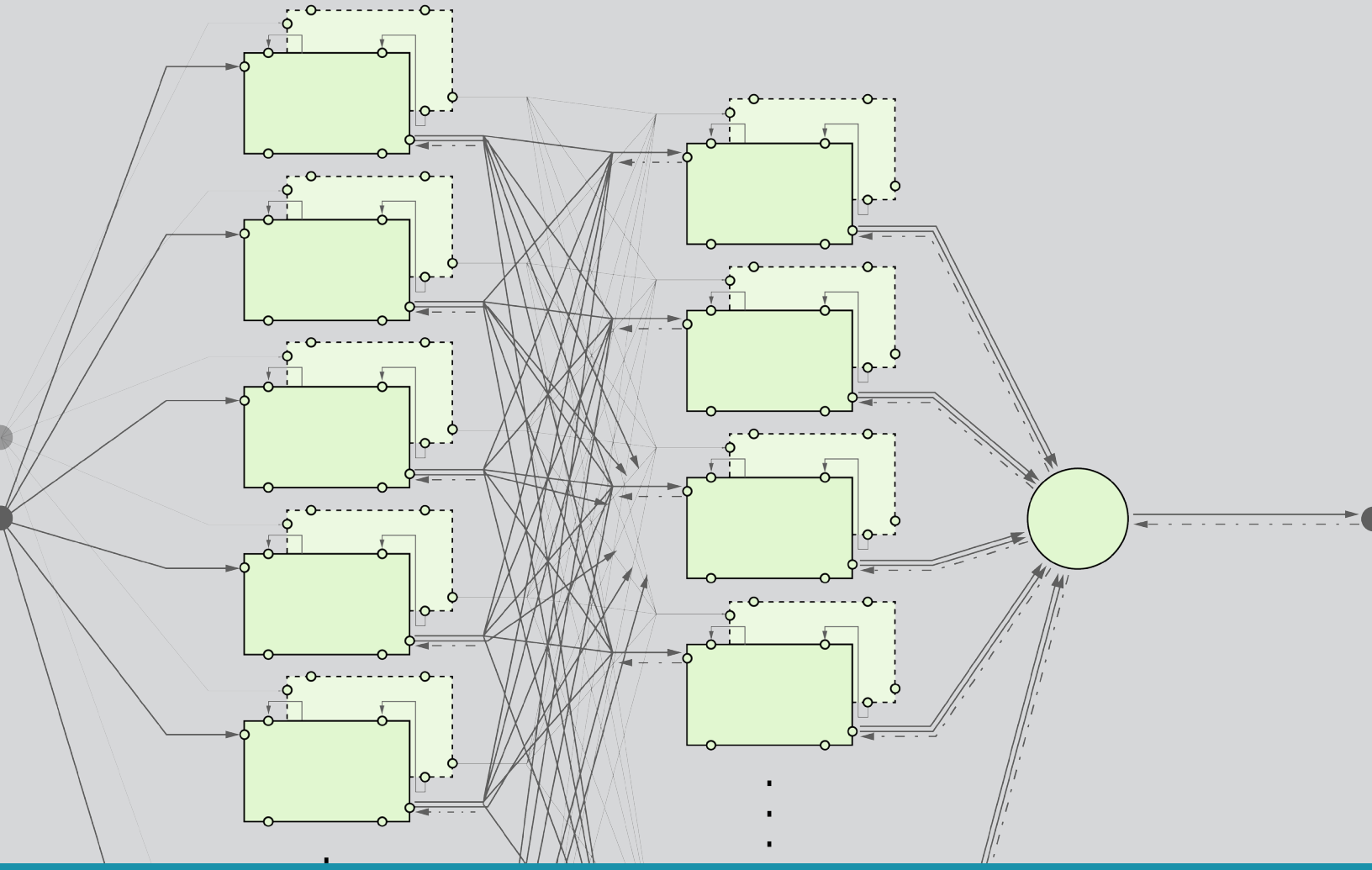


DEEP LEARNING STOCK VALUE PREDICTOR

Gil Akos

April 7th, 2017

Capstone Project Report

Machine Learning Nanodegree // Udacity

CONTENT

1 DEFINITION

Project Overview

Investment firms, hedge funds, and automated trading systems have used programming and advanced modeling to interact with and profit from the stock market since computerization of the exchanges in the 1970s[1]. Whether by means of better analysis, signal identification, or automating the frequency of trades, the goal has been to leverage technology in order create investment systems that outperform alternatives - either service providers (competitors like alternative hedge funds) or products/benchmarks (Exchange Traded Funds "ETFs" or the Standard & Poor's "S&P" 500 Index).

Today, the most promising and ascendant technology, Deep Learning, is the target of incorporation into advanced investment systems[2] offered by "Artificially Intelligent Hedge Funds"[3] and "Deep Investing"[4] as-a-service startups, with claims of outperformance of the S&P 500 Index of up to 87%.

> *Given profit opportunity that large, can a basic Deep Learning model built with publicly available technology achieve positive predictive performance?*

Even an order of magnitude less of an advantage (8.7%) than those claims over the noise in the market and a baseline of the S&P ETF (SPY) could be valuable for an amateur investor!

This project seeks to utilize deep learning models, specifically Recurrent Neural Nets "RNNs," to predict stock prices. Much academic work has been developed using this technique[5], as well as similar studies using Boltzmann machines[6] for both momentum trading strategies and time series prediction. As discussed above and in the below articles from sources ranging from technology magazines (Wired[3]) to the standard bearer for market information (Financial Times[2]), these models are also being applied to real world trading platforms[7]. In this project, I will develop a deep learning model to: predict stock prices using historical closing price and trading volume; visualize both the predicted price values over time; and log the performance metrics and optimal parameters for the model.

Problem Statement

The challenge of this project is to accurately predict the future closing value of a given stock across a given time period in the future. Because almost all stocks have a large range of historical daily closing price values, machine learning models for consideration should be capable of making predictions with time series data and should be trainable over a long period of time. The subsequently predicted values should be backtested for their accuracy against a sizeable period of time as well. With those constraints over the stock data duration and model capabilities, the project will experiment with various types of machine learning and deep learning models as well as various architectures of assembling such models.

The key questions this investigation seeks to answer are:

- How well do deep learning models perform compared to machine learning models for stock price prediction?
- Within the category of deep learning models that work effectively with time series data, which model types and architectures perform optimally?
- Can a relatively simple deep learning model predict well enough to define a trading strategy that outperforms benchmarks?

Before beginning development, the originally hypothesized solution for predicting stock prices was a based on recent literature[8] and was centrally composed of a Recurrent Neural Network built with Keras[9]. During development, the experimentation with model types diverged into alternate model types and more complex architectures, including those in the Long Short-Term Memory[10] category.

Metrics

The measures of performance for this project will be the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) calculated as the difference between predicted and actual values of the target stock ticker at daily market close. The MSE and RMSE are standards for measuring error for machine learning predictions and will offer consistency across all the studies developed, given that all the datasets will be normalized prior to model fitting. Additionally, the visualizations from each study will be graphically analyzed for their degree of fit, ability incorporate volatility or lag across time, and macro trends for diverging from the test data set. Lastly, the finalized study will be measured for the delta between the performance of the benchmark model(s) and our finalized and tuned deep learning

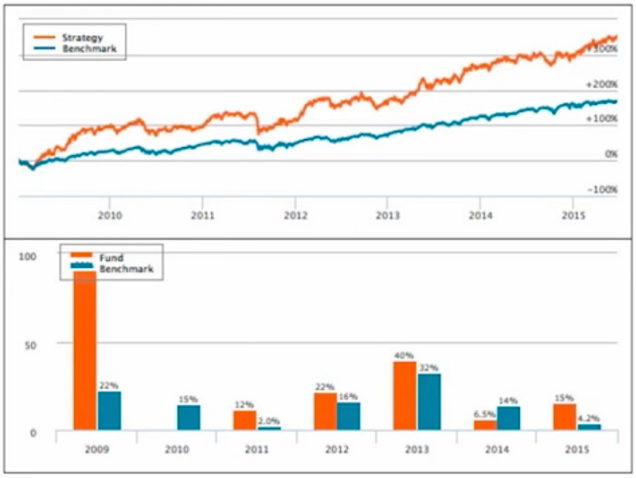
model. For this last metric, comparison will be made to the SPY ETF, the initial linear regression model developed, and the Lucena Research tool shown in the Machine Learning for Trading[11] course. The graph displayed in that course shows a 3.64x improvement over the S&P benchmark from 1/1/2009 to 6/18/2015, which will be used for both the date range for the test dataset and the target performance improvement.

S&P 500 -- % Change -- 1 month Performance 1/1/09 - 6/18/15

LUCENA RESEARCH

Investment Approach

Performance Versus Benchmark 1/1/09 to 6/18/15



	Strategy Overall	Bench Overall	Strategy YTD	Bench YTD
Abs. Return	346.73%	161.32%	15.30%	4.20%
Rel. Return	185%	N/A	10.36%	N/A
Beta	1.01	N/A	0.75	N/A
Std. Dev	1.42%	1.13%	0.83%	0.75%
Sharpe	1.14	0.92	2.32	0.78
Draw Down	-26.60%	-27.20%	-4.52%	-3.59%
IR	0.68	N/A	2.09	N/A
R^2	0.64	N/A	0.45	N/A

Figure 1: Sample performance versus benchmark screen from Lucena Research.

2 ANALYSIS

Data Exploration

The data used in this project is drawn from historical trading details of the United States stock market, including for each stock ticker utilized, the daily Adjusted Closing Price in USD and Volume of Shares traded. Additional common data points such as the Rolling Mean over ***n*** days were derived from the Adjusted Closing Price, which accounts for any historical share splits or adjustments. Because this dataset is indexed by date and many stocks have been publicly traded for many years, the time series data structure for the project goal is met and plenty of stock tickers exist to select that also meet the criteria of an adequately long training period before the target test period of 1/1/2009 to 6/18/2015. To define a time period that meets the test period length and also reaches far enough in the past to minimize correlated market trends but also not too far to weight training to periods where the market operated differently, I settled on a time series that ranges from 1/1/1995 to 6/18/2015 where the test period is the final 31.5%. For ease of reproducibility and reusability, all data was pulled from the Yahoo Finance[12] Python API[13] and converted into a Pandas DataFrame.

```
{'Adj_Close': '204.118213',  'Close': '212.779999',  'Date': '2015-06-18',  'High': '213.339996',
'Low': '210.630005',  'Open': '211.309998',  'Symbol': 'SPY',  'Volume': '165867900'},

 {'Adj_Close': '202.01736',  'Close': '210.589996',  'Date': '2015-06-17',  'High': '211.320007',
'Low': '209.360001',  'Open': '210.589996',  'Symbol': 'SPY',  'Volume': '126708600'},

 {'Adj_Close': '201.691205',  'Close': '210.25',  'Date': '2015-06-16',  'High': '210.350006', 'Low':
'208.720001',  'Open': '208.929993',  'Symbol': 'SPY',  'Volume': '85308200'},
...
```

Figure 2: Sample data received from Yahoo Finance API. Note the Dictionary format with daily entries in reverse chronological order.

Item	Date	Adj_Close	Volume
0	1995-01-03	30.574097	324300
1	1995-01-04	30.720218	351800
2	1995-01-05	30.720218	89800
3	1995-01-06	30.751472	448400
4	1995-01-09	30.782794	36800

Figure 3: Head of Pandas DataFrame. Note the ascending order of the items relative to the Date column.

For each study, the project includes results for each of the following tickers:

- SPY // S&P 500 Exchange Traded Fund // Nasdaq Stock Market
- GE // General Electric // New York Stock Exchange
- MSFT // Microsoft Corporation // Nasdaq Stock Market
- AAPL // Apple Inc. // Nasdaq Stock Market

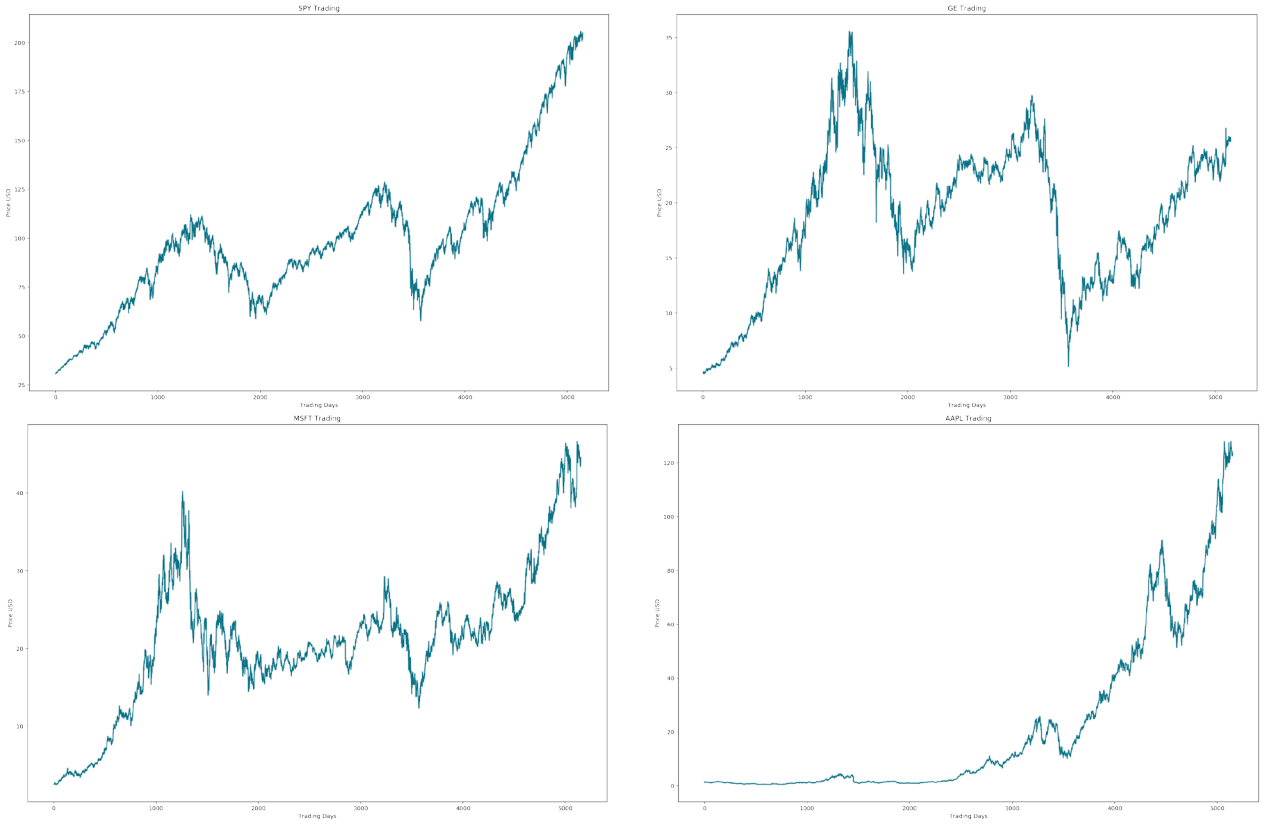


Figure 4, 5, 6, 7 (left to right, in rows): Adjusted Closing Price for SPY, GE, MSFT, and AAPL from 1/1/1995 to 6/18/2015.

These four stocks offer a range of study inputs through their diverse respective types (index, public company), sectors (composite, industrial, technology), trading price graph shapes (see Figures 4-7), and range of share price across time (see Figures 4-7).

Despite those differences across the four tickers, there are some consistent characteristics to the data. Regardless of the shape of the Adjusted Close graph in the first two-thirds of the data, all ticker graphs trend up and to the right in the final one-third. Given that this coincides with the Global Financial Crisis of 2008-2009 this is to be expected since all stocks saw value decreases with the market crash. This feature of the data may make difficult assessing a model's ability to generalize well. Despite that potential challenge, the graphs also all show a high degree of volatility over time with both small and large swings in price. This noisy-ness to the data calms concerns about the macro trend to the upper right, but also suggests that including a Rolling Mean across 5, 10, 20, or 40 days (equivalent to 1, 2, 4, and 8 trading weeks) may help the model manage the noise and prevent overfitting. Similarly, a strategy of including the SPY index in the dataset may also train the model to manage volatility or match prevailing market trends.

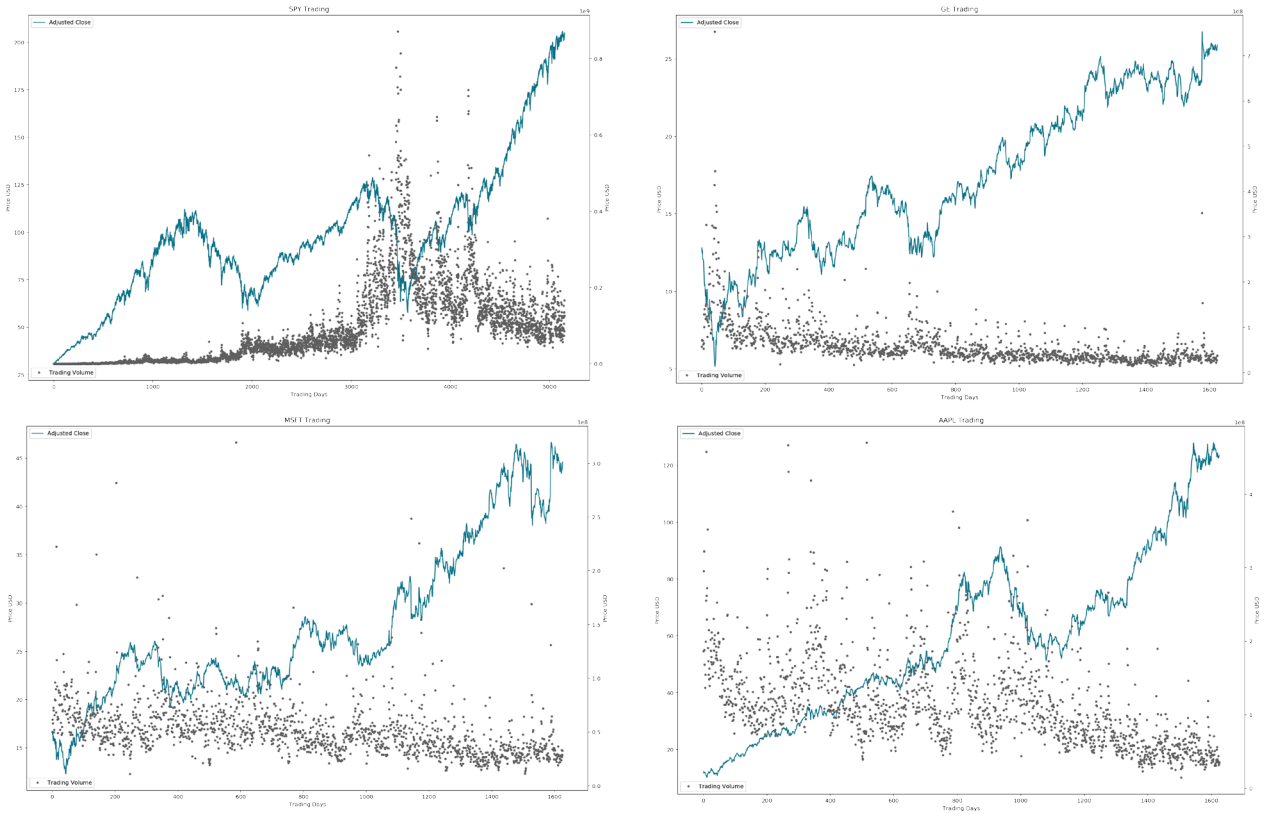


Figure 8, 9, 10, 11 (left to right, in rows): Adjusted Closing Price (line) and Trading Volume (dot) for SPY, GE, MSFT, and AAPL from 1/1/1995 to 6/18/2015.

Exploratory Visualisation

Given the prevailing volatility of Adjusted Closing Prices for all stocks selected, is there a correlation to a different parameter of the stock data? It could be reasonable to expect that a high volume of shares traded equates to a large swing in the stock price. Many shares traded could either mean that there is a big sell off or purchasing run. While this project is not incorporating other market influencers like analyst price targets or earnings calls and all the tickers selected are both popular and actively traded stocks, which should not indicate a high degree of correlation[14], it does appear that there is an absolute relationship between Volume and Adjusted Closing Price. If the Volume is high, the Price will swing significantly in either a positive or negative direction. As seen in Figures 12-15, there is a clustering of plots along the zero percent change axis; however, the cone shape particularly seen for MSFT in Figure 14 indicates that as you move increase volume traded when there is a net change in Adjusted Closing Price there is a hint of a linear relationship. Although this correlation is not starkly obvious, it was my hypothesis that one does exist and that a Neural Net of sufficient layer size could detect and act on the relationship between Volume and Adjusted Closing Price.

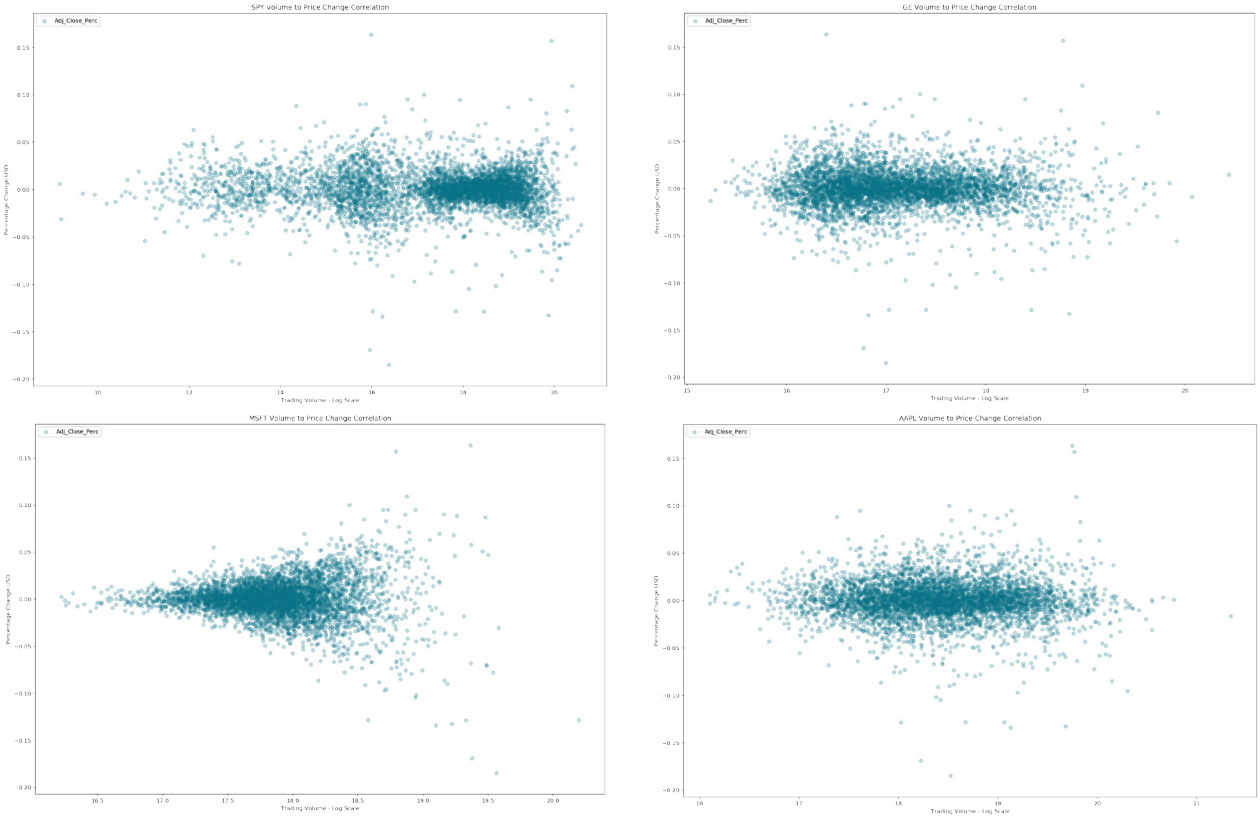


Figure 12, 13, 14, 15 (left to right, in rows): Correlation between Percentage Change in Adjusted Closing Price and Trading Volume (natural log scale) for SPY, GE, MSFT, and AAPL from 1/1/1995 to 6/18/2015.

Algorithms and Techniques

With the goal of training a deep learning model using a time series data set to predict stock prices, the project included studies primarily utilizing a Long Short-Term Memory (LSTM) model type. In contrast to a “vanilla” Neural Net, because of the internal architecture of each node, LSTMs act as a modular program as opposed to a function, optimizing predictions for sequences can be achieved with low set up overhead[15].

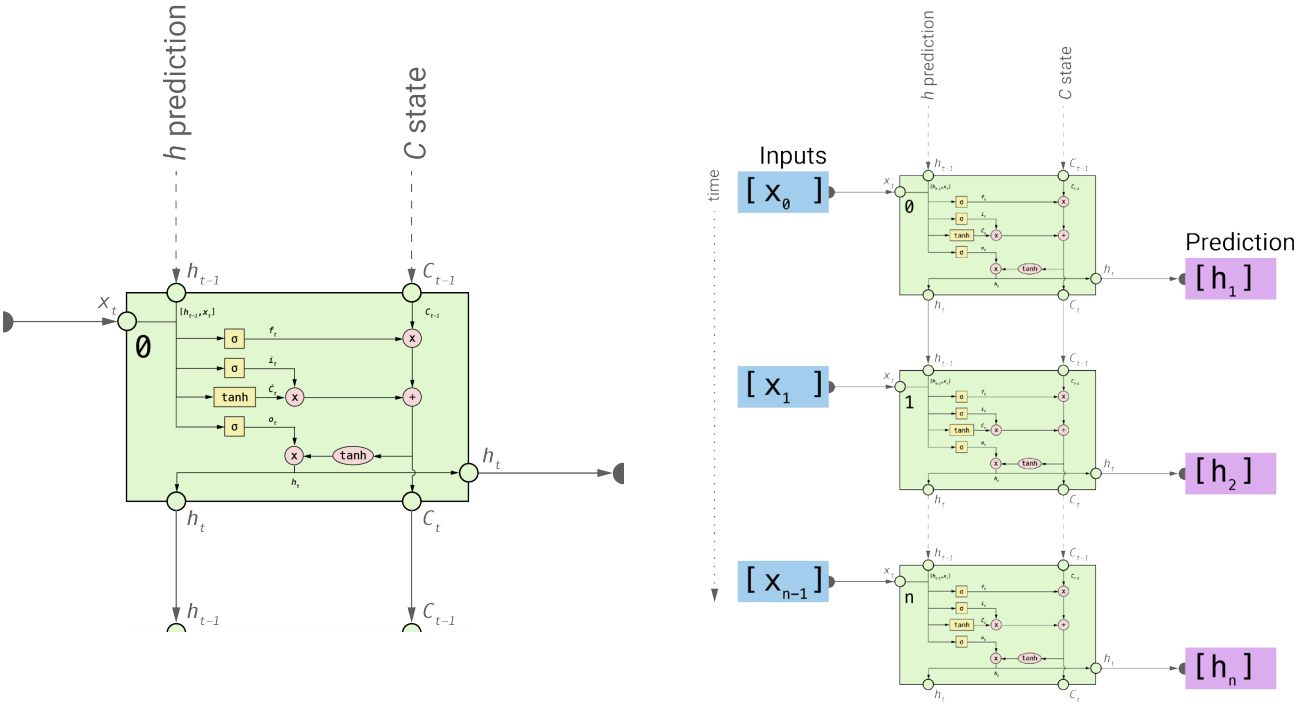


Figure 16, 17 (left to right): Long Short-Term Memory (LSTM) internal node architecture and a single LSTM node across *n* time steps. Diagrams created with inspiration from Understanding LSTM Networks[16].

For comparison purposes to this premise about LSTMs, an intermediary study was also undertaken with a Multi-Layer Perceptron model type. Additionally, the project included experiments for improving the LSTM with Ensemble techniques, specifically with the goal of increasing performance with the Volume to Price correlation or SPY to Price correlation described above.

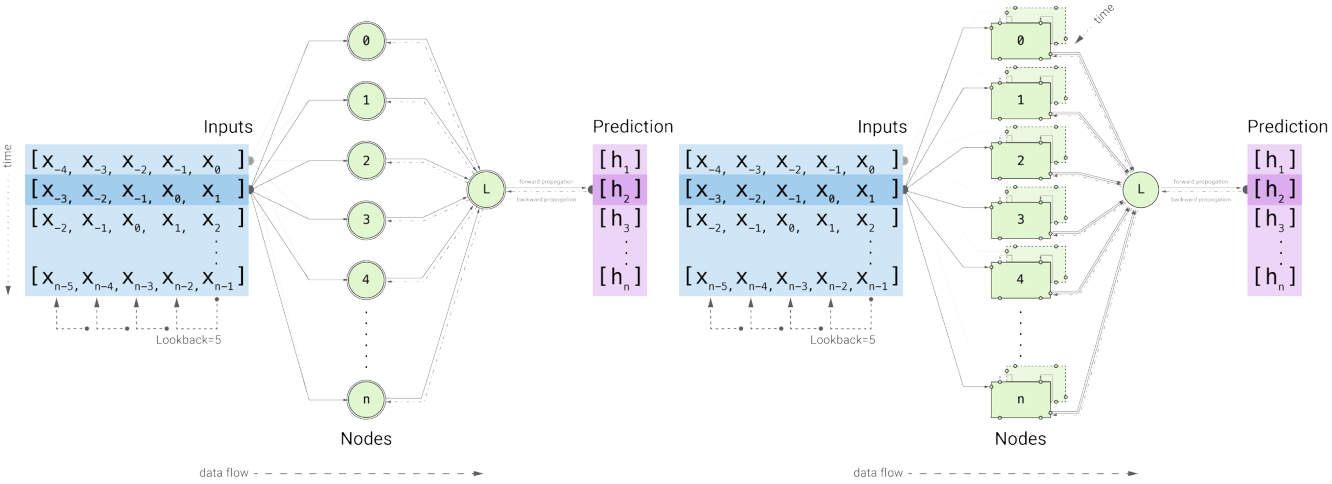


Figure 18, 19 (left to right): Multi-Layer Perceptron (MLP) Model and Long Short-Term Memory (LSTM) model architectures at time step 1. Note the connection across time steps for each LSTM node in the depth of this diagram.

In addition to adjusting the architecture of the Neural Network, the following full set of parameters can be tuned to optimize the prediction model:

- Input Parameters
  - Stock Ticker (any stock ticker String; used SPY, GE, MSFT, AAPL)
  - Preprocessing and Normalization (see Data Preprocessing Section)
- Neural Network Architecture
  - Model Type (MLP or LSTM; mostly focused on LSTM)
  - Number of Layers (how many layers of nodes in the model; used 1 or 2)
  - Number of Nodes (how many nodes per layer; tested 8, 16, 32, 64, 128)
- Training Parameters
  - Training / Test Split (how much of dataset to train versus test model on; kept constant at 68.5% and 31.5% for benchmarks)
  - Lookback (how many prior days are included in the input sequence; tested 1, 2, 5, 10, 20, 40)
  - Batch Size (how many time steps to include during a single training step; kept constant at 1)
  - Optimizer Function (which function to optimize by mimizing error; used “Adam” throughout)
  - Epochs (how many times to run through the training process; kept mostly at 1 for time savings until later studies)

Price	Prediction	PredDelta	State	Shares	IncShares	StockValue	Cash	Folio	Performance
78.8275	NaN	0	Buy	12	12	945.93	54.0698	1000	1
76.4661	77.0194	0.312782	Hold	12	12	917.594	54.0698	971.664	0.971664
76.7782	75.3711	-1.64831	Sell	0	12	0	975.408	975.408	0.975408
75.1337	73.5625	-1.80864	Pass	0	12	0	975.408	975.408	0.975408
73.3289	73.6977	0.135216	Buy	13	13	953.276	22.1323	975.408	0.975408

Figure 20: Head of Pandas DataFrame of the Virtual Porfolio for the trading algorithm. Columns include trading action, number of shares, amount of cash, and overall performance.

After the predictions were complete, a basic stock trading algorithm based on the predictions was defined to compare return performance on a virtual portfolio versus a readily available comparable system from Lucena Research[11]. Assuming trading can happen just after the closing bell, the logic of the strategy is as follows with an initial investment of x dollars on day one, for each day:

- Buy as many shares as x dollars can afford if tomorrow's predicted price is higher and no shares are owned
- Sell all shares if tomorrow's predicted price is lower
- Hold all shares if shares are owned and tomorrow's predicted price is higher
- Log the Buy/Sell/Hold action, cash value, portfolio value (Figure 20)

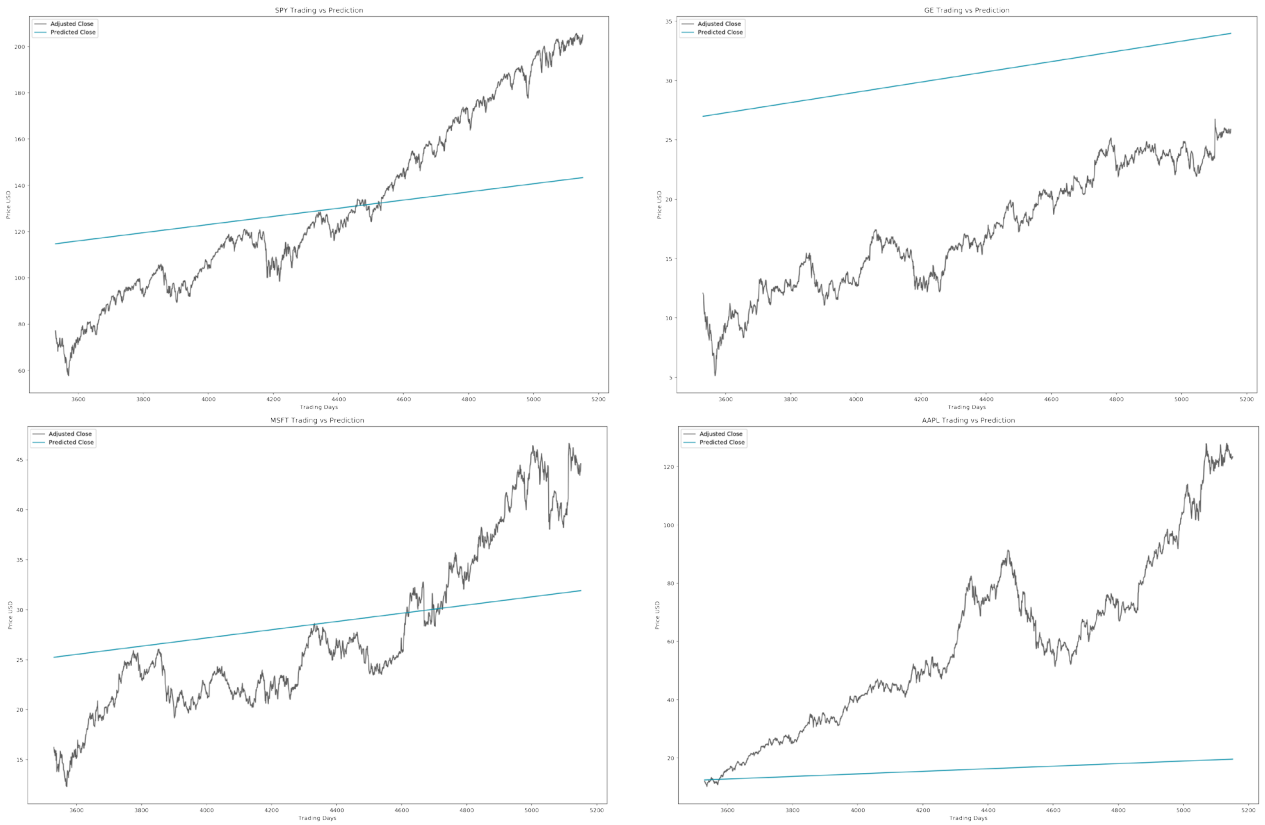


Figure 21, 22, 23, 24 (left to right, in rows): Benchmark Linear Regression models for SPY, GE, MSFT, and AAPL trained from 1/1/1995 to 12/31/2008 and visualized here during test period from 1/1/2009 to 6/18/2015. Adjusted Closing Price rendered gray and prediction rendered cyan.

Benchmark

The primary benchmark for this project is a Linear Regression model, as one of my goals is to understand the relative performance and implementation differences of machine learning versus deep learning models. This Linear Regressor was based on the examples presented in Udacity's Machine Learning for Trading course[11] and will be used for error rate comparison MSE and RMSE utilizing the same dataset as the deep learning models. As seen in these graphs, somewhat expectedly the predictions for either Linear (Figure 21-24) or Polynomial (Figure 25-28) do a poor job of fitting to the test data.

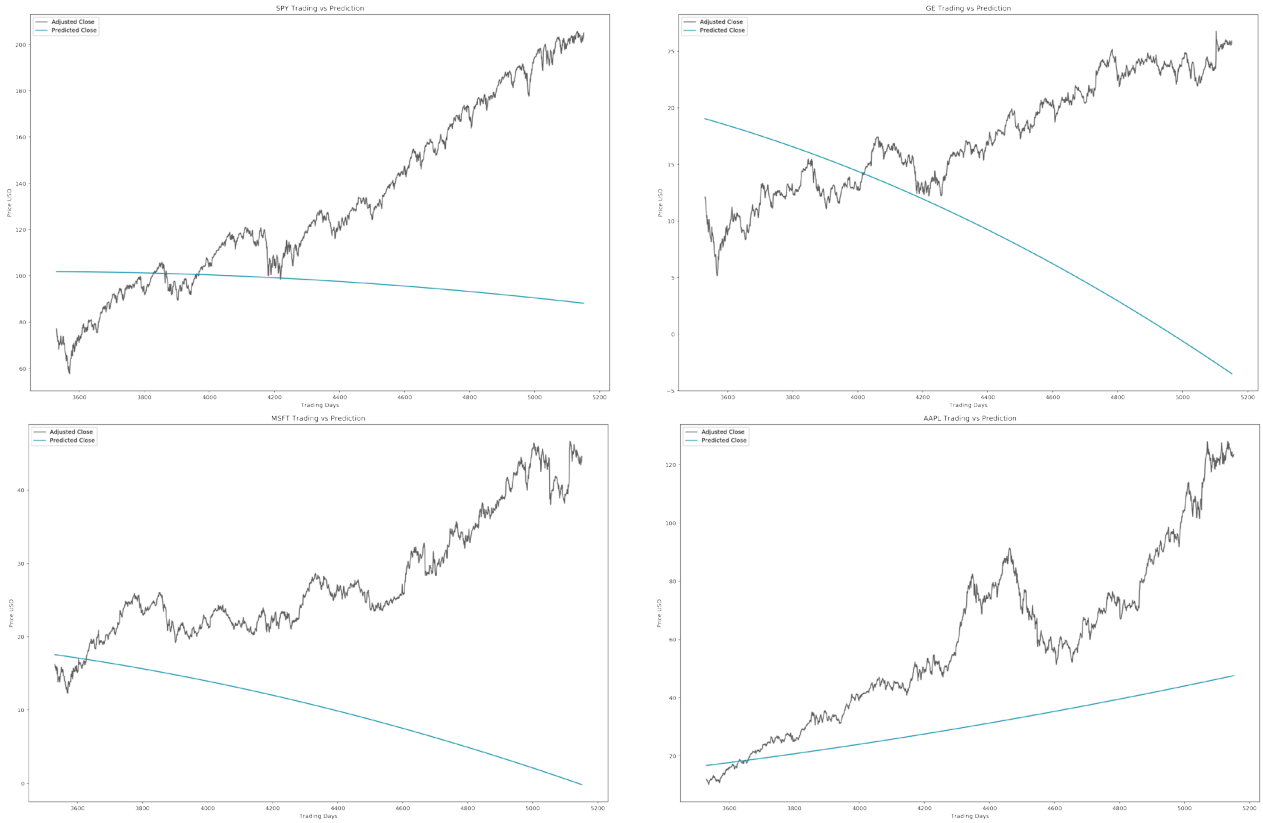


Figure 25, 26, 27, 28 (left to right, in rows): Benchmark Polynomial Regression models for SPY, GE, MSFT, and AAPL trained from 1/1/1995 to 12/31/2008 and visualized here during test period from 1/1/2009 to 6/18/2015. Adjusted Closing Price rendered gray and prediction rendered cyan.

To evaluate the performance of the basic trading strategy that is based on the model's predictions, the trading performance of the simple algorithm will be compared to a buy and hold long term strategy of each stock as well as the presented performance of the Lucena Research portfolio.

3 METHODOLOGY

Data Preprocessing

Acquiring and preprocessing the data for this project occurs following sequence, much of which has been modularized into the utilities.py file for importing and use across all notebooks:

- Request the data from the Yahoo Finance Python API
- Sort the key data points into a Pandas DataFrame for ease of organization and visualization
- Create the training and test datasets
    - Define train test split ratio
    - Define "Lookback" (how many prior days to include at each time step)
    - Define whether to normalize data or not (Optional, set to True to normalize 0.0 to 1.0 throughout for better performance than native price values)
    - Define whether to pad data or not (Optional, needed for each time step when using ensemble methods that have differing lookback values for inputs)
    - Store the incoming minimum and maximum values to scale predictions back to price values

The following sequence of images demonstrates the "Lookback" parameter. When creating the data sets, the number of prior values in the time series needs to be defined – for instance, should the model consider just today's (Lookback = 1) closing price when predicting tomorrow's, or should it take into consideration the prior four days as well (Lookback = 5)? Conceptually when usering LSTMs, this Lookback can be understood as a rough approximation to a rolling mean input. Furthermore, to work more effectively with deep learning models, the values should be normalized. In the How to Predict Stock Prices Easily lesson by Siraj Raval[17], each of the time step "windows" are normalized; however, I found that normalizing the entire dataset and not just the individual time steps gave much better results as well as offered an easier means of rescaling the predicted values back to prices in USD.

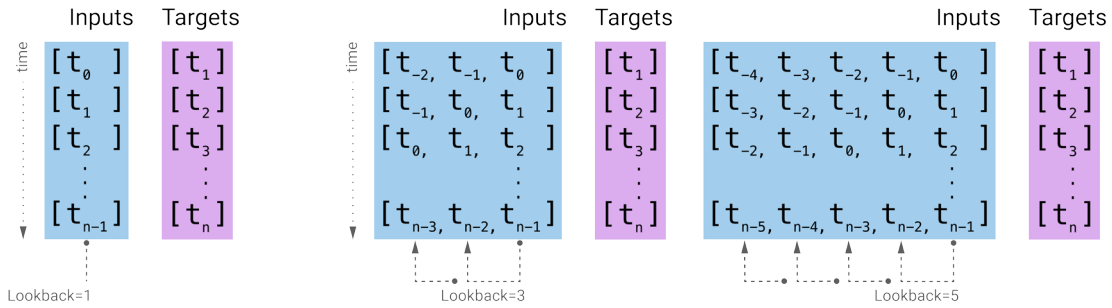


Figure 29, 30, 31 (left to right): Dataset creation with "Lookback" parameter of 1, 3, and 5. The larger the number, the more prior time steps are included in each input to the model.

Additionally, later studies that incorporate ensemble techniques to merge one Neural Network with another requires input datasets of consistent length i.e. using a Lookback of 1 and another of 5 for two branches of a model leads to an incompatible operation at the end of the architecture. To combat this, the input dataset can be created with a padding feature. In experimenting with whether that padding should be a constant such as **0** or a repeated value from the Lookback sequence and whether the padding should occur on the left-hand or right-hand side of the array of values, I found that repeating the value on the left hand side gave the best results. This minimized underfitting as well as lagging respectively with the LSTM predictions.

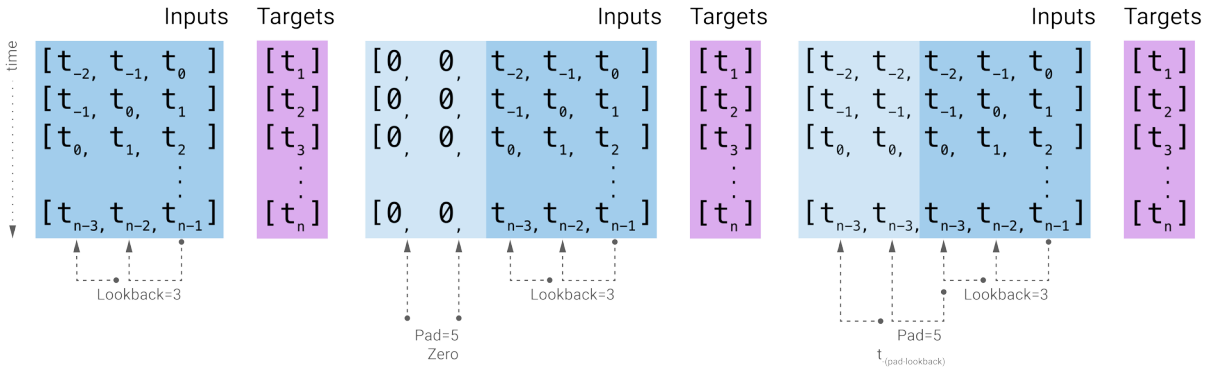


Figure 32, 33, 34 (left to right): Dataset creation with no padding, padding with 0, or padding with the last time step item. Padding on the left hand side vs the right hand side gives the LSTM nodes the most recent item as the changing value for each time step.

Implementation

Once the stock ticker has been defined and the trading data has been preprocessed, the implementation process occurs consistently through all studies as follow:

- Define training parameters
- Define model architecture (see below)
- Compile the model
- Train the model
- Generate incremental predictions
- Calculate error
- Rescale predictions to price values
- Plot actual and predicted prices
- Save error value, graph, and html of notebook

Across the sequence of studies the model architecture step above included versions that utilize one of three node type arrangements – Multi-Layer Perceptron (Figure 18), Long Short-Term Memory (Figure 19), or LSTM Stacked. Figure 35 below shows the stacked architecture at the second time step of training, where the first layer of LSTMs maintain the time series for processing in the second layer.

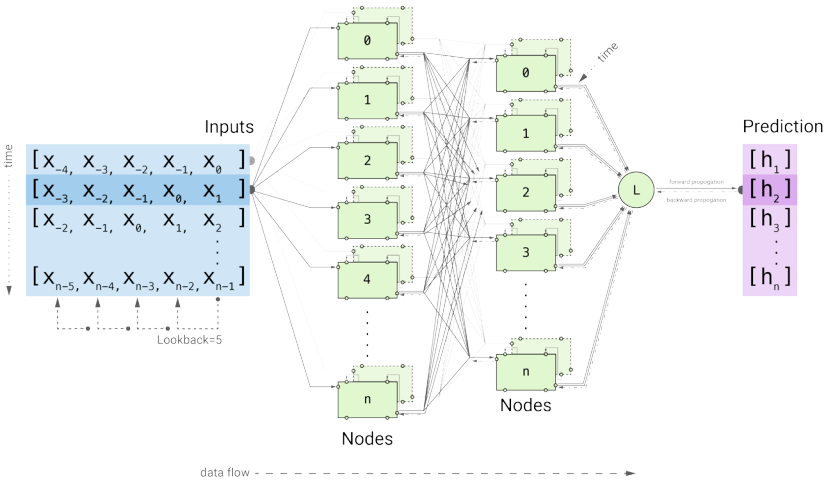


Figure 35: Long Short-Term Memory nodes in a stacked architecture at time step 1. Note the connection across time steps for each LSTM node in the depth of this diagram.

A hybrid version of the LSTM architecture was also experimented with to see if merging the best versions of each: Adjusted Close LSTM; Adjusted Close with a Stacked LSTM; and Volume LSTM would result in a lower error. The model corresponding to each combination of these three branches was also implemented for comparison.

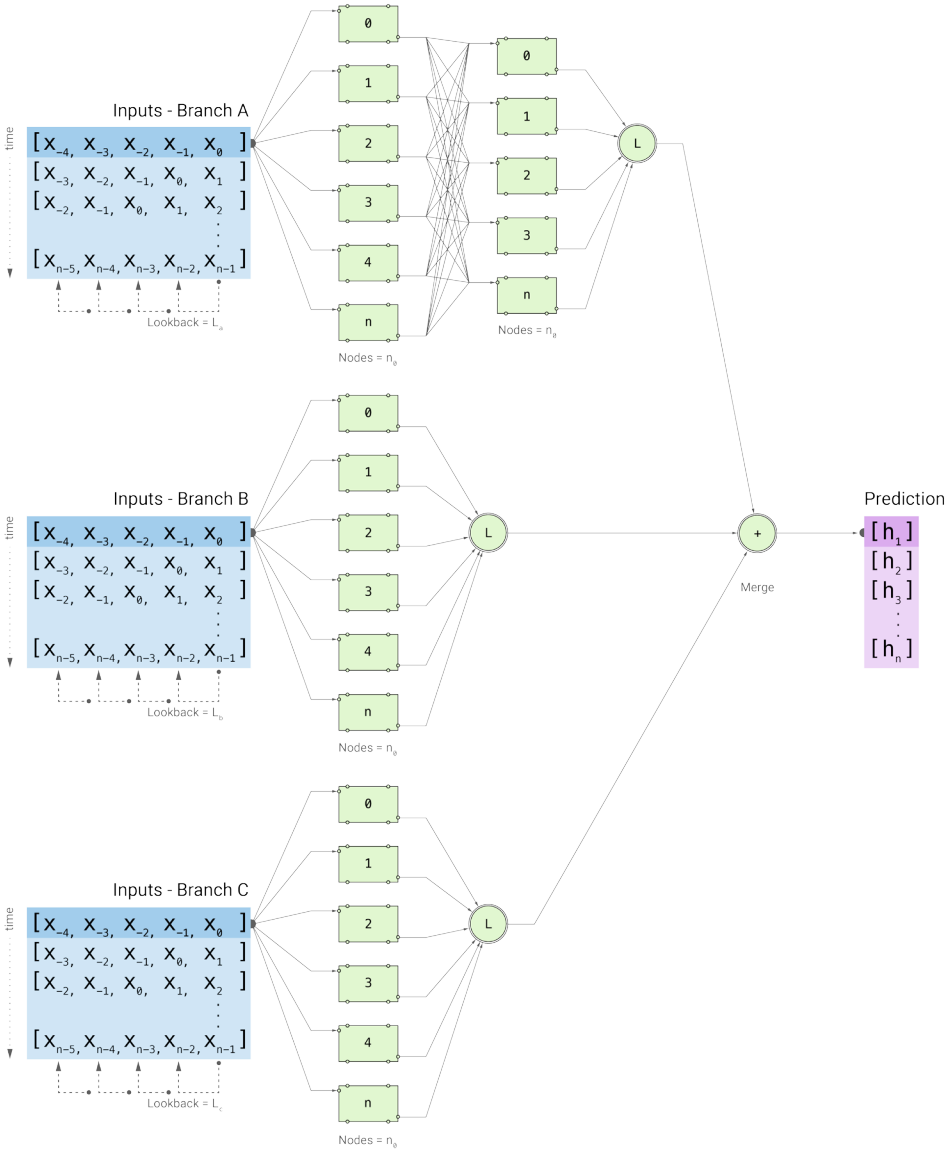


Figure 36: Three merged branches of a hybrid LSTM architecture at time step 0. Additional time steps, connections across time steps, and backpropagation are not shown for clarity in this diagram.

Refinement

Each study in the iterative sequence of development was tested either with four popular stocks tickers or across a range of parameters for the model using the SPY ticker for optimizing performance. After examining the stock data (Study 00) and establishing both regression (Study 01 + 02) and Multi-Layer Perceptron (Study 03) benchmarks, the remaining studies established a Long Short-Term Memory model (Study 04) and then iterated through variations of the model's parameters or architecture.

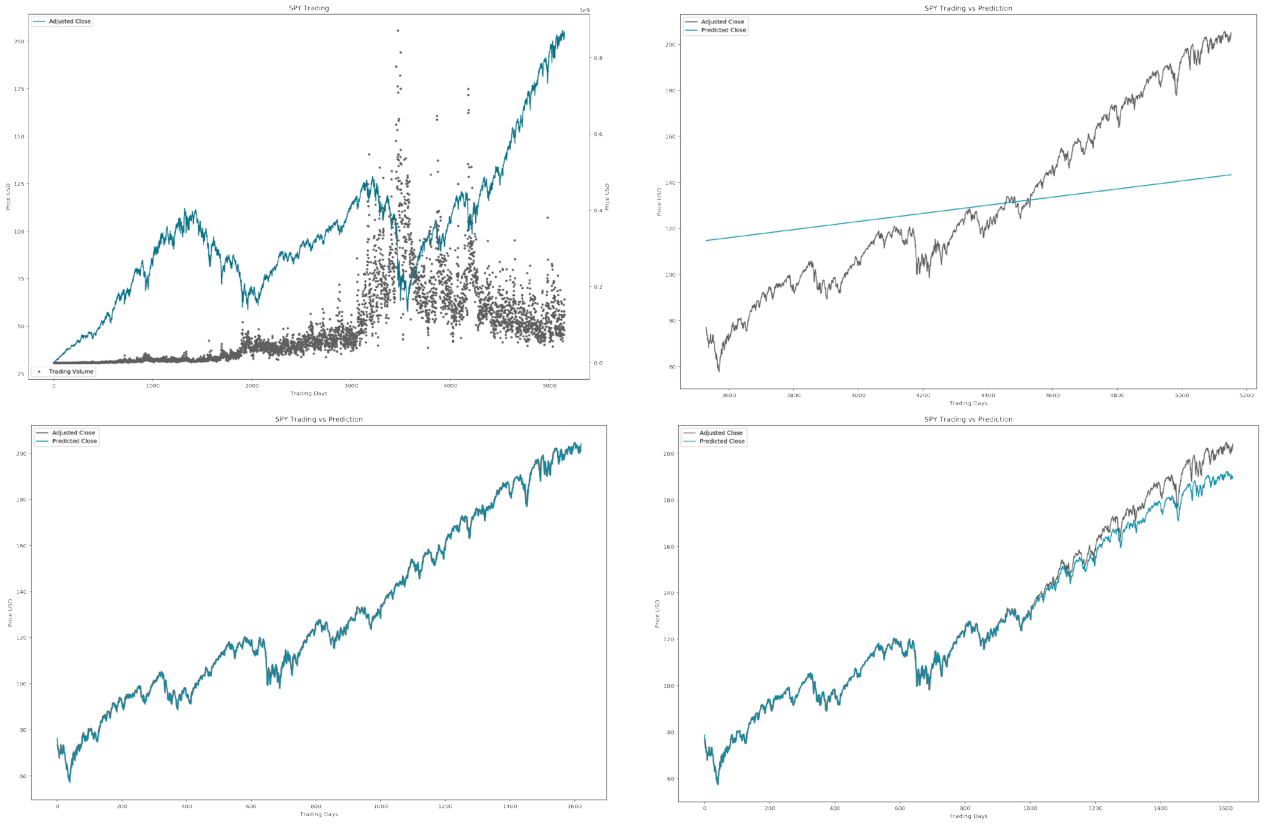


Figure 37, 38, 39, 40 (left to right, in rows): Study 00 SPY Adjusted Close and Trading Volume, Study 01 SPY Linear Regression Benchmark model, Study 03 SPY MLP model, and Study 04 SPY LSTM model.

The details for the entire sequence of studies developed for the project include:

- Study 00 // Data Exploration // Gathering and Analyzing Stock Data
- Study 01 // Benchmark Model // Linear Regression with Sci-kit Learn
- Study 02 // Benchmark Model // Polynomial Regression with Sci-kit Learn
- Study 03 // Deep Learning Model // Multi-Layer Perceptron with Keras and Tensorflow
- Study 04 // Deep Learning Model // Long Short-term Memory "LSTM" with Keras and Tensorflow
- Study 05 // Deep Learning Model // LSTM with Variable "Lookback" Dataset (Adjusted Close)

- Study 06 // Deep Learning Model // LSTM with Variable “Lookback” Dataset (Trading Volume)
- Study 07 // Deep Learning Model // LSTM with two “Stacked” LSTM Layers (Adjusted Close)
- Study 08 // Deep Learning Model // LSTM with two “Merged” LSTM Branches (Adjusted Close + Volume)
- Study 09 // Deep Learning Model // LSTM with two “Merged” LSTM Branches with “Zero” Padding Datasets (Adjusted Close + Trading Volume)
- Study 10 // Deep Learning Model // LSTM with two “Merged” LSTM Branches with “i” Padding Datasets (Adjusted Close + Trading Volume)
- Study 11 // Deep Learning Model // LSTM with two “Merged” LSTM Branches (Adjusted Close + SPY Adjusted Close)
- Study 12 // Deep Learning Model // Final LSTM with Tuned Dataset (Adjusted Close) and Parameters
- Study 13 // Deep Learning Model // Final LSTM with Tuned Dataset (Adjusted Close) and Parameters using four previously unused tickers

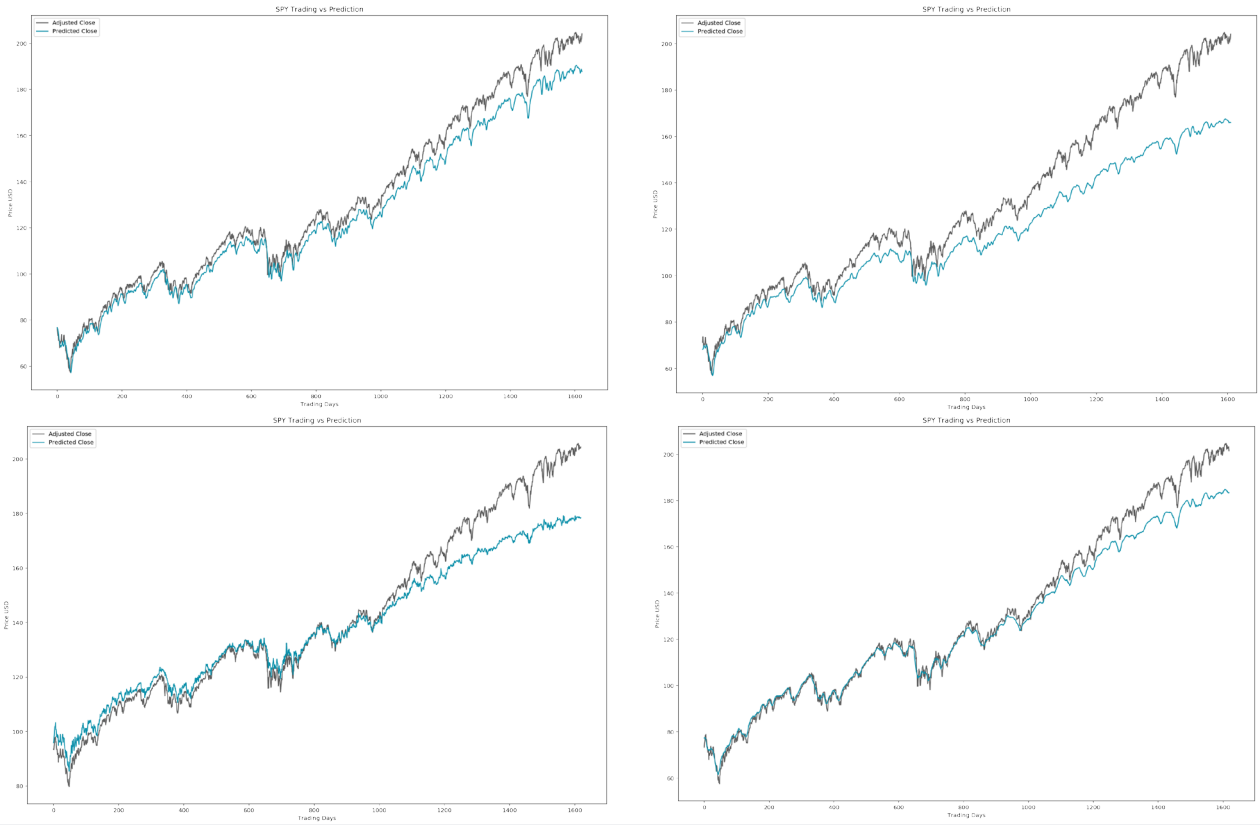


Figure 41, 42, 43, 44 (left to right, in rows): Study 05 SPY Variable “Lookback,” Study 07 SPY Stacked LSTM architecture, Study 09 SPY Merged (Adjusted Close + Trading Volume) LSTM architecture, and Study 10 SPY Merged (Adjusted Close Lookback 1 + 10) LSTM architecture.

A catalog of the results for each study can be found as a python notebook, html page, and exported graph in the Github repository. For each change in the model architecture or parameters used, the error rates (MSE and RMSE) were logged and compared to prior iterations to decide if the changes were improvements or not in moving towards the most robust deep learning model.

MSE	SPY	GE	MSFT	AAPL
STUDY 01	0.1219	0.7612	0.0867	0.6354
STUDY 02	0.3895	0.8083	0.9899	0.3034
STUDY 03	0.000056	0.000082	0.000127	0.000193
STUDY 04	0.000074	0.000075	0.000163	0.000100
STUDY 05	0.000086	N/A	N/A	N/A
STUDY 06	0.000084	N/A	N/A	N/A
STUDY 07	0.000080	0.000072	0.000146	0.008120
STUDY 08	0.000224	0.000576	0.000181	0.000118
STUDY 09	0.000790	0.000110	0.001070	0.013640
STUDY 10	0.001727	0.000131	0.001137	0.003160
STUDY 11	0.000079	0.000134	0.000413	0.001514

Figure 45: Study 01 - Study 11 Mean Squared Error Log. Highlighted are the best results per ticker before selecting final model. N/A indicated where parameter tuning with SPY was focus of study.

The best error rates highlighted (Figure 45) indicated that from a model architecture pespective, either the basic LSTM (Study 04) or Stacked LSTM (Study 07) variations displayed the best results. The studies that incorporated a second branch with a longer Lookback, Trading Volume, or additional SPY ticker did not perform as well and were removed for the final studies. Additional discussion on logic for selecting the final architecture can be found in the next section. Other parameters tuned through the process and used in the final studies because they performed the best include Lookback value of 1 and 128 nodes in the LSTM layer.

4 RESULTS

Model Evaluation and Validation

Comparing the sequence of studies that focused on Long-Short Term Memory Neural Networks, the two model architectures that performed best across the four stock tickers tested were the Basic single layer LSTM (Study 04) and the Stacked LSTM (Study 07) variations. Erring on the side of simplicity and faster compute times and because the range of performance across the tickers was narrower for Study 04, for the final study I selected the Basic LSTM architecture with a consistent Lookback value of 1 and I increased the overall number of epochs to 5.

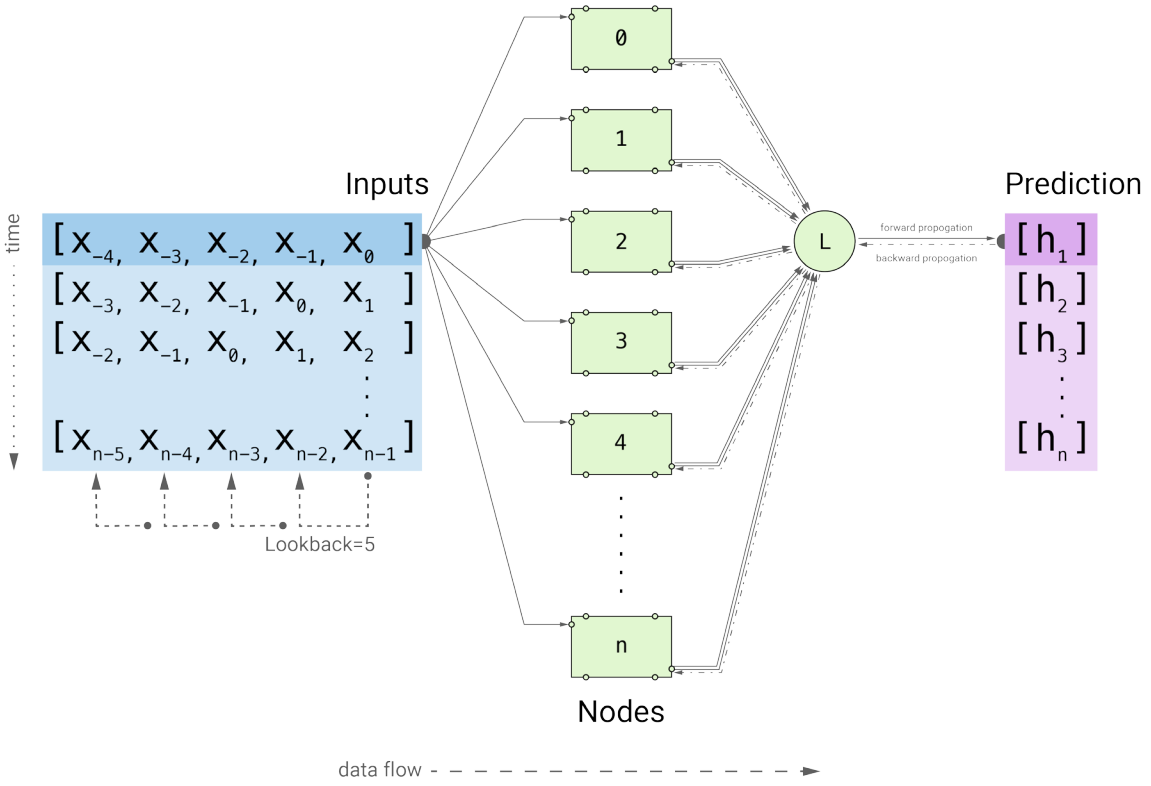


Figure 46: Final model architecture of a Long Short-Term Memory (LSTM) Neural Net at time step 1.

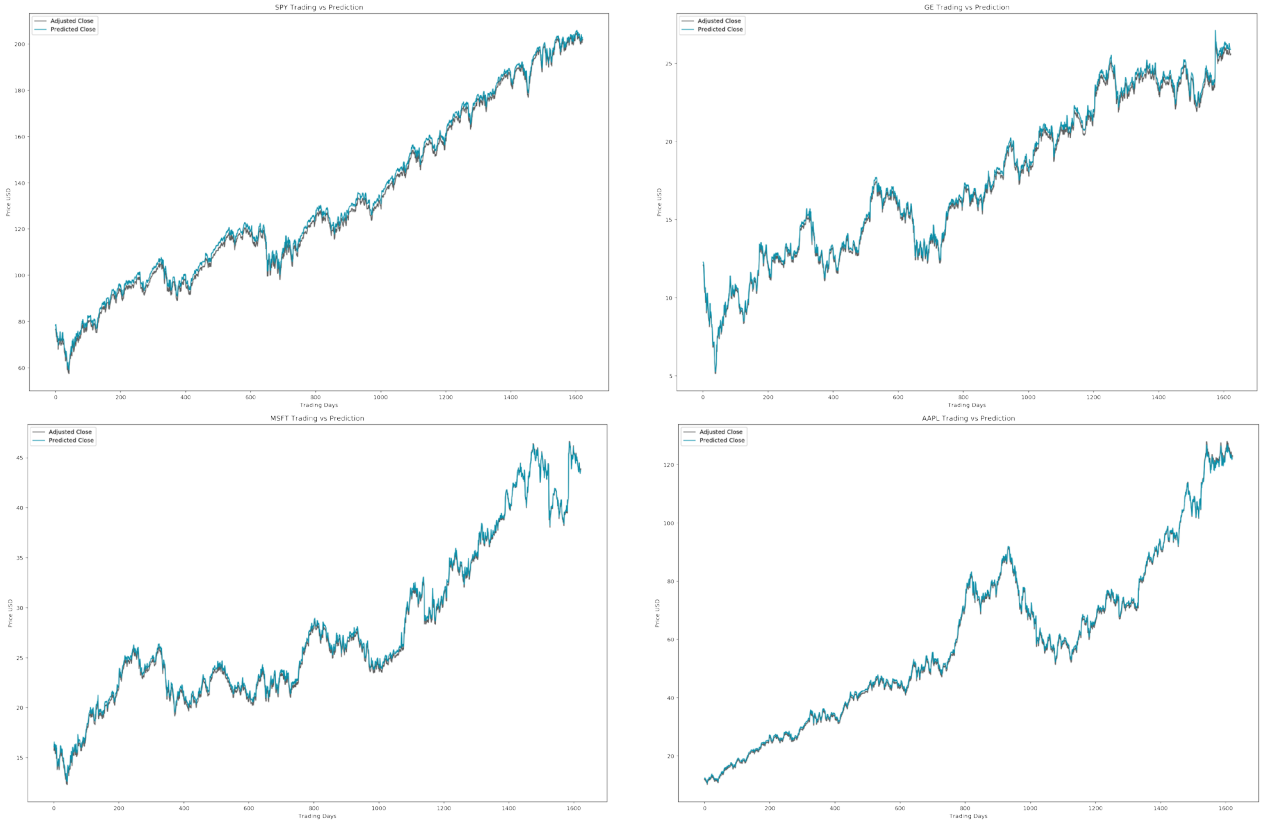


Figure 47, 48, 49, 50 (left to right, in rows): Final Study LSTM models for SPY, GE, MSFT, and AAPL trained from 1/1/1995 to 12/31/2008 and visualized here during test period from 1/1/2009 to 6/18/2015. Adjusted Closing Price rendered gray and prediction rendered cyan.

Based on the close fit in the prediction graphs (Figures 47-50) and the consistency of the final error metrics (Figure 51), this solution generalizes well across the stocks selected. Furthermore, based on the variability in the shape of the closing price across the entire test period as well as the volatility in the data across more localized time frames this model should be understood as robust enough for the problem given any stock ticker and adequate size of the corresponding training data. It should be noted that the error rates are both much lower than expected and extremely low compared to the first two non-deep learning models. The latter observation can be attributed to an incremental prediction routine implemented for the MLP and LSTM studies, which retrains and predicts one day ahead at a time and not for the entire date range shown for the test period. Knowing this would skew the difference between the linear regression and deep learning error rates, I was still surprised at how well the predictions tracked the actual stock values.

FINAL STUDY	SPY	GE	MSFT	AAPL
MSE	0.000178	0.000138	0.000139	0.000095
RMSE	0.013357	0.011735	0.011771	0.009769
AVERAGE DELTA PRICE (USD)	0.031085	0.004274	0.006115	0.012173

Figure 51: Final Study Error Rates for SPY, GE, MSFT, and AAPL. MSE demonstrates consistency across the various tickers.

With these results, at first I was certain there was a data leak across the input and targets or an error in my program. Given that I quadruple checked the dataset creation process, that some of the more complex model architectures perform poorly over time, and that the results of the final study also translated to a previously unseen dataset (see next section), it is my conclusion that the unreasonable effectiveness[15] of the internal workings of LSTMs paired with an incremental prediction routine just works that well.

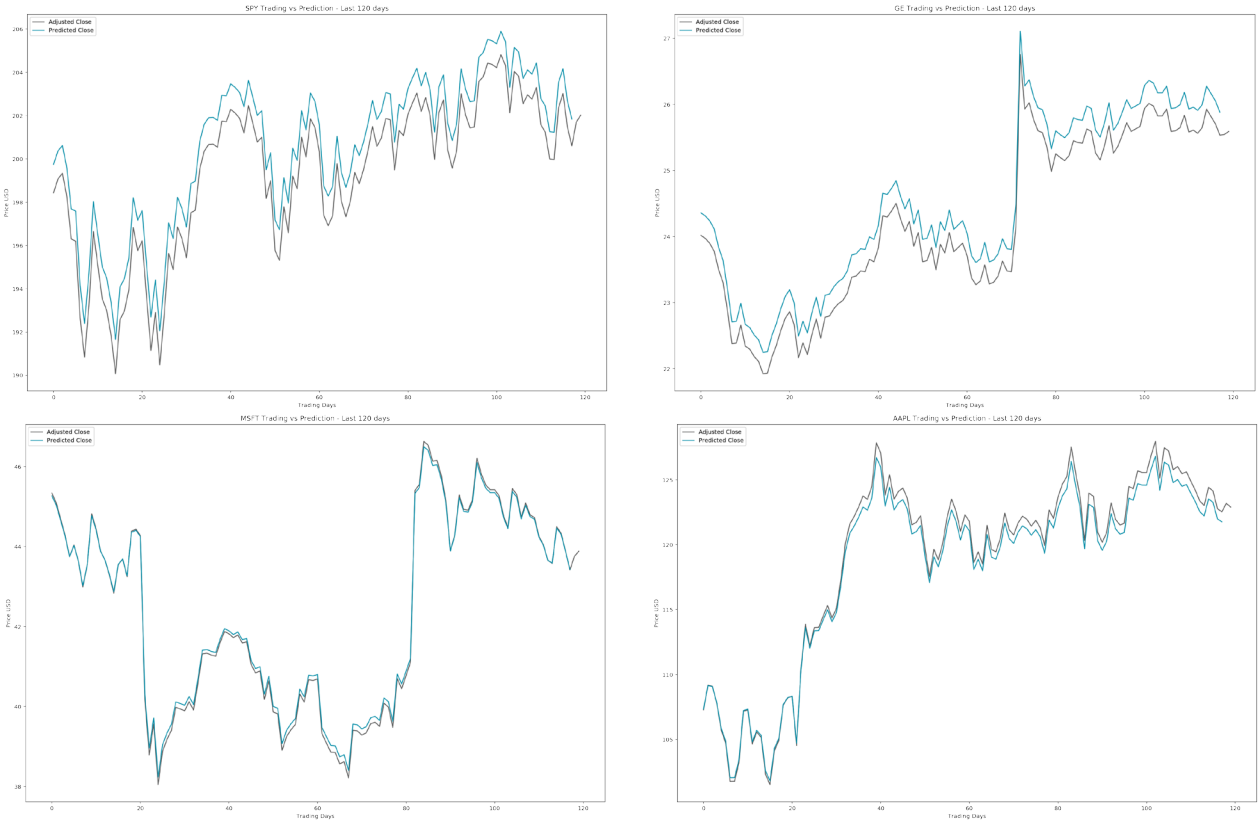


Figure 52, 53, 54, 55 (left to right, in rows): Last 120 trading days of Final Study LSTM models for SPY, GE, MSFT, and AAPL trained from 1/1/1995 to 12/31/2008. Adjusted Closing Price rendered gray and prediction rendered cyan.

This solution does however, have its limitations. Zooming into the final 120 days of the test data set compared to the first 120 days, there is a tendency for the predictions to accumulate a delta in predicted price over time (Figures 52-55). Additionally, the model's ability to predict far into the future is limited. The prediction routine could be changed to forecast prices farther into the future but the error rates will not hold under that circumstance.

Justification

Comparing the benchmark non-deep learning Linear Regression (Study 01) to the final LSTM model (Study 12) the Mean Squared Error improvement ranges from 623x to 6688x, demonstrating 2 to 3 orders of magnitude better performance with the Average Delta Price between actual and predicted Adjusted Closing Price values was three cents or less. The significant jump in improvement in the development of the project starts at the Multi-Layer Perceptron model (Study 03); however, the error rates of the final model are within a more consistent range suggesting that the model generalizes better to diverse datasets.

To further test the capabilities of the architecture and parameters of the final model on previously unseen data, I replicated Study 12 with four new stock tickers (TSLA, AMZN, GOOG, and NVDA) across a shorter and more recent time frame of 4/1/2014 to 4/1/2017. Despite less data for training, the predictions were still consistently accurate. The Average Delta Price was larger for these four studies, but the graph tracked the behavior of the stocks well. With this performance in hand, it is my conclusion that the final model is both an accurate and a robust solution to predicting stock prices.

FINAL STUDY	TSLA	AMZN	GOOG	NVDA
MSE	0.001255	0.001190	0.000658	0.000861
RMSE	0.035425	0.034493	0.025651	0.029351
AVERAGE DELTA PRICE (USD)	0.178667	0.713370	0.236584	0.088481

Figure 56: Final Study Error Rates for TSLA, AMZN, GOOG, AND NVDA. MSE demonstrates consistency across the various tickers despite shorter training period.

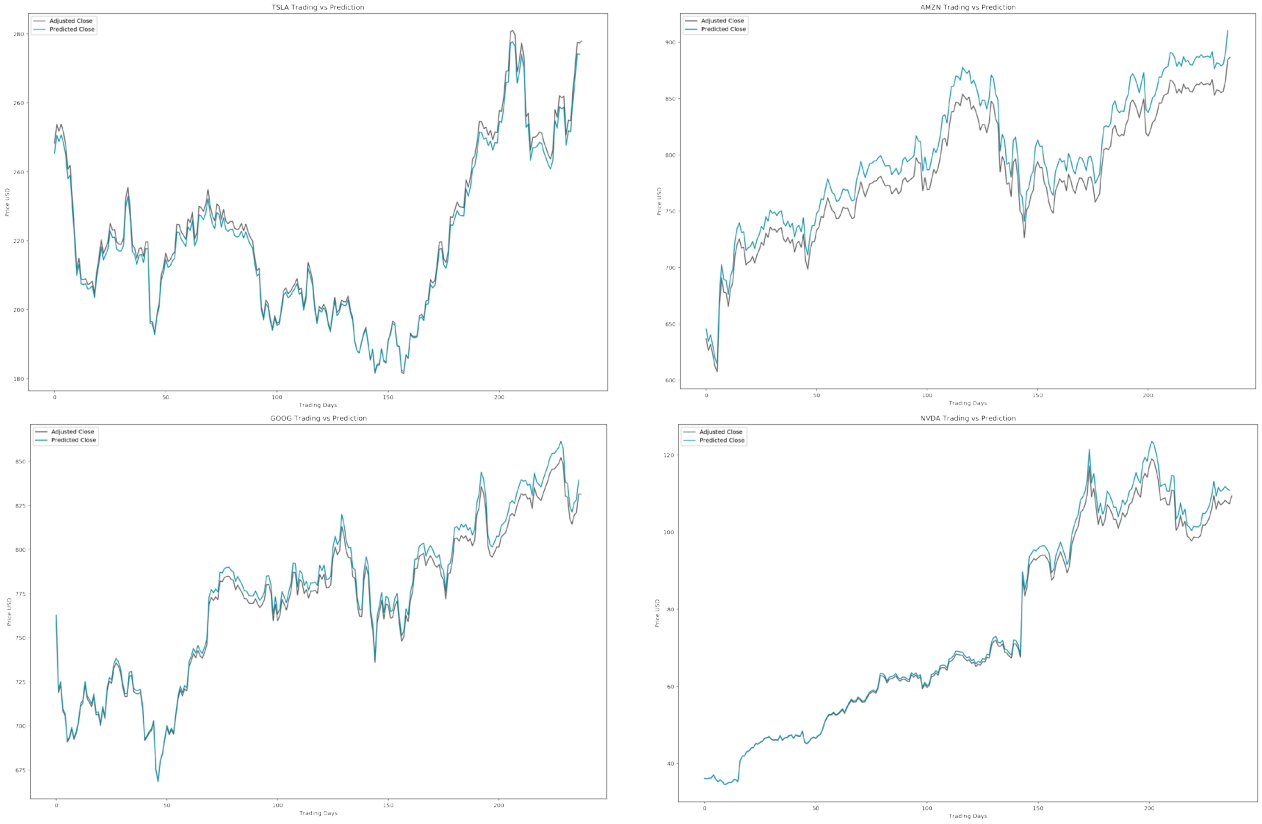


Figure 57, 58, 59, 60 (left to right, in rows): Final Study LSTM models for TSLA, AMZN, GOOG, and NVDA trained from 4/1/2014 to 8/31/2016 and tested from 9/1/2016 to 4/1/2017. Adjusted Closing Price rendered gray and prediction rendered cyan.

5 CONCLUSION

Free-Form Visualization

Given the high degree of prediction accuracy and the claims of Artificially Intelligent Hedge Funds, Deep Investing Firms, and Lucena Research, I developed a simple trading algorithm to utilize the incremental predictions of the final model to compare results. Because the final study LSTMs demonstrated a slight tendency to accumulate a delta in predicted price but still tracked the daily direction well, this strategy should focuse on the up and down predictive capability of the model.

The logic for the strategy is as follows:

- If shares are held:
  - If the price tomorrow is predicted to be higher than today hold the shares
  - If the price tomorrow is predicted to be lower than today sell the shares
- If shares are not held:
  - If the price tomorrow is predicted to be higher than today buy the shares
  - If the price tomorrow is predicted to be lower than today pass on any action
- Store the stock price, predicted price, trading action, shares held, and portfolio value

The premises for the trading strategy employed include:

- Trading can be executed at the closing bell to use the Adjusted Closing Price to predict tomorrow's price
- Only a single stock is considered for trading
- There is no cost to execute a trade
- There are no limits to how frequent or with what volume trading can be executed
- Capital Gains Tax is not incorporated

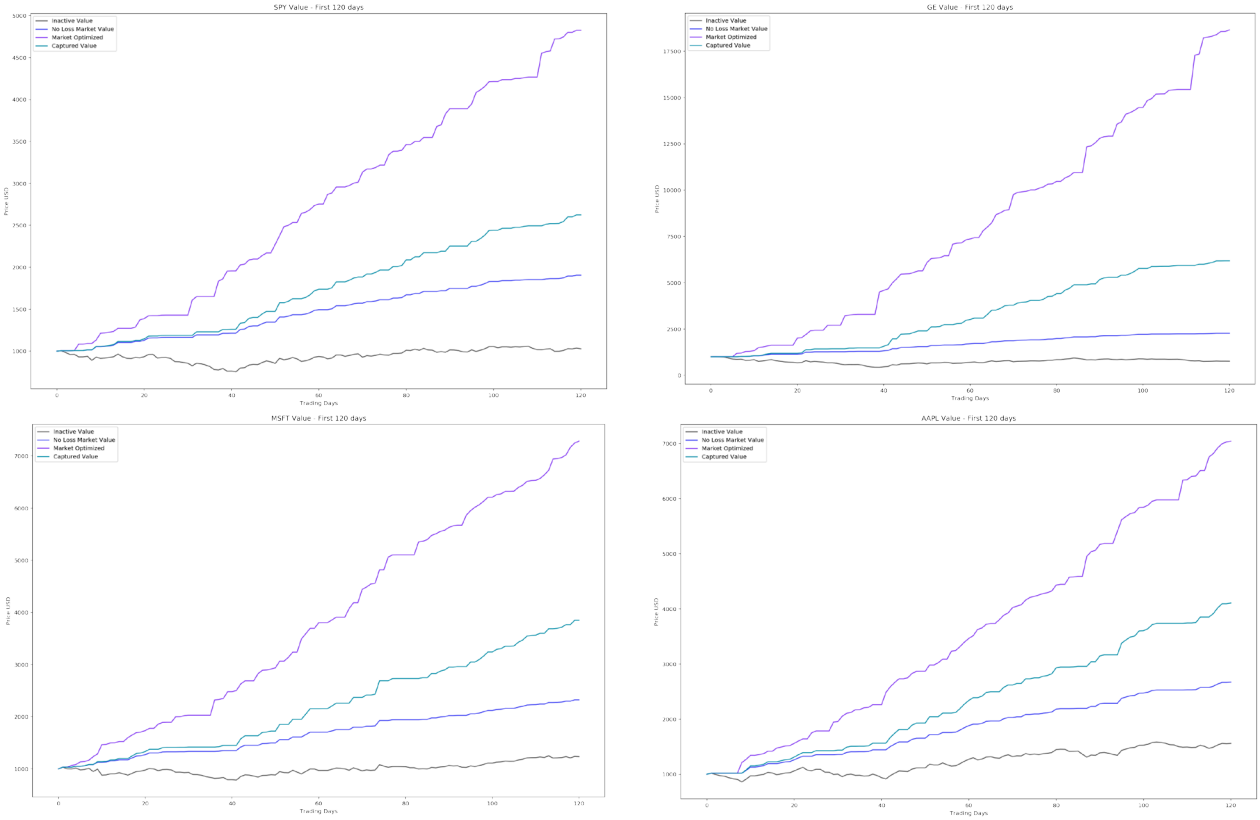


Figure 61, 62, 63, 64 (left to right, in rows): Benchmark Linear Regression models for SPY, GE, MSFT, and AAPL trained from 1/1/1995 to 12/31/2008 and visualized here during test period from 1/1/2009 to 6/18/2015. Adjusted Closing Price rendered gray and prediction rendered cyan. Polynomial regression studies can be found in the Appendix: A2

With these assumptions, over time this trading strategy does extremely well. Initial comparisons of the performance (Captured Valued) against a passive strategy of buying and only holding for the five plus years (Inactive Value) were hard to believe, upwards of 1000x. I then compared the results to if the same number of initial shares for the passive strategy never had any losses (No Loss Market Value), Because of the incremental increase in cash value from the buying and selling action, the virtual portfolio can accumulate more shares over time. Coupled with no cost to trade or Capital Gains tax, this leads to an exponential increase in portfolio value over time. In order to compare performance to a metric that is not limited in the same way the other benchmarks are, I also calculated a Market Optimized value that automatically increases the number of shares in parallel to the trading algorithm and shares the properties of the No Loss Market Value benchmark. This would be absolute perfect performance on the same playing field as the virtual portfolio. Since it outperforms the trading algorithm, the performance achieved is more believable given the assumptions albeit challenging to compare against real world scenarios. It is also interesting to note that the performance of this final study also outperforms the 185% relative performance of the Lucena Research platform in our virtual, non real-world setting.

Reflection

The high-level process I employed for developing this project included the following goals and corresponding key steps:

- Gain access to stock data
    - Develop a data request module using the Yahoo Finance Python API
- Preprocess the data gathered
    - Sort the relevant data points into a convenient format and plot
    - Create training and test datasets with parametric options (such as Lookback)
- Create a deep learning model
    - Define the model's architecture using Keras
    - Train the model
- Analyze the model's performance
    - Capture the model's predictions
    - Calculate the error
    - Plot the results

Because my primary goals of the project included developing a deep learning model as well as doing a survey of model architectures, I incrementally developed the series of studies and captured the notebooks as HTMLs, the graphs as images, and the error rates for each study. This was helpful to isolate what was changing from one model architecture variation from the next, particularly since much of the real work happens internal to each LSTM node; however, this also meant a lot of time spent organizing the large collection of files and re-running prior studies when bugs were identified. Additionally, my desire to use Keras and deep learning models meant finding good, clear examples on the internet to reference when building my own models. Despite these challenges, having achieved the results (as surprisingly good as they were) and having utilized this strategy for development, I think that I have a much more solid handle of this category of time series deep learning models.

During the process, I found a few interesting results and insights. Compared to my hypothesis that more data features would result in better predictions, the models that merged Adjusted Close inputs with Trading Volume did worse than just Adjusted Close. Having more days included in each step in the input dataset by way of the Lookback parameter also decreased the performance of the models. For volatile markets and LSTM Neural Networks, using Occam's razor to guide the model architecture (simpler is better) and incorporating the most up to date data as input leads to the best results, then let the LSTM Nodes to work their magic.

Improvement

To improve the performance of this deep learning stock price predictor with ongoing development, I would want to utilize additional tools to make the process and assessment easier, include more dynamic data to gauge its ability to adapt, and put the trading algorithm to work in a real-world context. With all the study version and output management (and all the waiting for each study to compute on my laptop CPU), I would first experiment with using Floyd Hub[18] to run each model and have easy recall of the results. Incorporating Tensorboard[19] would also be of use in understanding the architecture of the Neural Network. I briefly attempted to do so with Keras but found it challenging – rewriting the network in vanilla Tensorflow[20] and using Tensorboard would be a fun option. Given that the final model's performance at a daily interval was surprising, it would be interesting to see how well it performs if it was given stock data that was captured hourly as well as included after-hours and pre-market trading. Testing this would also allow for more real-world assumptions to be included in the trading algorithm. Gathering an understanding of how the LSTM plus daily trading approach would do "in the wild" on a platform such as Quantopian[21] with taxes and market volatility would be the ultimate test of performance.

A REFERENCES

Footnotes

1. History of Algorithmic Trading Shows Promise and Perils // Bloomberg
2. Money Managers Seek AI's 'Deep Learning' // Financial Times
3. The Rise of the Artificially Intelligent Hedge Fund // Wired
4. Stocks Neural Net
5. Deep Learning for Time Series Modeling // Enzo Busseti, Ian Osband, and Scott Wong
6. Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks // Lawrence Takeuchi and Yu-Ying (Albert) Lee
7. Will AI-Powered Hedge Funds Outsmart the Market? // MIT Technology Review
8. Time Series Prediction With Deep Learning in Keras // Machine Learning Mastery
9. Keras // Deep Learning library for Theano and TensorFlow
10. Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras // Machine Learning Mastery
11. Machine Learning for Trading // Udacity
12. Yahoo Finance
13. Yahoo Finance // Python API
14. Does Trading Volume Affect Stock Price? // Motley Fool
15. The Unreasonable Effectiveness of Recurrent Neural Networks // Andrej Karpathy
16. Understanding LSTM Networks // Christopher Olah
17. How to Predict Stock Prices Easily // Siraj Raval
18. Floyd Hub
19. Tensorboard
20. Tensorflow
21. Quantopian