Multivariate Time Series Error Detection

David Pham

September 15, 2018

Contents

1 Project overview

In the financial investment industry, the speed and the quality of processing information is what determines if a business is sucessful or failure. In the recent period, an emphasis has been set on machine learning algorithms for investment strategies. One common application is the statement of the algorithms to perform classification of companies to put them into sector.

This project aims to create and review algorithms for classifying companies by using their share price. We will test a heuristic classifier using sample correlation and train a deep neural network to mimic the performance the sample correlation and study the hidden layer of the network. The projects uses data from Quandl and SimFin.

2 Problem Statement

The goal of this project is to create a method to embed financial time series into a vector space and analyze qualitatively the embedding. The main challenge to complete this projects are following.

1. Filter the universe of the data and acquire the data from different sources.
2. Transform data to be usable in machine learning algorithm.
3. Create artifical data to transform the problem into a supervised learning problem.
4. Create a base model and train an advanced classifier to create the embedding.
5. Analyze briefly the embedding.

3 Metrics

We will use the *accuracy* defined as the ratio between the true positive and negatives predictions divided by the dataset size. Or in formula

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{sample size}}$$

This metric is relevant because at a certain level the algorithms should be able to assign correctly the stock price of companies to the sector of the companies.

4 Data

The below table shows the data and how to access them.

Type	Provider	Access to Datatest	Downloaded
Price	Quandl	Quandl API	2018-08-21
Company information	SimFin	S3	2018-08-21
Stocks Returns	Processed	S3	2018-09-11
Index Returns	Processed	S3	2018-09-11

In our analysis, we used the closed price of the US companies provided by Wikipedia and distributed by Quandl. As we additionally required the sectors from the company we needed to merge it with an additional databases provided by SimFin, which provides the taxonomy created by Global Industry Classification Standard (GICS). We could find the sectors of *726* companies which constituted our dataset. The prices range from 1962 to 2018.

5 Exploratory Analysis

6 Algorithms and Technique and

There is two classifiers. One is algorithmic and the is other one is based on a dense neural network using the right transformation of the input to incorporate non linear reliationship between the weights.

The first classifier computes correlations of timeseries and then assign to an input timeseries the *sector* with which it has the highest correlation.

For two random variables X and Y, with sufficient stability assumption, the Pearson correlation is defined as

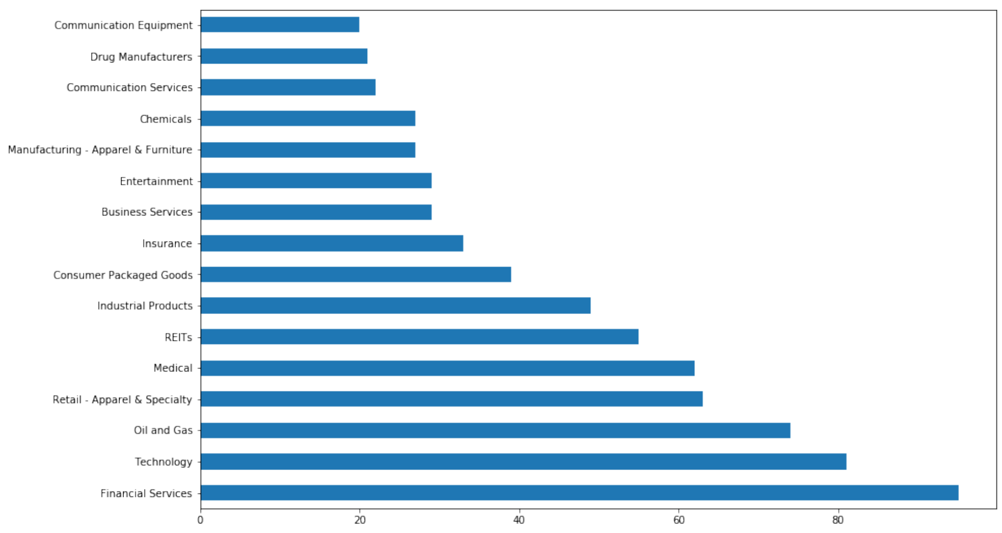

Figure 1: Distribution of sectors in the data.

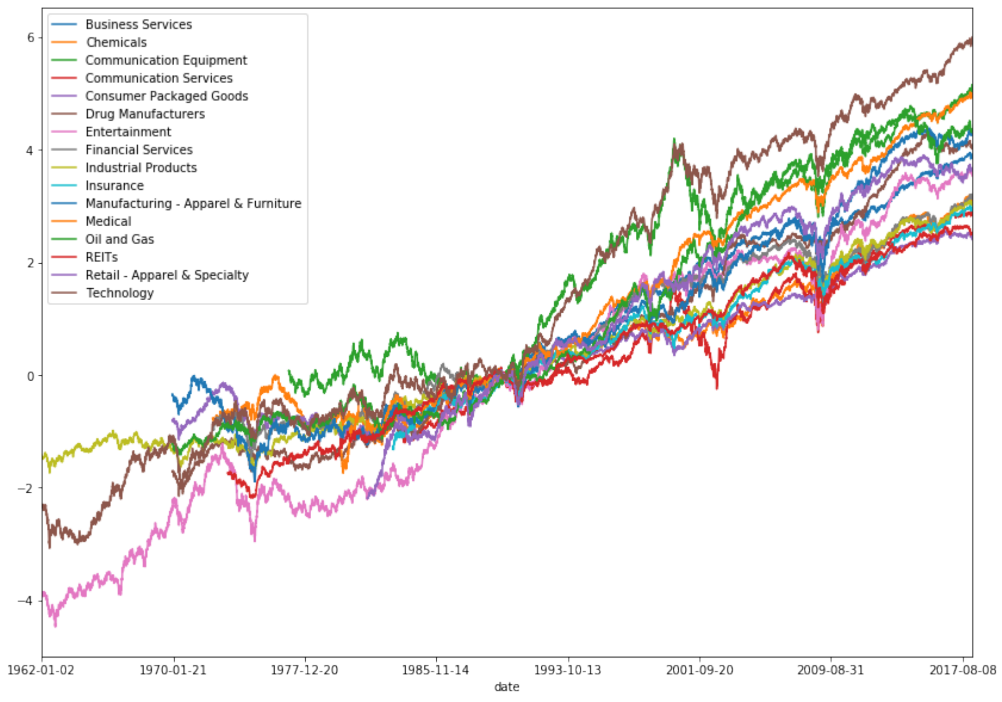

Figure 2: Synthetic indeces of sectors according to GICS. Index are set on 100 on the 1990-01-01.

$$\rho_p(X,Y) = E[XY] - E[X]E[Y] \approx \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i,$$

The second classifier use a deep neural network to create features and representation (or state) of the inputs to be able to separate linearly all the output sectors (or classes).

We train the neural network using *Adam*, which optimize the parameters of the network by minimizing the loss using stochastic gradient descent algorithm which standardizes the gradient and also updates it in iteration. The batch size was set to 128, the learning rate starts at 0.001 with a learning rate scheduler that diminish the learning by tenfold if the metrics has not been improved in the last 10 epoch.

Concerning the input of the model, the data was split into training, dev, and test set using with 80%, 10%, 10% of the overall data set. In the training phase, each instance of a batch we sampled randomly a companie of the training set, sampled a 3 months (63 days) of observation of the companie's stock returns along with the returns of the sectorial indices.

7 Benchmark

Intuitively, we could use random guessing as a benchmark (this would yield a 13% accuracy at best as Financial Services is the biggest represented class). A bit more challenging, we could use correlation as measure of association and using the sector with the highest correlation to our input series, This classifier gets an accuracy rates of 59%.

8 Methodology

8.1 Data preprocessing

From the Quandl dataset, the prepossessing involves keeping only the ticker and the close price for as many date as possible and as many companies as possible. Then the table is joined to the SimFin dataset containing sectors for 726 companies. In total we have 16 sectors, from which we can extract data. Due to the lack of GICS sectors, a few sectors were merged together to increase their size, e.g. all the Oil and Gas companies were merged into a single sector.

Then the sector indices were created by averaging the daily returns of the stocks within the sectors. The returns were floor and capped to 10% as it is unlikely that a indices of stocks lose or gain more than 5% in a single trading day and the 100 level was set for 1990-01-01.

8.2 Implementation

The integrity of the code follows a linear process in the `notebook` folder of the project. One should be able to run all the code in each notebook separately. It was considered to have a proper implementation in the project and to avoid code duplication, but under time constraint, copy paste solution were prefered. That being said, the code has been written using pure python functions to avoid spaghetti code.

The implementation using Tensorflow and the keras API linked in the library. The Keras API allowed to defined our network and our model into a simple function and wrap customized transformation into the `keras.Lambda` layer. The exact implementation can be found in the notebooks.

We launched AWS server with spot instance and launched a jupyter server there and made it accessible to our webbrowser. We develop code also in the terminal with emacs to adapt some code.

The first step was to download the data from the several providers and to process them. Then we needed to use several classes from keras to support asynchronous loading of the data thanks to the `Sequence` object. During training, each sector was sampled with equal probability and a random companie was then selected from the sector.

9 Refinement

The implementation has been performed with a simple function defining the network. We ran several experiment of the network, using adapation of inception units and residual units, known in the neural network for images, but they did not lead to any improvement of the model. Moreover, to avoid overfitting, we added several batch normalization layers as well as gaussian noise layer with a really small standard deviation. A few layers in the network were penalized L_2 regularization to insure that the features stayed as independent as possible.

The best models were the ones which were fed with correlations and forwarded to into a denses networks. The reason is associative measure a non linear and it is not common to multiply inputs with each other. That being said, we managed to create a convolutional layer that achieved an accuracy rate of 55%, a bit short from our best model and from the benchmark, but using only linear transformation.

9.1 Structure of the network

The network is depicted in Figure 9.1. From the input data, three transformation are performed. The first one create product of normalized observation in order to let the model to detect smaller pattern of interaction. The second transformation performs the same computation but on the sign of the input. This should create a more robust estimate of measure of association. The third is to compute the correlation matrix as feature for the model. We concatenate them and create a dense layer for creating the embedding from which we extract the classes.

10 Results

The base model using only correlation for the period of 3 months achieves $59 really good.$

As for neural network model, it achieves around 58% percent accuracy on a single observation of three months which is on par with our benchmark. However, when we provide 25 random samples of 3 months period to the classifier, the classifier achieves 80% accuracy. As it can be read in Table 1.

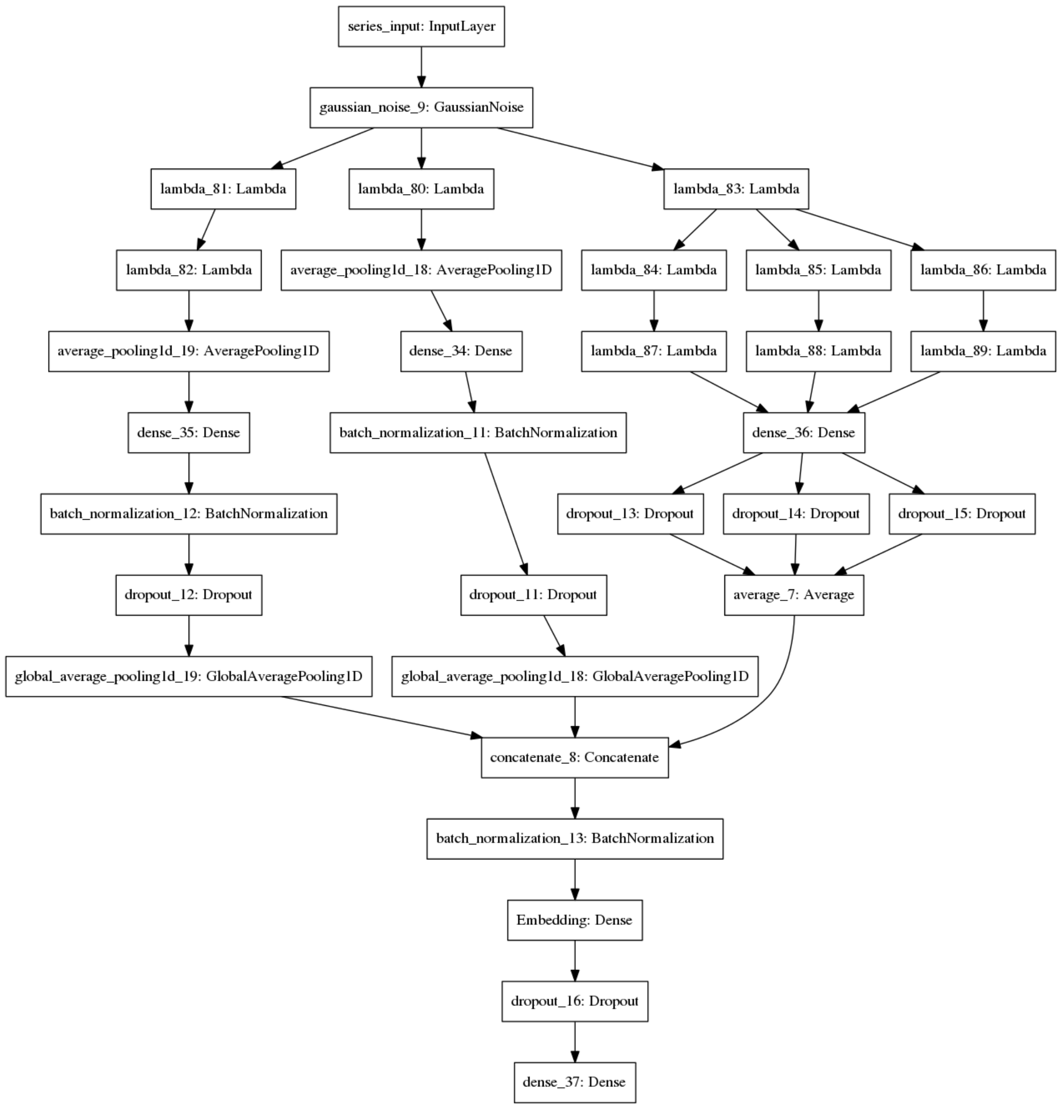

Figure 3: Neural Network structure

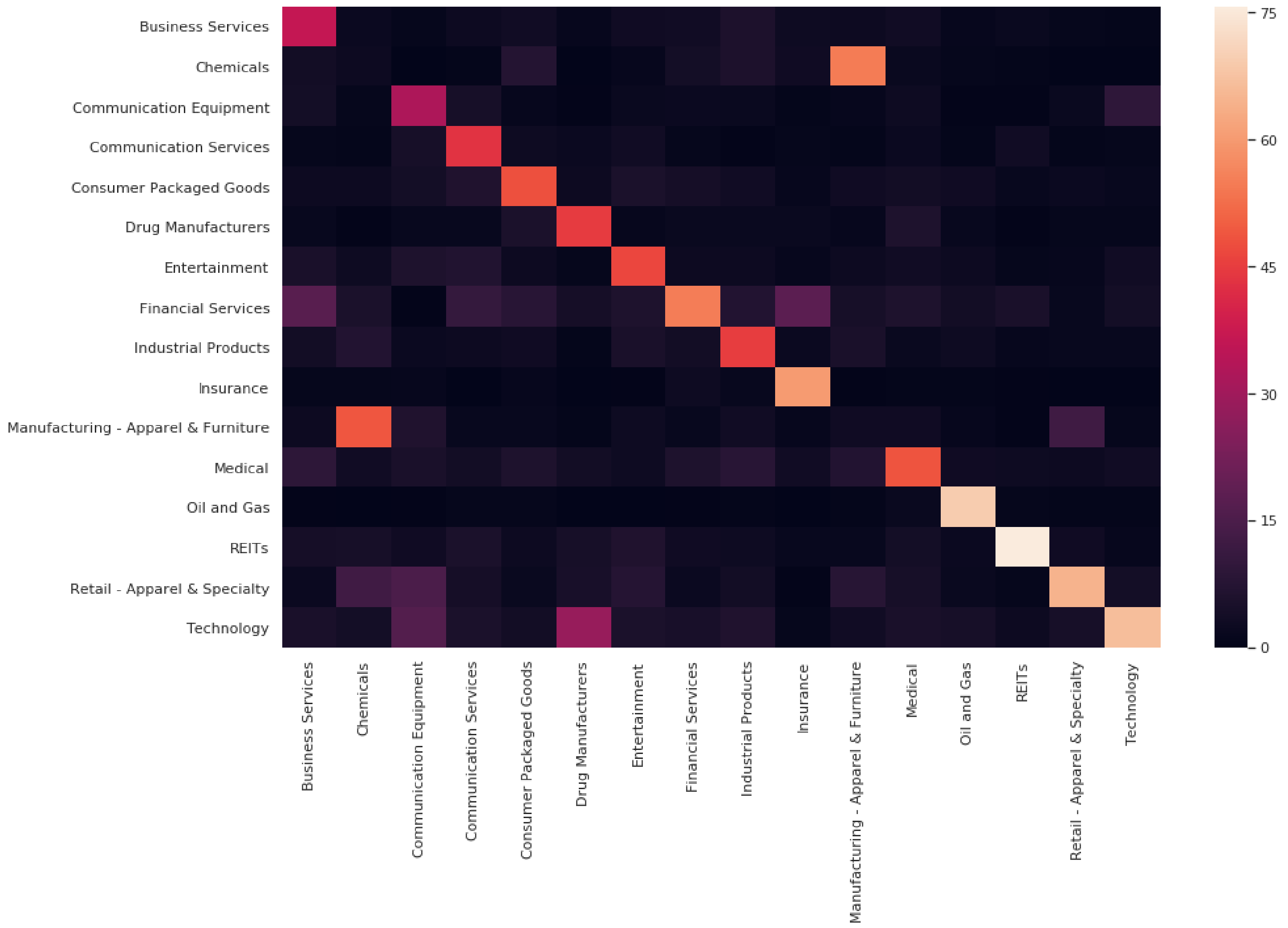

Figure 4: Confusion matrix of our predictor.

In Figure 10, we observe that the neural network model classifier does a fairly good job at classifying sectors with a notable exception of *Chemicals* and *Manufacturing - Apparels and Furniture*. The reason are probably that are little data.

Table 1: Confusion Report from the neural network classifier with resampled data.

Sector	precision	recall	f1-score	support
Business Services	1.00	1.00	1.00	3
Chemicals	0.00	0.00	0.00	3
Communication Equipment	0.00	0.00	0.00	2
Communication Services	1.00	0.50	0.67	2
Consumer Packaged Goods	0.60	0.75	0.67	4
Drug Manufacturers	0.67	1.00	0.80	2
Entertainment	1.00	1.00	1.00	3
Financial Services	0.90	0.90	0.90	10
Industrial Products	0.71	1.00	0.83	5
Insurance	1.00	1.00	1.00	3
Manufacturing - Apparel & Furniture	0.00	0.00	0.00	3
Medical	1.00	1.00	1.00	6
Oil and Gas	1.00	1.00	1.00	7
REITs	1.00	0.83	0.91	6
Retail - Apparel & Specialty	0.86	1.00	0.92	6
Technology	0.78	0.88	0.82	8
avg / total	0.79	0.82	0.80	73

11 Embedding

We are curious to look at the embedding produce by our neural network. We use t-SNE to create a low dimensional representation of it. This technique preserves the similarity between points.

In order to create an estimate of the embedding, we sampled the 25 periods of 3 months of each stock and averaged their embedding.

In Figure 11, it can be observed that stocks from the same sector tends to be near each other. The distinct cluster are finance and technology, which are also the most represented in our dataset. The model seems to have difficulty to differentiate somes chemical companies as their embedding seems to be close to some industrial production companies. In general, the more companies we had in the raw dataset the more precise the groups are.

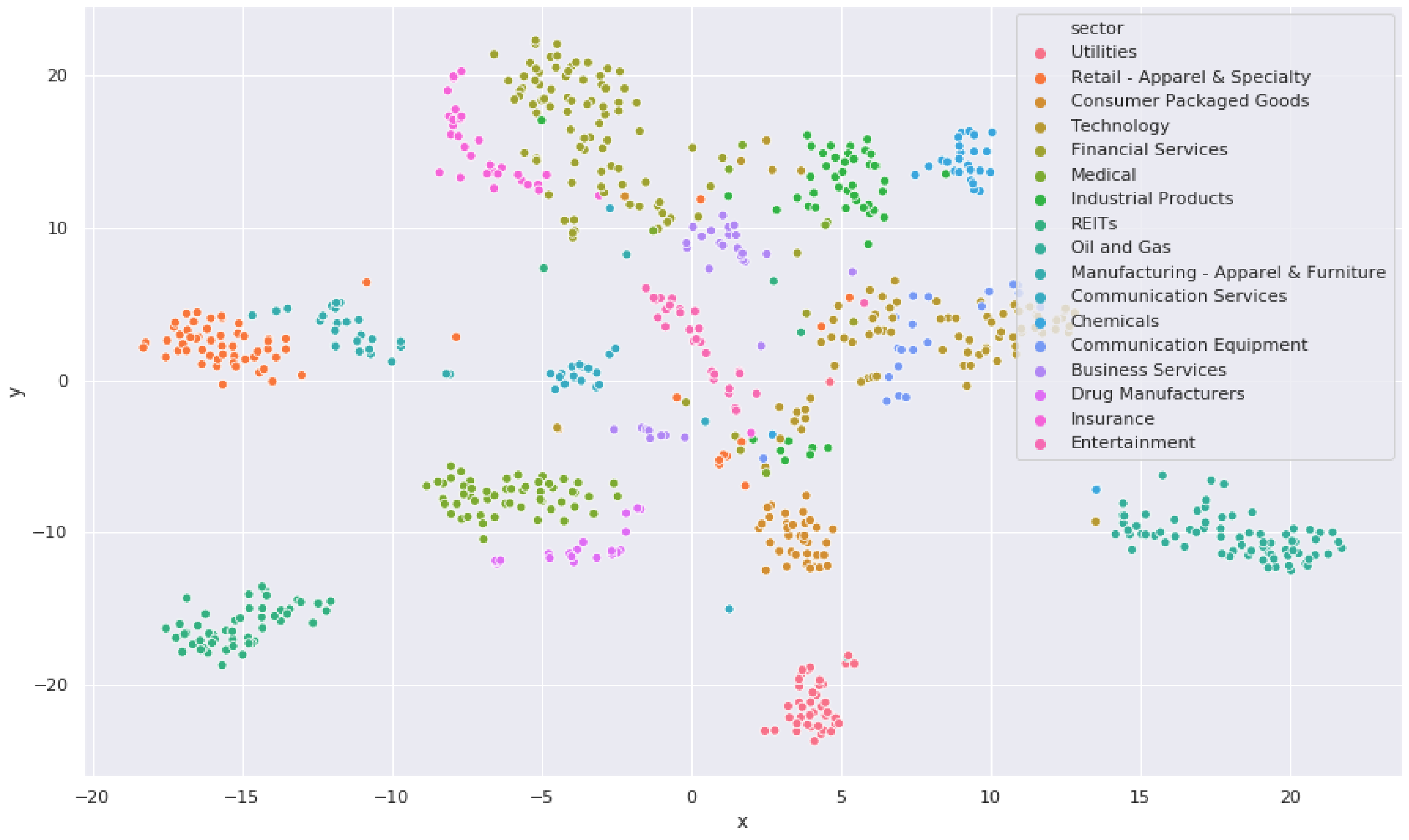

Figure 5: T-SNE of the embedding layers of the network.

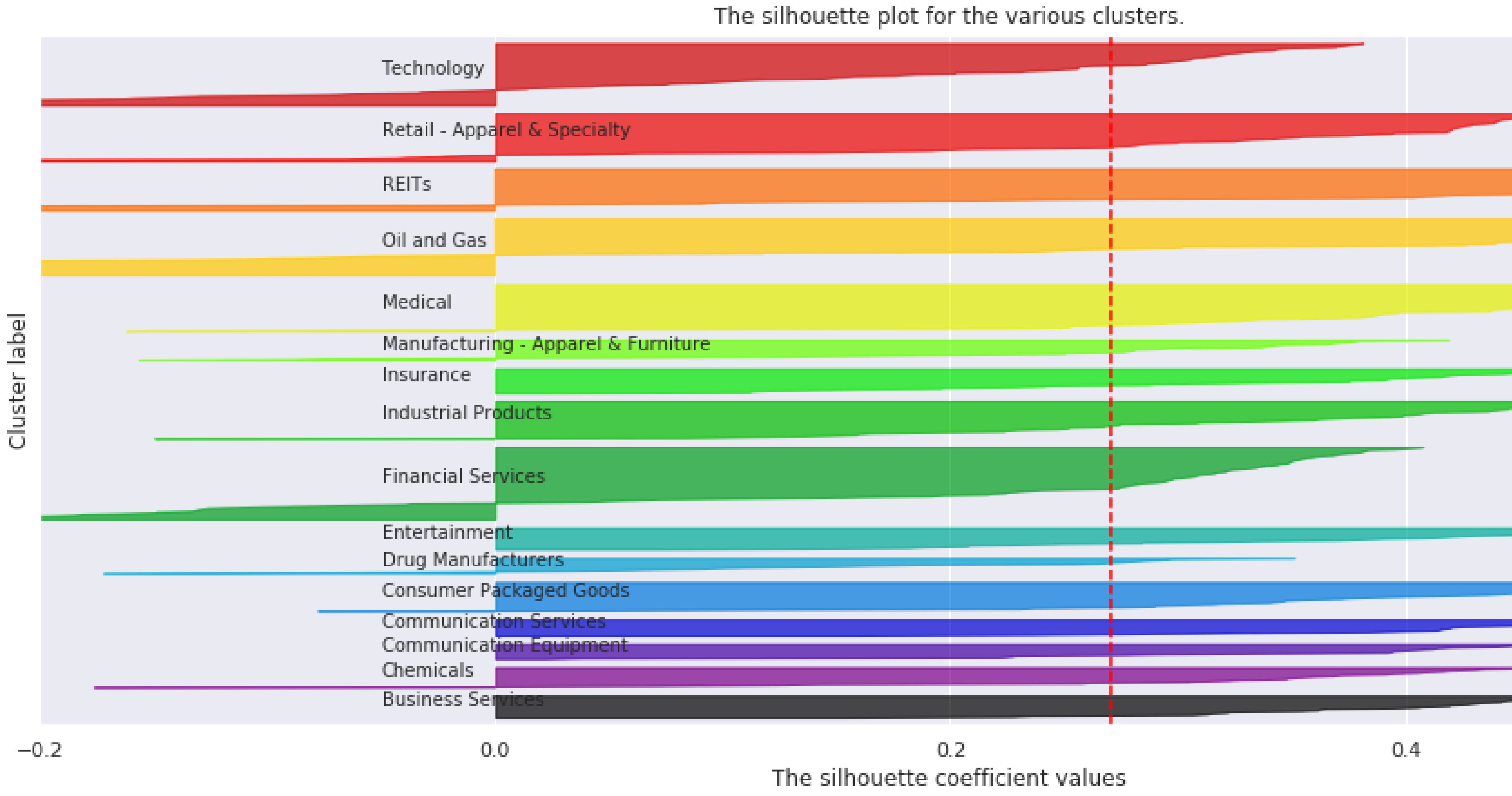

Figure 6: Silouhette score of the average embedding of the stocks

In Figure 11, the silouhette score is depicted for the several sectors in our dataset. The first observation is we see the advantage of a neural network classifier over k-means clustering, because the silhouette score overweights the mislabeled sample of financial and technology, which are the most precise sectors. Otherwise we can observe that REiTS, Oil and Gas, and the Insurance sector are grouped tightly making them quite distinct group.

12 Conclusion and Further applications

The first lesson I learned is data preparation and acquisition is much harder than thought and we should thank the machine learning community for providing so many labeled data set for our development. Indeed the financial industry still leverage on providing exclusive data and create a difficult task to leverage on alternative dataset, which could potentially provide added value.

Second, in deep learning, a bigger network does not necessarily translate into a better performance: training is much more difficult with more parameters even with regularizers and advanced optimization method. As for the training, balancing the classes improves a lot the training and can potentially improve the performance of the model. Financial data also proved to be tricky to handle without proper averaging. The signal to noise ratio is much higher than typical machine learning application domain.

The primary goal of the project was to find method that could create embedding for financial timeseries and Figure 11 provides a good proof that this goal has been reached. Our clustering abilities are still lacking as the accuracy rate for sample of three months is yet not better than a simple correlation measure. But the method has a higher accuracy when sampled with more data.

As for the improvement, we could try different architecture (LSTM and several skip convoluation). The LSTM models could allow flexible time input. Moreover it would have been interesting to apply som semi-unsupervised method to improve the model and embedding. We could have applied our existing predictor for stocks whose sector is missing and recompute the indices and maybe retrain the classifier. A really interesting step would have been to incorporate T-SNE or Uniform Manifold Approximation and Projection in the training and apply additional clustering technique. These low dimensional projections seems to cluster data efficiently. This could also potentially resolve our inability to detect new group in as we would need to train them. Zero shot learning would be an interesting project to the study.