Machine Learning Engineer Nanodegree

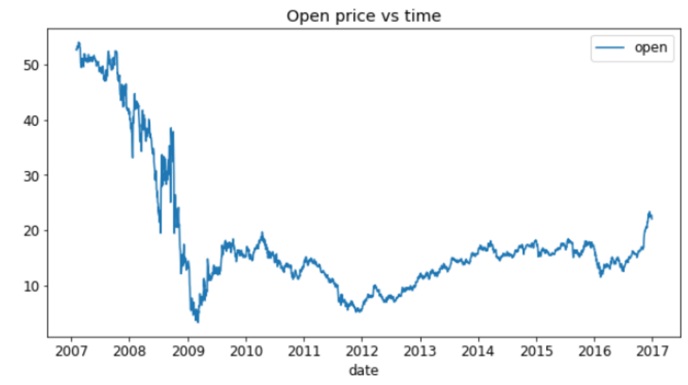
Capstone Project

Johnathon Schultz
October 15th, 2018

I. Definition

Project Overview

Predicting stock performance is a task most individuals and corporations have attempted to achieve at some level of accuracy since the initiation of exchanges. The challenge of such prediction is the speed with which data stagnates. For example, a piece of information may only be valid short term, until a new piece of information is received. To help mitigate this problem, folks have looked to correlate data available at a similar large volume and stagnation rate to that of stock price performance. One such data source is news analytics.



The ubiquity of data today enables investors at any scale to make better investment decisions. The challenge is ingesting and interpreting the data to determine which data is useful, finding the signal in this sea of information. By analyzing news data to predict stock prices, we have a unique opportunity to advance the state of research in understanding the predictive power of the news. This power, if harnessed, could help predict financial outcomes and generate significant economic impact all over the world.

Here are some example attempts at tackling this problem:

- https://medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877
- https://towardsdatascience.com/stock-prediction-in-python-b66555171a2
- https://www.quantinsti.com/blog/machine-learning-trading-predict-stock-prices-regression/

Data

For this project I have two data sources:

1. Market data (2007 to 2017) provided by Intrinio–contains financial market information such as opening price, closing price, trading volume, calculated returns, etc.

	time	assetCode	assetName	volume	close	open
0	2007-02-01 22:00:00+00:00	A.N	Agilent Technologies Inc	2606900	32.2	32.17
1	2007-02-01 22:00:00+00:00	AAI.N	AirTran Holdings Inc	2051600	11.1	11.08
2	2007-02-01 22:00:00+00:00	AAP.N	Advance Auto Parts Inc	1164800	37.5	37.99
3	2007-02-01 22:00:00+00:00	AAPL.O	Apple Inc	23747329	84.7	86.23
4	2007-02-01 22:00:00+00:00	ABB.N	ABB Ltd	1208600	18	18.01

2. News data (2007 to 2017) provided by Thomson Reuters–contains information about news articles/alerts published about assets, such as article details, sentiment, and other commentary.

	time	headline	bodySize	assetCodes	relevance	sentiment Negative	sentiment Neutral	sentiment Positive
0	2007-01-01 04:29:32+00:00	China's Daqing pumps 43.41 mln tonnes of oil i...	1438	{'0857.HK', '0857.F', '0857.DE', 'PTR.N'}	0.2357	0.500739	0.419327	0.079934
1	2007-01-01 07:03:35+00:00	FEATURE-In kidnapping, finesse works best	4413	{'STA.N'}	0.44721	0.600082	0.345853	0.054064
2	2007-01-01 11:29:56+00:00	PRESS DIGEST - Wall Street Journal - Jan 1	2108	{'WMT.DE', 'WMT.N'}	0.37796	0.450049	0.295671	0.25428
3	2007-01-01 12:08:37+00:00	PRESS DIGEST - New York Times - Jan 1	1776	{'GOOG.O', 'GOOG.OQ', 'GOOGa.DE'}	0.14907	0.752917	0.162715	0.084368
4	2007-01-01 12:08:37+00:00	PRESS DIGEST - New York Times - Jan 1	1776	{'XMSR.O'}	0.14907	0.699274	0.20936	0.091366

Each asset is identified by an assetCode (note that a single company may have multiple assetCodes). Depending on what you wish to do, you may use the assetCode, assetName, or time as a way to join the market data to news data.

To prevent lookahead bias, the data will be split into two sets: Train (2007 - 2016) and Test (2017).

Problem Statement

I will predict a signed confidence value, $\hat{y}_{ti} \in [-1, 1]$, which is multiplied by the market-adjusted return of a given assetCode over a ten day window. If I expect a stock to have a large positive return–compared to the broad market–over the next ten days, I will assign it a large, positive confidenceValue (near 1.0). If I expect a stock to have a negative return, I assign it a large, negative confidenceValue (near -1.0). If unsure, I assign it a value near zero.

Metrics

The final metric selected for this project is a variation of Relative Percent Difference (RPD). For each day in the evaluation time period, calculate:

$$x_t = 1 - \left(\frac{v - a}{|v| + |a|} + 1 \right) \times 2$$

where *v* is the predicted value and *a* is the actual target value on a particular day *t*. The relative distance is then scaled between [0, 1] and inverted–with higher values, closer to 1, being more accurate.

The model score is calculated as the mean of the daily x_t values:

$$score = \bar{x}_t$$

This metric is well suited for the model. It can be calculated per time-step and summarizes overall performance of the model–highlighting advances with higher scores.

II. Analysis

Data Exploration

Market data

The data includes a subset of US-listed instruments. The set of included instruments changes daily and is determined based on the amount traded and the availability of information. This means that there may be instruments that enter and leave this subset of data. There may therefore be gaps in the data provided, and this does not necessarily imply that that data does not exist (those rows are likely not included due to the selection criteria).

The marketdata contains a variety of returns calculated over different timespans. All of the returns in this set of marketdata have these properties:

- Returns are always calculated either open-to-open (from the opening time of one trading day to the open of another) or close-to-close (from the closing time of one trading day to the open of another).
- Returns are either raw, meaning that the data is not adjusted against any benchmark, or market-residualized (Mktres), meaning that the movement of the market as a whole has been accounted for, leaving only movements inherent to the instrument.
- Returns can be calculated over any arbitrary interval. Provided here are 1 day and 10 day horizons.
- Returns are tagged with 'Prev' if they are backwards looking in time, or 'Next' if forwards looking.

Within the marketdata, you will find the following columns:

- time(datetime64[ns, UTC]) - the current time (in marketdata, all rows are taken at 22:00 UTC)
- assetCode(object) - a unique id of an asset
- assetName(category) - the name that corresponds to a group of assetCodes. These may be "Unknown" if the corresponding assetCode does not have any rows in the news data.
- universe(float64) - a boolean indicating whether or not the instrument on that day will be included in scoring. This value is not provided outside of the training data time period. The trading universe on a given date is the set of instruments that are avilable for trading (the scoring function will not consider instruments that are not in the trading universe). The trading universe changes daily.
- volume(float64) - trading volume in shares for the day
- close(float64) - the close price for the day (not adjusted for splits or dividends)
- open(float64) - the open price for the day (not adjusted for splits or dividends)

- returnsClosePrevRaw1(float64) - see returns explanation above
- returnsOpenPrevRaw1(float64) - see returns explanation above
- returnsClosePrevMktres1(float64) - see returns explanation above
- returnsOpenPrevMktres1(float64) - see returns explanation above
- returnsClosePrevRaw10(float64) - see returns explanation above
- returnsOpenPrevRaw10(float64) - see returns explanation above
- returnsClosePrevMktres10(float64) - see returns explanation above
- returnsOpenPrevMktres10(float64) - see returns explanation above
- returnsOpenNextMktres10(float64) - 10 day, market-residualized return. This is the target variable used in competition scoring. The market data has been filtered such that returnsOpenNextMktres10 is always not null.

News data

The news data contains information at both the news article level and asset level (in other words, the table is intentionally not normalized).

- time(datetime64[ns, UTC]) - UTC timestamp showing when the data was available on the feed (second precision)
- sourceTimestamp(datetime64[ns, UTC]) - UTC timestamp of this news item when it was created
- firstCreated(datetime64[ns, UTC]) - UTC timestamp for the first version of the item
- sourceId(object) - an Id for each news item
- headline(object) - the item's headline
- urgency(int8) - differentiates story types (1: alert, 3: article)
- takeSequence(int16) - the take sequence number of the news item, starting at 1. For a given story, alerts and articles have separate sequences.
- provider(category) - identifier for the organization which provided the news item (e.g. RTRS for Reuters News, BSW for Business Wire)

- subjects(category) - topic codes and company identifiers that relate to this news item. Topic codes describe the news item's subject matter. These can cover asset classes, geographies, events, industries/sectors, and other types.
- audiences(category) - identifies which desktop news product(s) the news item belongs to. They are typically tailored to specific audiences. (e.g. "M" for Money International News Service and "FB" for French General News Service)
- bodySize(int32) - the size of the current version of the story body in characters
- companyCount(int8) - the number of companies explicitly listed in the news item in the subjects field
- headlineTag(object) - the Thomson Reuters headline tag for the news item
- marketCommentary(bool) - boolean indicator that the item is discussing general market conditions, such as "After the Bell" summaries
- sentenceCount(int16) - the total number of sentences in the news item. Can be used in conjunction with firstMentionSentence to determine the relative position of the first mention in the item.
- wordCount(int32) - the total number of lexical tokens (words and punctuation) in the news item
- assetCodes(category) - list of assets mentioned in the item
- assetName(category) - name of the asset
- firstMentionSentence(int16) - the first sentence, starting with the headline, in which the scored asset is mentioned.
    - 1: headline
    - 2: first sentence of the story body
    - 3: second sentence of the body, etc
    - 0: the asset being scored was not found in the news item's headline or body text. As a result, the entire news item's text (headline + body) will be used to determine the sentiment score.
- relevance(float32) - a decimal number indicating the relevance of the news item to the asset. It ranges from 0 to 1. If the asset is mentioned in the headline, the

relevance is set to 1. When the item is an alert (urgency == 1), relevance should be gauged by firstMentionSentence instead.

- sentimentClass(int8) - indicates the predominant sentiment class for this news item with respect to the asset. The indicated class is the one with the highest probability.
- sentimentNegative(float32) - probability that the sentiment of the news item was negative for the asset
- sentimentNeutral(float32) - probability that the sentiment of the news item was neutral for the asset
- sentimentPositive(float32) - probability that the sentiment of the news item was positive for the asset
- sentimentWordCount(int32) - the number of lexical tokens in the sections of the item text that are deemed relevant to the asset. This can be used in conjunction with wordCount to determine the proportion of the news item discussing the asset.
- noveltyCount12H(int16) - The 12 hour novelty of the content within a news item on a particular asset. It is calculated by comparing it with the asset-specific text over a cache of previous news items that contain the asset.
- noveltyCount24H(int16) - same as above, but for 24 hours
- noveltyCount3D(int16) - same as above, but for 3 days
- noveltyCount5D(int16) - same as above, but for 5 days
- noveltyCount7D(int16) - same as above, but for 7 days
- volumeCounts12H(int16) - the 12 hour volume of news for each asset. A cache of previous news items is maintained and the number of news items that mention the asset within each of five historical periods is calculated.
- volumeCounts24H(int16) - same as above, but for 24 hours
- volumeCounts3D(int16) - same as above, but for 3 days
- volumeCounts5D(int16) - same as above, but for 5 days
- volumeCounts7D(int16) - same as above, but for 7 days

Distribution

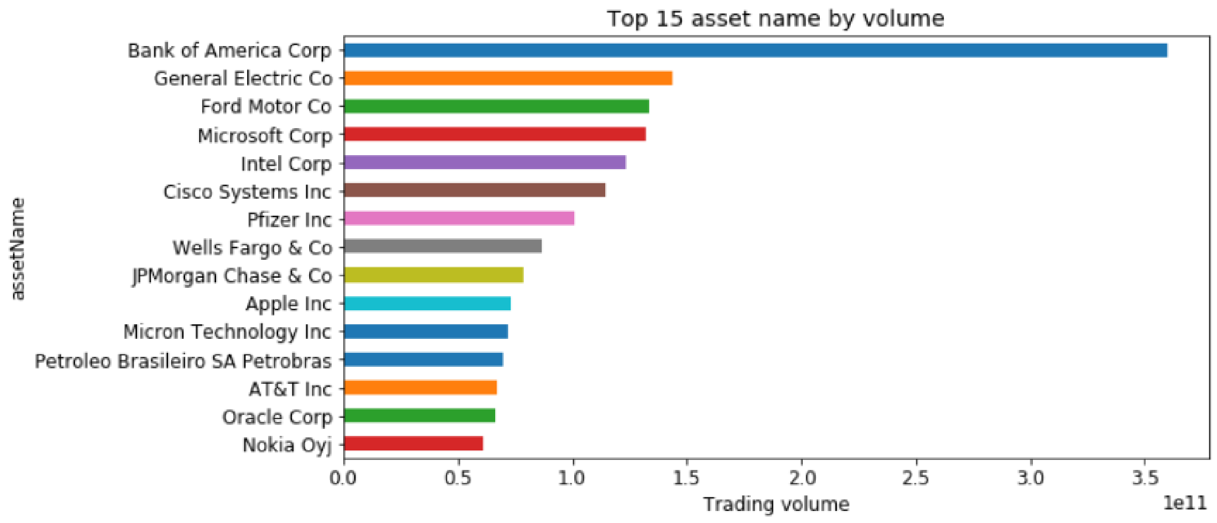
The distribution of data is heavily analyzed inside the *01-final-project-eda.ipynb*. The data is well distributed across the time-series. News data spikes in volume around major turns in the stock market, such as the crash in 2008.

Digging into the stock market data, the features are bounded and seem to be of good quality with little to no outliers. It becomes challenging to classify a data point on the stock market as an outlier since it represents the fluctuation in the market performance we are trying to predict.

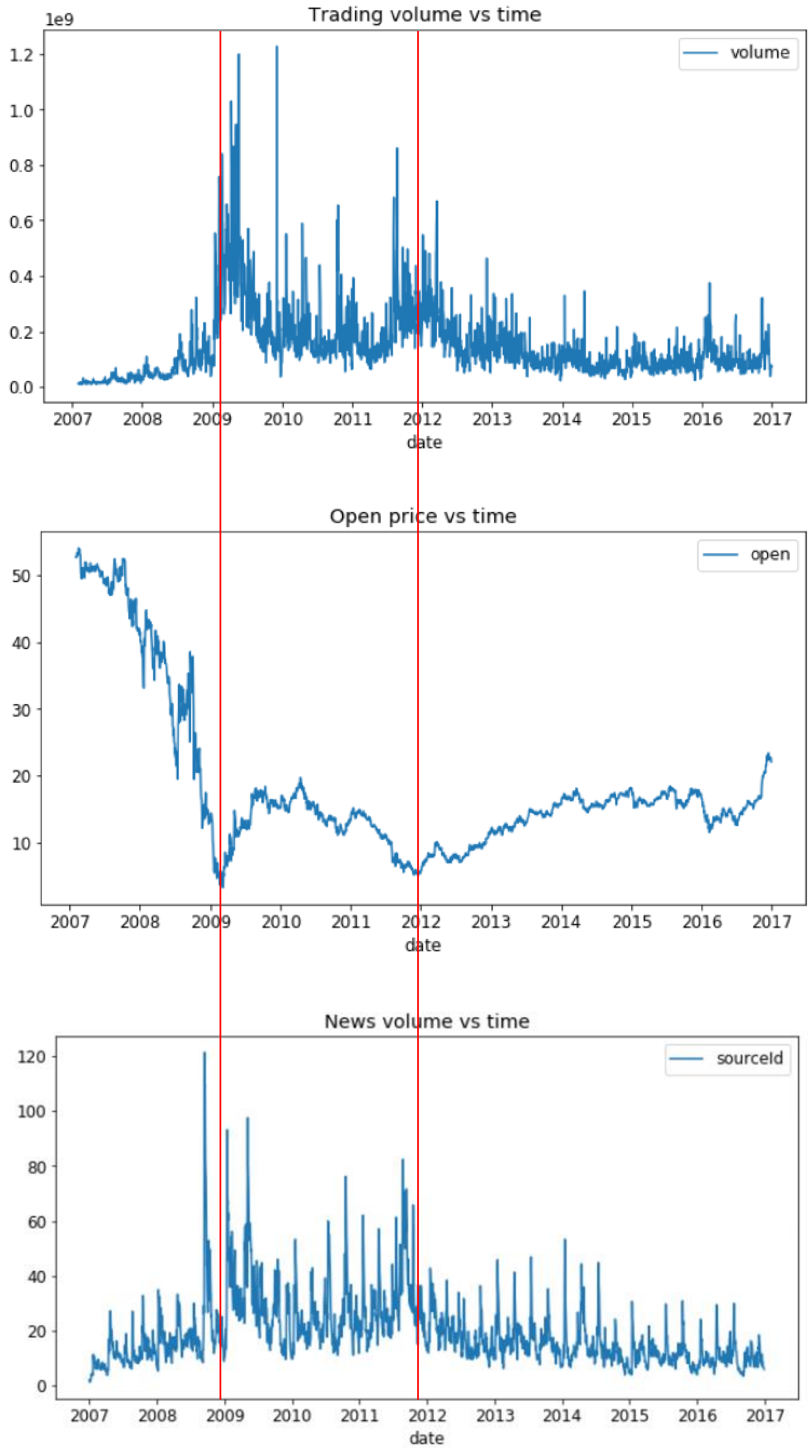
	volume	close	open
count	4.07E+06	4.07E+06	4.07E+06
mean	2.67E+06	3.97E+01	3.97E+01
std	7.69E+06	4.22E+01	4.25E+01
min	0.00E+00	7.00E-02	1.00E-02
25%	4.66E+05	1.73E+01	1.73E+01
50%	9.82E+05	3.03E+01	3.03E+01
75%	2.40E+06	4.99E+01	4.99E+01
max	1.23E+09	1.58E+03	1.00E+04

Exploratory Visualization

Due to the large size of the dataset, for this project, I utilized a single asset. With the volume of data available in relation to the remaining assets, Bank of America Corp was selected as the asset to be predicted across all models.



In the following charts (next page), it can be seen that trading volume is strongly associated with price, i.e. trade increase when price hits bottom. Return is also strongly fluctuated at such time. Further, the news increases in volume and urgency as price drops. This is a good indicator that news data features will add value and predictive capability to the model once added in.


Trading volume vs time
1e9
1.2
1.0
0.8
0.6
0.4
0.2
0.0
volume
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
date
Open price vs time
50
40
30
20
10
open
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
date
News volume vs time
120
100
80
60
40
20
0
sourceId
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
date

Algorithms and Techniques

First, implement a naïve benchmark model in code to produce the initial results for the project. The performance of this model will be used to compare to the performance of the target model. The goal is for the target model to outperform the benchmark model. This model is discussed further in the *Benchmark* section.

Next, implement a deep neural network to take in stock trading data. Since time is a large component of the problem, the architecture I would likely use would be that of a recurrent neural network (RNN). Specifically, I will be implementing a Long-Short Term Memory (LSTM) network for the benefits it possess in carrying forward relevant information from one time-step to future time-steps, as well as 'forgetting' irrelevant information that isn't predictive in the time-series.

> *The LSTM model does a forward pass on the data to produce a vector. Based on this vector and the provided labels, we use the loss function to calculate direction and magnitude to apply to the weight matrix for gradient descent. Then, through a process called back-propagation, the weights and memory cells inside the network are updated, and the process is repeated until training halts.*

Lastly, integrate news data features, such as sentiment, urgency, etc., into the LSTM model to evaluate any gain/loss in predictive performance.

All models are scored with the same metric function. The model with the highest score is indicative of the model with the most predictive ability for the dataset. The remaining models will be ranked subsequently according to their scores.

Benchmark

As a benchmark, the first model will employ a naïve approach of future price prediction. The algorithm I intend to use for benchmarking will be to predict uncertainty for the future timeslots for all input values. Thus, the output of the model for all assetCodes over a ten day window will be a confidenceValue of 0.0, indicating uncertainty of positive or negative return, on a range of [-1.0, 1.0]. Such a prediction would indicate the market price for the stocks remain unchanged over the prediction period, which is what we will use as a baseline, or benchmark model.

III. Methodology

Data Preprocessing

For time-series data, both market and news, some days have multiple reported values and other days have no data. To handle this, days with multiple values were rolled together with means for the day. To fill missing values, the data was forward-filled, taking one data point and repeating it to the next day to fill the gap. For missing values at the start of the time-series, the data was back-filled. The result is a complete dataset with values for all features during all days in the time-series.

Implementation

Determining a relative model score that represented the performance across the time-series to compare between models was challenging. Since the prediction values were between [-1, 1], the values first were shifted to positive number space between [0, 2]. With this new value range, the Relative Percent Difference (RPD) is calculated, as described in *Metrics* section. RPD produces values on the range of [-2, 2], so, the values were again shifted and scaled to [0, 1]. But, by default, RPD is a metric of difference, so to create a metric of performance where higher values are better, the scores were inverted so higher values, values closer to 1, are better.

Additionally, shaping the data into a 3D tensor to be fed into the LSTM model with a lookback value posed a challenge. The data had to be extracted from the pandas DataFrame into a numpy array since DataFrames do not effectively handle data over 2 dimensions. Once in numpy, the data had to be reshaped and the 3rd dimension, time, had to be added in. Once the data was augmented, the shape lined up to that with which the LSTM expected as input, and the data flowed through the model as expected.

Refinement

In terms of refinement, not much took place since the majority of the model architecture and parameters were held constant so that we could evaluate the effects of the addition of news features. Since the feature space is being expanded, however, the number of epochs for each model was tuned until the validation loss began increasing. The goal of tuning the epochs is to ensure both LSTM based models were trained to their fullest while mitigating overfitting the training set. Since both models were trained to a similar extent, improvements would be indicative of predictive value from news data features.

IV. Results

Model Evaluation and Validation

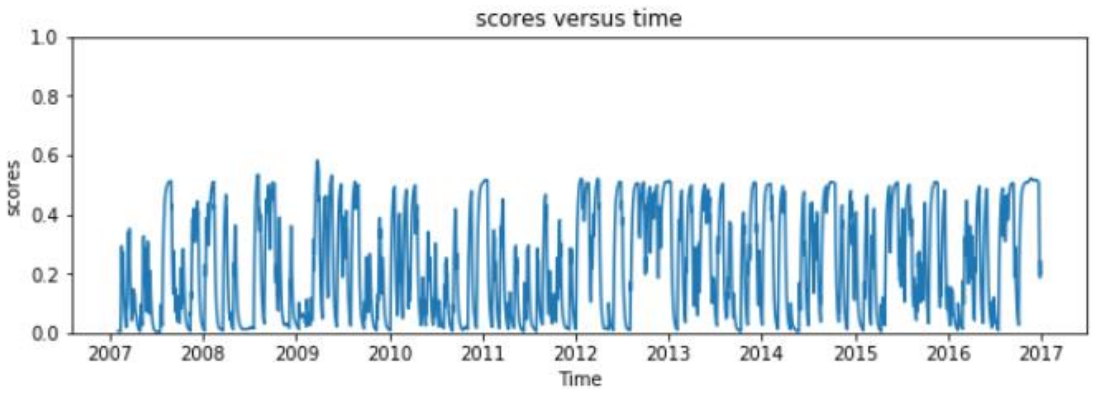
Shown on the next page, the benchmark model performed poorly. This was to be expected. The benchmark model is not taking into account any data from the dataset, rather, the model is naively predicting 0.0, indicating uncertainty of positive or negative return, on a range of [-1.0, 1.0]. The benchmark model never achieves a score higher than 0.6 for any time-step in the series, and overall the model scored 0.231.

Training an LSTM on market data features proved fruitful. The evaluation model with market data produced more accurate predictions overall, and, if we look at the period between 2008 – 2010 when the stock market was on a downturn, followed by a rebound, the model scores higher than that of the benchmark. This indicates that the market trends were learned during training and transferred through to inference time. The evaluation model is more accurately able to predict the returns of the stock market, with a peak score of over 0.8 and an overall model score of 0.302.

Finally, the addition of news data features to the evaluation model improved the ability of the model to predict stock trends even further. The predictions for the period between 2008 – 2010 show even higher score values for this model. And if we look at the period between 2011 – 2012, we see additional spikes in scores not seen in either of the previous models. The evaluation model with market and news data performs the best of all the models we tested with a peak score of 1.0, indicating a perfect prediction for that time-step, and an overall model score of 0.334.

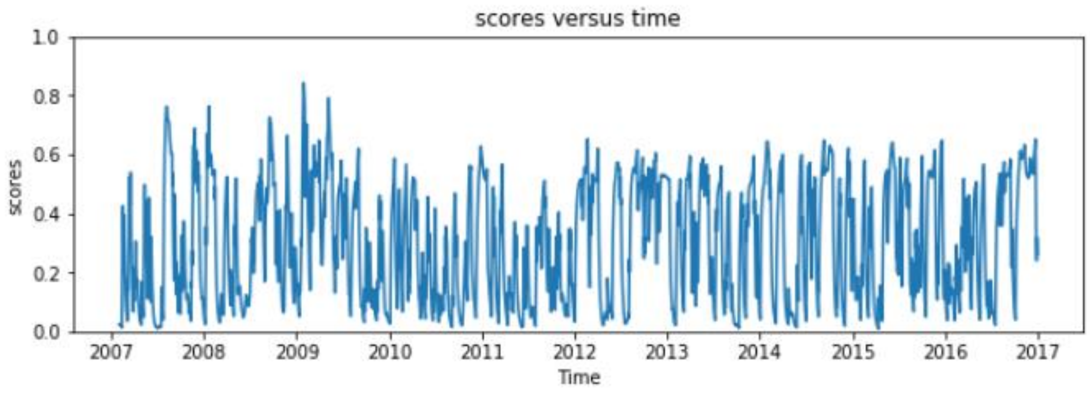
In all cases, the model performs equally as well on unseen testing data as it does on the training data used to obtain the models. As you can see in the charts below, the score results for the test data (2016-2017) do not differ drastically from those of previous years.

Benchmark model



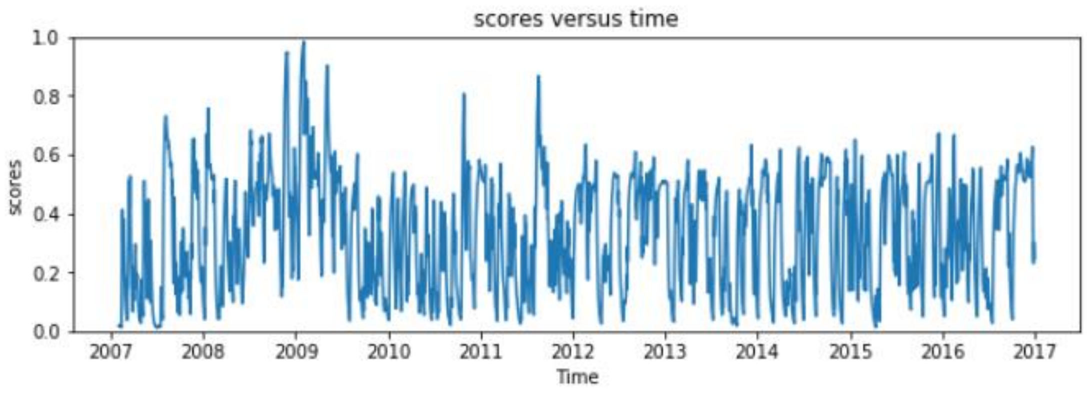
Baseline model score is 0.23099

Eval Model – Market only



Eval model score is 0.30243

Eval Model – Market and News



Eval model score is 0.33428

Justification

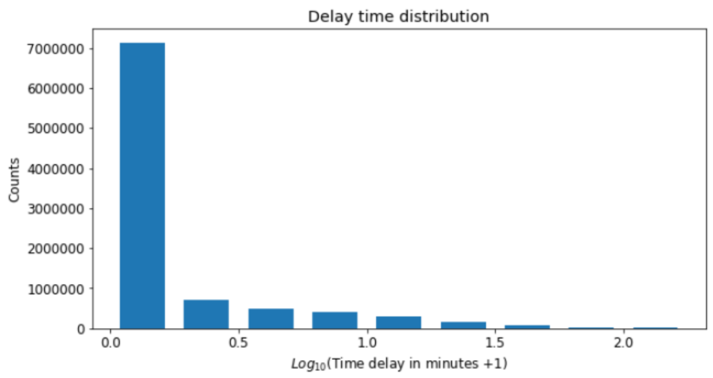
The benchmark achieved a model score of 0.231 and the evaluation model with market and news data achieved a model score of 0.334. Therefore, our final model shows a 45% improvement over the benchmark model.

Using only market data, the evaluation model achieved a score of 0.30243. With market and news data the evaluation model achieved a score of 0.334. Therefore, adding news data into the model shows a 10.5% improvement over the evaluation model using only market data.

An improvement of 10.5% is statistically significant and suggests stock market prediction models can benefit from the additional data points, trends, and sentiment gained when news data is considered as input into the model.

V. Conclusion

Free-Form Visualization



Another challenge in the domain, given the speed at which stock prices change, is the amount of time it takes to collect, analyze and consider additional data into a stock market prediction model. Pictured above is the delay in minutes from when a given news article was posted to the time it was collected and processed into the news dataset. We see in the chart above that although a vast majority of news data is available very quickly, some data is lagged behind. Not to mention, the data above only reflects the dates the news was posted. Another issue is the amount of time it takes for a piece of information to be digitized and made available for collection and analysis.

Reflection

Predicting the stock market is inherently difficult. There are a vast number of entities involved that affect the fluctuation and final pricing that it is challenging to capture a holistic view of the latent factors that affect price. To narrow the scope of the problem, we focused on how identifying and incorporating one source of outside information, news, into the prediction of stock market performance, affects the accuracy of such model. Once we worked through the challenges, identified in the *Implementation* section, we were able to show statistically, the benefit of incorporating such data source.

Throughout this project, it became apparent how important data quality and volume is for the model's performance. Intuitively it makes sense that a model can't rely on a poor signal to make accurate predictions. Artificial Intelligence and Machine Learning algorithms are not magic. They require engineering experience, domain knowledge, and quality data for signals and trends. As a result, this may require additional datasets to broaden the wealth of knowledge.

Improvement

Since most model parameters were held constant during our testing, an area for improvement would be tuning the architecture of the model used to predict the market trend. In this case we implemented the test with an LSTM to capture some of the information in the sequence of data; however, recent research has shown improvements in both computation costs and accuracy by leveraging, for example, convolutional neural network (CNN) architecture.

Additionally, aside from the number of batches, little hyper-parameter tuning was done to optimize the performance of the networks.

- Different activation functions could be leveraged, such as sigmoid, tanh, or selu.
- The number of neurons per layer can be optimized
- Experiments with different values for BatchNormalization and Dropout
- Features that don't benefit the model, or cause noise, can be removed from the input
- Experiments can be set up to identify an improved loss function and optimizer