Spark Near Real-Time Sentiment Analysis

*Mohan K. Patnam, CN Chen, Bruno Janota (CSCI E-63)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Motivation:** *This project examines whether social media feeds focused towards stock market and concerning companies, are a suitable data source for forecasting the direction and volatility of the underlying stock prices. In particular, we focus on StockTwits.com which is a popular social media platform where real investors, traders and general public alike express their opinion on companies and their performance in the form of tweets. We begin this project by providing a generic approach to download historical tweets for the chosen stocks over a chosen period. We then introduce a couple of models to calculate social sentiment scores using two variations of Bag-of-Words (dictionary) and a Naive-Bayes classifier. We apply basic text processing techniques using NLTK to preprocess tweets using the NLTK lemmatizer and stopwords. We further perform regression analysis to see if social sentiment can serve as a leading indicator to predict changes in underlying stock prices. We demonstrate an ability to sense the market pulse or sentiment using a near real-time stream of tweets. Our aim here is illustrate the use of modern machine-learning and real-time parallel processing technologies while answering a fundamental question of whether social sentiment matters for stock market performance.*

**Data set**:   
1. Two months worth of tweets for three chosen stocks (**TSLA, FB, AAPL**) from [www.StockTwits.com](http://www.stocktwits.com)

2. Two months of historical stock prices from <http://finance.yahoo.com> for the same set of stocks.

3. Intraday stock prices from Bloomberg Finance for the same set of stocks.

**Technology**

Python and various API - StockTwits RESTful, NLTK, Spark Streaming, Tensorflow API to implement various phases of the project.

**Benefits**:

* Ease of use - python and its seamless integration with web downloads, text analysis, spark streaming and machine learning packages.
* Ease of web downloads using RESTful API.
* Ease of natural language processing using NLTK lemmatizer and stop-words packages.
* Parallel processing using spark streaming API by simultaneously running sentiment analysis on multiple stocks.

**Challenges**:

* StockTwits RESTful API has a limit of 200 requests per hour. Each request returns 30 tweets for a chosen stock in JSON format.
* Signal-to-noise ratio in the user tweets. Tweets are often highly unstructured in an informal language. A significant portion of noise can be filtered using appropriate dictionary of words and better processing of text (stop-words, n-grams, lemmatization). Pre-processing has implication on the accuracy of the sentiment prediction models.

**Demo Workflow:** We implemented various phases of sentiment analysis as a work-flow.

* Download tweets for a stock (TSLA, FB, AAPL) using StockTwits RESTful api and json processing. Output is a csv file with each record representing one tweet having following fields -   
  *TweetID, Stock, Date, CreateTime, Text, Sentiment*
* Preprocess tweets csv file. Remove stop-words first. Then perform lemmatization to reduce the dimensionality for different usages of the same word (organize, organizes, organizing).
* Perform Bag-of-Words analysis to calculate sentiment score using two dictionaries. Loughran McDonald (Financial) dictionary and the Harvard dictionary. Present confusion matrix for model accuracy using calculated vs presented sentiments in each case.
* Perform Naive-Bayes analysis to predict sentiment score by training the model first and verify using the test data. Present confusion matrix for model accuracy.
* Plot sentiment score using spark streaming of tweets in real-time.
* Perform Regression analysis (linear, cubic) using Tensorflow api to see if changes in sentiment score can predict changes in underlying stock prices.

# Introduction

This project is divided into four phases. The first phase of the project focuses on data download and wrangling into a suitable format for further processing. We collected historical tweets for chosen stocks for a period of two months from [www.StockTwits.com](http://www.stocktwits.com) using their restful API. As per our research on Investopedia.com, there are few stocks that are very popular on stocktwits platform ([Most followed stocks in stocktwits.com](http://www.investopedia.com/articles/markets/021316/most-followed-stocks-stocktwits-nflx-fb.asp)). We chose three such stocks for our sentiment analysis in this project - AAPL, FB, and TSLA. We also collected historical stock prices (close price, traded volume) for the same stocks for the same period from Yahoo! Finance (<http://finance.yahoo.com>). The second phase of the project aims to build a couple of models to predict the social sentiment in the form of a weighted score. We use this score to categorize the social sentiment of a tweet into one of the three categories - Bullish (positive), Bearish (negative), and Neutral (none). The first two models use a deterministic bag-of-words approach applying different financial dictionaries to guide the lookup of keywords in the tweet and further derive the sentiment score. The second model is based on a probabilistic approach using a Naive-Bayes Classifier. We train the Naive-Bayes model using training-set data and use the test-set data to predict the score. We provide confusion matrices for both approaches to highlight the accuracy of the respective models. The third phase of the project aims to build a regression model by using an aggregate daily sentiment score as a predictor of stock close price. We use GradientDescentOptimizer to train the regression model first and then use the test-data to predict stock close prices. We document model results and accuracy in the form of Tensor board summaries (loss) and computing graph. The final phase simulates the StockTwits continuous data feed and provides a sentiment score in near real-time using Spark.

### Definitions

1. **Tweet** - a short message posted by an user on stocktwits platform with a max of 140 characters. A tweet on StockTwit will always include a “cashtag” prefix denoting the ticker of the company relevant to the message. See example below.  
   <https://www.stocktwits.com/symbol/AAPL?q=AAPL>  
   
2. **Social Sentiment** **Category** - The tweet (a.k.a message) can be optionally tagged with a category (Bullish or Bearish) indicating the sentiment towards a particular stock. For example: the above tweet is tagged with a bullish sentiment. A bullish sentiment indicates user is optimistic about a particular stock’s performance in the near future. A bearish sentiment indicates user is pessimistic about stock’s performance in the near future.
3. **Social Sentiment Score** - Besides a category, one can assign a score for each tweet using several approaches. A score is a numeric value and we normalize it to be within [-1 to +1] for this project purpose. Values in the range [-0.3 to 0.3] indicates a neutral sentiment. A value of 0.3 and above indicates a positive sentiment and likewise, a value of -0.3 and below indicates negative sentiment.
4. **Daily Sentiment Score** - An aggregate sentiment score assigned to a stock on a daily basis by processing the sentiment score of individual tweets for the same day. This is the raw score.
5. **Weighted Daily Sentiment Score** - An aggregate sentiment score for a day calculated using raw sentiment score weighted by the ratio of tweet-volume for the day over the average-tweet-volume for the chosen time period (2 months in our case). A bullish or bearish overall sentiment on a given day should be weighted higher or lower depending on the number of tweets posted on that day in comparison to the rest of the days.  
     
   (n is the day index)  
   *TweetVolume(n) = # tweets for ‘n’the day.  
   AvgTweetVolume = Average # tweets per day considering two month period.*

***WeightedDailySentimentScore(n)*** *= (RawDailySentimentScore(n)) \* (TweetVolume(n) / AvgTweetVolume)*In our case, we collected historical tweets for a period of 2 months from March 1st, 2017 to April 30th, 2017. In that period, there were 42 trading days excluding holidays and weekends.

1. **Stock Close Price** - Stock price of a chosen company as indicated by the close price on a given trading day. All stock prices mentioned in this document refers to close price.
2. **LoughranMcDonald or Financial Dictionary** - This “dictionary” is a set of multiple lists of words, where each list pertains to a certain quality (e.g. positive, uncertain, strong, weak) specifically in the financial domain, as defined by Tim Loughran and Bill McDonald in their 2011 paper “When is a Liability not a Liability?” The two word lists we use are FinPos (financially positive sentiment) and FinNeg (financially negative sentiment), with 354 and 2,355 words respectively.
3. **Harvard IV-4 Dictionary** - This relates to a generalist “dictionary” which applies to non-financial contexts as well. This dictionary is the updated, expanded version of the popular Harvard IV-4 TagNeg dictionary. The two lists used are called Positiv (1,637 words) and Negativ (2,006 words).

### Project Scope and Assumptions

1. Based on our research, we chose three popular stocks on StockTwits platform - AAPL, FB, TSLA. Typically technology and social-media stocks tend to be much popular on social media platforms.
2. Historical tweets - Collected for three stocks (AAPL, FB, TSLA) for a time period of two months starting from March 1st to April 30th. This period had 42 trading days excluding holidays and weekends. In other words, we ignored tweets during holidays and weekends.
3. Each tweet collected from StockTwits.com can have an optional sentiment attached by user (either bearish or bullish). A majority of tweets are classified as none with the assumption that underlying tweets doesn't have any keywords that strongly indicate any sentiment. In other words, these tweets are classified under 'neutral' sentiment for the project demonstration purpose.

**Sources**

1. Investopedia - [Most followed stocks in stocktwits.com](http://www.investopedia.com/articles/markets/021316/most-followed-stocks-stocktwits-nflx-fb.asp)
2. StockTwits API - <https://stocktwits.com/developers/docs>
3. Yahoo! Finance for historical stock prices - [AAPL Historical Prices](https://finance.yahoo.com/quote/AAPL/history?period1=1488344400&period2=1493524800&interval=1d&filter=history&frequency=1d)
4. Bloomberg for Intraday prices - Bloomberg Terminal (subscription required)
5. LoughranMcdonald Financial Dictionary - <http://www3.nd.edu/~mcdonald/Word_Lists.html>
6. Harvard IV-4 Dictionary - <http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm>
7. Naive Bayes Classifier - ??

**API and Tools**

1. Python 2.6 on Windows-7 64-bit machine (Sentiment Analysis)
2. Python 3.5 on Windows-10 64-bit machine (Data download tweets)
3. Python 2.7 on Cloudera VM Linux (for Spark)
4. IDE - PyCharm and IPython Notebook
5. Packages - numpy, tensorflow, matplotlib, urllib, json, nltk, pandas, blpapi…
6. StockTwits RESTful API
7. Tensorboard API
8. Spark API
9. Bloomberg API

# Phase:1 Data Collection and Wrangling

### Historical Tweets

1. Data Source for tweets. [www.StockTwits.com](http://www.stocktwits.com)
2. URLs to download and process tweets using RESTful API -

AAPL:<https://api.stocktwits.com/api/2/streams/symbol/AAPL.json>

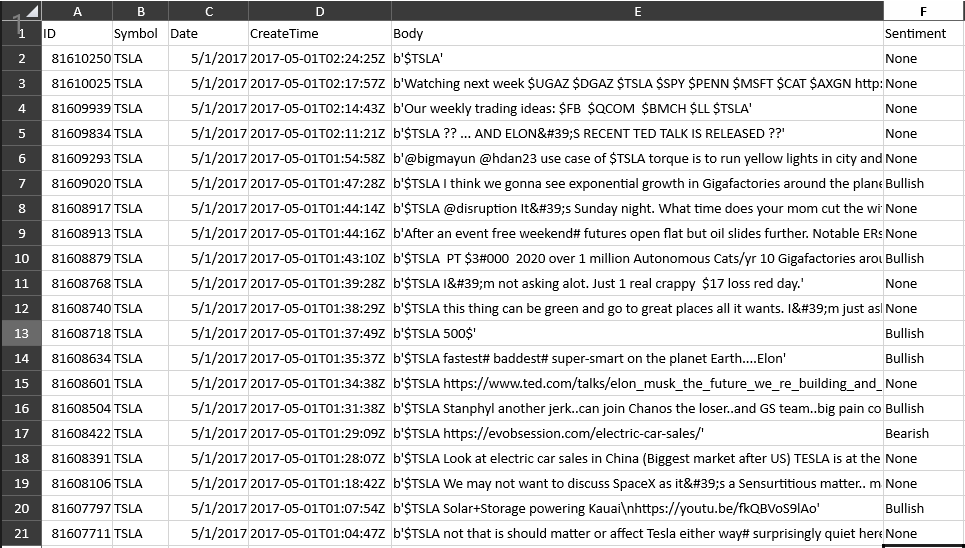
TSLA:<https://api.stocktwits.com/api/2/streams/symbol/TSLA.json>  
 FB: <https://api.stocktwits.com/api/2/streams/symbol/FB.json>

**Sample JSON response for AAPL tweets:**



**Pre-process tweet text:**

* Replace comma (‘,’) in the tweet body with #
* Replace non-ascii/non-printable chars in the tweet both with unicode-8 chars.

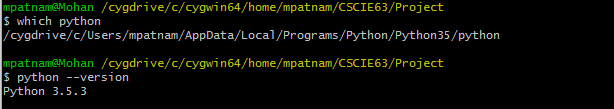


**Output CSV files:**

1. [Data/ClosePrices/TSLA.20170501.033001.csv](https://github.com/mpatnam/CSCIE63-Project/blob/master/Data/StockTwits/TSLA.20170501.033001.csv)
2. Data/ClosePrices/AAPL.20170430.191643.csv
3. Data/ClosePrices/FB.20170502.024702.csv

**Implementation details:**

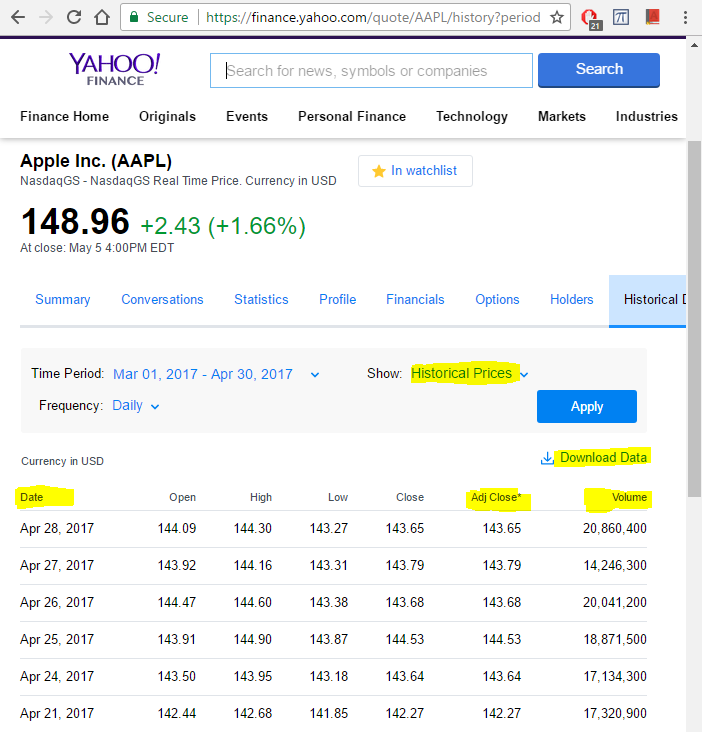
1. Python 3.5 on Windows-10 64-bit m/c.



1. See Code/Data Collection/do*wnload\_historic\_tweets.py* for download and pre-process script.  
   [GitHub link for download python script](https://github.com/mpatnam/CSCIE63-Project/blob/master/Code/Data%20Collection/download_historic_tweets.py)
2. Download logs: [GitHub link for sample AAPL download log](https://github.com/mpatnam/CSCIE63-Project/blob/master/Logs/Data%20Collection/aapl.20170430.log)

### Historical stock prices

1. Data source for historical prices: <https://finance.yahoo.com>
2. URLs to download csv files:   
   AAPL:<https://finance.yahoo.com/quote/AAPL/history?period1=1488344400&period2=1493524800&interval=1d&filter=history&frequency=1d>  
   TSLA:<https://finance.yahoo.com/quote/TSLA/history?period1=1488344400&period2=1493524800&interval=1d&filter=history&frequency=1d>  
   FB:<https://finance.yahoo.com/quote/FB/history?period1=1488344400&period2=1493524800&interval=1d&filter=history&frequency=1d>



**Output CSV files:**

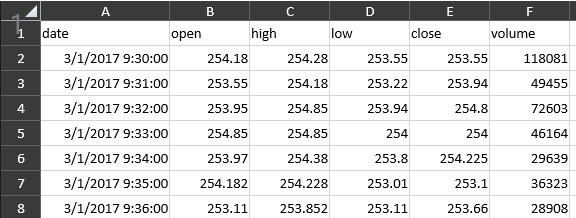
1. [Data/ClosePrices/ClosePrice.TSLA.csv](https://github.com/mpatnam/CSCIE63-Project/blob/master/Data/ClosePrices/ClosePrice.TSLA.csv)
2. Data/ClosePrices/ClosePrice.FB.csv
3. Data/ClosePrices/ClosePrice.AAPL.csv

Sample historical stock price (Stock: TSLA)



### Intraday stock prices (source Bloomberg)

1. Source: Bloomberg API (proprietory)  
   <https://www.bloomberg.com/professional/support/api-library/>
2. Sample csv file for intraday prices (Mar 28th) - Stock: TSLA



**Output CSV Files**

1. [Data/IntradayPrices/IntradayPrice.FB.csv](https://github.com/mpatnam/CSCIE63-Project/blob/master/Data/IntradayPrices/IntradayPrice.FB.csv)
2. Data/IntradayPrices/IntradayPrice.AAPL.csv
3. Data/IntradayPrices/IntradayPrice.TSLA.csv

# Basic Analysis

### Input CSV Files

1. [Data/ClosePrices/TSLA.20170501.033001.csv](https://github.com/mpatnam/CSCIE63-Project/blob/master/Data/StockTwits/TSLA.20170501.033001.csv)
2. Data/ClosePrices/AAPL.20170430.191643.csv
3. Data/ClosePrices/FB.20170502.024702.csv

### Summary CSV Files

1. [Results/Summary/SentimentVsClosePrices.TSLA.xlsx](https://github.com/mpatnam/CSCIE63-Project/blob/master/Results/Summary/SentimentVsClosePrices.TSLA.xlsx)
2. Results/Summary/SentimentVsClosePrices.FB.xlsx
3. Results/Summary/SentimentVsClosePrices.AAPL.xlsx

### Tweets basic stats:

**#days: 42**

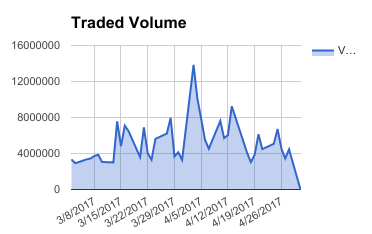
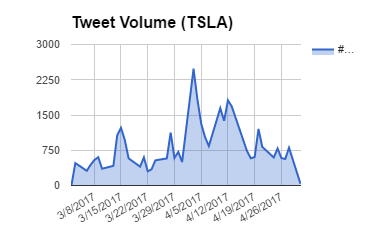
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stock** | **Total Tweets** | **Total Bullish** | **Total Bearish** | **Average Tweets/Day** | **Std Deviation (#Tweets)** |
| TSLA | 34,009 | 9175 | 5537 | 809 | 517 |
| AAPL | 24,550 | 7154 | 1988 | 584 | 265 |
| FB | 15,679 | 3883 | 1772 | 373 | 145 |

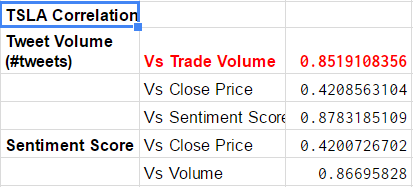
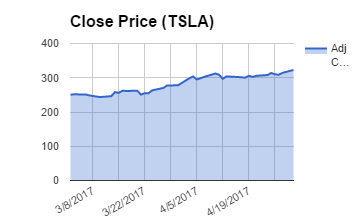
Note: Looking at standard deviation, it is clear that tweet volume varies significantly on a daily basis depending on company news, earnings etc.

### Basic time-series plots (Stock: TSLA):

#tweets vs close price,

#tweets vs traded volume





Observations:

1. Tweet volume and traded volume series shows high correlation (association) though it is not clearly if there is a causal relationship here.

# Phase:2 Sentiment Analysis - Bag-Of-Words Model

# Phase:3 Sentiment Analysis - Naive Bayes Classifier

Describe approach

Describe input data format

Any assumptions?

Show code snippets and logs

Results: Confusion Matrix

Daily Sentiment Score vs Close Price

# Phase:4 Real-time Streaming

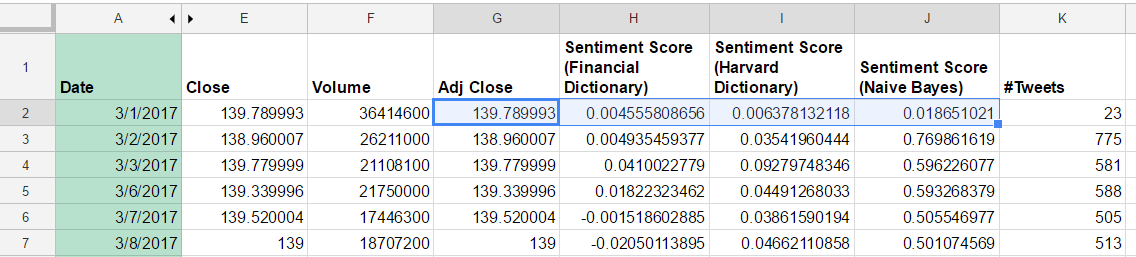
# 

# Phase: 5 Regression Model

### Input csv files:

1. [Results/Summary/SentimentVsClosePrices.TSLA.xlsx](https://github.com/mpatnam/CSCIE63-Project/blob/master/Results/Summary/SentimentVsClosePrices.TSLA.xlsx)
2. Results/Summary/SentimentVsClosePrices.FB.xlsx
3. Results/Summary/SentimentVsClosePrices.AAPL.xlsx

Sample summary document showing weighted sentiment scores from different models and stock close price (Adj Close).



### Regression Flavors (linear):

1. Predict Stock Price (Y) using Sentiment Score1 (Financial Dictionary)
2. Predict Stock Price (Y) using Sentiment Score2 (Harvard Dictionary)
3. Predict Stock Price (Y) using Sentiment Score2 (Naive Bayes)

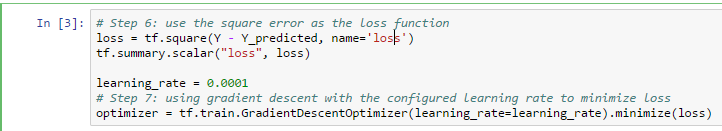
### Model Results:

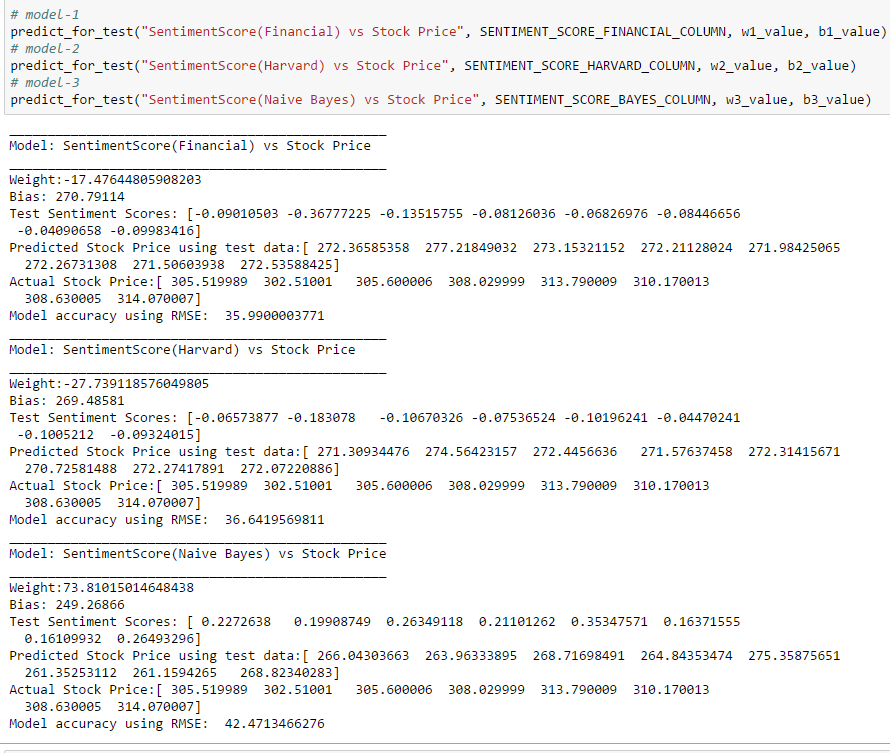
#iterations: 1000, Learning rate: 0.0001

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (RMSE)** | **Average Loss** |
| Sentiment Score (Financial Dictionary) vs Stock Price | 35.99 | 374.16 |
| Sentiment Score (Harvard Dictionary) vs Stock Price | 36.64 | 349.41 |
| Sentiment Score (Naive Bayes) vs Stock Price | 42.47 | 194.65 |

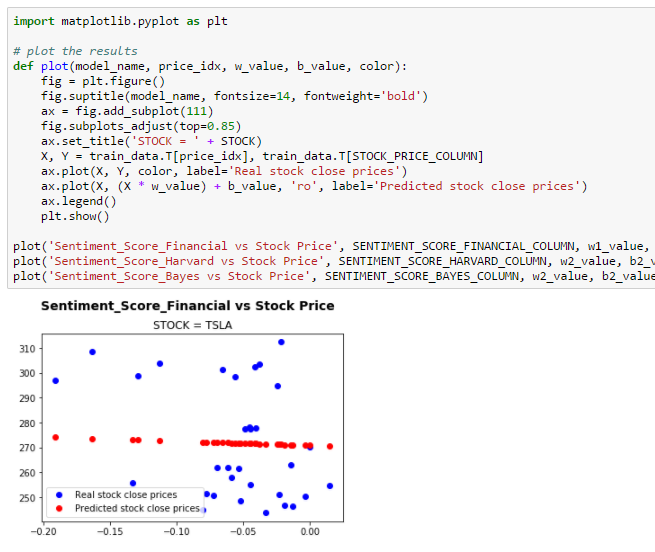
### Code:

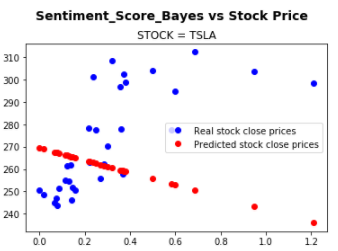
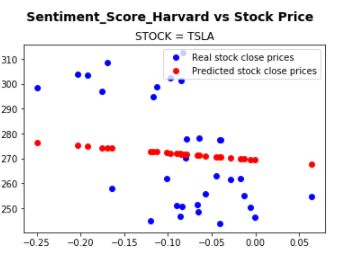
1. See [code/Regression/regression\_multi\_tensorflow.ipynb](https://github.com/mpatnam/CSCIE63-Project/blob/master/Code/Regression%20Model/regression_multi_tensorflow.ipynb) for linear regression models between SentimentScore (from various prediction models) vs Stock Close Price.
2. See [code/Regression/cubic\_regression\_tensorflow.ipynb](https://github.com/mpatnam/CSCIE63-Project/blob/master/Code/Regression%20Model/cubic_regression_tensorflow.ipynb) for cubic regression between SentimentScore (Naive Bayes model) vs Stock Close Price.



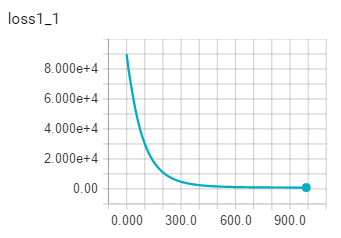
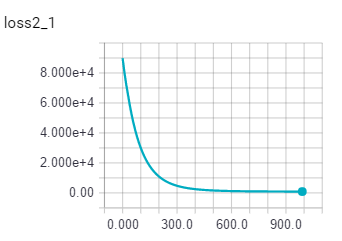


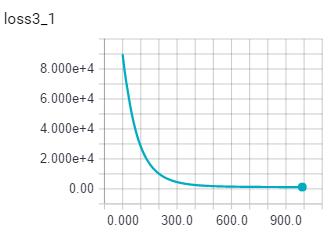
### Regression plots (Stock: TSLA)





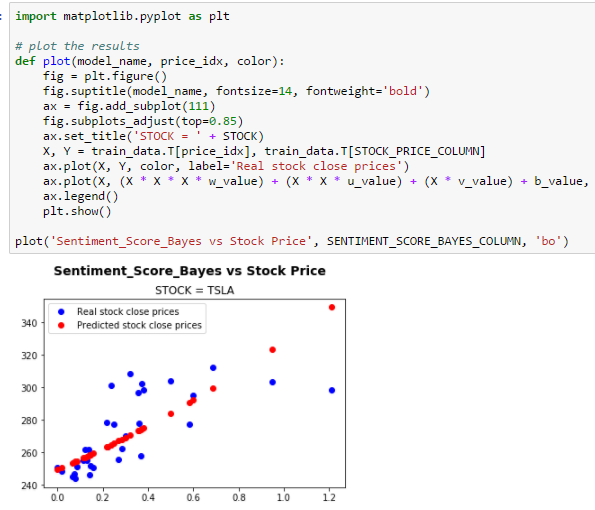
**Loss function from Tensorboard:**All three models converged at the same rate and stabilized after around 700+ runs.



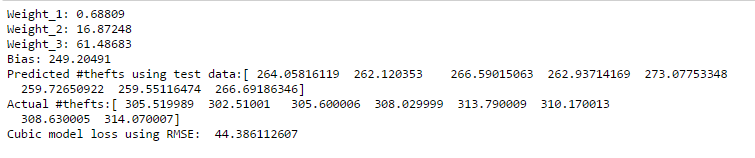


### Cubic regression (Stock: TSLA)

Evaluate Sentiment Score (Naive Bayes) against Stock Price

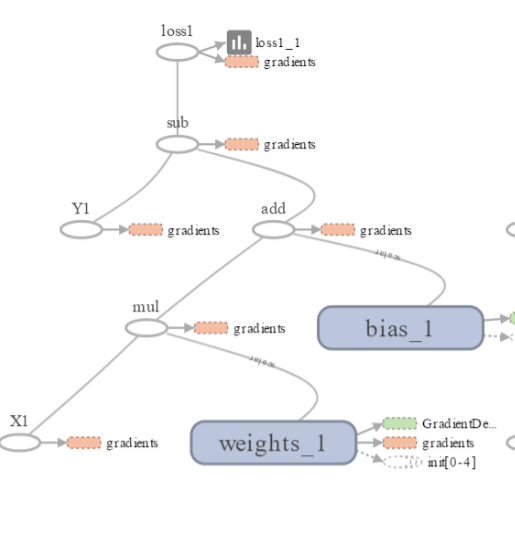


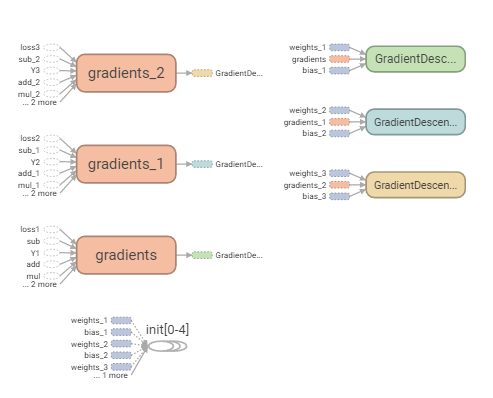
Cubic model parameters



### Tensorboard graph (Stock: TSLA)

Shows GradientDescentOptimizer training.





# Conclusion

Technology Benefits

* Provides a simple means to gauge the social sentiment of tweets and its effect on stock prices.
* Technology is built using Python with its ease of use and seamless integration with web downloads (RESTful API), text analysis (NLTK), streaming (Spark) and machine learning (tensorflow) packages.
* The system has the capability to parallel process the analysis using spark concurrency by simultaneously running on multiple stocks.

Drawbacks and Challenges

* StockTwits RESTful API limitation of 200 requests per hour. Each request returns 30 tweets only for a chosen stock. Running batch jobs continuously helped.
* Signal-to-noise ratio is often more. Tweets are often very unstructured mentioned in an informal language. Needs more sophisticated natural language techniques to derive the real intention of the tweet. Otherwise, this has implication on accuracy of models.

# Appendix

1. Data, code, logs and docs available as a github project: <https://github.com/mpatnam/CSCIE63-Project>
2. YouTube videos:
   1. Two minute (short):
   2. 15 minutes (long):
3. References:

* Investopedia - [Most followed stocks in stocktwits.com](http://www.investopedia.com/articles/markets/021316/most-followed-stocks-stocktwits-nflx-fb.asp)
* StockTwits API - <https://stocktwits.com/developers/docs>
* Yahoo! Finance for historical stock prices - [AAPL Historical Prices](https://finance.yahoo.com/quote/AAPL/history?period1=1488344400&period2=1493524800&interval=1d&filter=history&frequency=1d)
* Bloomberg API for intraday prices
* LoughranMcDonald Financial Dictionary - <http://www3.nd.edu/~mcdonald/World_Lists.html>
* Harvard IV-4 Dictionary - [http://www.wjh.harvard.edu/~inquirer/spreadsheet\_guide.htm](http://www.wjh.harvard.edu/~insuirer/spreadsheet_guide.htm)
* Tensorflow API - <https://www.tensorflow.org/api_docs/python/>