Statistical and Machine Learning

In-Class Kaggle Competition

27/03/2020

Jules Motyl

# Introduction

The challenge proposed to us was to predict whether a bank client subscribed to a term deposit or not depending on several variables including the bank data of the client, the last contact of the current campaign, the social and economic status of the client and a few other information.  
The data was already clean and almost ready for processing (set aside some minor encoding and formatting) so I applied several machine learning algorithm with several methods to achieve the best result possible in terms of Kaggle score.

I will walk you through several algorithm that I used and explain their principle, their advantages and their limits.

A small disclaimer considering the method and the objective: I worked considering the objective of maximizing my AUC without any concern for other problems. In a real life setting, some of my strategies might lead to overfitting, sometimes massively, and unreasonable consumption of computation power.

I also chose to work on python as it has been some time since I used the sklearn package and I needed a bit of a refresher.

# Algorithms

I describe my 5 best performing attempts at modelling.

## 1 - Logistic Regression

When linear regression is not appropriate, logistic regression is the simplest model I can think about.

Logistic regression is well suited for classification problems, deriving directly from the linear regression to adopt a different shape:

  
Figure 1: Linear VS Logistic

On the left, the linear regression follows this formula:

B0 is the intercept and B1 to Bn are the different coefficients associated to each variables. For simplification, the graph above only figures one coefficient (or one variable). The addition of all those coefficients and intercept leads to a precise number that evaluates an answer to a quantitative problem.

But as we can see, the line produced by the linear regression doesn’t really fit our points or our targets and the mean square error will jump to high values due to that. It can even give us value beyond 1 and below 0 that, in our specific problem, cannot exist. To better shape our graphical answer to our classification problem, we need a new equation:

B0 and B1 to Bn still have the same signification, but we put them as argument of a log function and divide this log function by itself +1, normalizing it and outputting a result between 1 and 0.

This gives us the graph on the right of Figure 1: much closer to our targets, fitting correctly the data and without any non-sense results as minimum is 0 and maximum is 1.

### The pros

The math are simple and the model is easily interpretable. For every single feature your model can give you a coefficient, a standard error and a p-value which will help you understand the true nature of your problem and the true meaning of your data.   
Its performance is, in most cases, not that bad and its training cost is reasonable.  
Many problems can be solved using a logistic regression, as long as your problem does not require an extreme level of precision and your problem does not encounter any of the limits of this model.

### The limits

If it performs well on most cases, it is almost never the best algorithm in terms of accuracy.  
It also has hard limits in term of use case:

* It can handle only two class to predict (keep in mind the 0 OR 1 output of the function)
* A high number of features will lead to increase in training time and decrease in result, especially if the features are correlated.

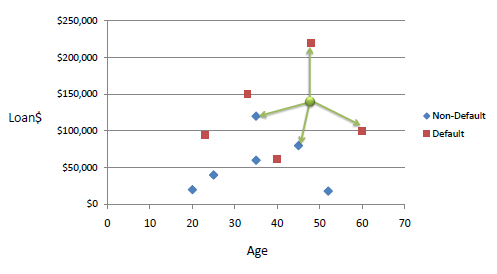
It is a jack-of-all-trade of two class classification problem but not the magical answer to everything.

Considering our specific problem, I would rank it 3rd in term of results, often competing closely with decision tree.

## 2 - KNN

K-Nearest Neighbors uses a different approach than linear or logistic regression, even if to some extend its results can appear similar.

KNN tries to group the data considering its closeness. If two points are close on a graph, they probably belong to the same group, so in case of a classification problem: to the same class.   
One of the advantage of KNN over linear/logistic regression is that it does not assume a linear relationship between the features and the result and therefore it has a much more flexible ‘shape’:

  
Figure 2: KNN prediction in 2D

In the above graph, a KNN model is predicting whether a person will default or not on his loan, depending on both its age and the value of its loan. Here, K = 4 meaning that it will look at the class of the 4 nearest neighbor and predict its class depending on the neighbor’s class. The distance can be calculated differently but Euclidian distance is often preferred, sometimes Manhattan distance can also make sense.  
For our specific example, we see than with K=4, we have two neighbors of each class, making the predicted class a random assignation between those two classes. If we switched to K=3 the result would be different but that makes our model not very reliable.

The two most important things in KNN are the value of K and the distance between observations. The value of K seems straightforward and it is. Tuning it between 3 and 15 almost always gives you a satisfying optimum.  
But the distance between observation is tricky, because it depends on the units of your features.  
In our example, we are comparing age and value of loan. The graph I presented you is scaled in a smart way : a distance of 10 years on the X axis is comparable in terms of distance to an increase of 50 000 dollars on the Y axis. But the algorithm doesn’t know that. For him, 1 more year, 1 more dollar, 1 more kilometer, 1 more house are all the same size in terms of Euclidian distance. If we do not normalize our variables to standardize the distances, we expose our results to non-sense sensibility on some variable variation!

### The pros

KNN can predict more than two classes, making its use cases larger than logistic regression.  
KNN is easy and fast to train, it does not use a lot of computational power.  
KNN can fit well on problem where features do not have a linear relationship with the result.

### The limits

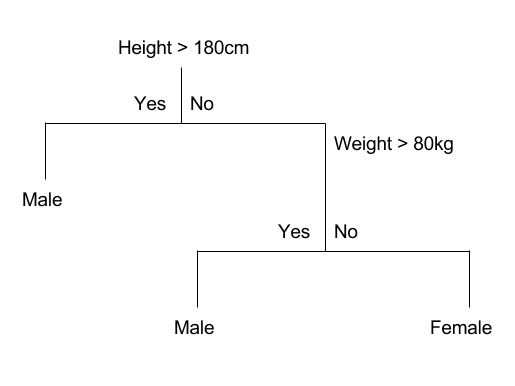
KNN is a non-parametric model, meaning that unlike logistic regression, we can’t have a clear look at the relation between each feature and the result. The normalization of features also makes the data more difficult to interpret.  
The tuning of the K parameter can lead to very different results and therefore the model can be unreliable in terms of results.

Considering our competition, KNN often achieved a great score and is on the long run my second best algorithm in terms of AUC.  
However, at the time I implemented it, I completely overlooked the normalizing of features. Oddly enough, it performed well.

## 3 - Decision Tree

The decision tree model is completely independent from any linear relationship between the features and the target. It splits the population/data into different subgroups depending on a group of variable and an associated threshold.

A decision tree can be decomposed in several components: it has branches, nodes and leaves.

  
Figure 3: Decision Tree

Above is an example of a classification tree. It tries to predict the gender of a person with two features: height and weight. Here we can see 2 internal nodes, being the splits where the tree separates the data input based on a specific threshold: is the height superior to 180cm?  
Yes: subject is a Male. This is a leaf or a terminal node.  
No: the decision continue processing on a new branch, evaluating the weight.

A tree can have as many branches as it wants, and at the end of each branch, the decision path splits in two direction.

### The pros

Decision tree can be used for regression but also for classification problems.  
While having no mathematic formula to explain the results, you can still visualize the tree easily and understand the logic behind it.

### The cons

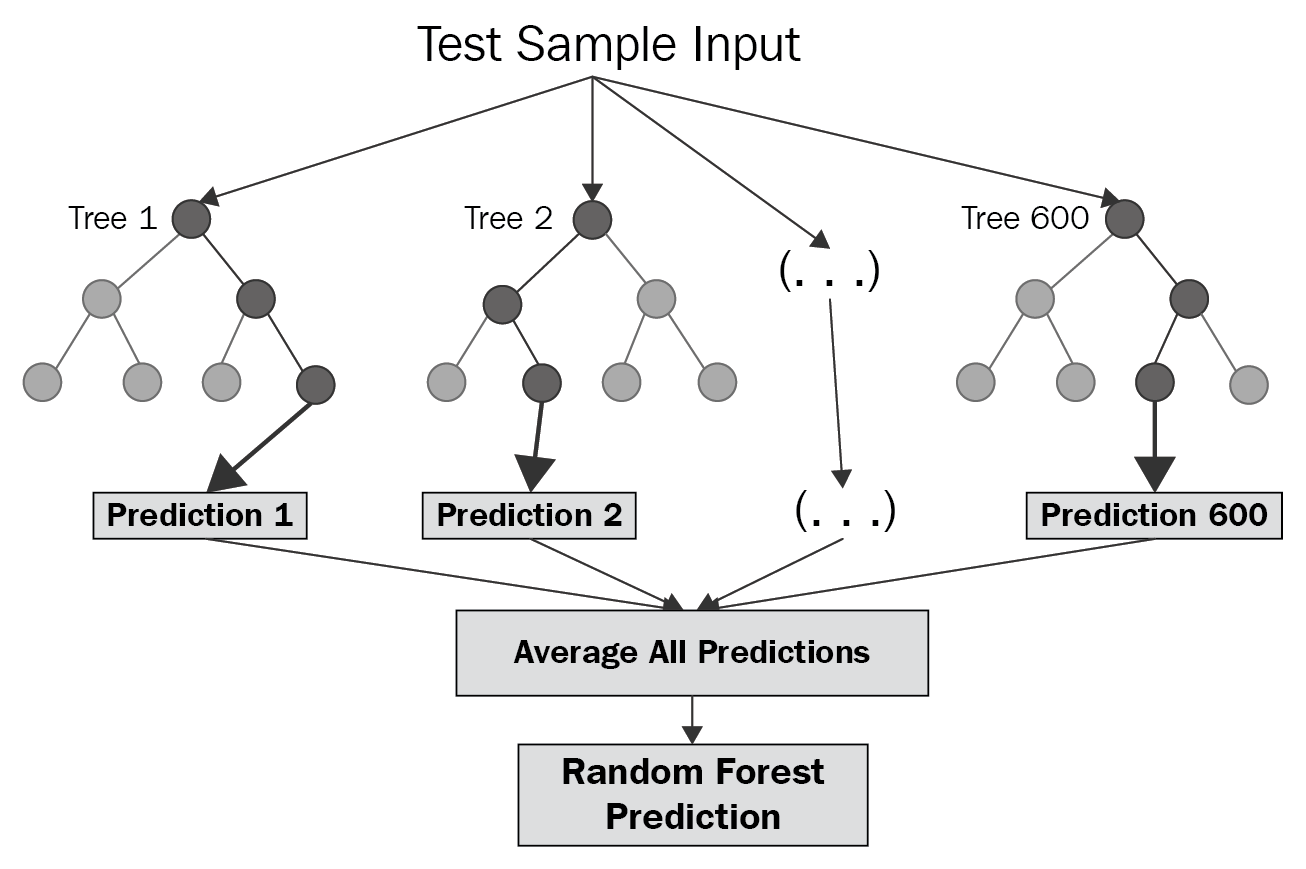
The model can be efficient and clear enough to be explained but it is very simple. Having one threshold per variable reduce the flexibility of the model a lot.

In our problem, the decision was doing alright in terms of results but it was a bit unreliable and the AUC had a high sensibility to changes on the hyper parameters.

## 4 - Random Forest

I remember discussing with a friend of mine working as a data scientist about one of the first modelling problem we had in our program. I asked him for advice about which model to choose to increase the accuracy and told me: “My stat teacher used to say Random Forest beats them all, and so far he has been right.”

Random Forest is just a compilation of decision tree. Well, it is almost as simple as that! Random Forest is training a bunch of different tree at the same time, meaning that they are not correlated to each other.

  
Figure 4: A random Forest

It can have tenths to tenths of thousands of trees, even if the optimum is generally found in the hundreds. But they are not built on the same data: for each tree, the algorithm creates a different subset of the training data and trains the tree on this specific process. Therefore if we inspect each tree, we see that the features selected and the threshold for each of them are never the same (which is of course more useful, otherwise a random forest with no diversity over all its trees would just be equivalent to a decision tree model). Another method is that every tree has a limited number of features, never taking all of them into accounts. It allows for some versatility and avoiding one important feature to override all others.

Once all the trees are creating, they give individual predictions about the result and the forest average them out to make its own prediction. This is an ensemble, or bagging method.

### The pros

By creating a high number of different trees, the random forest avoids both overfitting too much on one variable/part of the data but it also limits variance over its predictions.  
Random Forest usually is one of the best performing models in machine learning.

### The limits

Random Forest does take a long time to train and consumes a lot of computational power. This is due to the number of tree to be generated, which is often in the hundreds where the usual optimum is found.  
Such a large number of tree also makes interpretability extremely difficult and the model is getting very close to a black box system. You can still generate the importance of each feature in the prediction, but not mathematical relationship can be easily retrieved.

Considering the Kaggle Competition, Random Forest is the great winner of all my models. This can be explained by the gigantic time I spent tuning the hyper parameters, sometimes running grid searches for 6 to 10 hours. This unreasonable research for an optimum leveled up my results on the train set to 2% higher AUC, at the cost of a predictable overfitting of the data: I lost around 1.3% AUC between test and train. I expected a Higher loss, but looking back at all the model I submitted, Random Forest is the one that over fitted the less even if it was the most “hyper tuned gridsearched” one.

## 5 - Voting Classifier

The idea behind the Voting Classifier is to combine conceptually different machine learning classifiers and compile all their results as vote to decide on the class to predict.   
Two systems exist on how to compile those votes: a hard one and a soft one.

The hard method takes all the votes and choses the mode as the result. In the case of our two class prediction, it takes the class that has been predicted the most.  
The soft method is a bit more subtle: it takes the probability predicted for each class and for each algorithm and average them. It can then output the highest probability predicted.

This soft method is the one I used, mainly because as the scores were evaluated on AUC it made more sense to output the probability of the class rather than 1 or 0.

### The pros

It increases the reliability of your predictions as the ensemble method will flatten out the eventual bias some of your algorithm can produce.

### The limits

It has few chances of improving overall precision because it is just an average of predictions. Also, it can balance out weaknesses of several algorithm but it will never improve them at the individual level.

# Setup and results

## A word about preprocessing and processing

If you inspect my notebook, you will discover that I did not create any new variable, and I did everything I could to limit the number of features/columns of my training set. This is not out of laziness: it was crucial for me to limit the wideness of my dataset in order to increase the speed of the training of my algorithms as my strategy was heavily relying on grid searching on a large set of parameters.

When it comes to encoding, rather than a classical dummy encoding I replaced each unique value in categorical variables by an arbitrary number that would act like a group number.

## Selection of features

Most algorithm used here have their own features selection process : decision tree, random forest and voting classifier (who isn’t really an algorithm in itself anyway). Considering my strategy of keeping the number of total features as low as possible to gain time and be able to apply a “bruteforce” approach, the only feature selection I did is for Logistic regression, as this algorithm is not well suited for very large number of variables.  
In this case, I used a loop that selects at each iteration one more variable to feed to the logistic regression. Every time it selects the variable that increase the most the AUC of the model. Then, I just lookup the ideal mix of variable and start tuning hyper parameters from here.

## Metric

The metric used is AUC, has this is the metric that evaluates our performance in this competition.  
You might see some confusion matrices as well, I used them sometimes to better understand the performance of some models. Considering that the test set has a heavily unbalanced class distribution, some models could achieve a good AUC with a lot of false negatives.

## Results

|  |  |
| --- | --- |
| **Model:** | **AUC:** |
| Logistic Regression | 0.7809360938962889 |
| KNN | 0.7845304116867219 |
| Decision Tree | 0.7880261935336496 |
| Random Forest | 0.7991040168098978 |
| SVM | 0.7293445169692849 |
| Voting Classifier | 0.7761487261526606 |