Spark Near Real-Time Sentiment Analysis

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

Motivation: *This project studies whether social media feeds focused towards stock market and concerning companies, are a suitable data source for forecasting the direction and volatility in 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.*

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 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). 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 [-0.5 to 0.5] indicates a neutral sentiment. A value of 0.5 and above indicates a positive sentiment and likewise, a value of -0.5 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.

6. **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.

7. **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.

8. **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).

9. **<< definitions for naive-bayes model>>**

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
2. StockTwits API - https://stocktwits.com/developers/docs
3. Yahoo! Finance for historical stock prices - AAPL Historical Prices
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 3.5 on Windows-10 64-bit machine and Linux (Cloudera VM) platform
2. IDE - PyCharm and IPython Notebook
3. Packages - numpy, tensorflow, matplotlib, urllib, json, nltk, pandas, blpapi…
4. StockTwits RESTful API
5. Tensorboard API
6. Spark
7. Bloomberg API
8. ...

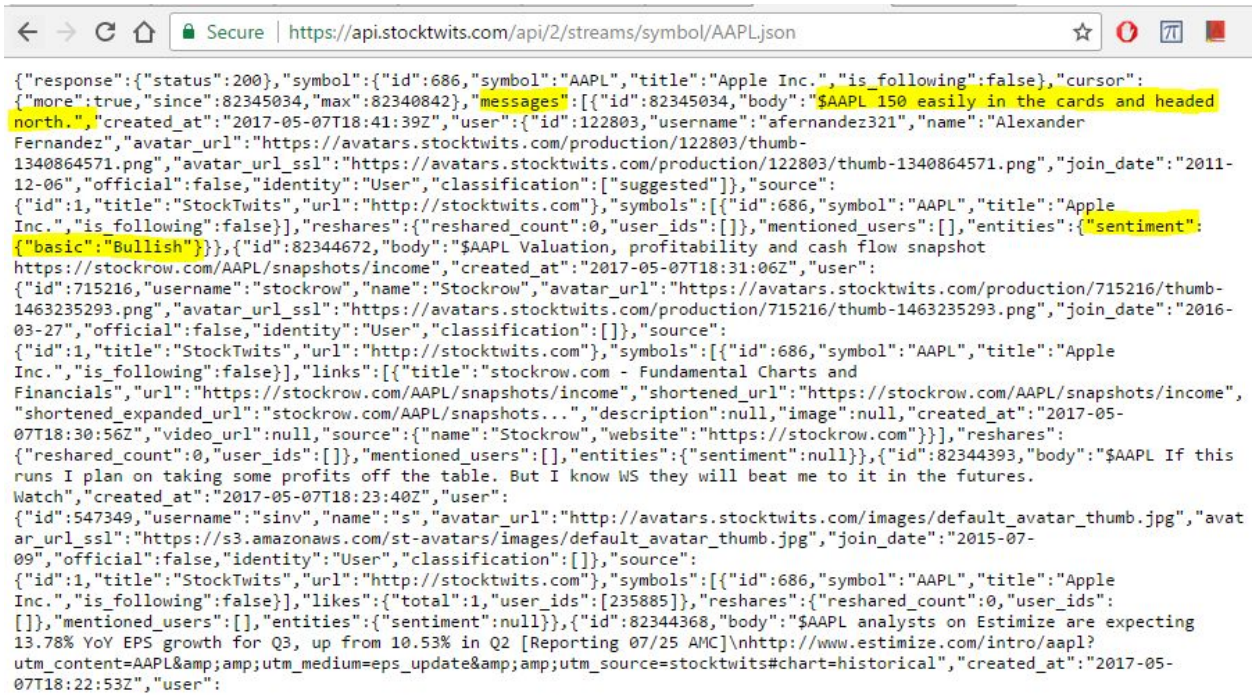
Data Collection and Wrangling

Data Source for tweets. www.StockTwits.com

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:



Show processed csv file

	A	B	C	D	E	F
1	ID	Symbol	Date	CreateTime	Body	Sentiment
2	81610250	TSLA	5/1/2017	2017-05-01T02:24:25Z	b'$TSLA'	None
3	81610025	TSLA	5/1/2017	2017-05-01T02:17:57Z	b'Watching next week $UGAZ $DGAZ $TSLA $SPY $PENN $MSFT $CAT $AXGN http:	None
4	81609939	TSLA	5/1/2017	2017-05-01T02:14:43Z	b'Our weekly trading ideas: $FB $QCOM $BMCH $LL $TSLA'	None
5	81609834	TSLA	5/1/2017	2017-05-01T02:11:21Z	b'$TSLA ?? ... AND ELON'S RECENT TED TALK IS RELEASED ??'	None
6	81609293	TSLA	5/1/2017	2017-05-01T01:54:58Z	b'@bigmayun @hdan23 use case of $TSLA torque is to run yellow lights in city and	None
7	81609020	TSLA	5/1/2017	2017-05-01T01:47:28Z	b'$TSLA I think we gonna see exponential growth in Gigafactories around the plan	Bullish
8	81608917	TSLA	5/1/2017	2017-05-01T01:44:14Z	b'$TSLA @disruption It's Sunday night. What time does your mom cut the wi	None
9	81608913	TSLA	5/1/2017	2017-05-01T01:44:16Z	b'After an event free weekend# futures open flat but oil slides further. Notable ERs	None
10	81608879	TSLA	5/1/2017	2017-05-01T01:43:10Z	b'$TSLA PT $3#000 2020 over 1 million Autonomous Cats/yr 10 Gigafactories arou	Bullish
11	81608768	TSLA	5/1/2017	2017-05-01T01:39:28Z	b'$TSLA I'm not asking alot. Just 1 real crappy $17 loss red day.'	None
12	81608740	TSLA	5/1/2017	2017-05-01T01:38:29Z	b'$TSLA this thing can be green and go to great places all it wants. I'm just as	None
13	81608718	TSLA	5/1/2017	2017-05-01T01:37:49Z	b'$TSLA 500$'	Bullish
14	81608634	TSLA	5/1/2017	2017-05-01T01:35:37Z	b'$TSLA fastest# baddest# super-smart on the planet Earth....Elon'	Bullish
15	81608601	TSLA	5/1/2017	2017-05-01T01:34:38Z	b'$TSLA https://www.ted.com/talks/elon_musk_the_future_we_re_building_and_	None
16	81608504	TSLA	5/1/2017	2017-05-01T01:31:38Z	b'$TSLA Stanphyl another jerk..can join Chanos the loser..and GS team..big pain co	Bullish
17	81608422	TSLA	5/1/2017	2017-05-01T01:29:09Z	b'$TSLA https://evobsession.com/electric-car-sales/'	Bearish
18	81608391	TSLA	5/1/2017	2017-05-01T01:28:07Z	b'$TSLA Look at electric car sales in China (Biggest market after US) TESLA is at the	None
19	81608106	TSLA	5/1/2017	2017-05-01T01:18:42Z	b'$TSLA We may not want to discuss SpaceX as it's a Sensurtitious matter.. m	None
20	81607797	TSLA	5/1/2017	2017-05-01T01:07:54Z	b'$TSLA Solar+Storage powering Kauai\nhttps://youtu.be/fkQBVoS9lAo'	Bullish
21	81607711	TSLA	5/1/2017	2017-05-01T01:04:47Z	b'$TSLA not that is should matter or affect Tesla either way# surprisingly quiet here	None

Tweets basic stats:

#days: 42

Stock	Total Tweets	Total Bullish	Total Bearish	Average Tweets/Day

AAPL	24,550	7154	1988	584
FB	15,679	3883	1772	373
TSLA	34,009	9175	5537	809

Implementation details:

Development environment:
Python 3.5 on Windows-10 64-bit./n

```
mpatnam@Mohan /cygdrive/c/cygwin64/home/mpatnam/CSCIE63/Project
$ which python
/cygdrive/c/Users/mpatnam/AppData/Local/Programs/Python/Python35/python

mpatnam@Mohan /cygdrive/c/cygwin64/home/mpatnam/CSCIE63/Project
$ python --version
Python 3.5.3
```

Download application logs:
https://github.com/mpatnam/CSCIE63-Project/blob/master/Logs/Data%20Collection/aapl.20170430.log

Historical prices (source Yahoo Finance)

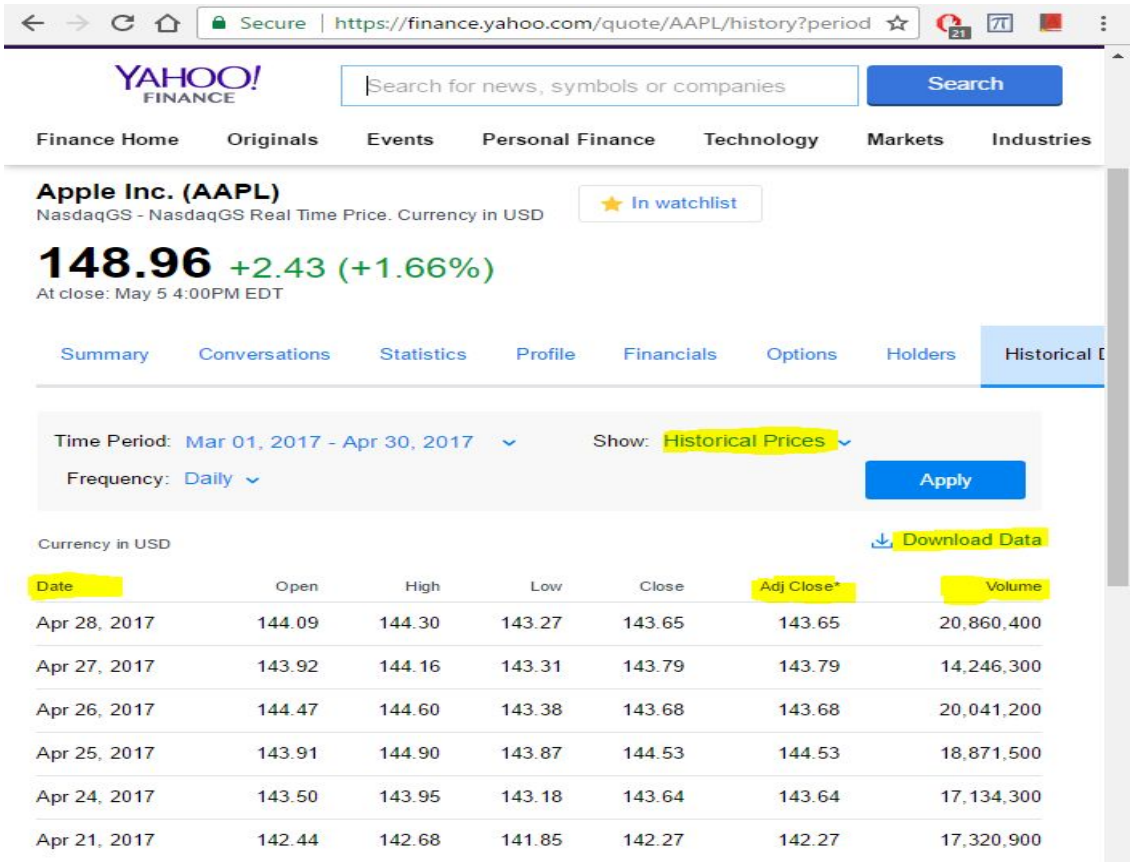
Show url for chosen stocks

Show sample table

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



Date	Open	High	Low	Close	Adj Close*	Volume
Apr 28, 2017	144.09	144.30	143.27	143.65	143.65	20,860,400
Apr 27, 2017	143.92	144.16	143.31	143.79	143.79	14,246,300
Apr 26, 2017	144.47	144.60	143.38	143.68	143.68	20,041,200
Apr 25, 2017	143.91	144.90	143.87	144.53	144.53	18,871,500
Apr 24, 2017	143.50	143.95	143.18	143.64	143.64	17,134,300
Apr 21, 2017	142.44	142.68	141.85	142.27	142.27	17,320,900

Intraday prices (source Bloomberg)
    Show sample table

Tweets processing - json/restful api
Python code details
Results: table showing #tweets collected for 3 stocks.

Sentiment Analysis

Basic analysis of tweets.
Describe input data format.
Basic stats for #tweets each day - SD/mean/median.
Show various plots.
    #tweets vs close price,
    #tweets vs traded volume

Bag-Of-Words Model

Describe Approach
Describe input data format

Any assumptions?
Show code snippets and logs
Results: Confusion Matrix
Daily Sentiment Score vs Close Price changes

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

Regression Model

Describe approach
Describe input data format

1	Date	Close	Volume	Adj Close	Sentiment Score (Dict)	Sentiment Score (Naive Bayes)	#Tweets
2	3/2/2017	250.479996	3342300	250.479996	-0.00244	-0.000215533	11
3	3/3/2017	251.570007	2919400	251.570007	-0.05781	0.145012757	477
4	3/6/2017	251.210007	3351200	251.210007	-0.01728	0.085262152	313
5	3/7/2017	248.589996	3449200	248.589996	-0.03891	0.017523558	439
6	3/8/2017	246.869995	3725200	246.869995	-0.01399	0.07358994	544
7	3/9/2017	244.899994	3861500	244.899994	-0.05968	0.067169388	601

Any assumptions?
Show code snippets and logs
Results: Linear regression plot
Tensorboard summaries and graph

Real time streaming

Future work

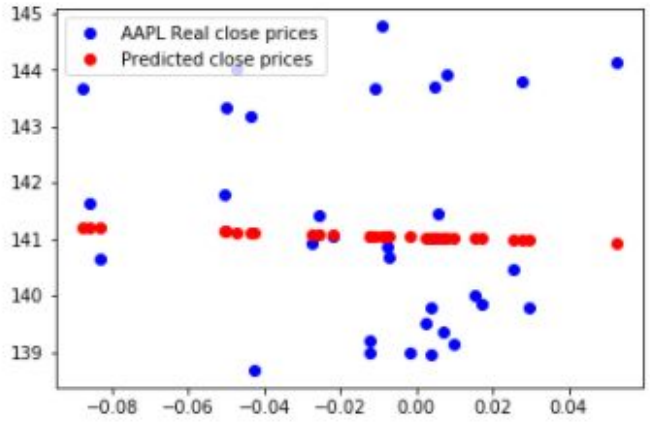
In [3]:
```
# Step 6: use the square error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')
tf.summary.scalar("loss", loss)

learning_rate = 0.0001
# Step 7: using gradient descent with the configured learning rate to minimize loss
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate).minimize(loss)
```

In [5]:
```
# calculate squared value (for RMSE purpose) given a predicted and actual value
def squared_error(pred, actual):
    return (pred - actual) ** 2

# Step 8: report results
print('Weight:{0}'.format(w_value))
print('Bias: %2.5f' % b_value)

#access test columns
X, Y = test_data.T[7], test_data.T[6]
print('Test Sentiment Scores: {0}'.format(X))

Y_predicted_test = X * w_value + b_value
print('Predicted Close Price using test data:{0}'.format(Y_predicted_test))
print('Actual Close Price:{0}'.format(Y))

# measure RMSE accuracy
accuracy_rmse = np.sqrt(squared_error(Y_predicted_test, Y).mean())
print('Model accuracy using RMSE: ', accuracy_rmse)
```

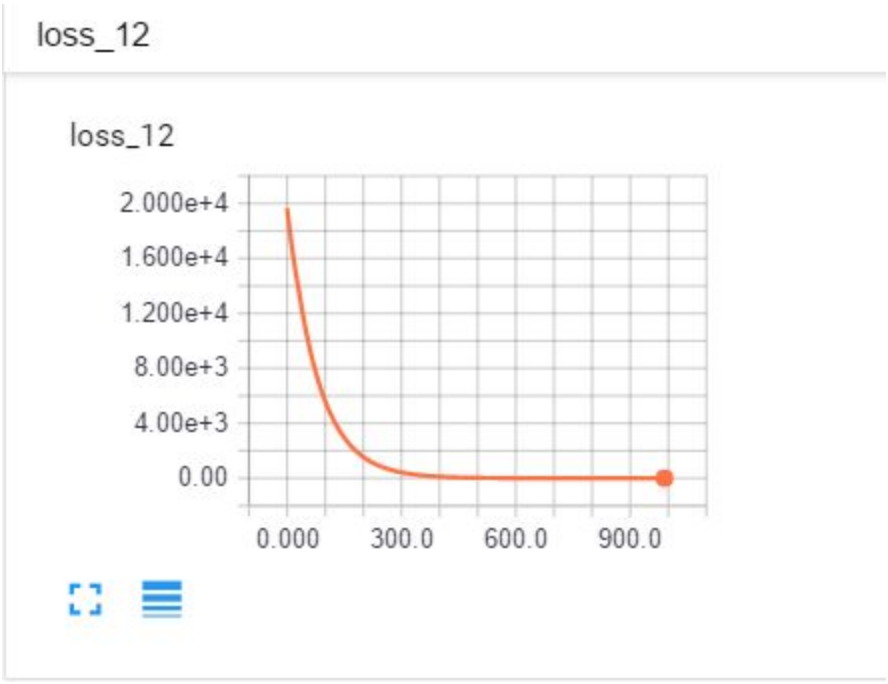
```
Weight:-1.8761721849441528
Bias: 141.03577
Test Sentiment Scores: [ 0.01009 -0.0343  -0.03518 -0.0207  -0.03843 -0.00984 -0.02089 -0.00961]
Predicted Close Price using test data:[ 141.01683602  141.10011931  141.10177034  141.07460337  141.1078679
  141.05422814  141.07495984  141.05379662]
Actual Close Price:[ 141.199997  140.679993  142.440002  142.270004  143.639999  144.529999
  143.679993  143.789993]
Model accuracy using RMSE:  2.12707369846
```

Regression plots for AAPL (Daily sentiment score vs close price)

```
In [115]: # plot the results
          X, Y = train_data.T[7], train_data.T[6]
          plt.plot(X, Y, 'bo', label=STOCK+' Real close prices')
          plt.plot(X, (X * w_value) + b_value, 'ro', label='Predicted close prices')
          plt.legend()
          plt.show()
```

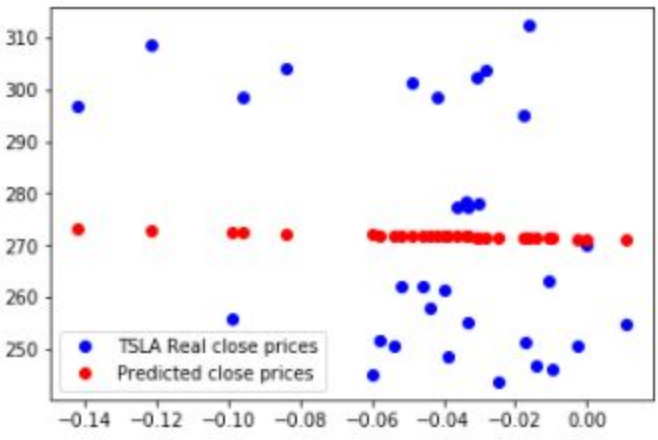


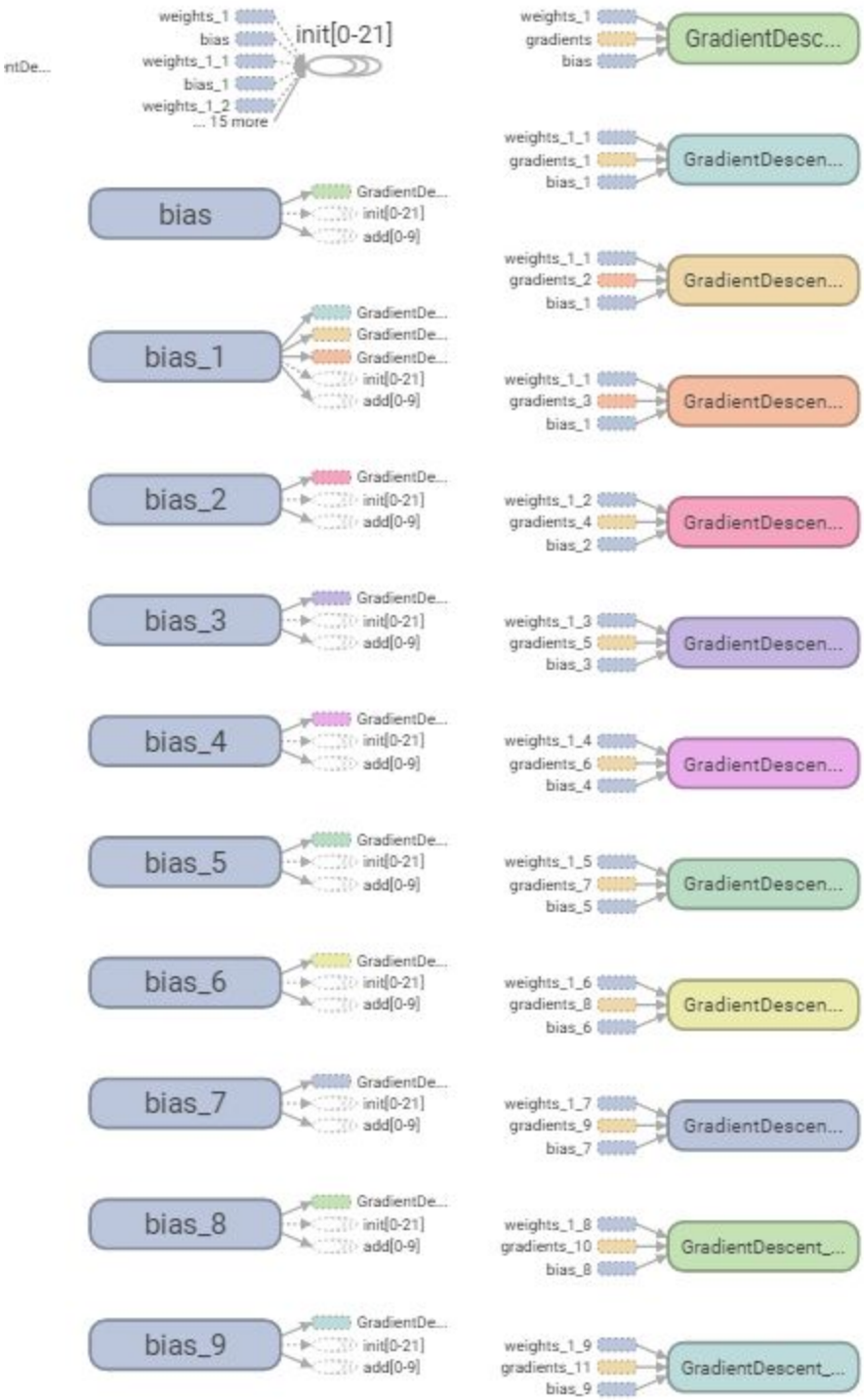
Loss function



Tensorboard graph for GradientDescentOptimizer training.

TSLA:


310
300
290
280
270
260
250
−0.14
−0.12
−0.10
−0.08
−0.06
−0.04
−0.02
0.00
TSLA Real close prices
Predicted close prices


weights_1
bias
weights_1_1
bias_1
weights_1_2
... 15 more
init[0-21]
GradientDesc...
gradients
GradientDescen...
gradients_1
gradients_2
gradients_3
gradients_4
gradients_5
gradients_6
gradients_7
gradients_8
gradients_9
gradients_10
gradients_11
GradientDescent_...
bias
bias_1
bias_2
bias_3
bias_4
bias_5
bias_6
bias_7
bias_8
bias_9
GradientDe...
init[0-21]
add[0-9]