Udacity Capstone Project

Hugo Perrier, March 2017

# Introduction

## Stock Market Overview

The ownership of a company can belong to a single person but most of the time the ownership is divided between several shareholders. The values of these company shares depend of the total market valuation of the company, which may change over time depending on a variety of factors. Stocks or company shares can be bought and sold; there is therefore a market for company shares called the Stock Market. Stock trades are performed in stock exchanges such as:

- NASDAQ

- London Stock Exchange Group

- Tokyo Stock Exchange Group

Depending on the country they are based in, these stock exchanges have different opening days and times. Stock trades can only be executed during the opening hours of a stock exchange.

The most valuable companies in 2015 according to the FT500 ranking (link) are:

- Apple

- Exxon Mobil

- Berkshire Hathaway

- Google

- Microsoft

A good capacity to understand and predict movements in the stock market is crucial for investors to make profitable investments. To help investors choose which investment will be most profitable, they can use large amount of data on the history of the stock prices of companies. To process all these data, predictive models are created using machine learning. Machine learning models are "trained" to predict future stock prices based on a set of available features. This report explores how machine learning can be used to predict stock prices.

## Objective of the project

The NASDAQ 100 index is a stock market index related to the capitalization value of the 100 largest non-financial companies. The objective of this project is to create a predictive tool that uses machine learning to predict the daily variation of the NASDAQ 100 index (NDX) using historical stock prices of different companies. An investor could use the predictive tool as follows:

* Predict if the NDX will go up or down
* If “Up” is predicted buy NDX stocks
* If “Down” is predicted sell NDX stocks

Section 2 of this report describes the data used to create a machine learning model for stock prices prediction, section 3 describes the numerical models involved in the creation of a prediction tool and section 4 shows results of the predictive model built.

## Metrics

In this project, we propose to determine whether the NASDAQ 100 index will go up or down. This is a classification problem where the type of misclassification (predict “Up” when the stock value goes down or predict “Down” when the stock value goes up) does not matter. Indeed, if an investor buys a stock that subsequently loses value or if he sells a stock that subsequently gains value, he does not get the best possible return on his investment and there is not a situation that is obviously worse than the other. In this condition, using an accuracy score to measure how a predictive model is appropriate. Accuracy score is defined as the ratio of correctly labeled predictions over the total number of predictions and it is blind to the type of classification error.

A set of predictions and corresponding true labels are presented below. The misclassifications are colored in red and the correct ones in green:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | 01/01/2014 | 02/01/2014 | 03/01/2014 | 04/01/2014 |
| **Predictions** | Up | Down | Up | Down |
| **True Labels** | Up | Up | Up | Down |

In this example, one day was misclassified out of four, the accuracy score is thus ¾ = 75%.

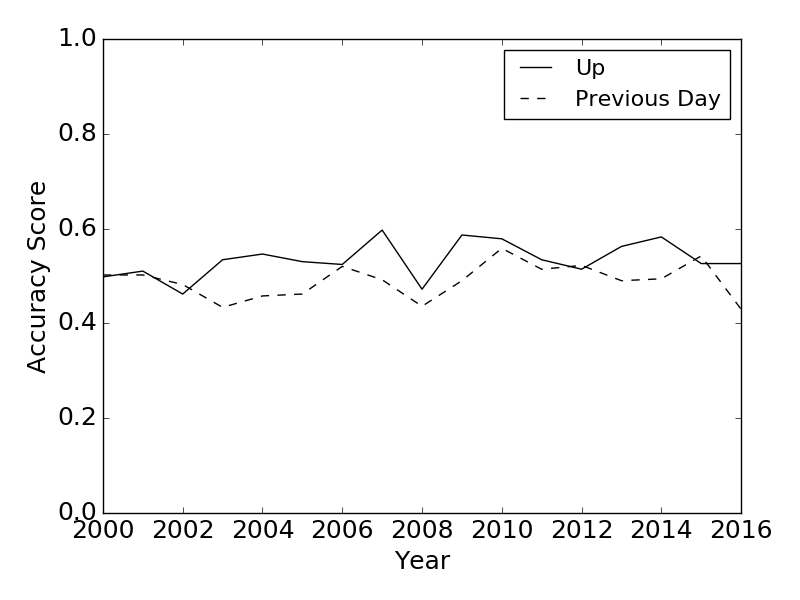
## Benchmark

As mentioned in section 1.3, the accuracy metrics is used to score the results of the predictive model we develop. To help with the interpretation and analysis of prediction results, a benchmark case is defined. The accuracy obtained with the different models developed in this project can then be compared to the benchmark score.

The maximum accuracy that can be reached is 1.0 (or 100%) when every prediction is correct. Since the prediction outputs are binary (“Up” or “Down”), randomly predicting the output would yield an accuracy score of 0.5 on average. An accuracy of 0.5 is thus the lowest acceptable accuracy for a predictive model. Two more options are tested for the benchmark case:

* Predict the same variation as the previous day all the time
* Always predict “Up” (NDX index value increases)

In the graph below, we calculate the accuracy score obtained at the end of each year by predicting either the same variation as the previous day (“Previous day”) or by predicting the NDX index to go up all the time (“Up”).



|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **Random (Theoretical)** | **Up** | **Previous day** |
| **Average** | 0.5 | 0.532 | 0.490 |

The “previous day” benchmark gets a score lower than 0.5, which could be explained by the oscillating behavior observed in the “Data visualization” section. On the other hand, the “Up” benchmark get a surprisingly high score, this could be explained by the general trend of the NASDAQ 100 index which goes up much more often than it goes down.

For the rest of this project, the “Up benchmark” is used as a reference and we aim to get an accuracy score better than 0.532.

# Data description

## Historical stock prices data exploration

To create a machine learning model to make predictions, it is necessary to first "train" the model using past data. In the context of stock market pricing, the model is trained using historical data of the stock prices. For example we can use data from the past period 2003 to 2005 to train a model and then use that model to make predictions about the future stock values. The stock price data consist of the following information:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Open | High | Low | Close | Volume | ExDividend | SplitRatio |

and adjusted values:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adj.Open | Adj.Close | Adj.Low | Adj.High | Adj.Volume |

For a given day, the "Open" and "Close" values are the values of a stock at the opening and closing of the stock exchange. The "High" and "Low" value are the maximum and minimum values that the stock has reached during that day. The "Volume" is the total amount of stock that was sold on that day. A company can decide to give dividend to shareholders ("Ex-Dividend" feature) or modify the number of shares that compose the company (adjusting the share value to keep the total capitalization value constant), this is described by the feature "Split Ratio". After this actions are taken, the values of Open, Close, High and Low are adjusted accordingly (Adj features).

All these features are of numerical type except “ExDividend” which is one if dividends are given away and zero otherwise.

A historical stock price dataset for a given company would typically look as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Open** | **Close** | **…** | **Adj.Volume** |
| **2000-01-01** | 10.0 | 11.4 | … | 1212121 |
| **2000-01-02** | 11.2 | 12.5 | … | 2001001 |
| **2000-01-03** | 12.2 | 12.4 | … | 1230010 |

The table below gathers statistics related to the stock prices of the company Apple over the period January 1st 2014 to December 31st 2015. This corresponds to a DataFrame with 504 rows (working days) and 13 columns (stock data).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 2.66e+08 |
| **Min** | 1.0 | 66.4 | 67.0 | 66.1 | 67.2 | 1.3e+07 |

It should be noticed that these data will used as training data to create a predictive model but not in this form. Data preprocessing will be necessary to create the features used by the machine learning algorithm to create the predictive model (discussed later in this report).

## Data Acquisition

**Quandl API**

The python Quandl API allows users to query historical stock prices from databases. With the free version only one data point per day can be accessed and there is a maximum amount of queries that can be performed in every 24h period. This is used to get the stock prices of the companies in the NASDAQ 100.

**Pandas stock price data reader**

Historical stock prices from yahoo finance can be queried directly using a pandas module. This is used to get the values of the NASDAQ 100 index.

**List of NASDAQ 100 companies**

To obtain the current list of companies included in the NASDAQ 100 index, web scraping is performed on the NASDAQ website.

## Data visualization

The graphs below show the evolution of the adjusted close price for the Apple stock price (left) and the NASDAQ 100 index (right) for the period 2014-2015.

|  |  |
| --- | --- |
| AAPL Adjusted Close price  (In US dollars) | NDX Adjusted Close  (Index value) |

On the graphs, we observe that the general trend is common to AAPL and NDX (An increase over the period) however over short periods of time (few days) the prices are very variable and seem to oscillate.

In this project, we want to predict daily variations of NDX therefore the short term oscillations will have much more impact than the trend over several months. It seems reasonable to limit the features used to make predictions to the data from the previous days (rather than months).

## Data Format

The data obtained from Quandl API queries are in a pandas DataFrame format. The size of a dataset corresponding one year of historical stock prices for a single company is about 26kb in size.

# Data Preprocessing

## Missing data

Historical stock prices for a company on a given day may not be available for the following reasons:

* The company didn't exist at that time (Google was created in 1998)
* The company existed but was not traded in the stock market (Facebook entered the stock market in 2012)
* The stock market where that company's stock is traded was closed on that day
* The historical stock prices of that company are freely accessible in Quandl API

However, to train a machine learning model, the datasets can not contain missing values. Two options are available to ensure the dataset does not contain missing values:

* Model the missing values
* Remove the datapoints containing missing values

Both options can impact the predictions therefore the way missing values are handled needs to be explained.

In this project, all the companies whose data were not accessible through Quandl were not included in the model. If on a given day some companies have stock price data but others don't, the missing data are replaced with the data from the previous working day (This is called a forward fill method).

Below is a list of data that couldn't be accessed using Quandl:

* JD, NCLH : Non American companies data are not available in the Quandl WIKI free database
* KHC : Kraft Heinz Company didn't exist in 2013, Kraft and Heinz merged in 2015
* PYPL : Paypal was a wholly owned subsidiary of eBay until 2015
* WBA : Walgreens Boots didn't exist in 2013

Changing the period used for the training set may require changing the list of companies used. This could cause problems for example if the predictive model is used to predict NDX of year 2016 but was trained using data from the period 2011-2015. Indeed, Facebook data that could be relevant in 2016 would not be used as features because of the missing data in year 2011. There is thus a tradeoff between increasing the range of the training period to have more data and limiting this range to avoid missing data from recent companies.

## Feature naming

When the data for different companies are queried from Quandl, they all have the same feature names; it is thus necessary to run a renaming operation when joining the datasets of the different companies. The company "ticker" symbol is simply added to the feature name: "Open" feature for "Apple" company (Ticker "AAPL") becomes "AAPL\_Open".

## Split of the data into train, cross validation and test datasets

To create a machine learning model we create a set of data called training set for which the true value of NDX are known. This set is used to find the model (with fixed hyperparameters) coefficients that minimize the error between predictions and true values of NDX.

Then a cross validation set similar to the training set is used to find the hyperparameters that make the predictions as general as possible. In other words, the model shouldn't just be able to predict NDX values from the dataset used for training but it should predict well NDX values for any other dataset.

Finally a test set is created to evaluate the performance of the model on data that were not used in the training process. The test dataset has to contain data posterior to the data in the training and cross validation sets as it is not possible to train a model using data from the future.

## Creation of the final features

The objective of a predictive model is to predict future values of the NASDAQ 100 index, therefore company stock prices of day N should be used to predict NDX value of day N+1 or N+x. It is thus necessary to shift in time the NDX column compared to the feature columns.

Finally, we need to choose which data are used as features to predict NDX, we can't use all of the historical data of every company prior to day N+1 to predict NDX on day N+1. It is therefore necessary to decide how many historical data to use for prediction, which features we want to use and adapt the dataset accordingly. A mock-up dataset that could be used in a machine learning model using just the "Open" data from the previous 2 days of companies "X" and "Y" is shown below (the date is written down to help with explanation but it is not used as a feature):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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” |

In the rest of the report, the features used to train a particular model will be written as follows:

Features: Company List = [“A”, “B”], nHist = C, Stock data = [“D”, “E”]

Where:

* Company list is a list of the companies whose stock data were used to train the classifier
* nHist is the number of days prior to the prediction used to train the classifier (ex. if nHist = 2 we use the data from the last 2 days)
* Stock data is the type of stock data used as input features, it could be any of the data presented in the section “Historical stock prices data exploration” (ex. “Open”, “Close”, “Variation”, …).

The size of the corresponding feature matrix is written as:

Feature matrix size: Nday x Nfeatures

Where

* Nday is the number of days available in the training dataset (number of rows).
* N features in the number of features available for each working day

In this project, data from the 2014-2015 period are used for the training and cross validation datasets, which corresponds to about 500 working days. Since 80% of the data in this period are used for training and 20% for validation, Nday is around 400.

# Numerical models

## Software requirements

The software requirements to build the predictive models are:

* python: Main programing language
* numpy: Package for scientific computing in python
* pandas: Package for data structures with python
* matplotlib: Plotting with python
* sklearn: Package for machine learning in python
* urllib: Package for data fetching across the web
* re: Package for regular expression operations
* Quandl: API to access historical stock prices from Quandl databases using Python

## Classification models

A classification model or classifier associates inputs (historical stock prices of companies) to an output label (NASDAQ 100 index "NDX" goes “Up” or “Down”). To correctly classify inputs, a classifier needs to be trained. For that, an iterative process is used where the model is fed a set of input data from a “training dataset”, predictions are made, the predictions are compared with the true label of these data and the model coefficients are updated to improve the classification. The

To do so, operations are performed on input features and result in a binary output corresponding to the labels “Up” or “Down”. The type of operations performed depends on the classifier used and parameters of the classifier set by the machine learning engineer.

The operations performed on the features to obtain a label depend on the classifier used.

A classification model or classifier relates a set of features (historical stock prices of companies) to a prediction (NASDAQ 100 index "NDX" goes “Up” or “Down”).

In Machine Learning, an optimization algorithm is used to find the model parameters that minimize the difference between the regression model output and the true value of the output on a "training" data set. Then the performances of different regression models (or similar models with different "hyperparameters") are compared on a different data set called "cross-validation dataset". This makes sure that the accuracy of model predictions are not limited to the training dataset but generalizable to other datasets (a model that does not generalize well to other datasets is said to "overfit").

The models used in the present work are:

* Linear Regression: parameters normalize (True/False)
* Support Vector Regression: parameters kernel (linear/poly/rbf)

## Data clustering

Each of the companies in the NASDAQ 100 has an influence on the value of the NASDAQ 100 index value but using the historical data from all these companies to create a machine learning model would have a high computational cost (270 working day per year per company = 270 datapoints per year, 100 companies \* 12 stock data per company per working day \* Nday (number of days of historical data used) = 1200 Nday features, input matrix has a size of 270\*1200Nday). It would then take a long time to train the models on a laptop.

To reduce the amount of data to work with companies that have similar behaviors can be grouped together. To do so, an unsupervised learning "clustering technique" is used: KMeans clustering. The Kmean clustering method takes as input the historical stock prices of all companies and a user defined number of desired clusters and outputs a list of companies in each cluster.

In practice, we calculate the daily variation for each datapoint:

"Variation" = "Close" - "Open" and use the "Variation" variable as input data for the clustering.

## Computational resources

The computer used for these calculations is a Macbook Pro 13-inch, Late 2011) with 8 GB of 1333 MHz DDR3 RAM and a 2.4 GHz Intel Core i5 processor. The linear regression models (linear regression + SVR(kernel="linear")) could be trained in a fraction of a second on a laptop with a 2 years long dataset for the training set, 2 features per companies and a few companies. On the other hand the training times for the non-linear models quickly become prohibitively long as the number of features and datapoints increases.

# Results and analysis

## Choice of datasets

In a section 3.3, we said that the full set of data should be split into a training set, a cross-validation (CV) set and a test set. The test set has to contain data corresponding to a period of time posterior compared to the training and CV sets. In this project we use the year 2016 for the test set. The choice of the time period for the train and CV sets is then influenced by the following factors:

* Train and CV set periods have to be prior to 2016
* To train a model that generalizes well to any dataset, the training dataset has to contain enough data
* New companies join the NASDAQ over time therefore if the training set corresponds to an old data periods, recent companies cannot be included as features. (ex. Facebook entered the stock market in 2012 so if the training dataset starts before 2012, Facebook stock data cannot be used as features).
* In 2008 there was a financial crisis so data from that year may strongly differ from other time periods

Based on these constraints, it was decided to use data from the 2009-2015 time period for the train and CV datasets. The amount of data available for this period is still limited (250 workdays per year \* 7 years = 1750 rows in the feature matrix) but the NASDAQ company list for that period is similar to the 2016 list which is the period we are interested in to make predictions. It also avoids including the data corresponding to the financial crisis of 2008.

Once the period for the train and CV sets have been chosen, there are two ways to split that data into training and CV sets:

* Randomly split data from the whole period into the two sets
* Choose two different time period for the train and CV sets

Here we compare results using these two strategies: Randomly split the 2009-2015 period or use 2009-2014 as training set and 2015 as CV set. The features used for to train the model 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 results are obtained:

|  |  |  |
| --- | --- | --- |
| **Strategy** | **Random split** | **Segregated split** |
| **Score on training set** | 0.749 | 0.550 |
| **Score on CV set** | 0,762 | 0.524 |

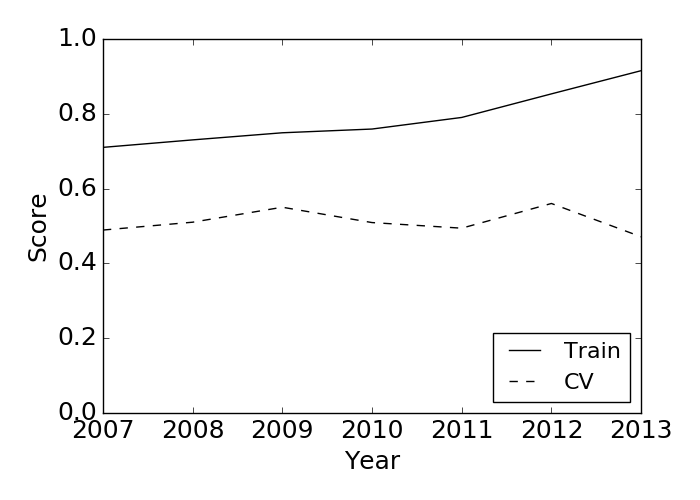
Using the random split strategy leads to better results on the CV set and less overfitting (difference between score on train set and CV set). With this strategy, the data in the training dataset are more diverse (they cover a longer period) so it seems realistic that it gives more generalizable results.

## Length of training dataset

In the previous section, the 2009-2015 period was used for the train and CV datasets. In this section we look at the influence of the dataset length on the prediction accuracy. For that we change the start year if the training set.

|  |  |
| --- | --- |
| **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]x[883-981] |

The results are shown in the graph below:



On the graph we see that if the training set includes data of the 2008 financial crisis (ex. Using range [2007-2015] or [2008-2015]), the cv score decreases below 0.5. We also observe very high train score if fewer data are present in the training set, this is probably due to overfitting. The best cv score are obtained using range [2009-2015] and [2012-2015].

In the rest of the project, the period [2009-2015] will be used for the train and cv datasets.

## Number of days of historical data

So far, we used only data from the previous day to make predictions about the next day NDX variation. In this section we try to use more days and check the influence on the cross validation accuracy score. It should be noticed that the number of features is doubled if we double the number of days of historical data used (nHist) so the total number of features can get very large. If the number of features is bigger than the number of rows in the feature matrix, strong overfitting and long calculation time can be expected therefore we limit nHist to 3 and we reduce the number of company used to 50 (This way the number of datapoints remains bigger than the number of features).

|  |  |
| --- | --- |
| **List of companies** | All available 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 |

The results obtained are very variable so it is not possible to draw conclusions on the influence of nHist on the cv\_score. However we see that the difference between train score and cv score is smaller when one day is used, the overfitting is reduced.

In the rest of this project, nHist = 1 will be used.

## Choice of stock data used for features

Several strategies to select the best combinations 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 adjusted stock data, using anything but adjusted stock data, using only stock data related to market closing…

After training models using many combinations of stock data as features, it was not possible to isolate stock data that perform better than others and overall using all stock data gave the most consistent results. When the number of feature is not too high, we will keep using all available stock data.

## Company clusters

We apply the clustering method described in the previous section for different number of clusters. The data used for the clustering corresponds to the stock prices data of the NASDAQ 100 companies for the year 2013. Results for nCluster = 10 are shown below:

**Composition of clusters:**

1. Analog Devices, Inc., Applied Materials, Inc., Broadcom Limited, Intel Corporation, KLA-Tencor Corporation, Lam Research Corporation, Microchip Technology Incorporated, Maxim Integrated Products, Inc., NVIDIA Corporation, QUALCOMM Incorporated, Skyworks Solutions, Inc., Texas Instruments Incorporated, Xilinx, Inc. |
2. Dollar Tree, Inc., Fastenal Company, Hasbro, Inc., Marriott International, Mattel, Inc., Mondelez International, Inc., Ross Stores, Inc., Tractor Supply Company, Ulta Beauty, Inc. |
3. American Airlines Group, Inc., Adobe Systems Incorporated, Akamai Technologies, Inc., Amazon.com, Inc., Baidu, Inc., Facebook, Inc., Liberty Interactive Corporation, Mylan N.V., Netflix, Inc., The Priceline Group Inc., TripAdvisor, Inc., Tesla, Inc., Yahoo! Inc. |
4. Alexion Pharmaceuticals, Inc., Amgen Inc., Biogen Inc., BioMarin Pharmaceutical Inc., Celgene Corporation, Gilead Sciences, Inc., Incyte Corporation, Regeneron Pharmaceuticals, Inc. |
5. Automatic Data Processing, Inc., Cognizant Technology Solutions Corporation, Monster Beverage Corporation, Paychex, Inc. |
6. Charter Communications, Inc., Comcast Corporation, Costco Wholesale Corporation, DISH Network Corporation, Liberty Global plc |
7. Activision Blizzard, Inc, Electronic Arts Inc., Intuitive Surgical, Inc. |
8. Apple Inc., Autodesk, Inc., Cerner Corporation, Cisco Systems, Inc., CSX Corporation, Cintas Corporation, Discovery Communications, Inc., Discovery Communications, Inc., Expedia, Inc., Fiserv, Inc., Twenty-First Century Fox, Inc., Twenty-First Century Fox, Inc., Alphabet Inc., Illumina, Inc., Intuit Inc., J.B. Hunt Transport Services, Inc., Micron Technology, Inc., O'Reilly Automotive, Inc., PACCAR Inc., Starbucks Corporation, Seagate Technology PLC, Symantec Corporation, Viacom Inc., Vodafone Group Plc, Verisk Analytics, Inc., Western Digital Corporation |
9. CA Inc., Check Point Software Technologies Ltd., Citrix Systems, Inc., eBay Inc., Express Scripts Holding Company, Microsoft Corporation, SBA Communications Corporation, Sirius XM Holdings Inc., T-Mobile US, Inc. |
10. Hologic, Inc., Henry Schein, Inc., Vertex Pharmaceuticals Incorporated, DENTSPLY SIRONA Inc. |

In this exemple, we can see that cluster 3 contains mostly tech companies, cluster 4 contains pharma companies and cluster 7 contains video game companies. Some clusters are more difficult to describe such as cluster 2 that contains hotel, food, beauty and variety store companies.

It should be noted that the clustering is not very stable, the following parameters significantly change the content of clusters:

* Time period of the input data
* Number of desired clusters
* Number of initialization (Kmeans clustering methods randomly initialize the center (in parameter space) of the clusters, the initialization might affect the final clustering so several runs with different random initialization are performed)

It is still a good way to reduce the amount of data we use to build the predictive model.

## Prediction of the Next day NDX variation: Basics

This section describes the general procedure followed to build a predictive model:

1. Clustering (Optional)

* All Nasdaq 100 Company 2013 stock price data are acquired
* Company that can't be accessed with Quandl are removed from the company list
* Missing values in the dataset are filled (forward fill)
* The "Variation" = "Close" - "Open" variable is calculated
* The desired number of company clusters is created using the "Variation" feature as input data.
* The first company (in alphabetical order) of each cluster is chosen to represent the whole cluster.
* A list of selected companies is saved to be used in the predictive model creation.

2. Predictive model

* Stock price data of the companies chosen in the clustering process are acquired. The 2014-2015 period is used for the training/cross validation datasets and the 2016 period is used for the test set.
* Missing values are filled (forward fill)
* The "Variation" = "Close" - "Open" variable is calculated. (It can be used as a potential engineered feature)
* NASDAQ 100 index data are acquired from yahoo finance (using pandas finance data reader).
* The features to be used to build the predictive model are chosen:
* Type of stock price data (ex. ["Open", "Close"] or ["Open", "Variation", "Close"])
* Number of days used for prediction (ex. data from the last 3 days, data from the last 7 days, ...)
* The final feature matrix is created (drop undesired features, create previous days features, give the feature matrix the right shape, etc)
* Choose a machine learning regression model
* Train the model using the training data
* Predict NDX and score the model on the train, cv and test sets

## Prediction of the Next day NDX variation: Single Companies

In this section, we use stock data from a single company as input feature to train our logistic regression classifier. We repeat the process with each company from the list of companies used to establish the NASDAQ 100 index. This way we determine if some companies are more useful than others to predict the NDX variation.

Features: Company List = [“X”], nHist = 1, Stock data = [“Variation”]

Where X covers sequentially all companies in the list of NASDAQ companies.

Feature matrix size: Nday x 1

|  |  |  |
| --- | --- | --- |
| **Features** | **Train Score** | **CV Score** |
| Open, Close |  | 0.191118 |
| Var, Close |  | 0.188922 |
| AdjOpen, AdjClose |  | 0.266558 |
| AdjOpen, AdjClose, Ex-Dividend |  | 0.266558 |
| AdjOpen, AdjClose, AdjLow |  | 0.515742 |
| AdjOpen, AdjClose, AdjHigh |  | 0.241427 |
| AdjOpen, AdjClose, Split |  | 0.260417 |

ULTA,0.555328,0.542169

JBHT,0.539554,0.538153

TMUS,0.550102,0.538153

MAT,0.545817,0.536000

CHTR,0.549098,0.532000

## Prediction of the Next day NDX variation: Features

In this section, the influence of the choice of features on predictions is tested. A linear regression model is fitted to the data of the companies Amazon and Netflix from the last 3 days.

|  |  |
| --- | --- |
| **Features** | **Score** |
| Open, Close | 0.191118 |
| Var, Close | 0.188922 |
| AdjOpen, AdjClose | 0.266558 |
| AdjOpen, AdjClose, Ex-Dividend | 0.266558 |
| AdjOpen, AdjClose, AdjLow | 0.515742 |
| AdjOpen, AdjClose, AdjHigh | 0.241427 |
| AdjOpen, AdjClose, Split | 0.260417 |

The results presented in the table show that the best prediction score is obtained when adjusted variables are used. In particular, using just the "AdjOpen" and "AdjClose" features give the best result, adding more features do not increase the prediction score. For the rest of the report the features "AdjOpen" and "AdjClose" will be used.

## Next day NDX prediction: Single company data

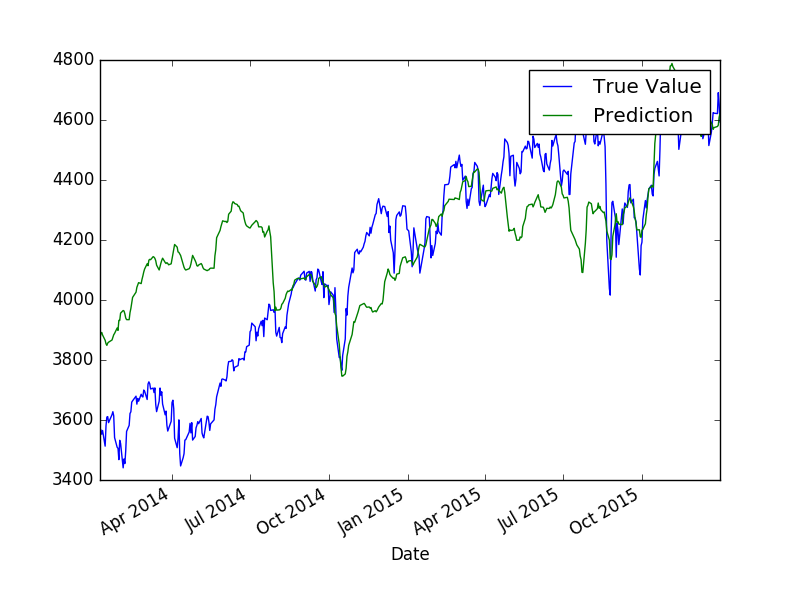
In this section a linear regression model is fitted using "AdjOpen" and "AdjClose" data from the last 3 days for a single companies. The table below shows the best prediction results on the test dataset.

|  |  |
| --- | --- |
| **Company** | **Score** |
| Maxim Integrated Products, Inc. | 0.823026 |
| Microchip Technology Incorporated | 0.796815 |
| Cisco Systems, Inc. | 0.706128 |
| Lam Research Corporation | 0.656457 |
| T-Mobile US, Inc. | 0.607240 |

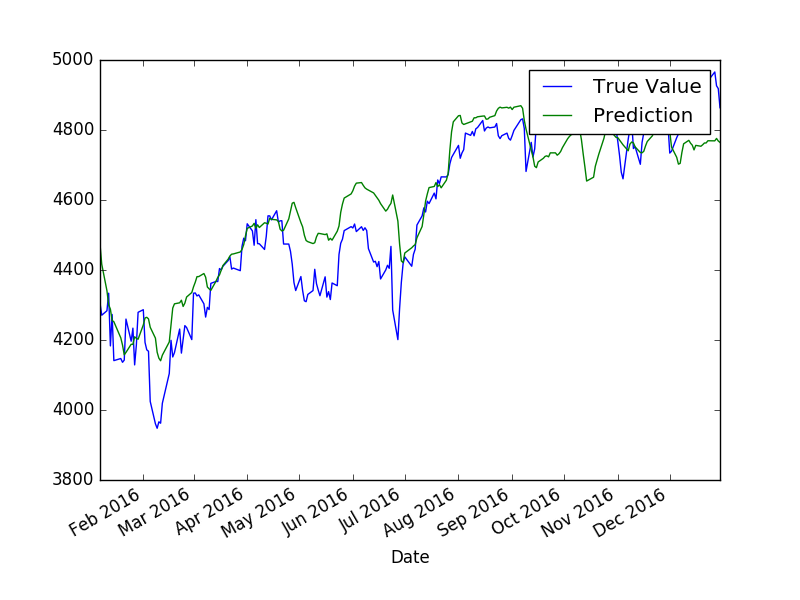
The company that yields the best prediction data is Maxim Integrated Products:

"Maxim Integrated is an American, publicly traded company that designs, manufactures, and sells analog and mixed-signal integrated circuits." [(Wikipedia)](https://en.wikipedia.org/wiki/Maxim\_Integrated)

**Results (using Maxim Integrated Products data as features) on the training set**

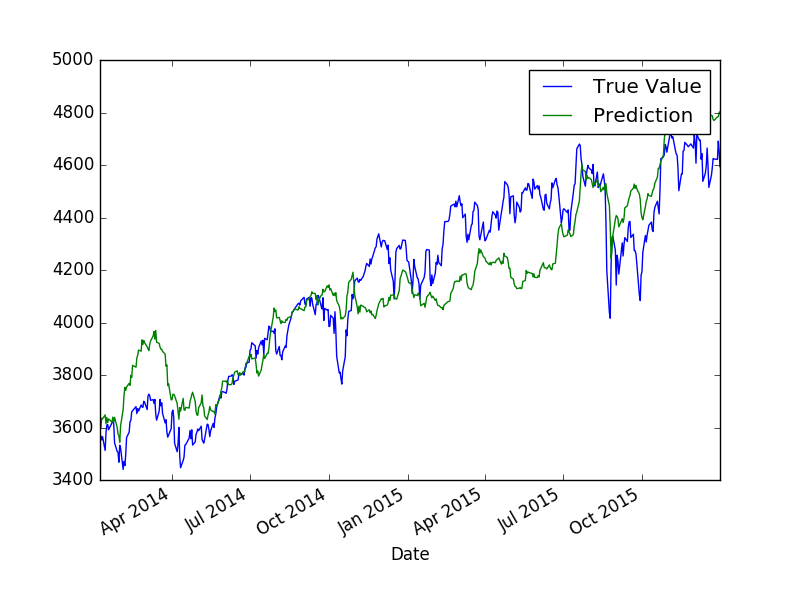


**Results (using Maxim Integrated Products data as features) on the test set**

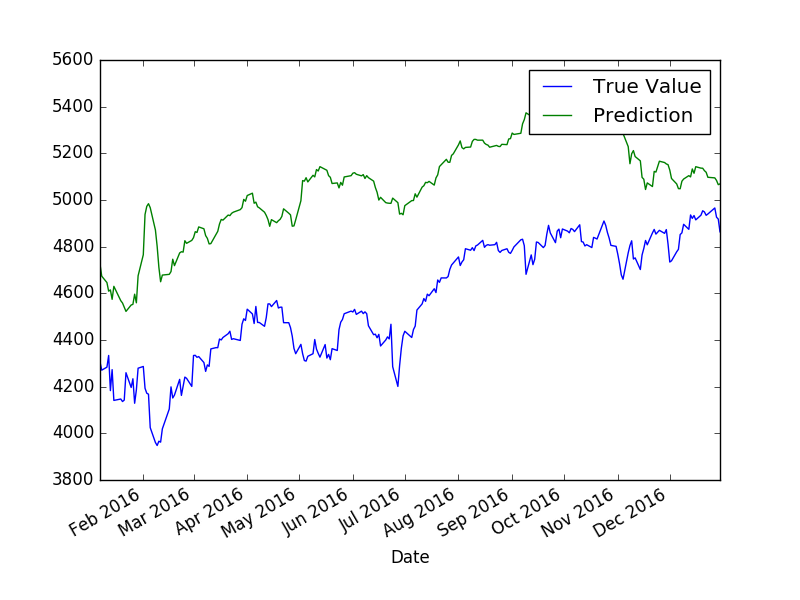
****

It appears that the predictions on the training set are not so good but predictions on the test set are much better. For comparison, the results for the company Facebook that gives much lower results on the test set (Negative score) are shown below:

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



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



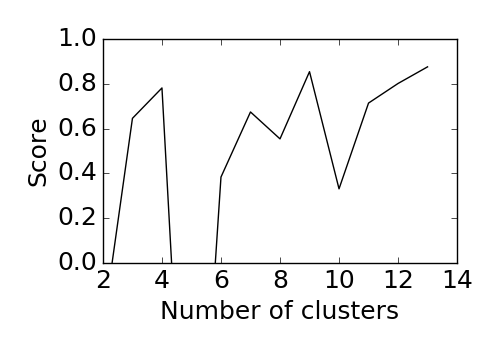
We observe that the predictions on the training set are better but the test set prediction are quite different from the true values.

## 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**

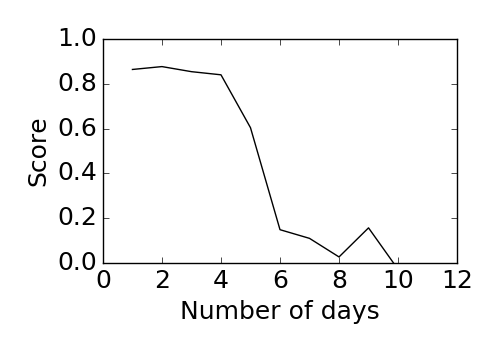


The graphs above show that the predictions on the test set are very variable, some have negative score (prediction very far away from expected result) and some yield better prediction score. As mentioned before, the clustering is very sensitive to the number of cluster chosen so that could explain the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

## Next day NDX prediction: Machine Learning 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.

# Conclusions

In this work a machine learning model was set up to predict future value of the NASDAQ 100 index using historical stock prices of NASDAQ companies. A pipeline to acquire data, preprocess data, create training, cross validation and test data sets, fit the machine learning model and make predictions was set up.

It was shown that using Adjusted data as features makes a big difference in the prediction score. Then predictions were made using data from just one company and it was shown that high prediction scores can be reached. Using the stock prices data from the company "Maxim Integrated Products" gave the best results.

A clustering strategy was tested to group similar companies and to use just a few companies to represent all NASDAQ 100 companies when doing predictions. The company clustering did not improve the prediction results compared to using just a single company to make predictions. Another parameter that was tested is the number of days of historical data used to make predictions. It was shown that using just the 2 or 3 previous days to make predictions gives the best prediction results.

Finally, different machine learning models were tested and the linear regression models gave the best results. It was argued that the limited number of datapoints (just daily stock prices) caused the non-linear models to overfit and that access to more data could improve the results.