Predicting stock prices for large-cap technology companies

Abstract

The goal of the project is to predict price changes in the future for a given stock. Information that is leveraged to make these predictions includes prices from previous days and financial news headlines related to the company of interest.

The final model outputs a classification (1 for + or 0 for -) of the sign of next day's predicted price change. The model implemented achieved 59% accuracy and an annualized return of about +16% on the example stock used (Facebook Inc. with symbol FB).

Introduction

Personally I've always been fascinated by how seemingly unpredictable the stock market can be. The price changes resulting from a mixed reaction of general market sentiment, the company's financial news, future prospect of the company and the industry in general, and sometimes whispers discussing the company's recent performance are hard to rationalize reliably given incomplete information. This project attempts to capture parts of this causal relationship using partial information.

A number of supervised learning algorithms were explored in an attempt to predict future stock prices using past prices and news headlines. This idea is based on the real-world observation that stock prices often rebound after heavy losses or self-correct after consecutive gains, and that surprising news or analyst comments often have large impact on stock prices.

Related Work

Some of the past attempts to predict stock prices tried to fit a single model that would apply to all stocks. These approaches failed to recognize one critical fact that not all stocks behave the same

way. For instance, the stock price of a small-cap cosmetics retailer would behave very differently from that of a large-cap technology company.

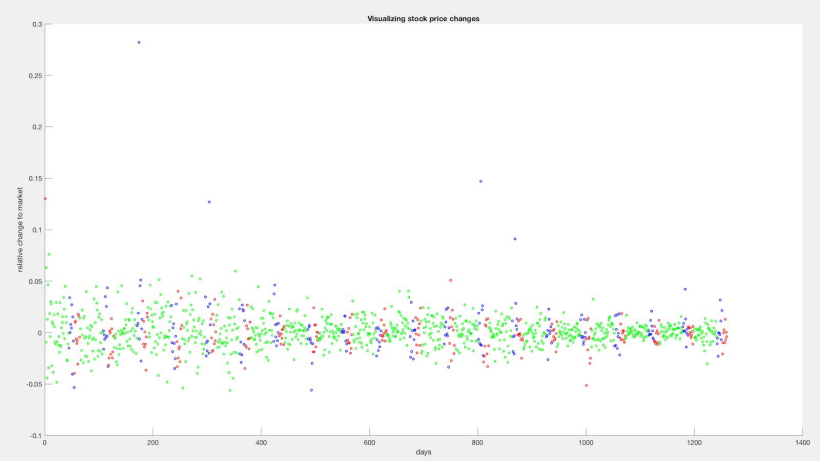
There were also works that attempted to predict prices for the entire quarter based only on textual analysis of the Earnings Report. They failed to acknowledge that everyday news could further steers the direction of the company's future prospect between the Earnings, and thus the stock prices.

Dataset and Features

5 years worth of stock price data were gathered for roughly 20 large-cap technology companies (e.g., FB, BIDU, NVDA, AAPL) and the NASDAQ-100 Technology Sector Index (NDXT) dating from 11/14/2012 to 11/14/2017.

The data were preprocessed to exclude 10 trading days both before and after a company's quarterly Earnings Report from the data, in the hope that the remaining days' prices have lower volatility. Furthermore, I subtracted daily NDXT percentage changes from every stock's daily percentage price changes to obtain an estimate of its relative change within the technology industry. With this adjustment, I hope to eliminate as much of the general market's movement as possible (e.g., changes due to unforeseen political events), because those events tend to affect the entire industry at once. With this relative change, we can focus on the relative performance of individual stocks within the technology industry.

The plot below gives a simple visualization of the processed data for an example stock (FB). Blue and red points are days immediately before and after an Earnings Report respectively (they tend to have larger percentage changes). Green points are the data we are interested in and serve as actual inputs for the model.



I've also retrieved news headlines data for one example stock (FB) for 3.5 years (March 2014 to Nov 2017, limited by the availability of the news API) to evaluate the usefulness of financial news in accurately predicting stock prices. The news data were noisy, and a number of preprocessing steps had to be done to remove duplicates, generate the dictionary, and pair up the news data with price changes on specific dates.

Methods & Results

Evaluation Metric

The evaluation metric for the price predictions is a practical goal: how much can we gain if we trade by strictly following the model predictions, i.e. we buy or hold if the model predicts a positive change and sell prior to a negative change.

Linear Regression

A standard linear regression with the standard least mean square errors was performed using only price changes from previous 20 days. This is inspired by the common observation that stock prices often rebound after consecutive losses or drop after consecutive gains.

Features for this model are simply percentage price changes in the last 20 days, and an intercept term. Because price changes are small in magnitude (usually below 0.10), I used $x_0 = 0.01$ instead of $x_0 = 1$ to help normalize the data (the alternative here would be to normalize the entire dataset). The sign of the output value $h_\theta(x) = \theta * x$ would be our prediction for the price change. For each company, I applied batch gradient descent with a fixed learning rate to learn the θ parameter, which can be used to predict future stock prices.

This model has performed well with training data, but not with test data (see the end of the Results section for numbers), indicating that we have overfitted the model to training data. In addition, θ ends up having very small values for each component, resulting in a near-zero output value regardless of the input to the model. I've also tried only including prices changes in the last 3-5, or 8-12 days as the features, but those models performed worse. Eventually, I added some cross features such as $(yesterday * two\ days\ ago)$, $(yesterday * two\ days\ ago * three\ days\ ago)$ to the model, it marginally improves the results, but the major issue of overfitting is still present.

Logistic Regression

I've also tried logistic regression with the same features (price changes in the last 20 days) to predict the sign of price change for a particular day. This model does not perform any better than linear regression, producing similar results.

Naive Bayes

The next intuitive addition to the model are some indications of general sentiment towards the company. This can be captured in a Naive Bayes analysis of financial new articles related to the company.

I chose to use Naive Bayes with the multinomial event model and Laplace smoothing. The model was trained on 400 days, and tested on 200 days of data. Output of the model is the probability of a positive relative price change. In this analysis, all punctuations were removed from the words, with numbers, percentages, and money amounts substituted with fixed tokens that consolidate these words together so that they appear as one word in the dictionary. The resultant dictionary has about 20 thousand unique words.

Deep Learning

With the ability to gauge positiveness of news for a particular company on any day, a neural network was then trained using all of the available information. The 41 input features to this model include price changes from previous 20 days, news positiveness predictions for the same 20 days, and news positiveness prediction for today. News from today is valid input for the model in practice if we assume we have a program that listens to news broadcasts and instantly makes trade decisions.

The neural network used has 1 input layer, 1 hidden layer, and 1 output layer. The hidden layer has 20 hidden units (roughly half the size of the input layer). The number of hidden units were experimented in the range of 1 to 40 and the best number was picked through trial and error. The activation function used at the hidden and output layers are both sigmoid. Mini-batch gradient descent (with batch size 20) with forward and backward propagation was applied to learn parameters in the neural network. L2 Regularization with lambda = 0.0001 was also applied to reduce overfitting to training data.

Summary Of Results

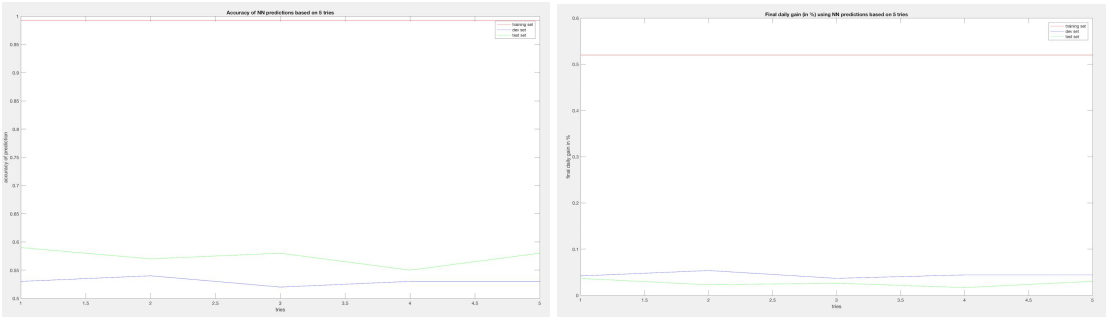
Model	Training Error (probability of wrong prediction)	Test Error (probability of wrong prediction)	Test Daily % Gain
Naive Bayes (News only)	0.0248 (403 samples)	0.445 (200 samples)	N/A
Linear Regression (Prices only)	0.4601 (589 samples)	0.4365 (126 samples)	0.0311
Logistic Regression (Prices only)	0.4584 (589 samples)	0.4762 (126 samples)	0.0018
Neural Network (Prices + News)	0.0084 (403 samples)	0.4100 (100 samples)	0.0423

The trained Naive Bayes model achieved 97.52% accuracy on training data and 55.4% accuracy on test data. There was a clear indication of insufficient data points that resulted in a large degree of overfitting to training data.

The linear regression model (and similar the logistic regression model) resulted in 56.35% prediction accuracy and a daily percentage gain of 0.0311%. The training and test errors were both high for this model, indicating that the model itself probably was never good enough for stock price predictions. This also means that the algorithm finds little pattern in next day price

change according to previous days' price changes. This echoes the fact that the stock market is way more complex than a predictable curve.

The deep learning approach using neural network achieved the best results with a combination of previous price and news positiveness prediction features. The training error was extremely low at 0.84%, and a test error at 41.00% seems acceptable for an initial version of the algorithm to produce some tangible gain (0.0423% daily gain). Both the training and the test error are smaller than errors from the Naive Bayes or the linear regression model alone, signalling that combining the two sources of information has improved our model overall. If we trade strictly following the neural network predictions, we would gain roughly 16% annually on the example stock (FB).



Conclusion & Future Work

To summarize, there was a lot of overfitting to training data despite the regularization techniques used, due to the fact that the stock market is way too complex to be captured in merely price changes and news headlines. In the future, I would work on augmenting the neural network by adding more input features, such as trends in other industries, political influences, competitor trends, just to name a few, to capture the full scale of complexities of a stock market.

The accuracy of the final neural network correlates highly with the Naive Bayes prediction accuracy. Therefore, if we could improve the accuracy for the Naive Bayes news predictor, it would in turn increase with the prediction accuracy of subsequent models. Ways to improve Naive Bayes predictions include removing insignificant words, reducing stemmed words into base forms, and some other corrections that might be revealed by detailed error analysis. Furthermore, higher quality news data would have helped with prediction accuracy. Finally, we could also choose to include the full news articles in the input instead of just the headline texts.

As for the neural network, different architectures (e.g. a different number of hidden layers and hidden units) could be explored to see if it could produce better results. Other activation functions such as ReLU or tanh could also be explored on the hidden layer and the output layer.