STOCK PREDICTION

A stock prediction model approach with tweets **sentiment analysis**

Giulio Mantoan, Giacomo Miolo, Mireia Riviere, Paolo Ticozzi
Professor: Marco Brambilla and Danilo Ardagna
Scalable Data Storage and Processing
Business Analytics and Big Data

Contents

Stock Market Prediction

Opinions and intuitions plan an important part in forecasting stock prices, and even the best mathematical models are affected by uncertainty.

Objective To predict the price movements from a company's traded in the stock market more accurately by introducing a variable related to the sentiment analysis of tweets.

Solution Taking advantage of the unprecedented high levels of use of social media. Introducing a new variable into the predictive historical data model based on social media, what seems a suitable solution to reduce market stock prediction accuracy.

How?

Collect past tweets, process them in a sentiment analysis, by indexing people's thoughts and ideas about a company and the stock. Apply a SARIMAX model, a predictive technique for time series forecasting with exogenous variables. The results obtained will be compared with different models.

Overall, the ultimate goal of this project is to create a model that together with traditional techniques and by introducing the outcome of the sentiment analysis on a set of tweets, can forecast more accurately how the market will behave.

Twitter

Tweets can provide a satisfactory reflection of public sentiment when taken in aggregate. With 500 millions of tweets generated on a daily basis on average[i] seems the best social media in which to base the Report's sentiment analysis, altogether with Twitter's Search API which include the exact time at which each tweet was sent, seems the best choice in which to make a sentiment analysis.

In addition to the huge amount of information that is daily created on twitter, according to Gales Thomas Panger the social media sentiment analysis is correlated with people's general emotional state[ii].

Sentiment Analysis

But why including a sentiment analysis into a Stock Analysis prediction? Building on the ideas of several behavioural economists which stated that decisions, even financials, are impacted by emotion and not just value[iii], the general opinion regarding a company traded in a stock market must have an impact into the markets, what makes it useful to include it in the prediction algorithm.

Report Reach This model will be tested on the company Tesla and its trading stocks. But this project's aim is to be applicable for every company with traded stocks, therefore this test will be done to test the extent to which it can be applicable and its well-functioning and accuracy level. A secondary aim of this report is to provide the same company with insights about how the company is perceived, being able to retrieve specific insights on time and location.

Methodology

Architecture

mongoDB

https://github.com/giacomomiolo/nosql-spark-scalable/blob/master/MongoDB.ipynb

APACHE Spark

https://github.com/giacomomiolo/nosql-spark-scalable/blob/master/Pyspark.ipynb

Dataset

In order to do this report structured and unstructured data had to be analyzed and brought together.

Stock Prices – are retrieved from Yahoo! Finance API.

Tweets – are retrieved from Archive.org and Twitter API. The tweets are in a .json format.

How these Data has been processed will be specified in the Body of the report.

Tweets Processing

MongoDB pipelines used

Queries are divided into 2 groups:

Find: selects documents in a collection (or view) and returns a cursor to the selected docs. It requires two main parameters:

<query>: Specifies selection filters using query operators
<projection>: Specifies the fields to be returned

Aggregation: it requires different stages in an array. Documents pass through all the stages in sequence. Stages:

addFields: add a new field into the document(s) selected
count: returns a count of the number of docs at this stage of aggregation pipeline
group: groups input docs by a specified identifier expression
indexStats: returns statistics regarding the use of each index for the collection
limit: passes the first n docs unmodified to the pipeline where n in the specified limit
project: reshapes each doc in the stream, such as by adding new fields or removing existing fields. For each input, it outputs one doc.
sample: randomly selects the specified number of docs
set: add new fields into the document
sort: reorders the document stream based on specified sort key

Data Source The tweets are in a .json format and are rich of information, containing over 150 attributes, however the ones considered are about the id of the user, the user name, the timestamp, the tweet content, the geolocation and the hashtags used.

.Json File which stores simple data structures and objects in JavaScript Object Notation format. It is primarily used for transmitting data between a web application and a server since they are lightweight files, text-based, human-readable and can be edited using a simple text editor. The main reason why the analysis was done on a text based storing format is that it's generated daily, and that Natural Language Processing is an optimal way to analyse people feelings towards a specific subject and which needs a document-based file to be applied to.

Furthermore, it can be interesting for a future development and inclusion of more data sources as it is well-supported by many different programming APIs (Application Programming Interface), which allows communication between different applications.

Structure It is classified as a NoSQL databased, because it doesn't require a predefined structure such as relational data models (SQL). It can include:

- Numbers: includes E notation for exponential, not NaN formats. No distinction between integer and floating point.
- Strings: sequence of zero or more Unicode characters, divided by double-quotation marks and support escaping syntax.
- Boolean Values: true/false.
- Arrays: ordered list of zero or more values.
- Objects: unordered collection of name-value pairs.
- NULL: empty value.

The format of this json was all the same: included the Created at feature with the name of the day, month, number, time, time zone, and year. Also contained the users' id, the text itself (which may include RT or URL), plus some Boolean values (checks of it it's truncated, a reply, etc). It also contained more users' information: such as name and location, its number of followers, etc.

In order to process this document-based database **MongoDB** which provides high scalability and flexibility and with the querying and indexing needed was used. It stores data JSON-like, meaning that the fields in each document can vary from document to document, and data structure can be changed over time.

It is characterized by Scalability, high availability through multiple Replica sets, and it supports both MapReduce and Aggregation Framework for

aggregation tasks. The second one is quite useful because it is implemented in C++ and it avoids wastes of resources and time.

MongoDB tasks for queries need to be always referred to a specific collection of data contained in specific database.

Due to the high volume of information and due that it was processed on a local MongoDB, the research is done on the month of June's tweets. In order to process more information solutions like Mongo DB atlas (on Amazon's AWS) must be implemented.

Scalability & Spark

Itinerating through 151 millions of tweets occupying 449,5 Gib (30 days of the month of June 2019)

In order to cope with Big Data, we implemented a fully functioning solution relying on Amazon Web Services.

We used Amazon S3 (Simple Storage Service) for distributed storage, Amazon EMR (Elastic MapReduce) for Hadoop framework running Spark and Amazon EC2 (Elastic Compute Cloud).

The details of the implementation can be found in the following Jupyter Notebook: https://github.com/giacomomiolo/nosql-spark-scalable/blob/master/AWS-Solutions.ipynb

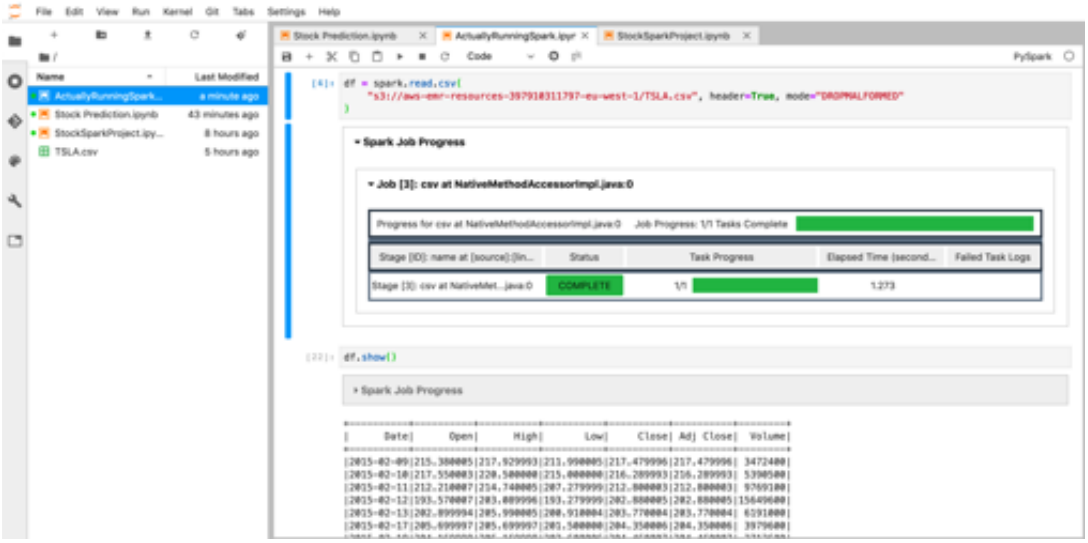


Figure 1 - Spark Job running on EMR:

Sentiment Analysis

In order to provide a more accurate result than Lexicon-based methods, a machine Learning-based method was implemented: TextBlob. In order to implement NLP tasks TextBlob library can be found in Python and is characterized by a simple interface compared to other more advanced models. All the functions done on the text based document, .json, was done with TextBlob and NLP tasks.

Selecting the tweets with the keywords – The text of the tweets goes through another selection process, in which only the ones that contain words company related. For example, in order to proceed to the sentiment analysis, the tweets related to Testla's sentiment polarity where the ones that mentioned at least once "Tesla" and other company related key words.

Results

The final results returned are a polarity in the form of a float from a range of [-1.0, 1.0]

https://github.com/giacomomiolo/nosql-spark-scalable/blob/master/SentimentAnalysis.ipynb

Stock Data Processing

The Data obtained from Yahoo! Finance is obtained in a CSV format with a daily frequency showing historical prices. The variables included in the file are: the date, opening prices, the highest, lowest, closing prices, the adjusted closing prices and the volume.

The main problem on the Stock Market data is that the market is not always open: has closing and opening times, on weekends it's closed and also on blank holidays. Versus the twitter (people sentiment's) which is produced on a continuously daily basis. What it should be dealt in a future analysis.

Stock Regression Model

Since the object of study are the stock prices (Adjusted Close) related only to one month, the data points are quite few and then we chose to set the proportion between training and test set equal to 0.66 and 0.33 respectively. The forecast has been made through different models.

A time series is usually modelled through a stochastic process Y(t), i.e. a sequence of random variables.

Due to that in a forecasting setting we find ourselves at time t and we are interested in estimating Y(t+h), using only information available at time t. In this case, we are interested in estimating Y(t+1). Due to the temporal dependencies in time series data, this Stock forecasting cannot rely on usual validation techniques. To avoid biased evaluations we must ensure that training sets contains observations that occurred prior to the ones in validation sets.

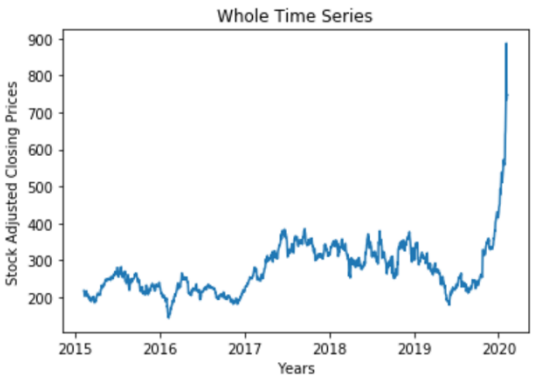


Figure 2 - Original Series

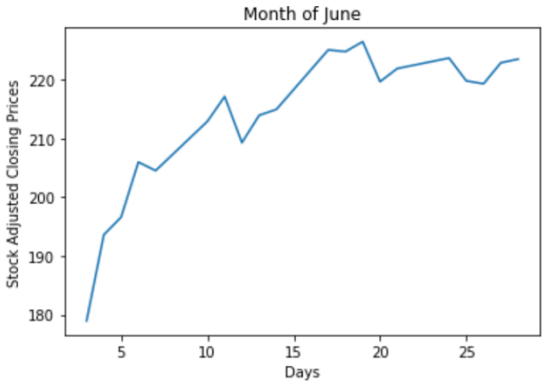
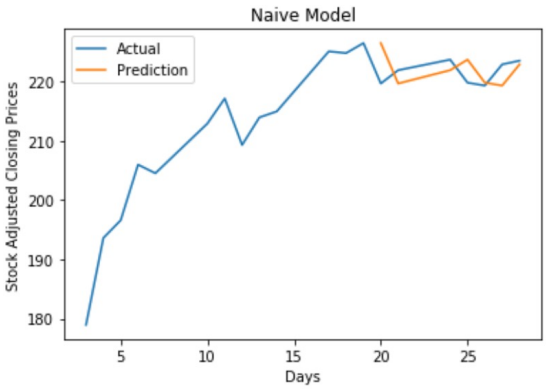


Figure 3 - Original June month

NAÏVE model – In the Naïve model, the forecasts for every horizon correspond to the last observed value.

$$\hat{y}(t+1 \mid t) = y(t)$$

This kind of forecast assumes that the stochastic model generating the time series is a random walk. This model has been used as a benchmark model.



Results:
Test MSE: 11.855 (unit of measure^2)
Test MAPE: 1.254%

ARIMA models are among the most widely used approaches for time series forecasting. The name is an acronym for Auto Regressive Integrated Moving Average.

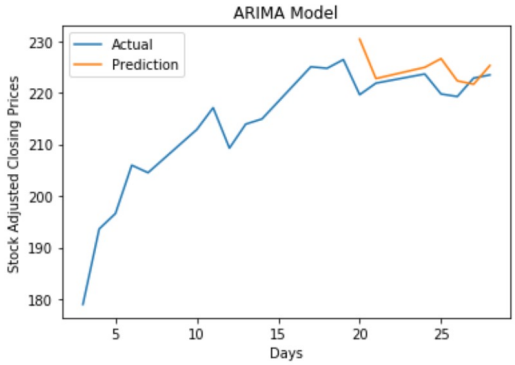
$$\Delta^D \hat{y}_t = \sum_{i=1}^{p} \varphi_i \Delta^D y_{t-i} + \sum_{j=1}^{q} \theta_i \varepsilon_{t-j}$$

In an Auto Regressive model, the forecasts correspond to a linear combination of past values of the variable ($\sum_{i=1}^{p} \varphi_i \Delta^D y_{t-i}$, where p is the number of autoregressive lags). In a Moving Average model, the forecasts correspond to a linear combination of past forecast errors ($\sum_{j=1}^{q} \theta_i \varepsilon_{t-j}$, where q is the number of moving average lags).

Basically, the ARIMA models combine these two approaches. Since they require the time series to be stationary, differencing the time series may be a necessary step, i.e. considering the time series of the differences instead of the original one ($\Delta^D y_t$, where D is the degree of differencing).

In order to get the optimal set of orders, a grid search approach has been adopted. The range of values for the parameters is:

- p = [0, 1, 2]
- d = [0, 1]
- q = [0, 1, 2]
-



Results:

Best orders set: (p,d,q) = (1,0,0)
Test MSE: 11.636 (unit of measure^2)
Test MAPE: 1.142%

SARIMA model (Seasonal ARIMA) extends the ARIMA by adding a linear combination of seasonal past values and/or forecast errors. It is written as follows:

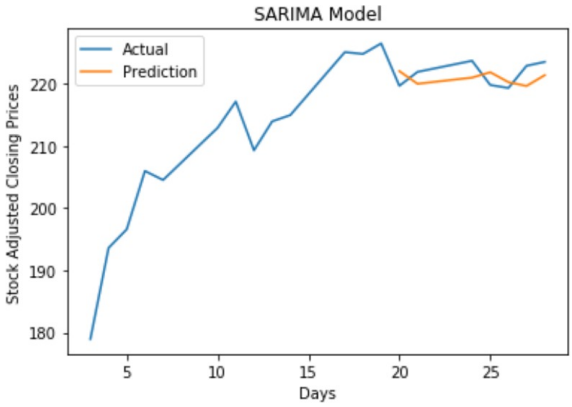
$$\text{ARIMA}\ \underbrace{(p,d,q)}_{\text{Non-seasonal part of the model}}\ \underbrace{(P,D,Q)_m}_{\text{Seasonal part of of the model}}$$

Where m is the number of observations per "season".

In order to get the optimal set of orders, a grid search approach has been adopted. The range of values for the parameters is:

- p = [0, 1, 2]
- d = [0, 1]
- q = [0, 1, 2]
- P = [0, 1, 2]
- D = [0, 1]
- Q = [0, 1, 2]
- m=[0,5]

There is another parameter that can be added, which indicates if there is a trend in the time series and of what kind. Values range is t = ['n','c','t','ct']; 'n' indicate no trend, 'c' indicates a constant (i.e. a degree zero component of the trend polynomial), 't' indicates a linear trend with time, and 'ct' is both. This parameter has been chosen with a grid search approach too.



Results:
Best orders set: (p,d,q)=(1, 0, 1); (P,D,Q)=(0, 0, 1,); m= 0, t='c']
Test MSE: 5.229 (unit of measure^2)
Test MAPE: 0.987 %

ARIMAX & SARIMAX models can be furtherly enriched by giving an exogenous input as well, which, in this case, is the result of the sentiment analysis made on the tweets. The forecasts are enriched with a linear combination of the sentiment.

Conclusions

Future Applications

The stock price prediction problem is complex and requires the integration of multiple and more complex algorithms. A "simple" SARIMAX alone won't be useful in a real-world application of the model.

The sentiment analysis is done with a relatively naïve approach, using the TextBlob NLP python library. Refining the sentiment value obtained from the tweets can be done in a future implementation using SpaCy open-source library and more advanced natural language processing techniques.

Due to budged limitations the model we proposed couldn't be run on the implemented AWS solution, mostly due to the storage cost of the massive Tweet dataset.

A future implementation could include a live stream of tweets from the Twitter API, a live stream of financial data from a financial data provider and return a forecast is almost real-time.

With this project we focused more on the technical implementation of the infrastructure needed to process Big Data rather than on the quality of the models applied to the time series.

Appendix

[i] Internet Live Stats (retrieved from: https://www.internetlivestats.com/twitter-statistics/). Last visit: 06/02/2020

[ii] Kahneman, D., Tversky, A.: Prospect theory: An analysis of decision under risk. Econometrica (1947)

[iii] Panger, G.T.: Emotion in Social Media. PhD thesis, University of California, Berkeley (2017)