loss contribution from a single positive instance where the predicted probability ranges from 0 (the completely wrong prediction) to 1 (the correct prediction).

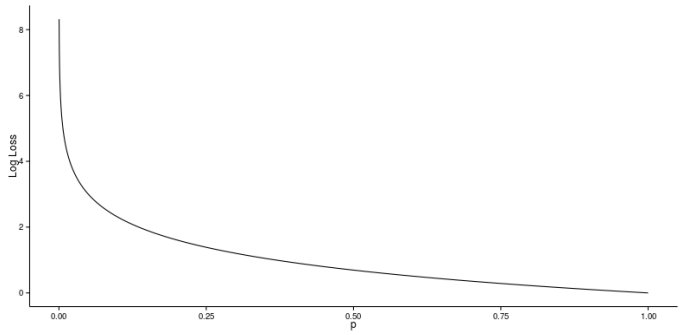


Figure 23: Logarithmic Loss

As one can see, the closer the probability get from the actual event (default = 1) the lower the Log-Loss. As such, a model that would perfectly capture the event we're trying to predict would have a Log-Loss of 0 (zero).

7 XGBoost Interpretation Techniques

7.1 Partial Dependence Plots (PDP)

It can be difficult to understand the functional relations between predictors and an outcome when using Black Box prediction methods as ensemble or boosting. One particularly effective way (Greenwell et al., 2018) to explain the output from Black Box models is with partial dependence plots (PDP plots) (Friedman, 2001).

The basic idea of PDP plots is to visualize the change in the average predicted value (probability of default in our case) as given feature(s) vary over their original distribution (Goldstein et al., 2015). To understand PDP plots, we will consider a model trained on a dataset D. This dataset has Nobservations y_k with $k = 1,2,\ldots,N$, along with p predictors $i = 1,2,\ldots,p$ denoted $x_{i,k}$. The model will produce predictions represented by the following function:

$$\widehat{y_k} = F(x_{1,k}, x_{2,k}, \ldots, x_{p,k})$$

for some mathematical function $F(\ldots)$.

Partial dependence plots of a predictor x_1 is produced by averaging the prediction of each $x_{1,k}$ for all possible $x_{i\neq 1,k}$ or $x_j$and plotting it over a useful range of x values:

$$\phi_j(x) = \frac{1}{N}\sum_{k=1}^{N} F(x_{1,k}, \ldots, x_{j-1,k}, x, x_{j+1,k}, \ldots, x_{p,k})$$

The function $\phi_j(x)$ tells us how the value of the variable $x_j$influences the model predictions $\widehat{y_k}$ after we have "averaged out" the influence of all *other* variables as shown in the figure 24 below.

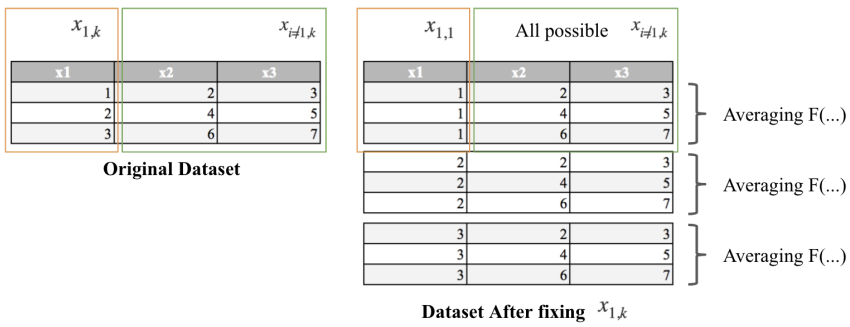


Figure 24: PDP Calculation - Simplified Representation

For linear regression models, the resulting plots would be a simple straight line whose slopes is equal to the model parameters for each predictor. For more complex algorithms, a partial dependence plot can show if the relationship between the target and a feature is linear, monotonic or more complex.

In the example below from a research paper, authors (Hastie et al., 2017) are showing the results of a partial dependence plots (presented in figure 25) on an open source dataset (the California housing dataset). They used a Gradient Boosting Regressor trying to predict the median house prices of different localities.

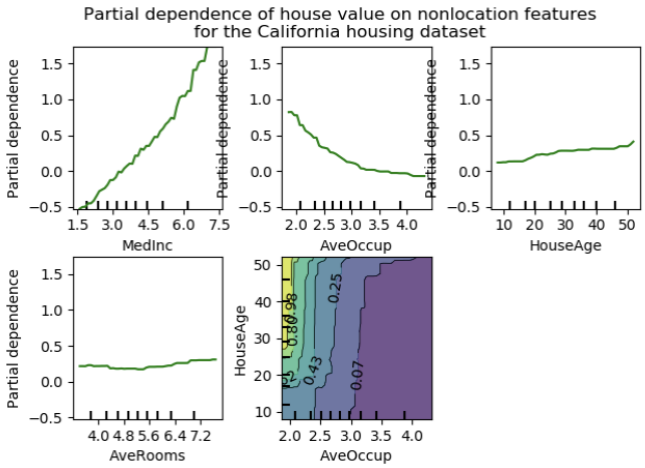


Figure 25: PDP California Housing Dataset example

From their work, we observe four one-way and one two-way Partial Dependence Plots. The predictor variables for the one-way PDP are *median income* (MedInc), *avg. occupants per household* (AvgOccup), *median house age* (HouseAge), and *avg. rooms per household* (AveRooms). In their work they make the following analysis:

> "*We can clearly see that the median house price has a "linear" relationship with the median income (top left) and that the house price drops when the avg. occupants per household increases (top middle). The top right plot shows that the house age in a district does not have a strong influence on the (median) house price; so does the average rooms per household. The tick marks on the x-axis represent the deciles of the feature values in the training data. Regarding the two-way Partial Dependence Plot, we can see that for an avg. occupancy greater than two, the house price is nearly independent of the house age, whereas for values less than two there is a strong dependence on age.*" (Hastie et al., 2017)

While Partial Dependence Plots are easy to implement and to interpret, they are not perfect and should be used under specific circumstances. Since it averages the predictions of other features than the one we are studying, it makes a strong assumption about the independence of the studied variable. In case of highly correlated or strong interactions between features, the output of the PDP might be biased and lead to wrong causal interpretations (Molnar, 2018).

Another known limitation of the PDP is that since it averages the effect of a change in a variable over the outcome, the final result is a generalization that might come to be untrue for some part of the population.

7.2 Shap (**SH**apley **A**dditive ex**P**lanations) Values

SHAP, based on Shapley value, is defined as a unified approach to explain the output of any machine learning model. It represents *"the only possible consistent and locally accurate additive feature attribution method based on expectations*" according to their authors (Lundberg, 2019).

Shapley values were introduced in game theory but they were used in the context of predictive modeling only very recently. Shapley value was initially used to determine how much each player of a collaborative game had contributed to its output. In our case, each SHAP value measures how much each predictor has contributed (positively or negatively) to an individual predicted risk of default. It is part of a family of interpretation methods that is known as "*additive feature attribution methods*" (Lundberg & Lee, 2017a).

As stated in the original SHAP paper, the idea is to retrain a given model on all possible ordered subsets of features (predictors) $S \subseteq F$, where F is the set of all features. Then, it assigns an importance value to each feature based on the average impact of including (or not) that feature on the model prediction (given the subset S). The model performance without the given feature is compared with a model including that feature (i.e calculating the difference of both predictions) for all combination of subsets, as represented in the figure 26 below.

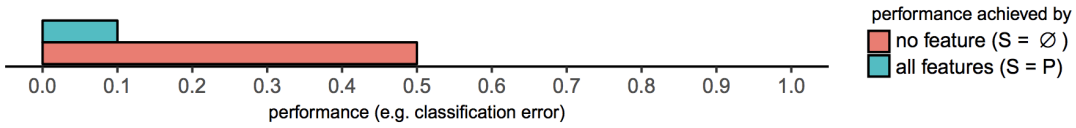


Figure 26: "Illustration of the difference in model performance that we want to fairly distribute among the features. The model performance (e.g., classification error) is 0.1 when using all features (green bar) and 0.5 when ignoring all features (red bar). Our goal is to fairly distribute the resulting performance difference of 0.4 among all involved features based on their marginal contribution". (Casalicchio et al, 2018)

Lundberg (Lundberg & Lee, 2017b) states that the impact of withholding a feature may depends on:

1) other features in the model and;

2) the order in which features are introduced.

To account for the two points above, the differences in prediction are computed for all possible ordering of all possible subset of features S. The Shapley values are then calculated as a weighted average of all the computed differences and are used as feature attributions. This mechanism can be defined by the equation

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}\left(x_{S \cup \{i\}}\right) - f_S(x_S) \right]$$

where $S \subseteq F\backslash\{i\}$ are all the possible subsets of F, $f_{S\cup\{i\}}$is the model including the feature i for which we want to measure the importance and $f_S(x_S)$ is the model using a subset of features ("excluding" i).

While conceptually straightforward, the equation above may require a colossal amount of computing power or time according to the dimension of the data. thus, the exact calculation of the SHAP value may not be feasible in some cases (Molnar 2018). An approximation function of the SHAP value, only working with decision tree models, was proposed by Lundberg in 2018 (Lundberg & Lee, 2017c). The implementation of this approximation consists in recursively keeping track of what proportion of all possible subsets flew down the trees during the construction of the ensemble. This "subsetting memory" ability is currently supported by the XGBoost and LightGBM[11] packages. Thanks to that, the calculation of the SHAP value can be estimated much faster and with significant degree of accuracy (Molnar 2018).

Although traditional feature importance exists, those will only tell us which features are most important across the entire population and this global approach might not be representative of each individual specificity across a population. A factor that is an important driver for one customer may be a non-factor for another.

By looking only at the global trends, these individual variations can get lost, with only the most common denominators remaining. With individual-level SHAP values, we can pinpoint which factors are most impactful for each customer as illustrated in the figure 27.

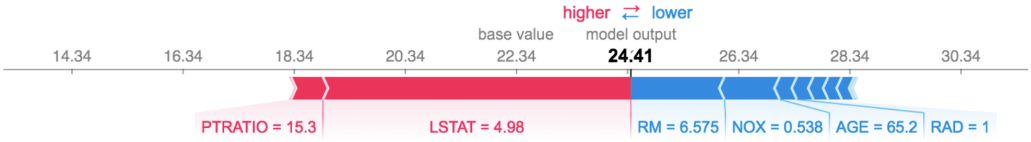


Figure 27: Shap Individual Feature Contribution

The above representation, taken from the Shap Python GitHub package, shows each feature contribution toward the final prediction of a specific application. The base value represents the average Probability of Default from the training set. Features pushing the prediction higher than the base value are shown in red while those pushing the prediction lower are in blue.

[11] Light Gradient Boosting Machine

8 Results and Discussion

8.1 Statistical Results

As defined earlier, the statistical performance of each model was assessed using the GINI /AUC metric and the Log Loss which are summarized for both model in the table 9 below:

Model	Test Set			Train Set	
	GINI	AUC	Log Loss	GINI	AUC
Scorecard	60,59	80,3	0,5114	60,93	80,46
XGBoost	62,91	81,45	0,1876	64,9	82,45

Table 9 : Statistical Evaluation

The XGBoost model is showing the best results both in term of GINI and log loss. On the test set, it has a GINI of 62,91 (≃2 GINI points above the Scorecard model).

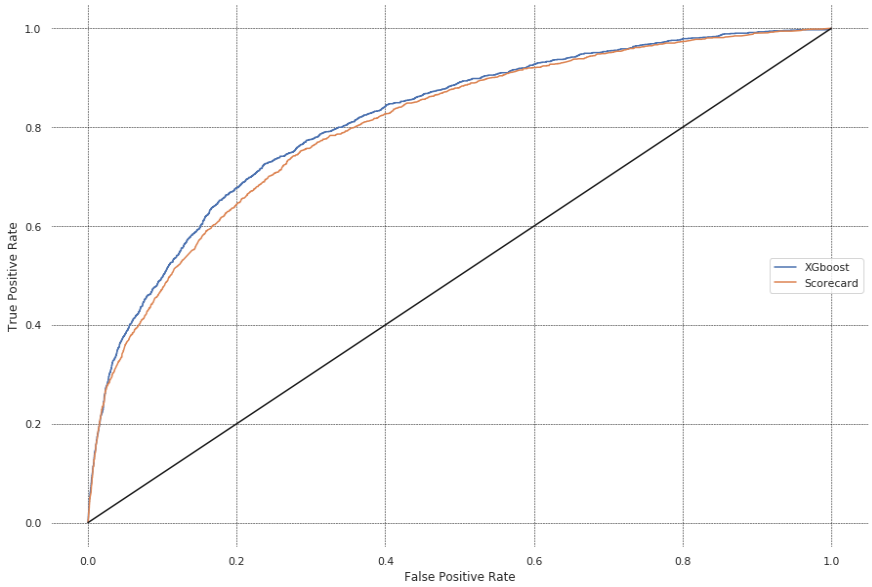


Figure 28: Test Sets Roc Auc Curves

As one can observe on the Roc Auc Curves above (figure 28), the 2 models are equally performant in identifying very good clients (bottom left part of the curve). On the rest of the population, the XGBoost algorithm is showing more performance.

If we take a look at the differences of GINI for each model between the test and the train set (illustration at figure 29), one can observe an increase in performance when applying the model on the train set.

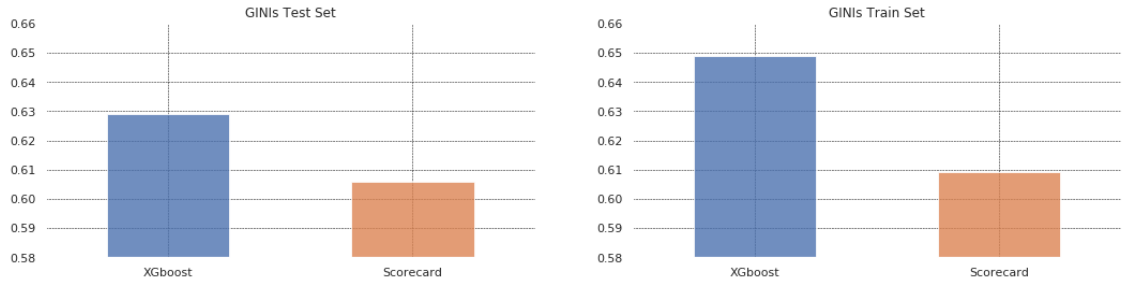


Figure 29: GINIs Train Vs Test Set

The increase observed is higher with the XGBoost since it represents two points of GINI while the scorecard's performance is only increasing of 0.34 points of GINI.
This behavior of the XGBoost between the train and the test set might be caused by overfitting and further investigations could be made in order to confirm this eventuality.

The outperforming results of the XGBoost on the GINI are reinforced by its Log Loss of 0.18 against 0.52 for the Scorecard, as represented below (figure 30). The higher GINI (ability to rank each individual) of the XGBoost might be partly explained by the higher accuracy of the predictions (distance to the real outcome - 1 or 0) it makes.

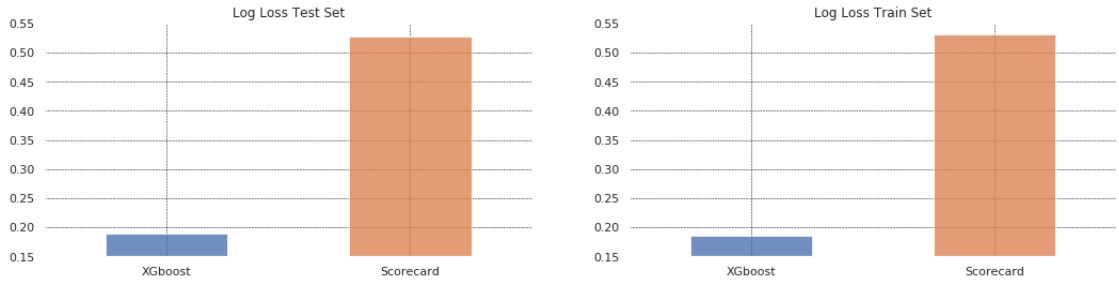


Figure 30: Log-Loss Train Vs Test Set

Given the results above, it may be hard to get a grasp of how each model would impact the decision making at portfolio level. To give more color to the fact of having two more GINI points, we conducted a very basic scenario simulation that could give an idea of how each model would allow to balance risk taking strategies. The experiment was done using the test set (simulating a new batch of applications) and had the following rationale:

First, we selected tree different cutoff probabilities based on several acceptance rate objective for all models. Applicants with a probability above this cutoff would be considered as not trustworthy and would not be granted a loan. The acceptance rate we'll be looking at are 90%, 50% and 25%.

Secondly, for each model, the level of risk in the remaining accepted population (those to whom we would have been granting loans) was assessed using the actual default rate within this population. Results are summarized in the following set of graphs (figure 31).

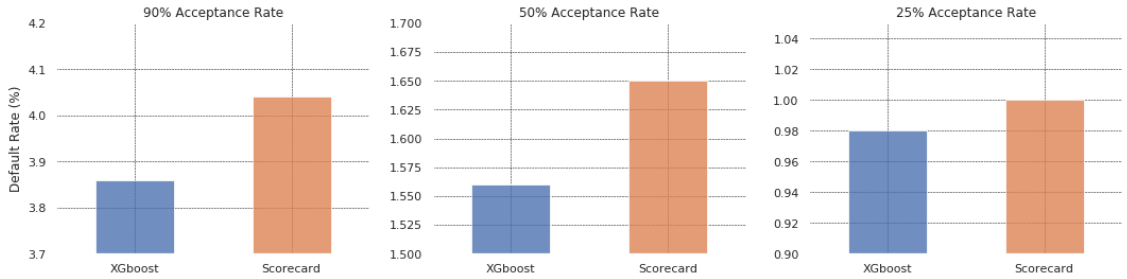


Figure 31: Scenario Simulation

As one can see, each model is performant in the sense that decreasing the acceptance rate would lower the default rate of the accepted population.

We can pinpoint however that in all the simulation tested here, the default rate of the population accepted by the scorecard was higher. In the left graph for example, simulating a 90% acceptance rate strategy, the Scorecard would lead to more than 4% (839 bad loans) default rate while being 3.84% (798) using the XGBoost. Extrapolating these results, we could generally expect 5% more default when using the Scorecard.

To sum up, we can conclude, given the information that we have above, that the XGBoost model is more performant at distinguishing potential goods from potential bad clients. However, it is important to emphasize that given that the better performance of one model over another is no the main scope of our study, we made the choice to generalize the data cleansing process for both models and to limit the statistical assessment. A custom data preparation process and a more exhaustive statistical or scenario simulation evaluation may yield to different results or conclusion about the pure statistical performance of our algorithms.

8.2 Global Interpretation

Scorecard

When analyzing the output of the Scorecard, each coefficient (Beta) can be interpreted as the expected change in the log-odds of being a default for one unit increase in the corresponding predictor variable (holding all the other predictor constant at certain value). Equally said, each exponentiated coefficient (Odd Ratios in the table above) correspond to the change in odds one unit increase in the corresponding predictor variable. Knowing the later, we analyzed each coefficient and compiled the results in the table 10 presented below.

Variable	Beta	Odd Ratios	Expected % change in odds
Number of Open Credit Lines and Loans	**1.1329**	3.1046	210.46%
Number of Times 90 Days Late	**-0.8668**	0.4203	-57.97%
Revolving Utilization of	**-0.782**	0.4575	-54.25%

Unsecured Lines			
Debt Ratio	**-0.7446**	0.4749	-52.51%
Age	**-0.5482**	0.578	-42.20%
Monthly Income	**-0.4568**	0.6333	-36.67%
Number of Dependents	**-0.3239**	0.7233	-27.67%
INTERCEPT	**0.00035**	**1.00035**	**0.04%**

Table 10 : Scorecard Coefficients

For *Number of Open Credit Lines and Loans***,** (holding everything else equal) we can expect a 210% increase in the odds of being a default case for a one-unit increase in the transformed (WoE) variable. Given that the WoE range (Difference between the maximum and the minimum) is 0.25 and that it has only two buckets, we know that above 6 Open Credit lines and Loan, the odds of being Bad will be around 52% higher. Thus, we can conclude that a higher number of credit lines will lead to a higher risk from the model's point of view.

For *Debt Ratio*, (holding everything else equal) we can expect a 53% decrease in the odds of being a default case for a one-unit increase in the transformed (WoE) variable. The range of the WoE being 0.85, the odds of being bad are around 45% higher with a *Debt Ratio* of one than with 0 (zero) since a high WoE means a lower *Debt Ratio*, we can conclude that having a high *Debt Ratio* will lead to higher risk from the model's point of view.

Based on the results and by following the same rationale as the two variables above, it is easy to construct a table such as the table 11 following:

Variable	Expected % change in odds	Human Interpretation
Number of Open Credit Lines and Loans	210,46%	A higher number of credit lines will lead to a higher risk
Number of Times 90 Days Late	-57,97%	A high debt ratio will lead to higher risk
Revolving Utilization of Unsecured Lines	-54,25%	A higher number of 90 days late will lead to a higher risk
Debt Ratio	-52,51%	A higher Revolving Utilization of Unsecured Lines will lead to a higher risk
Age	-42,20%	A higher Monthly Income will lead to a lower risk
Monthly Income	-36,67%	A higher Age will lead to a lower risk
Number of Dependents	-27,67%	A higher Number of Dependents will lead to a higher risk

Table 11 : Scorecard Interpretation Overview

XGBoost

As a first global understanding of the XGBoost model, we will use one of the function embedded within the XGBoost python package. After fitting the algorithm to the training data, we can call a simple "*plot_importance*" extension from the trained model.

Calling this method will lead to the visualization of 3 metrics (figure 32) commonly referenced as "**weight**", "**gain**", or "**cover**". Each one of them is defined as follow:

- "**weight**" is the number of times a feature appears in a tree
- "**gain**" is the average Information Gain of splits which use the feature
- "**cover**" is the average number of observations covered when splitting with the feature

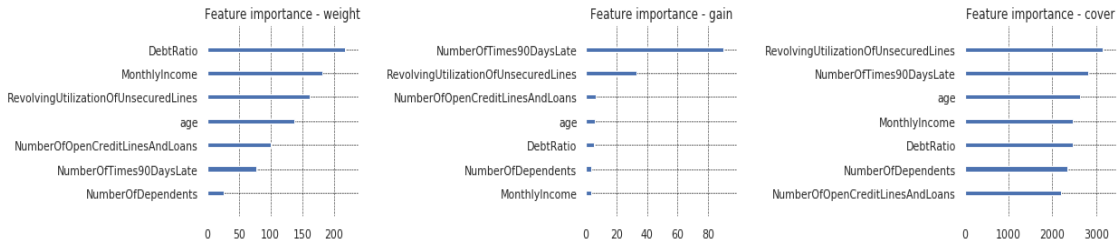


Figure 32: Classical Feature Importance – XGBoost

By Looking at the results, we see that *Debt Ratio* is used a lot of time (relatively to other) in the tree. it however seemed to be leading to low Gain for a low amount of observation. We can assume that *Debt Ratio* was used a lot at very deep level of the tree (when the population already had been split several times using other variables).

As an extreme opposite of *Debt Ratio* variable*, Number of Time 90 Days Late* was used as a splitting variable a relatively low amount of time, but it led to the highest average amelioration in term of Information Gain. *Number of Time 90 Days Late* appears to have a very strong entropy power and it seemed it was used as a (close to) root split node a significant amount of time, impacting a significant portion of the population (second position for "cover" importance). In between the two cases described above, it is interesting to node the stable position of the *Age* variable for the tree ranking metrics, showing the stable importance of the variable along the construction of the ensemble.

With those metrics only, it is hard to take a definitive conclusion toward the importance of a variable. All the above analysis was based on assumption and should not be taken as ground truth regarding the functioning of the model. As a simple example, those metrics don't show the direction of the interaction between the risk and the variable nor the interactions that might exists between features. Indeed, a few splits with a given features (that we might consider non-significance) may be actually leading major Information Gain along the path created by the variable's split.

By using the Shap Value defined in an earlier section of this paper, we are able to take these interactions into account for generating a potentially more accurate ranking (by testing all the possible combination of features) presented in the figure 33. Although the interactions that might be happening are not visible, it should give a more generic view on what variable contributed the most to the final output of each prediction.

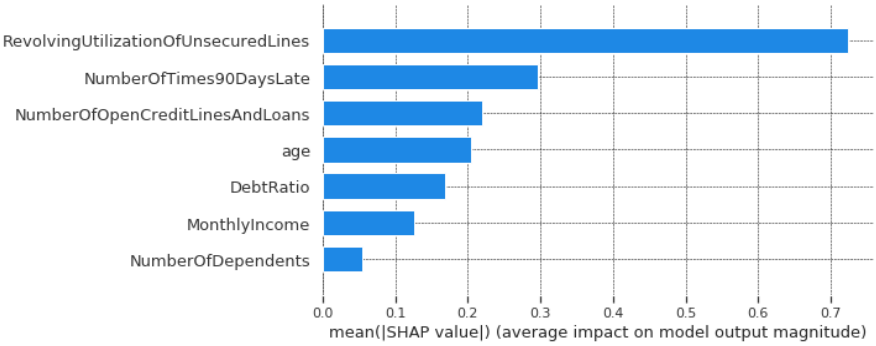


Figure 33: Shap Feature Importance

Regarding the Shap absolute importance, we will simply make two remarks:

- It is interesting to note the *Age* variable stayed in the middle position (as it was in middle position for all previous methods)
- Top variables (namely, *Revolving Utilization of Unsecured Lines* and *Number of Times 90 Days Late*) from previous "Gain" and "Cover" methods are still in the top using the Shap method. Information Gain and the average number of observations included in a features' splits seems to be significantly impacting the final prediction.

Now that we have a better view on which variable had the most significant impact on the model prediction, we still miss a major piece of comprehension that we should be able to get using Partial Dependence Plots defined earlier. By plotting the PDP for each variable of our training set, we obtained the results presented in the figure 34 below. For the sake of proper visualization, the entire range of some variables was reduced for the following representation. The complete PDP representation of *Monthly Income*, *Debt Ratio* and Number of Time 90 Days Late variables can be found in annex nº4.

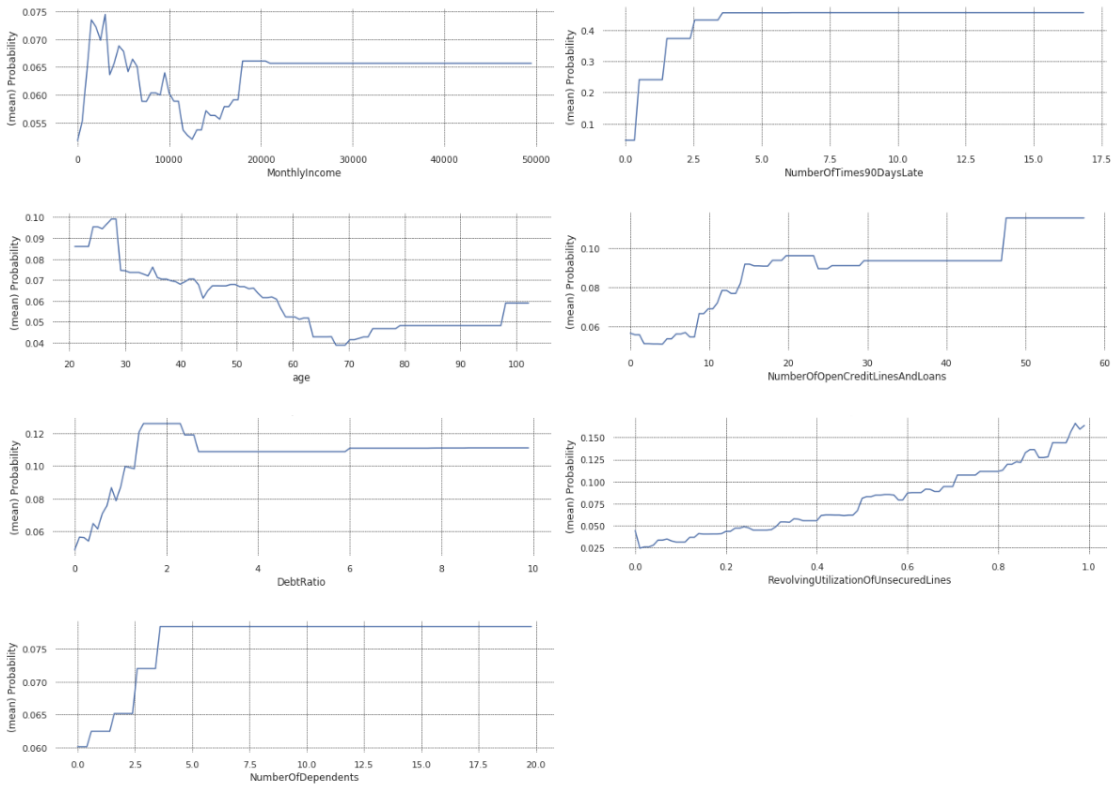


Figure 34: Partial Dependence Plots

Starting with the most "intuitive" cases from the PDPs above, we see that in average, the model predicts more risk when:

- **N***umber of Dependent* is high,
- *Number of Time 90 Days Late* is high;
- *Number of Open Credit Lines and Loans* is high;
- *Revolving Utilization of Unsecured Lines* is high**;**
- *Debt Ratio* is high**.**

When looking the scale of the respective averaged change in risk (y-axis) for all variables, we see that *Revolving Utilization of Unsecured Lines* has the most significant range, followed closely by *Number of Time 90 Days Late* and *Number of Open Credit Lines and Loans* variables (confirming more or less the ranking of the Shap method given earlier).

For *Number of Dependent* variable, we can point out the fact that above a value of 5, the algorithm is, in average, considering all instances with the same level of risk. As we were expecting, the algorithm was robust to the low number of potential outliers (11 observation above 8) since it doesn't seem to be taking very specific decisions for these cases.

For *Debt Ratio* variable, although the global average trend of the algorithm is to predict more risk when the value increases, we see that above 2, there is a small decrease before to get to a "plateau" situation until 10. Although it might not be statistically very relevant given the low

scale of the decrease, it is a very representative situation that would be hard to justify using common sense and business knowledge.

For *Monthly Income* and *Age*, we see that globally, the XGBoost is predicting lower level of risk as the variable increase in value. However, the trend reverses at 10200 for *Monthly Income* and 70 for *Age*. The reversal is much more significant for the *Monthly Income* variable. The scale of the respective averaged change in risk (y-axis) follows once again the ranking of the two features given earlier by the Shap importance ranking.

Key Learnings of Global models respective Interpretations

Conceptually much more straightforward, the Scorecard allowed for a fast and measurable understanding of each feature's impact on the model, in a way that can be understood by any human and that could be easily automated. On the other hand, we saw that the global analysis of the XGBoost required the understanding of several complex metrics that proved to be hardly measurable in a straightforward way (for production-ready purposes). Due to this, the resulting "natural language" that could be extracted from the analysis, while partially allowing for contrastivity, is still reflecting the uncertainty of the interpretation method.

8.3 Local Interpretations

For the framing of investigating local explanation methods, we selected tree instances from our dataset based on their predicted risk of default. One observation with a low risk, one with a medium risk and the last one with a high predicted risk (for both algorithms in all three cases). The characteristics of each one of these applicants is presented in the table 12 below.

Risk	Age	Number of Times 90 Days Late	Number of Open Credit Lines and Loans	Number of Dependents	Monthly Income	Revolving Utilization of Unsecured Lines	Debt Ratio
Low	54	0	18	0	18750	7.55%	3.,44%
High	40	1	1	2	4544	100%	3.63%

Table 12 : Low and High-Risk Application Characteristics

For each observation, we will try to get an understanding of what variable contributed to lowering or increasing the predicted risk of default for both algorithms. The prediction of each observation is as shown in the following table (table 13):

Risk	Scorecard Score	XGBoost PD
Low	521	2.8%
High	418	42.4%

Table 13 : Scores Low and High-Risk Application

For the Scorecard model, we will use the transformed log-odds (score contributions) associated with each characteristics of the client. As such, we believe it is more human friendly for intuitively assessing the contribution of each characteristics of an individual. When transformed, a higher score is related to having a lower risk of default. The Scorecard table is as follow (table 14):

Variable	Bucket	Partial Score
	0 - 3436	65
	3436 - 5330	68
	5330 - 6666	70
	6666 - 9831	73
Monthly Income	9831 - 3008750	76
	100.01% - 1000%	56
	50.41% - 100%	61
	33.81% - 50.4%	70
Debt Ratio	0% - 33.8%	74
	2 - 98	-1
	1	21
Number of Times 90 Days Late	0	78
	3 - 20	67
	2	68
	1	69
Number of Dependents	0	71
	21 - 35	62
	36 - 42	65
	43 - 50	67
	51 - 57	69
	58 - 63	75
Age	64 - 103	84
	7 - 58	66
Number of Open Credit Lines and Loans	0 - 6	74
	49.51% - 100%	48
	18.61% - 49.5%	75
	5.55% - 18.6%	91
Revolving Utilization of Unsecured Lines	0% - 5.54%	97

Table 14 : Final Scorecard Table

For interpreting the Scorecard algorithm, we will simply need to add up each partial score of an individual based on which bucket it belongs to for each variable. For the XGBoost, we'll use the Shap values introduced in the previous chapter and the contributions will be expressed as Probabilities of Default relative to the average model output (base value) of 0.0562 (5.62% risk of default) represented as a red vertical line in the figure 35 below.

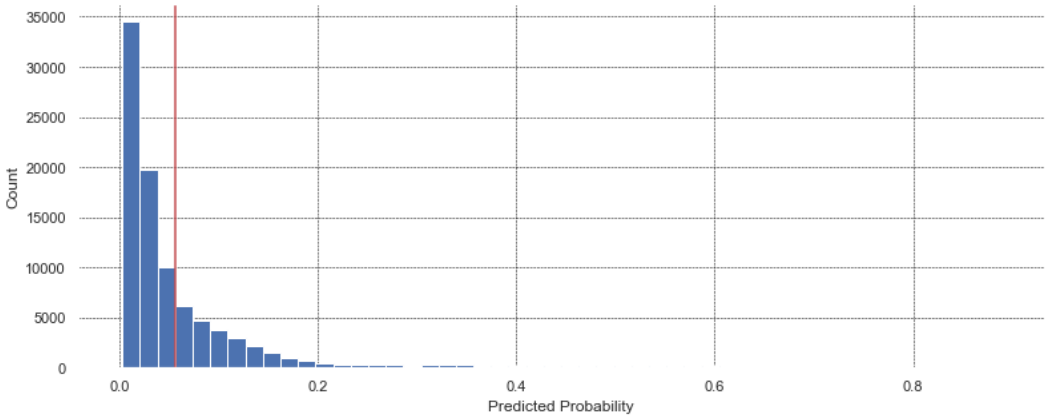


Figure 35: XGBoost - Distribution of Predicted Probability & Shap Base Value

The illustrations below (figure 36) represents the contributions of each variable for the predicted risk of the High-Risk observation according to both algorithms. The data used to generate these plots can be find in annex nº 5

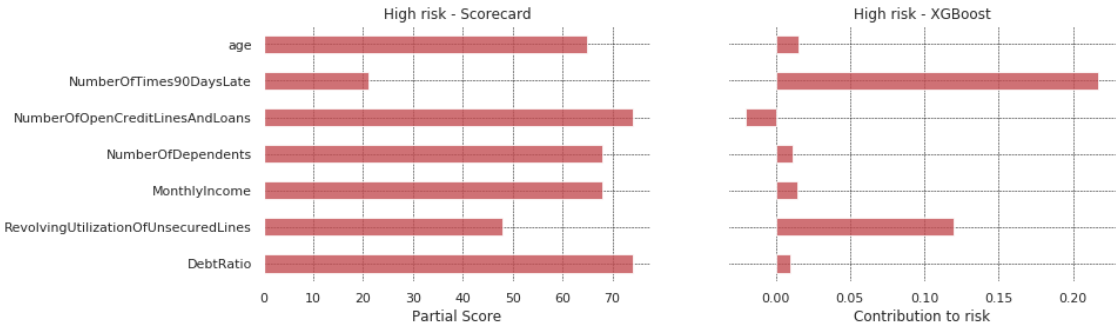


Figure 36: High-Risk individual Explanation

We can see that both models based their prediction of a high risk mostly due to two variables, *Number of Times 90 Days Late* (value of 1) and *Revolving Utilization of Unsecured Lines* (value of 100%).

The fact of having a 90 Days Late past experience is only adding 21 point to the final Scorecard score, significantly less than the rest of the characteristics. For the XGBoost, the same variable pushed the probability of default up by 21% from the average base value. From the Partial Dependence Plot presented earlier, we could potentially observe a lower negative impact of that contribution if the applicant had no 90 Days past due in the past. With the Scorecard, the partial

score of that variable could have topped up to 78 (pushing the application to the "low" risk edge of the full score distribution - around 500).

With a *Revolving Utilization of Unsecured Lines* of 100%, this application falls in the lowest partial score bucket of the Scorecard for that variable (partial score of 48). During a loan application process, we could imagine the lender to recommend the potential client to work on improving the situation for being considered as less risky. By looking at the XGBoost result, we see that the algorithm is also associating a significant part of the risk due to that same variable. It actually contributes of almost 12% increase in risk relative to the base value. By looking at the PDP, we could also expect a decrease in the risk of this application if the value was lower (through a less negative contribution of *Revolving Utilization of Unsecured Lines* variable).

Other than that, we can observe that the *Monthly Income* felt in one of the lowest bucket of partial Score possible for the Scorecard. It could then be a driver of improvement, but the maximum increase would be 8 points. Variables that contributed to increase the score the most are associated with *Debt Ratio* and *Number of Open Credit Lines and Loans* for which this application got the maximum possible number of points.

For the XGBoost, the low *Number of Open Credit Lines and Loans* is the only factor pushing the Probability of Default down. One could be surprised by the fact that although the *Debt Ratio* is very low (3%), the contribution of the variable is considered as being positive (increasing the predicted risk relative to the base value).

For the Low-Risk observation shown in the figure 37 below, we can see that both algorithms are associating the *Number of Credit Lines And loans variable* (value of 18) as the only characteristic that could be a driver of default.

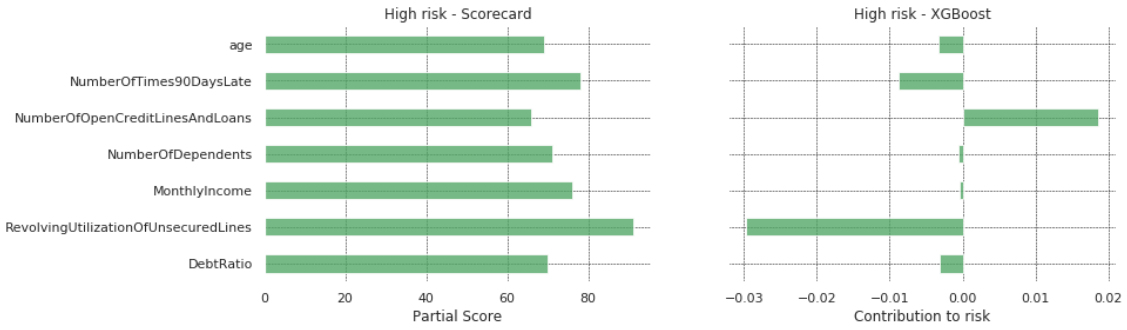


Figure 37: Low-Risk individual Explanation

For the scorecard, this variable is the one contributing the less to the final score with a partial score of 66. Apart for the *Age* variable, this application falls in high partial score buckets for all other characteristics.

For the XGBoost, *Number of Credit Lines And loans variable* is the only variable contributing negatively (pushing the Probability up) to the final prediction. For both models also, the low *Revolving Utilization of Unsecured Lines* (value of 7.55%) is the main characteristic making this application a good candidate for a loan.

8.4 Stability analysis

For the purpose of testing the stability of the explanation methods, we will analyze predictions of both models on the same applications as above after performing simple perturbation to the data. For the Scorecard, it will simply require one partial score to be changed with another while the XGBoost will imply some computations to get the new prediction and to calculate the Shap values.

After doing so, we will be able to see if for the XGBoost, the change in predicted risk and variables' contribution would match the intuition given by the PDP plots presented earlier. Since the *Revolving Utilization of Unsecured Lines* variable was identified as important for both model and played a significant role within the prediction of risk for the two application above, we decided to use it as a perturbation vector as described in the table 15 below.

Risk	Age	Number of Times 90 Days Late	Number of Open Credit Lines and Loans	Number of Dependents	Monthly Income	Revolving Utilization of Unsecured Lines	Debt Ratio
Low	54	0	18	0	18750	7,55% → 30%	35.43%
High	40	1	1	2	4544	100% → **30%**	3.63%

Table 15 : Perturbed Data Points

By setting the value of *Revolving Utilization of Unsecured Lines* to 30% for each observation, we expect the risk to increase for the Low Risk observation and to decrease for High Risk observation. The changes in scores and PD are given in the table 16.

Risk	Scorecard Score	XGBoost PD
Low	521 → **505**	2.8% → **4.1%**
High	418 → **445**	42.4% → **24.2%**

Table 16 : Perturbed Scores

As expected, we can observe an increase in the predicted risk of both algorithm for the initially Low Risk application. The opposite trend is observed for the initially High-Risk profile instance as both algorithms are lowering the predicted risk.

The contrast between contribution before and after perturbations is illustrated and commented below. The exact contributions numbers on which the plots below are based can be found in annex nº 5 of the document.

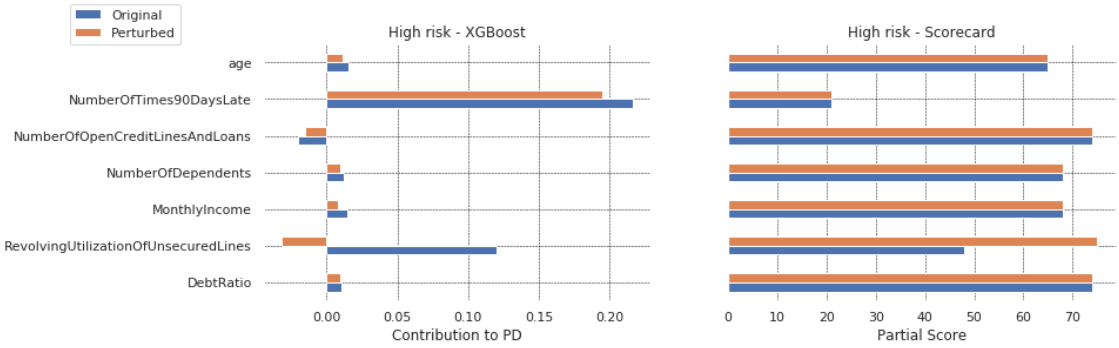


Figure 38: High-Risk Perturbed Individual Interpretation

For the Scorecard model, we can see from the figure 38 that the perturbation led to the increase of partial score associated with the *Revolving Utilization of Unsecured Lines* (from 48 to 75). As all other variables remained at the same value, the perturbed variable was the only vector of change in increasing the final score.

For the XGBoost model, we also observe a significant shift in the contribution of the *Revolving Utilization of Unsecured Lines*. While having a negative contribution in the original set of characteristics, it is now part of the variable that contributed toward the reduction of the risk for that applicant. Relatively to the base value, the variable that was contributing of almost 12% increase is now contributing to reducing the risk of 3.22%.

We also observe that changing the value of one specific variable previously defined led to the perturbation of the contribution of other variables in the output of the XGBoost. As an example, the Number of 90 Days past due is now contributing to a 19.4% increase of the Probability while it was accounting for 21.6% increase before.

In the case of the Low-Risk observation represented below (figure 39), we see that as expected, both algorithms are mainly or fully associating the increase in risk with the increase of *Revolving Utilization of Unsecured Lines*. For the Scorecard, all other variables remained the same so the decrease in score was easily predictable and can be quantified by a human.

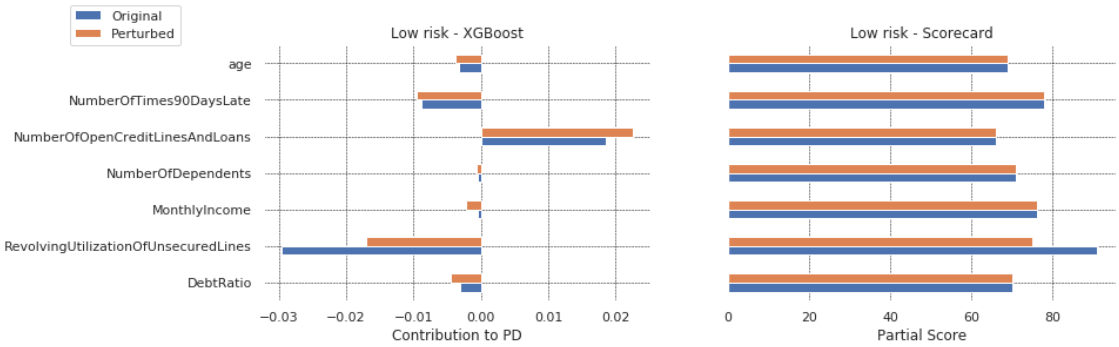
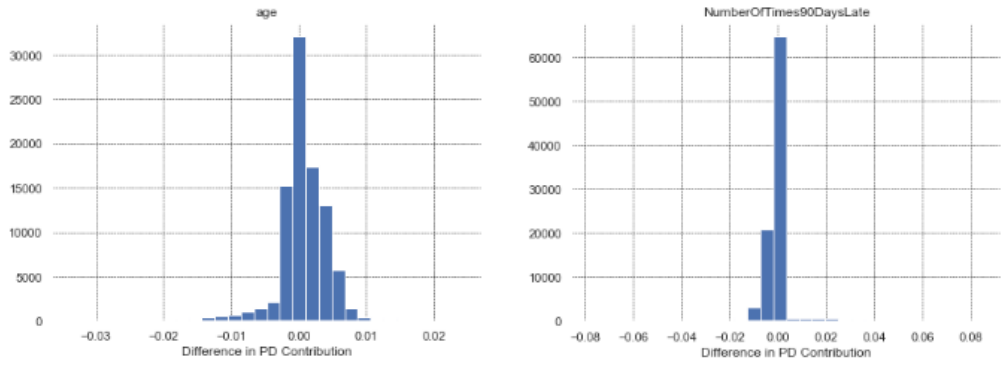


Figure 39: Low-Risk Perturbed Individual Interpretation

As for the XGBoost, although the most drastic change in contribution is observe for the perturbed variable (-2.9% originally against -1.7% after perturbation). We still see some changes in the contributions of other variables. The impact of Number of Time 90 days Past Due variable became relatively less positive while the negative impact of the Number of Credit Lines and Loans increased. Given the global information we currently have of the model, it was not possible to intuitively predict these changes in other variables.

In order to get a better view of the scale of the phenomenon observed above, we decided to analyze the distribution of the changes in contribution within the entire train set under study. After setting the value of *Revolving Utilization of Unsecured Lines* to 30% for the entire training population, we calculated the Shap values for each instance and compared it with its original Shap value. The distribution of this difference is shown in the group of graphs presented in the figure 40 below.



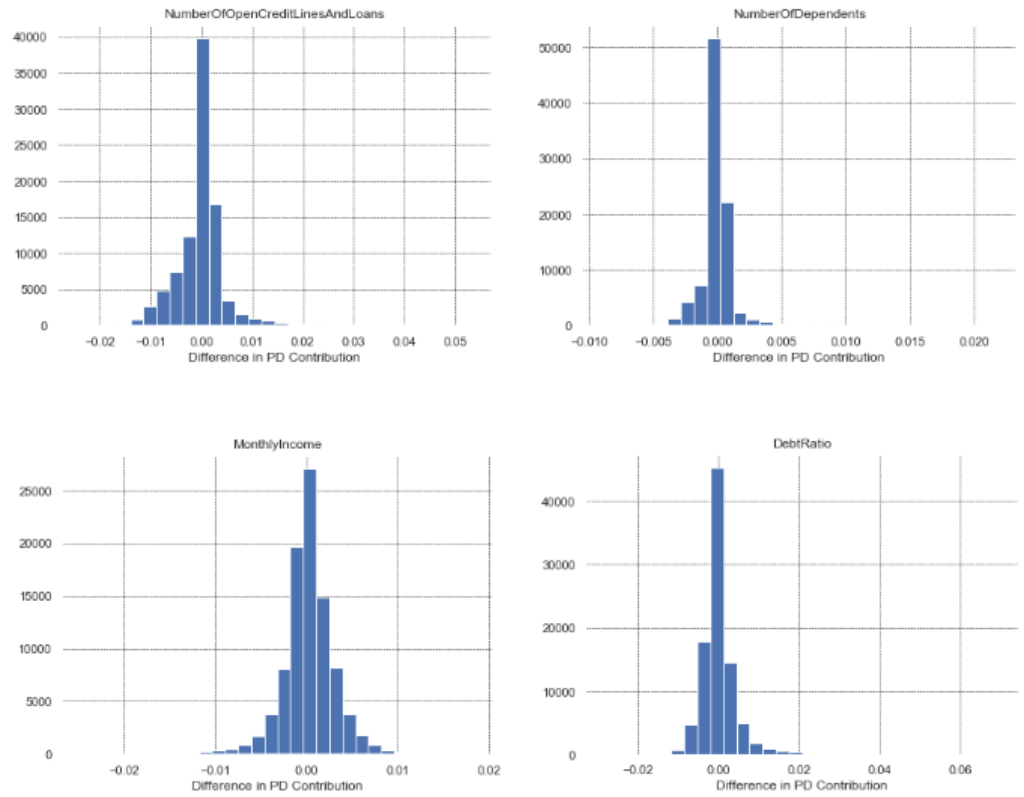


Figure 40: Distribution of Contribution Variations Across All Variables

As one can observe, the contributions of other variables are quite significantly affected by the change in of *Revolving Utilization of Unsecured Lines*, especially the Number Of Time 90 Days Late and *Debt Ratio* variables. Having an intuition about how the change in one characteristic could affect other variable would require a multi-dimensional level of interpretability that might mechanically become a lot less intuitive as each instance might reveal a specific interaction. It would therefore be almost impossible (given that we have seven variables in our model) to exhaustively and comprehensibly cover the full range of possible feature interactions in every set of possible dimensions. As such, it would be very hard to give a fast and intuitive counter factual interpretation of a prediction as we could do with the Scorecard algorithm.

To sum up, we saw that the Scorecard algorithm was more stable as the change in one variable is not affecting the contribution of other variables. Therefore, it is allowing for precisely estimating the impact of a change in the prediction given the perturbation of a given input. On the other hand, for the XGBoost, we observed that contributions could be volatile to change and that global interpretations given by the PDPs were not sufficient for intuitively expect a precise outcome given a specific change in characteristic.

9 Conclusion and Future Work

Conclusion

Throughout this work, we compared the use of a Scorecard (Linear model) with a Black Box model recently developed, XGBoost. We first show that the XGBoost performed better at a pure statistical level. With a higher Roc Auc and a Lower Log-Loss, it seems that the XGBoost has more ability to rank individual according to their risk and to output a probability that is close to the real observed event (default or non-default). We show that the higher performance of the XGBoost algorithm would lead to a lower default rate for a same level of acceptance rate when compared with the Scorecard algorithm.

Regarding interpretability, we tried to understand each model internal functioning at a global level. For the Scorecard model, we were able to extract quantifiable rules by simply reading the coefficients of the model. We could then easily convert these rules into human intuitive language (e.g., "*A high debt ratio will lead to higher risk*").

For the XGBoost on the other hand, we saw that the use of Partial Dependence Plots was limited by the fact that it only shows the averaged effect of a feature on the predicted risk. Doing so, it is not possible to extract simple and intuitive rules that can be extrapolated to the entire dataset. We could, at best, extract a hypothetical statement such as "*A high debt ratio will* ***potentially*** *lead to higher risk*".

For both models, at a local interpretation level, we were able to extract the main reasons that contributed to the risk of a specific application, making possible, for example, to explain to an individual the reason of its rejection. We could argue that at a static level (without trying to make any contrastive analysis by perturbing a given characteristic), the Shap feature contribution method might make more sense than the scorecard model as it actually distinguishes feature with negative and positive contribution.

We believe that the (non-intuitive) instability of local explanations of the XGBoost is a mechanic proof that we were not able to grasp the full functioning of the model at a global level (PDP didn't give any intuition about feature interactions effect that could better explain the predicted output). As such, we observed that perturbing a single characteristic of an instance had unexpected effects on the contributions to risk of other variables. By contrast, in the case of the Scorecard, the effect of a perturbation on a single characteristic only impacted the given characteristic and could easily be quantified through the simple lecture of the Scorecard table. We believe that this later observation is important as it shows the incapacity of intuitively come up with a counter-factual assessment of the risk of an individual calculated by the XGBoost (e.g., if the income was 5000 instead of 3000, the risk would decrease, and a rejected application would

become accepted). On the other hand, we could argue that an explanation given by a Scorecard doesn't reflect the potential subjectivity or specificity of each application.

Future work

The use of a tool such as XGBoost is not yet as scalable as a scorecard within a company. Indeed, a Scorecard model can be share through a sheet of paper, in a mail or an excel file to every risk analyst of a financial institution, each one of them being able to calculate a score as a help for decision taking. On the other hand, a business wide implementation of a tool such as XGBoost would require that each end user of the model would have access to the model for generating scores for new applicants. While this might not be an issue for an online lending platform, it might require substantial structural and logistic changes for more classic financial corporations. The same remark could be made regarding the complexity of the interpretation methods since the explanation needs to be computed for each new applicant.

However, the complexity associated with the XGBoost and the model-agnostic nature of the interpretation method might turn out to be a powerful advantage in the future. As banks and financial institution will adopt new types of models for better performances, these model-agnostic methods will allow the flexibility to change model whenever needed, without the need to change the interpretation method and the operational structure around it. The current (very specific) nature of the interpretation of the logistic Regression is actually what prevent from the testing and potential adoption of different methods.

Given the potential portability of these methods, we believe crucial to make sure that their interpretation can be trusted and can reflect desired business considerations. In that sense, some further work around constraining the learning of complex algorithms could lead the way through the reduction of interactions effects that prevent from a trustful adoption of complex models. Monotonicity constraint, for example, already under experiment within the XGBoost community, could potentially improve the quality of the interpretation methods by enforcing a linear trend (or other) between a feature and the predictions made by the algorithm.

10 Bibliography

Abdou, H. & Pointon, J. (2011) Credit scoring, statistical techniques and evaluation criteria: a review of the literature

Anderson, R. (2007). The Credit scoring toolkit: Theory and practice for retail credit risk management and decision automation. Oxford: Oxford University Press.

Bach, F.R., Heckerman, D., & Horvitz, E. (2006). Considering Cost Asymmetry in Learning Classifiers. Journal of Machine Learning Research, 7, 1713-1741.

Banasik, J., & Crook, J. (2010). Reject inference in survival analysis by augmentation. doi:10.1057/jors.2008.180

Bastani O., Kim C., & Bastani H. (2018). Interpreting blackbox models via model extraction. Retrieved from arXiv:1705.08504

Bolton, C. (2010). Logistic regression and its application in credit scoring. Dissertation (MSc)--University of Pretoria. **URI:** http://hdl.handle.net/2263/27333

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Breiman, L. (1996). Stacked regressions. Machine Learning, 24(1), 49-64. doi:10.1007/bf00117832

Buja, A. & Stuetzle W. (2006). Observations on bagging. Institute of Statistical Science, Academia Sinica. Vol. 16, No. 2 (April 2006), pp. 323-351. https://www.jstor.org/stable/24307547

Casalicchio, G., Molnar, C., & Bischl, B. (2018). Visualizing the Feature Importance for Black Box Models. *ECML/PKDD*.

Charpignon, M-L., Horel, E., & Tixier, F. (2014). Prediction of consumer credit risk, Stanford University.

Chen, J., Song, L., Wainwright, M.J., & Jordan, M.I. (2018). L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data. *CoRR, abs/1808.02610*.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16. doi:10.1145/2939672.2939785

Columbus, L. (2017, July 09). McKinsey's State Of Machine Learning And AI, 2017. Retrieved from https://www.forbes.com/sites/louiscolumbus/2017/07/09/mckinseys-state-of-machine-learning-and-ai-2017/#382f29c575b6

Craven M.W. & Shavlik J.W. (1996). Extracting Tree-Structured Representations of Thained Networks. Computer Sciences Department - University of Wisconsin-Madison. 1210 West Dayton St. Madison, WI 53706.

Crosman, P. (2017, February 14). Is AI making credit scores better, or more confusing? Retrieved from https://www.americanbanker.com/news/is-ai-making-credit-scores-better-or-more-confusing

Cui, Z., Chen, W., He, Y., & Chen, Y. (2015). Optimal Action Extraction for Random Forests and Boosted Trees. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 15. doi:10.1145/2783258.2783281

Doshi-Velez F. & Kim B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. Retrieved from arXiv:1702.08608

Efron, B., & Tibshirani, R. (1993). An introduction to the bootstrap. New York: Chapman and Hall. ISBN 9780412042317

Elith, J., Leathwick, J.R., & Hastie, T.J. (2008). A working guide to boosted regression trees. *The Journal of animal ecology, 77 4*, 802-13. doi: 10.1111/j.1365-2656.2008.01390.x

Ferri, C., Hernández-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. Pattern Recognition Letters, 30(1), 27-38. doi:10.1016/j.patrec.2008.08.010

Figini, S. & Maggi, M. (2014). Performance of credit risk prediction models via proper loss functions. DEM Working Papers Series 064, University of Pavia, Department of Economics and Management. Retrieved from RePEc:pav:demwpp:demwp0064

Fong, R. & Vedaldi, A. (2017). Interpretable Explanations of Black Boxes by Meaningful Perturbation. Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV). doi: 10.1109/ICCV.2017.371

Freund, Y. & Schapire, R. E. (1996) Experiments with a new boosting algorithm In Machine Learning Proceedings of the Thirteenth International Conference.

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1), 119-139. doi:10.1006/jcss.1997.1504

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. Ann. Statist. 29, no. 5, 1189--1232. doi:10.1214/aos/1013203451. https://projecteuclid.org/euclid.aos/1013203451

Garla S. (2013). Extension Node to the Rescue of the Curse of Dimensionality via Weight of Evidence (WOE) Recoding [Brochure]. SAS Institute Inc., Cary, NC. SAS Global Forum 2013

Gilpin, L.H, Bau, D., Yuan, B.Z., Bajwa, Y., Specter, M., & Kagal L. (2018). Explaining Explanations: An Overview of Interpretability of Machine Learning. Retrieved from arXiv:1806.00069

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. Journal of Computational and Graphical Statistics, 24(1), 44-65. doi:10.1080/10618600.2014.907095

Green, D. and Kern, H. (2010). Modeling heterogeneous treatment effects in large-scale experiments using Bayesian Additive Regression Trees. *The Public Opinion Quarterly, 76*(3), 491-511. Retrieved from http://www.jstor.org/stable/41684581

Greenwell, B.M., Boehmke, B.C., & McCarthy, A.J. (2018). A Simple and Effective Model-Based Variable Importance Measure. CoRR, abs/1805.04755.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys, 51(5), 1-42. doi:10.1145/3236009

Gup, B. E., & Kolari, J. W. (2005). Commercial banking: The management of risk. Hoboken, NJ: Wiley.

Hand, D.J. and Henley, W.E. (1997) Statistical Classification Methods in Consumer Credit Scoring: A Review. Journal of Royal Statistical Society, 160, 523-541. https://doi.org/10.1111/j.1467-985X.1997.00078.x

Hara, S. & Hayashi K. (2016). Making tree ensembles interpretable. Presented at 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016), New York, NY. Retrieved from arXiv:1606.05390

Hastie, T., Tibshirani, R., & Friedman, J. H. (2017). The elements of statistical learning: Data mining, inference, and prediction. New York, NY: Springer.

Hernandez, Manuel & Torero, Maximo. (2014). Parametric versus nonparametric methods in risk scoring: An application to microcredit. Empirical Economics. 46. 10.1007/s00181-013-0703-8.

Kazi Rashedul Hasan (2016). Development of a Credit Scoring Model for Retail Loan Granting Financial Institutions from Frontier Markets. International Journal of Business and Economics Research. Vol. 5, No. 5

Khandani, Amir E., Adlar J. Kim, and Andrew W. Lo. "Consumer credit-risk models via machine-learning algorithms." Journal of Banking & Finance 34 (2010): 2767-2787.

Kim, B., Khanna, R., & Koyejo O.O. (2016). Examples are not Enough, Learn to Criticize! Criticism for Interpretability. Part of: Advances in Neural Information Processing Systems 29 (NIPS 2016). Retrieved from https://people.csail.mit.edu/beenkim/papers/KIM2016NIPS_MMD.pdf

Kim, J. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Computational Statistics & Data Analysis, 53(11), 3735-3745. doi:10.1016/j.csda.2009.04.009

Letham, B., Rudin, C., Mccormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics, 9(3), 1350-1371. doi:10.1214/15-aoas848

Lipton, Z. C. (2017). The mythos of model interpretability. Communications of the ACM, 61(10), 36-43. doi:10.1145/3233231

Liu, H., Motoda, H., Setiono, R., & Zhao, Z. (2010). Feature Selection: An Ever Evolving Frontier in Data Mining. Journal of Machine Learning Research - Proceedings Track. 10. 4-13.

Lundberg, S. (2019). SHAP Documentation [Brochure]. https://media.readthedocs.org/pdf/shap/latest/shap.pdf

Lundberg, S., & Lee, S-I (2017a). An unexpected unity among methods for interpreting model predictions. no. Nips: 1–6. http://arxiv.org/abs/1611.0747

Lundberg, S., & Lee, S. (2017b). A unified approach to interpreting model predictions. *NIPS*. arXiv:1705.07874

Lundberg, S.M., & Lee, S. (2017c). Consistent feature attribution for tree ensembles. *CoRR, abs/1706.06060*.

Macedo, P.G. (2013). Receiver Operating Characteristic (ROC) Curve : comparing parametric estimation , Monte Carlo simulation and numerical integration.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence, 267, 1-38. doi:10.1016/j.artint.2018.07.007

Minhai, M. (2017). Cooperative Games. Department of Economics, MIT. Retrieved from https://ocw.mit.edu/courses/economics/14-126-game-theory-spring-2016/lecture-notes/MIT14_126S16_cooperative.pdf

Mitchell, T. M. (2017). Machine learning. New York: McGraw Hill.

Mockus, J., Tiesis, V., & Zilinskas, A. (2014). The application of Bayesian methods for seeking the extremum. Retrieved from https://www.researchgate.net/publication/248818761_The_application_of_Bayesian_methods_for_seeking_the_extremum

Molnar, C. (2019, January 12). Interpretable Machine Learning. Retrieved from https://christophm.github.io/interpretable-ml-book/index.html

Nielsen, D. (2016). Tree boosting with XGBoost: why does XGBoost win "every" machine learning competition? NTNU. http://hdl.handle.net/11250/2433761

Opdal, K., Rikard B., & Thomas H. (2017). Will machine learning and hyperparameter optimization become a game changer for credit scoring?

Opitz, D., & Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. Journal of Artificial Intelligence Research, 11, 169-198. doi:10.1613/jair.614

Ranbir Singh, S., Murthy, H., Gonsalves, T. (2010). Feature Selection for Text Classification Based on Gini Coefficient of Inequality. Journal of Machine Learning Research - Proceedings Track. 10. 76-85.

Ribeiro, M.T, Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. doi:10.18653/v1/n16-3020

Ribeiro, M.T., Singh, S., & Guestrin, C. (2016). Model-Agnostic Interpretability of Machine Learning. CoRR, abs/1606.05386.

Robnik-Sikonja, M. (2017). Explanation of Prediction Models with ExplainPrediction. Informatica (Slovenia), 42.

Rudin, C. (2018). Please Stop Explaining Black Box Models for High Stakes Decisions. *CoRR, abs/1811.10154*.

Satchell, S. E., & Xia, W. (2007). Analytic Models of the ROC Curve: Applications to Credit Rating Model Validation. SSRN Electronic Journal. doi:10.2139/ssrn.966131

Siddiqi, N. (2017). Intelligent credit scoring: Building and implementing better credit risk scorecards. Hoboken, NJ: Wiley.

Sigrist, F. (2018). Gradient and Newton Boosting for Classification and Regression. https://arxiv.org/pdf/1808.03064.pdf

Singh, S., Ribeiro, M., & Guestrin, C. (2016). Programs as Black-Box Explanations. Presented at NIPS 2016 Workshop on Interpretable Machine Learning in Complex Systems. Retrieved from arXiv: arXiv:1611.07579

Strumbelj, E. & Kononenko, I. (2010). An Efficient Explanation of Individual Classifications using Game Theory. Journal of Machine Learning Research. 11. 1-18. doi: 10.1145/1756006.1756007.

Sustersic, M., Mramor, D., & Zupan, J. (2009). Consumer credit scoring models with limited data. Expert Syst. Appl., 36, 4736-4744.

Sutton, C. D. (2005). Classification and Regression Trees, Bagging, and Boosting. Handbook of Statistics Data Mining and Data Visualization, 303-329. doi: 10.1016/s0169-7161(04)24011-1

Thomas, L. C., Edelman, D. B., & Crook, J. N. (2002). Credit scoring and its applications. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1-2.

Ustun, B., & Rudin, C. (2015). Supersparse linear integer models for optimized medical scoring systems. Machine Learning, 102(3), 349-391. doi:10.1007/s10994-015-5528-6

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. SSRN Electronic Journal. doi:10.2139/ssrn.3063289

Wagstaff, K. (2004). Clustering with missing values: No imputation required. In Classification, Clustering, and Data Mining Applications, pages 649–658. Springer.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data mining: Practical machine learning tools and techniques. Cambridge, MA: Morgan Kaufmann Publisher.

XGBoost Documentation. Introduction to Boosted Trees. (n.d.). Retrieved from https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Yang, C., Rangarajan, A., & Ranka, S. (2018). Global Model Interpretation via Recursive Partitioning. CoRR, abs/1802.04253.

Zeng, G. (2014). A necessary condition for a good binning algorithm in credit scoring. Applied Mathematical Sciences, 8, 3229-3242. doi:10.12988/ams.2014.44300

Zhao, Q., & Hastie, T.J. (2017). Causal interpretation of Black-Box models. Retrieved from https://web.stanford.edu/~hastie/Papers/pdp_zhao.pdf

Zhao, Z., Morstatter, F.., Sharma, S., Alelyani, S., Anand., A & Liu, H. (2010). Advancing feature selection research. ASU Feature Selection Repository Arizona State University. 1-28.

11 Annex

Annex nº1

- Scorecard Binning Process Results –

Revolving Utilization of Unsecured Lines	COUNT	WOE
[0, 0.132]	56910	1,334
(0.132, 0.5]	30076	0,428
(0.5, ++]	32943	-1,055
Age	**COUNT**	**WOE**
[0, 36.5]	19061	-0,547
(36.5, 43.5]	16502	-0,313
(43.5, 49.5]	17456	-0,212
(49.5, 56.5]	20156	-0,051
(56.5, 63.5]	19329	0,369
(63.5, 109]	27425	1,004
Debt Ratio	**COUNT**	**WOE**
<=0.423	67225	0,142
>0.423	52704	-0,159
Monthly Income	**COUNT**	**WOE**
[0, 3332.5]	23102	-0,355
(3332.5, 5320.5]	24258	-0,189
(5320.5, 7917.5]	46306	0,131
(7917.5, 1794060]	26263	0,398
Number of Open Credit Lines and Loans	**COUNT**	**WOE**
<=6	48076	-0,165
>6	71853	0,126
Number of Times 90 Days Late	**COUNT**	**WOE**
0	113273	0,386
1	4204	-1,965
>1	2452	-2,818
Number of Dependents	**COUNT**	**WOE**
0	72667	0,149
1	21030	-0,112
2	15569	-0,2
>2	10663	-0,344

Table 17: Scorecard Binning Process Results

Annex n°2

- Correlation tables with 10 variables -

	Revolving Utilization of Unsecured Lines	Age	Number of Time 30-59 Days Past Due Not Worse	Debt Ratio	Monthly Income	Number of Open Credit Lines and Loans	Number of Times 90 Days Late	Number Real Estate Loans or Lines	Number of Time 60-89 Days Past Due Not Worse	Number of Dependents
Revolving Utilization of Unsecured Lines	/	/	/	/	/	/	/	/	/	/
Age	-0,26	/	/	/	/	/	/	/	/	/
Number of Time 30-59 Days Past Due Not Worse	0,11	-0,05	/	/	/	/	/	/	/	/
Debt Ratio	0,09	-0,04	-0,01	/	/	/	/	/	/	/
Monthly Income	-0,03	0,03	-0,01	-0,06	/	/	/	/	/	/
Number of Open Credit Lines and Loans	-0,16	0,18	-0,04	0,2	0,09	/	/	/	/	/
Number of Times 90 Days Late	0,1	-0,05	0,98	-0,03	-0,01	-0,07	/	/	/	/
Number Real Estate Loans or Lines	-0,07	0,06	-0,02	0,32	0,12	0,42	-0,04	/	/	/
Number of Time 60-89 Days Past Due Not Worse	0,09	-0,04	0,98	-0,02	-0,01	-0,06	0,99	-0,03	/	/
Number of Dependents	0,08	-0,21	0	0,03	0,06	0,04	-0,01	0,12	-0,01	/

Table 18: Numeric Correlation Table - 10 Variables

Annex n°3

- Correlation tables with 7 variables –

	Revolving Utilization of Unsecured Lines	age	Debt Ratio	Monthly Income	Number of Open Credit Lines and Loans	Number of Times 90 Days Late	Number of Dependents
Revolving Utilization of Unsecured Lines	/	/	/	/	/	/	/
Age	-0,26	/	/	/	/	/	/
Debt Ratio	0,09	-0,04	/	/	/	/	/
Monthly Income	-0,03	0,03	-0,06	/	/	/	/
Number of Open Credit Lines and Loans	-0,16	0,18	0,20	0,09	/	/	/
Number of Times 90 Days Late	0,10	-0,05	-0,03	-0,01	-0,07	/	/
Number of Dependents	0,08	-0,21	0,03	0,06	0,04	-0,01	/

Table 19: Numeric Correlation Table - 7 Variables

Annex nº4

- Full PDPs for 3 *Monthly Income*, *Number of Time 90 Days Late* & *Debt Ratio* –

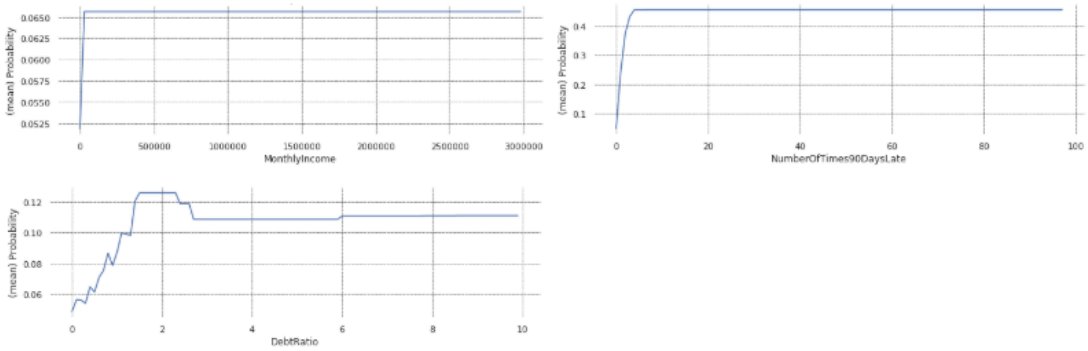


Figure 41: Full PDP for Monthly Income, Number of Time 90 Days Late & Debt Ratio

Annex nº5

- Data from Scorecard and Shap –

	Scorecard Partial Scores			XGBoost PD Contributions	
Variable	**High risk**	**Low risk**		**High risk**	**Low risk**
Debt Ratio	74	70		0.010056	-0.003150
Revolving Utilization of Unsecured Lines	48	91		0.119688	-0.029686
Monthly Income	68	76		0.014493	-0.000424
Number of Dependents	68	71		0.011730	-0.000537
Number of Open Credit Lines and Loans	74	66		-0.019972	0.018581
Number of Times 90 Days Late	21	78		0.216727	-0.008785
Age	65	69		0.015408	-0.003330

Table 20: High-Risk & Low-Risk Individual Explanation (Figure 36 & 37)

	Scorecard Partial Scores			XGBoost PD Contributions	
Variable	**High risk**	**Low risk**		**High risk**	**Low risk**
Debt Ratio	74	70		0.009987	-0.004520
Revolving Utilization of Unsecured Lines	75	75		-0.032291	-0.017022
Monthly Income	68	76		0.007696	-0.002297
Number of Dependents	68	71		0.009295	-0.000627
Number of Open Credit Lines and Loans	74	66		-0.014967	0.022597
Number of Times 90 Days Late	21	78		0.194903	-0.009505
Age	65	69		0.011674	-0.003755

Table 21: High-Risk & Low-Risk Individual Perturbed Explanation (Figure 38 & 39)