# Machine Learning Engineer Nanodegree

Capstone Project Report

Egar Garcia

April 2nd, 2019

# Stock Price Predictor

## I. Definition

### Project Overview

A stock market, also called equity market or share market, is a network of economic transactions, where the stocks of public companies can be bought and sold. The equity market offers companies the ability to access capital in exchange of a portion in the ownership of the company for interested outside parties.

In the stock market, other financial securities like exchange traded funds (ETF), corporate bonds and derivatives based on stocks, commodities, currencies, bonds, etc. can be traded. However, for purpose of this project only the exchange of stocks will be considered.

It is common to use stock market and stock exchange interchangeably, but the stock market is a general superset of the stock exchange. If someone is trading in the stock market, it means that it buys and sells stock/shares/equity on one (or more) of the stock exchange(s) that are part of the overall stock market.

The stock market offers an opportunity to investors to increase their income without the high risk of entering into their own businesses with high overheads and startup costs. On the other hand, selling stocks helps companies themselves to expand exponentially, when a company's shares are purchased it is generally associated with the increased in the company's worth. Therefore, trading on the stock market can be a win-win for both investor and owner.

The stock exchange that are leading the U.S. include the New York Stock Exchange (NYSE), Nasdaq, BATS and Chicago Board Options Exchange (CBOE). The Dow Jones Industrial Average (DJIA) is a price-weighted average of 30 significant stocks traded on the NYSE and the Nasdaq, it is the most closely watched market indicator in the world and it is generally perceived to mirror the state of American economy.

Projecting how the stock market will perform is a very difficult thing to do, there are so many factors involved in the prediction some of them emotional or irrational, which combined with the prices volatility make difficult to predict with a high degree of accuracy. Abundant information is available in the form of historical stock prices, which make this problem suitable for the use of machine learning algorithms.

A machine learning model is fed with historical stock price records, these are used to perform a training process on the model after which the model would be ready to perform predictions. There are multiple options to obtain historical records, a popular one (and the one used in this project) is an API provided by IEX Group Inc. which offers free access to historical stock price records up to five years old.

Investment firms, hedge funds and individuals have been using financial models to better understand the market behavior and attempt to make projections in order to make profitable investments and trades.

### Problem Statement

The purpose of this project is to build a stock price predictor, more specifically, the problem is to predict the closing price of a given company's stock in the trading days existing in a queried date range. For the scope of this project only the companies included in the Dow Jones Industrial Average are considered.

To address the problem a supervised learning approach is taken, the strategy to follow is to approach the problem as a particular case of forecasting in time series. As in supervised learning, a generalized function is tried to be inferred from the existing data which already contains the ground truth, the difference with this approach is that the existing data is in the past and the data to predict in the future, i.e. there is a clear separation of the predictor's values used for training and prediction, instead of being mixed and distributed along the possible set of values.

The different machine learning methods used in this project take historical stock data for a particular company over a certain date range (in the past) as training input, and outputs projected estimates for a given queried date range (in the future). The following ones are the methods taken in this project to address the problem of making predictions about how the closing stock prices will perform:

- ARIMA (AutoRegressive Integrated Moving Average): It is a very popular statistical method for time series forecasting that takes into account the past values to predict the future values. [1]
- Prophet: It is a time series forecasting library designed and pioneered by Facebook, that is claimed to be extremely simple to implement. [2] [3]
- LSTM (Long Short-Term Memory): It is a deep learning approach based in Recurrent Neural Networks (RNN) also used to make predictions in time series. [4] [5]

### Metrics

To measure the performance of the predictions, the predicted value needs to be compared against the real value in the test and/or validation dataset, the predicted value is a (floating point) numerical value, thus using pure accuracy would not be convenient and not provide a useful description about how well the prediction is performing or improving through training iterations.

Instead, a metric that can provide some sort of average of the total error would be more effective, since the purpose of the prediction is to get a value as close as possible to the real one, but it does not need to be the same to be useful. For this reason the Root Mean Square Error (RMSE) is going to be used as evaluation metric for this project.

The formula to calculate the Root Mean Square Error is the following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}$$

Where $N$ is the number of data points, $y_i$ is the observed value or ground truth for the datapoint $i$ and $\hat{y}_i$ is the predicted value for the data point $i$.

## II. Analysis

### Data Exploration

The original proposal contemplated to get the data from an open dataset called "EOD data for all Dow Jones stocks" in Kaggle, which contained the historical data for Dow Jones stocks, however for some reason this dataset is not longer available. Then the alternative and solution used in this project is to extract the data from the original source, which is an API provided by IEX Group Inc. (https://iextrading.com), it provides access to stock historical records to developers and engineers for free.

The API provided by IEX (which documentation can be found at https://iextrading.com/developer/docs/#chart) allows to retrieve historical stock price information for a maximum of 5 years back to the current date. The API can provide historical records for several companies, but for this project only the records for the ones in the Dow Jones are retrieved. An important aspect to notice is that the market is closed on weekends and some defined holidays.

The API retrieves the data in the in JSON format, which at the end contains records, where each record corresponds to the information of a trading day for a specific company, a company is identified by a ticker symbol (also known as stock symbol). For example to retrieve the historical stock prices for the ticker symbol MSFT (Microsoft Corporation) of the last 5 years, the call to the API would be https://api.iextrading.com/1.0/stock/msft/chart/5y

The historic stock price records contain the following columns:

- `date`: Trading day
- `open`: Opening price
- `high`: Highest price
- `low`: Lower price
- `close`: Closing price
- `volume`: Number of shares traded
- `unadjustedVolume`: Number of shares traded also considering companies adjustments
- `change`: Change of closing price relative to the day before
- `changePercent`: Change in percent value
- `vwap`: Volume Weighted Average Price
- `label`: Formatted version of the date
- `changeOverTime`: Metric considered to measure the changes bases on weighted dates

The following is a fragment of how the historical stock looks for the thicker symbol APPL (Apple Inc.):

| date | open | high | low | close | volume | change | changePercent | vwap |
|---|---|---|---|---|---|---|---|---|
| 2014-02-21 | 69.9727 | 70.2061 | 68.8967 | 68.9821 | 69757247 | -0.774858 | -1.111 | 69.4256 |
| 2014-02-24 | 68.7063 | 69.5954 | 68.6104 | 69.2841 | 72364950 | 0.302061 | 0.438 | 69.1567 |
| 2014-02-25 | 69.5245 | 69.5488 | 68.4239 | 68.5631 | 58247350 | -0.72101 | -1.041 | 68.9153 |
| 2014-02-26 | 68.7667 | 68.9492 | 67.7147 | 67.9446 | 69131286 | -0.618575 | -0.902 | 68.1373 |
| 2014-02-27 | 67.917 | 69.4457 | 67.7738 | 69.2999 | 75557321 | 1.3553 | 1.995 | 68.8615 |
| 2014-02-28 | 69.4851 | 69.9671 | 68.571 | 69.1121 | 93074653 | -0.187807 | -0.271 | 69.2731 |
| 2014-03-03 | 68.7417 | 69.6913 | 68.6616 | 69.3117 | 59667923 | 0.199626 | 0.289 | 69.1371 |

Per year there are from 251 to 253 trading days, since the market is closed on weekends and some holidays:

- 2014 had 252 trading days
- 2015 had 252 trading days
- 2016 had 252 trading days
- 2017 had 251 trading days
- 2018 had 251 trading days
- 2019 is expected to have 252 trading days
- 2020 is expected to have 253 trading days

That means that each company in the Dow Jones would have from 251 to 253 historical records per year, except for 'DWDP' (DowDuPont Inc.) since its records started to appear on 2017-09-01.

In a 5 years period from 2014-03-24 to 2019-03-23 (which was the one used during the development), there were 1259 historical records per company, with the exception of 'DWPD' which had 390 historical records. The historical records of the actual Dow Jones Industrial Average are also included in the dataset which would add another 1259 records. Then for this 5 year period there were 38160 historical stock price records in total ($1259 \times 29 + 390 + 1259 = 38160$).

The purpose of this project is to predict the future values of the closing price, which corresponds to the column `close`. To make predictions is necessary to chose a subset of columns which values can be known a priory and used for the prediction, unfortunately all the column values except date and label (which finally is a variant of the date) are unknown before the occurrence of the respective trading days. Then the only data that can be used to predict the future closing prices is the past closing prices and the company itself.

### Exploratory Visualization

In figure 1 it is presented a visualization of the historical closing prices for the 30 stocks of the companies in the Dow Jones, displaying the five years prior to March 23rd, 2019. The top image displays the prices along the time for all the companies, the bottom image is a comparison with the actual Dow Jones Industrial Average index (highlighted in black).

The visualization in figure 2 displays the historical prices of the stocks, but this time grouped by industry type, they are also contrasted with the Dow Jones Industrial Average index (identified with the symbol DIA and plotted in a dotted black line).

The Dow Jones Industrial Average is calculated with the stock prices of 30 selected public large companies, then it is not surprising that most of these stocks are behaving in a similar way to the DJIA. To calculate the DJIA, the prices are added and then divided by the Dow divisor, which is constantly modified.

In figure 3 is presented a visualization that shows the correlation of each one of the stocks in the Dow Jones against each other and against the actual DJIA. It can be observed that in most of the cases there is a high correlation, observed by a dominance of the red color in the matrix (which is the color to indicate a high correlation, i.e. close to 1), only 4 companies seem no to follow the same tendency as the the general DJIA.

Looking at the stocks one at a time, it looks like the stock prices fluctuates a lot and a clear pattern is not perceived, at least not at a human comprehensible level. This is understandable since the stock prices depend on a lot of different factors like the company's financial health, economic supply-demand and even involving human emotions like trust, euphoria or panic.

At a macro level it can be observed that they are common events that seem to affect the stock prices as a whole, like a rise and sudden fall of prices around the beginning of 2018, or a generalized drop on the stock prices at the end of 2018. However these kind of events also do not seem to have a (humanly) comprehensible pattern either.

### Algorithms and Techniques

A normal machine learning dataset is a collection of observations where time does not necessarily play a main role, predictions are made for new/unknown data, that might be considered as predicting the future, however all the prior observations are pretty much treated equally, and the order or the observations is not taken in consideration, even is a good practice to shuffle the observations to perform the training.

A time series dataset is different, because they have an explicit dependence of the order between observations, thus the time plays a main role. For these datasets the time is both a constraint and also a structure that provides additional information.

During the exploration of the dataset it was realized that the only data that we can know a priory and use to make the predictions are the dates, then this characteristic makes the problem to forecast the stock closing prices to be a time series forecasting problem.

There are some known methods to address the problem of forecasting time series, the ones that are are used during the implementation of this projects are described bellow.

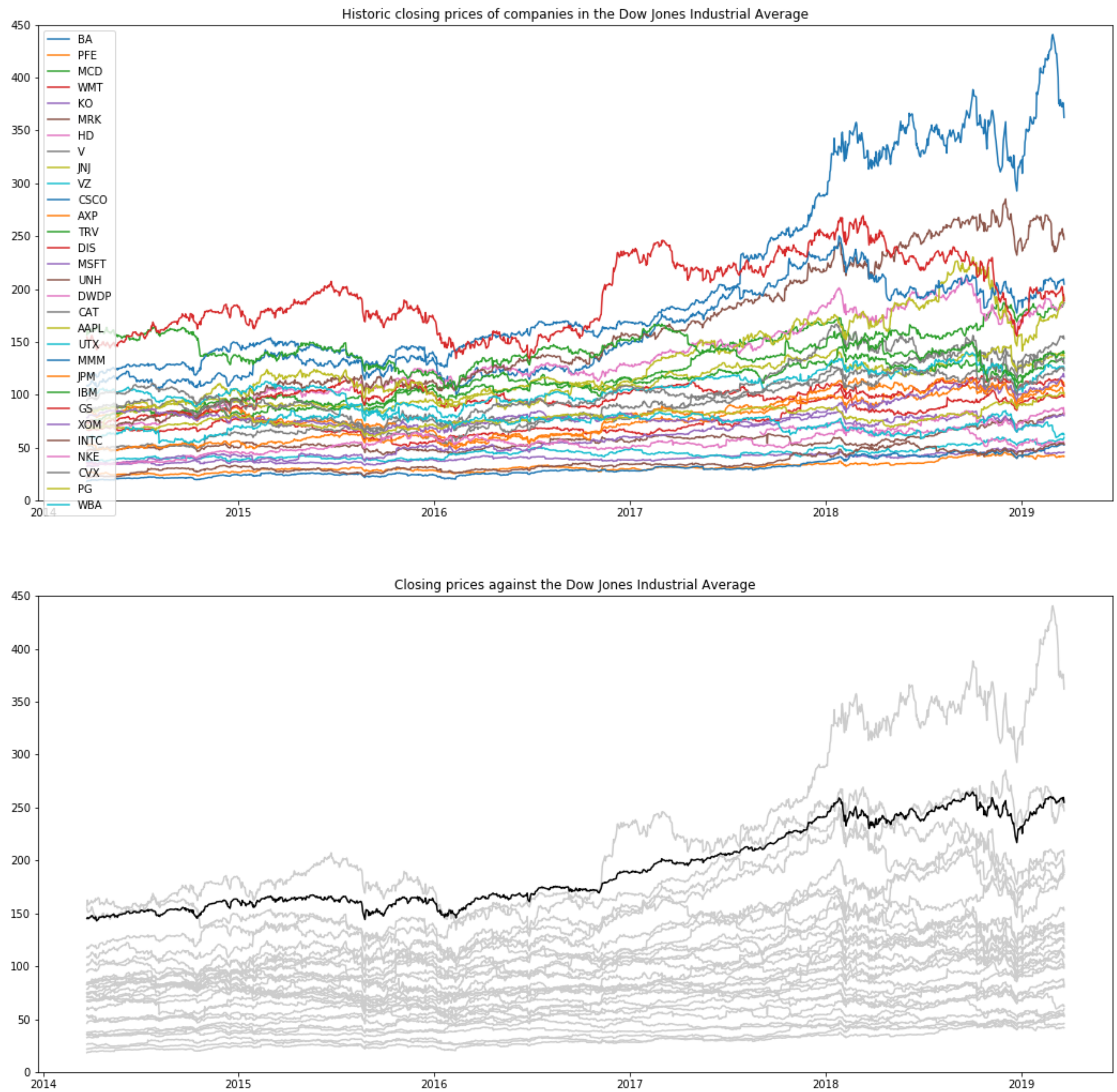


Figure 1: Dow Jones 5 year historic prices

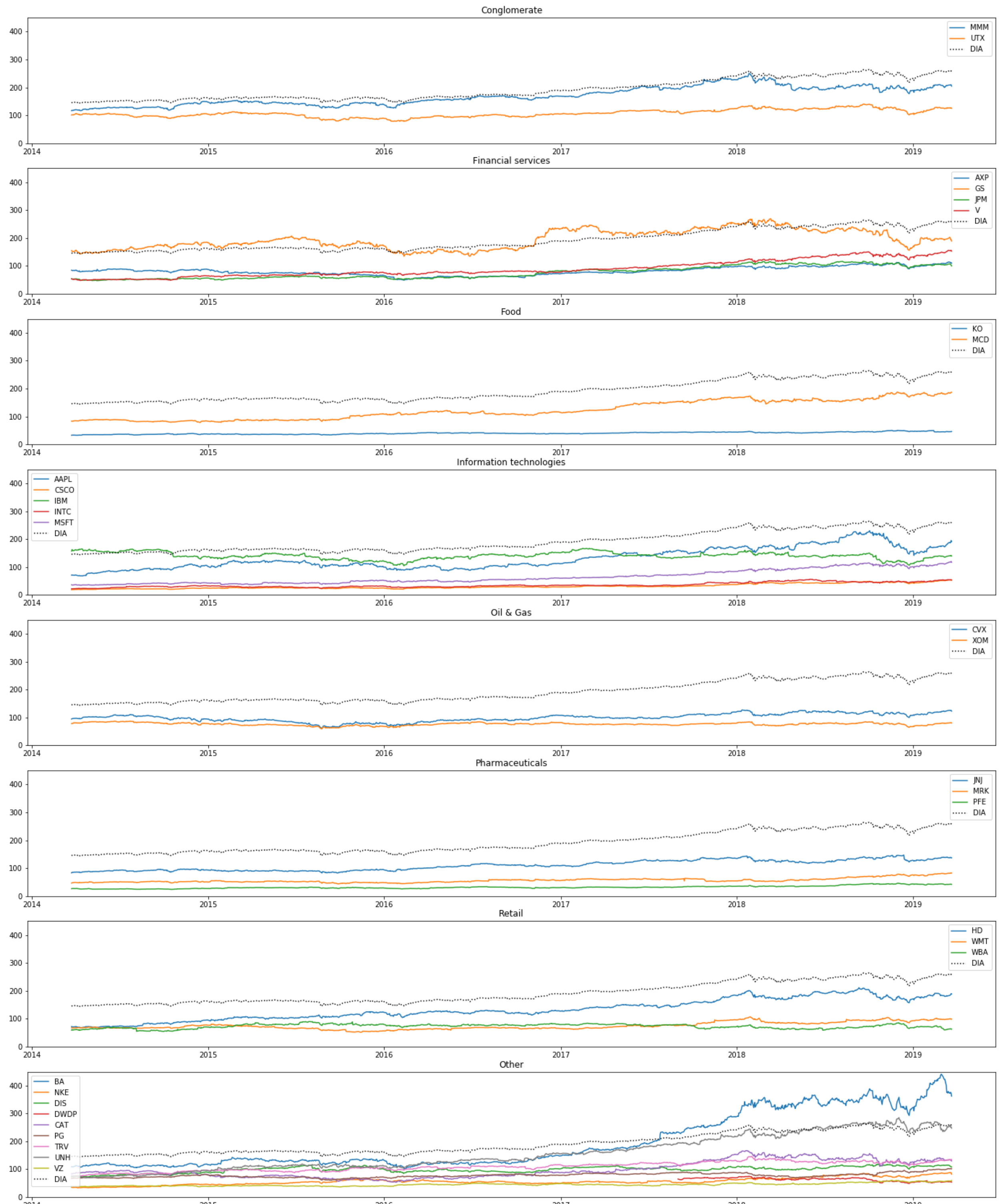


Figure 2: Dow Jones 5 year historic prices per industry

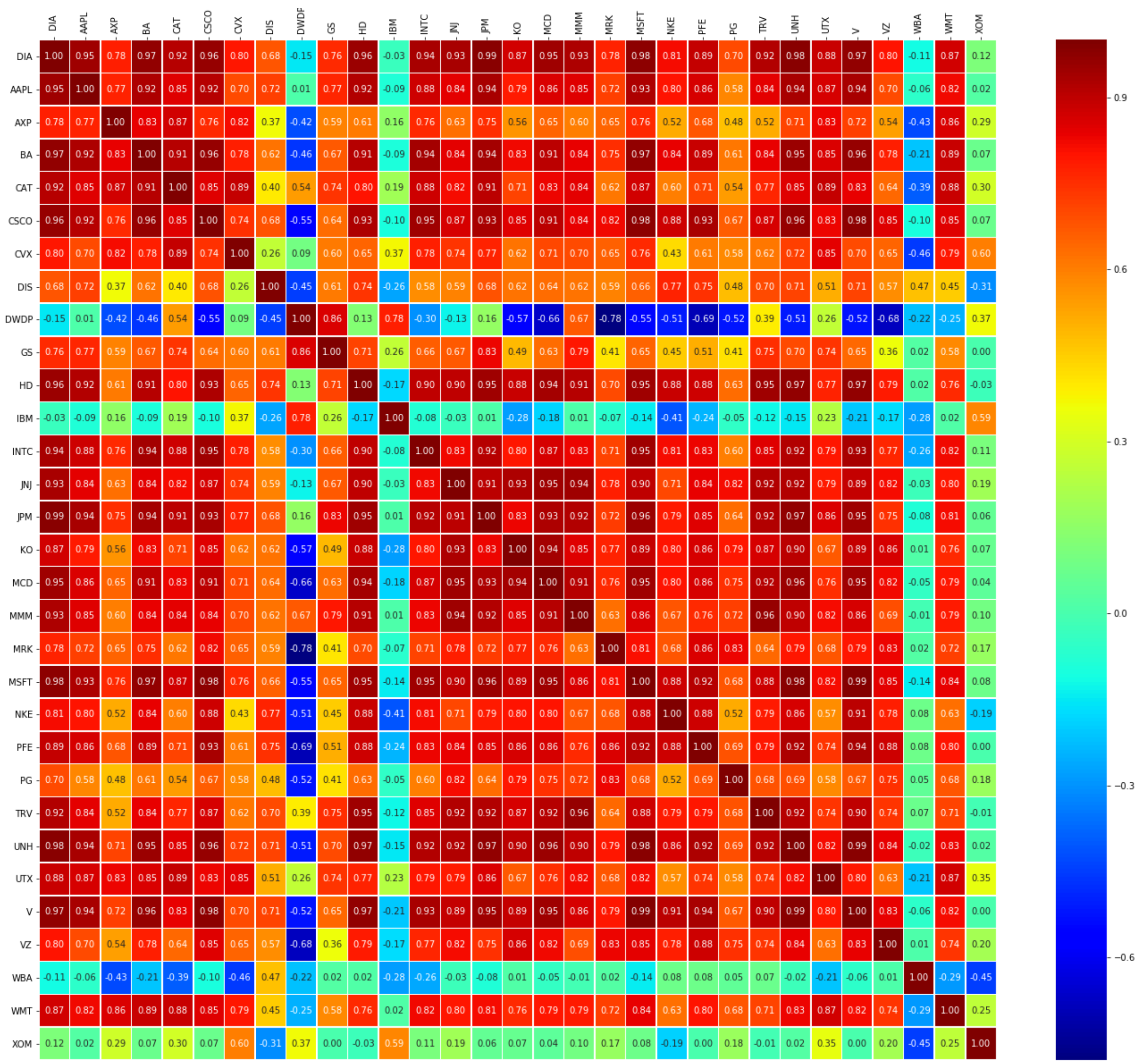


Figure 3: Correlation among Dow Jones stocks

### 1. Linear Regression

Linear regression is the first and naive Machine Learning method to implement, the idea is just to obtain the regression line for the closing prices against the dates, this is the one that is going to be used to benchmark the other methods during the evaluation.

A variant of linear regression is also going to be explored for this project, the idea is that that the date components: year, month, day, week, day-of-week and day-of-year, might play a role in the determination of the closing price of a stock, then linear regression is applied using these components as predictors.

### 2. ARIMA (AutoRegressive Integrated Moving Average)

It is a very popular statistical method for time series analysis and forecasting, its acronym is descriptive, capturing the key aspects of the model itself [6]:

- AR (Autoregression): A model that uses the dependent relationship between an observation and some number of lagged observations.
- I (Integrated). The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- MA (Moving Average): A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

### 3. Prophet

It is an open source forecasting tool developed by Facebook, it is optimized for the business forecast tasks encountered at Facebook. They claim that the default settings produce forecasts that are often as accurate as those produced by skilled forecasters, with much less effort. [2]

### 4. LSTM (Long Short-Term Memory):

A Recurrent Neural Network (RNN) can be thought of as multiple copies of the same network, each passing a message to a successor, aiming for them to learn from the past [7]. RNNs are good in handling sequential data but they have two main problems, the first one called "Vanishing/Exploding Gradient problem" presented as a result of the weights being repeated several times, and the second one called "Long-Term Dependencies problem" that happens when the context is far away [8].

Long Short-Term Memory networks are a special kind of RNNs (introduced by Hochreiter & Schmidhuber in 1997 [9]) with capability of handling Long-Term dependencies and also provide a solution to the Vanishing/Exploding Gradient problem [8]. They are currently used to address difficult sequence problems in machine learning and achieve state-of-the-art results [10].

### Benchmark

To test the prediction's accuracy/performance for the machine learning methods used in this project, a historical test dataset for a ticker symbol up to a given date is taken and used to perform the training. Then the prediction is performed and benchmarked over validation sets for the following 1, 5, 10, 20, 40, 60 and 120 trading days, which is almost equal to predict for the next trading day and then 1, 2, 4, 8, 12 and 24 weeks ahead, if disregarding the holidays in between.

To benchmark the performance of the different machine learning methods implemented in this project, they are compared against an initial naive solution which is produced via linear regression, where the predictor is the trading days' date, and the predicted value the closing price.

The results of the benchmarking can be expressed in terms of an improvement (or worsening) ratio against the linear regression. The ratio (expressed as well as percentage) of improvement is calculate with the following formula:

$$model\_improvement = (LinearRegression\_RMSE - Model\_RMSE)/LinearRegression\_RMSE$$

The model improvement ratio can be interpreted as follows:

- A value of 1 (100%) means that the model has no error, i.e. it was improved to perfection.
- A value of 0 (0%) means that the model has the same performance than the linear regression, i.e. it was not improvement or worsening.
- A positive value between 0 and 1 means that the model performs better than the liner regression, but still with some errors, the closer to 1 (100%) the smaller the errors in the predictions.
- A negative value means that the performance of the model is worst than the linear regression.

## III. Methodology

### Data Preprocessing

For this project different machine learning methods/models are used, and the requirements for the inputs they receive are diverse, of course for all of them the outcome is the same, i.e. the closing price of the stock for a particular trading day. It is also important to notice that all the methods perform the forecast for just one ticker symbol, then a common step is to filter the data set to get the records for the related ticker symbol (since the dataset contains the records for all the ticker symbols in the Dow Jones).

Each one of the methods/models addressed in this project requires its particular way to preprocess the data that is used in the training set, those are described bellow:

- Linear regression: The training set requires a numeric representation of the date as a predictor (because dates are not supported), the preprocessing mechanism to prepare the training set consists in selecting just the `date` and `close` columns, then for the date its timestamp is calculated and this value is used as a predictor instead of the actual date. The model is trained by using the timestamp as a predictor and the closing price as the outcome's ground truth.

- Linear regression using date components: In this approach to prepare the training set, also the `date` and `close` columns are selected, then for the date the components: year, month, day, week, day-of-week and day-of-year are calculated. The model is trained by using the these components as predictors, they are numerical values so the linear regression can support them, the closing price is used as the outcome's ground truth.

- ARIMA: This model only needs a sequence of ground truth consecutive values in the time series to do the training, then to prepare the training set only the column `close` needs to be selected, however an important aspect is that the order must be preserved.

- Prophet: This model receives the date as predictor and the ground truth of the value to predict as outcome, however their columns should be named `ds` and `y` respectively. Then to prepare the training set for this model, the `date` and `close` columns are selected and then renamed.

- LSTM: The preprocessing mechanism for this model is the most complex of all, because LSTM takes as predictors sequences of a given length (known as time-steps) containing consecutive values in a time series and the outcome is the next value in the time sequence, i.e. the predictor is a subsequence of the time series instead of a single value. To prepare the training set only the values in the column `close` are needed, the date is not necessary since the important factor is the order. The preprocessing for this method consists in creating the subsequences with the closing prices previous to the date of the respective outcome. Also as in most deep-learning approaches it's recommended that the values (to predict in this case) are normalized, then for this project the closing prices are scaled to the range from 0 to 1.

In terms of the prediction, the mechanism is to query by a date range, then the set of trading days in that range is calculated and gathered together in an array. Again, each particular of method/model requires its own way to preprocess the data to be able to perform the prediction, those are described bellow:

- Linear regression: The model requires as a numeric representation of the date to perform the prediction, the preprocessing mechanism to prepare the data for prediction consists on transforming the array of trading days to a data-frame with only one column `date`, then transforming the column `date` to a timestamp, which is the predictor used during training.
- Linear regression using date components: For this approach the preprocessing mechanism consist on transforming the array of trading days to a data-frame with only one column `date`, then the date is broken down in the components: year, month, day, week, day-of-week and day-of-year, which are the predictors used during training.
- ARIMA: For this model only the number of future points to predict is required, then the preprocessing mechanism only consist on getting the size of the array containing the trading days to predict.
- Prophet: Same as the previous one, this model only needs the number of future points to predict, then the preprocessing mechanism only consist on getting the size of the array containing the trading days to predict.
- LSTM: The preprocessing mechanism for the prediction set for this model is also complex, because LSTM takes as predictors sequences of a given length containing consecutive values in a time series and the predicted value would be the next value in the time sequence. The preparation/preprocessing of the prediction set consist in creating subsequences containing the closing prices previous to the date to predict, normalized to a range from 0 to 1, since these are the inputs that the LSTM network accept.

### Implementation

The implementation process it is documented in more detail and can be followed at https://github.com/egar-garcia/machine-learning-engineer-nanodegree-capstone/blob/master/StockPricePredictor-Project-Development.ipynb. The implementation stages are described bellow.

#### 1. Dataset management

An object oriented approach has been taken to create a class called `Dataset` which objects are intended to be used to manage the dataset by performing the following operations: * Creating a new dataset by retrieving the data from the source (which is the API provided by IEX Group Inc.) * Updating the data by retrieving the most recent information from the source. * Loading and saving the data to files. * Filtering the data by date range or ticker symbols.

Another class called `TradingDaysHelper` is created with the purpose of getting the existing trading days (the days that the stock market is open) in a date range, or to get a determined number of trading days after a given date (this is useful to perform validations for a number of days ahead). The market is closed on weekends and some holidays that can be loaded from a file (by default named `market_holidays.txt`). The use of objets of this class is very important since not all the dates in a date range are predictable, just the ones when the market is open.

#### 2. Data exploration and visualization

In this phase it was created the code in Python to generate the visualizations presented previously in the "Exploratory Visualization" section of this report. These visualizations aimed to plot the historical closing prices of the stocks, compare them with the actual Dow Jones Industrial Index and retrieve the correlation among the related companies' stocks.

Also another purpose was trying to identify patterns that could help to select some machine learning methods or to do some treatment to the data. Unfortunately not evident (at least by humans) patterns were observed, which lead to reinforce the decision of using techniques to forecast time series.

### 3. Implementation of the machine learning methods/Models

Again an object oriented approach was taken to represent each one of the machine learning methods/models as an object. A super class called `StockForecasterModel` represents a generic ML model which provides methods to do the training and perform predictions, as well as load or save the object (model) to a file.

A model's object is restricted to perform predictions for only one ticker symbol, this is because a time series only makes sense for the historical records of one particular ticker symbol.

To perform the training, a dataset is provided to get the historical records of the ticker symbol, and optionally a date range (given by a start and end date) can be provided to only consider the records that fall in that range for the training.

To do predictions a date range is given, then the purpose is to predict the closing prices in the trading days that exist in that range. The results are retrieved as a data-frame that contains the date of the trading days in the range with their respective predicted closing price.

For each-one of the methods/models implemented in this project a specific class is created to do the specifics for method, like preprocessing/preparation of the data used for training and prediction, instantiation and fitting the underlaying models, preparation of the results, etc. The models implemented in this project with their classes and underlying modules are listed bellow:

- **Linear Regression** implemented in the class `LinearRegressionStockForecaster`, uses the underlying module `sklearn.linear_model.LinearRegression`.
- **Linear regression using date components** implemented in the class which name is `DateComponentsLinearRegressionStockForecaster`, as the Linear regression it also uses the underlying module `sklearn.linear_model.LinearRegression`.
- **ARIMA** implemented in the class called `ArimaStockForecaster`, uses the underlying module `pyramid.arima.auto_arima`.
- **Prophet** implemented in the class called `ProphetStockForecaster`, uses the underlying module `fbprophet.Prophet`.
- **Long Short-Term Memory** implemented in the class called `LongShortTermMemoryStockForecaster`, uses the underlying module `keras.layers.LSTM`.

### 4. Evaluation and results

In this phase it was created the function to do the calculation of the evaluation metric which is RMSE (Root Mean Square Error), which is applied to measure the error in the validation set of the predictions of the closing prices against the ground truth.

Also in this phase it was created the code in Python to generate and automate the evaluation of the performance of the different implemented machine learning models, selecting a ticker symbol a data range for training and number of further trading days to do the validation.

The evaluation results comes as a data-frame, which indicates per each one of the models the RMSE of their predictions against the ground truth applied to a validation set containing a given numbers of trading days ahead. This is also accompanied with a plot displaying the predicted closing prices agains the real values, those evaluations are displayed in the "Results" section of this document.

For this phase, it was also created the code to display the reports of the performance (based on RMSE) per model and the percentage of improvement against the Linear Regression (which is the benchmarking model).

Finally after the evaluations were performed the results were visualized and analyzed, which discussion is presented on the "Results" section of this report.

### Refinement

The first aspect that was refined during the implementation was related with the mechanism for getting the dataset, the IEX's API only allows to get historical records for the previous 5 years, however the dataset management does not have to be limited by this constraint since an old dataset can be updated just by adding the missing records, and in this way having a mechanism for storing more data and keeping the dataset updated.

Another refined aspect was related to the prediction using linear regression, the idea was to do a variation of it to consider as predictors the date components (day, month, year, week, day-of-week and day-of-year) instead of just the date, wondering if they play a role in determining the closing price of the stocks, however as it will be shown in the "Results" section they do not seem to play a significant role.

The biggest refinement was done for LSTM. In studies like [11] the predicted closing prices seem to be "spectacular", however to do the prediction they use the validation set, which at the beginning seems to be like cheating, however in a second look it is actually a different approach to for the prediction mechanism. In all the other methods it is assumed that the known records are used in the training set, then the closing prices of dates to predict are completely unknown, and the model just tries to infer those values based on the training data.

However, the full potential of LSTM is reached if the dataset is periodically updated (ideally every day), that can help the model to get better predictions for the days ahead, without the need of retraining the model. This mechanism is identified in this project as "Long Short Term Memory - daily prediction" and the way it works is by updating the model at the end of each day and then doing the prediction for the next day. In this way the new data also play a role for predicting the future data without training the model again. This is the mechanism in which can be reached the "spectacular" results mentioned above.

The refinement done for LSTM was the incorporation of those two options, 1) inferring future closing prices assuming they are unknown (or if the dataset has not being updated), similarly to the other models and which might be useful for long term predictions, and 2) taking advance of the updated dataset to do the predictions for the next or future days, basically it consist in constructing the subsequence used as predictor with updated data. Details of this implementation can be seen in the code of the function `predict` in the class `LongShortTermMemoryStockForecaster`.

Also another important aspect of the refinement was tuning the parameters which are particular to each one of the three main models studied in this project: ARIMA, Prophet and LSTM. For linear regression there is not really anything to tune since the same algorithm can be seen as tuning the linear coefficients.

In regards to ARIMA, there are three important parameters to tune [11]:

- `p`: Past values used for forecasting the next value.
- `q`: Past forecast errors used to predict the future values.
- `d`: Order of differencing.

Parameter tuning for ARIMA consumes a lot of time, fortunately the module "Auto ARIMA" (used for this project) is available and has the advantage of automatically selecting the best combination of the values of (`p`, `q`, `d`) that provides the least error, this selection is done during the fitting/training process.

In the case of Prophet, Facebook claims that the default settings produce forecasts that are often as accurate as those produced by skilled forecasters, then there was not a lot of room for improvement, with the exception of the parameter `daily_seasonality` that was turned on to match with the problems nature of forecasting in daily basis. With this parameter turned on, it was actually observed a small improvement in the RMSE of the predictions of around 10%.

For the tuning of LSTM the list of parameters that were tuned during the refinement is described bellow:

- The number of time steps: This is the size of the subsequences used as input for the model, a number too small would undermine the predictive power and a number too large would take a lot of space in

the training set making it not suitable for small or moderate sets, the chosen value was 60 which was observed to offer a good balance.
- The number of layers in the architecture: LSTM layers seem to be very powerful that with just one layer (as in [10]) or two (as in [11]) can produce good results, the higher the number of layers the more resources and time it takes to perform training and predictions. The selected number of layers for this implementation was 2, which produced better results than just one layer and offered a good balance in terms of time/resources compared to architectures with 3 or more layers.
- The number of training epochs: LSTM surprisedly produced very low loss values in few training iterations, it was observed that just one epoch was not enough for the model to generalize, but for three or more epochs the training loss was so small that it was a clear symptom of overfitting (translated in not very satisfactory predictions during evaluation), the the number chosen was 2 for the number of training epochs.

## IV. Results

### Model Evaluation and Validation

Several examples of evaluations can be found at https://github.com/egar-garcia/machine-learning-engineer-nanodegree-capstone/blob/master/StockPricePredictor-Project-Development.ipynb where the models were evaluated for some ticker symbols and points in time. For purposes of space in this document only the official evaluation would be included.

The official evaluation consist in randomly select a date range for training, it should contain 750 trading days (which is around 3 years of historical records) and being old enough to leave at least 120 trading days in the dataset following the training end date to use for validation. Also it is selected a random ticker symbol in the Dow Jones, except 'DWDP', because it doesn't have enough historical records to do an evaluation with 750 trading days back.

The models/methods to evaluate are listed bellow:

- Linear Regression (the one used for benchmarking).
- Linear Regression - Date Components (linear regression using day, month, year, week, day-of-week and day-of-year as predictors)
- ARIMA
- Prophet
- Long Short Term Memory
- Long Short Term Memory - daily prediction (LSTM simulating the daily update of the dataset and daily prediction for the next trading day)

The RMSE is calculated agains the ground truth in the validation set for the following 1, 5, 10, 20, 40, 60 and 120 trading days after the training end date.

The results are observed in figure 4, the models that performed the best (with the minimum RMSE) are highlighted in yellow. For this report the random selection resulted in the ticker symbol 'WBA' (Walgreens Boots Alliance Inc.) and the training date range from 2015-04-21 to 2018-04-12.

In figure 5 it can be visualized how the predictions performed for each one of the evaluated models/methods in comparison with the validation set which contains the ground truth. As it can be observed "Long Short Term Memory - daily prediction" looks particularly impressive.

It's worth to mention that during the work done for this project two different prediction mechanisms have been found:

1. Predicting the closing prices for a date range in the future. In here the future data is completely unknown and the model would be used to forecast the closing prices for all the involved trading days or at least to describe a tendency. This is an extremely difficult problem and it seems that the models studied in this project are not able to do predictions with a lot of certainty, however by the evaluation

| | RMSE | | | | | | |
|---|---|---|---|---|---|---|---|
| **forecasting_days** | **1** | **5** | **10** | **20** | **40** | **60** | **120** |
| **method** | | | | | | | |
| **Arima** | 0.0105913 | 2.07859 | 2.15119 | 2.03018 | 2.30615 | 3.44772 | 9.24771 |
| **Linear Regression** | 9.17162 | 7.54199 | 7.64465 | 8.31271 | 8.38488 | 8.26837 | 6.48941 |
| **Linear Regression - Date Components** | 9.48351 | 7.92531 | 8.10425 | 8.80272 | 8.95529 | 8.81872 | 6.97792 |
| **Long Short Term Memory** | 0.125833 | 2.0095 | 1.97421 | 1.84207 | 1.70502 | 2.46651 | 6.86179 |
| **Long Short Term Memory - daily prediction** | 0.125833 | 1.54583 | 1.34463 | 1.52871 | 1.29984 | 1.81932 | 1.54195 |
| **Prophet** | 2.51819 | 1.33522 | 1.52532 | 2.46223 | 1.88238 | 2.26269 | 4.78276 |

Figure 4: Evaluation results by RMSE

results looks like Arima, Prophet and LSTM perform similarly and at least can give a tendency in short term even significant up to 40 or 60 trading days (8 or 12 weeks) in the future.

2. Predicting the closing price for the next day with and updated dataset. Assuming that the dataset is periodically (ideally every day) updated and the problem is reduced to predict the closing price for the next day. For this problem LSTM definitively excels, it reports consistently row RMSEs even for 120 trading days (24 weeks) in the future after training.

At the end more than creating a fixed predictor, the purpose of the project shifted to give options to the user to address those two described problems (or maybe something in between), for that reason rater than selecting only one model as a solution, the solution is to provide the user the option or using some of the models studied in this project.

### Justification

For the previous evaluation, in figure 6 are presented the improvement ratios (expressed in terms of percentage) against the benchmarking model (linear regression), the improvement ratio has been defined in the "Benchmark" section of this document.

The results are organized per number of following trading days after the training end date, the methods that performed the best are highlighted in yellow.

From those results it can be justified to rule out the method "Linear Regression - Date Components", since it does not really present an improvement, even in other evaluations included in the GitHub repository mentioned above it presents inconsistencies.

As stated in the bellow section the results shown that ARIMA, Prophet and LSTM present a similar performance behavior in short term and up to 40-60 trading days in the future. LSTM used to perform daily predictions in an updated dataset presents significant improvement even at 120 trading days ahead.

The the final solution is to provide the user the option of using the following models to do predictions for the stock closing prices:

- Linear Regression (since its usage is very standardized)
- ARIMA
- Prophet
- LSTM

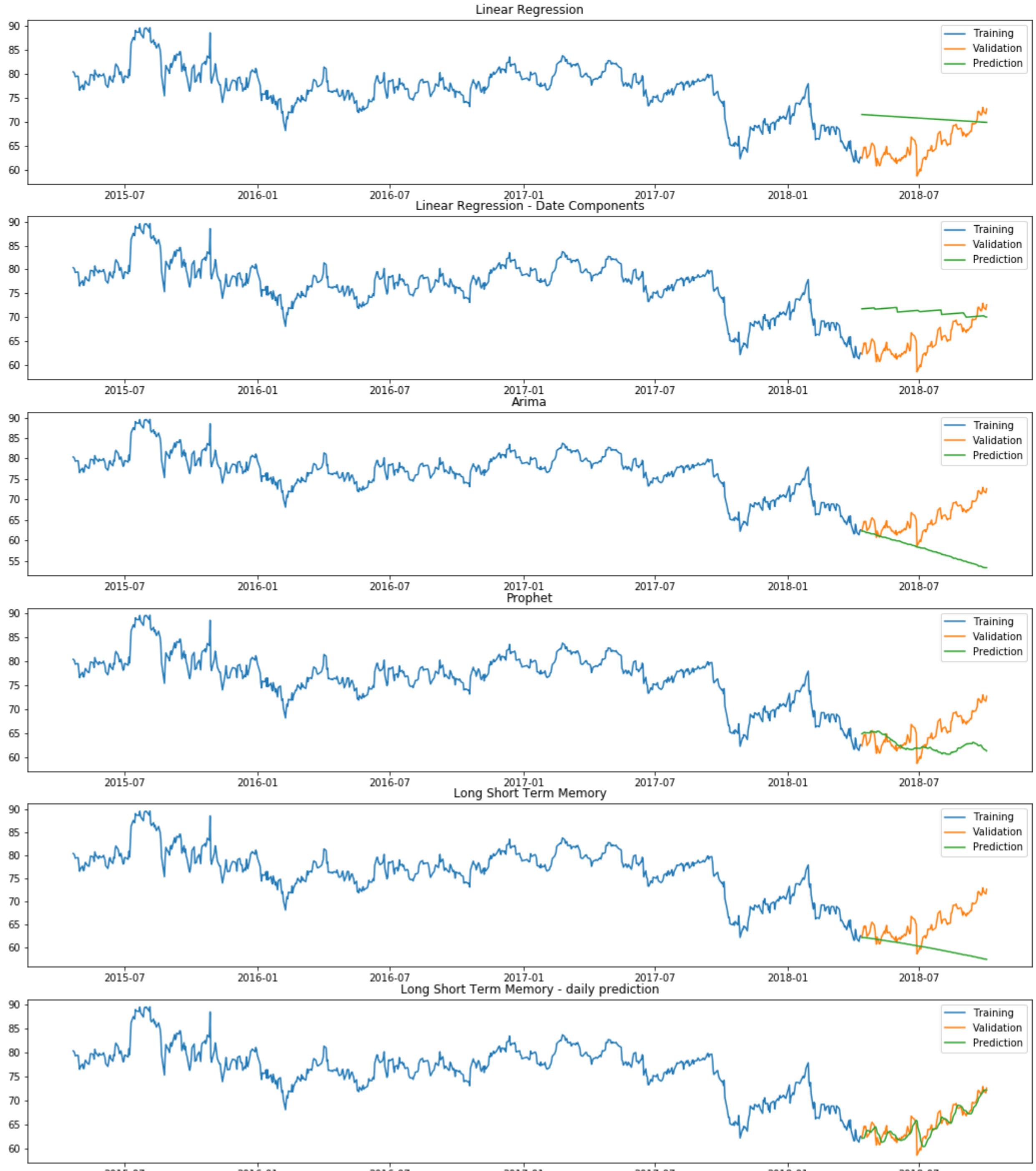


Figure 5: Predictions against ground truth of evaluated models

| | improvement | | | | | | |
|---|---|---|---|---|---|---|---|
| forecasting_days | 1 | 5 | 10 | 20 | 40 | 60 | 120 |
| method | | | | | | | |
| Arima | 99.88% | 72.44% | 71.86% | 75.58% | 72.50% | 58.30% | -42.50% |
| Linear Regression - Date Components | -3.40% | -5.08% | -6.01% | -5.89% | -6.80% | -6.66% | -7.53% |
| Long Short Term Memory | 98.63% | 73.36% | 74.18% | 77.84% | 79.67% | 70.17% | -5.74% |
| Long Short Term Memory - daily prediction | 98.63% | 79.50% | 82.41% | 81.61% | 84.50% | 78.00% | 76.24% |
| Prophet | 72.54% | 82.30% | 80.05% | 70.38% | 77.55% | 72.63% | 26.30% |

Figure 6: Improvement in regards to Linear Regression of evaluated models

The final solution is enclosed in the form of a Python file called `djia_stock_prediction.py` (available at https://github.com/egar-garcia/machine-learning-engineer-nanodegree-capstone/blob/master/djia_stock_prediction.py) which contains the constructions developed in this project to manage the dataset, and creating/training the models listed above. This solution is recommended to be used through a Jupyter notebook, since it provides a friendly environment where the user can perform experiments and visualizations. An example of the usage of this solution can be found at https://github.com/egar-garcia/machine-learning-engineer-nanodegree-capstone/blob/master/StockPricePredictor_example.ipynb

## V. Conclusion

### Free-Form Visualization

The visualization shown in figure displays the predictions using a Long Short-Term Memory forecaster for one year from 2018-04-01 for the ticker symbol 'BA' (Boeing Co.) and training data range from 2014-03-24 to 2018-03-31, i.e. more than 4 years of historical records for training, however just one year of training data is plotted –in blue– for visual purposes.

This is a radical example of the difference of doing predictions (for future dates) just with the information given in training –plotted in green–, against the mechanism when then dataset is daily updated and the prediction done for the next trading day –plotted in red–, they can be contrasted with the ground truth –plotted in orange–.

The red line looks to do a pretty good job with the predictions, even when this particular stock has a lot of volatility, the LSTM's training seems to be effective enough to correct the tendency and even performing well after 1 year without retraining. This is a clear example that shows how the predicting performance that LSTM can reached using by maintaining an updated dataset and predicting for the next trading day.

### Reflection

The problem of predicting stock prices was actually a lot more difficult than originally was expected, the stock prices have a lot of fluctuations where unexpected political, economical or social events can cause the prices to suddenly drop or rise, as mentioned in [guide_day_trading_online] the stock market mirrors the state of mind of the people involved, which are driven by human emotions like euphoria or fear.

Additionally, the problem is very different to other machine learning problems since the predictions should take place strictly in the future and not distributed along the domain of predictors/predictions. For this

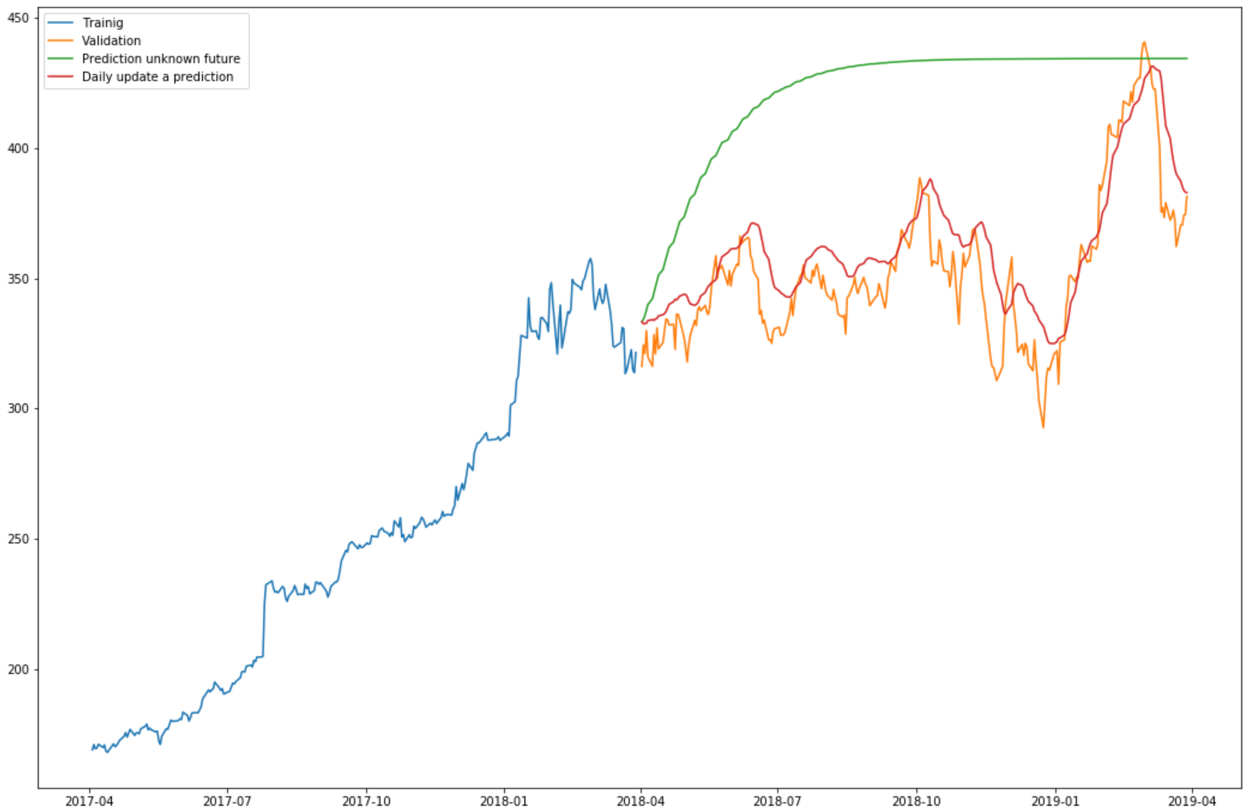


Figure 7: Daily update and prediction

particular kind of problems the way to address them is by using techniques for time series forecasting like the ones studied during this project.

The results obtained when addressing the problem of predicting stock prices in the future just by using the data already known (and used in training) where not very encouraging. The methods/models used in this project ARIMA, Prophet and LSTM can at least give a tendency, which looks to be relevant in short term (up to around 10 weeks ahead), however the results does not seem to be precise enough to predict with certainty a stock closing price for a particular day.

However, not everything is lost, with a variant on the mechanism for doing predictions, they can look way better, this is by using LSTM and periodically updating the dataset (ideally every day) and then doing the prediction for the next day, in this way the predictions for the next day would be more precise. Another advantage of LSTM is that retraining the model is not necessary, it was observed during the evaluation that after training the model this still gives good results for around 24 weeks ahead, it looks like even when the stock prices fluctuates LSTM can quickly adapt/correct the tendencies.

LSTM is not an easy algorithm to understand and then put in practice, there were some complications during its implementation, like implementing the preprocessing mechanism to create the sub-series used as predictors and the mechanism to take advance on the existing updated data to support the prediction process.

Originally in the proposal the idea of the final solution was creating a Python file to be used by the command line to perform the predictions, however the objective changed along the development of the project and instead shifted to provide a set of constructions (classes) able to be used in a Jupyter environment or similar, this is because it was realized that this kind of environments provide a more friendly framework to experiment and analyze the data, targeting it to be used for data scientists and/or (financial) data analysts.

### Improvement

One aspect to improve is the mechanism to perform long term forecasting, not aiming to predict the prices for a particular day but more like a long term average, basically to know if a stock its going to follow a tendency to increase or decrease in the future. A possible way to do that is by doing preprocessing of the training data that involves smoothing the historical records used in the training, for example by applying a weighted average applied to each trading day.

Another possible improvement would be by using technics of reinforcement learning for doing next day or even shorter term predictions, in last instance aiming to do forecast in real time targeting hours or minutes ahead, that would require an entire modification of the mechanism to keep the dataset update in shorter intervals, like the creation of an automated job to do the updates.

# VI. References


[1] A. O. A. Ayodele A. Adebiyi and C. K. Ayo, "Stock price prediction using the arima model," *International Journal of Simulation Systems, Science & Technology*, vol. 15, no. 4, pp. 105–111, 2014 [Online]. Available: http://ijssst.info/Vol-15/No-4/data/4923a105.pdf

[2] S. J. Taylor and B. Letham, "Prophet: Forecasting at scale." Facebook Research, 23-Feb-2017 [Online]. Available: https://research.fb.com/prophet-forecasting-at-scale

[3] R. Ritz, "Using facebook's prophet to predict mongolian stocks," 2018 [Online]. Available: https://medium.com/mongolian-data-stories/using-facebooks-prophet-to-predict-mongolian-stocks-cdf4feabd558

[4] Z. Pang X. and V. Chang, "Stock market prediction based on deep long short term memory neural network," in *Proceedings of the 3rd international conference on complexity, future information systems and risk (complexis 2018)*, 2018, pp. 102–108 [Online]. Available: https://www.scitepress.org/papers/2018/67499/67499.pdf

[5] A. C. M. P. David M. Q. Nelson and R. A. de Oliveira, “Predicting stock prices using lstm,” *International Journal of Science and Research (IJSR)*, vol. 6, no. 4, pp. 1754–1756, 2017 [Online]. Available: https://www.ijsr.net/archive/v6i4/ART20172755.pdf

[6] J. Brownlee, “How to create an arima model for time series forecasting in python.” Machine Learning Mastery, 09-Jan-2017 [Online]. Available: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

[7] C. Olah, “Understanding lstm networks.” 27-Aug-2015 [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[8] M. Soni, “Understanding architecture of lstm cell from scratch with code.” 21-Jun-2018 [Online]. Available: https://hackernoon.com/understanding-architecture-of-lstm-cell-from-scratch-with-code-8da40f0b71f4

[9] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 6, pp. 1735–1780, 1997 [Online]. Available: http://www.bioinf.jku.at/publications/older/2604.pdf

[10] J. Brownlee, “Time series prediction with lstm recurrent neural networks in python with keras.” Machine Learning Mastery, 21-Jul-2016 [Online]. Available: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

[11] A. Singh, “Stock prices prediction using machine learning and deep learning techniques (with python codes).” 25-Oct-2018 [Online]. Available: https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/