Machine Learning Engineer Nanodegree

Capstone Proposal

Vadzim Dabravolski
June 6, 2017

Proposal

In my capstone project, I'd like to explore a possibility to forecast stock price trend using machine learning techniques (such as word vector embeddings and Recurrent Neural Networks).

The business problem is certainly not new and many smart people approached it before me as well as are trying to solve right now. A significant analytical and math apparatus was developed along the way. However, as of now there is no uniformly accepted best approach. And while data mining & machine learning are certainly giving new opportunities to predict stock movement, many stock traders are still relying on people skills and intuition.

I'm skeptical if this project can turn things around, however, I still feel that this may be an interesting exercise. As my current full-time job is closely related to investment banking and stock trading, I'm looking forward to apply skills which I hope to gain to solve more narrow and practical use cases in future.

Domain Background

There are several most common approaches to do stock market predictions (wiki):

- **Fundamental analysis** involves experts who analyze the company and industry performance in comparison with others. This type of analysis is more focused on the company (performance indicators, strength and weaknesses of leadership, market share dynamics etc.) than on the studying the stock behavior itself. The idea behind is to find over- or under-valued companies on the market and leverage this knowledge.
- **Technical analysis** trading school tries to predict stock movements based on certain observed trading patterns and behavioral dynamics ("head and shoulders" strategy, stock momentum, mean reversion and others).
- **Data mining** which to some extend combines all available pieces of data about company, stock and market, and trying to capture hidden patterns and connections between different pieces. Various datasets and technical approaches have been suggested (see Reference section for some examples).

In this proposal, I will use data mining approach. Based on my domain knowledge I hypothesize that in most common case following factors influence on the company stock price:

1. Overall stock market trends (“bullish” vs “bearish” market);
2. Trends in specific industry sectors (e.g. decline of classical retailers such as Sears, Macy’s because of Amazon);
3. Known and hidden trading patterns on the market;
4. Companies fundamental parameters reported quarterly (such as profits, revenue, cost of revenue, market capitalization etc.);
5. Public sentiment of specific company (e.g. public company bummers, M&A, rotations of C-staff etc.);

Of course, the market can be manipulated using one of the known illegal schemas (like “pump and dump”) as well as trading based on the insider information. For the sake of my project, I assume that these manipulations are neglectful.

Problem Statement

Develop machine learning algorithm (or ensemble of algorithms) which will recommend whether to buy or sell stock price for given company based on publicly available data such as stock prices, company performance and company news. The algorithm will be trained for industry leading companies with high market capitalization.

Datasets and Inputs

My choice of the data sets is driven by hypothesized factors in the “Domain background” section and should represent these factors in some form:

- Trading patterns and trends are represented by historic stock prices (provided by Quandl);
- Public sentiment is represented by company and industry news (provided by Reuters, Google and Yahoo Finance);
- Fundamental parameters of companies are represented in Earnings reports published by SEC (available via this API).
- General company information is captured in company profile (available on Yahoo Finance)

Fundamental data

For fundamental analysis I merged two separate datasets:

- Dataset retrieved from usfundamentals.com which contains company’s SEC filings from 2010 to 2016 years;
- List of S&P500 companies including company ticker and industry sector.

The resulting dataset contains all available SEC filings for companies in the S&P index list.

```
In [68]: df_sp500.sample(n=10)
```

Out[68]:

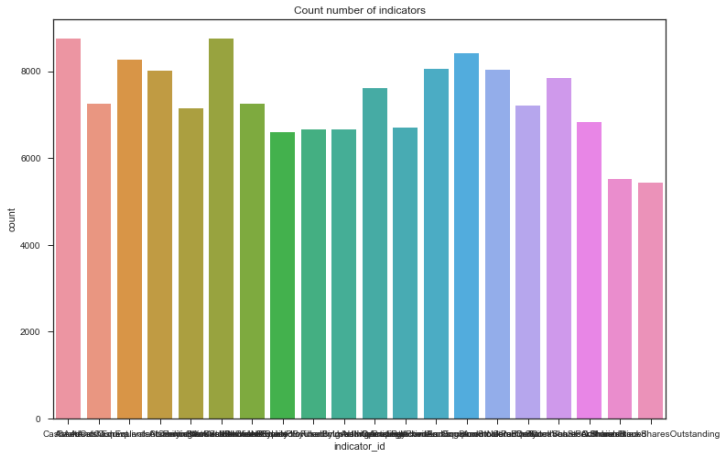
	company_id	indicator_id	ticker_id	sector_id	sector	ticker	year	value
49146	8670	CommonStockValue	11	6	information_technology	ADP	2011	6.390000e+07
82569	1060391	CommonStockSharesIssued	401	5	industrials	RSG	2013	4.110000e+08
53761	1013871	CommonStockSharesIssued	338	10	utilities	NRG	2012	3.991126e+08
67327	39911	LiabilitiesAndStockholdersEquity	208	0	consumer_discretionary	GPS	2012	7.470000e+09
35380	1492633	NetCashProvidedByUsedInFinancingActivities	335	5	industrials	NLSN	2011	-2.500000e+08
27273	1013871	PropertyPlantAndEquipmentNet	338	10	utilities	NRG	2011	1.362100e+10
44979	72971	NetCashProvidedByUsedInInvestingActivities	480	3	financials	WFC	2011	-3.504400e+10
153470	813828	StockholdersEquity	83	0	consumer_discretionary	CBS	2015	5.563000e+09
175681	63754	NetIncomeLoss	306	1	consumer_staples	MKC	2016	4.723000e+08
4843	1110803	NetCashProvidedByUsedInFinancingActivities	240	4	health_care	ILMN	2010	1.164740e+08

Sample of merged dataset

The resulting dataset has following caveats:

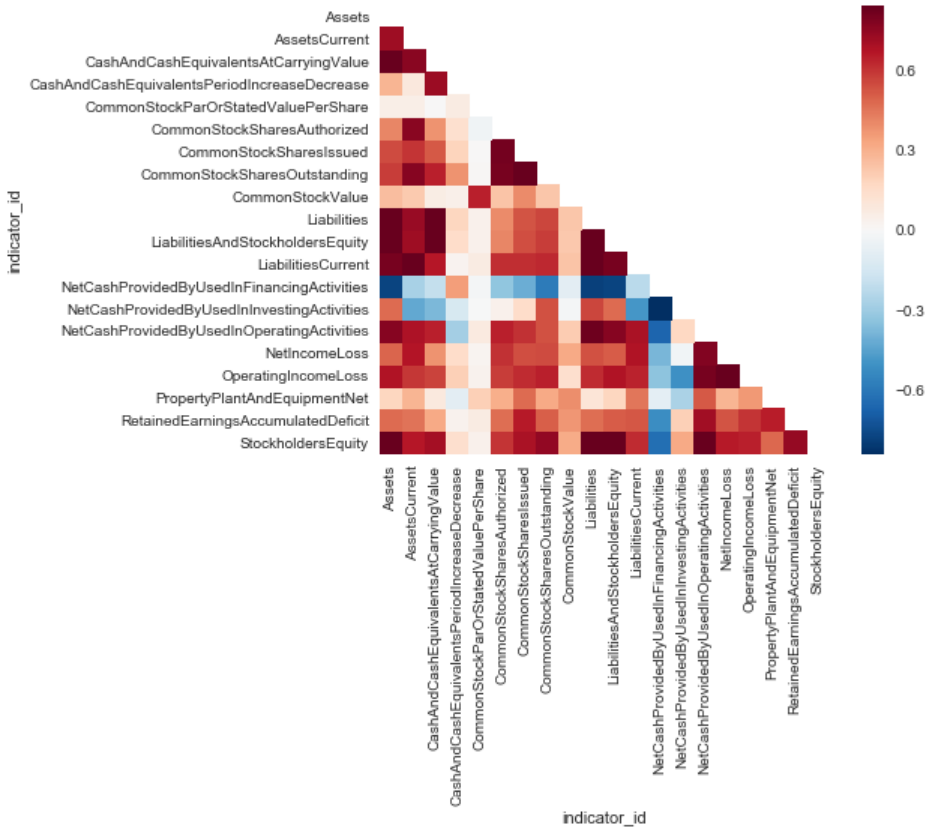
- Collection of company filings has been rolled out gradually, starting 2010. Therefore, not all companies have electronic filings in 2010 and 2011.
- Filings format and company indicators varies from company to company as there is no mandatory set of indicators which company have to file.

The later implies that it's reasonable to select only the most popular indicators and use them for training purposes. Below is frequency histogram of 20 most popular indicators.



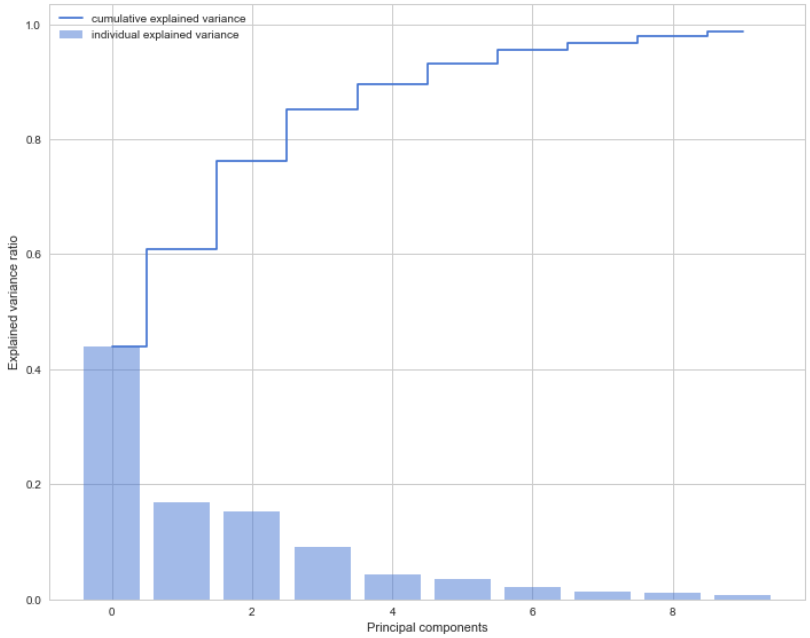
Number of records for top-20 indicators

Let's take a look on correlation between various indicators in the dataset. For this purpose, I selected a specific year (2015) and pivoted the data.



Correlation between top-20 indicators in 2015

As expected, many of the indicators are highly correlated. That gives us an opportunity to reduce data dimensionality without scarifying accuracy of our model. To prove it, let's use vanilla PCA and review how many components we will can use without losing much of information.



PCA analysis and explained variance for top-20 in 2015

As you can see from the chart, we can use only 4 PCA components and still have ~90% explained variance.

Historic stock prices data set

To being able to train in principle the machine learning algorithms, first let's prove that stock data has some meaningful patterns inside which can be captured by machine learning models.

For this, let's consider stock prices as complex system, which reacts to certain external conditions. In general, there are 3 types of the systems:

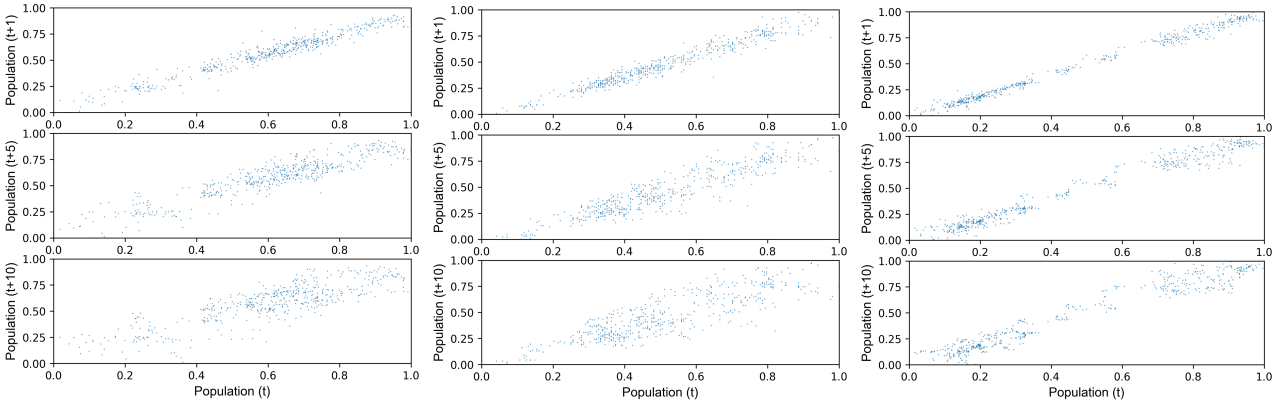
- Deterministic system – future state of the system is determined by only current state and inputs;
- Random system – future state of the system is random and doesn't depend on current system state and inputs;
- Chaotic system is characterized by extremely high sensitivity to inputs.

Apparently, stock prices is not deterministic system, otherwise, the original problem of this paper would be fairly trivial. Let's understand whether we can consider stock prices as random or chaotic systems (with some level of random noise). While random system cannot be predicted (we can only estimate probability of certain event), the chaotic system can be theoretical predicted.

Phase diagram

Let's confirm that stock prices can be considered as chaotic system. The simplest approach is to check if there is phase correlation between initial stock price and the price in next couple of days. For this purpose, I created a phase diagram for three companies with different industry profiles.

Here, under phase diagram I understand a scatter plot for system state at given time x_t and lagged in time system state $x_{(t+\tau)}$. Time lag τ has following values: [1, 5, 10]



Phase diagrams for Apple, Amazon and Marriott stocks

Visual analysis confirms that in most cases dependency between x_t and $x_{(t+\tau)}$ has linear characteristics and is not random. As we increase the time lag, the correlation becomes weaker.

Measuring sensitivity to initial conditions

Analytical way to confirm chaotic characteristics of the system is to calculate Lyapunov Exponent. This approach is based on a fundamental assumption of chaotic systems that the distance of two close states of the system over time will grow exponentially for large data population:

$$\Delta x_n = x_0 e^{n\lambda}$$

Lyapunov Exponent is calculated as:

$$\lambda = \lim_{n\to\infty} [\ (1/n)\ \ln(\Delta x_n\ /\Delta x_o)\ ].$$

Below is Lyapunov Exponent, calculated for stock prices:

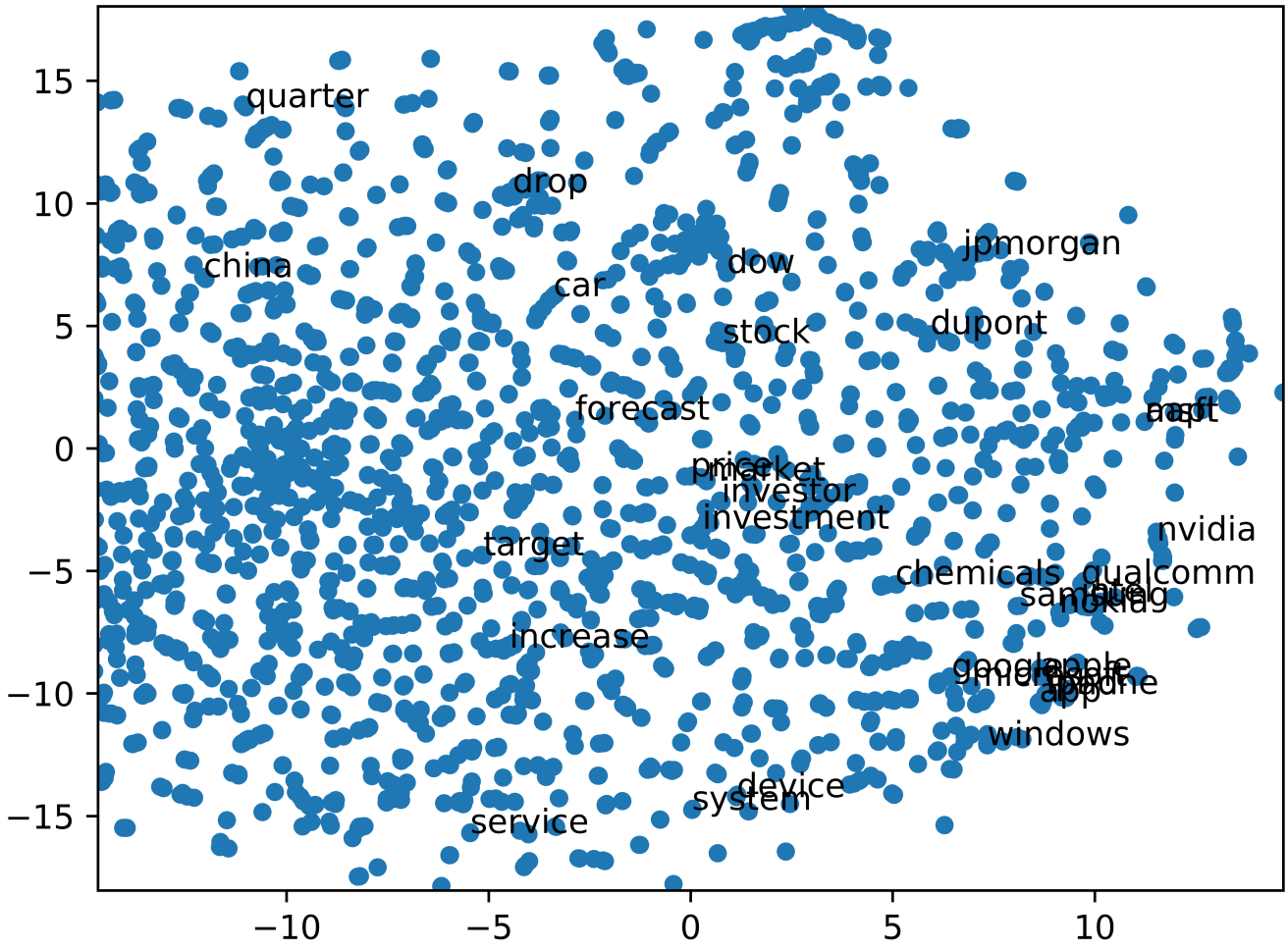
Company	Lyapunov Exponent
Apple	0.066006413154434784
Amazon	0.031947105071123895
Comcast	0.021156407892704024
Ford	0.023073254970082044
Marriott	0.019663767979487035
Chevron	0.036850700462073657

The positive value of Lyapunov Exponent indicates chaotic behavior of the stock time series and theoretical predictability.

Company News dataset

One of the most common techniques to do machine analysis of natural languages is word embeddings where each word of vocabulary is represented by vector of real numbers. Typically, each word vector (embedding) has hundreds of elements. Useful property of word embeddings is that words with similar meanings have vectors located closer in embedding space. That property allows models to “understand” and “compare” word meanings.

To illustrate that similar words are located closer in embedding space, let’s use news corpus collected from news sites (Bloomberg, SeekingAlpha, Reuters) and build t-SNE scatter plot for selected words. t-SNE technique allows to represent vectors of high-dimensional embeddings space in 2D or 3D spaces.



t-SNE embeddings for company news

From scatter plot one can see that words with similar meaning are located closer to each other. The following examples of co-located words can be observed:

- related terms “market”, “investment”, “investor”;
- companies “google”, “apple”, “microsoft”;
- tickers “aapl” and “msft”.

Solution Statement

My solution will solve the binary classification problem: for a given company to predict whether the stock price will grow or fall on next day. This should allow traders to choose between short position (if stock is believed to decline) and long position (if stock is believed to grow).

The envisioned solution has following simplifications and limitations:

- Only fact of increase or decrease of stock price will be forecasted (not the size of the increase/decrease).
- The intraday movements of the stock won't be considered.
- The price at market close will be assumed next day market open price.

From implementation perspective, the solution is an ensemble of machine learning algorithms which designed to address various factors:

- Public sentiment and its influence on the stock will be modeled using Convolution Neural Network (CNN) and word (or document) embeddings.
- Stock trends will be modeled using Recurrent Neural Networks (RNN).
- Fundamental company parameters will be models via RNN.

Benchmark Model

Historically, the most commonly used approach for stock model validation is backtesting, i.e. validation of model is happenings based on the past stock performance. I think the same approach can be used in my case as well.

In addition to backtesting I'd like to use ARIMA model, which is used for pure time series forecasting (connected to technical analysis school of thought). This model is rather generic and didn't account for any information other than structure of the time-series itself. The implementation of ARIMA model is available in statsmodel Python package.

Evaluation Metrics

Since we are solving classification problem, it's useful to visualize incorrect and correct predictions using contingency matrix:

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Figure 1 – Contingency matrix (source)

Given contingency matrix the following metrics can be useful for evaluation purposes:

- Binary accuracy. It's the most intuitive metric, but it performs poorly if we have not balanced classes (which I believe is the case of market stock which generally grows), so should be used with caution. Accuracy lies within [0,1] range.

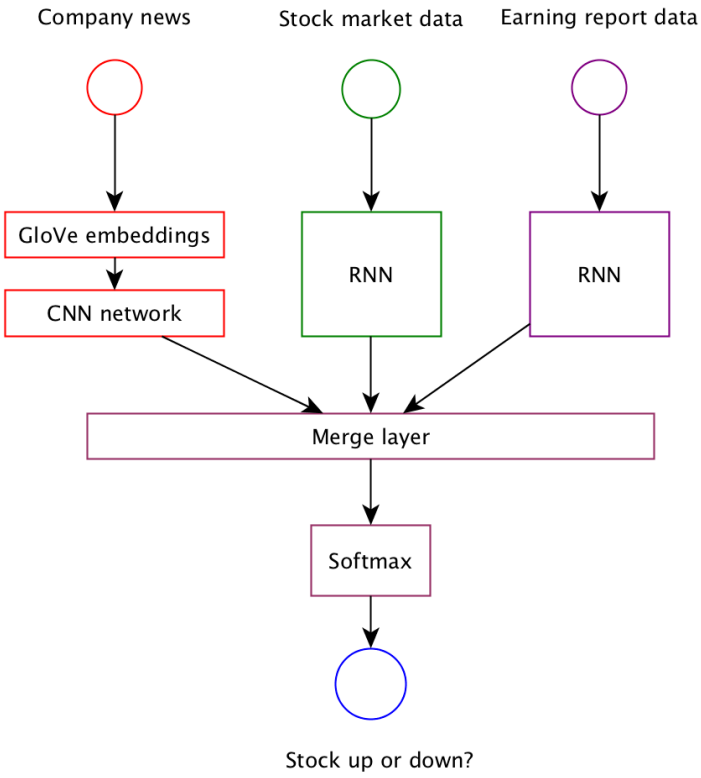
$$ACC = (TP + TN)/(P + N)$$

- Mathew's correlation coefficient (MCC). It addresses the flaws in accuracy and can be used for cases of imbalanced classes. MCC lies within [-1, 1] range.

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Project Design

Like I mentioned above, the idea behind this proposal to build a complex machine learning model which will account for various factors which influence on stock movements. As the different factors has very different dataset (different sources and internal data structure), I propose to select a several separate models which will be assembled together to make a final prediction.



Model selections

- **Word embeddings.** I spent some time choosing between word2vec and GloVe embeddings. Based on some of the published cases, both embeddings seems to have similar performance. So I chose GloVe as it's a bit easy to parallelize in case of training your own embeddings. Currently I'm planning to try two options:
    - Using pre-trained GloVe weights (trained on wiki data).
    - Train GloVe on selected company news. The reason why I may choose this option is that I'm expecting some confusion in wiki embeddings in such cases as "Apple" (company) and "apple" (fruit).
- **CNN for company's news feed**. I considered this model for following reasons:
    - Faster comparing to RNN.
    - As embeddings is effectively a 2D problem, CNN demonstrated effectiveness in working with such types of data (learning high-level features which in case of word embeddings can be associated with meaning of the phrase)
- **RNN for stock market data and earning report data.**
    - Non-linear, robust to noise, can handle multivariate data.
    - Can learn the temporal context of the sequence. So this avoid creating a fixed lag data sets.

Technology stack

To keep it simple, I will use later keras 2.0 as the main machine learning library. As of now it supports of the required functionality and yet provide an elegant API. I will use GloVe for word embeddings.

References

As part of work on this proposal I reviewed available literature and existing solutions. Some of them I found valuable, specifically:

- "Deep Learning for Event-Driven Stock Prediction" by Ding et al.
- "Market-Timing Strategies That Worked" by Pu Shen
- "Twitter mood predicts the stock market" by Bollen et al
- "Machine Learning Strategies for Time Series Prediction" by Bontempi
- "Financial time series forecasting with machine learning techniques: A survey" by Krollner et al.
- "Deep Learning for Time-Series Analysis" by Ganmoa
- https://github.com/edenbaus/Event-Driven-FinModel
- https://github.com/WayneDW/Sentiment-Analysis-in-Event-Driven-Stock-Price-Movement-Prediction
- https://github.com/llSourcell/How-to-Predict-Stock-Prices-Easily-Demo
- http://textminingonline.com/getting-started-with-word2vec-and-glove-in-python
- https://medium.com/@thoszymkowiak/how-to-implement-sentiment-analysis-using-word-embedding-and-convolutional-neural-networks-on-keras-163197aef623