Prediction of daily stock movements on the US market
CFM Challenge

Reda Ouhamma, Abdessamad Ed-dahmouni

February 2019

1 Introduction

Prediction of stock returns is of paramount importance in the investment domain. For the course **L'apprentissage par réseaux de neurones profonds**, we chose to work on the challenge proposed by CFM: **Prediction of daily stock movements on the US market**, more concretely, we have the returns between 9h30 and 15h20 over 5 minute windows and we need to predict the sign of the return from 15h20 to 16h. A major issue in handling this challenge is the fact that stock movements are practically random. In this report we present our approach to get to the 19th place over 45 in the public leaderboard.

2 Data analysis

A first glance at the data at hand tells us that we have 680 stocks for 1511 days in the training set and 648 days for the test set. Each instance of the data we have a unique identifier, an equity id, a date hash and the returns across the day.
In conclusion, we have a 596261 sample for training and 149066 for testing. A minor problem we noticed is that the dates are hashed so we cannot access the real order on the samples. However, the dates in the train and test datasets are distinct and all the stocks present in the test set are available in the training phase which is why we chose to sort the training by date set in order to separate it to a training and validations subsets in the same form as the original data (distinct dates with the same stocks).

Before we start analyzing any correlations or patterns in the data we should visualize the distribution of missing values:

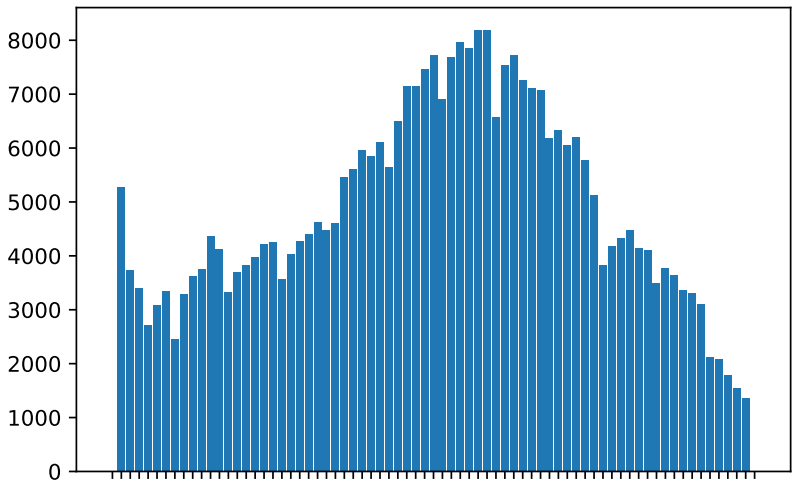

Figure 1: Missing values Distribution across features

The latter practically represents around 0.6% of the data which is not very significant, however, we did try to innovate in dealing with these missing values by using different techniques such as interpolation but they all seemed to yield the same results so we stuck with replacing by zero (it's also more logical since an unobserved price is best approximated by its last known value).

2.1 Feature distributions

The first and most straightforward property to review is the distribution of each attribute. Since we have 71 time feature for each day, we consider only the ones which are near the close to be able to visualize some insights. A simple boxplot shows:

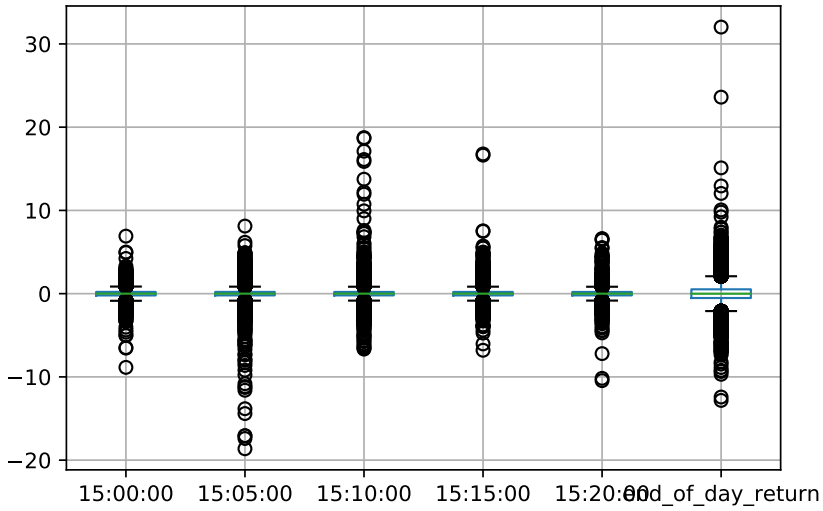

Figure 2: Feature Distributions

We can see that all attributes have a lot of outliers (samples located outside the box). We can even say that these attributes seem to have a relatively normal distribution. We can also look at the distribution of each attribute by discretizing the values into buckets and confirm normality by reviewing the frequency in each bucket as histograms:

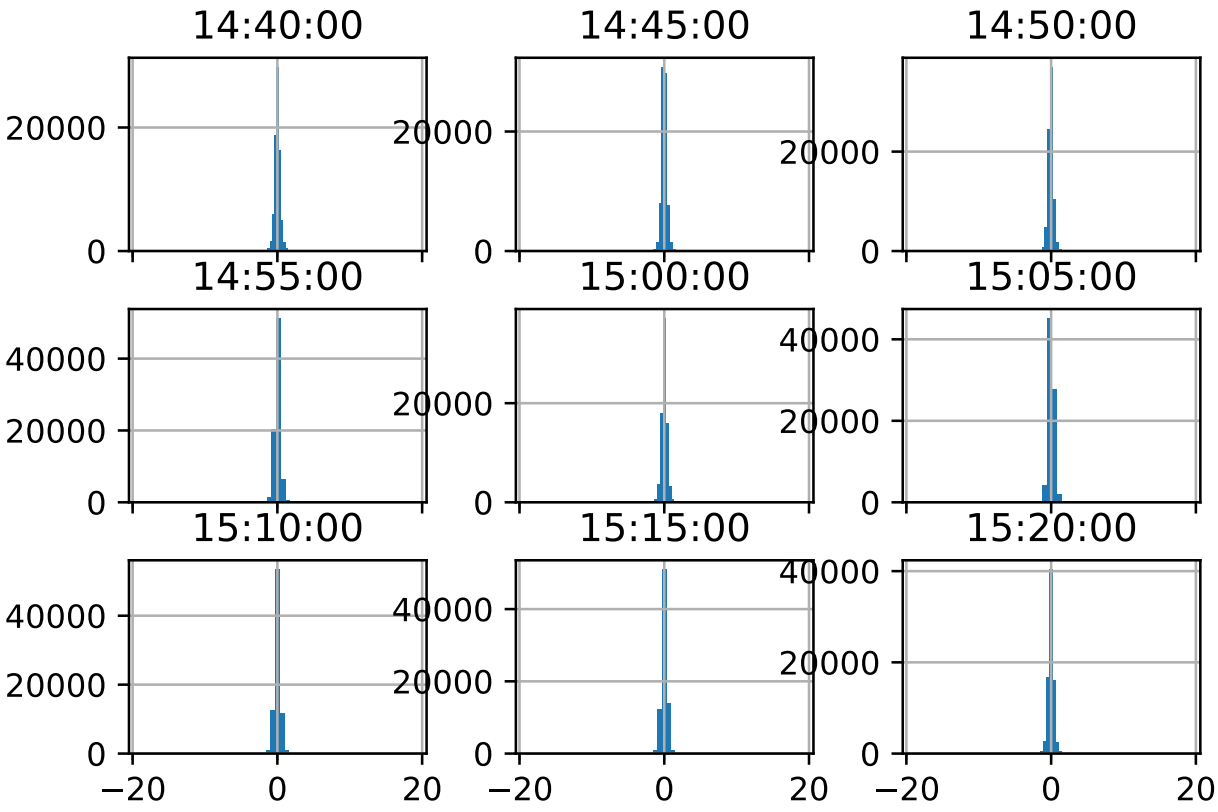

Figure 3: Features Histograms

2.2 Autocorrelations

Autocorrelations are the correlations of a signal with a lagged version of itself. They are very handy for detecting periodic patterns in signals. However, in our case, for equity stocks they are quite meaningless as we can see:

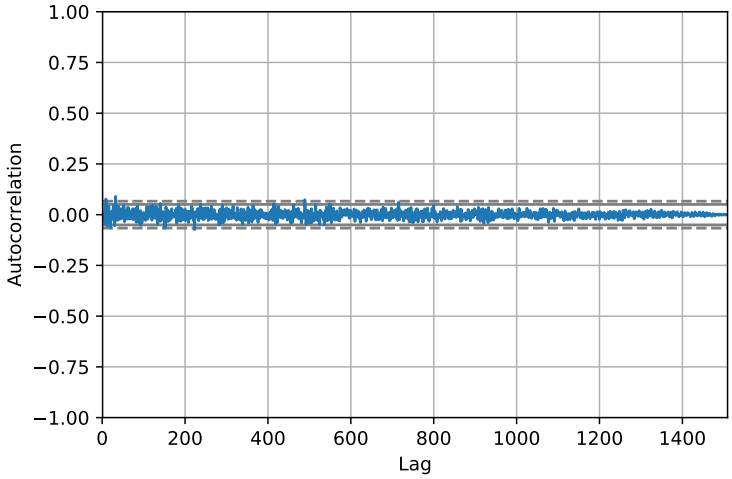

Figure 4: Autocorrelations for a random equity

2.3 Correlations

Another important relationship to explore is that of the relationships between the attributes. We can review the relationships between attributes by looking at the distribution of the interactions of each pair of attributes, we chose to do so for the first 4 measurements of a random stock and the results are as follows:

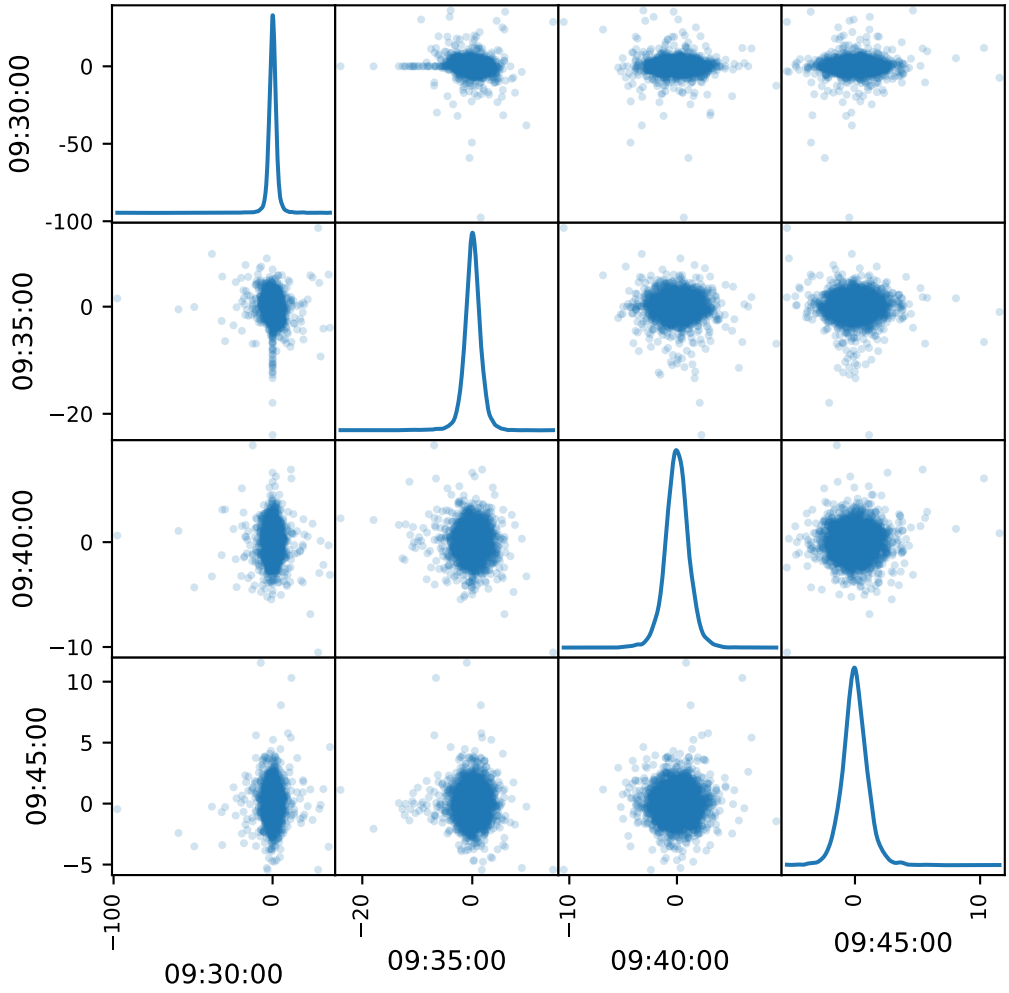

Figure 5: scatter plots of four attributes versus four attributes for a random stock

This is a powerful plot from which a lot of inspiration about the data can be drawn. However, one again the data being equity stock returns, no correlations seem to appear between the features.

3 Feature engineering

Given the financial nature of our problem, and as stated by challenge provider, every single one of the many available features should be used to its maximum in order to capture all the patterns we can find to predict the sign of the return between 15h25 and 16h. For that, we can safely consider that every return during the day provides some information about the last one and about the general evolution of the returns.

3.1 Features by stock

The data provides multiple features about the different returns during the day and the date and the equity id, we chose to start by manually constructing some intuitive features:

- Mean: mean over equity of the sum of all returns during the day

- Median: median over equity of the sum of all returns, seems to enhance the results even though it's very close to the mean.

- Std: standard deviation over equity of the mean of all returns.

- Scaled: a proposed scaling of the variable sum of all returns: $new\_feat = \frac{x - mean(x)}{std(x)}$

We have constructed four new features having then an intermediate dataset of 75 features (excluding the equity code and the date).

3.2 Time series features

Similar to the the features we designed to characterize each stock, we designed a vector of features that varies over time just as the returns. this vector contains:
• The returns themselves;
• Their exponentially weighted moving average with a halflife of 1.5 ($2\times$ halflife = 15 minutes)
• Rolling minimum and maximum over 6 returns $\{t-5, \dots t\}$;
• Rolling std over 12 returns $\{t-11, \dots t\}$;
• Rolling median over 12 returns $\{t-11, \dots t\}$;

$$X_t = \begin{bmatrix} r_t \\ \hat{\mu}_t \\ p_t \\ b_t \\ \sigma_t \\ \hat{q}_t \end{bmatrix}$$

Where p_t is the peak over n past periods, and b_t is the bottom over the same past periods. $\hat{\mu}_t$ is a trend estimator (an exponentially weighted average can be interpreted as a Kalman filtering in the stationary case when the Kalman grain stops varying $t \to \infty$) We tried adding the skewness and kurtosis to both features by stock and time series features, but it didn't improve any of our models. We tried feeding this time series per sample is fed to

4 Linear model

The first model we tested and the most simple one is ironically our best score. A simple linear regression with the original features and the handcrafted features by stocks. This model seems to achieve about 52.42% on the training/validation subsets but also on the public test data, this behaviour was expected since such a direct model doesn't overfitt. We also tried multiple variants with regularizations but the unregulzarized model yields the best accuracy.

classification $\iff$ regression The original formulation of the problem is to predict the sign of stock returns over the last period of the day. Thus, it is naturally a classification problem. However, one can easily transform it into a regression problem in order to have more possibilities for the choice of our algorithms. This transformation is what (on our valid set) yielded the best accuracy 52.612% but to our deception it only translated to 52.39% on the public leaderboard. However, we are convinced that this is the way to go since by choosing regression we give our model more information than just the sign of the return but this comes with the price of solving the harder task of regression.

5 Neural networks

5.1 A feed forward neural network

Good old fashioned percpetrons can be a good method to use for predicting stock movement. Over short windows, there might exist some meaningful statistical properties allowing us to predict the final return, and feed forward

neural networks can find these local patterns and fit them very well (FFNNs have great expressivity). When experimenting with these networks, we chose a simple structure consisting of two hidden layers, with dropout following each one, they don't seem to overfitt but their results aren't as good as expected as they only seem to reach $\sim 52.25\%$.

5.2 LSTM

Long Short-Term Memory (LSTM) units are among the most widely used models in Deep Learning for natural language processing today. LSTMs (along Gated Recurrent Units) are designed to combat the vanishing gradient problem that prevents standard RNNs from learning long-term dependencies through a gating mechanism. And for that they are naturally adapted to sequence prediction problems.

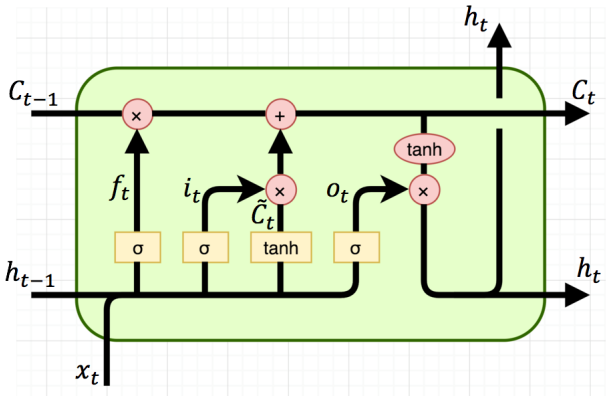

Figure 6: Long Short-Term Memory model[1]

$$
\begin{aligned}
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i\right) \\
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f\right) \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o\right) \\
\tilde{C}_t &= \tanh\left(x_t U^g + h_{t-1} W^g\right) \\
C_t &= \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right) \\
h_t &= \tanh(C_t) * o_t
\end{aligned}
$$

Here, i is the input, f is the forget gate, and o is the output gate. They have the same equations, just with different weight matrices (W is the recurrent connection at the previous hidden layer and current hidden layer, U is the weight matrix connecting the inputs to the current hidden layer). They care called gates because the sigmoid function maps the values of these vectors to the $[0, 1]$ interval, and by multiplying them element-wise with another vector, we define how much of this other vector goes through to the next time step.

The neural network we designed takes both the times series vector X_t and the features by stock as inputs, the first input goes through an LSTM unit with 144 hidden variables, and the output is fed to a dense layer of 24 variables, with no activation function. The result of this [LSTM, dense layer] combination is then combined with the features by stock, and fed to a feed forward network with to dense layers of 15 hidden variables, each with a ReLU activation function and a Dense(1) output layer.

We tried multiple loss functions:

- Mean squared error.
- Mean absolute error.
- Logistic loss, which we implement using Keras backend: $\ell(y, \hat{y}) = \frac{1}{\ln(2)} \ln(1 + e^{-y\hat{y}})$

Hinge loss with a threshold variable: $\ell_\rho(y, \hat{y}) = \max(\rho - y\hat{y}, 0)$
So far, the loss function that shows a steady improvement of accuracy on average is the mean absolute error. The logit loss didn't perform well, and neither did the hinge loss. Both losses try to penalize only the difference in sign of y and $\hat{y}$, which means that we are better off trying to accurately estimate the target y rather than its sign alone. I tried to find a convex loss function that is similar to the mean squared error, but is more severe when both y and $\hat{y}$ are close to 0, but failed to find a convenient one. (the reason I looked for such a function is to penalize more the distance between y and $\hat{y}$ when their are around 0, so that they tend to fall on the same side of either positive or negative values).

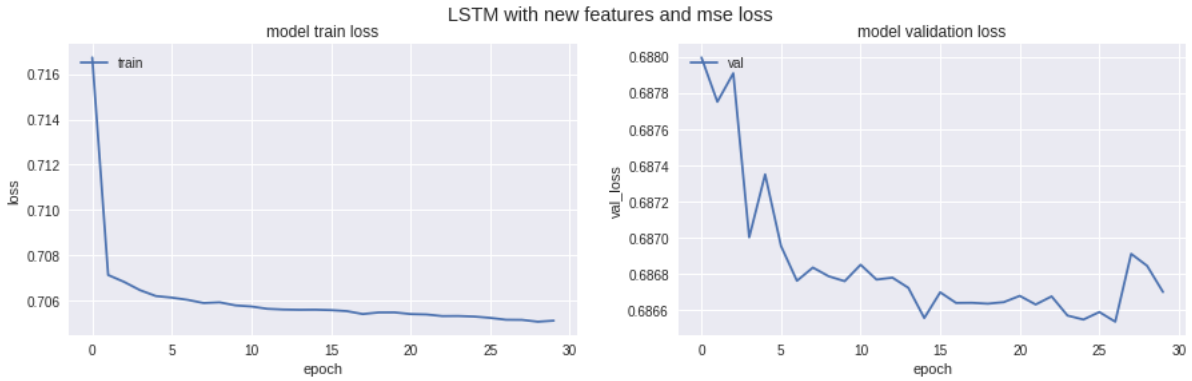

Figure 7: Training and validation MAE loss, with validation accuracy 0.52337

6 Conclusion

Critics The LSTM networks even if they seem the most adapted to our problem overfit a lot and take enormous time to train.

- Transforming the problem to a regression problem gives better overall accuracy

- Averaging / Smoothing : smoothing the returns signal by an exponential moving average amounts to running a Kalman filter on the long run, which aims at finding the true denoised trend behind the returns signal.
The optimality of Kalman filters among linear filters justifies our choice of an exponential moving average over a simple moving average.

Some possible enhancements are:

- More work on features by using some kind of convolutions like in image recognition

- Maybe construct a different model for each stock (or group of stocks based on similarity), find some way to include categorical features in NNs.

- Try other architectures for LSTMs

- Try data augmentation and bagging to avoid overfitting

Conclusion During this challenge, we had the possibility to apply what we learned during the **L'apprentissage par réseaux de neurones profonds** course on a concrete problem which isn't something we get to do all the time. We tried to construct some new features to better comprehend the problem and to build more consistent models. And we tried several NNs recipes to enhance our results (regularization, batch normalization, dropout, noise addition and so on).

In conclusion, this project was a good entry into the finance world and allowed us to use many skills learned during MVA and there's still a lot more possibilities to try to improve the results. This challenge was very exciting and gave us the chance to apply many interesting concepts.

7 References

[1] ISAAC CHANGHAU, LSTM and GRU - Formula Summary, link: https://isaacchanghau.github.io/post/lstm-gru-formula/

[1] Roondiwala, Murtaza Patel, Harshal Varma, Shraddha. (2017). Predicting Stock Prices Using LSTM. International Journal of Science and Research (IJSR). 6. 10.21275/ART20172755.