Stockbot: Stock Price Prediction with Historical and Sentiment data

Eric Chan
Paul G. Allen School of
Computer Science and Engineering
University of Washington
Seattle, WA 98195
yee96@cs.washington.edu

Kai-Wei Chang
Paul G. Allen School of
Computer Science and Engineering
University of Washington
Seattle, WA 98195
kwchang2@cs.washington.edu

Andrew Wei
Paul G. Allen School of
Computer Science and Engineering
University of Washington
Seattle, WA 98195
nowei@cs.washington.edu

1 Introduction

The financial market is a lucrative industry for many people. The main problem within this setting is to correctly price an item of stock in the market and exploit inefficiencies in the market in the form of a mispriced stock. Over the years, as the market becomes more efficient, the profit margins for each buy/sell order becomes ever slimmer. Therefore, more complicated and efficient methods have to be employed in order to stay competitive in this ever-changing market.

Historically, stock price prediction used past stock price data to extract patterns and trends that can be seen over time and within data that is generally too large for people to grasp by themselves, so they employ neural networks to learn and exploit these patterns to better model stock prices.

The inclusion of sentiment became a viable consideration with the advancement of Natural Language Processing (NLP) techniques and its increasing use of deep learning models. The idea is to use sentiment to augment our belief in the state of the market, since how people feel about the market or about some stock greatly affects its performance. Some examples would be if people believe that the market will crash, they may be reluctant to buy and be more willing to sell their current assets to avoid loss. Another example would be if news leaks or a scandal breaks out about a company or its leadership, then trust in the company may decline and their stock price may reflect that. Thus, it is reasonable to believe that public opinion (generalized as sentiment) of companies can be a good predictor or indicator of market trends.

In this paper, we revisit the idea of using sentiment from tweets as a method to capitalize on human reactions to stocks/companies to take a position in the market. In order to do so, we employ advancements in deep learning technology within natural language processing in order to first make predictions on the stock sentiment. Then we deploy a time-series method of predictions for stocks by using a recurrent model to extract patterns and make time-ordered predictions for stock prices.

We believe that with this aggregation of data, namely in the form of historical prices and news information, we will be able to create an effective model for stock prediction.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2 Related Work

There are papers that hypothesized a framework for a model that could provide real-time market information [2], papers that would combine both sentiment analysis and time-series modeling [6] or mention it as something to do in their future works [3, 5], but we could not find many papers that actually put such a model into practice or simulation.

Until around the 2000s, many people believed that any gains made from the stock market was entirely by chance and that it was impossible to model the stock market, much less determine good times to buy and sell. In 2002, A. Shakbar and I. Cloete used neural networks and historical price data to determine buy and sell points that offered better returns than just buying and holding [1]. This was one of the early events that sparked interest in the field of stock market modeling and stock price prediction using neural networks.

Stock market analysis can be seen as a streaming problem, taking in large amounts of data, either in the form of stock information or news/tweets, that is continuously being generated, and attempting to extract actionable knowledge from it. Ge et al. proposed a framework for dealing with this problem in a scalable fashion, using frameworks like Kafka, Spark, and Hadoop to ingest and store streaming data, and described how to integrate the data into a deep learning model [2]. They show their results in processing the Sentiment140 dataset for sentiment analysis. This is relevant because as these types of projects scale up and are implemented to work in real-time, there needs to be pipelines for streaming, ingesting, and processing the data in a way that can both keep up with the incoming data and produce reasonable results.

This type of project is not new, as there have been previous models that combine both sentiment analysis and historic price data to make predictions on stock price, many of which achieve decent returns. Das et al. leveraged the Twitter streaming API and processed the information with Spark and MLlib [3]. They train their system on 560k tweets from May 2005 to June 2017 and use a slightly finer-grained classification of sentiment than just good, neutral, or bad. The results they obtained generally followed the trends of the stock of the companies when overlaid.

The layering of machine learning techniques has been shown to be an effective technique for breaking down the complexities of problems. Sharma et al. does this on the Indian Stock Market by stacking sentiment analysis and a time series model and pushing it through a multivariate linear regression [6]. They collected Tweets using the Twitter API and scrape financial data from National Stock Exchange India. They perform sentiment analysis using Naive Bayes classification and SVM and use the count of positive, neutral, and negative tweets. Their time series decomposition uses a generalized additive model, which takes into account things like the general trend of the stock, seasonality, holidays, and error. They then took the sentiment data, forecasted price, and actual closing prices and put it through a multiple linear regression model.

Neural networks are often seen and used as black boxes, adapting to many types of situations and uses. Liu et al. used deep learning methods to predict stock prices by de-noising the data using a convolutional residual neural network [4]. Stock prices exhibit a lot of noise, so de-noising the data allows the network to fit to trends rather than noise. They de-noise the data using a sparse autoencoder made using a 1-D ResNet CNN, which reduces the dimensionality of the time series data. Then they pass the de-noised data to an LSTM, which is a recurrent layer that takes into account past information and outputs a prediction for the next step. They show that predicting the rate of change results in a more accurate model than just trying to predict the price. This gives us possible ideas for extensions we can make to our current ideas.

Including different types of data may create a more comprehensive model that better captures the intricate relationships that couldn't be expressed otherwise. Feng et al. attempt to model the relationships within industries to augment stock prediction [5]. They posit that relational connections between industries have an effect on the prices of stocks, like supplier-customer and sector-sharing relations. Specifically, they wanted to introduce a way of modeling stocks such that companies are not independent of each other. They do this with what they call a Temporal Graph Convolution, which adds a temporal element (changing edge weights over time) to Graph Neural Networks. They try to learn how one company affects another company in the graph and give a weight to the relationship, drawing ideas from PageRank. They used company information maintained by NASDAQ to group companies into similar industries and used Wikipedia to find relationships between companies.

3 Datasets

3.1 Financial Dataset

Our financial dataset provides us with all the opening price, closing price, high, low and volume amounts at a granularity of days for all the companies listed in Table 1 for the past 10 years. We were able to retrieve this dataset from Polygon. We implemented an interfacing script to retrieve this data and save it locally. In addition to that, we also implemented features to get live data in the event that we wanted to test our model in real time.

However, even though we were able to get the data from Polygon, we had to preprocess them a bit more, as there are some inaccuracies in the data. For example, when Apple Inc (AAPL) performed a stock split on June 9, 2014. The prices provided by Polygon did not reflect those changes, so we had to perform manual corrections to fix this error.

3.2 Sentiment Dataset

We have two datasets for sentiment analysis: one for a training sentiment analysis model and another one for general company/financial sentiment to predict stock price.

3.2.1 Training Dataset

Currently, we are training on Sentiment140, which has 1.6m tweets and is one of the largest, publicly available labeled Twitter sentiment datasets. The tweets are general, ranging from reactions to statuses.

When we predict the sentiment for companies, however, we gathered our own financial tweets that are more related to the company's current status. The reason that the model cannot train on financial tweets is because we cannot find a dataset that contains only financial tweets with sentiments labeled.

3.2.2 Company/Financial Tweets

To obtain the tweets related to general or financial company sentiment, we built a web scraper in Python using selenium, Google Chrome WebDriver, and BeautifulSoup. The scraper scraped tweets that contained a tag related to the company every day from the beginning of 2010 to the end of 2019. We accomplished this over the span of 6 weeks while respecting the crawling policies in Twitter's robots.txt.[1] We originally wanted to use the Twitter API, but they have a limit of requesting 25,000 tweets a month, while we have scraped a total of 2,479,869 tweets with hashtags[2] and 1,672,590 tweets with cashtags[3].

There were days that contained no tweets, but for the days that contained tweets, we obtained a subset of the "top" tweets according to Twitter's ranking algorithm. Twitter says its ranking algorithm ranks by relevance to the search, the popularity of the tweet, and other factors.[4] For the days we have tweets for, there was a large variance in the number of available tweets, which intuitively makes sense because tweets wouldn't be produced on a consistent basis.

We originally wanted to scrape tweets associated with hashtags of stock tickers, but learned that when searching for these tags, we were getting results that were not related to the company. For general tweets, we looked up a hashtag (#) of the stock ticker or the company name, depending on the relevance of tweets we found, which we described in Table 1.

For financial tweets, we looked up the cash tag ($) of the stock ticker. Cashtags were first supported by Twitter in 2012, but was used on the site before then.[5] A cash tag is like a hashtag, but specifically for tracking stock symbols. This means that the type of tweets we would obtain through these cashtags would be more relevant to a company's financial state. This type of information may be more relevant to us in terms of being able to predict fluctuations in stock pricing, as it is more closely related to its

[1] https://twitter.com/robots.txt

[2] These can be found here: https://github.com/nowei/twitter_tweets_by_tag/tree/master/twitter_data

[3] These can be found here: https://github.com/nowei/twitter_tweets_by_tag/tree/master/twitter_cashtag_data

[4] https://help.twitter.com/en/using-twitter/top-search-results-faqs

[5] https://twitter.com/Twitter/status/230098997010911233

Companies List with Hashtag and Cashtag			
Company Name	Hashtag	Cashtag	Details
Apple	#AAPL	$AAPL	
Bank of America	#BankOfAmerica	$BAC	#BAC gives results like the Baltimore Celtics and other unrelated news
Chipotle Mexican Grill	#Chipotle	$CMG	#CMG gives results like Checkmate Gaming
Coca Cola	#CocaCola	$KO	#KO gives tweets about boxing or fighting
Delta Airlines	#Delta	$DAL	#DAL refers to Dallas and an Indian food dish
Facebook	#Facebook	$FB	People who post stuff on Facebook and also Twitter seem to like posting with #FB
Google	#Google	$GOOG	The stock ticker is #GOOGL, since #GOOG now refers to Alphabet Inc., its parent company, but they split in 2015 and we needed financial data since 2010
JPMorgan Chase	#JPMorgan	$JPM	#JPM refers to John Magufuli, the current president of Tanzania
McDonald's	#McDonald	$MCD	Apostrophes break up hashtags and there were many tweets related to McDonalds using #MCD, but quite a few that were unrelated
Microsoft	#MSFT	$MSFT	
Pepsi	#Pepsi	$PEP	#PEP gives results about a soccer player named Pep Guardiola
Southwest Airlines	#SouthwestAirlines	$LUV	#LUV refers to American Rapper, Lil Uzi Vert, and is a common alternative for "Love"
United Airlines	#UAL	$UAL	
Visa	#Visa	$V	#V refers to a Korean pop star in the band BTS.
Wells Fargo & Co	#WellsFargo	$WFC	#WFC refers to Wake FC, an American soccer club

Table 1: Company name and searched hashtag and cashtag. The Cashtags looked up are the tags associated with the company's stock ticker. Hashtags were changed from the stock ticker if results resulted in tweets that were generally unrelated to the company. Note that searched hashtags are case-insensitive when searching on Twitter.

stock. When classifying the sentiment, we averaged the predicted sentiment of the tweets for a given day and gave a neutral sentiment for days without tweets.

We also looked into scraping news data from the New York Times and Yahoo Finance. For the New York Times, we felt like the articles were being released on an inconsistent schedule and we would not be able to get a consistent stream of data. For Yahoo Finance, we saw that there were many video articles intermixed to articles and we couldn't find transcriptions for them. On top of that, their website was relatively clunky and there didn't seem to be a filter by date option.

4 Models/Algorithms/Methods

4.1 Financial Model

Our price prediction model is a LSTM model that consist of 2 layers. We set the input dimension to be 5, which consist of the open, high, low, close and volume of a ticker. The hidden dimension of the model is 32 which is a hyperparameter we set. We perform a lookback of 30 days, meaning that we will pass in a sequence of 30 days worth of data to predict for the 31st day of data, and each day of data consists of the 5 features previously described. More formally, to predict day $t + 1$, we will train it from day 1 to day t.

We used two methods for training the financial model and we train a separate financial model for each company. One method was using an iterative training scheme and another method was without this scheme. For both methods, we start by performing a regular 80/20 split where 80 percent of the data is used for training and 20 percent of the data is used for testing.

We define iterative training as continually updating the LSTM model by moving actual closing prices into the training points after making the prediction for closing price for that day, and then running 2 epochs to update the model distribution. There may be some concern about the practicality of such a method, but we have seen that only a few epochs are needed to update the LSTM model to make decent predictions. This means that this technique is practical for real-world usage.

For both the iterative and non-iterative training, we first perform 400 epoch of training over the first 80 percent of the data. Then, if the data is non-iteratively trained, we would simply make predictions for the remaining 20 percent of data. Similarly, if the model is to be trained iteratively, we would then perform the iterative training method and move the test data into the training set a day at a time.

4.2 Sentiment Model

Before passing our data to the model for training, we had to do some data processing to clean up the dataset. We decided that we should 1.) strip out all the different white-space symbols, 2.) remove all the urls, mentions, and hashtags because we wanted to capture the sentiment of the text, 3.) separate all punctuation from the word, and 4.) turn every character into lower case. This means that the sentence "This is `\t` an Example. @Apple" would be turned into ["this", "is", "an", "example", "."]. In addition, in order to better deal with unseen words in future, we UNK words of frequency 1 with 60% probability.

Our sentiment analysis model is based on a two-layer, bi-directional GRU that receives a sequence of data as input and produces a single number as output to indicate the sentiment score. Data first passes through an embedding layer that encodes the integer representation of the text into a higher dimensional space. Then, they are fed through the GRU before being mapped to a linear layer that reduces the dimension down to a single integer. Lastly, this integer (ranging from negative infinity to positive infinity) is passed to a sigmoid function. The final output would be in a range of 0 to 1 where 0 indicates strong negative sentiment while 1 indicates strong positive sentiment.

To assist with our training, we used batched gradient descent with the Adam optimizer. We ran multiple trials to find the optimal hyperparameters. First, we understand that as we increase the embedding and hidden sizes, the model is able to learn more information. We tried to increase both these hyperparameters to as large as possible. This, of course, leads to overfitting. Then, we reduce the embedding and hidden sizes while increasing the weight decay parameter for the Adam optimizer until it no longer overfits. We then train over 20 epochs or until the validation accuracy no longer increases.

4.2.1 First Try - Pure Sentiment Analysis

First, we trained on the Kaggle Sentiment140 dataset. This dataset features 1.6 million tweets, each labeled as either negative, neutral, or positive. We trained on this dataset for 20 epochs.

4.2.2 Second Try - Price-change Based

We thought about a second approach that we could attempt. Tweet sentiments and stock price may have some correlations, so we thought about training the model based on tweets and price changes

instead of the actual sentiment. For this part, we trained on a portion of the tweets we scraped and validated on the rest for a total of 1.2 million tweets.

For each tweet, we label it as the price change of the closing price of that day compared to the closing price of the previous day. Since we would be feeding the sentiment score of tweets to the price-prediction model, this makes sense as we are leveraging a short-term price change to calculate a score for a collection of tweets which would then be fed into the price-prediction model that uses historic prices to predict the price of the next day.

5 Results

Figure 1: Coca Cola Stocks without Iterative Training

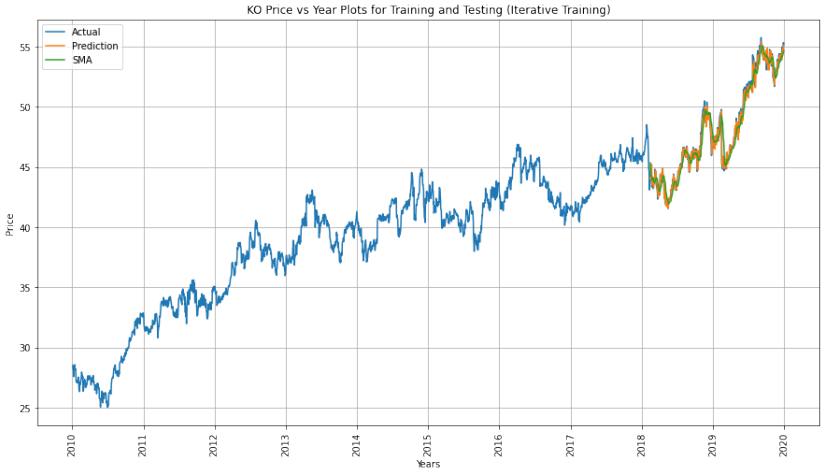


Figure 2: Coca Cola Stocks with Iterative Training

5.1 Financial Model

Using the stock of Coca Cola (KO) as an example in the figures above, we present the results of two models. One is trained with iterative training and the other without. For the plot without iterative training, notice that the predictions generated from the model trails off the true value as time progresses. This is the opposite of the Simple Moving Average (SMA) baseline, which still keeps track of the price changes. Compare this with the plots with iterative training below. Note that the prediction follows the actual price closely, indicating that it is able to update its predictive distribution as time progresses.

Stocks Symbols	SMA (10 days)	RMSE (w/o Iterative)	RMSE (w/ Iterative)
AAPL	3.80	11.36	4.60
BAC	0.63	0.52	0.47
CMG	17.70	13.43	11.76
DAL	1.34	0.96	0.87
FB	4.20	3.49	3.45
GOOGL	24.41	18.64	18.29
JPM	1.88	2.07	1.68
KO	0.69	0.96	0.52
LUV	1.29	1.02	0.91
MCD	2.22	7.45	2.37
MSFT	1.52	16.46	2.28
PEP	1.54	2.31	1.36
UAL	2.24	1.99	1.41
V	1.78	13.80	2.73
WFC	1.14	0.70	0.69

Table 2: Root mean square error of the prediction models with and without iterative training compared to the Simple Moving Average baseline.

Stocks Symbols	Trend Prediction Accuracy (%)
AAPL	51.94
BAC	49.75
CMG	49.05
DAL	48.16
FB	47.46
GOOGL	52.61
JPM	51.54
KO	50.75
LUV	49.55
MCD	50.05
MSFT	52.34
PEP	47.86
UAL	49.85
V	49.75
WFC	49.65

Table 3: Trend prediction results of iterative training. This is defined as how accurate our trend predictions are, i.e. when the price went up or down, how often did we predict it increasing or decreasing respectively.

We see that the model doesn't predict trends accurately, but still achieves a lower error than the Simple Moving Average over the last 10 days. We believe that while the model does not predict trends very accurately, it may be predicting prices in a way that if it is wrong, it is not very wrong.

5.2 Sentiment Analysis

To calculate the accuracy of the model, we decide to classify the outputs and labels < 0.5 to be negative and > 0.5 to be positive. We then compare the percentages of outputs that match the labels. For the model we trained using our first approach (pure sentiment analysis), we achieved 88% test accuracy on a portion on the same untrained dataset. However, when we feed the financial tweets we scraped, we found that a lot of sentiments scores don't match the tweets' actual sentiment. We assumed that since the Kaggle dataset contains very general tweets and not financial tweets, it would perform poorly when trying to predict the sentiment for company-specific tweets (more discussion on this in section 6.2).

We used the same method to calculate the accuracy for our second approach, the price-change-based model. Unfortunately, we faced difficulties trying to improve the validation accuracy. After 10 epochs, the train accuracy went from 51.7% initially to 68%, but the validation accuracy stayed

around 52%. At first, we suspected that it overfits because the test loss and accuracy decreases after epoch 1. However, after adding more weight decay and reducing hidden size, training accuracy barely improves over time. We are unsure of how to improve the model performance and are also suspecting that this approach might not be feasible given that tweets may have very little, if any, correlations with stock price. This can explain why the model was not able to train on the data using this approach.

6 Discussion

6.1 Financial Model

Our price prediction model is a vanilla LSTM model because we want something more explainable to make predictions with. This also explains our reasoning to only have 2 layers for the models. By keeping it simple, we were able to have a better understanding of the model's results. In this case, it it clear that through iterative training, we are able to see the improvements over regular training.

6.1.1 Iterative Training

On a high level, iterative training allows the model to learn from the mistakes it makes in the past so that it can perform better in the future.

People generally use a training and testing split to try to see how well their model generalizes to data it hasn't seen before. While it is normally heretical in the machine learning community to train on test data, we believe that there are certain situations in which it makes sense to include that data in some manner.

One reason for why we might want to do this is that we can't obtain new data about something that has happened in the past. This means that the total amount of publicly available data is bounded by when the company made its initial public offering to when they go out of business. Then company stock prices depends on the many factors, so the price will likely shift as the company grows or experiences important events. Thus, the amount of data increases and the distribution of stock prices changes over time.

In our application, we want to perform a time series prediction given previous data, so we don't care much about how it performs in (generalizes to) a different setting, we only care about how well it performs on predicting the next day. Thus, it makes more sense to include the data for a day after we have made our predictions for that day and retrain, allowing us to make better predictions in the future, as we then have access to more locally relevant data. In doing so, we include more information that gives our model better ideas about local trends. Then, over time, this allows the model to potentially build a better global view of overall trends.

The crucial idea is that we want the model to be constantly updated, as we hypothesize that the distribution of the stock changes over time. In fact, we have seen in our experiments that 1-2 epochs are sufficient to perform the update on the models.

6.2 Sentiment Analysis

There may have been problems with our sentiment model generalizing, as we were training the model using general tweets, but we wanted tweets dealing with the sentiment people held towards companies. Thus, there is a gap between the data we are training our model with and the data we want to use the model to give predictions for. Then when we used our model to try to predict the sentiment of companies on a given day, we would exhibit some sort of generalization error, making it hard to rely on the outputs. This also holds for when we were using cashtags instead. As in that situation, we would've preferred a model that was trained to classify the sentiment of financial tweets, as the language used in the domain of finance may differ from those used in general day-to-day settings.

Trying to train the model using price change instead of sentiment labels makes sense if a majority of tweet sentiment correlated with price changes for a given day. One major drawback with how we approached this was that we were not sure which period of time the tweets were talking about. Were the tweets about the past, present or future? This makes it hard to associate correlations between sentiment and price changes to tweets, especially as a method for labeling tweets.

We were surprised that the validation accuracy for this method was constantly around 52%, even though training accuracy slowly increased. Overfitting is possible, but not likely, when the accuracy is only 52%. We also tried to treat this as if we were dealing with overfitting by reducing hidden size and adding weight decay, but our test loss failed to decrease after we made these changes. Since this was the same model as the one we used for our first approach, we suspect that tweets may actually have very limited correlations with the stock price.

6.3 Twitter data

Upon inspecting the tweets, we found that there were related posts that did not contain the hashtag, but contained mentions of the searched tag. We did not filter these tweets out, as we felt like they were still relevant for capturing the general sentiment people had of the company.

We could've used company names in the general hashtags to avoid collisions with mostly unrelated topics if we wanted to get general sentiment. We believed that while general hashtags could capture the general sentiment surrounding a company, these tweets may also be more polarized. Specifically, we believe that people who post their reactions online generally tend to be those who felt stronger reactions. This means that for tweets that aren't simply about company news, there may be more strongly positive or strongly negative posts related to the company's behavior or products rather than how it is performing financially.

We wanted to look up the company stock's ticker as the tag instead of its name to hopefully avoid some of these issues, but many of the company tickers also collided with other popular tags. Tweets directly from companies may also contain some sort of bias, as they generally tend to focus on what they're doing well rather than shining a light on their problems. While we believe that general sentiment is a possible indicator of company stability, it may not the best reflection of financial status.

Lastly, a potential issue is that tweets may not contain important information. Specifically, Twitter is open to anyone with internet and an email address, so anyone can post tweets to Twitter, with any hashtag or cashtag they want. Thus, we may get tweets from financial professionals as well as hobbyists, bots, amateurs, or even people who accidentally tag things. This makes it more difficult to use these tweets, as we currently weigh all these tweets equally, without user information to maintain anonymity.

7 Future Work

7.1 Financial model

Notice that in Table 3, the trend predictions made by our LSTM model are roughly 50 percent accurate. In order to improve the performance, we propose to add additional features that encompasses the trend prediction of the model for the next day as an input to train the model for better performance. If the model takes in the predicted trend for the next day to make price prediction, we believe that it can make better predictions overall.

7.2 Sentiment model

In this project, we used a very basic GRU model for our sentiment model. Though we did try to apply weight decay to our optimizer to avoid overfitting, we think that there are more better ways of improving the overall model. For instance, we can apply random dropout on our embedding layer so portions of training data have a random chance of being set to $\vec{0}$. This can assist our model in handling unseen words and utilize more of the context to deduce general sentiment.

In addition, we could try out using word2vec instead of building our own labeling from the ground up. word2vec already maps out the word-to-word relationship for many vocabularies, so even if we haven't seen the words previously in our training set, word2vec can capture the meaning of our words and allow the model to understand the approximate meaning to better produce the sentiment output.

7.3 Company sentiment data

7.3.1 Tweets

In the future, it may be interesting to train a sentiment model using a labeled sentiment dataset with an emphasis on financial tweets, as the training data for the sentiment model we used in these experiments did not generalize to the domain. We expect this to perform better with respect to how our current models are set up.

It may also be helpful to rank the tweets in a way that reflects how correct the tweet was. If we tracked users who did this, we could give them a rank and weigh future tweets proportional to how accurate their previous tweets were. We can also include data on how popular the tweet was based on likes or shares, as it makes sense that more shared or liked tweets may have a greater reach/impact.

Lastly, it may also make sense to get specific tweets from "experts" such that we can get potentially more relevant tweets, as they possess domain knowledge that could be useful in predicting stock trends. Although it may be hard, as people generally tend to be more selective about who they share this type of information with, as doing so may make or lose people a lot of money.

7.3.2 News

We may also try this again with financial news, as more people probably read the news than try to look up somewhat esoteric tweets about companies from pretty much anyone. The reasoning for this is that news generally has a wider reach, especially if the provider is trustworthy. The people who write these articles probably also have a history with finance, so they may also possess some sort of domain knowledge that may be useful for predicting trends and prices. Thus, the sentiment collected from financial news may be more effective at predicting stock prices than tweets.

8 Conclusion

In this project, we attempted to model the stock market using price and sentiment data. Our main contributions in this paper include:

- an iteratively trained stock prediction model
- an alternative to training directly on sentiment
- ~4m scraped tweets with specific company hashtags or cashtags

We believe we have made great strides in our first foray into the rich field of stock market modeling and dealing with problems related to utilizing sentiment analysis in a real application.

From our financial model, we successfully demonstrated that a model with iterative training would be better than one without. This is because the model can continuously learn from new data, thus being able to produce better predictions as we get further away from the initial training set. We believe that we still have a long way to go in terms of an accurate price prediction model, but we believe that these steps will be crucial to our understanding of time series stock price prediction using LSTMs going forward.

For our sentiment analysis model, we learned that a poor training dataset can be a huge stumbling block to successfully training a model. While we were successful in producing a good pure-sentiment analysis model that provides good sentiment accuracy for general tweets, it performed poorly with finance-specific tweets. We believe we would've achieved better results had we trained a model based purely on financial tweets/discussions.

Our effort of addressing this issue, unfortunately, did not work out as expected. The idea to train an NLP model based on price change is an innovative approach that should work theoretically. Even though our training was not successful, we discussed numerous improvements that we can make if time allows. We expect this idea to help those who try to use an NLP model to predict stock price.

9 Contributions

We fail to outline individual contributions, and we are fine with getting the same grade.

References

[1] Andrew Skabar and Ian Cloete. 2002. Neural networks, financial trading and the efficient markets hypothesis. Aust. Comput. Sci. Commun. 24, 1 (January 2002), 241–249.

[2] Ge, Shihao et al. “A Scalable Framework for Multilevel Streaming Data Analytics Using Deep Learning.” 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC) (2019): n. pag. Crossref. Web.

[3] Das, Sushree et. al. "Real-Time Sentiment Analysis of Twitter Streaming data for Stock Prediction" International Conference on Computational Intelligence and Data Science (ICCIDS 2018): n. pag. Crossref. Web.

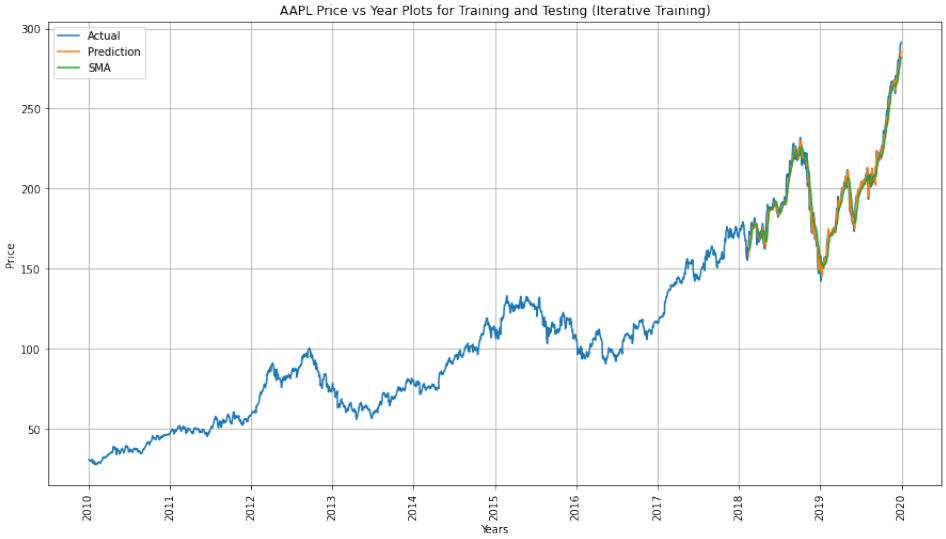
[4] Liu, Jialin, et al. "Stock Prices Prediction using Deep Learning Models." arXiv preprint arXiv:1909.12227 (2019).

[5] Feng, Fuli et al. “Temporal Relational Ranking for Stock Prediction.” ACM Transactions on Information Systems 37.2 (2019): 1–30. Crossref. Web.

[6] V. Sharma, R. Khemnar, R. Kumari and B. R. Mohan, "Time Series with Sentiment Analysis for Stock Price Prediction," 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), Jaipur, India, 2019, pp. 178-181.

[7] Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid, “Practical deep reinforcement learning approach for stock trading,” 2018.

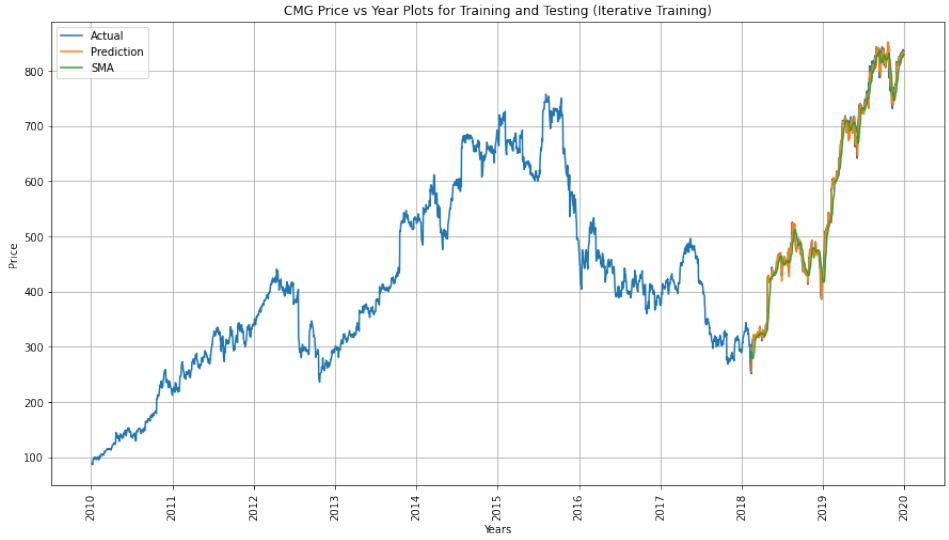
Appendix: Graphs



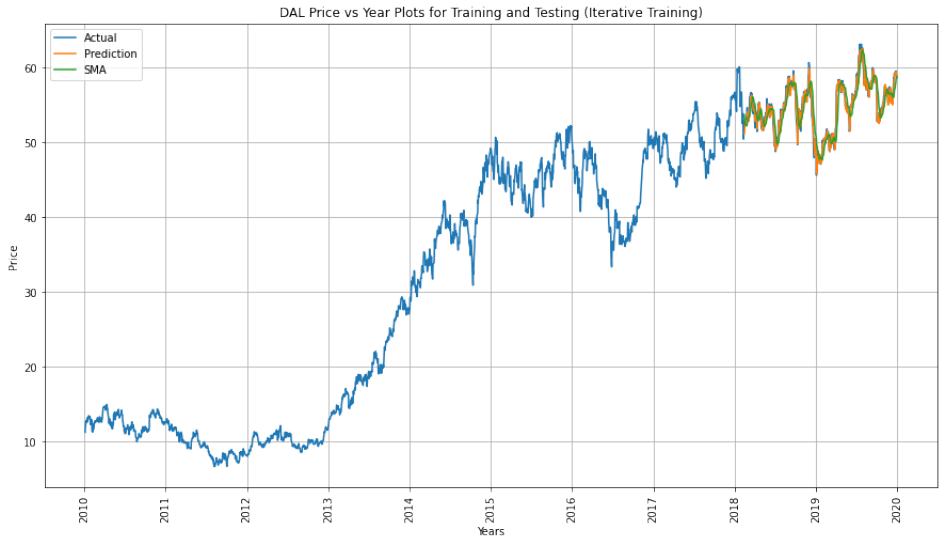

BAC Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years
35
30
25
20
15
10
5
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020


BAC Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years
35
30
25
20
15
10
5
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

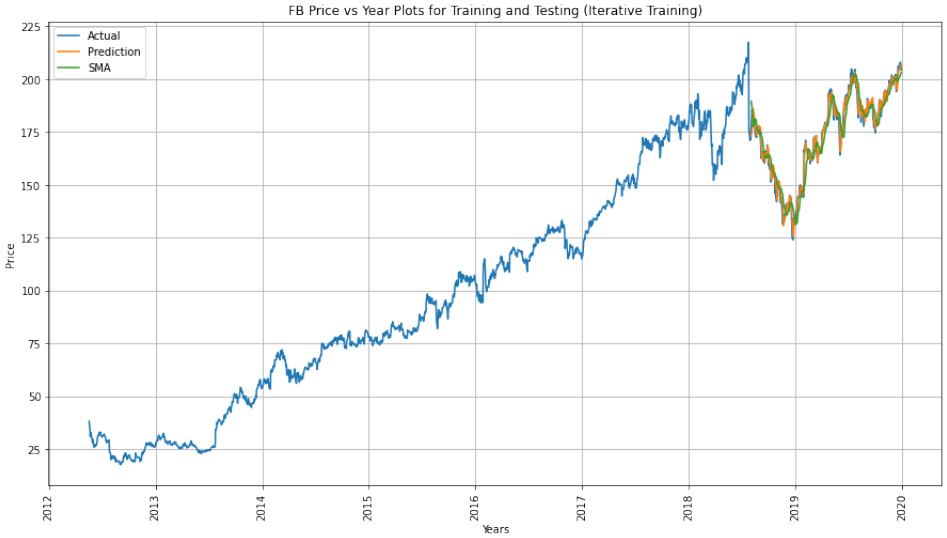

CMG Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years
800
700
600
500
400
300
200
100
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020


CMG Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years


DAL Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years


DAL Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years

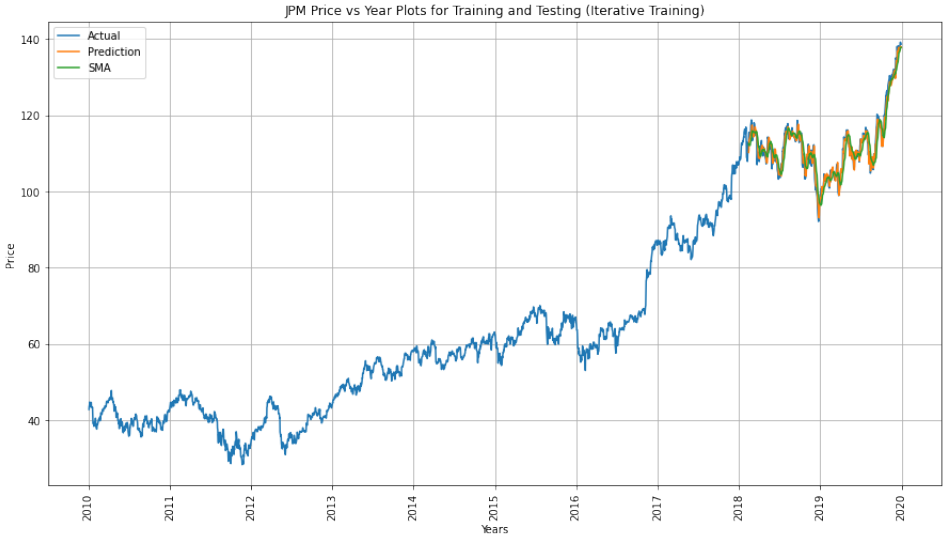

FB Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years
225
200
175
150
125
100
75
50
25
2012
2013
2014
2015
2016
2017
2018
2019
2020


FB Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years
225
200
175
150
125
100
75
50
25
2012
2013
2014
2015
2016
2017
2018
2019
2020

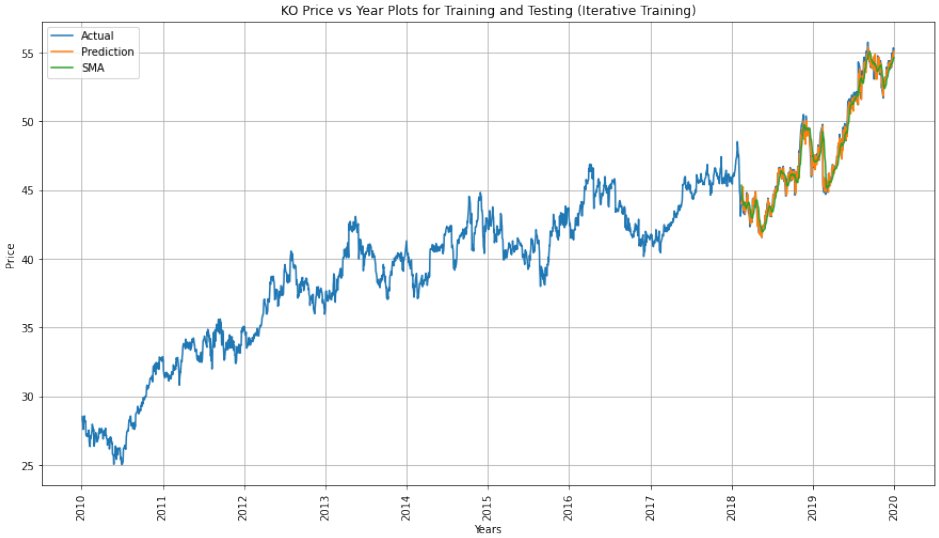

GOOGL Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years
1400
1200
1000
800
600
2014
2015
2016
2017
2018
2019
2020


GOOGL Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years

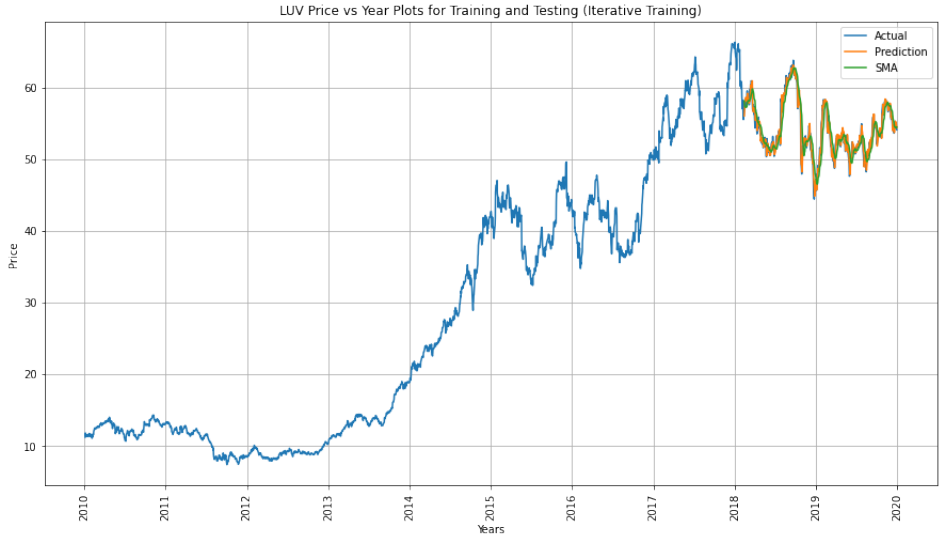

JPM Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years

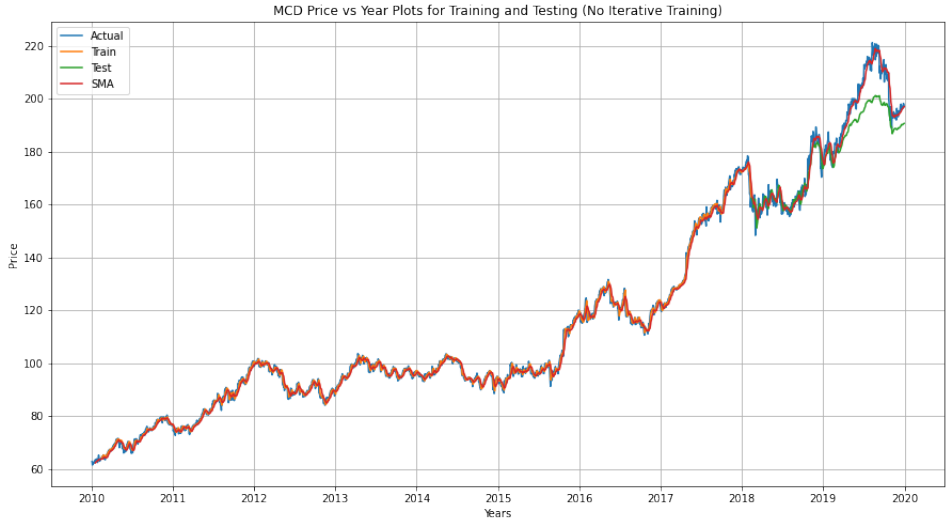

JPM Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years

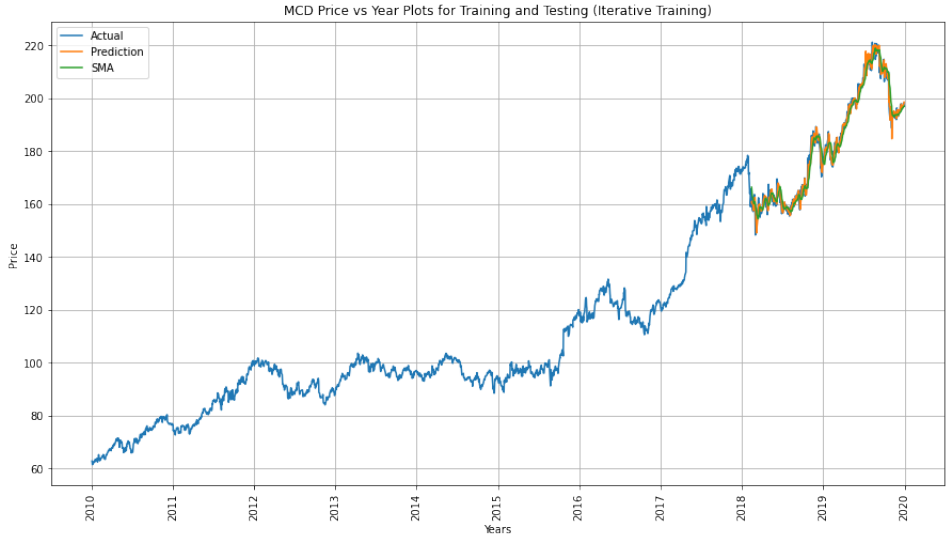

KO Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years

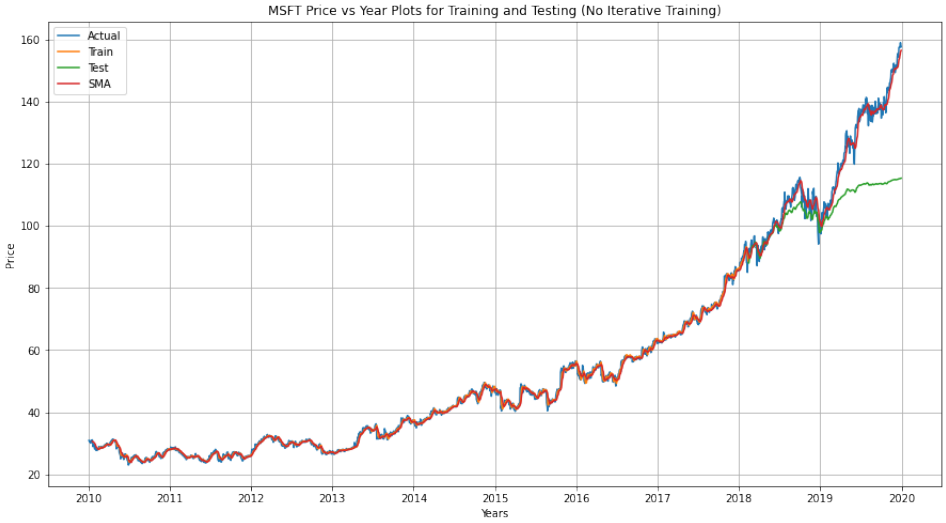

KO Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years

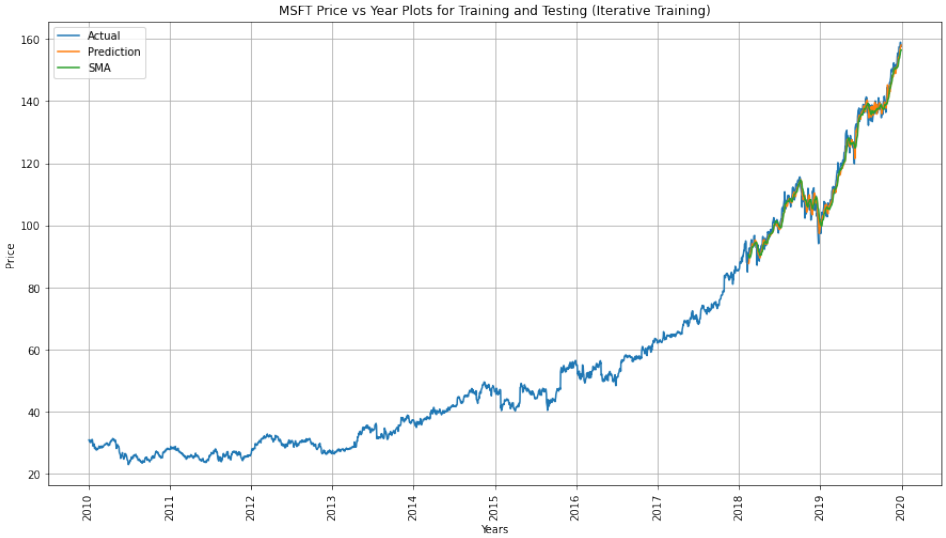

LUV Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years


LUV Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years


MCD Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years


MCD Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years

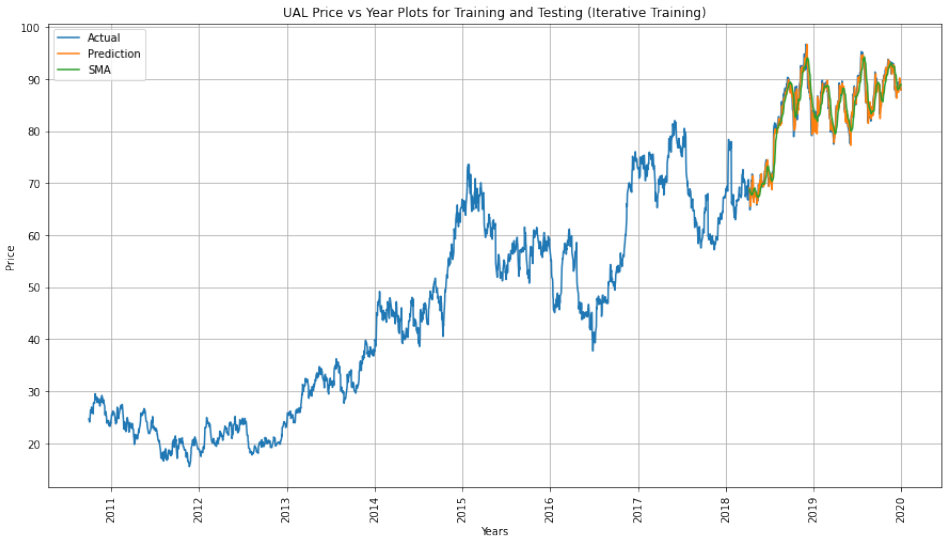

MSFT Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years

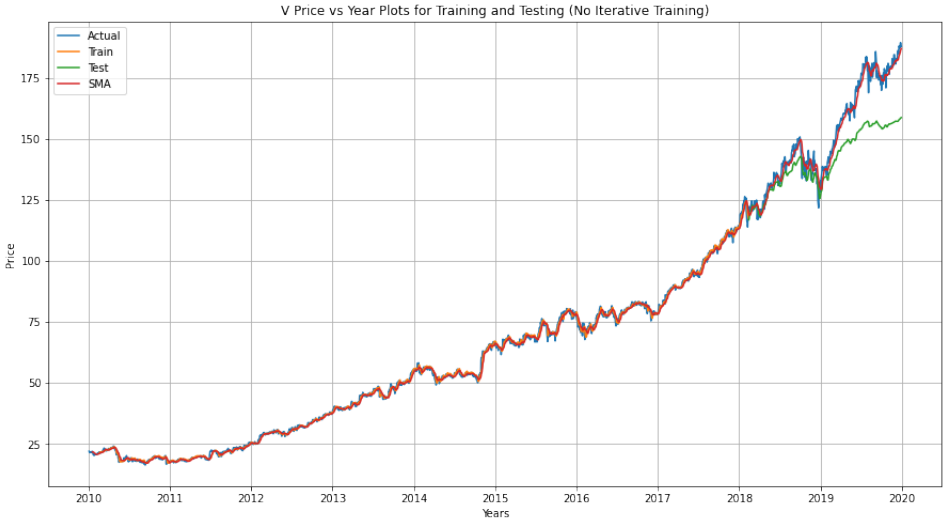

MSFT Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years

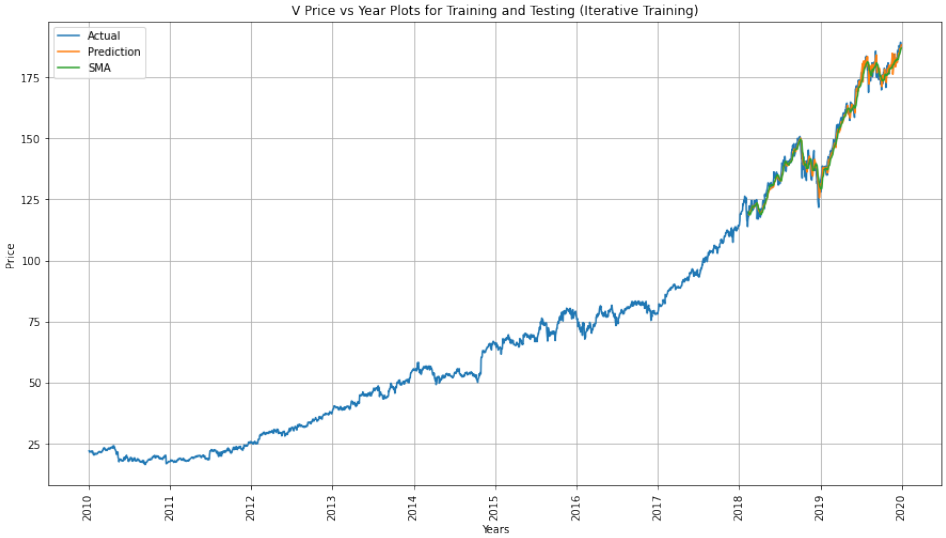

PEP Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years


PEP Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years

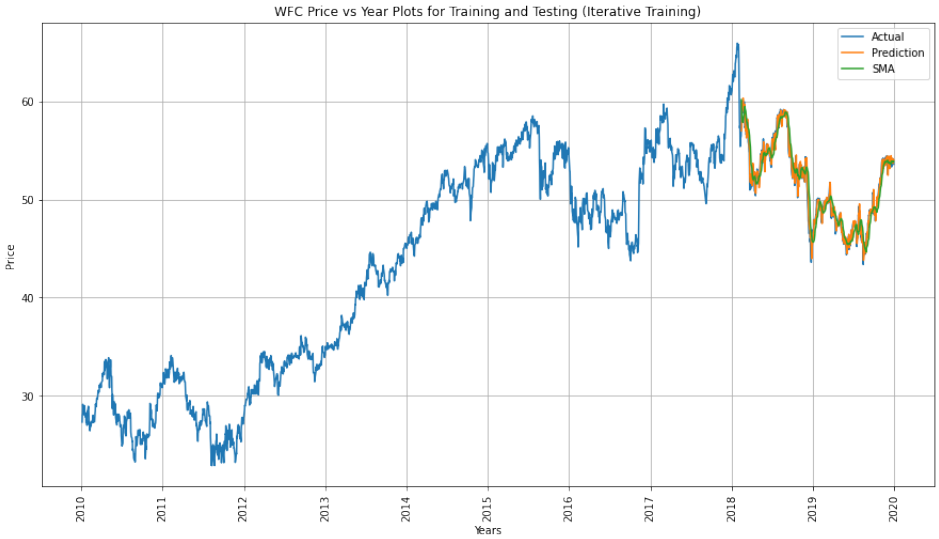

UAL Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years


UAL Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years


V Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years


V Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
Years


WFC Price vs Year Plots for Training and Testing (No Iterative Training)
Actual
Train
Test
SMA
Price
Years


WFC Price vs Year Plots for Training and Testing (Iterative Training)
Actual
Prediction
SMA
Price
60
50
40
30
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
Years