Machine Learning Engineer Nanodegree
Capstone Project

Ronak Tanna
December 27th, 2017

I. Definition

Project Overview

The stock of a corporation is constituted of the equity stock of its owners. A single share of the stock represents fractional ownership of the corporation in proportion to the total number of shares. A share price is the price of a single share of a number of saleable stocks of a company, derivative or other financial asset.

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information.

A lot of research has been done in this domain, few of which are: Citation: Saahil Madge."Predicting Stock Price Direction using Support Vector Machines". Independent Work Report Spring, 2015. This research paper is relevant to this section because it provides details on how to calculate price volatility and momentum for individual stocks and for the overall sector. Citation: Shunrong Shen, Haomiao Jiang and Tongda Zhang. "Stock Market Forecasting Using Machine Learning Algorithms". This project exploits the temporal correlation among global stock markets and various financial products to predict the next-day stock trend with the aid of SVM.

Problem Statement

Stock price prediction is one of the most widely studied and challenging problems, attracting researchers from many fields including economics, history, finance, mathematics, and computer science. The volatile nature of the stock market makes it difficult to apply simple time-series or regression techniques. Financial institutions and traders have created various proprietary models to try and beat the market for themselves or their clients, but rarely has anyone achieved consistently higher-than-average returns on investment. Nevertheless, the challenge of stock forecasting is so appealing because an improvement of just a few percentage points can increase profit by millions of dollars for these institutions. We will try to predict whether the price of a stock will increase or decrease in the following day which help us in identifying the 'winning' stocks.

The basic idea is to acquire the stock price historical data of NASDAQ 100 index companies. We then plan to create a pipeline of pre-processing the data and then merging the datasets coming from different sources. Finally we build Machine Learning model which can be trained on the various features of the dataset, therefore be able to make a prediction about whether a stock is going 'Up' or 'Down' on a particular day.

Metrics

In this project, our aim is to predict whether the NASDAQ 100 index companies will go 'Up' or 'Down'. As this a binary classification problem the accuracy metric we will use is accuracy. Accuracy is defined as the number of correctly classified points in comparison to the total number classifications made.

II. Analysis

Data Exploration

We wanted to see how the SVM model, which has had such success in previous literature, would work in such an abnormally volatile market. We will also look into other Supervised
Learning algorithms and see how much difference they show when compared to SVM. We focus specifically on the technology sector. Focusing on a sector as opposed to the broad market allow us to test the model on companies that are similar to each other, making our results relatively standardized. We use the NASDAQ-100 Technology Sector Index (NDXT) as the general technology sector index. We look at 34 of the 39 stocks in the index. For each individual company we look at daily price data from the start of 2009 through the end of 2016. This allows us to analyze the fall of each company during the Recession as well as the recovery up to current times.

To create a machine learning algorithm that can make predictions, it must be first 'trained' on known data. In case o Stock Pricing, historical prices of the stock data is taken and made train on the model. The parameters present in the data are:

Open High Low Close Volume ExDividend SplitRatio

and adjustment parameters:
Adj.High Adj.Low Adj.Open Adj.Close Adj.Volume

For a given stock the 'Open' and 'Close' values are the values of the stock at the opening and closing of the stock exchange. 'High' and 'Low' tell us the maximum and minimum value a stock has reached in a day. 'Volume' is the total amount of stock that was sold on that day. A company can decide to give dividend to its shareholders(ExDividend) or modify the number of shares that compose the company by splitting the shares(SplitRatio). After all these actions are taken, the values of the stock are adjusted. Hence, the Adjustment parameters.

These are the statistics for the historical stock data of Apple over period of January 1st, 2014 to December 31st, 2015:

Statistics	**Open**	**High**	**Low**	**Close**	**Volume**	**ExDividend**
Mean	207.7	209.4	205.9	207.7	4.4e+07	0.0185
Std	181.2	182.7	180.2	181.5	2.5e+07	0.2064
Max	649.9	651.26	644.5	647.35	1.9e+08	1.0

Min	90.2	90.7	89.7	90.28	5.7e+06	0.0

Statistics	**SplitRatio**	**Adj.Open**	**Adj.High**	**Adj.Low**	**Adj.Close**	**Adj.Volume**
Mean	1.01	101.8	101.8	100.9	102.7	5.7e+07
Std	0.27	17.8	17.71	17.55	17.9	2.6e+07
Max	7.0	129.3	128.0	126.9	129.4	2.6e+07
Min	1.0	66.4	67.0	66.1	67.2	1.3e+07

It should be noted that this data will be used in creating the mode to predict trends, however they won't be used in this form. Some amount of pre-processing is required to be done on the data.

Data Sources

Quandl API

The python Quandl API allows users to query historical stock prices from databases. This is used to gather the stock prices of companies in the NASDAQ 100.

Pandas stock price data reader

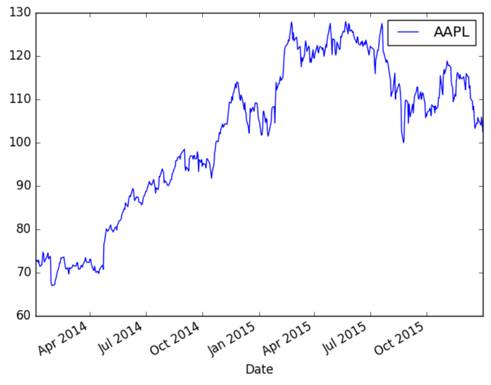
Historical stock price data from Yahoo Finance can be queried in python using the pandas module. Hence, we use this as well to gather data.

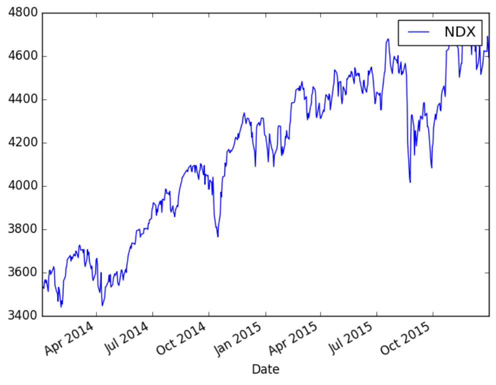
List of NASDAQ companies

The companies belonging to the NASDAQ 100 index are obtained through web scraping the NASDAQ website.

Exploratory Visualization

In the visualization below we can see the general trend of the Apple stock on the left. We can also see the general trend of the NDX index in the period of years selected for training. The visualization tells us that in the long term the trend of the stock is to increase and go up. However, our problem statement is predict the movement of the stock on a daily basis and not on long term basis.





Data format

The data obtained from the Quandl API is queried as a pandas DataFrame, The size of the dataset for each company for a single year's worth of data is 26kb.

Algorithms and Techniques

A classification model consists of a set of input features and output labels. For a classifier to be able to make classifications, it first needs to be trained on the input features. In order to carry out this process, some input data is fed in iteratively, known as 'training data' and predictions are made. Based on the misclassification, the classifier is updated to improve the prediction labels.

The models used in this project are:
Logistic Regression:
In case of logistic regression, a function is determined between the dependent variable(Variation of NDX) and independent variable(Features). The parameters of this function are modified in the training process in order to be able to predict one when the stock goes 'Up' and zero when the stock goes 'Down'. However, this is linear classifier. Hence, if our data is not linearly separable, we can never achieve 100% accuracy with this model.

Linear Support Vector Machines:
SVM draw a separation in parameter space that separates the two classes(in this case 'Up' and 'Down') and in the training process the separation line and the closest data is maximized.
Parameters:
C: The C parameter is penalization term in the cost function that helps prevent over-fitting.

Decision Tree:
A Decision Tree classifier classifies a data point based on a series of Yes or No questions asked to the values of input features. For example:

- Did the stock of Apple increase yesterday? Yes/No
- Was the Tesla highest stock greater than X yesterday? Yes/No

During the training process, the decision tree learns which of the input features are important and it learns what questions to ask to determine the threshold values.
Parameter:
Max Depth: Max Depth determines the number of 'questions' the tree will ask. Limiting max depth helps prevent over-fitting.

Random Forest:
A Random Forest is basically a collection of Decision Trees which are made to train on small subsets of input features and data points. The idea is to take a vote of classification from every tree then make the final decision.
Parameter:
Number of trees: With the increase in number of trees, the algorithm tends to over-fit.

Benchmark

As mentioned earlier, the metric for measurement of our model will be accuracy. In order to compare our model with something we need to create of Benchmark model. According to the Efficient Market Hypothesis it is impossible to predict the trend of the stock market. Hence, the benchmark Model for our algorithm to beat would be to make a prediction which better than random guessing. Hence a simple logistic regression model will be considered as the Benchmark Model. Such a model gives us a classification accuracy of 51 percent. Therefore, our objective is to create a model that can get a greater accuracy than 51 percent.

III. Methodology

Data Preprocessing

Missing Data:
Few reasons why the historical stock prices data of a company may not be available:

- The company did not exist at the time.
- The company did exist, however it's stocks were not traded in the stock market.
- The market in which the said company's stocks are traded was closed on the given day.

However, missing data can affect our machine learning model greatly in terms of training and accuracy. Hence, all the missing data points should either be removed or be filled with an appropriate value.
Both options will impact the machine learning model in different ways. In this project, only the data of the companies in who's data is available in Quandl is used. If a data point for a company on a particular day is missing, that data point is replaced with data from the previous day(Forward-fill method).

The companies whose data couldn't be accessed are:

- JD,NCLH: Non American company data is not available in Quandl.
- KHC: Kraft Heinz Company didn't exist in 2013. Kraft and Heinz merged in 2015.
- PYPL: PayPal was a subsidiary of eBay until 2015.

Feature Naming:
Since the data of all companies is queried from Quandl, they all have the same feature names such as 'Open' and 'Close'. In order to give the input feature of each company a unique identity, we simply add the 'ticker' symbol of each company to the corresponding feature. For example, the 'Open' feature for 'Apple' will be 'AAPL_Open'.

Splitting of Data:
To create a machine learning model we need a set of data for which we know the true output value of NDX. This set of data is used to find coefficients without changing the hyperparameters that help in minimizing the error between the prediction and true value. We then create another set known as 'Validation Set' which helps in creating a model which is as general as possible. This means that the model should show accuracy not just on the data it's been trained on but also on unseen data. Finally a 'Testing Set' is created which can be used to evaluate the performance of the model.

The catch is that the testing data must contain values posterior to the data present in training and validation set as it is not possible to train a model using data from future.

Final Features:
The objective is to predict the stock value of the NASDAQ 100 index. Therefore the idea is to use the stock prices of a company of day N to predict the stock value at day N+1 or N+x. As a result it is necessary to shift in time the NDX column compared to the features column.

We also need to decide on how much data prior to day N+1 we should use to train the classifier. Using all the historical data is not feasible, hence use a mock dataset to see how it affects the performance of our model.

We use the 'Open' feature for 2 companies X and Y for 2 previous days and see the performance as given in the table below:

Date	X Open Day N-2	X Open Day N-1	Y Open Day N-2	Y Open Day N-1	NDX Variation Day N
01/01/98	100	105	10	11	'Up'
02/01/98	105	110	11	12	'Down'

Through the report, the data used to train our ML model will be represented as follows:
Features: Companies=['A','B'], nHist= C, Stock Data=['D','E']
where:

- Companies is the list of companies used to train the classifier
- nHist is the number of days prior to the prediction used to train the classifier
- Stock Data are the features used to make the prediction in our classifier(Eg. 'Open')

As a result the size of the feature matrix is written as:
    Feature size: Nday x Nfeatures
where:

- Nday is the number of days available for training set
- Nfeatures is the number of features available for each day

The data used for training and cross validation set is the historical data of stock prices in period 2014-2015 which corresponds to 500 working days. Since 80 % of the data is used for training and 20 % data is used for testing, the testing dataset will have 100 working days.

Implementation

Basic Workflow:
This section talks about the sequence of actions taken to create a model that will predict how the NDX index will vary the next day:

1. Data collection
   - As data collection takes time, all of the data was acquired all at once and stored in csv
   - Get Quandl data

- Get NDX data
- Removing missing values
- Renaming features
- Creating new features like 'Variation'='Close'- 'Open'
- Merge the Quandl and NDX dataset
- Save the dataset as a csv

2. Parameter Tuning
   - Choose the desired data range for training, cv and testing
   - Choosing companies that can be used as features
   - Choosing the stock data to be used as features
   - Choosing the number of historical points used to make predictions
   - Choosing the right classification model
3. Data Preprocessing
   - Create and initialize an dataset object for training+cv and testing
   - Open the CSV file and read it as a pandas DataFrame
   - Check if the chosen companies are available in the chosen date for training(ignore if they are not present)
   - Create feature matrix:
     -Drop undesired features
     -Shift data to create historical data features
   - Randomly split the Train+CV dataset using seed in order to make the results reproducible
4. Training of Classifier
   - Use the training dataset to create a classifier
5. Score
   - Use accuracy metrics to get a classification score on Train, CV as well as Test set

Using Dataset:
As mentioned earlier, the test set consists of data posterior to the data present in train and cross-validation set. As a result our training and cv data will be up until 2015 and test data will stock prices on 2016. The reasons behind selecting this time period are:

- Train and CV set have to be prior to 2016
- To a train a model that generalizes well, the training set must contain enough data
- New companies get added to NASDAQ 100 every now and then. Hence, companies that got added during thi period will not be considered as features. Eg. Facebook which became an IPO in 2012
- 2008 was the year of the great financial crisis. Hence, the data of this year will differ significantly from other years. Hence, thi year is not considered.

Keeping all the constraints in mind, 2009-2015 is the time period selected for the training and CV set. As a result the dataset contains 1750 rows as there are 250 working day per year and we are considering data for 7 years.
Once this decision is made, the next decision is to decide the way to split the data into training and CV set.

- We could either make random split of the dataset into two sets
- Or we could select one time period for training and one time period for CV set

We then try these two strategies on our ML model: Random split between Train and CV set between 2009-2015 or train set consisting of data from 2009-2014 and CV set containing data of 2015.The results obtained are summarized below:

List of companies	All available companies
Number of days of historical data	1
Stock Data	All available stock data
Classifier	Logistic Regression
Feature Matrix Size	1760x926

The following are the results:

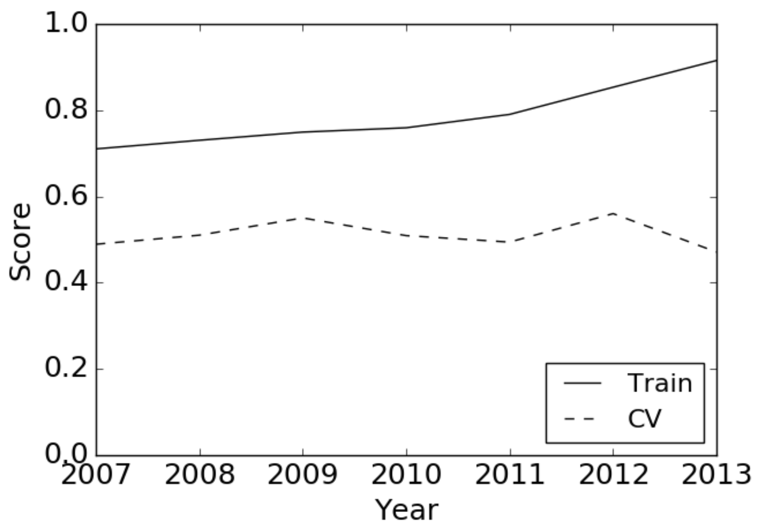
Strategy	**Random split**	**Segregated Spl**
Score on training set	0.749	0.550
Score on CV set	0.762	0.524

These results suggest that random splitting helps the our model train better than a segregated split. With this strategy the data received by the model is much more diverse, therefore helps to generalize better.

Length of the training dataset

We mentioned earlier the advantages of taking the data between 2009-2015 for train and CV sets. However, here we will look at a different time period and also vary the amount of data that will be fed in for training and create a visualization to see the change in accuracy.

List of companies	All available companies
Number of days of Historical data	1
Stock Data	All available stock data
Classifier	Logistic Regression
Train+CV set starting date	2007-2012
Feature matrix size	[502-2264][883-981]



Looking at the visualization we can understand that the CV score decreases to below 0.5 during the 2008 period. We also notice that the if the fewer data is supplied then the train score increases. This suggests overfitting.

After this analysis we realise that the best data to use for our model is 2009-2015 for train and CV dataset.

Number of days of historical data:
So far we have used the information available from just the previous day and tried to predict the behaviour of the next day. However, now we'll look at how the performance of our model changes by using more than just the previous day's data but also the N-1 and N-2 days data as well. We also reduce the number of companies used as features to 50. This helps create a dataset where the number of data points is greater than the number of features.

List of companies	All companies
Number of days of historical data	[1-3]
Stock Data	All available stock data
Classifier	Logistic Regression
Feature matrix size	1760x[578-1734]

The results are given in the following table:

nHist	**1**	**2**	**3**
Score on Training set	0.686	0.763	0.755
Score on CV set	0.530	0.465	0.530

These results are very variable and cannot be considered informative enough to draw conclusions. However, we do notice that the accuracy on CV set is better for the value of 1. Hence we consider nHist=1.

Choice of features
Different strategies to select the various features of stock data were tried:

- Making predictions with just one stock data and determining which gives the best results
- Making predictions with all stock data but one and determining which in which case the CV score was increased or decreased.
- Using custom set of stock data, for example using just the adjusted stock data using anything but adjusted stock data, using only stock data related to market closing

After training the model with different combinations of the above mentioned strategies we realise that it is difficult to isolate a particular data and use it for the model. We also realise that the number of features are not many, therefore we keep using all available data.

Choosing the classifier:
Here we will talk about the different classifiers that were tried along with the different hyperparameters that were chosen. We try to observe the performance and accuracy of the model for different classifiers and then determine the best classifier to use.

List of companies	All available companies
Number of days of historical data	[1]
Stock Data	All available stock data
Classifier	Various models
Feature matrix size	1760x1010

Score	**Training set**	**CV set**
Random Forest 100 trees	0.989	0.470
Random Forest 10 trees	0.762	0.524
Random Forest 5 trees	0.535	0.478
Decision Tree Max depth 3	0.619	0.559
Linear SVM, C=1	0.549	0.591
Linear SVM, C=0.5	0.580	0.591
Linear SVM, C=0.2	0.566	0.583
Linear SVM, C=1.0	0.473	0.413
Logistic Regression	0.751	0.555

Overall, the random forest with 10 trees or more seem to overfit the data and the decision trees with max depth 5 as well. The linear SVMs do not seem to suffer from overfitting issues but the results are very variable if we change the parameter slightly. The logistic regression model has a tendency to overfit when many features are used.

In the end, linear SVM with coefficient C=0.2 is chosen.

Prediction on the test dataset
In this section we test our fully optimized model and measure its performance. We using the testing dataset to make predictions and then try to measure the accuracy of the classifier.

List of companies	All available companies
Number of days of historical data	[1]
Stock Data	All available stock data
Classifier	Linear SVC, C=0.2
Feature matrix size	1760x1010
Test set accuracy score	0.544

We obtain an accuracy of 0.54. This is better than 0.51, which is the accuracy we got on the test dataset using Logistic Regression. This shows that we have been able to achieve our objective of beating the benchmark score.

Robustness of Model:

In this section, the influence of the Machine Learning model is tested.

Non linear models

- The SVR(kernel="poly") model could not be used as the computation cost was too high.
- The SVR(kernel="rbf") strongly overfitted the training dataset, this is probably due to the small number of data (540 for a 2 years period) compared to the number of features. The predictions on the test dataset were then very bad.

Linear models

The linear model used (Linear Regression, Ridge Regression, SVR(kernel="linear")) showed very similar results. The ridge regression was just slightly better than other models.

To make better predictions, it would probably be better to have access to much more data than daily stock prices. Some expensive databases content stock prices data with frequencies lower than a second. With this amount of data it would be possible to prevent the non-linear models from overfitting and yield better results.

IV. Conclusion

In this project we created a machine learning model to predict the trend of the stock market based on the NASDAQ 100 index. We created a pipeline to collect data, pre-process the data, split it into train, CV and test data, train a classifier using the acquired data and make predictions thereafter.

The time period chosen for training and CV set was 2009-2015 and testing set was chosen to be 2016. The choice of this period was very important. It helped us avoid the financial crisis of 2008 which otherwise would have affected our model very much due to its variation from stock data of other years.

We also looked at the fact that the number of preceding days used for training our model affected the performance by some margin. Increasing the number of days didn't improve the accuracy much. In fact, with the increase in the historical data we see the model tends to over-fit. We also tested the effect of each type of classifier and came to a conclusion that Linear Support Vector Machines is the best solution as it overfits the least.

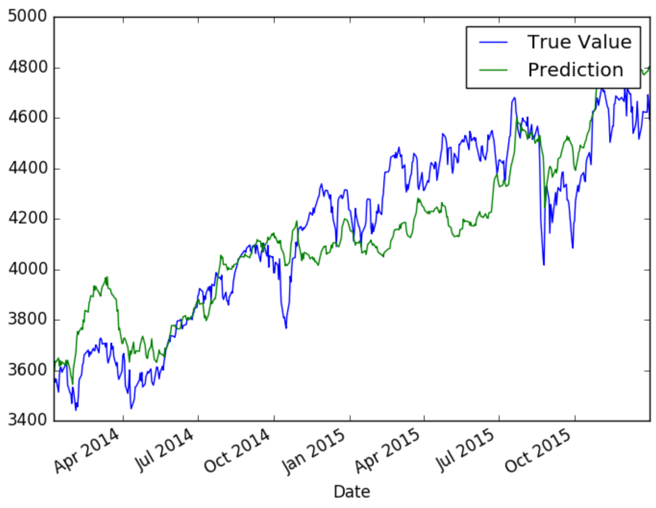
We also tested different combinations of features by excluding certain companies and also trying to train the classifier with the data of just one company. We concluded that it was not possible to isolate the stock data and hence decided to use all companies as features.

Another issue faced the variability in the results based on the way the data is split. Random splitting of data gave a better accuracy than the a sequential break-up of training and CV dataset. This was obvious as the random split gave our classifier a very diverse dataset, therefore it generalized better.

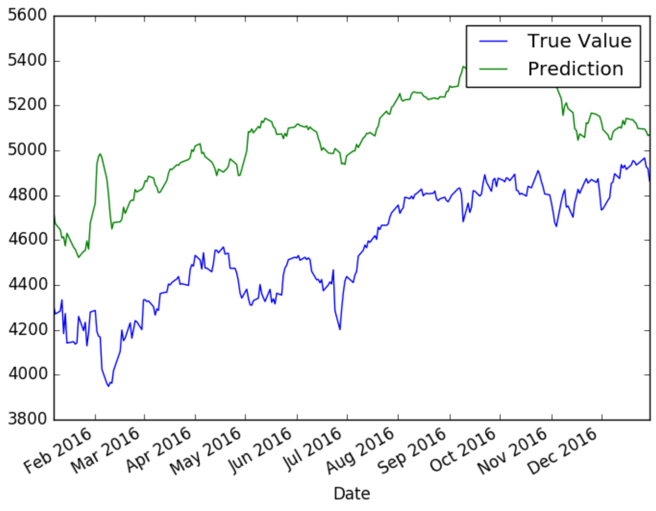
To improve the model further, it seems important to get access to more than one data point per day; however, most free financial databases do not offer that possibility. Another important feature that could be looked into is Sentiment Analysis. Using twitter, we could extract information about a particular company/product and determine whether positive or negative sentiment is attached with it. Based on this sentiment analysis we could try to predict whether a stock price will increase or decrease.

Free-Form Visualization

Results (using Facebook data as features) on the training set



Results (using Facebook data as features) on the test set



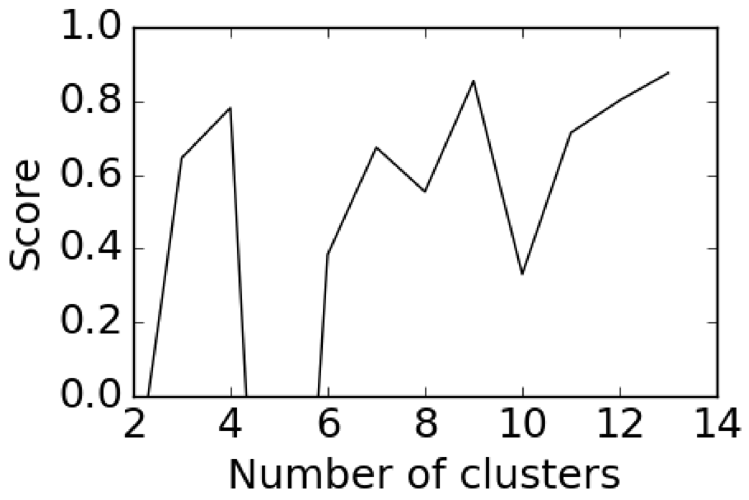
We observe that the predictions on the training set are better but the test set prediction are quite different from the true values.

Next day NDX prediction: Number of clusters

Using data of more than one company could improve the results, however including too many companies would increase the computational cost. Selecting the companies that give the best results on the single company test is probably not a good idea as this company might carry mostly the same information (Indeed the top 3 companies are all related to sales of electronic components). A better way to do it is to use a clustering algorithm to group similar companies and pick a company companies from each cluster to make predictions.

In this section the number of clusters is varied and we show the influence on the prediction results. Here the features "Open" and "Close" of the last 3 days are used for predictions.

Result on the test dataset

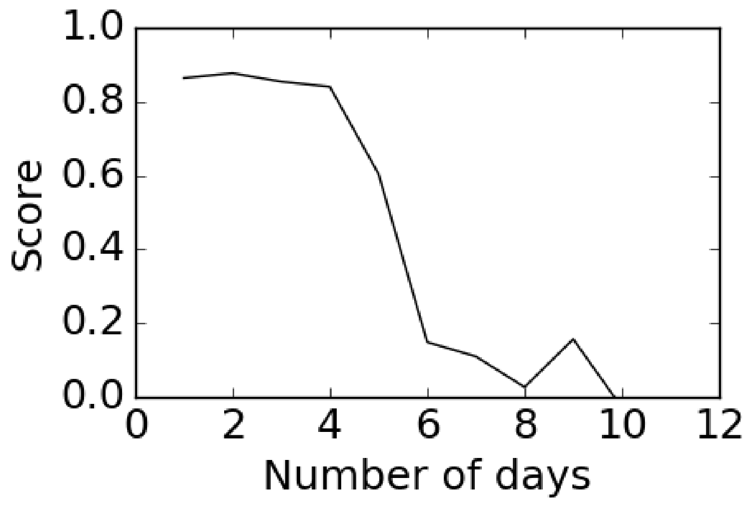


The graphs above show that the predictions on the test set are very variable, some have negative score (prediction very far away from expected result) and some yield better prediction score. As mentioned before, the clustering is very sensitive to the number of cluster chosen so that could explain the variability of the results.

Next day NDX prediction: Number of days of historical data

In this section, the influence of the number of days of historical data used for prediction is tested.

Result on the test dataset



It appears that using fewer days for prediction gives better results. The best prediction score is obtained using 2 days.