Report for data mining project

Jian Qian, Ryad Belhakem

March 16, 2018

1 Introduction

For the purpose of our project we will present a paper published by Stanford undergrad's who showed that ESN is quite prowerful[3]. The idea is not to use RNN which would be our first intuition due to difficulty of training (cf vanishing gradient), but to use some specific RNNs which have been developed to overpass this issue based on early stopping and forgeting data from time to time. The two main known RNNs of such purpose are ESN(Echo State Networks) and LSTM (Long Short-Term Memory) which are shown to be very efficient in modeling nonlinear dynamical systems[4]. In our project, we adopted the second type (LSTM) and tried to enrich it with more stock data and news papers, we then compare it with more "classical" measures like Kalman filter and GAM models.

1.1 Mathematical model

As usual in time series we try to predict $X_{t+1} = F(X_t)$ where X_{t+1} is the target to predict, the novelty here is that we tried to take advantage of news title to sharpen the prediction so our first guess was that, $X_{t+1} = F(X_t, y_t)$ where Y_t is the title we have even if in fact we showed that X_{t+1} is highly likely to be independent from y_t.

1.2 Source of the data

Our data is taken from old Kaggle contest, the format is simple we have daily prices of the stock market and the 25 biggest news title per day we have about 6 years of data point thus the feature matrix dimension is about a million.

1.3 Literature

As shown by Bernal et al[3]. Neural network are pretty efficient at predicting prices, in particular ESTN network seems to capture the structure of the data in such a way that it work with any company stocks and have a better result then the classical Kalman filter. In our project we aimed to achieve the same result using LSTM instead which have memory and maybe more adapted to the problem.

2 Replicating results with LSTM

2.1 LSTM

Long short-term memory(LSTM) units(or blocks) are a building unit for layers of a recurrent neural network(RNN). A RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer(or feed-forward) neural network: that is, they compute an activation(using an activation function) of a weighted sum. Intuitively, they can be thought as regulators of the flow of values that goes through the connections of the LSTM; hence the denotation "gate". There are connections between these gates and the cell. The expression long short-term refers to the fact that LSTM is a model for the short-term memory which can last for a long period of time. An LSTM is well-suited to classify, process and predict time series given time lags of unknown size and duration between important events. LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods in numerous applications[1]. The structure is as shown in Figure 1.

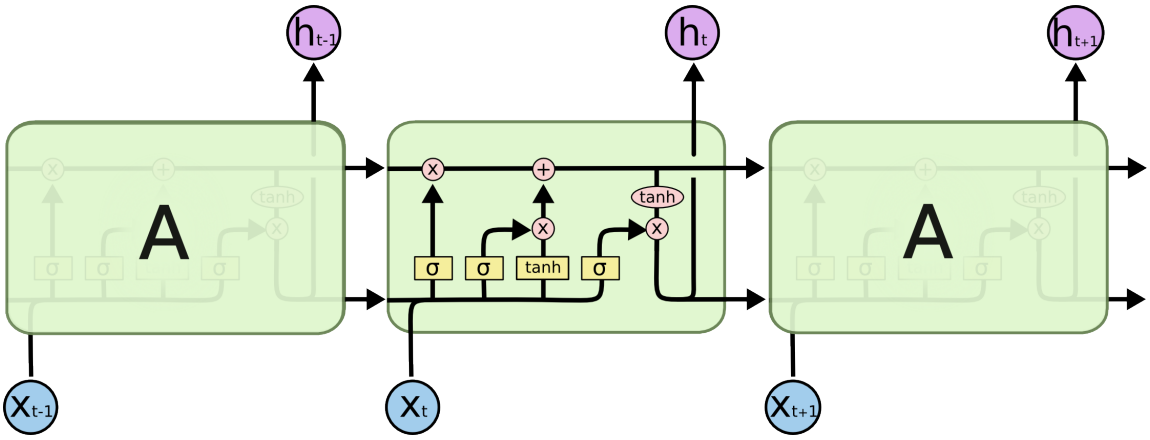

Figure 1: LSTM structure

The updating process is:

- $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
- $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
- $\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
- $C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t$

- $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
- $h_t = o_t \cdot \tanh(C_t)$

2.2 Comparison

The results we have are as good as theirs, as shown in Figure 2.

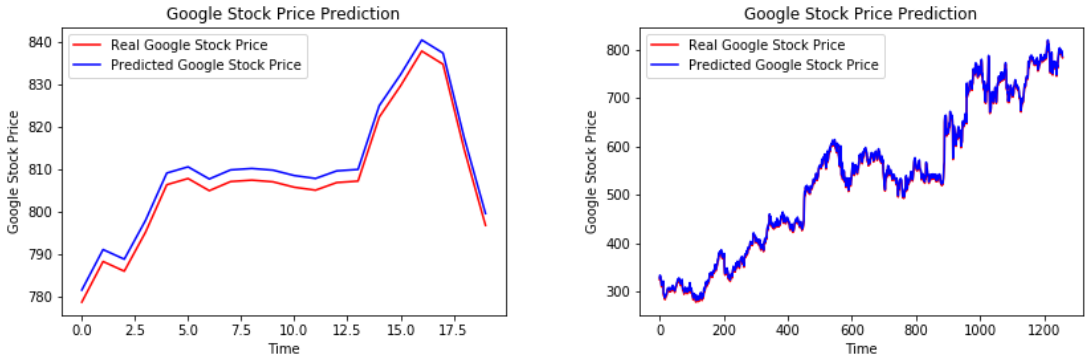

Figure 2: Reproducing the Results with LSTM

2.3 Conclusion

As we can see in the figure, our prediction is almost parallel to the real curve. Moreover, LSTM seems to work as good as ESTN.

3 Test the effect of the news

3.1 Methodology

We want to know if the news today can predict the changes of the stock market today, so we tried to develop a machine learning task where the input is the news and the target is to predict the price goes up or down today. There are three parts of this system, in the first part we transform the news into vectors that we can handle. The second part of the system is to try various methods with the problem. The third part is to aggregate all the methods to see it we can do better.

3.2 Natural Language Processing

To transform the news into a vector, we adopt the pre-trained word to vector model by Google on news[2].

Word to vec

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

The model we are using is trained on Google news which we think will be enough to capture the information in the news.

3.3 Mathematical model

- Data: (X_i, y_i)
  - X_i is the vector transformed from the news.
  - y_i is 1 if the stock price next day goes up, -1 otherwise.

3.4 Principles for different methods

Gaussian Naive Bayes

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. Gaussian Naive Bayes classifier assumes that the value of a particular feature is independent of the value of any other feature, given the class variable.

- Assuming the distribution is Gaussian:

$$p(X|y = y_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(X-\mu_k)^2}{2\sigma_k^2}}$$

- Finding the maximal likelihood for $(\mu_k, \sigma_k)_{k=1}^n$
  - Using the independence assumption, we can find μ_k, σ_k for each k specifically.
- Classify $f(X^{new}) = arg\max_{y_k} P(Y = y_k)P(X^{new}|Y = y_k)$

Decision tree and Random Forest

Decision tree

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete

set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

- Rules based on variables' values are selected to get the best split to differentiate observations based on the dependent variable
- Once a rule is selected and splits a node into two, the same process is applied to each "child" node (i.e. it is a recursive procedure)
- Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met. (Alternatively, the data are split as much as possible and then the tree is later pruned.)

Random forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

- Draw B resampled datasets from a single one using a uniform with replcement scheme (Bootstrap)
- For each resampled datasets, construct a tree using a different randomly drawn subset of variables at each split.

Results: As shown in Table 1.

Table 1: Random Forest with number of predictors vary from 6 to 21

Number of predictors	6	9	12	15	18	21
Accuracy	0.49	0.54	0.47	0.46	0.47	0.50

K-Nearest Neighbors

The k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification. The input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification.

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small).

- Using local average to approximate the value

$$\mathbb{E}[Y|X] \simeq \mathbb{E}[Y|X' \in \mathcal{N}(X)] \simeq \frac{1}{|\{X_i \in \mathcal{N}(X)\}|} \sum_{X_i \in \mathcal{N}(X)} Y_i$$

- Classify $f(X^{new}) = \begin{cases} 1 & \frac{1}{|\{X_i \in \mathcal{N}(X^{new})\}|} \sum_{X_i \in \mathcal{N}(X^{new})} Y_i > 0 \\ -1 & otherwise \end{cases}$

  Results: As shown in Table 2.

Table 2: K nearest neighbors with number of neighbors vary from 6 to 21

Number of neighbors	6	9	12	15	18	21
Accuracy	0.53	0.54	0.51	0.53	0.53	0.51

SVC

Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

- Minimize

$$\frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot X_i - b)) + \lambda ||w||^2$$

- Classify $f(X^{new}) = sign(w \cdot X^{new} - b)$

  Results: As shown in Table 3.

Table 3: Support vector machines with C vary from 3^{-2} to 3^3

C	3^{-2}	3^{-1}	3^0	3^1	3^2	3^3
Accuracy	0.51	0.51	0.50	0.5	0.51	0.51

Logistic Regression

The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor. The model is a direct probability model.

- Using assumption of the conditional probability:

$$\mathbb{P}(Y = 1|X, \beta) = \frac{\exp(X\beta)}{1 + \exp(X\beta)}$$

- Minimize the negative log-likelihood:

$$
\begin{aligned}
-\log \mathbb{P}(Y, X|\beta) &= -\sum_{Y_i=1} \log(\frac{e^{X_i\beta}}{e^{X_i\beta}+1}) - \sum_{Y_j=1} \log(\frac{1}{e^{X_j\beta}+1}) \\
&= \frac{1}{n}\sum_{i=1}^{n} \log(1+e^{-Y_i(X_i\beta)})
\end{aligned}
$$

AdaBoost

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

- Base estimator: Decision Tree
- Greedily minimize exponential loss:

$$\min_{(h,\alpha)} \sum_{i=1}^{n} e^{-y_i \sum_t \alpha_t h_t(X_i)}$$

- Classify: $f(X^{new}) = sign(\sum_t \alpha_t h_t(X^{new}))$

Neural Network

Artificial neural networks (ANNs) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" (i.e. progressively improve performance on) tasks by considering examples, generally without task-specific programming.

- Try to find $f(X, \theta^*) = h_k(g_k(...h_1(g_1(X, \theta_1), \theta_1')..., \theta_k), \theta_k')$

3.5 Results and comments

The results of the methods are shown in Table 4 and Figure 3. In the horizontal axis, it is the time step t, and the y-axis shows the accuracy up till time step t.

Table 4: Results for methods other than neural networks

Methods	Gaussian Naive Byes	Random Forest	KNN	SVC	Logistic Regression	Adaboost
Accuracy	0.48	0.53	0.54	0.51	0.48	0.50

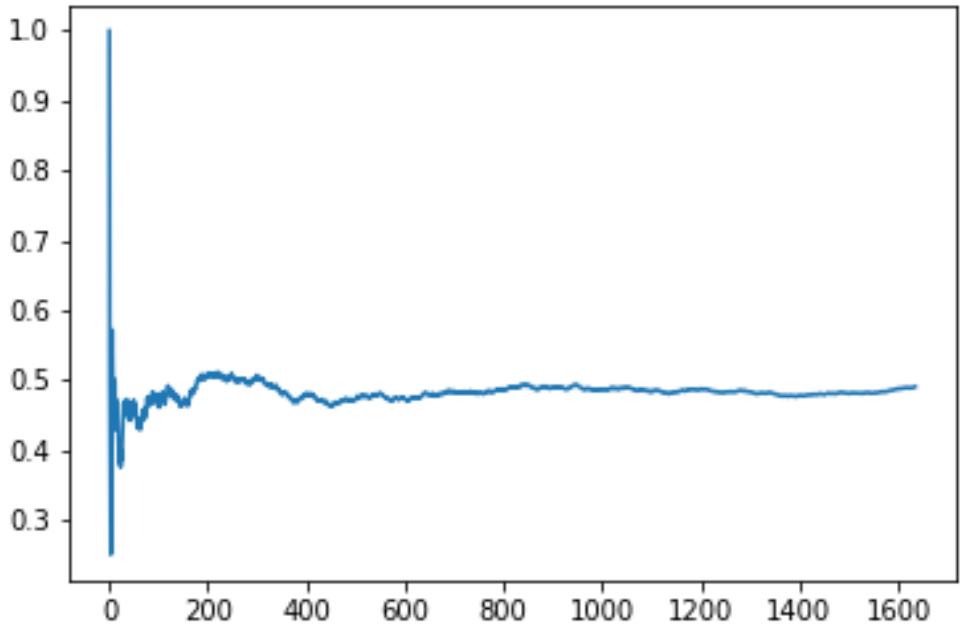

Figure 3: Result for neural networks

The figure shows that it is basically hard to have any prediction depending on the news.

4 Aggregation the results

4.1 EWA algorithm

Exponential weighted average is an algorithm for aggregating different predictors in the sequential setting to achieve a regret which in average tends to 0.

- Use all the algorithms except neural nets as candidates.
- Calculate the exponential loss as weight $p_k \propto e^{-\alpha \sum_{t=1}^{T} l^{0,1}(f_k(X_t), y_t)}$
- Use the weights to combine the predictions $f(X^{new}) = sign(\sum_{k=1}^{K} p_k f_k(X^{new}))$

4.2 Result and comment

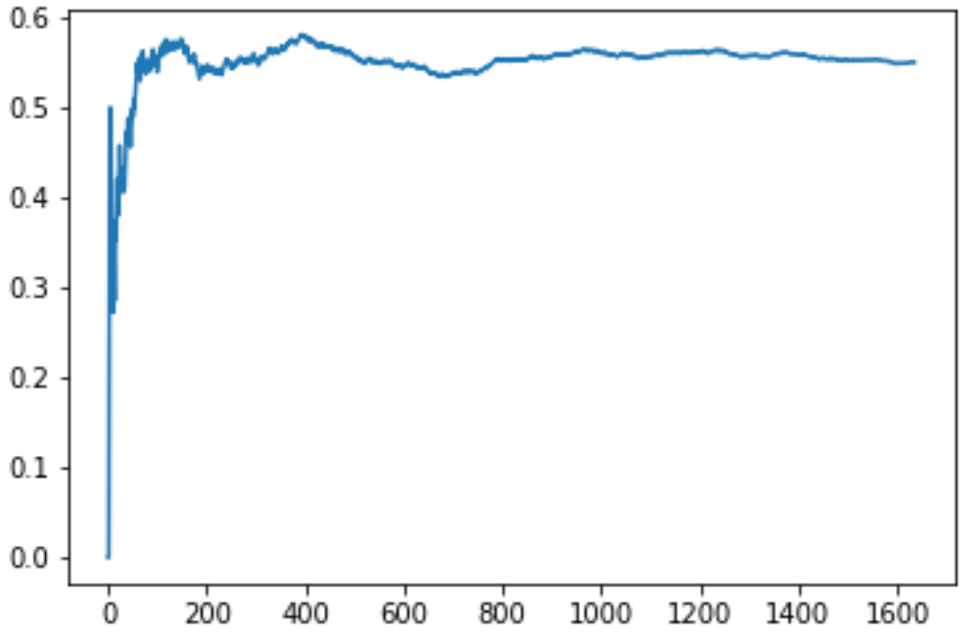

Figure 4: Aggregation

As shown. in Figure 4, there is still no significant progress by aggregating the predictors.

5 Stock Data

5.1 Description of the task

Now we consider the more complex setting where not only we have to predict the trend of the market, but how much it would actually change and for more than one companies. So now it is in a sequential setting:

- Data: $(X_{pre} = \{(X_1, ..., X_t)\}_{t=1}^{T-1}, X_{present} = \{X_{t+1}\}_{t=1}^{T-1})$
- Goal: find f such that $f(X'_{t-1}, ..., X'_1) = X'_t$

5.2 Principles for different methods

Generalized Additive Model

In statistics, a generalized additive model (GAM) is a generalized linear model in which the linear predictor depends linearly on unknown smooth functions of some predictor variables, and interest focuses on inference about these smooth functions.

Kalman Filter

Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe.

5.3 Scaling

The vast majority of the algorithm we saw scaled badly in our setting except Deep Learning, in order to overcome this difficulty we used basic feature selection extra tree regressor to be specific, the efficiency of the technique is easily explainable, we computed the matrix of covariance and we noticed that stock variable are highly correlated so getting rid of some of them does not affect the prediction.

There correlations are shown in Figure 5, where for each pixel in the figure, if the correlation between its x-coordinate feature and its y-coordinate feature is larger than 0.8, then it is white, otherwise it will be red until dark.

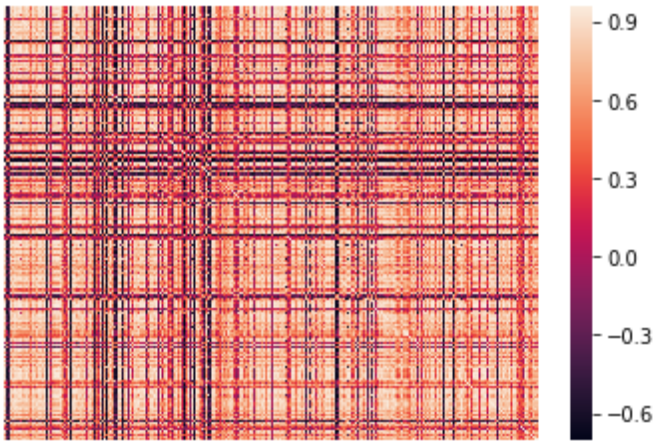

Figure 5: The hitmap for the correlation

5.4 Results

GAMs model

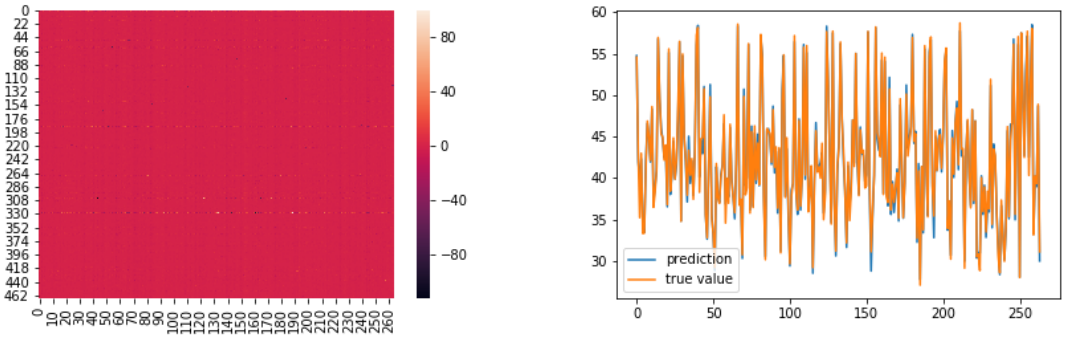

Figure 6: The results of GAMs model

The picture above shows a heat map of (prediction-real value) the fact that it is almost 0 everywhere indicates that the prediction is really precise, the plot on the right shows what actually the curve looks like for a fixed prediction GAM model capture the behavior of the market.

Random Forest

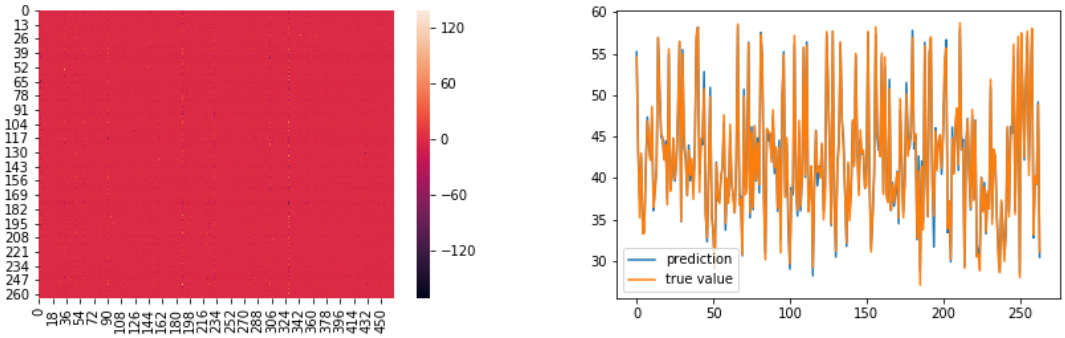

Figure 7: The results of Random Forest

The picture above shows a heat map as in the GAM model, random forest method predict the value with a high precision and both algorithm are running on the full matrix

5.5 Neural networks

Keep in mind that we used less 10% of the variable to predict 100% of the values, so knowing that the algorithm behave pretty well we tried a bunch of architecture this one is the one that had the best result on the test set.

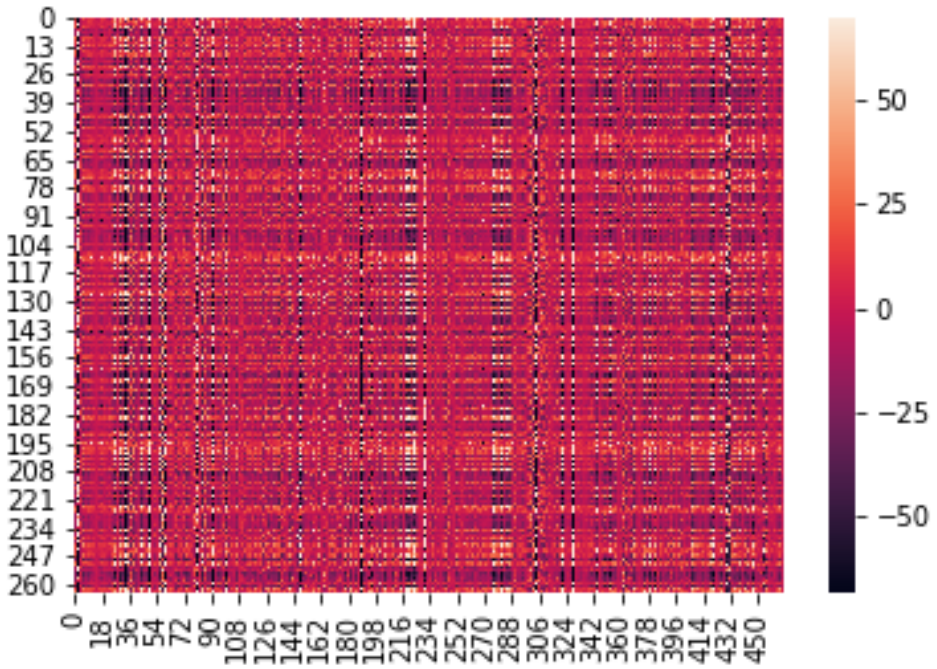

Figure 8: Result for Deep Learning

5.6 Kalman filter

Using the same trick as for DL (we selected variable of interest) so the idea is to reconstruct Y knowing just a part of it the trick here is to use matrix of correlation as some sort of projection function and apply it to Y. We had good results as the explained variance score 0.86.

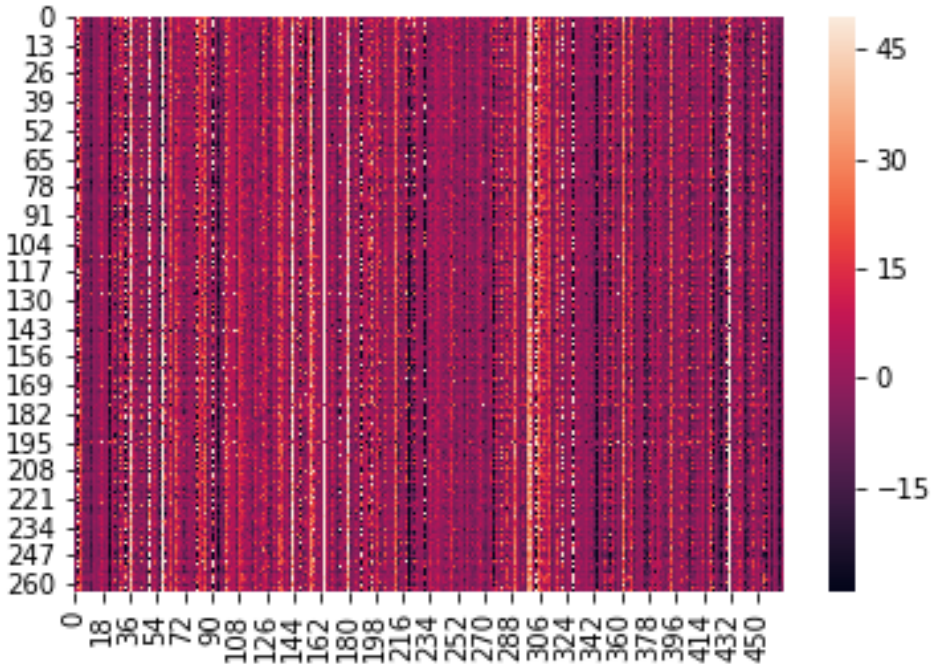

Figure 9: Result for Kalman filter

5.7 The procedure

As the name suggest the extra tree regressor is a tree based method which simply said run's on a reduced number of variable and are chosen the variable that helps the most to do a good prediction obviously with enough data we can expect to have good results otherwise overfitting is to be managed.

6 Conclusion

We noticed three points we wanted to emphasize, the first one is that obviously the model works and works pretty well it seems that in the chaos of stock market exist some pattern that traditional algorithm captures (GAM , Trees , Kalman) and non conventional one (LSTM), the second one is that we didn't succeed in taking advantage of news data is it because the bag of words point of view is too simple or because there is no correlation between news and prices is a an open question for further investigations, the last one is how can we get better at predicting the prices one my look for the best architecture of neurons but this could be inconclusive due to the lack of theory in DL fields thus we are left with only one improvement

possible which is to take account of more data news or social media as in[5] leads to nice improvements.

Reference

[1] `https://en.wikipedia.org/wiki/Long_short-term_memory`.

[2] `https://github.com/mmihaltz/word2vec-GoogleNews-vectors`.

[3] Bernal, A., Fok, S., and Pidaparthi, R. Financial market time series prediction with recurrent neural networks, 2012.

[4] Jaeger, H. The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'.

[5] Tsui, D. Predicting stock price movement using social media analysis.