IN-STK5000 – Adaptive Methods for Data-Based Decision Making
Project: Credit risk for mortgages - part 1

Luca Attanasio, S M Mamun Ar Rashid

September 2019

1 Introduction

In this project, five bankers were used to collect utilities and analyze their outcome. In particular, the bankers proposed are:

1. deterministic banker: always grant
2. deterministic banker: never grant
3. random banker
4. random forest banker
5. neural network banker (not fully optimized)
6. perfect banker

The first task of the assignment is to develop a banker's strategy that maximizes the expected utility with a linear utility function $U(r) = r$. More in depth, the task is to choose between granting the loan or not to an individual and maximizing the sum of the expected utilities on a certain number of samples from the testing set. This is considered an individual test. Many tests were done (100) in order to have a better estimation of the performances of each banker, resulting in having 100 total utilities per banker. The perfect banker returns the best action so that we can understand how far the models are from the perfect behaviour.

2 Performances

In this section we analyze the results obtained. The results in Figure 1 show that the best utility is achievable by using a deterministic (always grant) strategy, a neural network implementation or a random forest implementation. Depending on the test, the random forest usually outperforms the other strategies Figure 2.

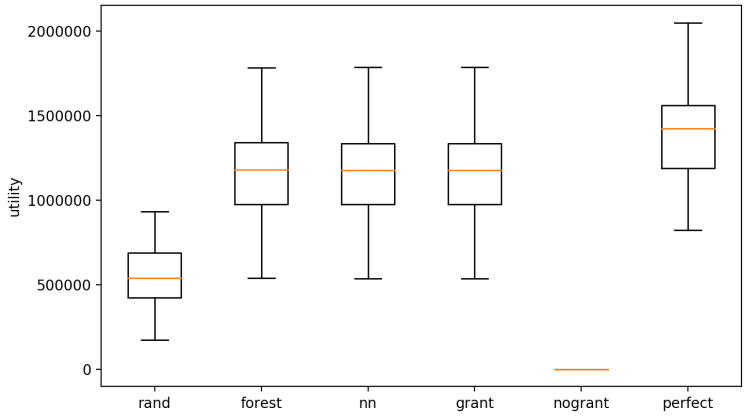

Figure 1: Boxplot of the utilities for 100 tests (random = random action, our = random forest).

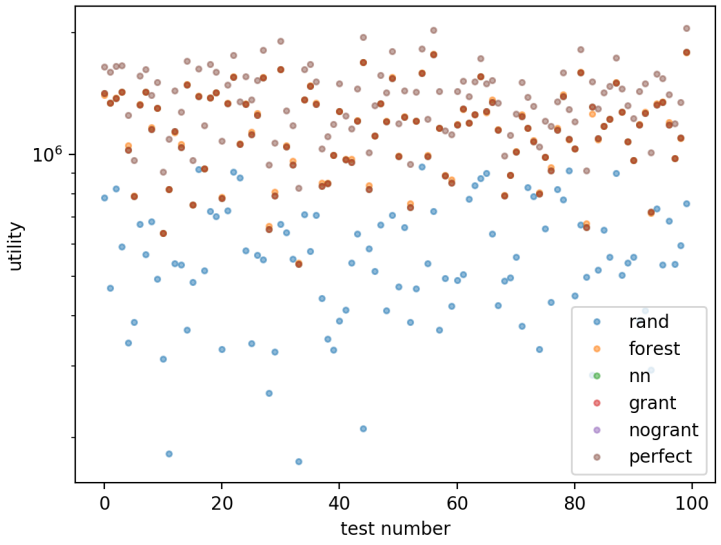

Figure 2: Utilities sum per test.

The boxplot, in fact, can be misleading because we mostly care about the utility on each single test Figure 2. However, it is helpful to understand which banker is performing well enough.

The worst performances are achieved if we never grant the loan. This behaviour is expected because the expected utility is always 0. This happens because we have a high interest rate per month 5%. On the other hand, giving some of the loans to individuals when we are sure about the fact that they would pay back the full amount would be a much better strategy, but it still would not get results better than always granting the loan or by using properly trained classifiers. We are aware that having a much lower interest rate may change the results (e.g. having 0.5% per month); in that case the best option is probably to never grant the loan. In fact, if we have a classifier with a 75% accuracy, then if we grant a loan which won't be repaid, we would loose much more than what we can gain from all the other loans.

The second worst performances are those of the random classifier. Randomly choosing to grant or not a loan is not a good idea because we don't take into consideration any data from the input user. On average, half the loans are granted and half are not, but we don't make any assumption on which can be good and which can be bad.

The deterministic banker that always grants the loan has surprisingly good performances compared to the random forest banker and the neural network banker. Always granting the loan is a good option with 5% interest rate, since we will always gain money if the users can pay it back. On this dataset, many users can pay back the loan, that's why this option would be an interesting one also because we always get the full amount back as the hypothesis say.

The random forest banker has great performances, it outperforms the always grant classifier on most tests, taken individually, but it does not have a perfect accuracy on the testing set. The random forest banker is granting the loan most of the times and not granting the loan just in some cases. This is because the classifier is not super-accurate (75% on average on the test set) and the function which chooses the best action, prefers to give the loan even if the classifier tends to avoid giving it (explained more in depth in the section below). Using the correct utility function is preferred with respect to relying purely on the classifier, since the total utility is higher. The random forest, was fully optimized with respect to the $max\_features$ parameter and max_depth parameter, using cross fold validation.

The neural network banker can still be improved, not much time was used to develop its model but it can surely be optimized to get the same results or better than the random forest. The neural network was built with *tanh* and *sigmoid* activation layers. To get better performances out of the classifier, class weights are calculated. This neural network does not provide better results with respect to the random forest because it acts as an always granter. This behaviour is explained by the fact that the classifier is not performing too well and the resulting best action is to assigns *grant*. More in depth, the predicted probability, in case we grant the loan, is not either too close to 0 or too close to 1. This tunes the always grant behaviour because the utility function always

outputs a positive value (even if small) in case the action is to grant the loan (explained more in depth in the section below).

To improve the accuracy of both the random forest and neural network, the MinMaxScaler was used, in order to rescale the features. Considering the perfect behaviour from perfect banker, the neural network model could be improved to achieve better performances, between the perfect banker and always grant banker.

3 Calculating the expected utility

In this section, the methodology used to develop the expected utility is explained.

The probability $P(\vec{x})$ of being a bad loan, output of the classifier for a given sample (*predict_proba*() function), is used to evaluate the expected utility U of a sample. In particular the formula used to get the expected utility is the following in case we grant the loan for a single sample $\vec{x}$:

$$E(U|A) = m \cdot (1+r)^n \cdot (1 - P(\vec{x})) - m \cdot P(\vec{x}) \tag{1}$$

where m is the amount of the loan, $r = 5\%$ is the interest rate (per month), n is the lending period (in months) and A is the action of granting the loan.

If we don't grant the loan, then the expected utility is null, since we don't gain anything but we also don't risk to loose anything:

$$E(U|B) = 0 \tag{2}$$

where B is the action of not granting the loan.

4 Maximizing the expected utility

In order to evaluate the best action between granting or not granting the loan, we can take the maximum between the expected utility of granting and not granting the loan. The corresponding action is our result. The value of

$$\max\left(E(U|A), E(U|B)\right) \tag{3}$$

is 0 if $E(U|A) < 0$ and $E(U|A)$ otherwise. The most common action is to grant the loan, because usually

$$m \cdot (1+r)^n \cdot (1 - P(\vec{x})) > m \cdot P(\vec{x}) \tag{4}$$

Comment: Instead of maximizing the utility, another option would be to put a lower or upper threshold on the expected utility. This could avoid granting some loans where we are not sure of the outcome and avoid risking too much on users who require really high loans. Setting a lower threshold on the utility doesn't improve the results by much. A better way could be to analyze the distribution of the amount that can be gained in Figure 3.

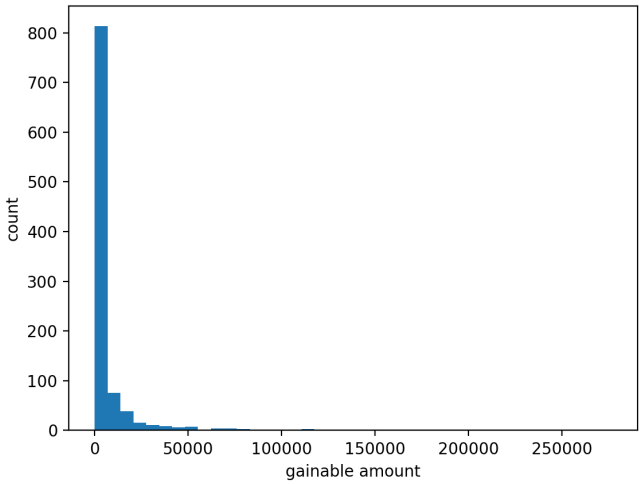

Figure 3: Amount that can be gained.

Comment 2: Another way to solve this problem is to treat it as a regression problem and predict the amount that can be gained. If the amount that can be gained is too low, then the action is to not grant the loan.