Stock market movement prediction using Twitter Sentiment Analysis

Mayankkumar Tank (AU1841057)
Rahul Chocha (AU1841076)
Jeet Karia (AU1841109)
Prince Dalsaniya (AU1841124)

Abstract

Nowadays, social media is heavily influencing the opinions of people and people's opinions play huge role in stock market movements. And we can gather people's opinions, insights, and views from various social media forums, blogs which includes twitter, stockwits, and many more. Here we are demonstrating how sentiments on social media forums are influencing the trends of stock market prices.

Keywords

Sentiment Analysis, Stock Market Prediction, Supervised learning, Naive Classifier, Granger Causality Analysis (GCA), Auto-regressive integrated with Moving Average (ARIMA), Correlation, OLS Regression.

I. Introduction

Days back when Internet was just a dream for common people, at that time they were not surrounded by so much of information that we have now, or opinion which try to influence people's minds and change the whole trends of stock markets and many other important markets like Gold market, even the elections in some cases.

Here we are focusing on the stock market and its features like close price, open price, Returns, etc. And we are also establishing a relationship between trends of twitter features (Positive tweets, Negative tweets, Bullishness attitude) and stock market features (Close price, Open price, Volatility, Return) and seeing up to what extent they are correlated and if found correlatedness, we are forecasting one of the features on the basis of another using different forecasting techniques. We have used Correlation, Granger's Causality Analysis (GCA), Augmented Dickey Fuller test (ADF test) and F-test to find relationship.

And have used forecasting techniques like ARIMA (Auto-Regressive Integrated Moving Average) and OLS (Ordinary Least Squares) Regression.

II. Literature Survey

Anuja P Jain and Prof Padma Dandannavar have implemented different ML techniques and modes to find the Twitter sentiment analysis and compared their accuracy among different techniques (like naive bayesian classifier, logistic regression and etc.) [2].

They have also stated some of the different techniques to classify the sentiment of tweets. T. Rao and S.Srivastava have done accurate work related to predicting the stock market movements using Twitter sentiments. In their work, they work upon the particular timeframe which gives them better accuracy and also gives the better forecast results for the stock price. They also discussed what features are helpful from Twitter to forecast the stock price and also stated how to calculate it [1].

T.P. Souza and O.Kolchyna have done a similar kind of work where they are also applying the twitter sentiment to finance [4].

III. Implementation

A. Tweets Extraction

First, we have contacted Twitter API's developer rights for extracting tweets from Twitter according to our need. But the free account of Twitter's Developer API has constraints like using only 1000 tweets/per month which is not enough for us to move ahead. So, we started to scrap the data from the twitter web using web scrapping in python which gives us the tweets related to our works which means tweets about particular company and about particular time frame.

B. Tweet Data Cleaning and Preprocessing

Now, we have raw tweets scraped from the web. So the first task is to clean and pre-process it. So, we cleaned all tweets which include the removal of mentioned people and tag, removal of hashtags, removal of RT (Retweet) tag, removal of URLs present in the tweet. Next, we have removed all stop words from the tweets because they don't contain any sentiments and also removed all the punctuation marks from the tweets. Next, the Task in preprocessing of the tweets includes the spell correction and finding of the root word from any word which can be achieved by stemming and lemmatization. After all this processing we have completely cleaned and preprocessed and ready to use for classification.

C. Classification

In order to build a classification model we need a training dataset in which we have tweets and it's corresponding sentiment (positive,negative or neutral) . This is achieved using

the TextBlob library (in python) which gives us the polarity of the sentence. After achieving the balanced data set, the next task is to train the model for sentiment classification. Where, every unique word in dataset is a valuable feature for good accuracy.

Talking about the model, we have implemented the Multinomial Naive Bayes (MNB) model from scratch for sentiment classification. It assumes the conditional independence between every pair of features. The representation of the MNB given as, $P(y|x_0, x_1, ..., x_n)$. where, y is the label (positive, negative or neutral) and x_i are extracted features.

Here, p(label) is prior probability. That is calculated as,

$$p(y) = \frac{N_y}{N}$$

where N_y is number of tweets with label y and N is total number of tweets.

Next, calculation of conditional probability (i.e. probability of word given positive or negative) given as,

$$p(x|y) = \frac{N_{yi} + \alpha}{N_y + n\alpha}$$

where, N_{yi} is the frequency of a word in the sentences with label y and N_y is the total no of tweets with label y.

After calculating the $p(x_i|y)$ we can apply the equation of MNB. As MNB assume the conditional independence between each feature we can use the following formula,

$$p(y|x_0, x_1...x_n) = p(y) \prod_{i=0}^{n-1} p(x_i|y)$$

After finding the probability of a sentence with each label we can find the max($p(y|x_i)$) and assign the label to the sentence. i.e. if $p(positive|tweet) > p(negative|tweet)$ the tweet has positive sentiment. Here the value of α should be > 0. because if N_{yi}=0 which can lead to zero probability given label y after multiplication. Hence, we used α for smoothing.

D. Tweet Feature Extraction Related to stock market

After classifying tweets into positive and negative, We have to go for a finding features related to the stock market. Features are bullishness, agreement, and message volume, and they are calculated from the number of positive and negative tweets in particular time frames. So, we find the all features in various time frames dynamically. We found all the features day-wise, month-wise, and also week-wise. In order to find the number of positive and negative tweets from the dataset can be founded by grouping in the dataset. As per the literature, a number of positive tweets in a predefined time frame defined as $M_t^{Positive}$ and the number of negative tweets defined as $M_t^{Negative}$. In order to find other features, we have used the formula given in the [article-n]. Defining the bullishness for timeframe t given as,

$$B_t = ln\left(\frac{1 + M_t^{Positive}}{1 + M_t^{Negative}}\right)$$

Message volume for particular time frame t defined as natural logarithm of the total number of tweets for a particular company.

$$MessageVolume = ln(M_t^{Positive} + M_t^{Negative})$$

Agreement for particular time frame among positive and negative tweets is defined as,

$$A_t = 1 - \sqrt{1 - \frac{M_t^{Positive} - M_t^{Negative}}{M_t^{Positive} + M_t^{Negative}}}$$

Basically, Agreement shows that the all the tweets about particular company is bullish or bearish. If all the tweets related to particular company is positive (bullish behaviour) or negative (bearish behaviour) in that case agreement would be 1. To conclude, we have stock market related feature from the tweets are positive, negative, Bullishness, Message Volume and Agreement.

E. Financial Data Extraction

We have made use of yfinance API in python to extract financial data features of a company whose index we are supposed to give.

We can also provide the function with the range of days in which we want to do analysis.

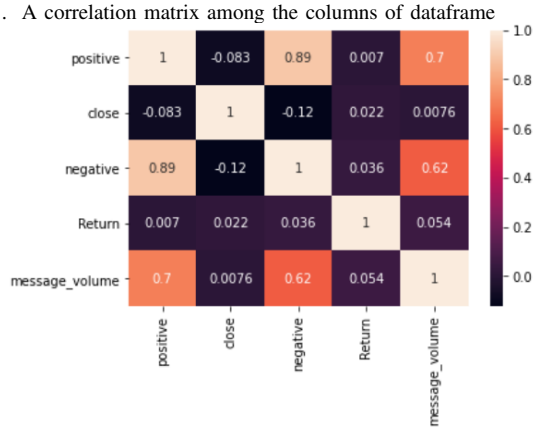
```
import yfinance as yf
msft = yf.Ticker("MSFT")
hist = msft.history(start=start_date, end=end_date)
```

MSFT is the ticker symbol for Microsoft stock prices and range of days are provided as start day and end day in above show code snippet.

F. Relationship among the features

First step was correlating both the data's features in the dataframe data type in python of Pandas. Below shows the figure of the correlation matrix heatmap which shows how much 2 different series are dependent on each other.

Fig. 1. A correlation matrix among the columns of dataframe



Then, to do Granger's Causality Analysis to see which feature causes which one, we need to first see if our series is stationary or not. Because in GCA, it is an assumption to use stationary series.

So, to check series' stationarity we do Augmented Dick Fuller Test (ADF Test).

1) ADF Test: A time series is stationary when its mean or standard deviation doesn't change with increase in time and if it changes then is called a non-stationary series. The null hypothesis in ADF Test is that the time series is not a stationary series.
And if on doing ADF test, we get the p-value > 0.05, we don't reject the null hypothesis and say that the series is non-stationary.
And if p-value $<= 0.05$, we reject the null hypothesis and the series is stationary.

2) F-test: Then, we also need to find the maximum lag for our series with the help of which we will convert our non-stationary series to stationary series. And differentiating the series to convert it to the stationary one.
So, first calling function named grangercausalitytests in which we provide arbitraty lag and using the f-test value, the lag with the highest F-value is the lag that we will choose.

3) Non-stationary to Stationary: After obtaining the lag value, we differentiate the series using diff function in numpy library of python and convert the series to a stationary one.

4) Granger's Causality Analysis: Now, after converting to stationary series, we use Granger's Causality Test to find which series causes which.
If p values for 2 variables turns out to be larger than some threshold value, then we say that change in one leads to change in other.
Below is the figure which shows the GCA matrix obtained for our features.

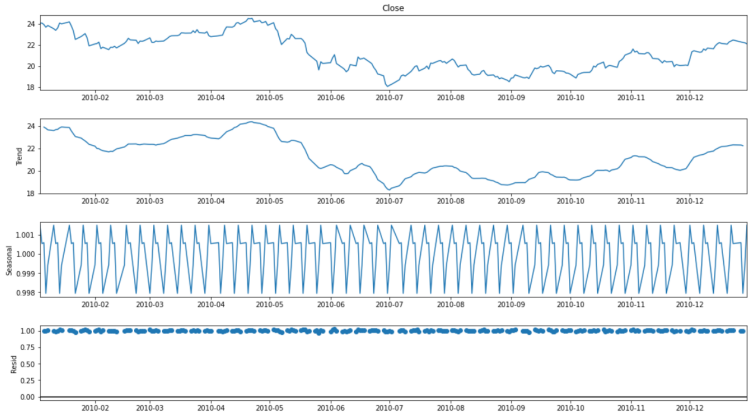
Fig. 2. A GCA matrix among the columns of dataframe

Inferences from above matrix are:
(i) Positive tweets, Negative tweets, Return causes Close prices.
(ii) Message-volume causes Positive tweets and Negative tweets.
(iii) Positive tweets, negative tweets causes Return.

G. Prediction and Forecasting

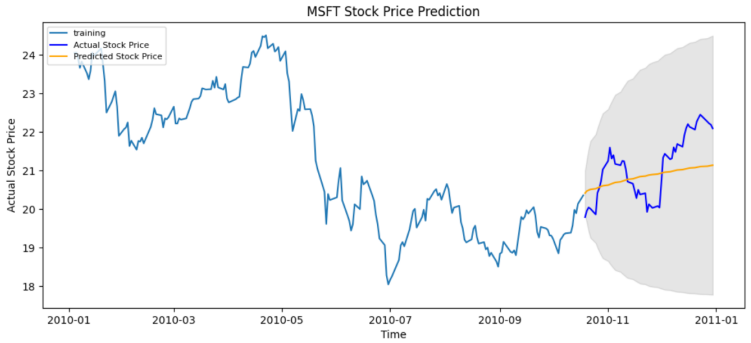
Now, after knowing which columns to drop and which to keep, we create a final data and will forecast the future prices of stocks based on that columns.
Initially using Auto Regressive Integrated Moving Average (ARIMA), we have just forecasted the closing prices based on just the past closing prices of the stock index.
The forecast that we have made is just showing the overall trend of the close prices, it doesn't include the short term changes in the closing price of the stock which are called 'seasonal changes'.
For clearer idea, see the below figure which shows the original time series, its long term trend, and short term seasonal trend.

Fig. 3. Trends and seasonal trend decomposed with window size of 1 week



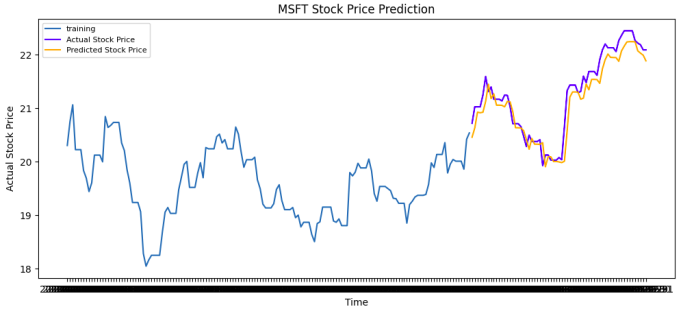
Then, we split our data into training and testing with 80/20 ratio. Then, for ARIMA model to use, we will need optimal values of p, d and q where: p is the number of autoregressive terms, d is the number of nonseasonal differences needed for stationarity, and q is the number of lagged forecast errors in the prediction equation.
And after finding optimal values from auto-arima function in python, we use that values to define the ARIMA model and forecast it on the test data which produced the plot given below (without twitter features). Here, you can see a grey part attached figure. which consist two lines one UCL and second LCL which is basically, upper confidence Level Line and Lower confidence Level Line.

Fig. 4. Univariate time series forecasting for 51 days



Now, we even have tried predicting close prices time-series from its other dependent features like positive, negative and previous day's close prices using OLS method for Linear Regression.
But as one can see in the below image, due to training on less data points the plot that is produced is over-fitting (with tweet features).

Fig. 5. Close price prediction using OLS regression for 51 days



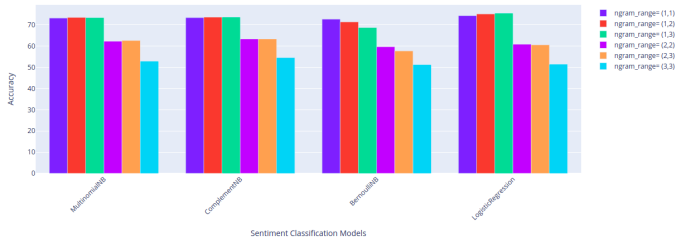
IV. Results

Results related to the cleaning and preprocessing of the raw tweets. In the below figure you can see that there are many punctuation marks, stopwords (ex. is, to), spelling mistakes, and hashtags. After pre-processing, we can see that punctuation marks are removed. Stopwords, like is and to, are removed. Hashtags are also removed and also spell correction is also done.

```
Raw Tweet: Today, Microsoft is going to craseh #MSFT
Preproccessed Tweet: Today Microsoft going crash
```

To find the best model for classification we have compared different naive Bayesian classifiers and logistic regression with different ngram ranges. The below figure shows the comparison between different models.

Fig. 6. Comparison of different classification models with different N-gram ranges



As we can see that the multinomial naive bayesian and logistic regression gives the pretty much same accuracy so we have selected naive Bayesian. Now if we compare the ngram range we can see that for a combination of monogram and bigrams accuracy is maximized. so we have selected ngram range=(1,2) in further work.

Below are the figures which shows the Root Mean Squared Error (RMSE) between the ground-truth values and the predicted/forecast values, which shows that the forecasting with the twitter features works better than without the twitter features.

Fig. 7. RMSE for ARIMA model for univariate Close price time-series without Tweet feature

```
[RMSE] Root Mean Squared Error = 0.7198362386270959
```

Fig. 8. RMSE for OLS regression model for multivariate time-series data with tweet features

```
[RMSE] Root Mean Square Error: 0.05589899310252812
```

V. Conclusions

Using just the univariate time series, we get the rough idea of overall trend that close price will be following throughout the forecast time.

To obtain the values of close price from different time series like positive tweets, negative tweets and return we have used OLS regression which gives much better results with more data. The problem that occurred with us was, we were having very less data points.

Twitter sentiments do play a significant role in deciding the stock market prices and maybe other features too.

VI. References

1) T. Rao and S. Srivastava, “Analyzing stock market movements using twitter sentiment analysis”
2) A.Jain, P. Dandannavar, “Application of Machine Learning Techniques to Sentiment Analysis”
3) Time series Forecasting using Granger’s Causality and Vector Autoregressive Model
4) T. P. Souza,Olga Kolchyna,Philip Treleaven and Tomaso Aste,”Twitter Sentiment Analysis Applied to Finance: A Case Study in the Retail Industry”,July 2015
5) Feroz Kazi,”Using Granger Causality Test to Know If One Time Series Is Impacting in Predicting Another?”,Jul 2020
6) Aishwarya Singh,”A Gentle Introduction to Handling a Non-Stationary Time Series in Python”,Sep 2018
7) Aishwarya Singh,”Stock Market Price Trend Prediction Using Time Series Forecasting”,Nov 2020
8) Jason Brownlee,”How to Create an ARIMA Model for Time Series Forecasting in Python”,Dec 2020
9) Jason Brownlee,”How to Check if Time Series Data is Stationary with Python”,Aug 2020
10) Robert Nau,”Statistical forecasting:notes on regression and time series analysis”