Capstone Project 2: Twitter Sentiment Analysis and Stock Prediction

Problem Statement: Analyze stock market movements using Twitter sentiment analysis to find the correlation between ”public sentiment” and ”market sentiment”

I. Background

The value of individual stocks often do not seem to reflect the fair value due to human error. Instead, the price stocks trade at seem to be determined more by the human perception of the stock. Behavioral economics states that the emotions and moods of individuals affect their decision making process. People and news outlets are constantly voicing their opinions about a variety of subjects, stocks included, on Twitter. Though a single tweet may not be significant, a large collection of them can provide data with valuable insight about the common opinion on a particular subject. Twitter, therefore, can be used to gauge the public sentiment and possibly predict stock price movements. The famous research paper by Bollen et al had performed Twitter sentiment analysis to predict price movement of the Dow Jones Industrial Average (DJIA). However, this study will focus on 4 individual stocks: Netflix ($NFLX), Disney ($DIS), Amazon ($AMZN), and Google ($GOOGL).

II. Impact

Applying sentiment analysis to stock movement forecasting has been prominent field due to the potential financial gains from it. Without a doubt, investment banking firms have done extensive research and built model based of the idea. Twitter API makes its data readily available to the public, so why shouldn’t individual investors benefit from the wealth as well? By analyzing these trends and monitoring public opinion of companies, we can possibly build a predictive model to exploit market inefficiencies and anticipate changes in the market before they happen.

III. Data

A. **Twitter data**: The Twitter API will be used to pull tweets mentioning the stocks of interest. It is important to note that Twitter prevents users from pulling tweets past 7 days old with the standard API key. Sentiment Analysis “VADER” will be used to analyze the sentiment of the tweet. The data will include the following information:
   1. Date tweet was posted
   2. Content of tweet
   3. Follower count
   4. Percentage of Negative sentiment
   5. Percentage of Positive sentiment
   6. Percentage of Neutral sentiment

B. **Historical Stock Data**: Alpha Vantage API will be used to retrieve historical data for the stocks of interest. It will only be pulled starting from the earliest tweet pulled. The data will include the following information:

1. Date
2. Open
3. Close
4. High
5. Low
6. Traded Volume

Data Collection

To store all the data that I would need for the project, I created a SQLite database. SQLite is a relational database management system contained in the C library. A separate table was created for each company's twitter data and stock data. In total, 8 tables were created in the database.

Twitter Data

I collected tweets for the time span *05-15-2019 until 06-26-2019.* I used the Standard search API to collect the tweets I need for the project. It random samples the twitter pool and returns a collection of relevant tweets that match the parameters given. It has parameters to specify the date range to collect tweets from as well as query to filter by the content. When a user wants to explicitly mention someone or a product, they use "@" sign to gain attention from them. Therefore, if a user wrote a tweet specifically for a product, they would "@" the product. Some of the companies I am focused on have more products than their main product. To account for those as well, I have used the following queries for the respective companies:

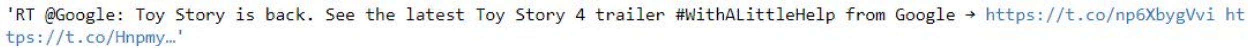
- Netflix: '@Netflix OR $NFLX OR Netflix'
- Disney: '@Disney OR @ESPN OR @ABCnetwork OR @Pixar OR @Marvel OR $DIS'
- Amazon: '@Amazon OR @PrimeVideo OR @awscloud OR @TwitchPrime OR @Alexa OR @WholeFoods OR $AMZN'
- Google: '@Google OR @Android OR @Waymo OR $GOOGL'

For each tweet, I collected the text of tweet, date created, and number of followers. Simultaneously, I used VADER (Valence Aware Dictionary and sEntiment Reasoner) to analyze the tweet. It is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. It is able to properly handle typical negations, conventional use of punctuation to signal increased sentiment intensity, slang words, and emojis. VADER then provides a positive, negative, and neutral score which are ratios for proportions of text that fall in each category. These score should add up to 1. It then computes a compound score which sums the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). The compound score is the normalized, weighted composite score.

Before using VADER to analyze the tweet, each tweet was cleaned. User mention was helpful in filtering out the tweets for the project, but it does not provide any sentiment value. Hyperlinks included in the tweet also do not provide any sentiment value. Hashtags may contain a lot of sentiment, so the text of the hashtag should be kept while the sign itself should be removed. Twitter has a "retweet" feature which allows users to quickly share a tweet by reposting it. It is prefaced with

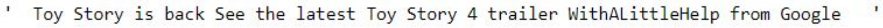
“RT,” but it does not contain any information and may confuse the tool. All “RT” should then be removed. The Regular Expression library was used to find strings of the sort and remove them.

Here’s an example of a pre-cleaned tweet:



```
'RT @Google: Toy Story is back. See the latest Toy Story 4 trailer #WithALittleHelp from Google → https://t.co/np6XbygVvi ht
tps://t.co/Hnpmy…'
```

Here is the cleaned tweet:



```
'  Toy Story is back See the latest Toy Story 4 trailer WithALittleHelp from Google   '
```

Stock Data

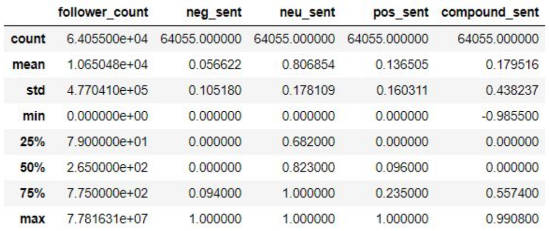
Alpha Vantage provides APIs for real time and historical data on stocks. With the help of the alpha_vantage package, collecting stock data was a breeze. I was able to easily search by specifying the abbreviation of the stock of interest. It had information for the date, open price, closing price, highest price, lowest price, and volume traded for each day. It is important to note that it skips weekends and holidays since the market is not open on those days.

Data Wrangling and Exploratory Data Analysis

Twitter Data

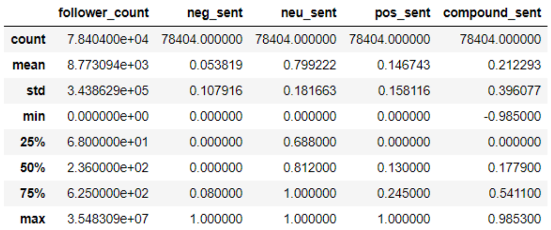
There wasn’t any missing values for any of the fields. Here is the summary statistics for each of the tables.

Netflix



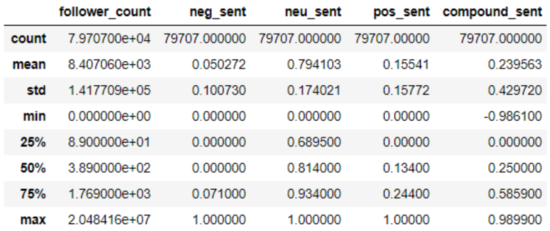
	follower_count	neg_sent	neu_sent	pos_sent	compound_sent
count	6.405500e+04	64055.000000	64055.000000	64055.000000	64055.000000
mean	1.065048e+04	0.056622	0.806854	0.136505	0.179516
std	4.770410e+05	0.105180	0.178109	0.160311	0.438237
min	0.000000e+00	0.000000	0.000000	0.000000	-0.985500
25%	7.900000e+01	0.000000	0.682000	0.000000	0.000000
50%	2.650000e+02	0.000000	0.823000	0.096000	0.000000
75%	7.750000e+02	0.094000	1.000000	0.235000	0.557400
max	7.781631e+07	1.000000	1.000000	1.000000	0.990800

Disney



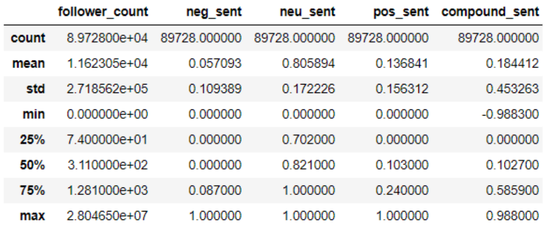
	follower_count	neg_sent	neu_sent	pos_sent	compound_sent
count	7.840400e+04	78404.000000	78404.000000	78404.000000	78404.000000
mean	8.773094e+03	0.053819	0.799222	0.146743	0.212293
std	3.438629e+05	0.107916	0.181663	0.158116	0.396077
min	0.000000e+00	0.000000	0.000000	0.000000	-0.985000
25%	6.800000e+01	0.000000	0.688000	0.000000	0.000000
50%	2.360000e+02	0.000000	0.812000	0.130000	0.177900
75%	6.250000e+02	0.080000	1.000000	0.245000	0.541100
max	3.548309e+07	1.000000	1.000000	1.000000	0.985300

Amazon



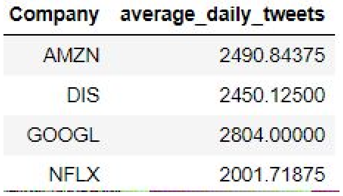
	follower_count	neg_sent	neu_sent	pos_sent	compound_sent
count	7.970700e+04	79707.000000	79707.000000	79707.00000	79707.000000
mean	8.407060e+03	0.050272	0.794103	0.15541	0.239563
std	1.417709e+05	0.100730	0.174021	0.15772	0.429720
min	0.000000e+00	0.000000	0.000000	0.00000	-0.986100
25%	8.900000e+01	0.000000	0.689500	0.00000	0.000000
50%	3.890000e+02	0.000000	0.814000	0.13400	0.250000
75%	1.769000e+03	0.071000	0.934000	0.24400	0.585900
max	2.048416e+07	1.000000	1.000000	1.00000	0.989900

Google



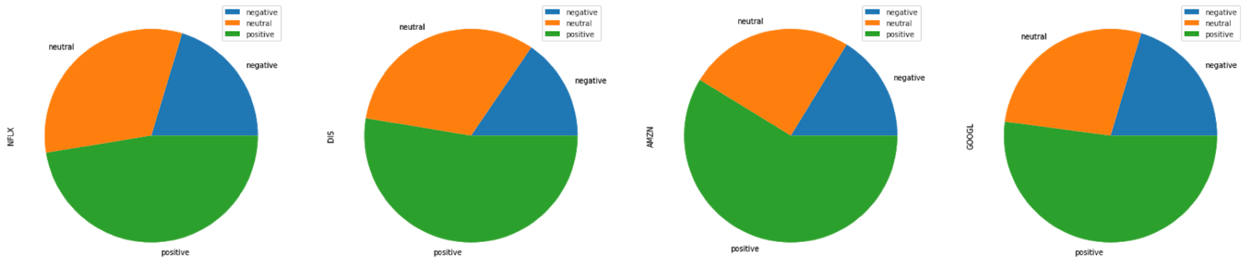
	follower_count	neg_sent	neu_sent	pos_sent	compound_sent
count	8.972800e+04	89728.000000	89728.000000	89728.000000	89728.000000
mean	1.162305e+04	0.057093	0.805894	0.136841	0.184412
std	2.718562e+05	0.109389	0.172226	0.156312	0.453263
min	0.000000e+00	0.000000	0.000000	0.000000	-0.988300
25%	7.400000e+01	0.000000	0.702000	0.000000	0.000000
50%	3.110000e+02	0.000000	0.821000	0.103000	0.102700
75%	1.281000e+03	0.087000	1.000000	0.240000	0.585900
max	2.804650e+07	1.000000	1.000000	1.000000	0.988000

Google, overall, had the highest number of tweets collected (89,728 tweets) Netflix had the lowest number (64,055 tweets). The following is the average number of daily tweets per company.

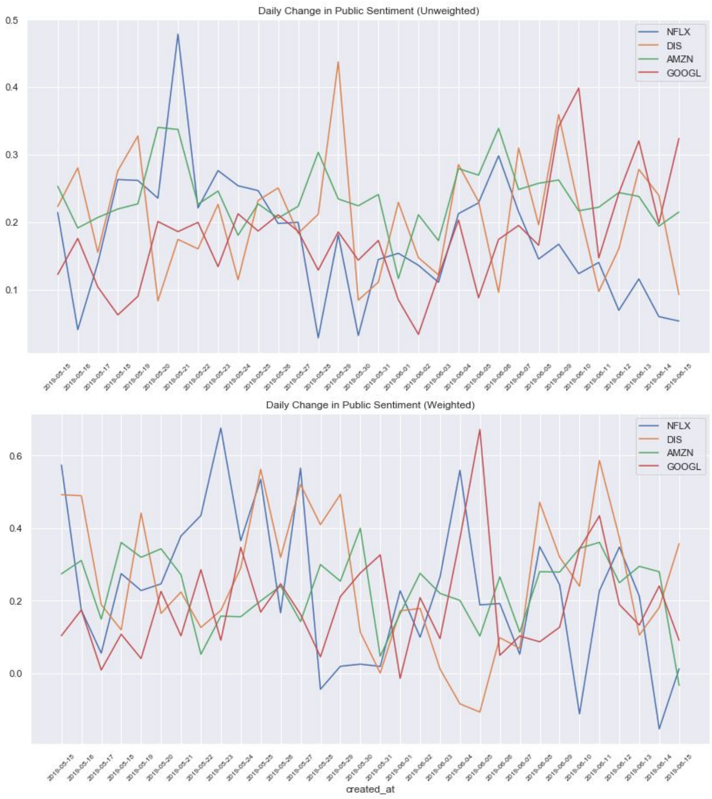


Company	average_daily_tweets
AMZN	2490.84375
DIS	2450.12500
GOOGL	2804.00000
NFLX	2001.71875

For easier interpretation of the compound scores, I have classified the scores into 3 groups in the “sentiment” column. The documentation for VADER advises that compound scores greater than 0.05 should be categorized as positive, less than -0.05 as negative, and those in between to be considered neutral. I also added a column for the name of the company so that all the data can still be distinguished once I merge all the tables together.



As shown by the pie plot, most of the tweets the companies received were positive. Amazon had the highest ratio for positive tweets while Netflix had the highest ratio for neutral tweets.

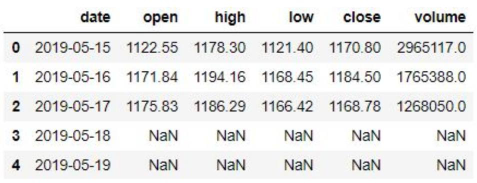


This table on the left illustrates the daily change in average public sentiment per stock. All the tweets are given equal weight when computing the average. However, twitter accounts that have more followers have more influence on the community since their tweet reaches more people. To incorporate this, we have given weight to each tweet according to proportion of followers they have compared to the total amount of followers for each day. The resulting graph is the graph below.

<u>Stock Data</u>

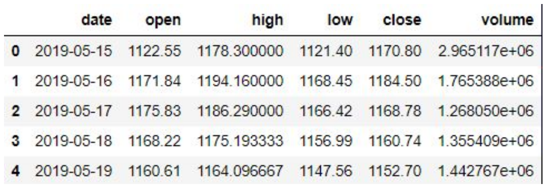
There wasn’t any missing values for any of the fields for these table either other than the weekend and holiday gaps. However, to see the trend in the time series, I opted to have continuous data. The information for the weekends and holidays were filled in using forward-filling linear interpolation. It estimates a new value by connecting two adjacent known values with a straight line.

Original Table:



	date	open	high	low	close	volume
0	2019-05-15	1122.55	1178.30	1121.40	1170.80	2965117.0
1	2019-05-16	1171.84	1194.16	1168.45	1184.50	1765388.0
2	2019-05-17	1175.83	1186.29	1166.42	1168.78	1268050.0
3	2019-05-18	NaN	NaN	NaN	NaN	NaN
4	2019-05-19	NaN	NaN	NaN	NaN	NaN

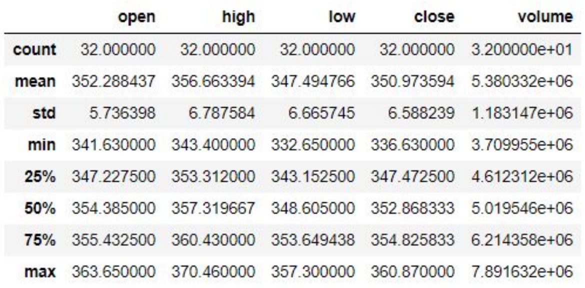
New Table After Interpolation:



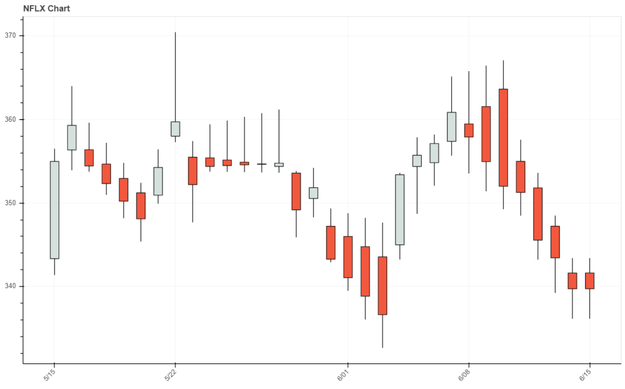
	date	open	high	low	close	volume
0	2019-05-15	1122.55	1178.300000	1121.40	1170.80	2.965117e+06
1	2019-05-16	1171.84	1194.160000	1168.45	1184.50	1.765388e+06
2	2019-05-17	1175.83	1186.290000	1166.42	1168.78	1.268050e+06
3	2019-05-18	1168.22	1175.193333	1156.99	1160.74	1.355409e+06
4	2019-05-19	1160.61	1164.096667	1147.56	1152.70	1.442767e+06

Here are the summary statistics candlestick graph for each stock. The days in green means that the price of the stock increased throughout the day while red means the price decreased throughout the day.

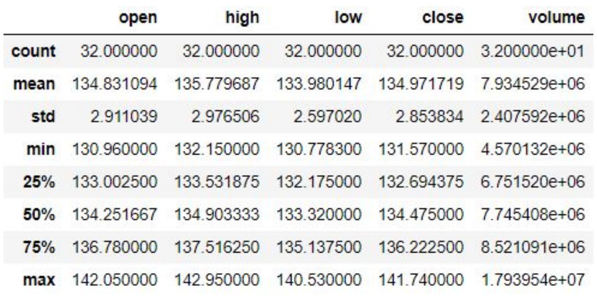
Netflix



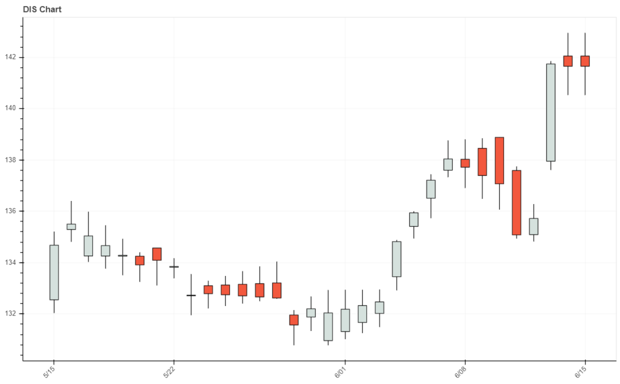
	open	high	low	close	volume
count	32.000000	32.000000	32.000000	32.000000	3.200000e+01
mean	352.288437	356.663394	347.494766	350.973594	5.380332e+06
std	5.736398	6.787584	6.665745	6.588239	1.183147e+06
min	341.630000	343.400000	332.650000	336.630000	3.709955e+06
25%	347.227500	353.312000	343.152500	347.472500	4.612312e+06
50%	354.385000	357.319667	348.605000	352.868333	5.019546e+06
75%	355.432500	360.430000	353.649438	354.825833	6.214358e+06
max	363.650000	370.460000	357.300000	360.870000	7.891632e+06



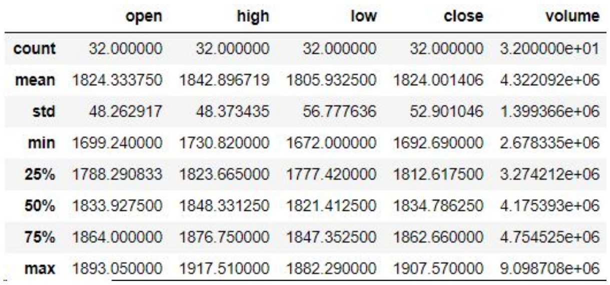
Disney



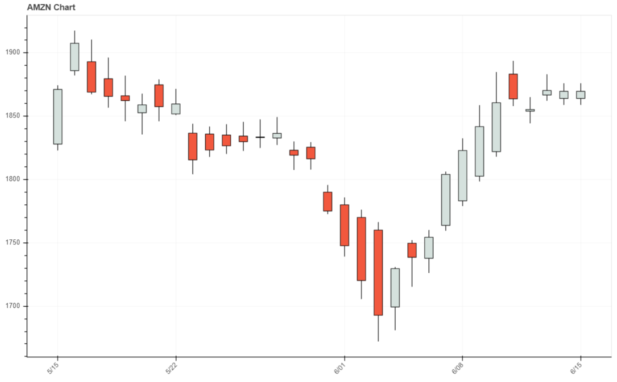
	open	high	low	close	volume
count	32.000000	32.000000	32.000000	32.000000	3.200000e+01
mean	134.831094	135.779687	133.980147	134.971719	7.934529e+06
std	2.911039	2.976506	2.597020	2.853834	2.407592e+06
min	130.960000	132.150000	130.778300	131.570000	4.570132e+06
25%	133.002500	133.531875	132.175000	132.694375	6.751520e+06
50%	134.251667	134.903333	133.320000	134.475000	7.745408e+06
75%	136.780000	137.516250	135.137500	136.222500	8.521091e+06
max	142.050000	142.950000	140.530000	141.740000	1.793954e+07



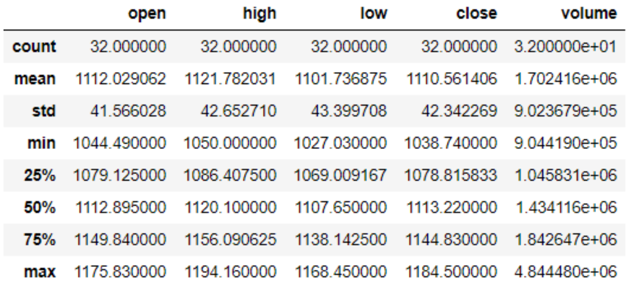
Amazon



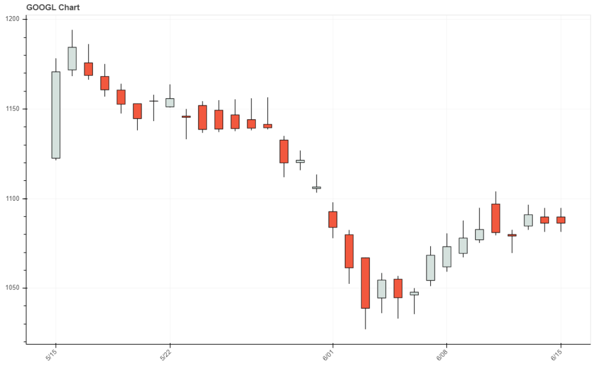
	open	high	low	close	volume
count	32.000000	32.000000	32.000000	32.000000	3.200000e+01
mean	1824.333750	1842.896719	1805.932500	1824.001406	4.322092e+06
std	48.262917	48.373435	56.777636	52.901046	1.399366e+06
min	1699.240000	1730.820000	1672.000000	1692.690000	2.678335e+06
25%	1788.290833	1823.665000	1777.420000	1812.617500	3.274212e+06
50%	1833.927500	1848.331250	1821.412500	1834.786250	4.175393e+06
75%	1864.000000	1876.750000	1847.352500	1862.660000	4.754525e+06
max	1893.050000	1917.510000	1882.290000	1907.570000	9.098708e+06

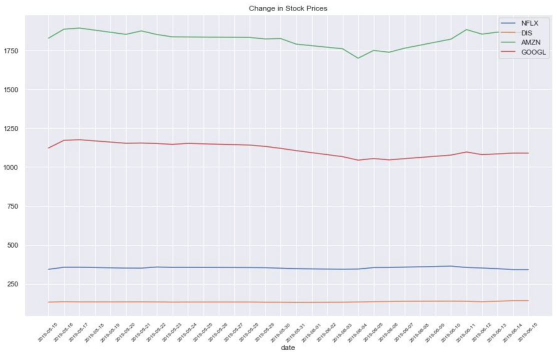


Google

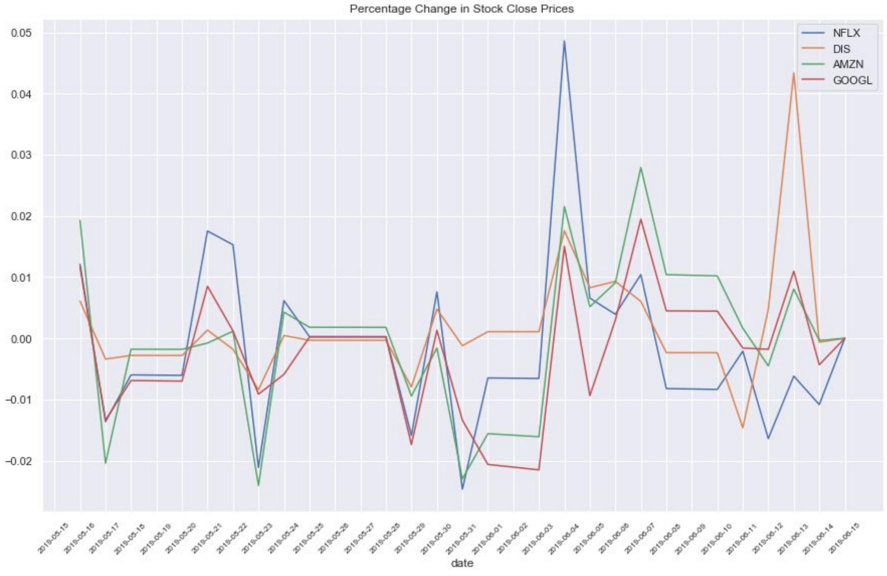


	open	high	low	close	volume
count	32.000000	32.000000	32.000000	32.000000	3.200000e+01
mean	1112.029062	1121.782031	1101.736875	1110.561406	1.702416e+06
std	41.566028	42.652710	43.399708	42.342269	9.023679e+05
min	1044.490000	1050.000000	1027.030000	1038.740000	9.044190e+05
25%	1079.125000	1086.407500	1069.009167	1078.815833	1.045831e+06
50%	1112.895000	1120.100000	1107.650000	1113.220000	1.434116e+06
75%	1149.840000	1156.090625	1138.142500	1144.830000	1.842647e+06
max	1175.830000	1194.160000	1168.450000	1184.500000	4.844480e+06





As shown by the average price per stock, Netflix and Disney are sold at way lower prices than Amazon and Google. It would be hard to see the trend in all the stock prices on one graph. Therefore, the prices must be standardized so that price movements would be comparable. I have computed the percentage change in price via difference in natural log of stock close prices.



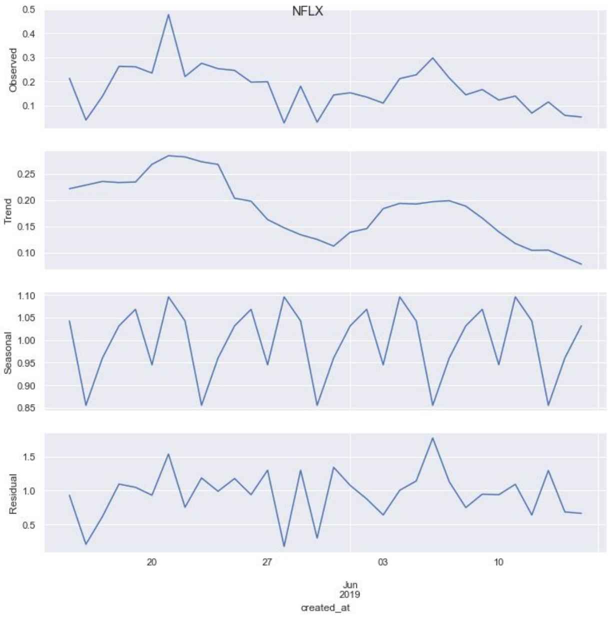
With this graph, it is more apparent when stock prices decrease or increase from day to day. Most of the stocks plummeted on 5-23-2019 then spiked up on 06-04-2019 and spiked again 6-13-2019.

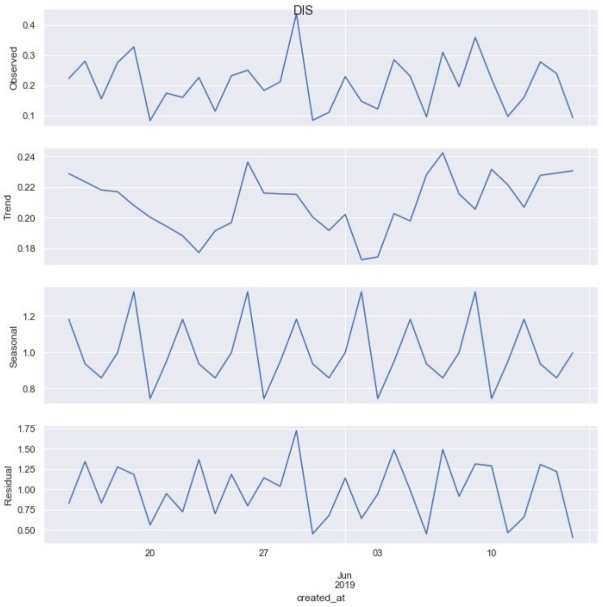
Time Series Analysis

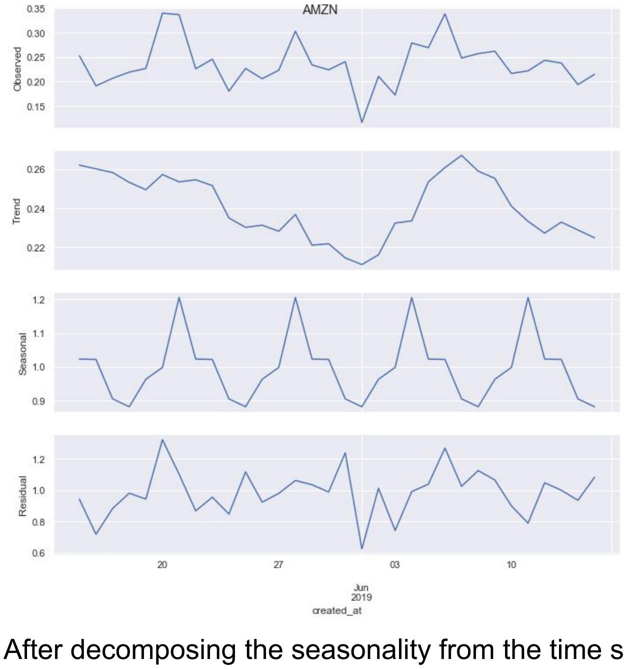
Time Series Decomposition

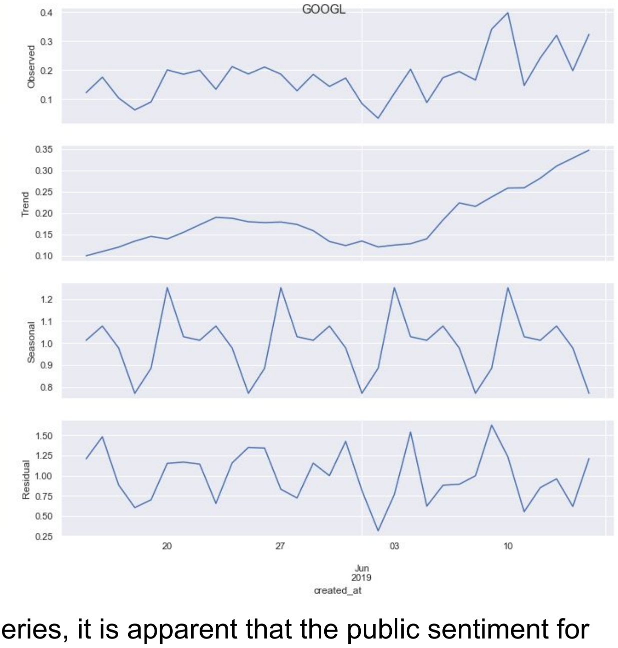
Time series may be split into: Base Level, Trend, Seasonality, Error. Below, I have used the multiplicative seasonal decomposition to break down the time series for public sentiment per company.

Multiplicative Time Series: Value = Base Level x Trend x Seasonality x Error









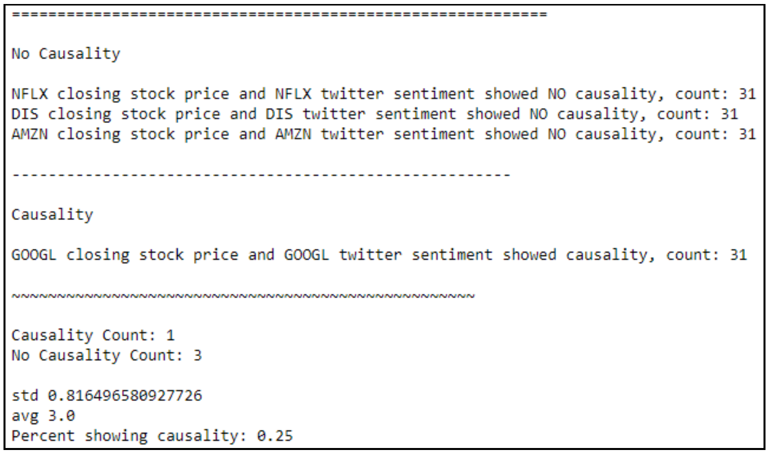
After decomposing the seasonality from the time series, it is apparent that the public sentiment for Netflix has decreased over time. Google, on the other, has an upward trend in public sentiment. The other two have a more balanced trendline.

Granger Causality

Granger causality tests whether the time series in the second column Granger causes the time series in the first column. Granger causality means that past values of x2 have a statistically significant effect on the current value of x1, taking past values of x1 into account as regressors. We reject the null hypothesis that x2 does not Granger cause x1 if the p-values are below a desired size of the test. In this case, I am testing if the public sentiment Granger causes shifts in the stock price at the 5% significance level.

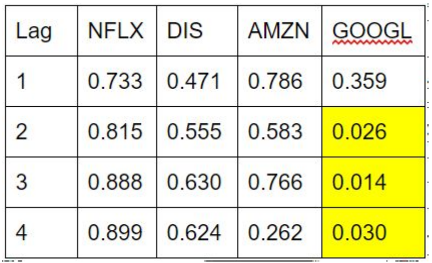
H0: The public sentiment for the company does NOT Granger cause the movement in stock price for that company.

***H1*:** The public sentiment for the company Granger causes the movement in stock price for that company.

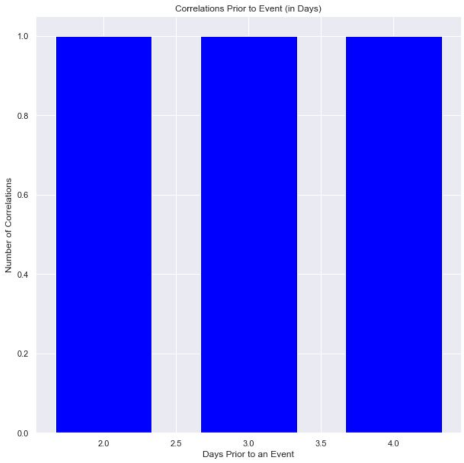


```
============================================================

No Causality

NFLX closing stock price and NFLX twitter sentiment showed NO causality, count: 31
DIS closing stock price and DIS twitter sentiment showed NO causality, count: 31
AMZN closing stock price and AMZN twitter sentiment showed NO causality, count: 31

--------------------------------------------------------

Causality

GOOGL closing stock price and GOOGL twitter sentiment showed causality, count: 31

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Causality Count: 1
No Causality Count: 3

std 0.816496580927726
avg 3.0
Percent showing causality: 0.25
```

Below are the p-values obtained using Granger causality analysis with different lags (in days)



Lag	NFLX	DIS	AMZN	GOOGL
1	0.733	0.471	0.786	0.359
2	0.815	0.555	0.583	0.026
3	0.888	0.630	0.766	0.014
4	0.899	0.624	0.262	0.030



It looks like the only relationship that was able to reject the null hypothesis was Google's public sentiment and market sentiment time series. There was a correlation shown between the public sentiment and market sentiment for 2,3,4 and 6 days prior to the stock price movements.

There's not enough evidence to prove that the public sentiment Granger caused the stock price movements for the other companies at the 95% confidence level.

Building the Forecasting Model

Feature Engineering

Each row in the twitter data set represents one tweet. As mentioned earlier, each company approximately received 2,500 tweets per day and the training set covers the span of 31 days. Therefore, the twitter data must be condensed into daily summary statistics. I am using the compound sentiment score to represent public sentiment. I computed the weighted average compound sentiment score, in which the weight is the number of followers for the tweet divided by the total number of followers for the day. This places higher weights on tweets that have more followers since their tweets would be seen by more people hence more influential. Another statistic that would seem useful is the proportion of tweets classified as negative, positive, and neutral. To prevent multicollinearity, the proportion of tweets classified as neutral has been left out.

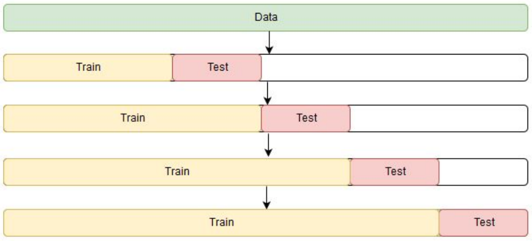
Fundamental analysis is a method of measuring a stock's intrinsic value by examining related economic factors. The daily percentage change in stock prices has been calculated to align all the company stocks onto the same scale. This will be the target variable. The volume of stocks traded for the day was also standardized. In addition to using historical prices of the stocks, I include the S&P 500 index, specifically the percentage change in price, as part of the market sentiment. Like the previous stock data, the weekend and holiday data is missing due to market closure. The gaps have been filled in using linear interpolation.

When dealing with time series data in machine learning, time lags must be introduced into the model in order to use past values to predict future values. Features revolving stock data has been shifted by one day. Features regarding the tweets has been shifted by 2 days since the Granger causality concluded that a 2 day lag was statistically significant. Since the data was shifted, 2 days had to be dropped from the model resulting in a train dataset with 30 rows.

Train Data Set	
Target Variable	Predictor Variables
- *Stock Percentage Change*	- *Stock Percentage Change (1 day lag)* - *Standardized Volume (1 day lag)* - *S&P 500 Percentage Change (1 day lag)* - *Weighted Compound Score (2 day lag)* - *Proportion of Negative Tweet (2 day lag)* - *Proportion of Positive Tweet (2 day lag)*

Machine Learning Model

The data from dates 05-15-2019 to 06-15-2019 is used as the training data while the last 10 days (06-16-2019 to 06-26-2019) is kept as testing data. The cross validation I will be using is the TimeSeriesSplit method. This cross-validation object is a variation of KFold. In the kth split, it returns first k folds as train set and the (k+1)th fold as test set.

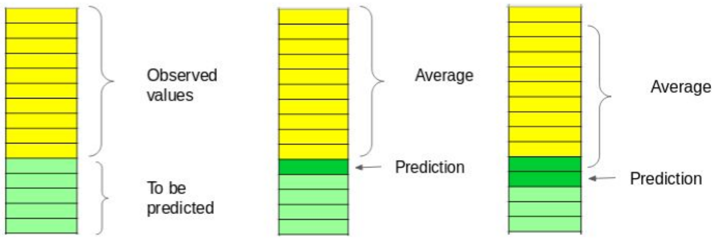


To evaluate the effectiveness of our methods, I use the root mean square error (RMSE), mean absolute percentage error (MAPE), and direction accuracy as metrics. RMSE is the standard deviation of the prediction error. MAPE measures the size of the error in percentage terms. For both

RMSE and MAPE, the lower the value, the better the prediction. The direction metric refers to if the model correctly predicted the direction of percentage change. In this case, the higher the value, the better the prediction.

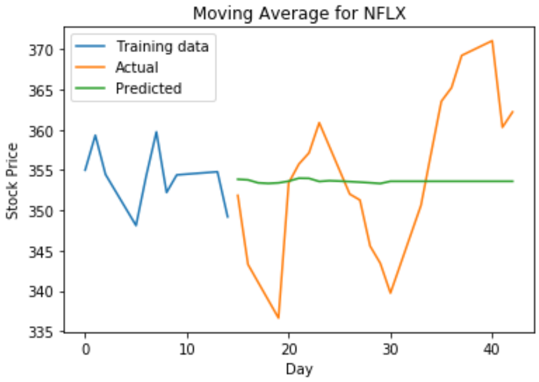
I. Simple Moving Average Model

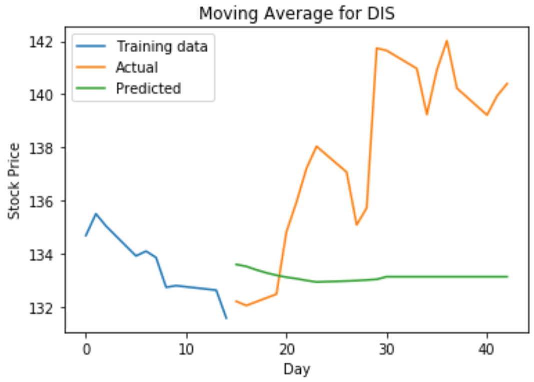
To lay the foundation of the model, I experimented with a simple moving average model that only incorporates historical stock data. The predicted closing price for each day will be the average of a set of previously observed values. The moving average technique uses the latest set of values for each prediction. In other words, for each subsequent step, the predicted values are taken into consideration while removing the oldest observed value from the set.

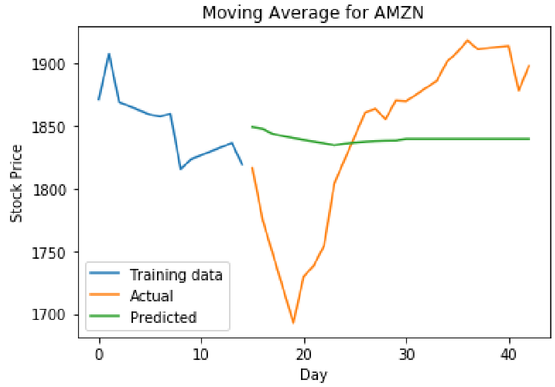


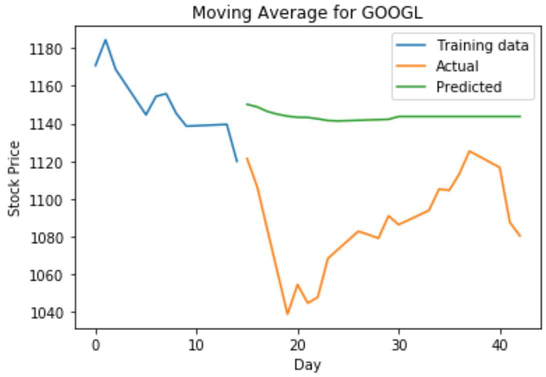
This model did not capture the volatile movements well. The predictions made were very steady with the initial average.

Company	RMSE	MAPE
NFLX	9.06	2.11%
DIS	4.57	2.70%
AMZN	71.14	3.24%
GOOGL	70.97	6.37%





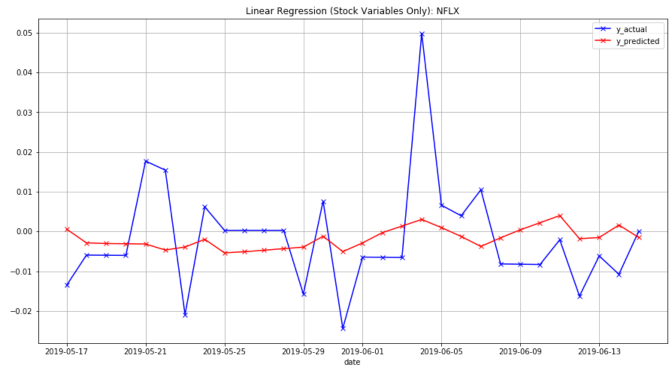




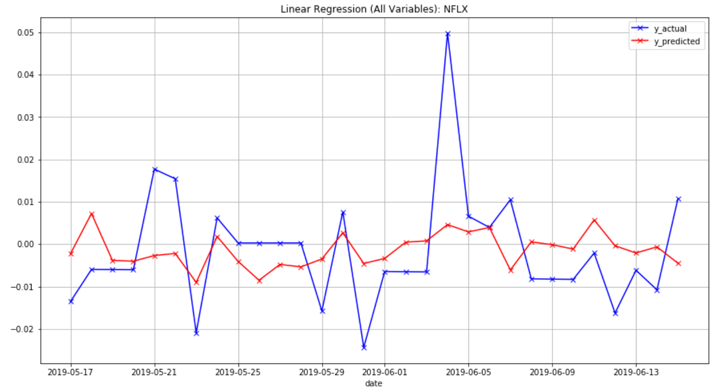
II. Regression Algorithms

I, then, start experimenting with the more machine learning algorithms. One of the algorithms used is Linear Regression, which finds the linear relationship between the target variables and predictors. The algorithm finds the best fit line with the smallest prediction error throughout. For the first linear regression model, I train the model using only the stock related variables (volume, previous stock growth, S&P 500) to predict stock growth. The model seems to generally follow the movement in stock growth, but it is not able to capture the magnitude of the growth.

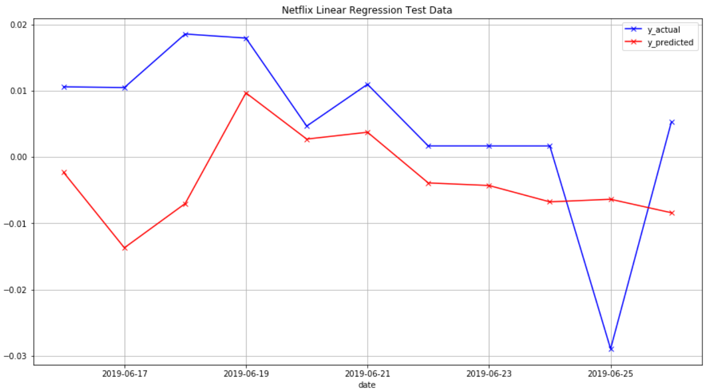
(Note: I trained a separate model for each company; however, individual model results and graphs for only Netflix will be shown. To see how the models performed for other companies, please refer to the github repo.)

NFLX	● RMSE: 0.258 ● MAPE: 54.94 % ● Direction: 44 %	

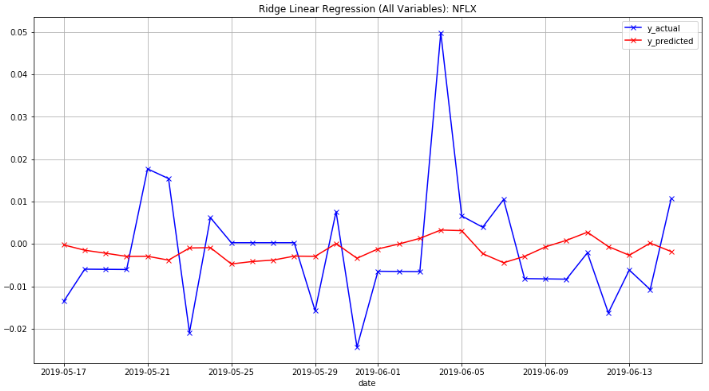
To improve the model, I run another linear regression model that uses the stock related variables as well as the sentiment variables. Although the MAPE worsened, both RMSE and direction accuracy improve. RMSE penalizes large prediction errors. Since investing requires as much accuracy in the predicted value and large errors are undesirable, RMSE will be used as the main metric in deciding the best model.

NFLX	● RMSE:0.040 ● MAPE: 32.93 % ● Direction: 32.0 %	

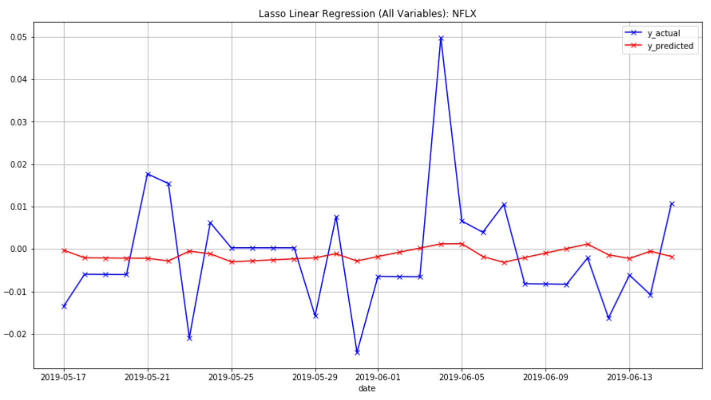
The model performed fairly well on the test data. Growth in S&P 500 for the previous day seemed to have the highest positive effect while stock growth for the previous day seemed to have the most negative effect on stock growth. All the sentiment related variables have a positive effect on stock growth. Interestingly, the percentage of tweets classified as negative had the highest positive effect out of the 3 sentiment variables.

NFLX	● RMSE: 0.014 ● MAPE: 1.096 % ● Direction: 18.18 % Coefficients ● sp_growth_l1: 0.716 ● percent_neg: 0.063 ● percent_pos: 0.025 ● w_avg_sent: 0.008 ● volume_l1: 0.003 ● stock_growth_l1: -0.305	

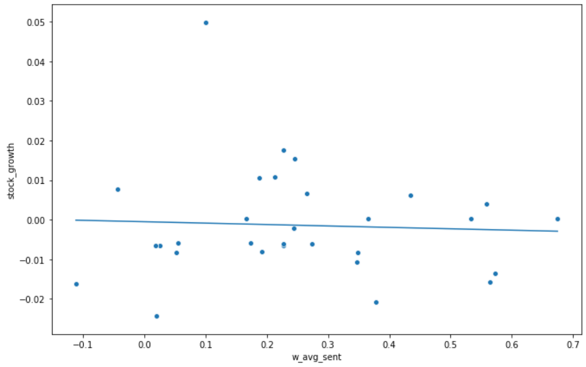
There are 2 other linear regression algorithms that regularize the model when it tends to overfit to the training data. Although our model does not overfit, I explore them to see how they perform. The first algorithm is Ridge regression. Ridge regression shrinks the coefficients and it helps to reduce the model complexity and multi-collinearity. In Linear regression, the cost function is the sum of squared prediction error. However, Ridge regression adds a penalty to the cost function that is equivalent to the square of the magnitude of the coefficients. As a result, the coefficients in Ridge regression converge towards 0 due to the shrinkage.

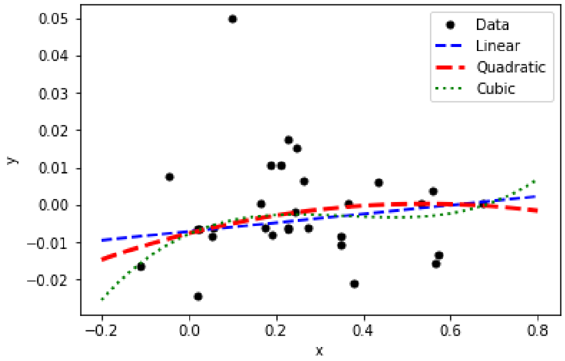
NFLX	● RMSE: 0.015 ● MAPE: 2.34 % ● Direction: 48.0 % Coefficients ● percent_neg: 0.004 ● volume_l1: 0.002 ● sp_growth: 0.000041 ● stock_growth_l1: -0.000212 ● w_avg_sent: -0.00092 ● Percent_pos: -0.00153	

The other algorithm is Lasso regression, in which the acronym stands for “least absolute shrinkage and selection operator.” Similar to Ridge, it also shrinks coefficients by adding a penalty to the cost function. However, the penalty is equivalent to the absolute value of the magnitude of the coefficients. This type of regularization (L1) can lead to zero coefficients, meaning some of the features are completely neglected for the evaluation of output. Therefore, Lasso regression not only helps in reducing overfitting, but it also helps in feature selection.

NFLX	● RMSE: 0.015 ● MAPE: 2.28 % ● Direction: 52 % Coefficients ● volume_l1: 0.001141 ● stock_growth_l1: 0 ● sp_growth: 0 ● w_avg_sent: 0 ● percent_neg: 0 ● percent_pos: 0	

Linear regression captures the patterns in the data better when the relation between the dependent variable and the independent variable is linear. However, the relationship between the target variable and predictor variables is not linear. To illustrate this, I have graphed the relationship between stock growth and average sentiment value. There is more variance towards the lower values in sentiment value than higher values. The cubic function seems to capture the relationship better.



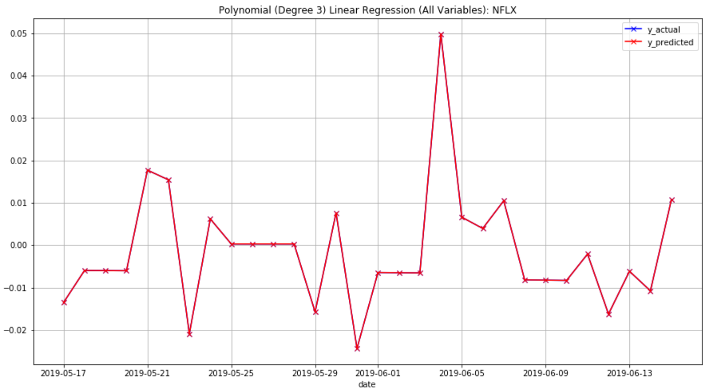


Polynomial regression increases the complexity of the model to overcome underfitting. To generate a higher order equation, it adds powers of the original features as new features. It is still considered to be linear model since the coefficients/weights associated with the features are still linear. I am fitting the model to polynomial function of degree 3 since it resulted in terms of all 3 metrics after I tested the model with degrees 1 to 5. When I used Linear regression as the algorithm, the model did well in terms of RMSE, but it seemed to have overfitted to the data. The polynomial function adds up to cubic powers to all the original features as well as interaction terms. As a result, there are 84 features opposed to the original 6. Luckily, Ridge and Lasso regression can regularize the model by penalizing the coefficients. It will diminish the coefficients that do not have much effect on stock growth.

<table>
<tr><th>Algorithm</th><th>Metrics</th><th>Visualization</th></tr>
<tr><td>Linear Regression</td><td>

- RMSE: 0.11
- MAPE: 79.8 %
- Direction: 52.0 %

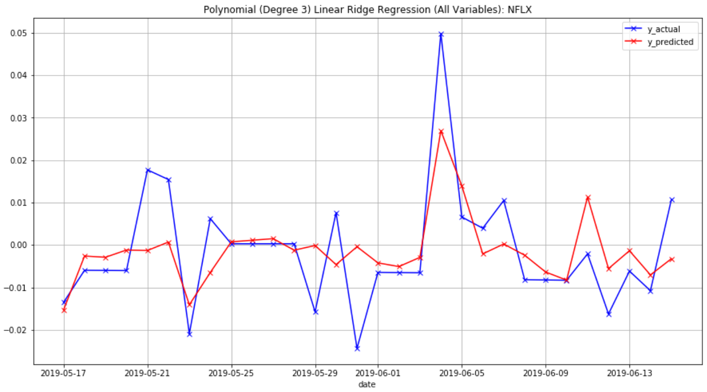
</td><td>



</td></tr>
<tr><td>Ridge Regression</td><td>

- RMSE: 0.021
- MAPE: 5.5 %
- Direction: 60.0 %

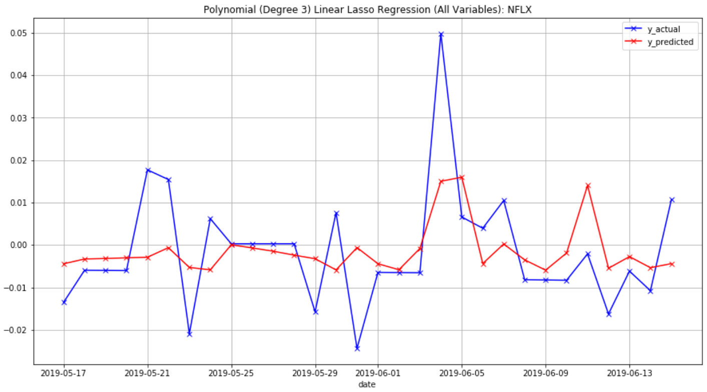
</td><td>



</td></tr>
<tr><td>Lasso Regression</td><td>

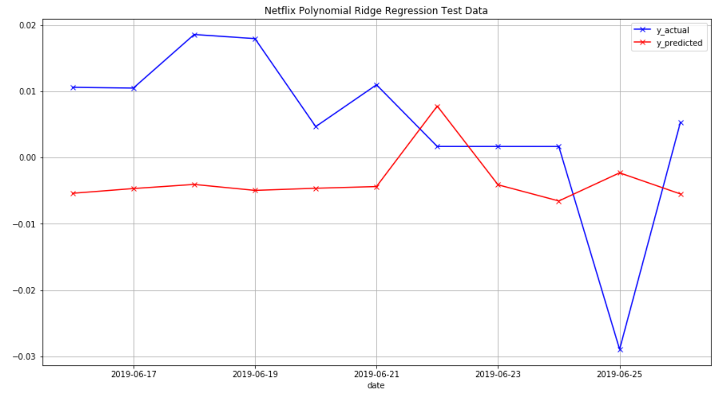
- RMSE: 0.015
- MAPE: 2.61 %
- Direction: 64.0 %

</td><td>

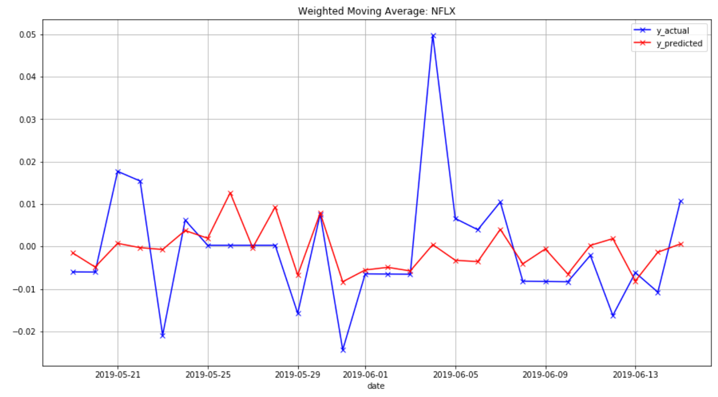


</td></tr>
</table>

Out of 3 models, polynomial regression with lasso performed the best in terms of all 3 metrics. It seems to perform better in predicting the stock growth values when it is close to 0. The model does not capture large fluctuations well. This is most likely due to the shrinkage of the coefficients which results in a model that loosely follows the trend. However, it is more important that we capture the fluctuations. Ridge regression seems to be a good middle ground between overfitting and underfitting. This model performs similarly to the linear regression without the polynomial regression. The RMSE is only a little worse (0.001 difference) and the direction accuracy remained the same. Both models did not anticipate the drop in growth on 6/25/2019.

NFLX	● RMSE: 0.015 ● MAPE: 2.19 % ● Direction: 18.18 %	

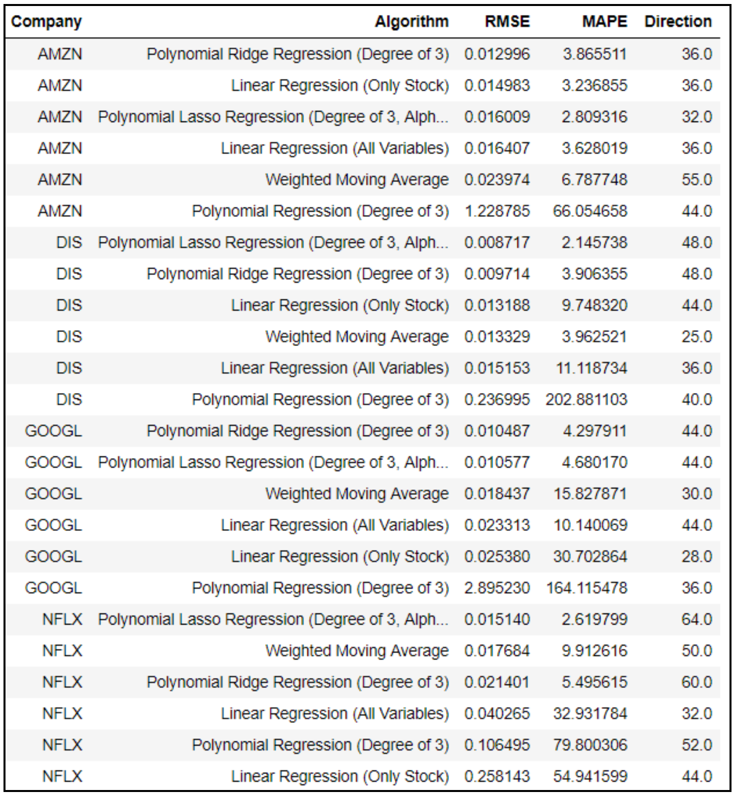
Since the polynomial regression may have over complicated the model, I try the weighted moving average model. In the weighted moving average model, I run a linear regression on historical stock growth and sentiment value data. The model determines the value of the weights given to each lag in the variables. For example, if I am running the model with a 3 day lag, the model will be trained on the stock growth and the sentiment value for the past 3 days. I tested the performance of models trained on historical data ranging from the past 2 days to the past 6 days. The model with the 2 day lag had the best RMSE score; however, it did horrible in terms of direction. The variables seem to be given too high of a weight in the model. The model with the 3 day lag, on the other hand, did well in terms of both RMSE and direction. Therefore, the weighted moving average model will be trained on data from the previous 3 days.

NFLX	● RMSE: 0.018 ● MAPE: 9.91 % ● Direction: 50 %	

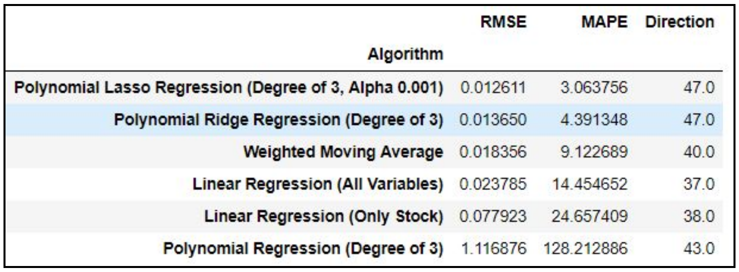
This model performs slightly better than the previous polynomial ridge regression model with a 0.003 difference in RMSE, but worse in direction accuracy. The coefficients of this model are expected to diminish in magnitude towards 0 as lag from the current day increases. However, the coefficients for the models seems to fluctuate over the 3 day lag, except for Google which converges towards 0 over time. This may mean that the number of lag days need to be increased, but there's not enough training data.

Coefficients	**Netflix**	**Disney**	**Amazon**	**Google**
Stock_growth_l1	-0.028	0.1464	0.1915	0.2215
Stock_growth_l2	-0.026	-0.1823	0.2210	0.1260
Stock_growth_l3	-0.058	0.0590	0.1622	0.0819
W_avg_sent_l1	0.007	-0.0033	0.0009	-0.0037
W_avg_sent_l2	-0.006	0.00823	-0.0145	0.0204
W_avg_sent_l3	0.026	-0.0111	-0.0339	-0.0081

Results



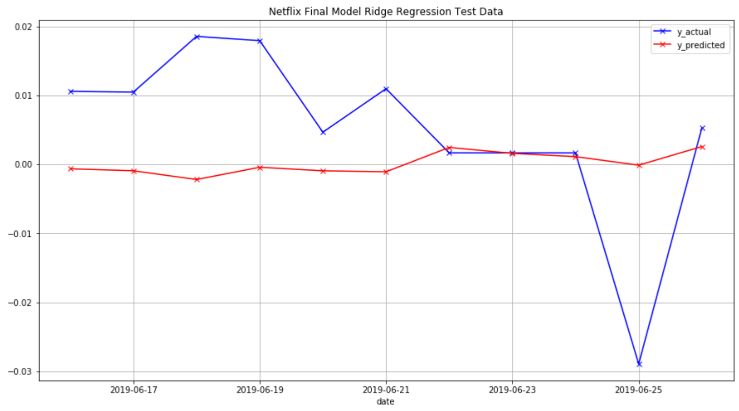
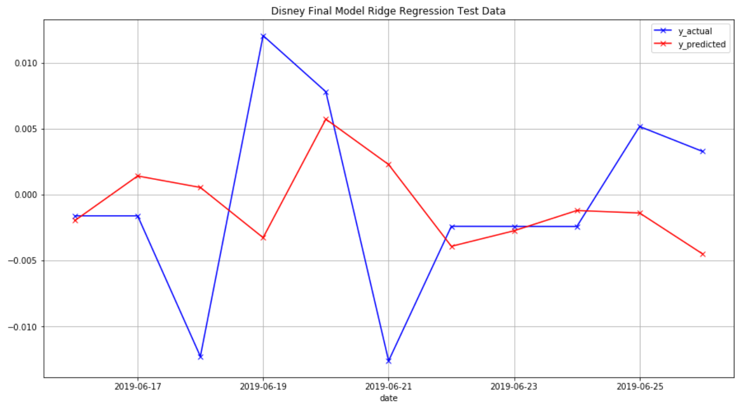
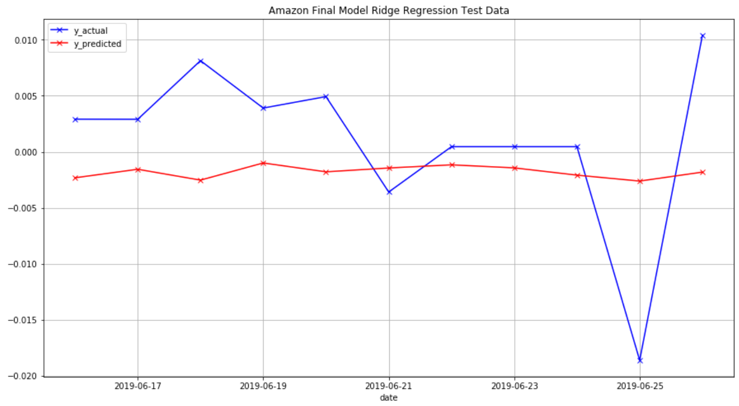
Company	Algorithm	RMSE	MAPE	Direction
AMZN	Polynomial Ridge Regression (Degree of 3)	0.012996	3.865511	36.0
AMZN	Linear Regression (Only Stock)	0.014983	3.236855	36.0
AMZN	Polynomial Lasso Regression (Degree of 3, Alph...	0.016009	2.809316	32.0
AMZN	Linear Regression (All Variables)	0.016407	3.628019	36.0
AMZN	Weighted Moving Average	0.023974	6.787748	55.0
AMZN	Polynomial Regression (Degree of 3)	1.228785	66.054658	44.0
DIS	Polynomial Lasso Regression (Degree of 3, Alph...	0.008717	2.145738	48.0
DIS	Polynomial Ridge Regression (Degree of 3)	0.009714	3.906355	48.0
DIS	Linear Regression (Only Stock)	0.013188	9.748320	44.0
DIS	Weighted Moving Average	0.013329	3.962521	25.0
DIS	Linear Regression (All Variables)	0.015153	11.118734	36.0
DIS	Polynomial Regression (Degree of 3)	0.236995	202.881103	40.0
GOOGL	Polynomial Ridge Regression (Degree of 3)	0.010487	4.297911	44.0
GOOGL	Polynomial Lasso Regression (Degree of 3, Alph...	0.010577	4.680170	44.0
GOOGL	Weighted Moving Average	0.018437	15.827871	30.0
GOOGL	Linear Regression (All Variables)	0.023313	10.140069	44.0
GOOGL	Linear Regression (Only Stock)	0.025380	30.702864	28.0
GOOGL	Polynomial Regression (Degree of 3)	2.895230	164.115478	36.0
NFLX	Polynomial Lasso Regression (Degree of 3, Alph...	0.015140	2.619799	64.0
NFLX	Weighted Moving Average	0.017684	9.912616	50.0
NFLX	Polynomial Ridge Regression (Degree of 3)	0.021401	5.495615	60.0
NFLX	Linear Regression (All Variables)	0.040265	32.931784	32.0
NFLX	Polynomial Regression (Degree of 3)	0.106495	79.800306	52.0
NFLX	Linear Regression (Only Stock)	0.258143	54.941599	44.0

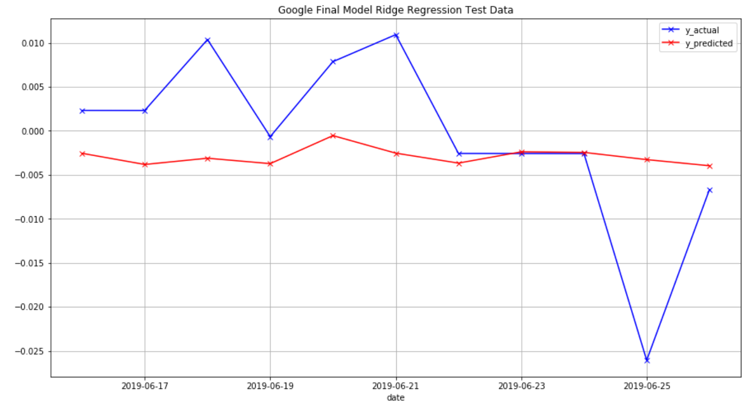


	RMSE	MAPE	Direction
Algorithm			
Polynomial Lasso Regression (Degree of 3, Alpha 0.001)	0.012611	3.063756	47.0
Polynomial Ridge Regression (Degree of 3)	0.013650	4.391348	47.0
Weighted Moving Average	0.018356	9.122689	40.0
Linear Regression (All Variables)	0.023785	14.454652	37.0
Linear Regression (Only Stock)	0.077923	24.657409	38.0
Polynomial Regression (Degree of 3)	1.116876	128.212886	43.0

The table on the left summarizes the performance of each model for each company. The model had improved given the public sentiment information as opposed to only stock variables for all the companies. Overall, it seems as if the polynomial regression with some sort of regularization modeled the data best.

To pick the final model, I average the RMSE across the different algorithm to see which algorithm has the lowest RMSE score. Polynomial regression with lasso regularization seems to perform the best in all 3 metrics. However, as stated before, it underfits the model with it's extreme regularization and does not capture the fluctuations that investor would typically want to know. Investors do not care if the stock grows by 0.01%; they care more if the stock would grow by 5%. That's when an investor would make the most money. Therefore, the final model will be the polynomial regression with ridge regularization.

Company	Metrics	Visualization
Netflix	● RMSE: 0.014 ● MAPE: 0.844 % ● Direction: 45.45 %	 Netflix Final Model Ridge Regression Test Data
Disney	● RMSE: 0.008 ● MAPE: 0.841 % ● Direction: 45.45 %	 Disney Final Model Ridge Regression Test Data
Amazon	● RMSE: 0.008 ● MAPE: 2.207% ● Direction: 18.18 %	 Amazon Final Model Ridge Regression Test Data

Google	● RMSE: 0.0097 ● MAPE: 1.442 % ● Direction: 54.54 %	

Although the RMSE scores are low, the models did not capture the drop on 6/25/2019 as I hoped it would. As shown from the time series decomposition earlier, the stock price movements for Netflix, Amazon, and Google had a flat trend. This could be the cause of why the values the model predicted does not have much variance and waivers around the same value. The model for Disney did best in terms of RMSE and direction accuracy. It is the only stock out of the 4 stocks in which the trend graph was not stable at all. This means that within the training time period, the Disney stock had a lot of movement, so it was able to capture these movements when the trained model was applied to the test data.

Conclusion and Limitations

In conclusion, stock price movements are considered unpredictable, but with data analysis and machine learning, there may be some predictor variables that explain these movements. Stock price essentially fluctuates due to supply and demand. The demand for the stock is affected by the public sentiment of the company, which this study had explored. In the simple model, the model was trained on solely historical stock data such as past stock price growth, volume, and S&P 500 stock price growth. Although it was able to follow a general pattern of the stock, the model was improved by adding variables related to public sentiment. The model was able to predict values closer to the actual values and had better accuracy in predicting if the stock price was going to increase or decrease. The model was ultimately improved when more features were engineered via polynomial transformation, which added features up until cubic power and interaction terms between. 81 features were created from this process, causing overfitting when trained. Therefore, Ridge regularization was incorporated to suppress the coefficients of the features that were not as important.

There are limitations to this study that should be considered when doing future work. The performance of the model was limited due to the size of the dataset. The time period needs to be extended so that the model is not trained on data with a flat trend. If there are more fluctuations in the data, the model would be able to capture price movements better. Another suggestion is having a better sentiment analyzer. The VADER did a decent job in scoring the distinctive tweets; however, it misclassified a lot of tweets as neutral when they had sentiment value. The predictor variance in the model may not explain enough variance in stock growth. It is known that stock movements are unpredictable, but a cumulation of a variety of predictors can capture as much of the variance as possible. The model used only stock data and S&P growth to explain the variance in the market. However, fundamental analysis typically incorporates information from business statements and economic factors to forecast stock prices. Exploring more regression algorithms may be another option in capturing the noise the model does not currently recognize. This study mostly explores variations of linear regression due to easy interpretability, but a more complex model may be capture the “randomness” better.

Reference

1. https://arxiv.org/pdf/1010.3003.pdf
2. https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/
3. https://blog.projectpiglet.com/2018/01/causality-in-cryptomarkets/
4. https://medium.com/cindicator/backtesting-time-series-models-weekend-of-a-data-scientist-92079cc2c540
5. https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b
6. https://towardsdatascience.com/machine-learning-with-python-easy-and-robust-method-to-fit-nonlinear-data-19e8a1ddbd49