Solving financial and banking problems with Machine Learning

Selma SKIREDJ 20113718
Cheikh Tidiane CISSE 20092990
Mohamed CHIBANE 925760
Cheikh Ahmed Tidiane FALL 20088392
Hack YACOUBOU SOKA 20125793
Université de Montréal
IFT 6390 - Machine Learning

Abstract

Classification algorithms are used in a multitude of ways and simplify a problem into providing us binary or multiclass solutions. We would like to see how classifiers deal with two very different financial problems. The first one applies to trading stocks daily using technical and economic indicators as tools for determining order placement. We would like to see how four different classification algorithms fare in predicting daily price direction on a ETF tracking the SP 500 index (NYSE: SPY). The second financial problem we seek answers to is, considering the characteristics of a client base and records of previous marketing campaigns, whether a new client be likely to subscribe to a term deposit at a banking institution using the same four classification algorithms. To some extent this might be useful to implement new products and assess the receptiveness of the client base and better target it.

1 DATA ANALYSIS

1.1 TRADING DATA

1.1.1 Creation (and description)

The problem of developing profitable trading strategies is one that many people are continuously attempting to solve. It all starts with what type of data are we going to base our trading decisions on. There are different tools that one could use in order to generate trading signals. For the sake of the exercise, we created a dataset that contains 12 features (2 technical, 10 macroeconomic). Each feature was selected considering the fact that the S&P 500 is the world's largest index and therefore is more likely to be strongly influenced by macroeconomic factors. For example, the Federal Funds rate is key in guiding economic growth and reducing inflation which directly impact the corporations that are listed in the SP 500. The principal hypothesis on which the dataset is built upon are:

1. Overnight market changes hold some information.
2. Short and medium term fluctuations affect the following day fluctuations.
3. Currencies, oil and gold prices are relevant indicators.
4. Central bank rates from major economies hold important information.
5. Inflation affects asset values.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

The dataset contains a total of 4725 days of trading with 12 features (4725 x 12) and a multi class output (4725 x 1).

1.1.2 Preprocessing

To make our dataset ready to be applied to the different classification algorithms, we had to input daily raw market data from january 2000 to november 2018 and then transform it into ready to use data. All relevant features were also extracted from the same period. Market prices are usually provided in a "open, high, low, close" format and to make use of them we had to process them. The output is multiclass:

(i) **1** = Buy when the market opens (take a long position)

(ii) **0** = Do nothing when the market opens

(iii) **-1** = Sell when the market opens (take a short position)

To determine what constitutes the decision boundaries and to make the exercise more realistic we introduced a commission rate of 0.5% of the trade value which makes it that not every positive or negative return is worth causing a trade. More details on the specific methods for calculating the features and outputs are provided in the annex.

1.2 BANK TELEMARKETING DATA

1.2.1 Description of the banking telemarketing database

This dataset is the result of direct marketing campaign from a Portuguese banking institution. The marketing campaign was essentially based on phone calls to clients to attempt to make them subscribe to a term deposit. We have at our disposal a dataset with 41188 examples ordered by date (from May 2008 to November 2010), 20 features and 1 output variable labeled "yes" or "no" to refer to the success or failure of the client contact. The dataset forms a matrix of size 45211x21. There is no empty field in the database nor missing values. More details about the attributes of the database are provided in Annex table 2. We are working with two types of features mixed in the database: categorical (10) and numerical (10) that are then spread over 4 categories:

(i) bank client data,

(ii) related with the last contact of the current campaign,

(iii) other attributes,

(iv) social and economic context attributes,

With such a combination of different types of factors we now have the necessary features to attempt to predict adhesion to a deposit term with our models.

1.2.2 Visualization of the bank telemarketing database structure

To better capture the relationships between the variables and the regarding classes, we visualize the dataset. This allows us to show the distribution characteristics of our variables (categorical and numerical) and their significance in relation to the output variables. Doing so we have a more precise understanding of the problem and helps us in determining the best approach to achieve our goal of building and effective and accurate predictive model (See figure 5 in the Annex).

1.2.3 Data insight and main statistics

To gain knowledge of the data we are working on, we will be investigating the structure of the bank clients features listed in the database. Examining the bank client data category, we observed that it is essentially composed of married people (61% of the client base), aged around 33 years (clients between 29 and 39 years of age equals nearly 50% of the database), working as administratives, blue-collars or technicians (62% of the database), with a university degree or high school diploma (53% of the database). We also noticed the bank recorded only 3 acknowledged credit defaults while 20% of the clients credit situation is litigious. Knowing the majority of the clients posses a housing loan (approximately 51% of the client base), we can assume with a certain degree of confidence that

they would not be interested in saving or investing money on a long term basis as they are already engaged in paying back that debt. This intuition is confirmed in the sense that more than 88% of the client base declined the subscription to a term deposit. In addition, we were provided with the fact that more than 95% of the client base was never contacted before the current campaign and among them 90% dismissed the bank term deposit proposal. Moreover, we observed that the clients were approached mainly during the summer season with an emphasis put on May and July, directly via their cellular phones. An important issue we anticipate to be facing in the following sections of our work is about unbalanced data as the output class "no" really prevails in the output data (88%).

2 CLASSIFICATION ALGORITHMS OVERVIEW

2.1 Support Vector Machine (SVM)

SVM is a learning algorithm for classification. For a given classification problem, this algorithm learns a decision function that can be expressed in one of the following forms :

for binary classification :

$$f(x) = sign(g(x)), \qquad g(x) = \mathbf{W}^T\mathbf{x} + b\ .$$

Where g is a linear discriminant function, $\mathbf{W}$ and b are the parameters.

The goal of the algorithm is to find a hyperplane that that find the best way to separate the data and which provides the biggest margin. The margin is the distance from the hyperplane to the nearest training example.

2.2 Logistic Regression

The Logistic Regression algorithm is mostly used for binary classification ($t \in [0, 1]$, where t is the target), the aim is to estimate the conditional probability $\mathbb{P}(t = 1|\mathbf{x})$ with $\mathbf{x}$, the inputs vector. A smooth nonlinear function is chosen to predict that probability, like $\mathbf{sigmoid}$ for instance, this type of functions is also called activation function.

Here,

$$f(\mathbf{x}) = sigmoid(\mathbf{W}^T\mathbf{x} + b),\ sigmoid(x) = \frac{1}{1 + e^{-x}}\ .$$

2.3 Random Forest

A Random Forest is a learning algorithm that can be applied to classification, regression or all other problems for which one can apply decision trees. The principle of this algorithm is based on a set of trees in which one trains each decision tree slightly differently by using both :

- **Bootstrap :** random selection of examples with replacement.
- **Random selection:** on a subset of features (reduction of the input dimension).

Each tree has a depth chosen by cross-validation. In order to predict the classes, one uses a majority vote like the one used in Bagging.

2.4 Neural Network

A neural network aims to imitate biological neural networks. It offers a framework that makes many different machine learning algorithms work together in order to process complex data inputs. There are 3 kinds of layers :

- The first one contains the inputs, each element of the inputs is considered as a neuron.
- The second type of layers is what we call the hidden layers, they can be as many as we want. But note that, the more they are, the more complex the network is.

- The last one is the output layer, that contains the outputs.

All the neurons are connected to each other as with synapses in biological neural networks. From one layer to another, we apply an activation function such as **sigmoid** or **RELU**, etc, to get the outputs, so that an output layer for a layer is an input layer for the next one. The objective of the Neural network is to learn all of the links between every layer to be able to predict accurately an output from a given input.

3 THE TRADING PROBLEM

3.1 What do we wish to do?

The dataset we wish to use to allow for the algorithm to predict the correct decision to make daily holds a multitude of complexities but none greater than the fact that each point holds a certain time dependency with the data around it and the overall economical and political context of the time. A specific period might hold a certain pattern, a certain level of volatility, etc. that another doesn't share with it and therefore how we determine the splitting of the dataset for the train phase holds a very important value in the quality of the outputs. Simply splitting the full data set into one simple split is overly naïve and, from a purely financial perspective, holds no logical sense. We therefore decided to work with 3 different formulations as to how to split the data and then picked out the one we find is most fitting to a trading problem. More details are provided in the next section.

3.2 How to prepare the algorithms?

3.2.1 Description of the splitting strategies

Splitting strategy 1: All records up to the split point are taken as the training dataset and all records from the split point to the end of the list of observations are taken as the test set. The first 80% of the data are taken for training the models. The last 20% are taken for testing the models.

Splitting strategy 2: Splitting the time series into train and test sets multiple times. Over time the training set is extended, and we retrain the model on each split. In the end, we average the split accuracies together to determine the overall model accuracy. This method is called "extending window".

Splitting strategy 3: We used a method called "sliding window". The reasoning behind this method is that, as time passes, data closer to the moment we wish to predict hold more relevancy and therefore to use very old data is not as important. We end up only considering the time equivalence of a quarter which is approximately 60 working days for training the model and then predicting the output for a single day. For each following day, we slide the 60-day training period by 1. As the size of the training set is always the same (60), and as the nature of the variables and the observations are consistent through time, we decided to dedicate roughly the first 25% of the data (1000 observations) to the task of finding the optimal hyper parameters that will then be used to train the data several times on training sets defined with a sliding window of the same width (60).

3.2.2 Determining the hyperparameters

In order to find the proper hyperparameters to apply to each of the models, we used the sklearn tool class called GridSearchCV which tests out all possible combinations of hyperparameters and finds the ones that maximize the accuracy (minimize loss) on the validation set by using cross-validation. For the Logistic Regression optimal hyperparameters, the SVM classifier and Random Forest, the optimal hyperparameters are provided in table 2 in the annex.

With regards to the neural network, a large number of hyper-parameters could be tested such as the batch size, the learning rate, the number of epochs, the activation functions, the number of hidden layers, the number of hidden neurons and so on. A grid search that would compute all the combinations of these hyper-parameters would take a very long time to be run. Indeed, this strategy was tested on a sample of the dataset using parallelization, but didn't yield to satisfying results. The use of a large amount of data was necessary but the procedure was too long. For this reason, we

decided to tune the hyper-parameters for the neural network manually. This led us to the final hyper parameters provided in table 3 in the annex.

3.3 Results and comparison

Table 1: Test accuracy score for the trading data

	Strategy 1	Strategy 2	Strategy 3
Logistic	71.6%	44.5%	60.4%
SVM	71.5%	58.7%	60.5%
Random Forest	71.5%	57.4%	60.5%
Neural Network	71.3%	51.2%	X

The results show that Strategy 1 holds the highest accuracy score. In addition, it also possesses the fastest computation time as it takes a few seconds for the fastest algorithms (Logistic regression, SVM) to run and a few minutes for the slower algorithm (Random Forest) to run. This computation time includes the necessary time for completing a grid-search for all of the necessary hyper parameters, fitting the model on the training dataset and predicting the outputs on the testing dataset. However, the big disadvantage of this strategy is that it gives as much weight to recent data as to the oldest example in the dataset whereas, in the markets, more recent data holds a certain amount of information that older ones do not. If the data contains local abnormalities, or some specific behaviours related to a real, particular context (a financial crisis for example), these could be hidden among the large amount of data and ignored while predicting new behaviours.

Strategy 2 describes a context where new data is periodically received and added to previous data. Even though it theoretically gives more robust estimations of the expected performance of the chosen method and configuration on unseen data, this method present a lot of disadvantages. First of all, it requires multiple models to be trained and evaluated. This method is the one that takes the longest time because of the repeated grid search everytime the date is expanded (see description of splitting strategies). In addition, it is not intuitive and is hard to be interpreted in a financial/trading context, it is preferred to take into account recent events to predict the coming ones. However interesting results can be shown from this method as observed in the following figure:

The drastic drop in the accuracy in the third split, which corresponds to predicting data from the 26th of February 2008 to the 26th of October 2008 shows that it was harder to predict the returns during the financial crisis. The crisis caused a period of high volatility which makes patterns harder to predict.

Strategy 3 is a good compromise between the first two as it consistently yields the second highest accuracy score and is more robust as multiple models are trained. The greatest advantage by using this split is that it is easier to interpret since only recent values are taken into account to predict the returns and we aim to predict only the day following the set used for training the model. The model is constantly evolving over time and concentrates the learning phase only on recent data. In addition to this, this splitting strategy also takes a relatively reasonable time to be computed.

All in all, for the Logistic Regression, SVM and Random forest algorithms, we will use the results from the splitting Strategy 3 in order to compare the performance with the bank telemarketing dataset algorithms because we believe that it holds the most validity in financial theory. However, Strategy 2 will be used for comparison of the neural network. The reason why we did not apply Strategy 3 to the neural network is because of computation time. We simply do not have the computing power to train a total of 3665 neural networks. Moreover, as described in a previous section, the sliding window covers 60 examples at a time which we believe is not sufficient to training a complex and deep neural network. Indeed, simple neural networks on this problem were not efficient as they could only predict only one or two classes over the 3 classes of the output variable.

4 THE BANK TELEMARKETING PROBLEM

4.1 Formulate the problem

The research is focused on targeting through telemarketing phone calls to sell long-term deposits at a banking institution. The models we want to create and implement must predict in which category, a client response to a telemarketing call from a Portuguese bank is likely to belong. As explained earlier, in our classification problem, there are very low number of samples for the "yes" class. As a consequence, the challenge faced is that the algorithm does not get sufficient look at what constitutes a "no" classification in order to properly generalize. We will seek to test which type of algorithm is not affected by this data structure and, for the ones that are, we will provide some methods circumvent this issue.

4.2 The data preprocessing phase

The duration attribute highly affects the output target (e.g, if the duration=0 then y="yes") therefore this feature is discarded. The correlation matrix of the features given in figure 2 shows two strong dependencies ("euribor3m" and "nr.employed") to one feature ("emp.var.rate").

With regards to categorical features, we encoded them into one-hot vectors. We do this to transform categorical features into numerical ones without losing the key information we want to learn. We previously talked about the necessity of addressing the problem of unbalanced data in our dataset if we found our algorithms sensitive to this type of structure. A method we adopted to address this, is to restore good balance in the dataset between the two class output. To do so we will compensate the under-representation of the "yes" class by giving more weight to this class during the learning phase. Also this method is simple to follow, we do not ignore the possibility of losing important information in the process; that is why we try to keep the representativity of the main class merely intact in the training data. And finally, to avoid overfitting, we proceed to the split of the data, by separated the dataset into a 70% train-valid set and the other 30% is allocated to the test set.

4.3 Determining the hyperparameters

To get the best combination of hyperparameters for the Logistic Regression, SVM and Random Forest, we performed the grid search method, which was previously explained. The resulting hyperparameters are provided in table 5 of the Annex. Regarding the neural network, our algorithm is a supervised "vanilla" neural network with one input layer of 64 nodes, one hidden layer of 4 nodes and one output layer with two dimensions (Figure ##). This is a basic form of a neural network since the task is relatively simple: binary classification. A critical hyperparameter is the activation function. We chose a logistic activation function instead of rectified linear unit function to make an appropriate network regarding the task of binary classification. Doing so, we are building a network above the probit regression principle, hoping to learn deeper than a basic logistic regression.

4.4 Modeling strategy and performance evaluation of the algorithms

We experimented with two strategies: a naive one that ignores the particular structure of the data and a second that does. For the first one, we did not attempt any corrective technique on the unbalanced data. The second strategy holds a compensation weight as explained in section b.

Table 2: Algorithm performance in terms of accuracy on the test set for the two strategies

	Strategy 1	Strategy 2
SVM	89.30%	88.75%
Neural Network	89.91%	89.78%
Logistic	90.06%	89.81%
Random Forest	90.12%	89.38%

Strategy 1: No compensation for the under-represented class in the dataset applied.
Strategy 2: Compensate the under-representation of the minority class in the dataset for the algorithms using **class_weight parameter**.

In addition, we used two criteria for evaluating the performance of our models: the test accuracy and the confusion matrix (see the general confusion matrix table in the Annex section). The results show a pretty good performance globally with the two strategies considered for the four algorithms. Because of the soundness of the hypotheses of the second strategy, we will put forward in the comparative analysis with the trading dataset. The model based on random forest satisfies us the most in this framework. It is easily interpretable, accurate and detects well the presence of the minority class in the test set.

As simple as logistic regression can be in our minds, it showed outstanding performance compared to other algorithms with higher capacity. The random forest algorithm achieves the same performance as logistic regression, but it has the advantage of easy interpretation. Despite our expectations and hype that neural networks would perform greatly, it turns out if we add its complexity we are inclined to favor simpler algorithms. Concerning the SVM algorithm, we are mitigated because it reacts quite the same with the neural network algorithm in terms of accuracy but performs a bit worse than neural network, even, it is very reliable in terms of the second strategy. The discriminative factor in our choosing the more suitable method to target good prospect for the bank is the capacity to remain relevant knowing the bank would like to use an algorithm strong and robust enough to last a reasonable period of time. In that sense, we would recommend the algorithm that has been trained taking account of the unbalanced data. We would suggest then to pick the Random forest and its learned parameters with the second strategy to achieve our mining predictions.

5 Comparative analysis of the algorithms performance over the two different learning problems

Table 3: Comparison of best model test accuracy for both datasets

	Strategy 1	Strategy 2
Logistic Regression	60.4%	89.9%
SVM	60.5%	89.7%
Random Forest	60.5%	89.7%
Neural Network	71.0%	89.8%

The first and most general observation that can be made is that all the algorithms performed better on the banking dataset. The accuracy on the test set is approximately 90% for all of the models. The features used to predict whether the clients would subscribe to a term deposit allow the algorithms to learn well enough to generalize with very high accuracy. On the other hand, the trading problem provides a much smaller accuracy on the test set but it is expected[1] to be lower considering the problem at hand. As the market opens every day, we can randomly attempt to predict the market direction and anything that provides a better chance than a random walk is worth taking a look into.

Generally speaking, simple algorithms such as logistic regression and SVM do a very good job on binary and 3-class classification and seem to be the best compromise between good accuracy and computational time. Hence, for simple problems as the ones treated here, ensemble methods and deep learning algorithms do not outperform more traditional methods and are not necessary.

An interesting point to discuss though, is the use of neural networks. A surprising intermediate result observed in both problems is that sometimes, the neural network only predicts one class (banking problem, trading problem) and 2 classes over 3 (trading problems). To counter this problem, the solution brought in the first case was preprocessing the data (standardization) and providing weights to under-represented classes; and constructing large neural networks in the latter.

6 To go beyond...

While providing interesting results, this project allowed us to notice a few interesting ideas that could be developed.

[1] efficiency theory of the financial markets

As far as the trading problem is concerned, one could reformulate the classification problem into a regression problem by using as a target, the intermediate quantitative variable that allowed us to construct the output used in the present project. By twisting this point of view, a good track to be explored would be the use of RNN's and LSTM's to predict the daily price change. Indeed, these are powerful types of neural networks designed to handle dependencies in a sequence and that can bring a new dimension in taking into account long term dependencies in the data. This strategy would develop and complexify the sliding window strategy that aimed to focus on recent observations by filtering information that is worth to be remembered through time.

Concerning the bank marketing problem, we could think of programming a robot to make the calls and learn instantly during the live interaction with the client what formulation or approach could lead with a high probability to a favorable outcome. The robot could be more prepared to handle multiple types of clients, adjust its parameters constantly with regards to how the interaction evolves and answer right back. This type of project is a field of deep learning techniques, specially built upon NLP algorithms and is still developing, however it is possible to simply try to improve accuracy by boosting our current algorithms and train the commercial agents to better utilize the resources provided by the Machine Learning. Not limiting the perspective to the bank marketing for term deposits, more generally, this reasoning can be applied to innovative products for financial institutions according to the market tendencies, the economic welfare and client responsiveness.

Annex

Here is an overview of how the features were generated:

(i) Exponential moving averages (EMA10, EMA25) (%): The exponential moving average aims to give a smoothed idea of the price of the ETF given a time period (10 and 25 days in our case) while giving more weight to recent data. The EMA's of the previous n days were calculated and then compared to the opening price of the ETF from which we calculated a difference in relative terms.

(ii) Oil, Gold, Currencies (%): We extracted from the previous day closing price and the current day market open price the relative difference in values of these financial assets using their values in the futures market. There are gages of how overnight markets moved.

(iii) Federal Funds rate (USA), Base rate (UK), Deposit rate (EU) (%): The current day rates.

(iv) T10-T2 treasury yields (%): The difference between the 10-year US treasury yields and 2-year US treasury yields. It can be used as a gage for estimating inflation.

References