Capstone Project CS467
November 30, 2018
Draco:WEB8 - Stock Market Forecasting
Final Report

Daniel Beckham
James Fitzwater
Bridget McGinn

Our website: https://draco-web08.appspot.com/

I. Introduction

As our semester comes to a close we are ready to submit our CS467 Capstone Project. Our initial intentions were to create a web-accessible interface that displays historical performance for a set of stocks and makes simple future projections. Our group has successfully created a database-backed website from which users can find current and historical stock data, view the Simple and Exponential Moving Average Data, view team-generated information about each stock's percent change, and 50-day predictions for each stock. These predictions were made using a specialized Recurrent Neural Network (RNN) called Long Short-Term Memory (LSTM) which builds on the concept of short-term memory cells by adding in a long-term memory gate.

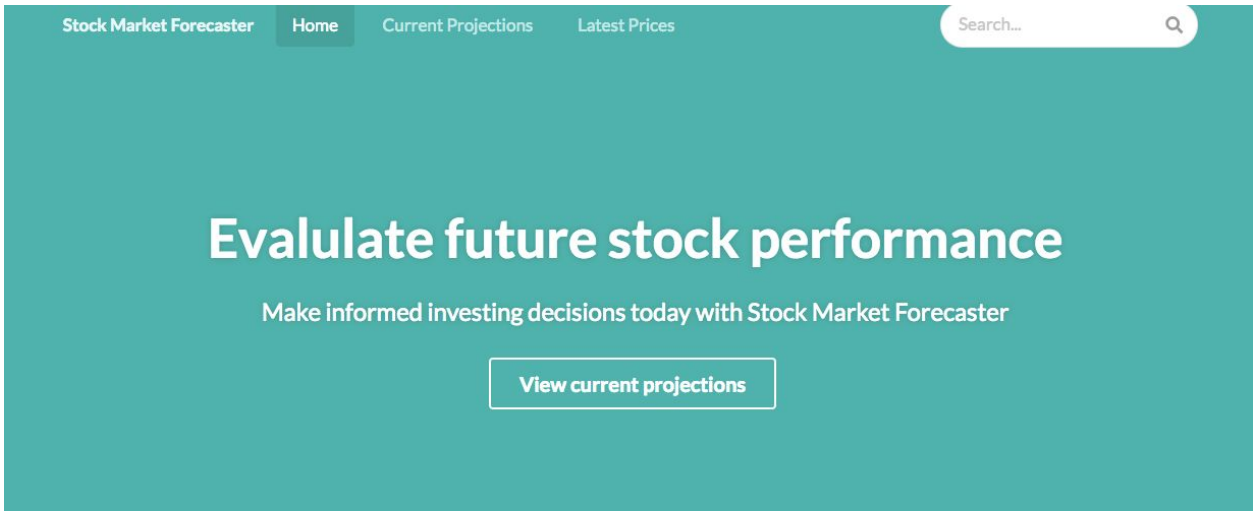
This project has enabled our group to utilize many aspects of our education from OSU, including front and back-end web development, databases, cloud application development, API communications, as well as combining different languages into a singular output. The project back-end and Machine Learning code were done in Python, with particular emphasis on the numpy, pandas, matplotlib, and keras libraries. Because we had not studied these Python libraries before, we spent significant energy reading books, watching tutorials, and following blogs to develop the necessary understanding for completing this project. The front-end of this project was created using React and React Stock Charts for sleek visualization of the data.

II. User Perspective

The website enables users to search for stock information over a range of stocks and find 50-day-out prediction of those prices. The user can see the 50-day Simple and Exponential moving averages of each individual stock. The user can also access Bollinger Bands on their Stock Chart. (These are defined below.) Lastly, the user has access to a page of all the available stocks with daily updated information on cost and change.

III. Usage Instructions

When the user accesses our webpage (https://draco-web08.appspot.com/) they have an easy to navigate splash page which immediately gives them the option to view current projections as machine learning was the primary objective for this website. They can also navigate to the current projections or latest prices using the navigation bar at the top of this home page:

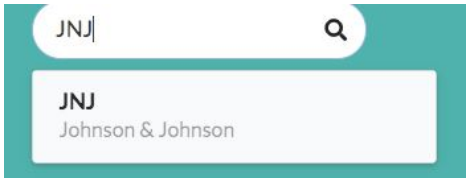


This URL is https://draco-web08.appspot.com/predictions and brings the user to this interface:

Symbol ▲	Name	50-Day Prediction	Change	Percent Change
AAPL	Apple Inc.	175.361	-3.22 ↓	-1.80%
AXP	American Express Company	109.477	-2.79 ↓	-2.49%
BA	The Boeing Company	322.191	-24.57 ↓	-7.09%
CAT	Caterpillar Inc.	104.678	-30.99 ↓	-22.84%
CSCO	Cisco Systems, Inc.	47.30	-0.57 ↓	-1.19%

On this page, they can view the expected price in 50 days for each stock, the change in price from the current price, and the percent change.

Users can either click on a stock in the list or search in the search box for stocks. It will populate with available stocks contained in the database for display. Using the search function to locate a stock, for example JNJ, and clicking the box that opens below it (shown below)

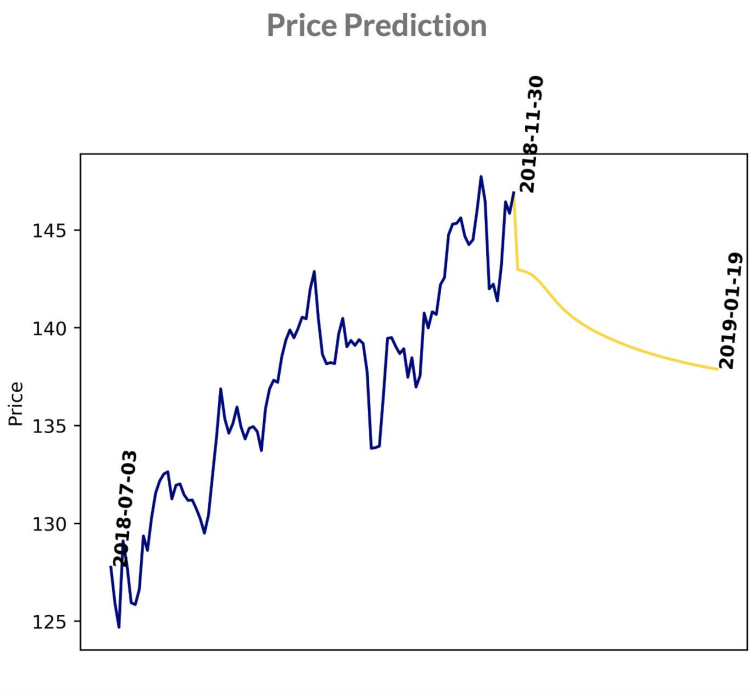


takes one to the individual stock's home page. This page can be accessed here:

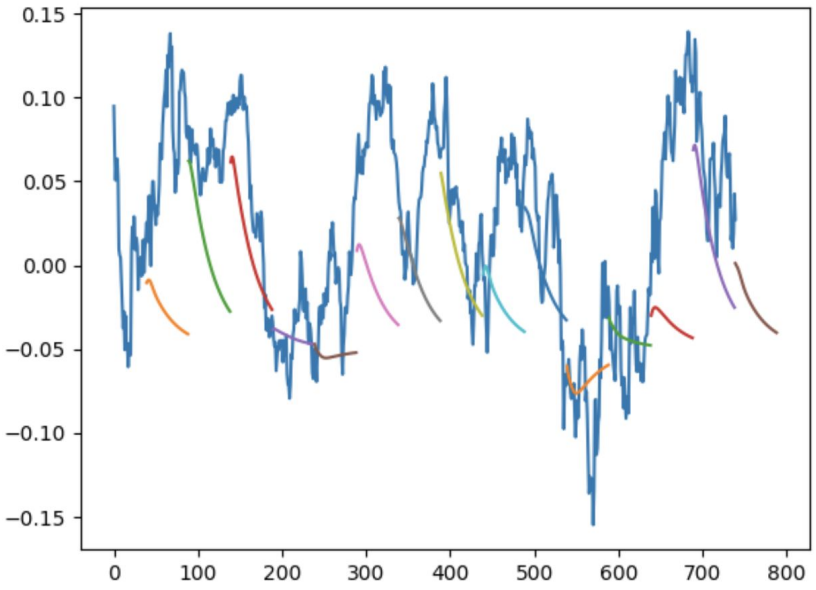
https://draco-web08.appspot.com/stocks/JNJ

On this page, one can zoom in-and-out by scrolling over the graph, hover over the green line to receive Simple Moving Average information, and hover over the red line to receive Exponential moving average information. The volume on each day appears as a bar graph below the stock price line graph.

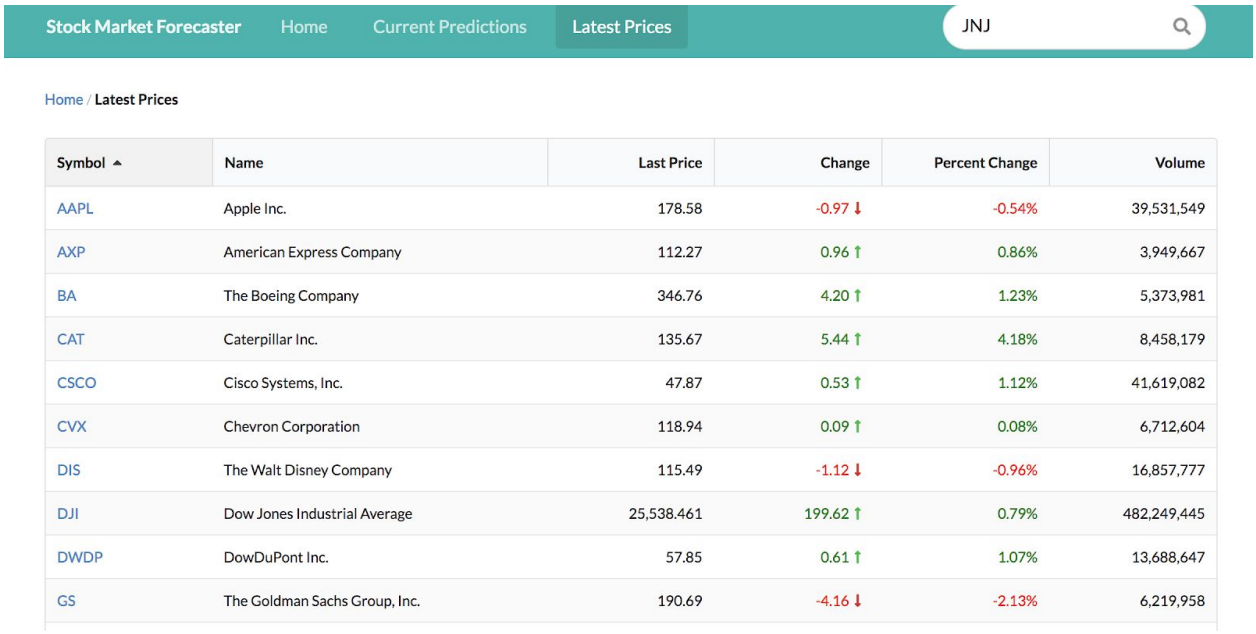
Beneath that users can see a pyplot generated graph with the predicted stock price 50 days out from the present day.



Scrolling further down on the same page, users can see PNGs of plots made on pyplot with our training data (around the last 700 days' data). The actual data (normalized between -1 and 1) is overlaid to view the accuracy of the testing portion of the results. Last, the prediction of the stock, which is demonstrated in the brown projection line below (right-hand side of the graph):



Next, one can click on the Latest Prices page, which loads the page below:



Symbol	Name	Last Price	Change	Percent Change	Volume
AAPL	Apple Inc.	178.58	-0.97 ↓	-0.54%	39,531,549
AXP	American Express Company	112.27	0.96 ↑	0.86%	3,949,667
BA	The Boeing Company	346.76	4.20 ↑	1.23%	5,373,981
CAT	Caterpillar Inc.	135.67	5.44 ↑	4.18%	8,458,179
CSCO	Cisco Systems, Inc.	47.87	0.53 ↑	1.12%	41,619,082
CVX	Chevron Corporation	118.94	0.09 ↑	0.08%	6,712,604
DIS	The Walt Disney Company	115.49	-1.12 ↓	-0.96%	16,857,777
DJI	Dow Jones Industrial Average	25,538.461	199.62 ↑	0.79%	482,249,445
DWDP	DowDuPont Inc.	57.85	0.61 ↑	1.07%	13,688,647
GS	The Goldman Sachs Group, Inc.	190.69	-4.16 ↓	-2.13%	6,219,958

On this page, the user can see the latest price, price change, percent change, and daily volume. The user can also click on any stock listed to be brought to the stock's home page. Thar page can be accessed at this URL: https://draco-web08.appspot.com/prices

Our displays were made using the following statistical calculations:

Simple Moving Average (SMA):

Calculating the Moving Average (MA) first involves choosing a specific time range for your Average. One may choose 10 days, for instance, and then calculating the SMA would be done by dividing the closing prices of the last 10 days and dividing by 10. One can create a continuous graph of the SMA by recalculating each day that involves adding the most recent closing price and dropping the 10th day. The file **ExponentialMovingAverage.py** demonstrates the calculation of this.

Bollinger Bands (BB):

There are three Bollinger Bands: the middle band is the 20 day SMA; the upper band is the 20 day SMA + (20 day standard deviation x 2); the lower band is the 20 day SMA - (20 day standard deviation x 2). The closing price of a stock hitting either BB indicates a trend occurring in the stock. The file **bands.py** demonstrates the calculation of this.

Exponential Moving Average (EMA):

The EMA is related to the MA but places greater emphasis on the more recent prices in a stock's history. A common number of days used in EMAs for short-term trends is 26 days. For long-term trends, often 200 day EMAs are used. A stock crossing the 200 day EMA indicates a reversal has occurred. To calculate the EMA: first, compute the SMA; second, compute the multiplier, which is [2/(time period length + 1)]. With those two values, one can compute the EMA: [Closing Price - EMA (previous day)] x multiplier + EMA (previous day). The file **ExponentialMovingAverage.py** demonstrates the calculation of this.

Recurrent Neural Networks (RNN):

RNNs are a network of connections between nodes. Stated simply, input data comes into a node (or simultaneous nodes) where it is processed into an output prediction. RNNs are used heavily for prediction in language modelling, as these nodes maintain a form of short-term memory, which is required for recognizing patterns when processing parts of a sentence or spoken phrases. If x is entered typically y is output. This pattern recognition is held in the nodes.

Long Short-Term Memory (LSTM):

LSTM is an extension of RNN. What RNNs lack is long-term memory. Often simpler RNNs are good for doing short sentence level language work. However when the network requires maintaining information about the topic of a passage at a paragraph or page level in order to better predict what might be coming next, LSTMs are better suited. A cell either maintains or discards information from the incoming data. This is decided by the “forget gate.” For instance, if the incoming information was the word “those” we would want to preserve the information that the next word might be a plural noun. After that, we may want to discard that information because a new subject could be being used. Then, some data may be stored long-term by integrating the output with the cell state.

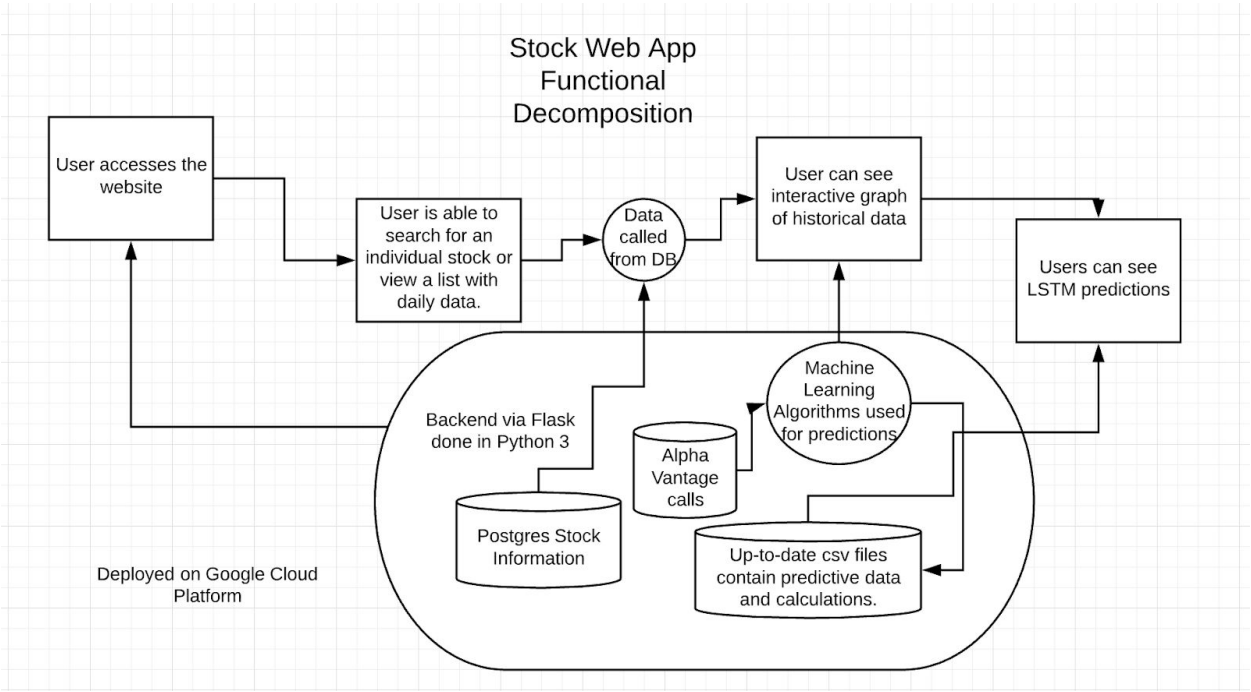
When applied to financial modeling, these same forget gates assist to retain marked trends over the long-term without disrupting—overwhelming—more immediate evaluative periods.

Training and Testing:

Machine Learning requires a training and testing batch of data. We chose to use 85% of the 20 year historical data for training and 15% for testing. We chose to test and predict in windows of 50 days. Both the closing price and the volume data were used for the creation of the LSTM model.

IV. Software and Systems

Here is an updated version of our functional composition from the planning phase:



As shown above, our Postgres database is hosted on Google Cloud Platform. From that database we draw data to display up to date information on stocks. Our machine learning algorithms were done with Python3 and are rerun nightly to ensure that the newest information is added to our LSTM model. That nightly-updated data, which includes pngs of the testing data predictions and prediction data, is stored in CSV files which is then displayed on the website using calls from React.

V. Libraries and Resources

A. Libraries

We used the **pandas** library for creating dataframes, reading information from csv files and to csv files and doing data analysis. We used the **numpy** library for doing calculations with our datasets and working with multidimensional arrays. The **matplotlib** library was used for creating plots of the data we computed. The **keras** library is a neural network library which we used to do our LSTM analysis. We used **SQLalchemy** for getting data from our Postgres database. We also used **Marshmallow** to convert data to and from Python datatypes.

On the front end we developed using a **React** framework. We included **React Stockcharts** for display of our data.

B. Languages

Python3 is used with a Flask framework for the backend. We used the React library for the front end of the site and, therefore, coded that in Javascript.

C. APIs

We used the AlphaVantage API to retrieve stock data for our site. Our code gets the latest daily data and reruns the code each night at midnight to provide the most up to data data and predictions.

D. Development Tools

We deployed the website on Google Cloud Platform. We used a Nginx server on a Google Cloud Instance to serve our PNG and CSV files generated by the machine learning code.

E. Resources

We got a lot of help from several blogs with and without GitHub repositories. We began with a lot of our LSTM code from here:

https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

This page also had a video and blog linked to it for explanation. We were able to change up that code to fit our needs. Changing the code to fit the needs of our project was hugely educational as it required following the work of each line of code and better understanding the keras library.

The following blogs were also extremely helpful during our project:

https://github.com/kartik-joshi/Stock-predection
https://www.datacamp.com/community/tutorials/lstm-python-stock-market
https://www.kaggle.com/raoulma/ny-stock-price-prediction-rnn-lstm-gru

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
http://www.asimovinstitute.org/blog/
https://github.com/joseamaita/webdev-python/tree/master/flask

The first portion of this course from Udacity covered the power of NumPy:
https://www.udacity.com/course/machine-learning-for-trading--ud501

Further, the following books represent a sample of the titles that contributed to our research:

Brownlee, Jason. *Introduction to Time Series Forecasting with Python*. 2018.

Brownlee, Jason. *Machine Learning Mastery with Python*. 2018.

Copperwaite, Matt & Charles Leifer. *Learning Flask Framework*. Birmingham, UK: Packt Publishing, 2015.

Dwyer, Gareth, Shalabh Aggarwal, & Jack Stouffer. *Flask: Building Web Services*. Birmingham, UK: Packt Publishing, 2017.

Duplain, Ron. *Instant Flask Development*. Birmingham, UK: Packt Publishing, 2013.

Grus, Joel. *Data Science from Scratch*. Sebastopol, CA: O'Reilly Media, 2015.

Hilpish, Yves. *Python for Finance*. 2nd Ed. Sebastopol, CA: O'Reilly Media, 2015.

Idris, Ivan. *Python Data Analysis Cookbook*. Birmingham, UK: Packt Publishing, 2016.

Keller, Benjamin Walter. *Mastering Matplotlib 2.x*. Birmingham, UK: Packt Publishing, 2018.

Kirk, Matthew. *Thoughtful Machine Learning with Pytho*n. Sebastopol, CA: O'Reilly Media, 2017.

McKinney, Wes. *Python for Data Analysis*. 2nd Ed. Sebastopol, CA: O'Reilly Media, 2013.

Rhodes, Brandon & John Goerzen. *Foundations of Python Network Programming*. New York: Apress, 2014.

Richert, Willi & Luis Pedro Coelho. *Building Machine Learning Systems with Python*. Birmingham, UK: Packt Publishing, 2013.

VI. Team Member Contributions

The Draco:WEB8 worked together to create this project. We had meetings and connected often over Google Hangouts to coordinate project work. We shared resources and code via Google Drive and Github. The contributions of each member are as follows:

Daniel – Daniel was the front end expert of the team. He was able to get React working to display all of the database data, csv data, and pngs. He was also able to create beautiful stock charts using React StockCharts. He designed the webpage as well. He did a great deal of configuration with Google Cloud Platform and was able to get Flask and React working together.

James – I worked on the following (all items include requisite troubleshooting, testing, and contributions from my two team members—*this was a team effort*): getting our primary (or initial) virtual instance running and accessible to the team; researching deployment options of Flask framework methods and, in part, front-end connective options; researching (mastery) as well as selection of machine-learning algorithmic options; utilizing NumPy and Pandas solutions to reshape stock-performance data for machine-learning input as well as front-end output (*i.e.*, an accessible deliverable); tailoring of Long short-term memory (LSTM) machine-learning framework to our team's selection of stock performance inputs; and generating our stock performance charts via Matplotlib.

Bridget – I worked on setting up the initial database on Google Cloud Platform. This meant populating the database with data and manipulating the downloaded csv files for the format we required for our site. This was done with basic python scripts. I also worked on the Machine Learning aspects of the project with James. We worked on learning, finding tutorials, modifying code, creating code, and making our machine learning code function with our data as well as returning workable data. I also created programs which took API data and output plots and data of Bollinger Bands, Simple Moving Average, and Exponential Moving Average. I set up the Nginx server. I also worked on the poster and final report.

VII. Closing

We completed over 300 hours of work on our project and are proud of what we were able to produce. Our final product exceeded our initial expectations as we were able to generate LSTM predictions, show basic statistical computations, and have it displayed on a user friendly website. We hope to continue to study Machine Learning techniques and the powerful Python libraries that enable them.