Capstone Project

Machine Learning Engineer Nanodegree

Gabriel S. de Jesus

August 13th, 2018

Using machine learning and technical analysis on stock prices predictions

Definition

Project Overview

Predicting the future is a challenging task from the earliest days of mankind, whether for climatic, economic, and so on. The technical analysis is a tool that was created with the purpose of helping the human being to make predictions in economic areas, although it was popularized in the 80's, it has been used since the 17th century as a tool for predicting future prices[1] . This tool underwent several modifications due to the technological advance and, currently it is widely used by traders to create strategies in the stock market.

The use of artificial neural networks as a model of machine learning on the other hand, had its starting point in the 1940s, though it became popular only after the 2000s due to the great advance of GPUs[2]. These algorithms nowadays are being used by machine learning engineers in the economic field as the same intention that the technical analysis is used by traders, to use of the quotations of a company to predict its future prices.

But, can we actually combine these two tools in order to accurately predict the stock prices? Trying to answer this question I created a predictive algorithm that uses not only the daily quotations of a company as input data, but also technical analysis indicators created from these quotations, in order to generate good predictions.

In order to complete this task, we used the Amazon daily quotes - AMZN from 01/03/2007 to 12/30/2017 (mm / dd / yyyy) collected from Yahoo Finance, we created several technical analysis indicators for this dataset and we used as a model of machine learning, recurrent neural networks based on long short term memory cells to predict the price of the stock at the time of closing on the same day, based on the prices and indicators of a given day.

This project was developed in python 3 through the Jupyter notebook and python files, you can find an html version of the notebook in the folder of this project, the python files can be found in the Helpers folder.

---

[1] "Análise técnica"<https://pt.wikipedia.org/wiki/An%C3%A1lise_t%C3%A9cnica#cite_note-7> (2018).

[2] "A History of Deep Learning" <https://www.import.io/post/history-of-deep-learning> (2018).

Problem Statement

As we said in the proposal of this project, the stock prices prediction is a challenge since the emergence of the stock market, however daily traders and machine learning engineers try to predict them. For this reason, we took some tasks as a challenge:

1. Get the historical data from a particular stock listed on the S&P 500
2. Create technical analysis indicators from these data
3. Create and improve a basic model based on support vector machines
4. Create and improve a model based on deep learning (LSTM neural network)
5. Compare the results obtained between them

It is expected that our final model will be able to make good predictions about the closing price of a given stock, based on the quotations and technical analysis indicators of a given day.

Metrics

Mean squared error (MSE) is a common metric for regression models, it evaluates the difference between an estimated value and the actual value and it is based on the mean square value of the error in the range of patterns presented. The closer its value is to zero, the better the prediction is, for this project, we expect to achieve a mean squared error in the fourth decimal place.

The mean squared error can be calculated as follows:

$$MSE = \frac{\sum_{t=1}^{n}(target_t - output_t)^2}{n}$$

At where:

- n is number of patterns presented
- target is target value
- output is output value of the m

Analysis

Data Exploration

The data set used for this project was pulled from Yahoo Finance and matches daily Amazon - AMZN quotes from January 3, 2007 to December 30, 2017 and can be found in the AMZN.txt file in the Data folder of this project.

The dataset takes the following form:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2007-01-03	38.680000	39.060001	38.049999	38.700001	38.700001	12405100
1	2007-01-04	38.590000	39.139999	38.259998	38.900002	38.900002	6318400
2	2007-01-05	38.720001	38.790001	37.599998	38.369999	38.369999	6619700
3	2007-01-08	38.220001	38.310001	37.169998	37.500000	37.500000	6783000
4	2007-01-09	37.599998	38.060001	37.340000	37.779999	37.779999	5703000

Table: The first five columns of the AMZN dataset

At where:

- The Date column represents the date that the quotes were recorded
- Open column represents the opening price of the market on that particular day
- High represents the maximum price that the stock was traded on the day
- Low, is similar to High however represents the lowest price traded in the period
- Close represents the price of the stock at the time of closing
- Adj Close (adjusted closing price) represents a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time before the next day's open
- Volume: Volume is the number of shares or contracts traded in a security or an entire market during a given period of time

From these data, the following technical analysis indicators were calculated (discussed later in the Algorithms and Techniques section):

- Moving Average - MA10
- On Balance Volume - Obv
- Upper Bollinger Band - UpBand
- Lower Bollinger Band - LowBand
- Relative strength index - RSI
- Stochastic Oscillator K - K
- Stochastic Oscillator D - D
- Exponential Moving Average - EMA
- Rate of change and momentum - ROC
- Moving Average Convergence Divergence - MACD

In this project, we aimed to predict the closing prices of the stocks on a given day (Close Column) to perform such a task, we chose not to use the High, Low and Adj Close columns as direct input because in addition to being used to calculate the indicators of technical analysis, in a real application, it would only be valid to use these columns as input if we used this information from the previous day, since we would not have access to this information on the day of the prediction. No anomaly was found in the dataset however we verified the occurrence of some issues we should solve, described below.

The tables below show the statistical description of the numerical data of our dataset:

	Open	Close	Volume	RSI	MA10	EMA	MACD
count	2697.000000	2697.000000	2.697000e+03	2697.000000	2697.000000	2697.000000	2697.000000
mean	330.745814	330.782447	5.791825e+06	50.692084	328.468250	328.891562	-2.886193
std	280.335706	280.181034	5.016062e+06	0.359470	277.700908	278.067662	7.770532
min	35.290001	35.029999	9.844000e+05	50.171309	39.869000	40.829737	-39.603311
25%	118.709999	118.870003	2.999400e+06	50.484783	118.962000	118.969191	-5.400720
50%	235.410004	234.779999	4.507300e+06	50.606913	232.952000	233.554650	-2.059759
75%	422.959991	421.779999	7.068200e+06	50.822792	401.850000	412.484975	1.001646
max	1204.880005	1195.829956	1.043292e+08	53.274836	1179.721997	1177.369563	35.165931

Table: Numerical data – part 1

	Roc	K	D	Obv	UpBand	LowBand
count	2697.000000	2697.000000	2697.000000	2.697000e+03	2697.000000	2697.000000
mean	0.021303	-0.000646	-0.000647	4.533747e+08	345.379995	307.365068
std	0.088971	0.000704	0.000671	1.617220e+08	286.980792	264.488093
min	-0.388015	-0.005140	-0.005010	-5.439200e+06	44.602052	31.482519
25%	-0.029405	-0.000921	-0.000907	3.075317e+08	128.994719	109.674316
50%	0.017691	-0.000491	-0.000483	4.698280e+08	248.115870	216.116316
75%	0.064854	-0.000188	-0.000190	5.704414e+08	432.160426	385.059614
max	0.506238	0.001295	0.000488	7.732871e+08	1210.983895	1140.552939

Table: Numerical data – part 2

Through the statistical description above, we notice through the mean that the scale order of each feature are in a very large scale order such as OBV ($1x10^8$) and Volume ($5x10^6$) while other variables are in an extremely low scale order such as the case of stochastic oscillators ($1x10^{-4}$) and Roc ($1x10^{-2}$), this behavior of the data can also be noticed by the values of minimum and maximum, which are completely different from each other. Letting the data in this way can harm our models, since only a few variables will be 'focused' while the others will be reduced. That said, we normalized our dataset, getting the final form of our dataset:

	Open	Close	Volume	RSI	MA10	EMA	MACD
Mean	0.252615	0.254783	0.046518	0.167801	0.253190	0.253455	0.491072
Std	0.239687	0.241369	0.048537	0.115826	0.243629	0.244662	0.103927
Min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Table: Numerical data normalized – part 1

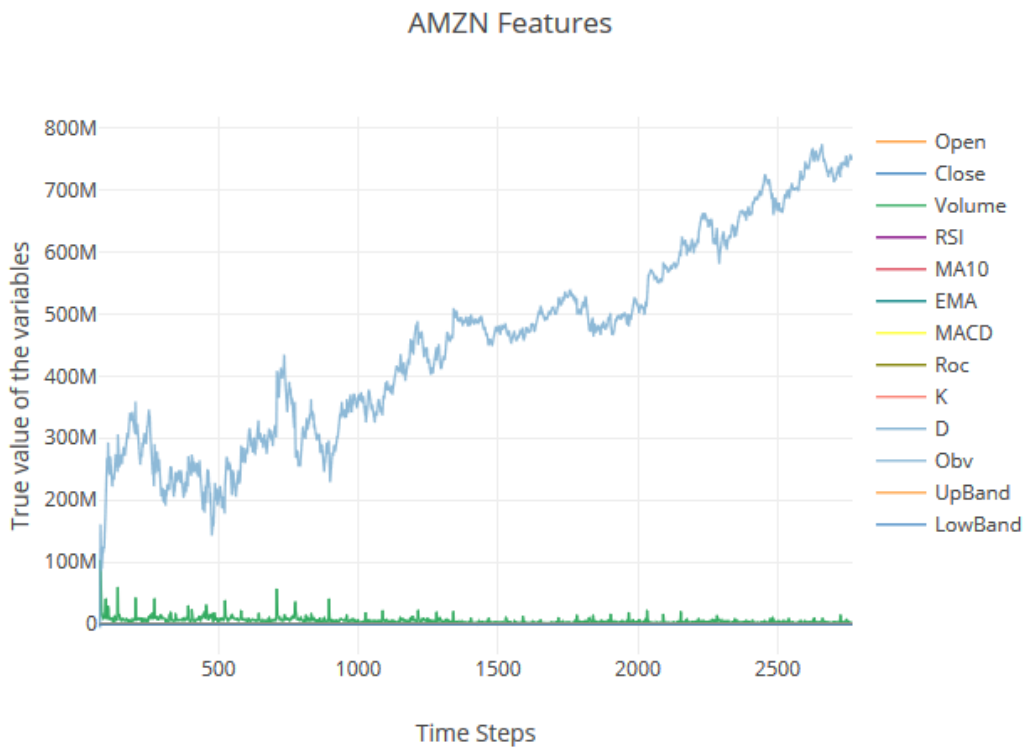
	ROC	K	D	Obv	Up Band	Low Band
Mean	0.457721	0.698383	0.793665	0.589185	0.257873	0.248751
Std	0.099492	0.109443	0.122079	0.207675	0.246044	0.238477
Min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Table: Numerical data normalized – part 2

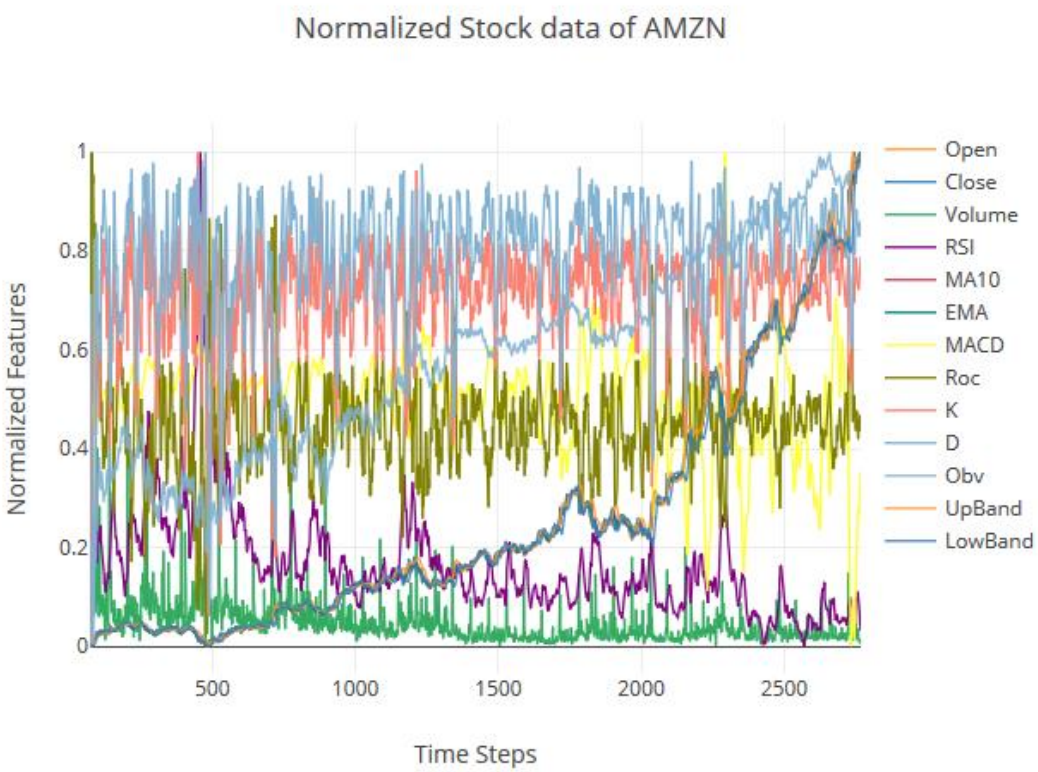
We can verify that the whole data are normalized in a range between 0 and 1, we can also notice that the average of the data are now in the same scale and that the whole dataset are also with a low standard deviation, being in this way, correctly normalized and ready to be used by machine learning models.

Exploratory Visualization

Another way of visualizing the problem of non-normalized of characteristics is to graphically visualize the values of our characteristics, as shown below:



We can clearly see from this graph that some variables like OBV and Volume are on a much larger scale than the other variables, graphically it is even impossible to visualize the variations of the other variables due to the large variation of the range between them. However, after normalized we got at this result:



Here we can see the change according to the time of all variables, which indicates graphically, a correct normalization.

Algorithms and Techniques

In this project it was sought to use technical analysis tools together with machine learning tools, thus creating a precise predictor. The indicators of technical analysis give us a sense of direction of where the market is going, we use these indicators as a technique to improve our input data of our models, we aimed to calculate at least one technical analysis indicator for each group as showed below:

- Trend indicators: The trend indicators provide the direction of the market, showing if it is rising or falling. In this project we use three of these indicators.
    - Moving Average: provides the average value of a quote in a time interval, here was used 10 days. It is characterized as "moving" because for each value included in the averaging calculation, the oldest value is excluded.
    - Exponential Moving Average: Similar to the moving average, however, the indicator gives more emphasis to the most recent values, giving higher weights to these values and lower to the oldest ones. The oldest weights fall exponentially, from where comes the name of the average.[3]
    - MACD (Moving Average Convergence Divergence): represents the difference between a short-term moving average and a long-term moving average.

[3] "Redes neurais artificiais para predição no mercado acionário brasileiro" Fábio Maritan Pereira.

- ❖ Moment Indicators: A momentum indicator measures the rate-of-change of a security's price. As the price of a security rises, price momentum increases.[4] In this project we use four momentum indicators.
    - ROC (Rate of Change and Momentum: Measures the percentage change between today's price and the price in a given period n days ago.
    - Relative strength index (RSI): Compares the magnitude of the mean of gains versus the mean of losses on a scale of 0 to 100. The closer to 100, the higher the gain of a given period in the past.
    - Stochastic Oscillator K: Refers to the point of a current price in relation to its price range over a certain number of days.
    - Stochastic Oscillator D: represents the 3-day moving average of the Stochastic Oscillator K.
- ❖ Volume Indicators: Volume indicators are based on the fact that the volume precedes the price movement. We use just one indicator of this modality.
    - OBV (On Balance Volume): It is an indicator used to measure the positive or negative flow of the volume, which correlates with the trend of the stock price.
- ❖ Volatility Indicators: They show if prices are too volatile (going up and down without a definite trend). In this project we used the Bollinger bands as an indicator of volatility, subdividing them into upper-band and lower-band.
    - Upper Bollinger Band: Consist at K times an N-period standard deviation above the moving average (MA + Kσ).
    - Lower Bollinger Band: Consist at K times an N-period standard deviation bellow the moving average (MA - Kσ).

As a computational model for stock prediction, was used artificial recurrent neural networks, which are the state-of-the-art algorithm for sequential data. Because of their internal memory, RNN's are able to remember important things about the input they received, which enables them to be very precise in predicting what's coming next[5]

However, when we train a Neural Network model using Gradient based optimization techniques the gradients of loss tend to get smaller and smaller as we keep on moving backward in the Network in the backpropagation technic[6] (The Vanishing Gradient Problem), not allowing us to learn from past. To solve this, we used Long-Short Term Memory Neural Networks, that are a type of recurrent neural networks, with LSTMs, the information flows through a mechanism known as cell states, this way, LSTMs can selectively remember or forget things[7] enabling us to learn patterns based on past events in a better way.

---

[4] "Introduction to Technical Indicators and Oscillators " <https://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:introduction_to_technical_indicators_and_oscillators> (2018).

[5] "Recurrent Neural Networks and LSTM" <https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5> (2018).

[6] "The Vanishing Gradient Problem" <https://medium.com/@anishsingh20/the-vanishing-gradient-problem-48ae7f501257> (2018).

[7] "Essentials of Deep Learning : Introduction to Long Short Term Memory " <https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/> (2018).

Benchmark

As proposed for this project, we created and Improved a Support Vector Regression (SVR) in order to compare the performance between the SVR and the LSTM Neural Network.

Due to the fact that SVMs can efficiently perform a non-linear classification using what is called the kernel trick, was expected from the SVR model be able to generalize your data, however it has a terrible. The MSE in the training set reach at 0.00400146, and in the test data it had a result of 0.11320664, which is extremely bad, since a good regressor should have its mean squared error as close to zero (approximately four decimal places).

In order to make a fair comparison with our final model, we decided to improve our benchmark model, for that, we used a grid searching algorithm, which is a process of scanning the data to configure optimal parameters for a given model[8]. For that we used the following parameters:

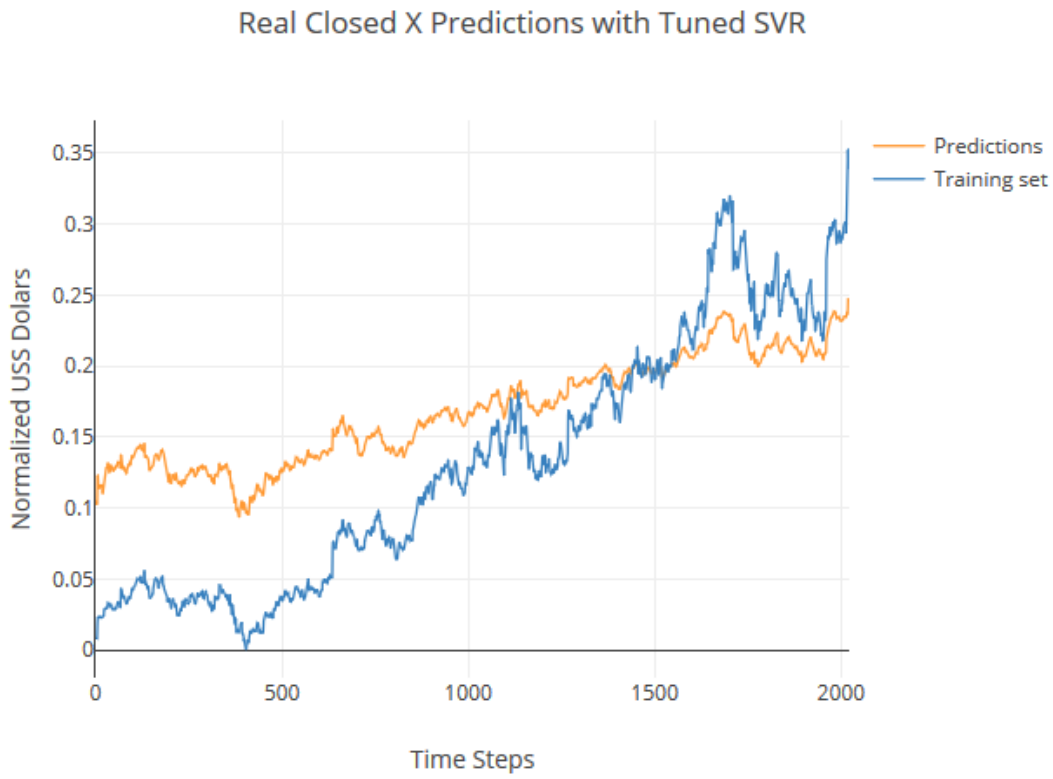
- C: 1.0, 1.2, 0.8
- epsilon: 0.1, 0.01, 0.001
- coef0: 0.0, 1e-3, 1e-4, 1e-5
- kernel: 'rbf', 'linear', 'sigmoid
- tol: 1e-3, 1e-4, 1e-5

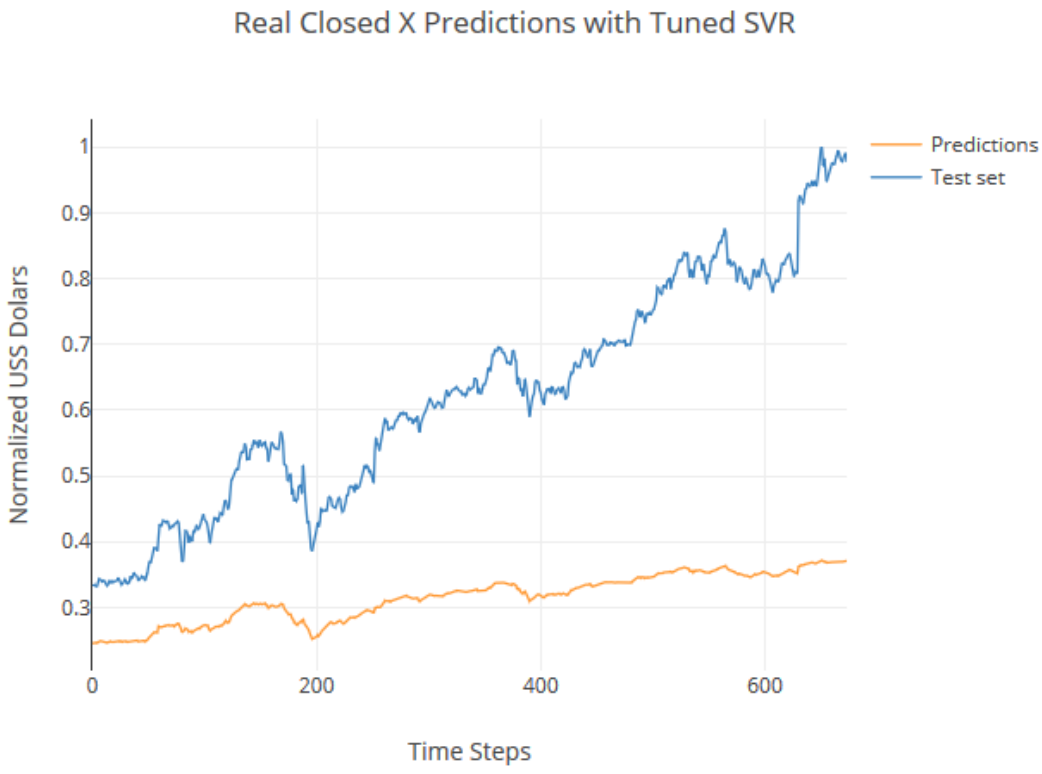
After searching, we find the follow parameters as the best:

- C: 0.8
- epsilon: 0.1
- coef0: 0.0
- kernel: 'rbf'
- tol: 1e-05

However, even after our model was improved it only had a small improvement, obtaining MSE 0.00398093 in the training data and 0.11338206 in the test data. We can easily verify by the charts below that our support-based model failed to generalize our data.

[8] "Grid Searching in Machine Learning: Quick Explanation and Python Implementation" <https://medium.com/@elutins/grid-searching-in-machine-learning-quick-explanation-and-python-implementation-550552200596> (2018).


Real Closed X Predictions with Tuned SVR
Predictions
Training set
Normalized USS Dolars
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
500
1000
1500
2000
Time Steps


Real Closed X Predictions with Tuned SVR
Predictions
Test set
Normalized USS Dolars
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
200
400
600
Time Steps

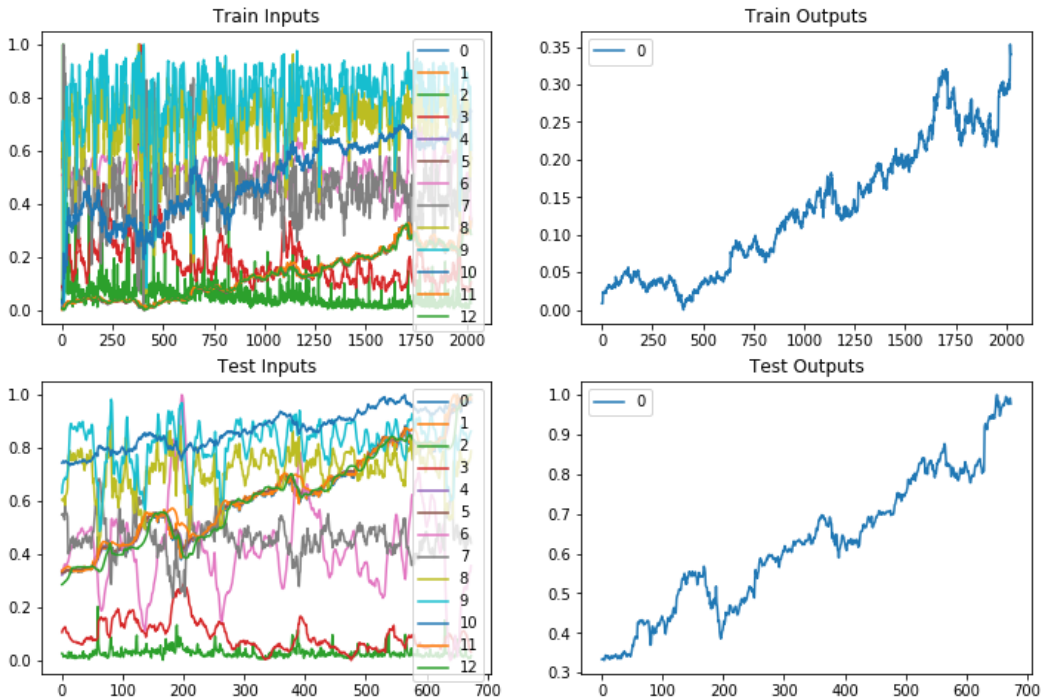
Methodology

Data Preprocessing

The preprocessing was done in the “Preparation of dataset” section of the notebook and consisted of the following steps:

1. Getting the data: Here we received the AMZN data from Yahoo Finance
2. Adding new information to the dataset: We added all the technical analysis indicators described in algorithms and techniques.
3. Removing unnecessary information from our data: Here we remove the High, Low and Adj Close columns from our data set, since they have already been properly used for calculating the technical indicators.
4. Analyzing our variables: We explore data through views and codes to understand how relevant each feature is and how it relates to each other.
5. Normalizing data: As a result of the analysis of variables, we verified the need to normalize our dataset to a range of compact values (we use a range between zero and one), done in this section.
6. Splitting the dataset: Here divided our dataset into inputs (X) and outputs (y). In order to avoid overfitting, we will also split our data between training set and test set, so that way we can test our model on previously unseen data.
7. Visualizing the final form of the dataset: And finally, we demonstrate a final representation of our data in this section.

At the end of the data processing, we obtained the following result:



Through this graphical description, we can observe our 13 input variables, listed from 0 to 12 correctly normalized, as well as our outputs. By visualizing the final form of our dataset, was noticed that the outputs of the test data have higher values than those of training, however we assumed that the models must be able to detect this increase due to the variations of the input characteristics.

Implementation

To compare the models we created, we used as described in Metrics, the mean squared error for all models created in this project. After our data were collected and preprocessed, we started the implementation, and following steps were taken:

1. We created and trained a model based on support vectors as benchmark model, however, the model was not able to achieve the results expected by it, and due to this we decided to improve it.

2. We improved our benchmark model and again we evaluated its performance, and although we have passed a number of parameters to better design it, its improvement was minimal, almost nonexistent.

3. We analyze the results of this model and conclude that the model was not able to generalize the dataset, barely able to tell the market trend.

4. We created and trained our neural network model, with the following structure:

    a. A layer of LSTM cells with 30 units

    b. A dropout layer with a percentage of 0.2

    c. An activation layer using a linear function

5. We compiled the model having as loss function the mean squared error and as optimizer, the Adam optimizer, which you can read better here[9].

6. We evaluate the LSTM model and verified both by MSE and graphically that it performed much better than the SVR, however, it was not yet making predictions with the expected accuracy.

7. We improved our model as described in the Refinement section

8. At the training step, the model was reset and trained five times, aiming at a better initialization of the synaptic weights, being maintained the one that obtained the better performance.

9. We evaluated our model and finally, we got a good result, both for the test data and for the training data.

10. At the end, after comparisons and discussions, we implemented an optional code cell in which it registers a Tensorflow event so that we can visualize through TensorBoard our model in more detail.

[9] "An overview of gradient descent optimization algorithms - Adam" <http://ruder.io/optimizing-gradient-descent/index.html#adam> (2018).

Refinement

As mentioned in the Implementation section, the LSTM architecture trained with Keras did not achieve the expected accuracy, because of that, we improved our model by using the following techniques:

- Add new layers in our network
- Increase the number of units in the first layer of LSTM cells
- Use a higher percentage in the last dropout layer (0.4 was used)
- reset and retrain the model multiple times, keeping the one that obtained more accurate results (5 times).

With this, the architecture has changed from a layer of LSTM cells with 30 units followed by a dropout layer with a percentage of 0.2 and an activation layer using a linear function, to the following form:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_22 (LSTM)               (None, None, 60)          17760
_________________________________________________________________
dropout_22 (Dropout)         (None, None, 60)          0
_________________________________________________________________
lstm_23 (LSTM)               (None, None, 512)         1173504
_________________________________________________________________
dropout_23 (Dropout)         (None, None, 512)         0
_________________________________________________________________
lstm_24 (LSTM)               (None, None, 256)         787456
_________________________________________________________________
dense_17 (Dense)             (None, None, 256)         65792
_________________________________________________________________
dropout_24 (Dropout)         (None, None, 256)         0
_________________________________________________________________
lstm_25 (LSTM)               (None, 128)               197120
_________________________________________________________________
dense_18 (Dense)             (None, 128)               16512
_________________________________________________________________
dropout_25 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 129
_________________________________________________________________
activation_7 (Activation)    (None, 1)                 0
=================================================================
Total params: 2,258,273
Trainable params: 2,258,273
Non-trainable params: 0
_________________________________________________________________
```

When evaluating this model, we managed to reduce the mean squared error from 0.01867950 to 0.00017158, achieving, in this way, desired results.

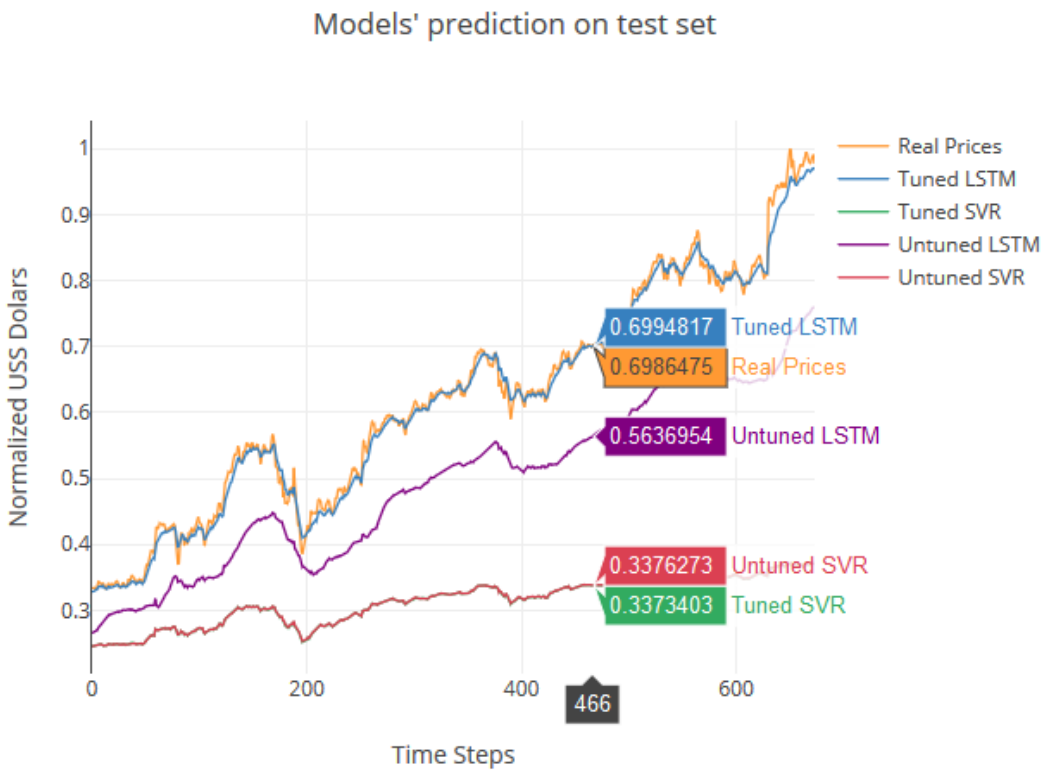
Results

Model Evaluation and Validation

In summary, we had four predictor models during this project, both had the mean squared error as a metric in order to compare the results between them. All the models developed during the project were trained in the training set and evaluated in the testing set, for check their robustness we also used the test data due the fact that is a data that the model had not previously seen. The following results were obtained:

MODEL:	*MSE:*
Untuned SVR	0.11320664
Tuned SVR	0.11338206
Untuned LSTM	0.01867950
Tuned LSTM	0.00017158

Table: Checking the robustness of the models



The following list refer to the complete description of the final model shown in the Refinement section:

- A LSTM layer with 60 units, returning sequences
- A Dropout layer with 0.2 percent of activation
- A LSTM layer with 512 units, returning sequences
- A Dropout layer with 0.2 percent of activation
- A LSTM layer with 256 units, returning sequences
- A Dense layer with 256 units
- A Dropout layer with 0.2 percent of activation
- A LSTM layer with 128 units
- A Dense layer with 128 units
- A Dropout layer with 0.4 percent of activation
- A Dense layer with 1 unit
- An Activation layer with function linear

Justification

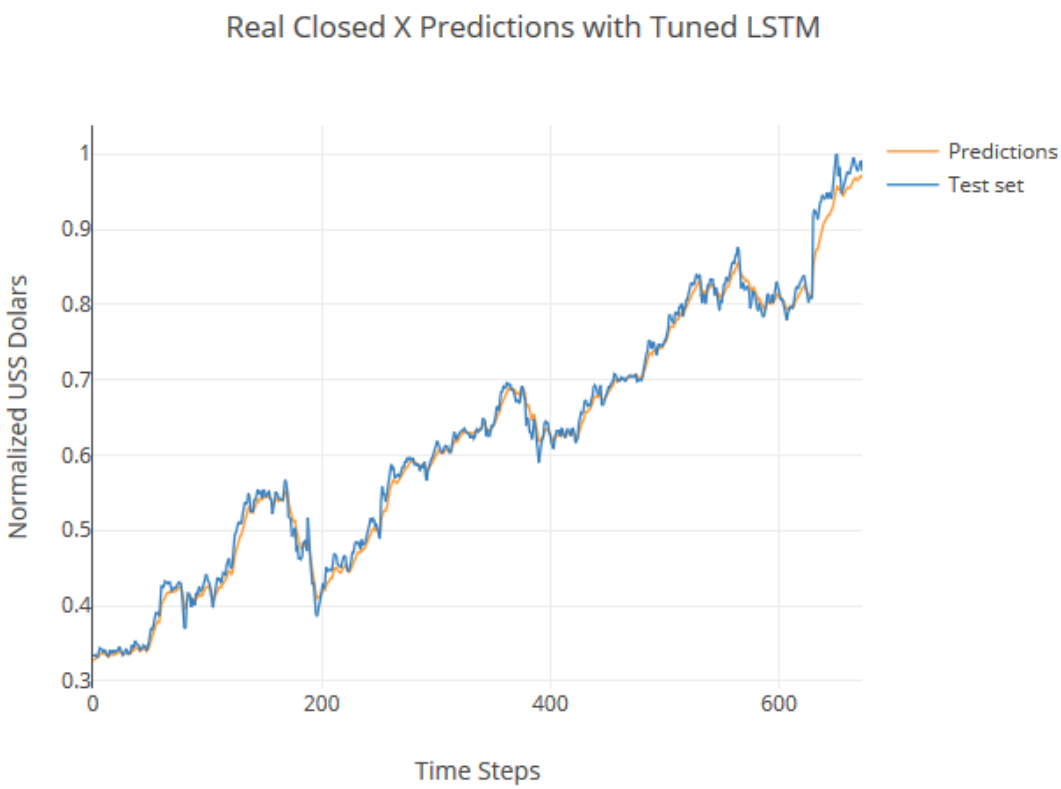
Using the LSTM Neural network we arrive at a mean squared error of 0.00017158 which was considered a very accurate and much more accurate result than our model based on support vectors that had an MSE of 0.11338206, with this we analyzed the reason of our benchmark model not having been as successful as our best model and also, we were able to answer our question that was the starting point for the development of this project.

To understand how successful the final application is, it is also important to take a look at the final predictions, shown in the Free-Form Visualization section, there we can clearly see the success of predictions using technical analysis indicators along with our final model of machine learning, we can note that such a model may be applicable in the real world, which can change the way many traders trade stocks , however, I cannot say that even with my goal for this project has been completed there is nothing more to do, actually there are many improvements that can and should be done for a real-time application, we discussed this topic better in the Improvement session.

Conclusion

Free-Form Visualization

For a final view, I chose the prediction results from my improved model based on the long-short term memory cells, which we obtained the following results in the test set:



We can see from the graph above that the predictions of our model were very accurate both in analyzing the market trend and in the high and low points, the moments in which the predicted values moved further from the real were from step 600, where there was a sharp rise in Amazon prices, however, even with this abrupt change, the model continued to predict its bullish trend, it just predicted more smoothly, which was probably due not only to the LSTMs' power to predict time series, but also to the three Trend Indicators we used to improve the inputs of our neural network. Because of this, I would say that this project fulfilled its purpose with success, being able to answer clearly our initial question creating one more bond between machine learning and technical analysis.

Reflection

Recapping, this project was developed in Jupyter Notebook with Python, using the APIs Keras (that runs on top of TensorFlow) and the Scikit-learn API as helpers in the creation of predictive models. Was used a dataset of Amazon-AMZN quotes from January 3, 2007 to December 30, 2017, collected from Yahoo Finance. The process used for this project can be summarized using the following steps:

1. Preparation of dataset
    - ❖ Getting the data
    - ❖ Adding new information to the dataset
    - ❖ Removing unnecessary information from our data
    - ❖ Analyzing our variables
    - ❖ Normalizing data
    - ❖ Splitting the dataset
    - ❖ Visualizing the final form of the dataset

2. Bench Mark Model
    - ❖ Untuned Bench Mark Model
        - Creating
        - Training
        - Evaluating
    - ❖ Tuned model
        - Tuning the model
        - Evaluating the Tuned model
    - ❖ Visualizing the Bench Mark Model's predictions

3. LSTM MODEL
    - ❖ Reshaping the inputs
    - ❖ Untuned Model
        - Creating the model
        - Training the model
        - Evaluating the model
        - Visualizing the Model's predictions
    - ❖ Tuned model
        - Creating and Training the model
        - Evaluating the improved model
        - Visualizing the Model's predictions

4. Comparing the results
    - ❖ Tabular Results
    - ❖ Results in Charts
    - ❖ Conclusions
    - ❖ Winner model on Tensor Board (Optional)
5. Discussions for further projects
6. Report

With the development of this project we were able to answer our initial question in which we wondered if we could combine the tools of technical analysis and machine learning together. Given the final results of our best model, we can say that the combination of the two tools generated very accurate predictions.

As the test data were in a different range of training data, support vector models were unable to generalize predictions, even with input variations, whereas neural network-based models were able to measure this correlation, however the support vector machines did not become a disposable approach for future projects, but instead, as listed in the next session improvement, a good challenge would be split in a different way the training and test data to try to achieve better results with the support vectors.

Improvement

As was said in the Jupyter Notebook of this project, we had a great success predicting the daily closing prices of the Amazon stocks, but there are much that can still be improved and / or tested with the same. For this reason, we provided a list, described below, of options that new projects can implement to from this:

- ❖ Use the data set from other companies to train models
- ❖ Add other technical analysis indicators as input and check if there are improvements
- ❖ Adapt the models and try to predict the closing of the actions in a longer period of time (a week, a month, ...)
- ❖ Collect the intraday stock prices and try to predict the stock closing price in a shorter period of time (per minute, every five minutes, half hour, ...)
- ❖ Split in a different way the training and test data to try to achieve better results with the support vectors