Machine Learning Engineer Nanodegree

Capstone Project

Roberto Pinto Martins Junior
March 23th, 2018

I. Definition

Project Overview

Access to credit is important for market economies to develop and flourish, having a big impact on companies and people lives. Banks and financial institutions play an important role in the market, defining who is going to have access to credit.

Money is a limited resource and financial institutions try to evaluate the risks using methods to guess the probability of default and then decide whether or not a loan should be granted. One of these methods is credit scoring which assigns a score to the borrower - it helps the bank evaluate the risk of default and compute interest rates. Using this score, a bank may also assign some credit score category or band, such as bad, poor, fair, good or excellent.

There has been an increase in the interest in machine learning models in credit risk and scoring, as observed in this paper [1]from Moody's, the risk analysis agency. It has also become a business where you can build models: GiniMachine[2].

This project is based on Kaggle competition "Give Me Some Credit"[3] and the dataset was provided along with the metrics used to rank the submissions. This project will follow the same metrics and requirements and add some.

Problem Statement

The goal is to create a model that predicts borrowers' credit score, so the bank will be able to determine the likelihood of a default and then compute the interest rate for the loan. In this binary classification problem, the model will classify if the potential borrower will experience a 90+ day past due delinquency, but will also output the probability of the classification, as required by the competition.

[1]https://www.moodysanalytics.com/risk-perspectives-magazine/managing-disruption/spotlight/machine-learning-challenges-lessons-and-opportunities-in-credit-risk-modeling “Machine Learning: Challenges, Lessons and Opportunities in Credit Risk Modeling”
[2] http://ginimachine.com “GiniMachine”
[3] https://www.kaggle.com/c/GiveMeSomeCredit "Give Me Some Credit"

This model may also be used by borrowers to help them make the best financial decisions.

Using machine learning to solve this problem will involve the following tasks:

1. Load the dataset
2. Explore and visualize the data, trying to better understand it
3. Prepare the dataset to be processed, handle missing information, scale and encode features, if needed.
4. Split the dataset into training and test sets
5. Train some classifiers
6. Evaluate the performance in the test set using appropriate metrics
7. Select the best classifier and fine tune its hyper parameters

Metrics

Evaluation of the models will be based on the requirements of the Kaggle competition: area under curve (AUC); after computing the receiving operating characteristic (ROC) and therefore, the closer the classifier gets to 1.0, the better. Using this metric, it's possible to plot the coordinates created by false positive rate and true positive rate. The closer the area under the curve gets to a square, the better.

Additionally, it's also going to consider the F-beta score for model evaluation, considering it's the kind of problem in which recall is important. This importance is crucial to the bank or credit institution, which need to correctly classify bad payers and thus avoiding giving credit to someone who is going to be late or not paying at all. Therefore, the model must show a high rate of correctly classified bad payers over all the borrowers who previously experienced credit delinquency.

II. Analysis

Data Exploration

This project dataset contains financial and credit information from 251,503 anonymous individuals, already divided in 150,000 records for training and 101.503 for testing, although this test set is meant to be used when submitting the results to the competition.

Table 1 - Data dictionary

Variable	Description	Type
SeriousDlqin2yrs	Person experienced 90 days past due delinquency or worse	Y/N
RevolvingUtilizationOfUnsecuredLines	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
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years	integer
DebtRatio	Monthly debt payments, alimony, living costs divided by monthly gross income	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards)	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years	integer
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children etc.)	integer

Performing a quick exploration of the dataset, it was observed to be definitively an unbalanced one. Some features are sparse, most notably our target class SeriousDlqin2yrs. In this class, individuals who experienced credit distress account for only 6.684% (10,026) of our samples.

This adds some serious implications to the model, even though predicting that SeriousDlqin2yrs is always 0 (zero), it might achieve high precision.

Table 2 - Naïve assumption metrics

Samples in serious credit delinquency	10,026
Percentage of delinquency	6.684%
False positives in naive assumption	0
True positives in naive assumption	0
False negatives in naive assumption	10,026

True negatives in naive assumption	139,974
Accuracy in naive assumption	93.316%
Precision in naive assumption	0%
Recall in naive assumption	0%

If we consider that our model predicts that everybody has good credit (SeriousDlqin2yrs = 0), our model will still present an astonishing accuracy of 93.316% while increasing the credit risk for the financial institution.

$$Accuracy = \frac{true\ positives + true\ negatives}{total\ population} = \frac{0 + 139{,}974}{150{,}000} = 93.316\%$$

It was also observed some missing values, interpreted as NaN values in the dataset, which should be handled accordingly during data preparation.

Table 3 - NaN count in each feature

Feature	**# of NaN**	**%**
SeriousDlqin2yrs	0	
RevolvingUtilizationOfUnsecuredLines	0	
age	0	
NumberOfTime30-59DaysPastDueNotWorse	0	
DebtRatio	0	
MonthlyIncome	29731	24.72%
NumberOfOpenCreditLinesAndLoans	0	
NumberOfTimes90DaysLate	0	
NumberRealEstateLoansOrLines	0	
NumberOfTime60-89DaysPastDueNotWorse	0	
NumberOfDependents	3924	2.69%

Due to the significant amount of missing values in **MonthlyIncome** and **NumberOfDependents**, a strategy must to be devised to handle these missing values. One possible strategy is replacing these values by the mean or median.

Furthermore, missing values are not the only abnormality that may be found in a dataset. Here the identification of possible outliers is done using Tukey's method[4], which however doesn't make any assumption about the distribution of the data.

Table 4 - Outlier study using Tukey's method and a fence factor of 2.0

Feature	Q1	Q3	IQR	Lower Fence	Upper Fence	# of Outliers
RevolvingUtilizationOfUnsecuredLines	0.03	0.56	1.06	-1.03	1.62	528
age	41.00	63.00	44.00	-3.00	107.00	2
NumberOfTime30-59DaysPastDueNotWorse	0.00	0.00	0.00	0.00	0.00	23982
DebtRatio	0.18	0.87	1.39	-1.21	2.25	30815
MonthlyIncome	3400.00	8249.00	9698.00	-6298.00	17947.00	32865
NumberOfOpenCreditLinesAndLoans	5.00	11.00	12.00	-7.00	23.00	1898
NumberOfTimes90DaysLate	0.00	0.00	0.00	0.00	0.00	8338
NumberRealEstateLoansOrLines	0.00	2.00	4.00	-4.00	6.00	473
NumberOfTime60-89DaysPastDueNotWorse	0.00	0.00	0.00	0.00	0.00	7604
NumberOfDependents	0.00	1.00	2.00	-2.00	3.00	7777

Observing the information above, it is clear that the type of distribution must be considered before identifying and removing outliers - simply removing outliers based on Tukey's method may remove valuable information.

Exploratory Visualization

While exploring the dataset, it was observed the presence of some sparse features and abnormalities as well. One characteristic that could particularly affect this project is the distribution type and skewness.

To better understand those distributions, histograms of the features are shown below.

[4] http://datapigtechnologies.com/blog/index.php/highlighting-outliers-in-your-data-with-the-tukey-method/ “Tukey’s method”

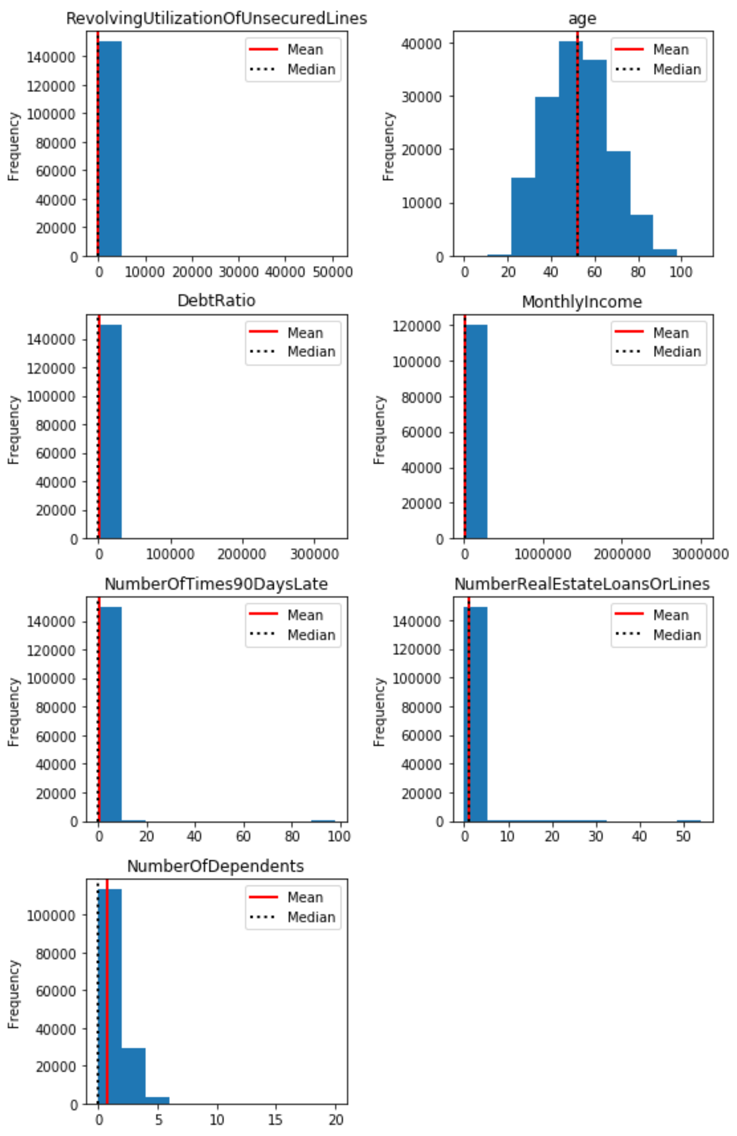

Figure 1 - Dataset distribution by feature

Due to the high frequency of some values in most of the features, it's not possible to observe and explore the distributions in the current scale. Although **age** seems to be the only exception, in which is possible to observe a normal-like distribution.

Algorithms and Techniques

The solution will predict if the borrower will be in financial distress within two years and will also provide the probability of being in that category. First, some aspects of the dataset will be visualized to improve understanding over the dataset and then prepared for being processed in the training and testing stages.

In this binary classification solution, some models will be tested, such as Gradient Boosting, KNeighbors and SGD. These models will be evaluated by metrics such as accuracy, f-beta score and area under curve. The metric that will definitively select the best model is *area under curve*, which is expected to be more suitable to this kind of sparse dataset and which is also a requirement of the competition.

The selection of the algorithms is partially based in skLearn model selection "cheat-sheet"[5].

Stochastic Gradient Descent, or just SGD, has been used since 1960 to train a wide range of models including (linear) support vector machines, logistic regression[6]. It's a very efficient model and was successfully applied in other machine learning problems with sparse dataset[7], and therefore SGD may be a good candidate to solve our problem.

KNeighbors may be one of the simplest algorithms but it is used in some interesting classification problems, such as handwritten digits and image classification. This type of classification works by consulting the dataset and assigning a label to the datapoint based on nearest neighbors, hence called instance-based learning or non-generalizing learning[8]. KNeighborsClassifier may be a good candidate, considering that credit delinquency may share similar characteristics and therefore, KNeighborClassifier can compare a potential delinquent to its neighbors and classify it accordingly.

Gradient Boosting or just GBM has been used to build ranking models for information retrieval systems, search engines and recommender systems, to name a few of them[9],[10]. It's an ensemble method, that is, it's built from simpler, weaker learners (CART usually) and they are iteratively combined to form a strong learner, which tries to improve where it performed poorly in the last iterations. Due to the sparseness of the dataset, a robust algorithm like GBM may perform better where other algorithm may fail.

The dataset will be prepared based on its characteristics and abnormalities found in data exploration section and will be split into training and testing sets, in a stratified fashion, due to the sparseness of our target feature.

These algorithms will be tested using the default parameters whenever possible, except in some few cases. SGD classifier is created using 'log' loss function to be able to use *predict_proba()* method and *max_iter = 1000* to avoid getting different results depending on the sklearn version, which may have different default values. KNeighbors will be used with *n_neighbors = 2* instead of the default 5, since it's a binary classification problem. Gradient Boosting classifier will be

[5] http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html "Choosing the right estimator"
[6] https://en.wikipedia.org/wiki/Stochastic_gradient_descent "Wikipedia - Stochastic Gradient Descent"
[7] http://scikit-learn.org/stable/modules/sgd.html#sgd "sklearn - Stochastic Gradient Descent"
[8] http://scikit-learn.org/stable/modules/neighbors.html "Nearest Neighbors"
[9] https://en.wikipedia.org/wiki/Gradient_boosting "Gradient Boosting"
[10] https://en.wikipedia.org/wiki/Learning_to_rank "Learning to rank"

created with *random_state = 0* to avoid differences in the result of different code executions and *presort = False* to avoid sorting the sparse dataset.

After selecting the best performer, based on benchmark and metrics, it will have its hyper parameters fine-tuned and the competition test set processed and saved.

III. Methodology

Data Preprocessing

Data preprocessing will prepare the input of our models and it is essential to robust and consistent results. It usually involves the selection, preprocessing and transformation of data to be better processed later.

Replacing NaN and missing values

All the missing values interpreted as NaN must be replaced by the median. This strategy is used to avoid missing the entire sample and using the median won't affect the feature statistics. As detected earlier in Data Exploration section, only **MonthlyIncome** and **NumberOfDependents** have NaN values.

Log transformation

The distributions found while exploring and visualizing the dataset, presented some highly skewed pattern. For these highly-skewed distributions, it is common practice to apply a logarithmic transformation[11] on the data so that the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers.

[11] https://en.wikipedia.org/wiki/Data_transformation_(statistics)

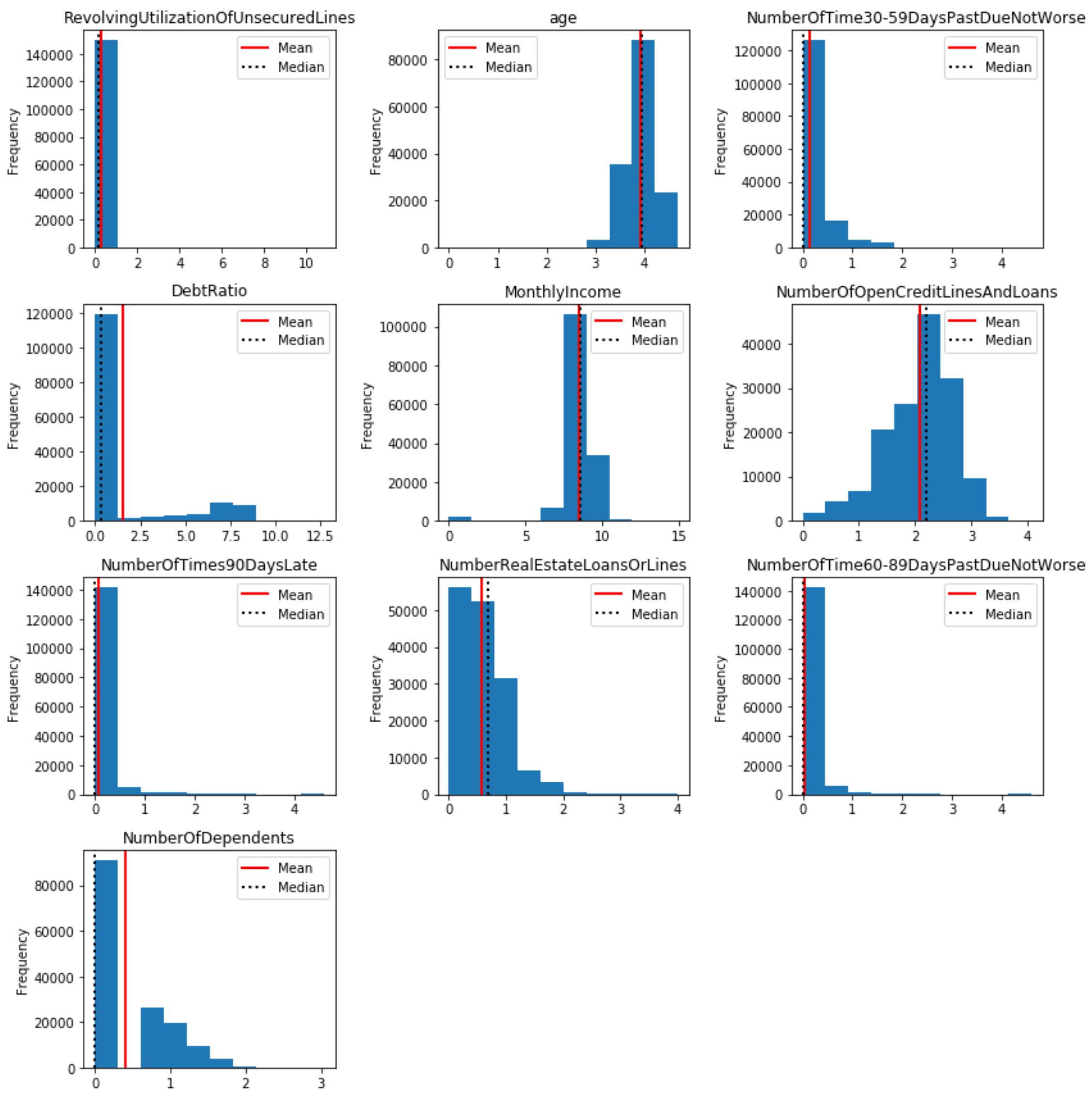

Figure 2 - Dataset distribution by feature after log transformation

Since all features are numerical, preprocessing stages such as conversion of non-numeric categorical data won't be needed.

Training and Test sets

As a final stage of our data preparation, the normalized dataset will be split (both features and labels) into training and test sets in a 80-20% ratio, respectively. The training set will be used to train our model and the test set will be used to evaluate them using 'unseen' data.

Benchmark

As observed before, our dataset is unbalanced – samples of credit delinquency account only for 6.68% of the samples - and therefore one possible benchmark model is the naïve predictor, which could consider that all borrowers have good credit (hadn't experienced 90+ days past due delinquency) and yet would present a 93.316% accuracy. However, we need a better benchmark for our future model.

After preparations in our dataset, a linear logistic regression (with no parameter tuning) will be used as benchmark model. It's a simple predictive model that will be used in place of the naïve predictor, and hopefully, will provide means to compare our models and eliminate the worst performers.

Due to some abnormalities found in the dataset (NaN values), it was necessary to perform the bechmarking after data preparation. Otherwise, the chosen algorithm for the benchmark will fail to process the samples.

Implementation

Some training models will be used and compared, such as Gradient Boosting, KNeighbors and SGD classifiers.

All three algorithms will be used to train a model and gather enough information during the process to evaluate the best one.

The best model will be evaluated using area under curve (a competition requirement), but accuracy and f-beta score will also be used to study each one in the context of a sparse dataset.

Training

The split dataset will be used to train and later evaluate the model. Three different algorithms will be used: Stochastic Gradient Descent, KNeighbors and Gradient Boosting classifiers; and the best performing will be selected for future fine tuning. The choice of algorithms was based in the sklearn chart for choosing estimators[12].

At this point, three metrics will be used to evaluate the models: accuracy, f-score and area under curve using the receiver operating characteristic. In the case of f-score metric, a beta of 2.0 will be used to consider more recall than precision in the harmonic mean.

After training different algorithms, all the information gathered about predictions and metrics will be used to compute and display the Receiver Operating Characteristic (ROC) curve, as well as the Area Under Curve (AUC)

[12] http://scikit-learn.org/stable/tutorial/machine_learning_map/ "sklearn - Choosing the right estimator"

which is a requirement of the competition. Additionally, the training time for each one will be displayed along with metrics for accuracy, fbeta 2.0.

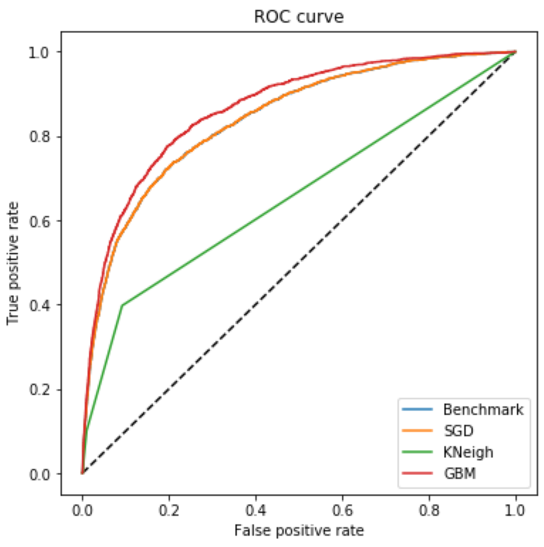

Figure 3 - Candidate models ROC Curves

Table 5 - Candidate models metrics

	Training time	Accuracy	fbeta2	ROC AUC
kneigh	00:00:02.359149	0.930833	0.119468	0.655625
Benchmark	00:00:00.747426	0.935100	0.150496	0.839294
sgd	00:00:39.435598	0.935233	0.149457	0.839456
gbm	00:00:26.220707	0.936167	0.223358	0.865726

It was observed that the best performing algorithm is Gradient Boosting. Gradient Boosting and SGD scored higher than the benchmark, being SGD too close to the benchmark though.

KNeighbors was well below the benchmark and won't be considered for further analysis.

Refinement

The best performance out-of-box was achieved using the Gradient Boosting Classifier, which scored 0.865726 in the Area Under Curve criteria.

After fine-tuning GBM's hyper parameters using Grid Search[13], the following results were obtained. Grid Search exhaustively searches over the specified

[13] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html "Grid Search"

parameters values using the desired estimator (Gradient Boosting), while evaluating every pass using ROC Area Under Curve metric.

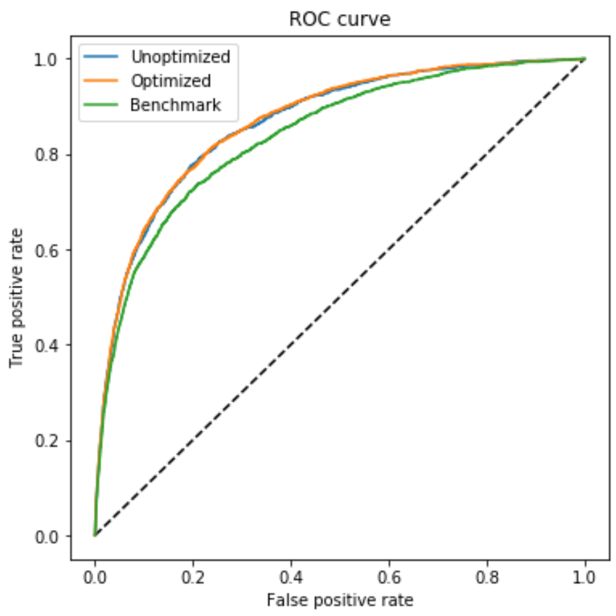

Figure 4 - ROC curves: benchmark, GBM and optimized model

Table 6 - Benchmark, GBM and optimized model metrics

	Training time	**Accuracy**	**fbeta2**	**ROC AUC**
Benchmark	00:00:00.747426	0.935100	0.150496	0.839294
gbm	00:00:26.220707	0.936167	0.223358	0.865726
optimal	00:00:57.790197	0.936333	0.224009	0.867495

IV. Results

Model Evaluation and Validation

The final, best performing model was chosen based on the evaluated metrics, mainly Receiver Operating Curve - Area Under Curve, which proved to be the best metric for this kind of dataset and problem - it's unaffected by class distribution. As expected, accuracy proved to be highly ineffective - our naïve assumption provided a 93.316% accuracy, so did all the models, which presented values ranging from 93.0% to 93.6%.

Table 7 - Models metrics, ordered by ROC AUC

Algorithm	**Training time**	**Accuracy**	**fbeta2**	**ROC AUC**
kneigh	00:00:03.693158	0.930833	0.119468	0.655625

Benchmark	00:00:00.916679	0.935100	0.150496	0.839294
sgd	00:00:51.549854	0.935233	0.150567	0.839410
gbm	00:00:26.674639	0.936167	0.223358	0.865726
optimzed gbm	*00:02:38.668936*	*0.936333*	*0.224009*	*0.867495*

The best model was the one using Gradient Boosting classifier with its hyper parameters adjusted to the values in the optimized column:

Table 8 - Optimized hyper parameters

Hyper ***parameters***	**Default**	Gradient Boosting	**Optimized** Gradient Boosting
loss function	deviance	deviance	exponential
n_estimators	100	100	200
presort	auto	FALSE	FALSE

Nevertheless, it is necessary to justify our statement with further analysis. The optimized model is compared against the benchmark using two evaluation tools: classification report and confusion matrix.

Table 9 - Benchmark classification report

	precision	**recall**	**f1-score**	**support**
0	0.98	0.76	0.85	27995
1	0.18	0.76	0.30	2005
avg / total	0.92	0.76	0.82	30000

Table 10 - Optimized model classification report

	precision	**recall**	**f1-score**	**support**
0	0.98	0.79	0.87	27995
1	0.21	0.79	0.33	2005
avg / total	0.93	0.79	0.84	30000

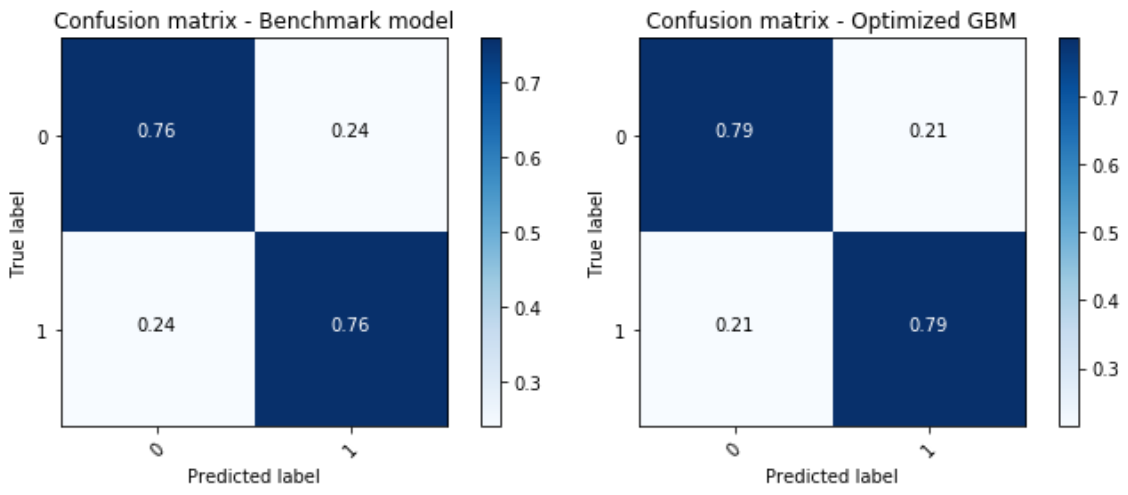

Figure 5 - Confusion matrices

As observed in the classification report and in the confusion matrix, the optimized Gradient Boosting Classifier achieved better scores and metrics.

In the classification report, the recall is one of the most important metrics and Gradient Boosting achieved 0.79, which also influenced the confusion matrix results, showing 0.79 for True Positive rate and True Negative rate.

V. Conclusion

Free-Form Visualization

One possible, and useful, way to visualize the predictions made by our model is creating some kind of credit score ranging from 0 to 100 based on the predicted probabilities, where the higher, the better is someone's score.

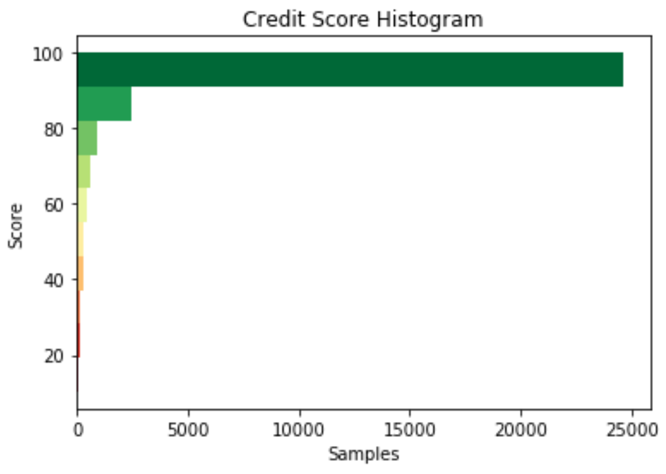

Figure 6 - Proposed credit score

The histogram is also in line with the characteristics of our dataset, which has few samples of persons in financial distress.

Reflection

First and foremost, the dataset was loaded, explored and right there I realized that there was missing data and kinds of distributions quite difficult to visualize, such as skewed distributions and sparse data. Furthermore, these characteristics along could jeopardize the results of the entire project.

The exploring phase prepared the ground for the data preprocessing that would come next. In the preprocessing I had to come up with a method to replace the missing values instead of losing the sample and which would affect the dataset as little as possible - done replacing the missing values with the mean for that feature. The next challenge was transforming the features so that their sparseness, skewness wouldn't be a problem later. The log transformation achieved these results and also reduced the effect of outliers on the algorithms used later in the implementation.

Finally, in the implementation phase, I had to adjust the implementation of the models and the refinement by trial-and-error, which involved lots of iterations and time, but at the end I came up with a solution that allowed me to save and compare the results later. Furthermore, I believe the model could be used by people who wishes to computer their own credit score, helping them deciding if a new mortgage or loan will be a good decision.

Improvement

Actually, there is room for improvement. I would like to include a Principal Component Analysis to identify which features are responsible for most of the variance in the dataset and to use the identified components in the model, observing how they influence it.

Another possible strategy is always research some different algorithms, which could boost either training time and metrics. One strong candidate is XGBoost.