Machine Learning

Financial Distress Prediction

Salza Riccardo (3213766), Vento Andrea (3224274), Campi Alessandro (3195247), Lupo Francesco (3196898)

Abstract

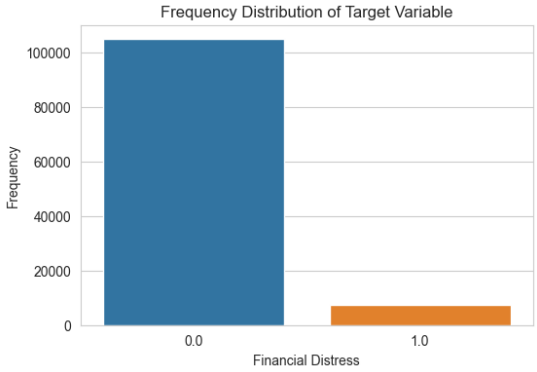
Predicting financial distress holds significant importance across various sectors: the aim is to predict the likelihood of facing financial challenges within a specified time-frame. In the case of individuals, for instance, credit scoring algorithms are generally utilized by banks to assess the probability of default when considering loan applications.
The following analysis is done to build a credit score algorithm to predict the probability that someone will experience financial distress in the next two years. Such model goal is to offer loan providers a predictive tool to make the best financial decisions.

1 Explanatory Data Analysis

The training dataset presents itself with 112500 entries and 12 features. Our target variable, the binary feature 'SeriousDlqin2yrs', is highly imbalanced (the defaulters are roughly 7% of all the observations).
Given that misclassifying a defaulter is worse than misclassifying a non-defaulter, we need to prioritize the former, focusing on the recall and the AUC metric (proportion of actual defaulters identified), possibly accepting a trade off with precision. Despite possibly leaving some interest fee gains on the table, this approach will minimize the default risk exposure for financial institutions. In order to properly determine which factors have the highest influence on the target variable, one needs to investigate the distribution of the available features, as well as the nature, direction and magnitude of the interactions among them.

1.1 MonthlyIncome' & 'DebtRatio

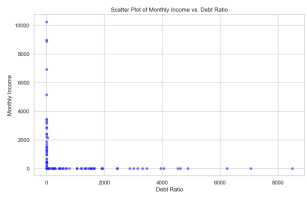
As Monthly income has 22187 NA values (about 20% of observations), we decided to fill them up. However, before proceeding, we noticed how monthly income is crucial in evaluating another feature which is defined as 'monthly debt payments, alimony, living costs divided by monthly gross income': **'DebtRatio'**.



For this reason, ahead of imputing synthetic values for MonthlyIncome, suspecting that the missing values could be useful to understand the behavior of 'DebtRatio', we decided to study its distribution.
The variable is heavily skewed with approximately 21% of the total observations exhibiting a DebtRatio higher than 188%: such measure is concerning.
Furthermore, about 90% of outliers do not show a MonthlyIncome value which could indicate that, for these instances, the DebtRatio variable is actually an absolute Debt, a possible explanation to these unusual high values.

In addition, we discovered that, with respect to observations with missing MonthlyIncome, all the DebtRatios below 1 are actually zeros, possibly indicating that both MonthlyIncome and DebtRatio were missing for these individuals.

From the table on the right, we can clearly see that DebtRatio's extremely high values are registered whenever the monthly income is equal to (or dangerously close to) 0, further supporting the idea that values of DebtRatio above a certain threshold could be interpreted as raw debt instead of a percentage.



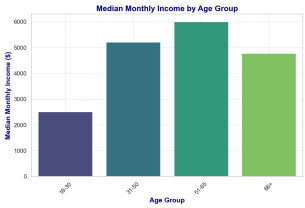
To further strengthen our intuition, it emerged that individuals with extremely high registered DebtRatio (greater than 5) exhibit delinquency rates compatible with the average individual. Having taken into consideration the above information, we decided to consider all DebtRatio values above 5 as absolute raw debt, classifying the MonthlyIncome associated with these observations as missing.

Following this procedure, we have treated the MonthlyIncome of individuals showing extremely high DebtRatio as NA, resulting in these values to be imputed using statistically robust methods, and ultimately transforming the raw Debt into a partially imputed DebtRatio.

We then flag the observations with missing values for Monthly Income as it will be useful when we will have to recompute the DebtRatio of such observations as the ratio between the raw debt and newly imputed MonthlyIncome.

Let's now focus on MonthlyIncome:
As MonthlyIncome has 23961 NA values (about 22% of observations), we decided to fill them in. We initially tried a K-NN for imputation, but this dataset presents many skewed variables, so we decided to try a different approach.
We adopted a grouped-median approach based on the age of the individual.
As people aged differently probably exhibit different median incomes, we divided the individuals into 5 distinct groups: ['18-30', '31-50', '51-65', '66+']. We then proceeded with filling the missing data with the respective group's median monthly income.

Supporting our suspicions, as it emerged from the graph, older individuals generally tend to earn a higher wage. However, we can notice a drop-off in the wage of individuals older than 66, as their median is probably affected by the high number of retired workers: this category of individuals will be further analyzed later in our research.



1.2 NumberOfDependents

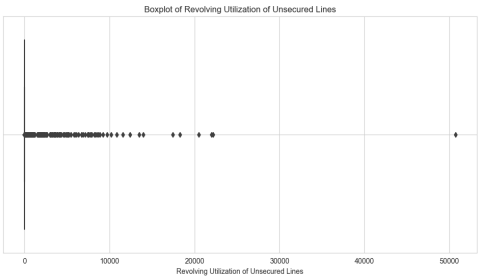
By controlling the null variables, it is found that **'NumberOfDependents'** contain null values: to be more specific, it emerged that it misses 2945 null values (about 2,6% of the overall observations).
As we found that **'NumberOfDependents'** distribution was characterized by a right skewness, with a particularly high frequency for the value 0, the mode, we decided to proceed with filling the NA values following an age-group mode approach by imputing the most popular value (mode) among the respective age group to the missing observations. Since most of the observations fall within the 0 dependents category, most of the imputed values were 0. However, this didn't radically modify the distribution since only 2.6% of original observations were missing.

1.3 Age

Focusing on **'Age'** we saw that there was just a single inappropriate value under 18 (precisely -1). In order to preserve the information contained in that observation, believing it was just an input error, we decided to replace it with the median age.

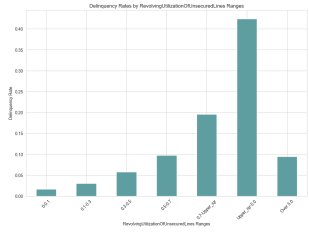
1.4 RevolvingUtilizationOfUnsecuredLines

Analyzing the **'RevolvingUtilizationOfUnsecuredLines'** variable was key to produce some interesting insights. To be specific, RevolvingUtilizationOfUnsecuredLines provides the percentage one gets by dividing how much it is owed on credit cards and personal lines of credit (not including real estate or loans like car payments) by the total amount of credit available. This figure rarely is above 100%. By computing the interquartile range, we saw that approximately 0.5% of the total values (591 observations) exceeded such IQR.



In order to understand and decide how to handle this restricted group of observations, we investigated the differences in the delinquency rates between different utilization ranges.

As expected, the analysis clearly showed a positive relation between utilization values and the delinquency rate. However, from the inspection, it also emerged that individuals characterized by extreme high values of **'RevolvingUtilizationOfUnsecuredLines'** (>5) don't follow such a trend.



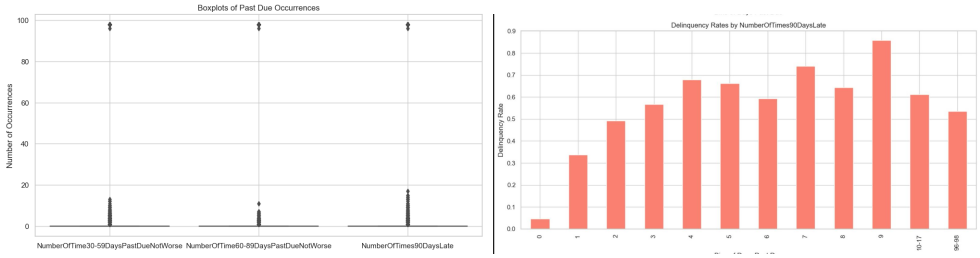
The explanation behind such deviation phenomenon could be traced down to two ideas:

- As there are just 190 observations of this type, they could simply be due to errors in data entry;
- On the other hand, such values might also reflect something peculiar, representing special individuals which leverage sophisticated financial strategies enabling them to sustain these singular Utilization rates.

In both these cases, these data wouldn't be helpful to achieve our goal of correctly predicting the probability of default for the highest number of individuals with the lowest possible error: we decided therefore to drop these observations to better generalize our model.

1.5 NumberOfTime-PastDueNotWorse

Inspecting the variables describing the number of times an individual has been past due on their payments for a specific range of days (**'NumberOfTime30-59DaysPastDueNotWorse'**, **'NumberOfTime60 -89DaysPastDueNotWorse'**, **'NumberOfTimes90DaysLate'**) we discovered that they are all characterized by right-skewed distributions and a high correlation as they measure similar aspects of payment behavior over consecutive periods.
Furthermore, as it is clearly depicted by the three box plots, there is an observation cluster with value 98 possibly indicating an input error or probably, as presumed by us, used as a placeholder for unknown real values.



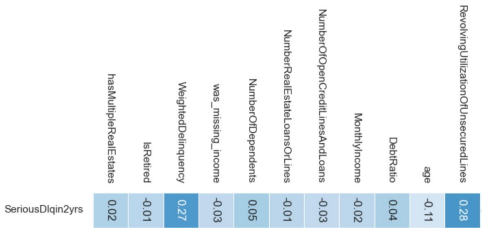
As the same cluster has emerged for the three cases (a validation of our hypothesis), we concluded that it probably was a placeholder. However, noticing the delinquency rate among outliers to be higher than the average individual (roughly 0.55), we decided not to discard these observations, but to substitute their values with the respective maximum non-outlier value for each 'late payer' feature. We then decided to combine these three variables into a single weighted average metric to reduce multicollinearity, which can degrade the model performance by making estimates overly sensitive to minor changes in the input. The consolidated metric simplifies the feature set, making the model easier to manage and interpret. To be specific, in the weighted average, a greater weight was assigned to longer-term delinquencies, reflecting their increased severity and predictability of financial distress.
This approach aligns the metric closely with the underlying financial risk, enhancing the model's robustness and predictive accuracy. Subsequently we displayed the box-plot of this newly generated variable, verifying that higher values of weighted delinquency scores were associated with higher serious delinquency rates.

1.6 Feature Engineering

To improve the information derived from our dataset, we introduced two binary variables: one indicating whether an individual is retired, and the other denoting ownership of more than two properties. The rationale behind these additions is that individuals who own multiple properties are generally viewed as more reliable by banks due to their substantial asset base. Similarly, identifying retirees helps explain instances of low or absent income.

1.7 Multivariate Analysis: Correlation

We then studied the correlation between out target variable and the other feature. Despite not showing a strikingly high correlation value, both WeightedDelinquency and RevolvingUtilizationOfUnsecuredLines are positively associated with the delinquency rate. This information is crucial since, as seen below, these two variables were proven to be accountable for most of the descriptive power of our models.



2 Choosing the model

As we already divided our observations into training and validation data, we simply needed to set them as training and testing groups.
In order to ensure consistency, since some of our models involved randomness, we decided to run the models for 'epochs' number of times, averaging the obtained results.
However, before deploying our models, we must take into account a crucial consideration: as we previously noticed, our target variable is highly unbalanced (roughly 7% of individuals are delinquent). Such disparity in frequency negatively influences learning algorithms to 'ignore' the delinquent class, minimizing the loss function by always guessing '0', i.e. not delinquent.
In particular, despite consistently predicting 0 will yield a 93% accuracy rate, our algorithm will prove largely ineffectual for financial institutions."
In order to create a useful model, we will focus on different metrics to assess the performance of our models:

– **ROC_AUC metric** is crucial as it captures the trade off between 'True positives rate' and 'True Negative rate' across all possible thresholds. This will ensure a way of properly assessing the model performance even for an unbalanced dataset like ours, providing a good measure for the performance of the model;

– **Recall Metric (Sensitivity)**: the true positive rate measures how many delinquent individuals are actually identified. It depends on the threshold used;

– **Precision Metric**: it measures the proportion of delinquents properly identified between the ones flagged as defaulters. Again, it implies a trade off with recall and it also depends on the adopted threshold.

– **F1-Score**: Measuring the balance between recall and precision

We will prioritize the ROC-AUC metric since it accurately reflects the model's ability to distinguish between classes irrespective of specific threshold settings, making it a robust indicator of our model's overall performance.

Logistic Regression
Logistic regression serves as the baseline for our study on financial distress prediction. It models the probability of distress using the variables of our dataset as predictors within a logistic function, valued for its simplicity and interpretability. Although the model demonstrated high accuracy in some instances, this metric can be misleading due to the unbalanced nature of our data, requiring further analysis.

Ensemble Methods
We decided to use ensemble methods (XGBoost, AdaBoost, Random Forest) as they are highly effective for this type of objective due to their ability to improve prediction accuracy and robustness by combining multiple models. They effectively handle imbalanced datasets and provide insights into feature importance, which aids in focusing on significant predictors. This approach is crucial for achieving stable and reliable predictions in scenarios where the cost of errors can be substantial.

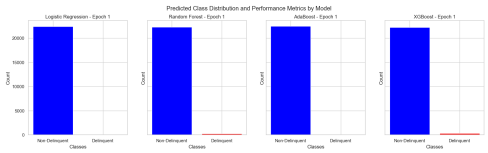
2.1 Techniques and Strategies

Firstly, we run basic versions of our chosen models. Subsequently we started using the following methods to improve their performances: these techniques serve as adjustments to address our dataset's significant imbalance by enhancing the model's focus on the minority class, thus ensuring a more effective classification of defaulters.

Baseline models: Prior to deploying any technique to deal with unbalancedness of our dataset, we decided to run our models on raw data, without even scaling them. At 0.5 threshold, each of the 4 models classified the vast majority of individuals as 'non-delinquents', achieving an accuracy score of 0.93, suffering from the disparity of counts among the target variable. Despite the mediocre recall and precision metric, the AUC score for the models oscillates between 0.79 and 0.83, indicating that, choosing the right threshold value, the discriminatory ability of the models is not that bad.

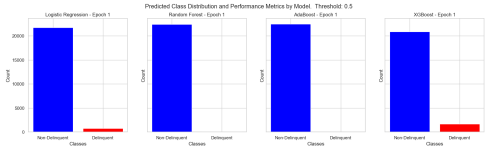
Scaling: since the available features do vary in scale and distribution, by using a scaler as the 'StandardScaler' or 'RobustScaler', we ensure that those with a higher range don't dominate the feature space, leading the learning algorithm to ignore the lower range attributes. Furthermore, some algorithms (such as LogisticRegression) benefit from feature scaling since it uses GradientDescent, hence leading to faster convergence.
In addition, due to the presence and high incidence of outliers and skewed variables, we decided to exploit RobustScaler as it utilizes the median and IQR for scaling. On the other hand, however, tree-based models don't benefit much from scaling as they are characterized by scale invariance: we therefore don't expect much progress for these models.

Weighting: as previously mentioned, our target variable is highly unbalanced. To this end, we opted for testing a simple strategy to increase, weighting, the importance of the minority class: a higher weight will be given to the instances of the 'delinquent' class, possibly improving performances metrics as recall and precision



As expected, Scaling improves the performance of LogisticRegression, while the improvement in XGBoost is primarily due to the higher weight given to the minority class.
From the Precision and Recall metrics, we can see that while RandomForest and Adaboost kept the number of predicted '1' (i.e. predicted defaulters) low focusing on precision, LogisticRegression and XGBoost were incentivized by the weighting to increase the number of predicted defaulters, possibly at the cost of some precision.
In particular, roughly half of the delinquents predicted by AdaBoost and RandomForest were true defaulters, while LogisticRegression and XGBoost were correct in about 25% of the cases. On the other hand, thanks to the higher number of predicted defaulters, the latter models were about to identify more true defaulters in absolute terms.

SMOTE: Synthetic Minority Over-sampling Technique We then decided to apply the SMOTE technique to the models, which is an oversampling technique consisting in creating synthetic instances of the minority class, aiming at improving generalisation and balance across the target variable.
The synthetic observations are created starting from the original instances of the minority class and through the identification of its neighbours. Subsequently SMOTE randomly selects one of these neighbours and creates an instances of the minority class.
Despite its strengths, SMOTE is prone to amplify the minority class noise, possibly creating noisy synthetic observations coming from an outlier in target variable.



Resulting from the oversampling approach, Logistic Regression increased the number of predicted delinquents up to 5000, boosting the Recall metric and identifying the highest amount of delinquents so far. However, this comes at the cost of the precision, only 1 out of 5 predicted delinquents turned out to be an actual delinquent.
On the other hand, AdaBoost improved its performance by correctly identifying 62% of delinquents, being correct in 3 out of 10 cases in which it flagged an individual as delinquent. As we can clearly see, despite weighting and oversampling theoretically serving the same purpose, which is increasing the importance of the minority class, in practice algorithms respond differently based of the technique used. However, although this technique is useful to deal with the imbalanced nature of the dataset, it does not come without its drawbacks. The generation of synthetic samples could, in fact, lead to loss of information and overfitting, as the synthetic samples may not accurately represent the underlying distribution leading to the model learning spurious patterns coming from random noise. Furthermore, this technique increases the dataset size, significantly increasing the computational time required for training the model.

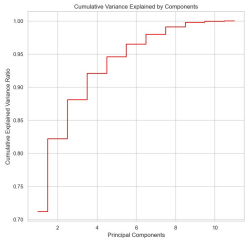
Undersampling Moreover, to avoid the generation of synthetic samples, we also tried the counterpart of oversampling to account for imbalanced datasets, hopefully reducing performance issue: undersampling. This alternative approach involves discarding instances of the majority class to balance the dataset, but it has its drawbacks as well. The loss of data could, in fact, remove important information useful for identifying the majority class, which could lead to reduced model performance. Furthermore, as there are fewer training data points, there is a greater risk of overfitting as the model might learn to fit the noise present in the training data, having little to work with.

2.2 HyperParameters Tuning

In our analysis, AdaBoost and XGBoost emerged as the top-performing models. We therefore decided to tune their hyperparameters in order to maximise their performance. Tuning was critical for optimizing model parameters to our specific dataset characteristics, enhancing both models' ability to accurately prioritize influential predictors. In particular, Adaboost proved to be the best model, maximising the ROC-AUC metric (score = 0.87) whenever it was using 200 estimators and a 0.1 learning rate step with an undersampling technique, performing very well with the right threshold for the classification decision boundary.
On the other hand, XGBoost optimised its performance deploying 100 estimators and limiting the maximum depth of each tree at 3, achieving 0.86 ROC-AUC. Both models particularly emphasized the variables 'WeightedDelinquency' and 'RevolvingUtilizationOfUnsecuredLines', though they assigned their importance differently. AdaBoost attributed significant weight not only to those two variables but also to others like 'NumberRealEstateLoansOrLines' and 'Age', showing a more distributed importance across several factors. Conversely, XGBoost demonstrated a stark prioritization, assigning substantial importance to 'WeightedDelinquency' while markedly less to other variables, illustrating a concentrated focus on this key predictor.

2.3 PCA: XGBoost

Looking the at the importance of the features, we suspected that out model could be achieving similar performances by using less features than we've been feeding to it. To test our hypothesis we decided to run a Principal Component Analysis on one the best performing models, XGBoost. The results are clear, the returns of using an additional feature in our models exhibit a marginal diminishing pattern, meaning that few variables are accountable for most of the predictive power of our model. Unsurprisingly, WeightedDeliquency and RevolvingUtilizationOfUnsecuredLines consist in up to 85 percent of the explanatory power of our model. From the above results, we can clearly see that most of the variance can be explained using a small number of features, possibly leaving space to benefit from the dimensionality reduction. However, one needs to be aware of the downside of this approach, which favours model simplicity and computational efficiency over performance and explanatory capabilities.



2.4 Conclusion and Further Analysis

In conclusion, we decided to deploy the tuned version of the AdaBoost in order to predict the probabilities of delinquency on unseen data. However, when deploying this model in the real world, financial institutions needs to carefully think about their risk tolerance profile, choosing the right threshold to maximise their profits while minimising the default risk.
For further analysis, neural networks for unbalanced binary classification problems could be employed, but they would need to solve the issues of restricted data availability and high requirement for computational power. Furthermore, we are aware that our parameter tuning isn't optimal as it is based on a grid search strategy computed on some of the combinations of the hyperparameters that we thought would perform well for this kind of problem. More sophisticated methods could be employed to further optimize hyperparameter tuning to ameliorate the performance of the models.