Machine Learning for finance

DEEP LEARNING WITH LONG SHORT-TERM MEMORY NETWORKS

Written by:

Ethel Cedric NZATI
Georges Mbissane SARR
Comlan Rodrigue TOSSOU

Teachers :

Huyên PHAM
Jean-David FERMANIAN

May 1st, 2022

I Introduction

The development of machine learning application frameworks in the recent years has sparked the interest of the academic and the investment worlds, because those tools have enabled relatively faster computations, and some of the models they implement can take into account a variety of variables. In the main article, **1**, the authors conducted the implementation of the Long Short-Term Memory (LSTM) model which is inherently adapted for sequence learning, therefore adapted for time series learning. In order to get a benchmark for their model performance, they also implemented a Random Forest (RF) model along with a Deep Neural Network (DNN) and a Logistic Regression (LG). The S&P500 stocks close prices between 1st January 1990 and 31st December 2015 was fetched to build the data set.

II Methodology

II.1 Data management

The aim is to predict whether or not the daily return of a stock will go beyond the cross-sectional median of the available S&P500 stocks the next day. To be more specific, the authors of **1** and **2** have built the so called study periods. Each study period is equal to 1000 days and is composed of a training period, which is made of the first 750 days and a trading period, which is made of the last 250 days. The study periods are constructed such that the next study period is the forward translation of the current one by 250 days (so that the trading periods of two consecutive study periods do not overlap/intersect).

From January 1990 to the end of 2015, there are roughly $26 \times 250 = 6500$ days. So by translating forward blocks of 1000 days (i.e study periods) from the beginning of 1990 to the end of 2015, we get a number x of study periods given by:

$$1000 + (x - 1) \times 250 = 6500$$

That is $x = 23$ study periods. Let n_i denote the number of S&P500 constituents at the very last day of the training period in study period i, so n_i should be very close to 500.

The training set of study period i is composed of all n_i stocks with the history they have available. Some stocks exhibit a full 750 day training history, some only a subset of this time frame, for example, when they are listed at a later stage. The trading set of period i is also composed of all n_i stocks. If a constituent exhibits no price data after a certain day within the trading period, it is considered for trading up until that day.

Let $(P_t^s)_t$ designate the sequence of daily prices of stock s at time t, with s among the n_i stocks of period i, and $R_t^{m,s}$ the m-day(s) return of s defined as follows:

$$R_t^{m,s} = \frac{P_t^s}{P_{t-m}^s} - 1$$

The authors have set first $m = 1$ and have normalized the returns as follows:

$$\tilde{R}_t^{m,s} = \frac{R_t^{m,s} - \mu_{train}^m}{\sigma_{train}^m}$$

where μ^m_{train} and σ^m_{train} are respectively the average return and the standard deviation of returns obtained from the training set of the considered period.

On the one hand , normalizing the returns does not seem to be pertinent, because by definition returns are normalized, and generally stocks daily returns vary between -1 and 1: a stock return strictly lower than -1 means the stock price changes sign from a day to the next, and a return greater than 1 means the price doubles from a day to the next, those scenarios seem very unlikely. But on the other hand, the stocks are linked to companies belonging to different sectors related to different risks, so normalizing is pertinent in that regard.

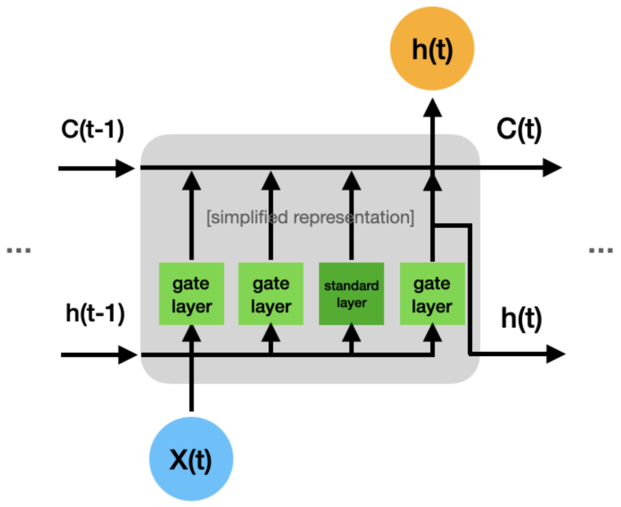
The target Y^s_{t+1} for a stock s at date t takes value 1 if the (normalized) return $\tilde{R}^{m,s}_{t+1}$ at date $t+1$ of stock s is greater than the median of the (normalized) returns of all the stocks at date $t+1$, 0 otherwise.

II.2 The different models

✻ Long Short-Term Memory (LSTM) networks :

LSTM is a special type of Recurrent Neural Networks (RNN), designed to handle such long range dependencies much better than standard RNN. Similar to RNN, LSTM detect the sequential patterns in the data. Informations are passed through the network with cycles, i.e., the information is transmitted back into the model. But in particular, LSTM has 3 additional fully-connected layers in each repeating block compared to the one of the standard RNN. These additional layers are also called gate layers because of their different activation (sigmoid) compared to the standard layer (tanh). As the information passes through each LSTM cell, the cell state, $C(t)$, can be updated by adding or removing information via the gate layers. This is how the LSTM model decides whether to retain or drop information from previous time steps.

Figure 1: Simplified LSTM at time t



Source : Multivariate Time Series Forecasting with Deep Learning

The specified topology of the trained LSTM network is as follows:

- Input layer with one engineered feature and 240 timesteps

- LSTM layer with 50 hidden neurons.
- A sigmoid activation function for the single neuron indicating the Bernoulli outputs whether the stock outperforms the market

For benchmarking purpose, the authors choose **random forests** , a standard **deep net** (to show the advantage of the LSTM), and **logistic regression** (a standard classifier as baseline). We add **XGBoost Classifier** (as an alternative training algorithm of random forests) and **LightGBM** (as an efficient learning method).

✻ **Logistic Regression (LOG) :**

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. Logistic Regression is used when the dependent variable (target) is categorical. In other words, the logistic regression model estimates the probability $\mathbb{P}(Y = 1|X = x)$ for a given input x as follows

$$\mathbb{P}(Y = 1|X = x) = \frac{1}{1 + e^{-(\beta_0 + x'\beta)}}$$

Logistic Regression is one of the most popular ways to fit models for categorical data, especially for binary response data in data modeling. It is the most important (and probably most used) member of a class of models called generalized linear models. Unlike linear regression, logistic regression can directly predict probabilities; furthermore, these probabilities are well-calibrated when compared to the probabilities predicted by some other classifiers, such as Naive Bayes.

✻ **Random Forest (RF) :**

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees trained on randomly selected subset of training set and features set. It aggregates the votes from different decision trees to decide the final class of an input.

✻ **XGBoost (XBG) and LightGBM (LGB) Classifiers :**

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

✻ **Deep Neural Networks (DNN) :**

The multilayer perceptron (often simply called neural network, multilayer perception (MLP)) is perhaps the most popular network architecture in use today both for classification and regression. The MLP is given as follows:

$$\hat{y} = v_0 + \sum_{j=1}^{NH} v_j g(w_j^T x')$$

The MLP is a heavily parametrized model, and by selecting the number of hidden nodes NH we can control the complexity of the model. The breakthrough that lent credence to the capability of neural networks is the universal approximation property. Under certain mild conditions on the hidden node functions g, any given continuous function on a compact set can be approximated as close as arbitrarily given using a network with a finite number of hidden nodes. While this is a reassuring result, it is critical to avoid overparametrization, especially in forecasting applications which typically have a limited amount of highly noisy data. To obtain the weights a customized loss (for classification tasks binary crossentropy is used in general) is defined, and the weights are optimized using gradient techniques. The most well-known method, based on the steepest descent concept, is the backpropagation algorithm.

II.3 Evaluation metrics

One common method for determining the performance of a classifier is through the use of a confusion matrix. A confusion matrix is a table that categorizes predictions according to whether they match the actual value. In a confusion matrix, TN is the number of negative instances correctly classified (True Negatives), FP is the number of negative instances incorrectly classified as positive (False Positive), FN is the number of positive instances incorrectly classified as negative (False Negatives), and TP is the number of positive instances correctly classified as positive (True Positives). The table below show this confusion matrix.

Table 1: Confuxion Matrix

Confuxion Matrix	**Predicted**	
Actual	**Positive (1)**	**Negative (0)**
Positive (1)	TP	FN
Negative (0)	FP	TN

❶ **Accuracy** : From the confusion matrix, many standard evaluation metrics can be defined. Traditionally, accuracy is the most popular performance metric in a binary classification problem. It is simply a ratio of correctly predicted observation to the total observations :

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

❷ **Precision - Recall measure** : Their are computed from the confusion matrix as :

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the positive class is correctly recognized (a small number of FN).

❸ **AUC Measure - ROC Curves** : A receiver operating characteristic (ROC) curve is a graph representing the performance of a classification model for all classification thresholds. This curve plots the rate of true positives as a function of the rate of false positives. The area under curve (AUC) measure determines the area under receiver operating characteristic (ROC) curve. One way of computing AUROC is, given n_0 points of class 0, n_1 points of class 1, and S_0 as the sum of ranks of class 0 examples :

$$AUC = \frac{S_0 - n_0(n_0+1)}{2n_0n_1}$$

❹ **F_β-Measure** : A common metric is the F_β-Measure. F_β-Measure is a family of metrics that attempts to measure the trade-offs between precision and recall by outputting a single value that reflects the goodness of a classifier in the presence of rare classes. While ROC curves represent the trade-off between different TPRs and FPRs, F_β-Measure represents the trade-off among different values of TP, FP, and FN. The formula for F_β-Measure (where β represents the relative importance of precision and recall) is:

$$F_\beta = (1+\beta^2) \times \frac{Precision \times Recall}{Precision + Recall}$$

In this study, we will compute F_1.

II.4 Forecasting, ranking, and trading

For all models, the probability $\tilde{P}^s_{t+1|t}$ is calculated for each stock s to outperform or underperform the cross-sectional median in period $t+1$, using only information up to period t. Then, they rank all stocks for each period $t+1$ in descending order of this probability. The top of the ranking corresponds to the most undervalued stocks that are expected to outperform the cross-sectional median in $t+1$. As a result, to build a long-short portfolio consisting of $2k$ stocks, go long the top k and short the flop k stocks of each ranking.

III Results

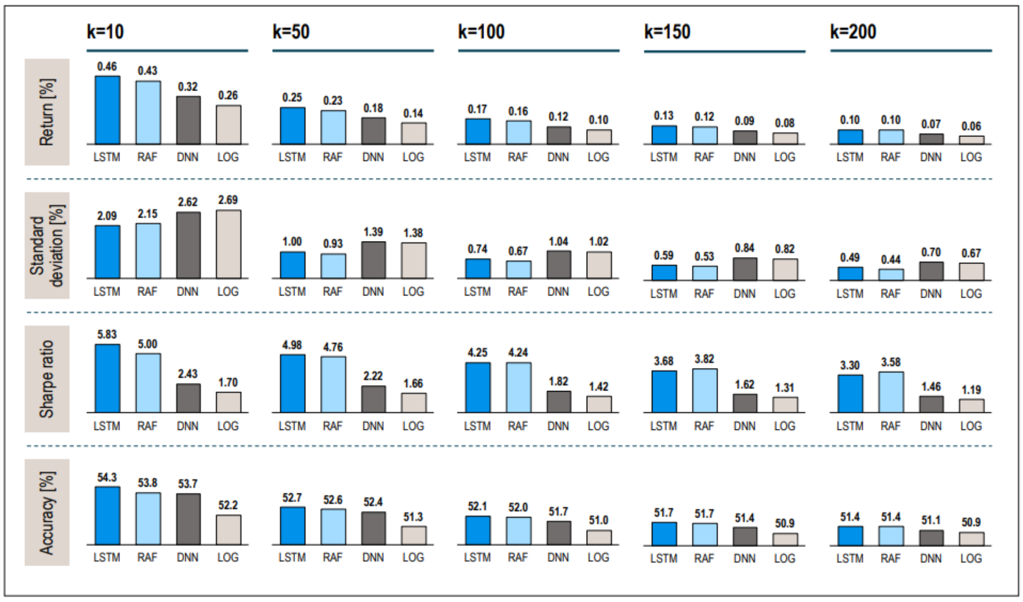
The authors present their results in three steps:

- comparison of the performance of the LSTM network to that of the random forest, the deep neural network and logistic regression (analysis of the performance before and after applying a transaction cost of 5 bps per half turn);
- deduction of common patterns in the high and low scoring stocks, thus revealing the sources of profitability;
- development of a simplified trading strategy based on these findings

III.1 Models performance

For different $2k$ stocks portfolio, the authors constructed the graph below which compares the performance of the LSTM with the other approaches along some dimensions.

Figure 2: Daily performance characteristics for long-short portfolios of different sizes



Source : *Deep learning with long short-term memory networks for financial market predictions*, T. Fischer, C. Krauss (2017)

Irrespective of the portfolio size k, the LSTM shows favorable characteristics vis-a-vis the other approaches. The authors focus their subsequent analyses on the long-short portfolio with $k = 10$ because the graph shows a clear advantage of the LSTM for this value.

In order to benchmark the predictive accuracy of the LSTM forecasts against those of the other methods, they deploy:

- the Diebold-Mariano test (1995) to evaluate the null that the forecasts of method i have inferior accuracy than the forecasts of method j, with $i, j \in \{LSTM, RAF, DNN, LOG\}$ and $i \neq j$
- the Pesaran-Timmermann (PT) test to evaluate the null hypotheses that prediction and response are independently distributed for each of the forecasting methods.

The results of these test are presented in the table below and show that:

- DM tests individual null hypotheses that the LSTM forecasts are less accurate than the RAF, DNN, or LOG forecasts are rejected at a five percent significance;
- PT tests null hypothesis can be rejected at any sensible level of significance.

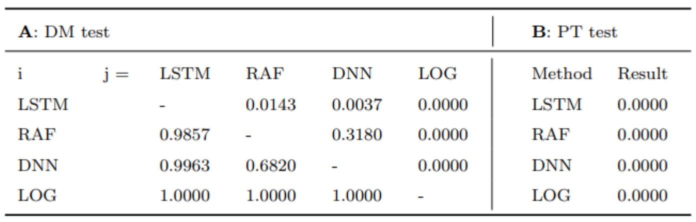
IV Our experiments

IV.1 Data collection and management

The first step was to fetch the price of SP500 stocks from 1900 to 2015. To do so Wikipedia[1] keeps track of the changes that occurred in the composition of the index. The list provided was used to build approximatively the composition of the index over time.

[1]https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

Figure 3: Diebold-Mariano and Pesaran-Timmermann tests

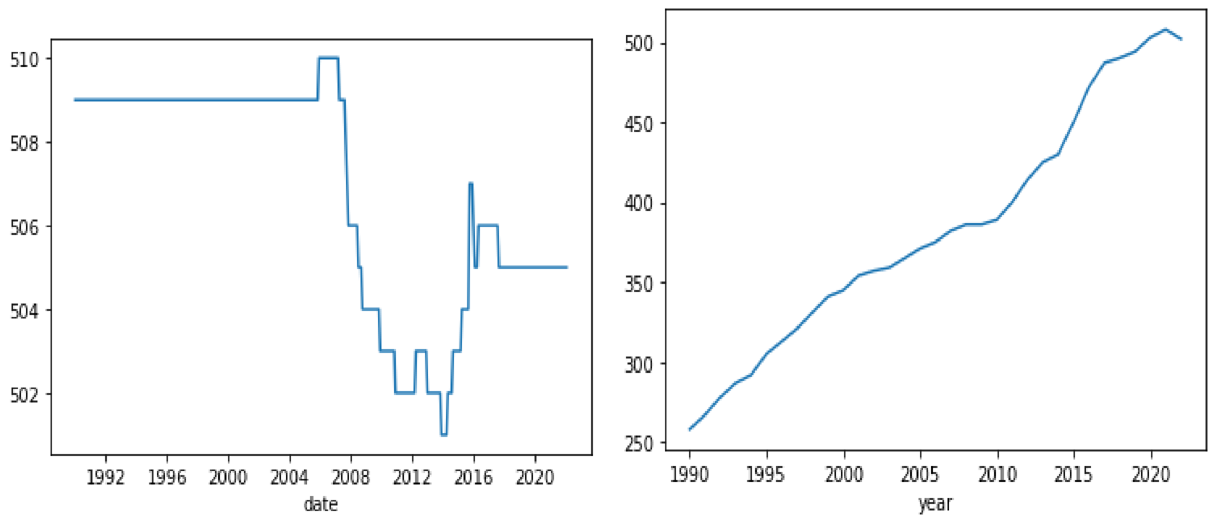


A: DM test						**B**: PT test	
i	j =	LSTM	RAF	DNN	LOG	Method	Result
LSTM		-	0.0143	0.0037	0.0000	LSTM	0.0000
RAF		0.9857	-	0.3180	0.0000	RAF	0.0000
DNN		0.9963	0.6820	-	0.0000	DNN	0.0000
LOG		1.0000	1.0000	1.0000	-	LOG	0.0000

Source : *Deep learning with long short-term memory networks for financial market predictions*

Figure 4: S&P500 size over time using Wikipedia information versus the data collected

(a) Number of S&P500 constituents from Wikipedia (b) Number of S&P500 constituents in our data



Source : Our work.

The python package pandas-datareader was used to fetch the ajusted close price of the stocks making up the index. One can notice on Figure **4** that roughly half the constituents are missing in our dataset.

After year 2000 the rate is, as expected, around 50%. However before this year, the target rate is below 50%. This is due to the fact that some daily normalized returns are equal to their cross-sectional median, so the corresponding stocks are labelled 0.

IV.2 Modelling

We experimented the different models presented in the article along with models that were developed after the release of the paper: xgboost[2] (XGB) and lightGBM[3] (LGB). The models are trained and validated on the training set[4]. Predictions are done as follows: predict 1 if $\tilde{P}^s_{t+1|t} > 0.5$, 0 otherwise.

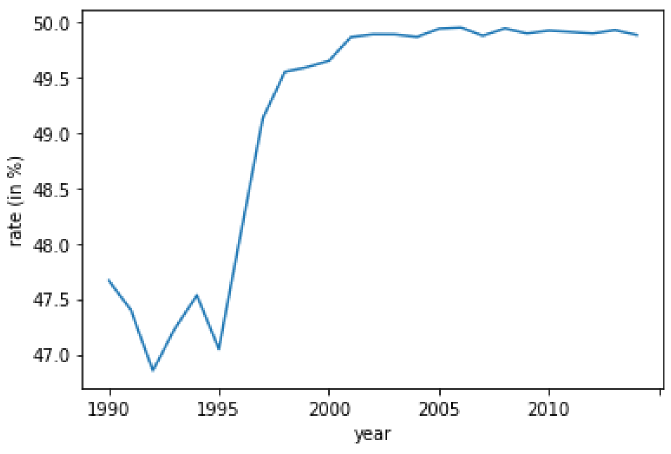
We first focused on the global metrics of the different models in other to evaluate how much they perform when it comes to predicting whether or not a stock return will go over the cross sectional

[2]with 50 estimators and a max depth of 4

[3]with 100 estimators and a max depth of 4

[4]For each period, the 20000 last data of the training set are used for validation and the rest for training. Trading is conducted using the model trained on the training set.

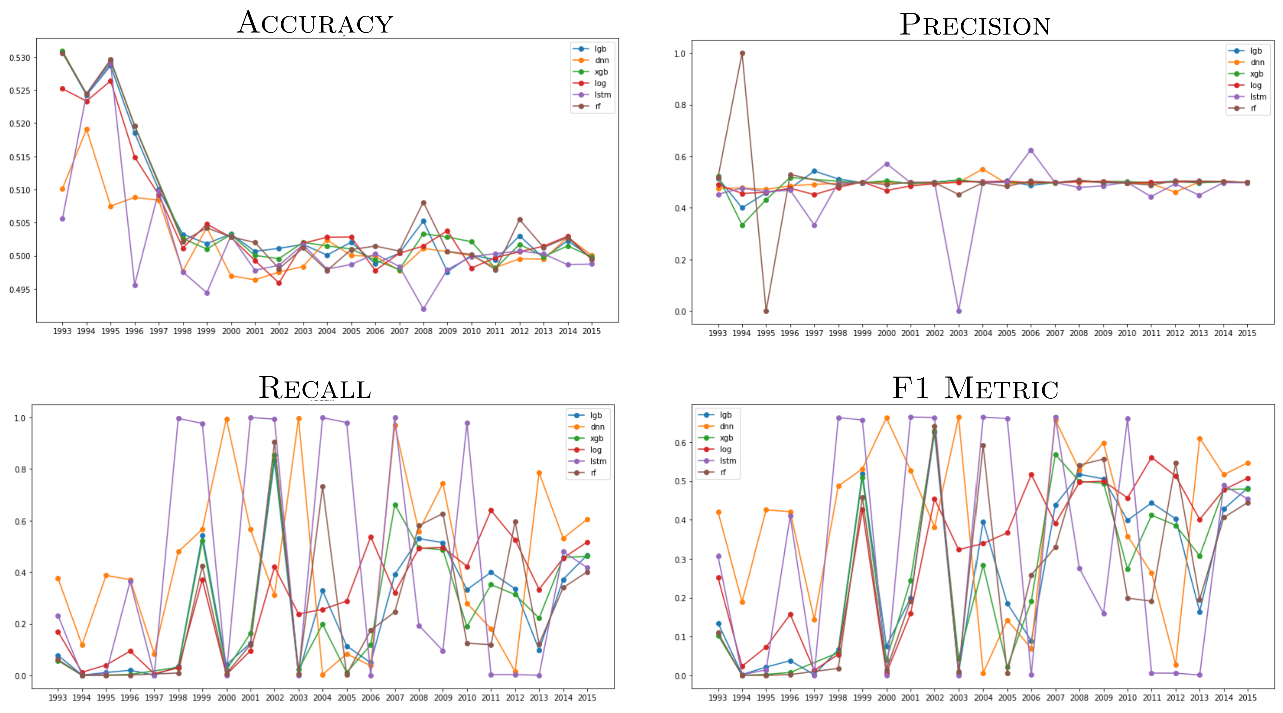
Figure 5: Target rate over time



Source : Our work.

median return. On figure 6, DNN and LSTM perform better in terms of recall and F_1 score, but worst in terms of accuracy.

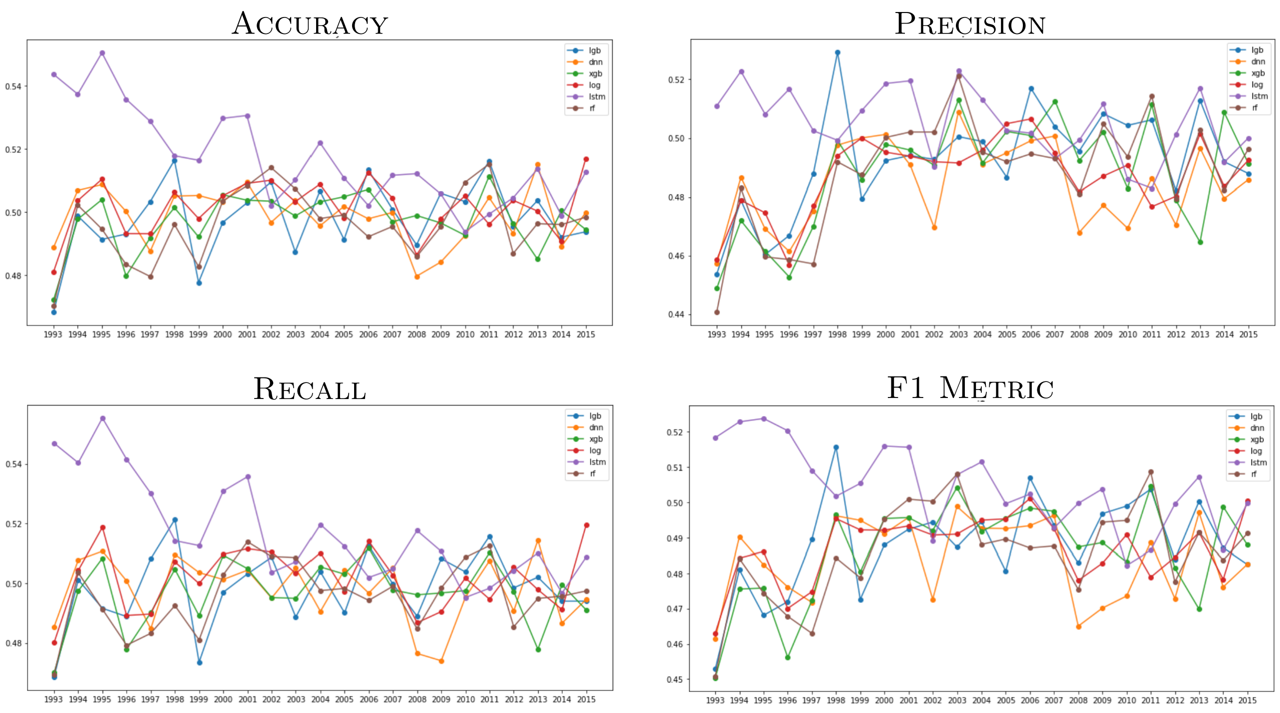
Figure 6: Global metrics



Source : Our work

However, these global metrics are not very important regarding the constitution of portfolios with reasonable size. In fact the most important criteria is the adequate prediction of the top stocks and the flop stocks, that is why we also computed the same metrics limiting ourselves only on the k top and k flop stocks, as the authors did. On figure 7 are presented the results. The LSTM model seems to outperform the rest and, unexpectedly, logistic regression performs reasonably well for $k = 10$.

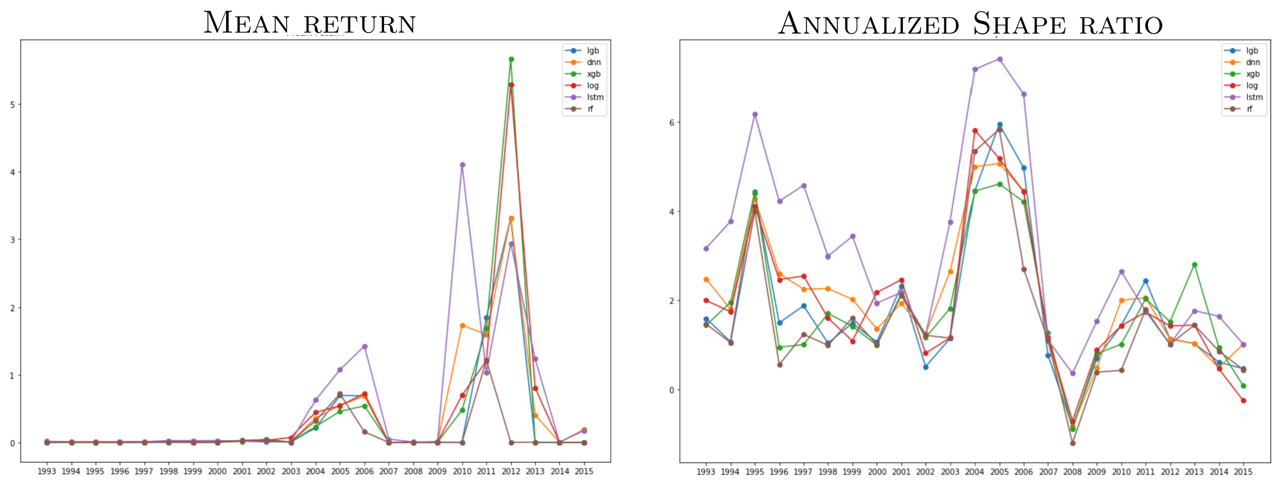
Figure 7: Model metrics for $k = 10$ portfolio



Source : Our work

Regarding the daily mean return and annualized Sharpe ratio, LSTM looks better than the rest.

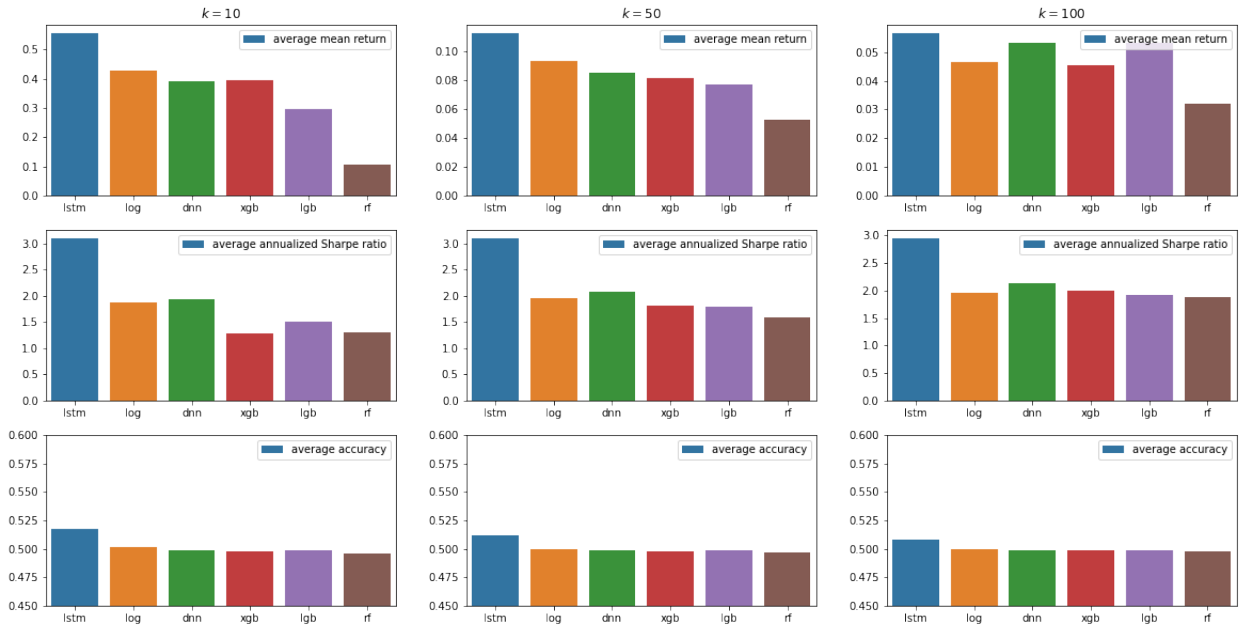
Figure 8: Portfolio of size $2k = 20$ performance



Source : Our work

Finally, to see the portfolio and model performances with respect to k we present the average mean return, the average annualized Sharpe ratio and the average accuracy of each model. Again LSTM seems to outperform the rest with an average accuracy of 0.52. In order to see if the performance LSTM is achieve by pure luck, consider the accuracy a_{random} of such random classifier which predicts 1 with probability 0.5 and 0 with probability 0.5 for any input x, it satisfies $na_{random} \sim \mathcal{B}inomial(n, 0.5)$, where $n = 102000$ is the trading size for year 2015. We want to compute $\mathbb{P}[a_{random} \geq 0.52]$. There is a closed form for such a probability involving binomial coefficients, but since n is too large the exact computation is difficult, instead by means of the Central Limit Theorem, we can approximate the law of a_{random} by $\mathcal{N}(0.5, 0.25/n)$. This probability is estimated at 1.13×10^{-37}, meaning that it is very unlikely that LSTM performance is pure luck.

Figure 9: Average mean return, average annualized Sharpe ratio and average accuracy for different values of portfolio size $2k$ from 1993 to 201



Source : Our work.

V Conclusion and critics

Our results show that LSTM can more efficiently capture temporal dependencies than usual machine learning algorithms. For financial market predictions it can be more suitable.

However, some study decisions made by the authors can be discussed. In particular, the normalisation made over the portfolio to include stock inter-dependencies can be improved by including into the model scheme the space correlation. To achieve that goal, we can test in further studies the combination of LSTM and Convolutionnal Neural Networks (CNN). Finally, we can take in account more features like the sector or macroeconomic indicators to improve the performance of our model.

Bibliography

[1] T. Fischer, C. Krauss (2017) : Deep learning with long short-term memory networks for financial market predictions, Ideas:886576210

[2] Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 259(2), 689–702.