Challenge QRT:
Stock Return Prediction

Philippe Ganshof
Ecole Normale Supérieure
Cachan, France
philippe.ganshof@hotmail.com

Yoann O. Jayer
Ecole Normale Supérieure
Cachan, France
yjayer@gmail.com

Abstract

With the recent success of machine learning in a wide range of fields, people put in a lot of effort trying to predict the stock market based on historical data. Determining the future value of a company stock can lead to significant profits. However, the behaviour of time series stock is close to a random walk which makes it particularly difficult finding patterns. In this report, we explain our approach for predicting the relative return based on historical data in the challenge proposed by Qube Research and Technologies. Our optimal model is based on a Random Forest Classifier and by aggregating features found with several clustering algorithms applied directly on the time series. We obtained a score of 51.77 percents on the public data (for comparison, benchmark is at 51.31 and best score is at 52.02).

1 Introduction

Understanding and, even better, predicting financial time series in the area of stock market trading has been a longstanding problem. Motivation can be both better understanding the market and trading, and thus it has drawn the attention of both economists and finance institutions. The large datasets available, with the recent boom in computational power, open the way to Machine Learning in this field. As stated in (4), more than half of the implementations of deep learning are focused on this area. Some variations exist but the main centre of interest lies on predicting the next movement of the underlying asset.

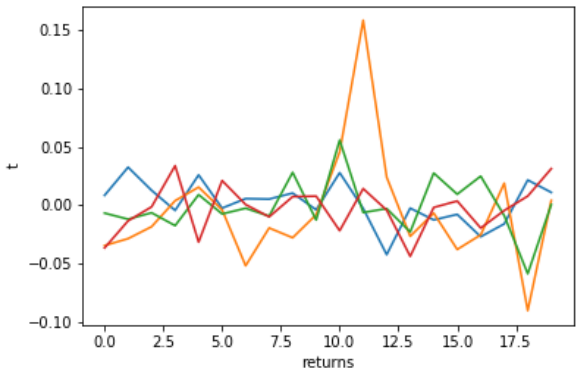

Figure 1: **Noise.** Four time series with the same date index taken randomly in the training set.

The essence itself of the market makes it hard to predict. Indeed, experimentally, stock market time series are really noisy and can be easily mistaken for a random walk. Worst, this noisiness finds a theoretical interpretation. The Efficient Market Hypothesis (Eugene Famas, 1970) states

that stock prices reflect all information on the market and that stocks are always traded at their fair values. Therefore, it should be impossible to outperform the overall market through expert stock selections or market timing. The only way an investor can obtain higher returns is by purchasing riskier investments, making our problem moot.

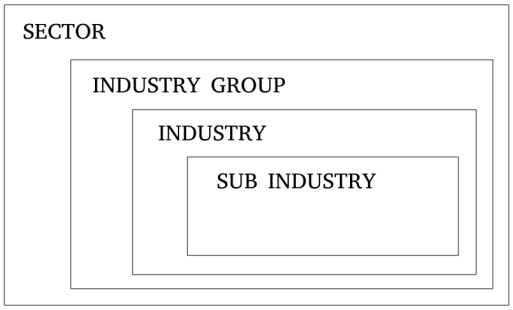

Figure 2: **Hierarchical structure of group features.**

Nonetheless, the stock market is not a perfect game and the market might not always be efficient. Indeed (1) showed that trends can still be weakly predicted by machine learning algorithms and Machine Learning and Deep Learning techniques have been heavily applied to financial times series with sometimes partial success (see (7) for a literature review of the field).

In practice, people are not so much interested by the actual prediction but rather the price movement trend prediction. Time series forecasting is thus often transformed into a simpler classification problem as it is the case in this challenge. Additional information about the stocks can also often be a key factor to success. Therefore, Machine Learning can learn and has been widely used with more or less successes. Some examples are (4) that use Support Vector Machine to and (6) that applied Artificial Neural Networks but the LSTM dominates the financial time series forecasting domain.

In the case of the data challenge proposed by *QRT*, we had access to time series of relative stock returns and volumes in the US stock market for 223 time periods of 20 days, with data points recorded at one day intervals. Additional information were provided for each time series, including the sector, group industry, industry and sub industry with a hierarchical structure (see 2). Time periods were randomized in order to avoid links between any dates and divided into a train dataset of approximely 600'000 examples and a test inputs representing one third of the training dataset. The forecasting task consisted of classifying the input time series by predicting the sign of relative returns of the next day. To better visualize the data, a short description of a few training examples is shown in table 1.

DATE	STOCK	INDUSTRY	INDUSTRY_GROUP	SECTOR	SUB_INDUSTRY	RET_1	VOL_1	...	RET_20	VOL_20
0	2	18	5	3	44	-0.015748	0.147931	...	-0.002155	-0.000937
0	3	43	15	6	104	0.003984	Nan	...	-0.034722	Nan
...	...	...	...	...	...	...	...	...	...	...
223	5710	33	10	4	83	0.012248	-0.627169	...	0.003679	-1.393662
223	5713	26	7	4	60	0.076162	-1.325986	...	0.003679	-1.393662

Table 1: **Training set.** Part of the table

For this project, we first implemented a Random Forest Classifier (as in the benchmark) with some smart features taking into account the sector, industry group, industry and sub industry of the stocks. We then show, in section 3, that the they gather only little information on the stocks and developed our own data-driven classification of the stocks using clustering. In section 4, we implement again a RFC but with features from our data-driven classification. Finally, we also explain some other models we explored during this project that were less successful and conclude with an open discussion about the project.

2 Random forest Classifier

2.1 Pre-Processing

Pre-processing often plays an essential role in the success of a model and thus before going through our first approach, we talk about the form of our dataset. Since the time series were given in terms of relative (with respect to the previous day) returns and volumes, most absolute values do not exceed 1. We therefore did not apply any rescaling techniques such as normalization or standardization in this chapter and already obtain satisfying results. We did try normalization (dividing by euclidean norm) for improving our optimal model but we observed a decrease in the performance.

As we can see in table 1, another specific property of the data is the presence of *Nan* values. They mostly appear in the volume features and often over the entire time series all at once. It makes it impossible to replace them by the previous volume in the time series. We tried different strategies such as replacing them by the mean conditional on the date and sector for example but none of them stand out. The impact is not significant enough and results vary depending on the model, essentially staying in a range of $\pm 0.05\%$. We therefore opted for a classical solution which consists of replacing the *Nan* values by 0 for all our models in order to keep homogeneity.

2.2 Models

We investigated in a first time the performance of several classical classifiers on the original time series. As we can see on table 2, considering only the previous 4 days leads to better results in general. It is not surprising as the data are really noisy and weakly correlated. We also find that, after hyper tuning, the Random Forest seems to be the suitable option for finding patterns in the data, obtaining an average accuracy of 51.44% by performing 4-Fold cross validation on the training dataset. The associated optimal hyper parameters are 515 for the number of estimators and 8 for the maximal depth of trees. We keep a similar configuration for the rest of the paper. The Random Forest algorithm builds a large set of decision trees reducing the variance of each of them and combining these decision trees with a bagging approach.

Even tough we found many deep learning implementations in the literature such as (5), our attempt with LSTM and deep neural network lead to poor results. LSTM, by its construction relies on the temporal properties of any time series but does not extract the appropriate features in our situation. There might exists some efficient techniques using deep learning but we decided to focus on the random forest classifier for the next two sections.

Classifier \ Previous days	3	4	5	6	7
Linear SVM	50.64%	50.84%	50.49%	50.32%	50.35%
Random forest	51.30%	**51.44%**	51.31%	51.34%	**51.44%**
Logistic Regression	51.09%	51.07%	50.98%	50.94%	51.03%
Multi-layer Perceptron	51.14%	51.09%	50.91%	50.89%	50.96%

Table 2: **Performance of Classifiers.** We compare the average accuracy using 4-Fold cross validation on the training dataset with different classifiers and by changing the number of previous days. For instance, considering only the 2 previous days consist of taking the features [RET_1, VOLUME_1, RET_2, VOLUME_2] for training and testing.

So far, we have not yet incorporated the extra descriptive features at our disposal. They obviously contain some useful information and a part of this challenge consists of finding a smart way of using them. For instance, the specific date index can tell us about the general temporal trend.

We choose to aggregate features with some statistics as it is done in the benchmark proposed by *QRT*. In this way, we hope to help the Random Forest find interesting patterns and improve the best result from table 2. Two statistics were considered, the mean and variance, on different targets (e.g. RET_1, VOLUME_1) conditionally to some features (e.g. Date index, Sector,...). There exists many possible combinations and table 3 illustrates some of them. We essentially choose RET_1 and VOL_1 as target features while conditioning on the date and one of the four groups (Sector, Industry group, Industry and Sub industry) in order to have a feature describing the general trend of a specific group

for each day. In theory, we expected the mean to be more useful than the variance as the mean really has the advantage of reducing noise from the signal, even tough some information can be lost in the process. Indeed, we obtained a slightly higher accuracy in general as we can see in table 3.

	Target \ Conditional Features	Sector	Industry group	Industry	Sub industry
mean	RET_1	**51.48%**	**51.50%**	51.34%	51.32%
	VOL_1	51.36%	51.40%	**51.40%**	51.45%
variance	RET_1	51.17%	51.33%	51.29%	**51.51%**
	VOL_1	51.40%	51.49%	51.40%	51.41%

Table 3: **Performance when aggregating one feature.** We compare the average accuracy of the Random Forest Classifier using 4-Fold cross validation on the training dataset when aggregating different features. Each feature represents the mean or variance of RET_1 or VOL_1 conditional on the date and one of the four group sector, Industry group, Industry and Sub industry).

While limiting ourselves to aggregate only one feature, we were able to increase the accuracy to 51.51% when taking the variance of RET_1 conditioned on the date and the sub industry. We then started to aggregate several features at the same time. Again, the number of possibilities is very large and so we focused our search on features from table 3 performing well while ensuring mutual diversity. By combining the best feature of each column (marked in bold on table 3), we reached an accuracy of 51.66%. Below is a figure illustrating the importance of each feature associated to this result and we note the strong influence of the aggregated features.

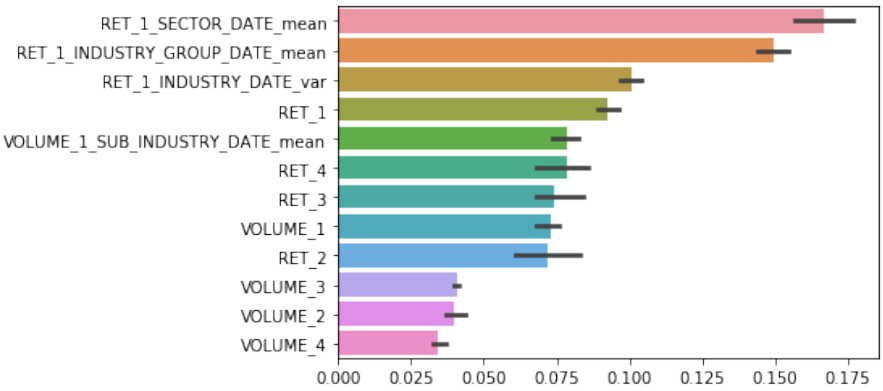

Figure 3: **Importance of each feature.**

Using the extra descriptive features at our disposal lead to an improvement 0.18% in the accuracy. However, natural questions can be raised, how significant are these information for our prediction? Can we find better features? the question is studied in the following chapters.

3 Pre-clustering

In the past section, we observed that the Random Forest Classifier performances were greatly improved when trained with features containing information about the stocks' fields (Sector, Industry Group, Industry, Sub Industry). Indeed, taking the mean or variance tends to reduce the noise of the time series and better show temporal tendencies.

In this section, we will assume that it is pertinent to design clusters to reduce the noise and we will work on techniques to do so. In the first part, we will answer the question: How pertinent are the clusters derived from the fields of the stock? Then, we will develop techniques to learn clusters directly from the time series.

3.1 Clusters derived from knowledge of the stocks

More formally, we are going to study how much variability of the stocks are contained in the features sectors, industry..

To do so, let's define the variability of the stocks in a field F.

$$Var(F) = \frac{\sum_{f \in F} ||f - E_F||_2}{|F[} \tag{1}$$

where $E_F = \sum_{f \in F} s$ and $||.||_2$ is the 2-Euclidean norm.

The variability consists of the average distance between a time series from the field F with the average of the time series from the field F. One value even more interesting is the relative variability Var^*.

$$Var^*(F) = \frac{\sum_{f \in F} ||f - E_F||_2}{\sum_f ||s - E||_2} \tag{2}$$

The relative variability is the fraction of the variability of all the stock that is also contained in the cluster (or field). If the relative variability is close to 1, the cluster is not representative of the stocks in it. If the relative variability is close to 0, all the stocks in the cluster behave almost the same. In our case, we want the cluster (or field) to be representative to indicate a tendency but still have some variability to average the noise.

Let's compute the relative variability for all four types of fields (Sector, Industry Group, Industry and Sub Industry).

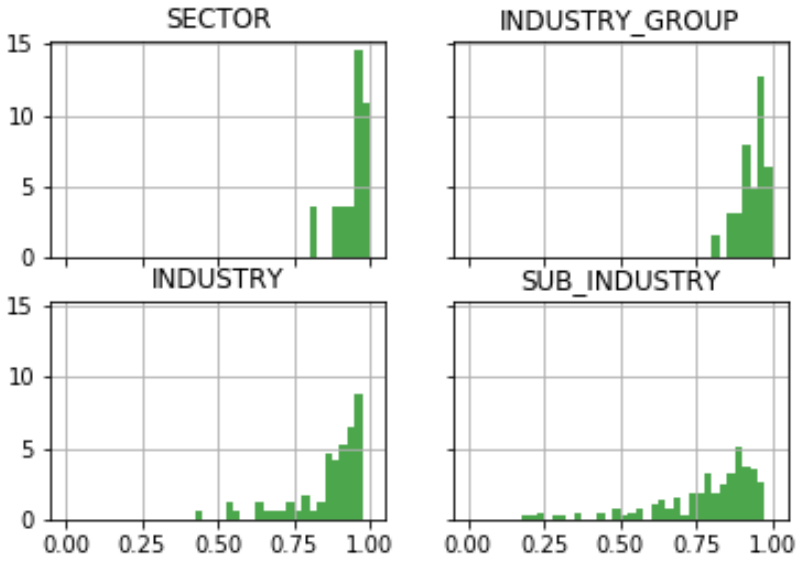

Figure 4: **Relative variabilities for various clusters derived from stocks information**

From Figure. 4, we observe that, as expected, the smaller is the field, the more representative it is. Yet, even for the Sub Industry fields, there still is a large majority of clusters that are almost not at all representative of the stocks in them. The reasons might be that the fields are too vast and contain too different stocks or that, in some sectors, similar stocks do not necessarily behave the same . For us, that means that most of the fields gather almost no knowledge about the stocks in them and cannot help us make good predictions.

The fields given in the challenge are only helpful for a minority of stocks. That explains why they improved performances, but only a little. In the next part, we propose to construct our own cluster directly from the time series.

3.2 Data-driven clustering

In this part, we propose to construct clusters designed especially to reduce their relative variability. To do so, we applied classical clustering methods for time series

$$R^{i,t} = (R^{i,t}_{20}, R^{i,t}_{19}, ..., R^{i,t}_{2}, R^{i,t}_{1}) \in R^{20}$$

where $R^{i,t}$ is the time series of the returns of the stocks i for the time period t.

The first algorithm we propose is the well-known K-means algorithm. It consists of considering that the data is generated by several Gaussian variables with Id covariance matrices. Each Gaussian variable has therefore a region of the space where it is dominant over the other (every point of this area is generated by this Gaussian with the highest probability), which corresponds to the different clusters. The algorithm is trained by successfully updating the centre of the Gaussians and the lattence variable that state on from which Gaussian a datapoint was generated with the Expectation-Minimization algorithm. K-means is very sensitive to initialization so it is usually initialized multiple times and only the best results is kept.

For the time period $t = 1$, with 85 clusters (the bests of 50 random initializations), we observe the following relative Variabilities (see 5).

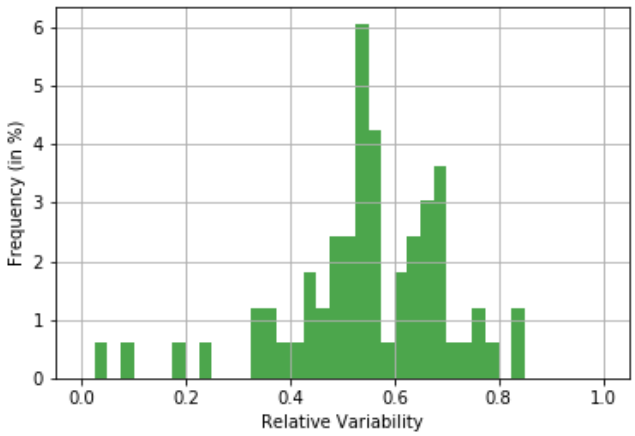

Figure 5: **Relative variabilities for K-means clustering**

We observe almost no relative variabilities greater than 0.8 and lower than 0.35. We note that, with K-means, most clusters gather information about the stock in them.

We tried the same procedure with another clustering algorithm, the Gaussian Mixture algorithm. Gaussian Mixture is an extension of K-means with unconstrained covariance for each generated Gaussian variables. For this project, we are still enforcing the covariance matrices to be diagonal as it is cost effective and does not change much the results in our case.

For the same time period $t = 1$, with 90 clusters (the best of 20 random initializations), we observe similar, yet a little higher, relative variabilies (see 6).

For this challenge, we implemented two other clustering algorithms, which are:

- The K-means algorithm with the Dynamic Time Warping (DTW) metric, another tool for measuring the similarity between two temporal sequences, often used for stock prediction.

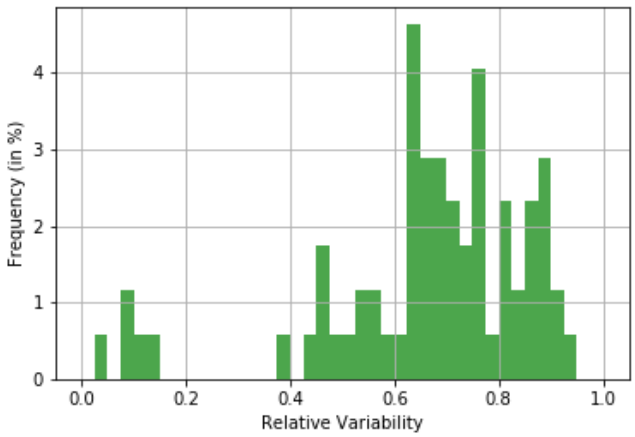

Figure 6: **Relative variabilities for Gaussian Mixture clustering**

The idea lies on building one-to-many and many-to-one matches so that the total distance can be minimised between the two. The optimal match gives the DTW distance.

- The affinity propagation algorithm is a clustering algorithm which consists of seeking for exemplary points in the dataset that represents well the others through a message passing algorithm. An adavntage of affinity propagation is that it does not require a fixed number of clusters as a parameter.

We will see in the next section that, while K-means and Gaussian Mixture are quite efficient and improve the quality of the predictions, both DTW-Kmeans and Affinity Propagation were deceiving. This can be explained as DTW-Kmeans implement a distance classical for time series with strong temporal correlation and our data are too noisy. Similarly, the data might also be too noisy for affinity propagation so none of the time series are exemplary.

4 Random Forest Classifier with date-driven Pre-Clustering

In this section, we describe our main algorithm for this challenge which consists of a Random Forest Classifier (sees section 2) with data-driven clustering beforehand (see section 3). We present our results for various clustering algorithms.

4.1 Random Forest Classifier with K-means pre-clustering

Let's consider the algorithm which consists of a Random Forest Classifier with a K-means pre-clustering beforehand. For the Random Forest Classifier, we copy what worked well in section 2, that is to say a maximum depth of 8 with 515 estimators. Besides the last 4 returns and 4 volumes of the stocks, we consider also the last 4 returns means by cluster. K-means is initialized by K-means++ with 50 random initializations.

To tune the model completely, we must answer the issue of the number of clusters. The choice of the number of clusters represents a trade off. Indeed, from section 3, the higher the number of clusters is, the more representative is the cluster. However, the noisier it also is as it averages over less stocks. Thus, there is an optimal number of clusters.

Experimentally, the performances of the algorithm is quite sensitive to the number of clusters (see Table. 4). We thus set the number of clusters to 85 with the K-means algorithm.

number of clusters	50	80	85	90	120
Accuracy (in percents)	51.60	51.63	51.72	51.67	51.51

Table 4: **Accuracy for Kmeans + Random Forest Classifier.** We compute the accuracy for different number of clusters in K-means.

4.2 Random Forest Classifier with Gaussian Mixture pre-clustering

We tuned in a similar way the Random Forest Classifier with Gaussian Mixture pre-clustering. We initialized the Gaussian Mixture algorithm randomly 20 times and chose 90 clusters (see Table. 5).

number of clusters	80	85	90	105
Accuracy (in percents)	51.80	51.82	51.77	51.71

Table 5: **Accuracy for Gaussian Mixture + Random Forest Classifier.** We compute the accuracy for different number of clusters in Gaussian Mixture.

We observe that Gaussian Mixture leads to a slightly better performance that K-means, which is not surprising as it is a more subtle clustering algorithm (Gaussian Mixture is an extension of K-means).

4.3 Random Forest Classifier with other clustering algorithm

As stated in section 3, we trained our model with two other clustering algorithm. Let's observe the best results for each clustering algorithm in 6.

clustering algorithm	Kmeans	GaussianMixture	DTW-Kmeans	AffinityPropagation
Accuracy (in percents)	51.72	51.82	51.25	50.90

Table 6: **Accuracy for Gaussian Mixture + Random Forest Classifier.** We compute the accuracy for different number of clusters in Gaussian Mixture

We observe that, while K-means and Gaussian Mixture improves the quality of predictions of the model, DTW-K-means and Affinity Propagation degrade the performances of the model (see section 3 for details on the algorithms).

4.4 Implementation

Let's describe in this part a few implementation issues:

- It could be interesting to note that the performances of the Random Forest Classifier with pre-clustering beforehand has a strong dependence on the clustering algorithm and especially its initialization. Therefore, a similar algorithm can have quite strong difference of performances if the random state is not the same. This is especially true for K-means.
- With a fixed given pre-clustering, changing the hyperparameters can highly change the performances of the model. Nonetheless, almost all the time, a maximum depth of 8 and 520 estimators are the best parameters (sometimes a depth of 9 can perform better)
- Through these challenges, we replace all the NaN values with 0. We tried replacing them by the mean of their the same variable for their respecting field (sector, industry group, industry, sub industry) but it decreased the performances. Replacing the NaN by the mean of the data-driven cluster they belong in does not change much the performances and is more computational demanding.

4.5 Random Forest Classifier with multiple pre-clustering

In this part, we tried to combine clustering of different algorithms and for different numbers of clusters to improve the performance of the model.

Practically, adding information from different clusters in the feature map only decreased the performances of the Random Forest Classifiers. The algorithm did not manage how to combine the two different means of the returns for different clusters.

We also tested methods from ensemble learning where we trained several models separately and use a voting system to make a prediction. The combination that worked the best was a simple 50% / 50% of our two best results respectively with K-means and with Gaussian Mixture Pre-clustering.

Indeed, with K-means pre-clustering (85 clusters, 50 random initializations) and with a Random Forest Classification (max depth of 8, 515 estimators, 4 last returns, volumes and means of the returns for its cluster), we obtained 51.72 % by cross-validation.

With Gaussian Mixture pre-clustering (90 clusters, 20 random initializations) and the same tuned Random Forest Classification, we obtained 51.82 % by cross validation.

By combining the last two models wth a 50% / 50% voting scheme, we obtained a result of 51.93 % by cross validation. For the public dataset, we obtained 51.77 % (for comparison, the benchmark scored 51.31 % and the best score is 52.02/=%).

5 Further Experiments

The previous chapters follow a coherent way of thoughts that lead to our optimal model, submitted in the challenge. However, other paths were explored during this project and we give in this section a short insight of some of them.

We tried to use an unsupervised method that learns universal embeddings of time series published by (3). They propose an encoder built as a deep convolutional neural newtork with dilated convolutions that outputs a vector representation of fixed length. The model is then trained in a unsupervised way using a triplet loss with the objective of attributing to similar time series similar representations. Our idea was to learn representations of our times series and classify them using the Random Forest Classifier and finding clusters from these time series as in the previous chapters. We used the implementations found in the github of J-Y.Franceschi (Franceschi) for learning the representations. Unfortunately, we obtained poor results, an accuracy of 50.94% using 4-Fold cross validation on the training dataset and gave up on this path.

Even tough most of our deep neural network architectures lead to poor results, one stands out, multi-task learning. We built an model with two outputs, similar to a feed forward neural network but by splitting half way. More precisely, we first added four consecutive dense layers of width 64 and then split the model in two different ways while adding 2 consecutive dense layers to each of them. The model was trained jointly to predict both which cluster (built with Gaussian Mixture) the time series belong to and the sign of RET. In this way, we hoped that learning to predict the associated cluster for each time series would help the model find an embedding suitable for the main task. The cross-entropy was used as loss function for both outputs and we put twice as much weight on the output predicting the relative return. The model was trained on the training data considering, as suggested, only the previous 4 days as inputs. The results were quite promising on the training set as we obtained an accuracy of 52.2% . Unfortunately, we could not prevent overfitting and the accuracy fell down to 50.90% on the test set.

6 Conclusion

Financial time series in the stock market are very noisy and classical binary classifiers do not manage to learn clear patterns. Indeed, the Random Forest Classifier, which performs the best, cannot predict more than a little bit over 51%. Nonetheless, we observed that considering new features that average the values of the time series over representative clusters can help reducing the noise and prompt better performances. At first, we design those clusters from the additional knowledge of the time series we had from the data (the sectors, industry groups, industries and sub-industries). Then, we noted that clusters directly driven from the data through classical unsupervised clustering algorithms such as K-Means or Gaussian Mixture managed to also reduce the noise while conserving more information and performs better. Finally, we combined Random Forest Classifiers from both K-Means and Gaussian Mixture pre-clustering through a voting scheme and obtain our best results which

is 51.93% by cross-validation and 51.77% on the public score ranking. We also considered Deep Learning algorithm but did not succeeded to beat the Random Forest Classifier.

References

[1] Das, S.R.; Mokashi, K. C. R. (2018). Are markets truly efficient? experiments using deep learning algorithms for market movement prediction.

[Franceschi] Franceschi, J.-Y. Github.

[3] Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. (2020). Unsupervised scalable representation learning for multivariate time series.

[4] Huang, W., Nakamori, Y., and Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Comput. Oper. Res.*, 32(10):2513–2522.

[5] Kai Chen, Y. Z. and Dai, F. (2015). A lstm-based method for stock returns prediction: A case study of china stock market.

[6] Qiu, M. and Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PLOS ONE*, 11(5):1–11.

[7] Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. (2019). Financial time series forecasting with deep learning : A systematic literature review: 2005-2019.