Predicting the directionality of S&P500

Capstone Project

Yemitan Isaiah Olurotimi

November 3rd, 2019

I. Definition

Domain Background

S&P 500 is a stock market index that measures the stock performance of 500 large companies listed on stock exchanges in the united states. The index is widely considered as the best indicator of the day to day performance of the U.S stocks.

Over 50 years for data was analyzed, It was deduced that there is approximately 52% chance that the stock price will increase and 48% that it will decrease for any given day.

The dataset is in time series in which it contains a timestamp and data spaced by a day. The data was gotten from the yahoo finance website. Where the CSV file can be downloaded.
Link: https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC

This project aims to use some machine learning techniques to predict the directionality of stock prices.

My personal motivation for this topic stems fro

m the fact that I currently work with Openinvest which is a company which brings openness and transparency to investors. We use different data such as s&p500, as a technical person, I felt this is an opportunity for me to understand the finance and investment field more and be able to apply machine learning technique to my current job.

Some academic papers, research and journals related to this topic are listed below;

- https://www.academia.edu/36810883/Prediction_for_Stock_Marketing_Using_Machine_Learning
- https://www.researchgate.net/publication/316177357_Stock_Prediction_using_Machine_Learning_a_Review_Paper
- https://www.researchgate.net/publication/330456642_Algorithmic_Machine_Learning_for_Prediction_of_Stock_Prices
- https://pdfs.semanticscholar.org/e567/e0ae41ac2b757b1e069c6d345c6a7413d9c1.pdf
- https://www.hindawi.com/journals/mpe/2019/7816154/
- https://www.sciencedirect.com/science/article/pii/S1877050918307828

Problem Statement

Stock prices are usually described as a statistical process called “random walk” which means each days closing value is unpredictable and random.

The problem to be solved with this project is to apply machine learning to predict the directionality of a stock price for any given day using over 50 years of historical data.

For solving the problem, Yahoo finance data for S&P500 was downloaded in a CSV file. The data was preprocessed and normalized accordingly, the preprocessed data was then divided into training and testing data. Then I split the data into training and testing data. The data was trained using LSTM and Arima and the results were compared with the actual data and the benchmark.

Outline of the workflow includes;

- Collecting Data
- Cleaning and pre-processing data
- Normalize data
- Splitting data to Training and testing set.
- Use the various models, architecture and tune the parameters
- Train model
- Test model
- Compare model results with the benchmark.

This is data contains 12,614 rows and 7 columns which are;

- Date: Date for the given data.
- Open: The stock opening price for the given date.
- High: Highest stock price for the given date.
- Low: Highest stock price for the given date.

- Close: The stock closing price for the given date.
- Adj Close: Adjusted closing price for the given date.
- Volume: Volume of stock traded for the given date.

Link: https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC

-

Metrics

In this project, the model was evaluated using Mean squared error and mean absolute percentage error.

Mean squared error measured the average of the squared of the errors (Average of the difference between the estimated values and actual value)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Mean absolute percentage error is a measure of prediction accuracy of a forecasting method. It is used for regression problems.

$$\mathrm{M} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|,$$

II. Analysis

Data Exploration

The dataset for S&P 500 is publicly available and will be obtained from yahoo finance.
The time period for the data set is between October 1969 to October 2019 which is 50 years.
This is data contains 12,614 rows and 7 columns which are;

- Date: Date for the given data.
- Open: The stock opening price for the given date.
- High: Highest stock price for the given date.
- Low: Highest stock price for the given date.

- Close: The stock closing price for the given date.
- Adj Close: Adjusted closing price for the given date.
- Volume: Volume of stock traded for the given date.

Link: https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC

Exploratory Visualization

The important columns that were used are; Date, Open and Adj Close. Adj close was preferred over close as it is the final closing price for the given date.

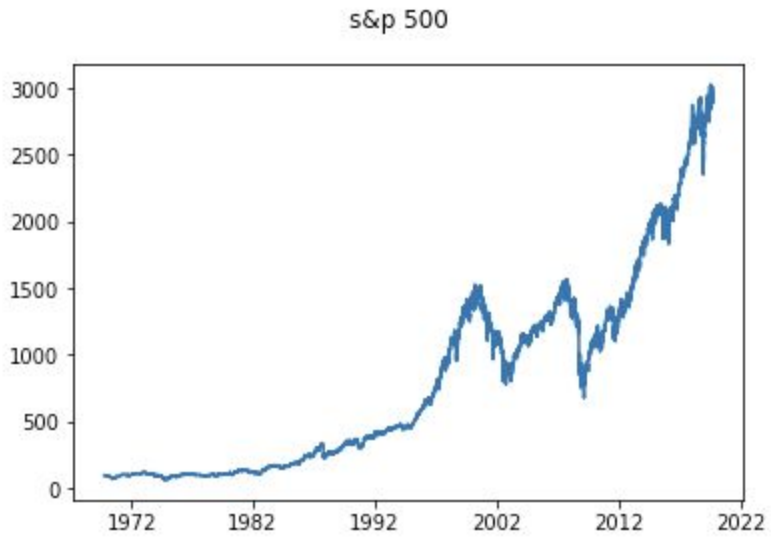
Below is a sample of the data

	Date	Open	High	Low	Close	Adj Close	Volume
0	1969-10-13	93.559998	94.860001	93.199997	94.550003	94.550003	13620000
1	1969-10-14	94.550003	96.529999	94.320000	95.699997	95.699997	19950000
2	1969-10-15	95.699997	96.559998	94.650002	95.720001	95.720001	15740000
3	1969-10-16	95.720001	97.540001	95.050003	96.370003	96.370003	19500000
4	1969-10-17	96.370003	97.239998	95.379997	96.260002	96.260002	13740000

Description of the data

	Open	High	Low	Close	Adj Close	Volume
count	12614.000000	12614.000000	12614.000000	12614.000000	12614.000000	1.261400e+04
mean	804.955881	809.684315	799.922962	805.128490	805.128490	1.321124e+09
std	743.595826	747.094008	739.739642	743.668664	743.668664	1.709035e+09
min	62.279999	63.230000	60.959999	62.279999	62.279999	6.650000e+06
25%	131.232502	132.649994	129.915001	131.262497	131.262497	5.667250e+07
50%	467.059998	468.914993	465.544999	467.089997	467.089997	3.046950e+08
75%	1277.127533	1283.952484	1267.552551	1277.345001	1277.345001	2.606752e+09
max	3024.469971	3027.979980	3014.300049	3025.860107	3025.860107	1.145623e+10

Data Chart



Algorithms and Techniques

The task for this project is to predict the directionality of stock prices of the next N day(s)/ month in the future using historical data as its input.

Two ML algorithms were used to predict the closing stock price. The algorithms that will be used are Long short term memory (LSTM), Autoregressive integrated moving averages (Arima) and random forest for the benchmark. These models will be used individually and compared to see which performs best.

Long Short-Term Memory (LSTM) is an artificial recurrent neural network architecture that is very applicable in time-series analysis and prediction. LSTM can be used to address sequence and time-series problems and achieve good results.
LSTM architecture contains 3 types of gates which are;

- Forget gate: Conditionally decides what information to throw away from the block
- Input gate: Conditionally decides which value to update the memory state.
- Output gate: Conditionally decides what to output based on input and the memory block.

With these, we can achieve sophisticated learning and memory from the layers of LSTM.

The project uses 3 layers for LSTM.

1. 128 units and a sigmoid activation function,
2. Dropout (0.2)
3. Dense layer

The model was then compiled using mean-squared-error for loss and Adam for its optimizer. The LSTM model was then fitted with the data after compiling.

Autoregressive Integrated Moving Average Model (ARIMA) is an architecture used to fit time series data, to be able to better understand the data, make analysis or to predict the future points in the time series.
The parameters for Arima models are defined as;

1. p: Number of lag observation included in the model
2. d: Number of times the raw observations are differenced
3. q: Size of the moving average window

In the project I used p = 0, d = 1, q = 4. I tried various parameters and arrived at these as they perform better. Arima(0, 1, 4)

Random Forests contains a large number of decision trees that operate as an ensemble. Radom forest is ensemble learning methods for classification, regression and other tasks. Radom forest was applied in the project because the S&P500 is perceived as a random walk hence random forest is good for providing a benchmark for the project.
RandomForestClassifier was used from Keras in the project.
Default parameters were applied as this just for the benchmark.
Before applying the random forest algorithm, the training output data were preprocessed by using Labencoder to encode the labels, the results were also transformed back to the appropriate data.

Benchmark

The benchmark employed is random forest. Random Forests contains a large number of decision trees that operate as an ensemble. Radom forest is ensemble learning methods for classification, regression and other tasks. Random forest was applied in the project because the S&P500 is perceived as a random walk hence random forest is good for providing a benchmark for the project.
The random forest has a high predictive ability for time series data, it can be applied well with the time series data.

As seen below, it can be confirmed that the model (LSTM) performs better than the benchmark (random forest). Comparing the mean squared error and mean absolute error, we can conclude that the project is successful.

Results table

Bechmark/Model	MSE	MAE
Benchmark	2135.71	33.97
Model	416.98	19.98

```
mse = mean_squared_error(y_true=test_data[['Close']], y_pred=test_data['rf'])
print("Mean squared error for random forest is: ", mse)
```

```
Mean squared error for random forest is:  2135.7122222907983
```

```
mae = mean_absolute_error(y_true=test_data[['Close']],
                          y_pred=test_data['rf'])
print("Mean absolute error random forest is: ", mae)
```

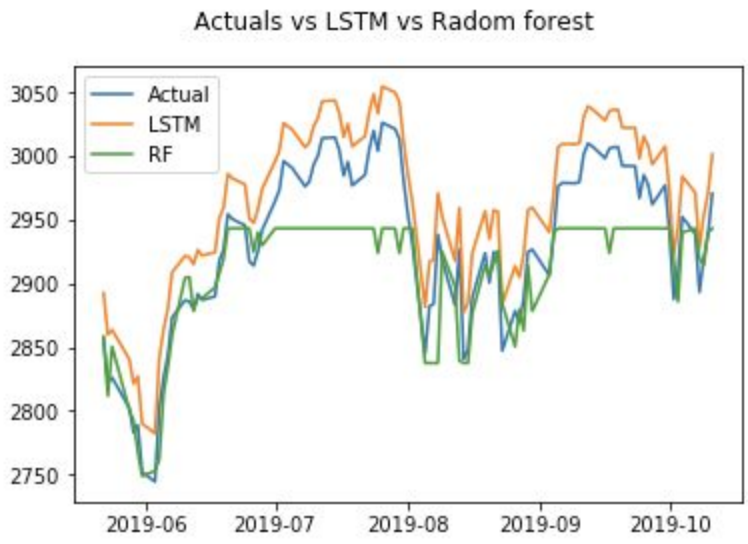
```
Mean absolute error random forest is:  33.97790048999998
```

```
# calculate MSE and MAE for LSTM

from sklearn.metrics import mean_squared_error, mean_absolute_error
lstm_mse = mean_squared_error(y_true=test_data['Close'],
                              y_pred=test_data['lstm'])

print("Mean squared error for LSTM is: ", lstm_mse)
```

```
Mean squared error for LSTM is:  416.9757557317105
```

```
lstm_mae = mean_absolute_error(y_true=test_data['Close'],
                               y_pred=test_data['lstm'])

print("Mean absolute error for LSTM is: ", lstm_mae)
```

```
Mean absolute error for LSTM is:  19.98159670765623
```



III. Methodology

Data Preprocessing

We preprocessed the data by splitting the data into the training and testing data. For LSTM, MinMaxScaler was used to scale and normalize the data.
For Arima, we took the logarithm of the input and applied time-shifting.

Implementation

Long Short-Term Memory (LSTM)

The LSTM architecture contains 3 layers which include;

1. 128 units and a sigmoid activation function,
2. Dropout (0.2)
3. Dense layer

The model was then compiled using mean-squared-error for loss and Adam for its optimizer.
The LSTM model was then fitted with the data after compiling.
Before training the data with the LSTM, the input was preprocessed by transforming and scaling the features of the data using MinMaxScaler.

Autoregressive Integrated Moving Average Model (ARIMA)

The parameters for Arima models are defined as;

- p: Number of lag observation included in the model
- d: Number of times the raw observations are differenced
- q: Size of the moving average window

In the project I used p = 0, d = 1, q = 4. I tried various parameters and arrived at Arima(0, 1, 4). It was important for the time-series data to be used in Arima to be stationary which means mean and variance should be constant over time.
With this idea, I had to perform some operations on the data in other for it to be stationary. I took the logarithm of the data, performed some time shifting and differencing. I eventually used the logarithm of the data for the Arima model.
I had to take the exponential of the fitted values to get the actual fitted values.

Refinement

The refinement process was mostly trial and error. To improve the model, I tried different layers, activation functions, number of epochs, changed the hyperparameters to see which is the most appropriate for the model. I played around with the dropouts, numbers of layers, normalization, and some parameters.

LSTM

Initial 1
Activation: tanh
2 Dropouts 0.2
1 Dense layer
Loss: mean squared error
Optimizer: Adam

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 128)               66560
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
_________________________________________________________________
dropout_3 (Dropout)          (None, 1)                 0
=================================================================
Total params: 66,689
Trainable params: 66,689
Non-trainable params: 0
_________________________________________________________________
```

MSE = 342631.79

MAE = 585.18

Initial 2

Activation: sigmoid, 256 units
Dense layer
Dropouts 0.2
Dense layer

Loss: mean squared error
Optimizer: Adam

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, 256)               264192
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 257
_________________________________________________________________
dropout_12 (Dropout)         (None, 1)                 0
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 2
=================================================================
Total params: 264,451
Trainable params: 264,451
Non-trainable params: 0
_________________________________________________________________
```

MSE = 186065.77
MAE = 431.15

Initial 3
Activation: sigmoid, 256 units
Dropouts 0.2
Dense layer

Loss: mean squared error
Optimizer: Adam

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 256)               264192
_________________________________________________________________
dropout_13 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 257
=================================================================
Total params: 264,449
Trainable params: 264,449
Non-trainable params: 0
_________________________________________________________________
```

MSE = 2730.549
MAE = 49.87

Final

Activation: Sigmoid
1 Dropout 0.2
1 Dense layer
Loss: mean squared error
Optimizer: Adam

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128)               66560
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 66,689
Trainable params: 66,689
Non-trainable params: 0
_________________________________________________________________
```

MSE = 416.97
MAE = 19.98

IV. Results

Model Evaluation and Validation

The best model which is LSTM with the following hyperparameters

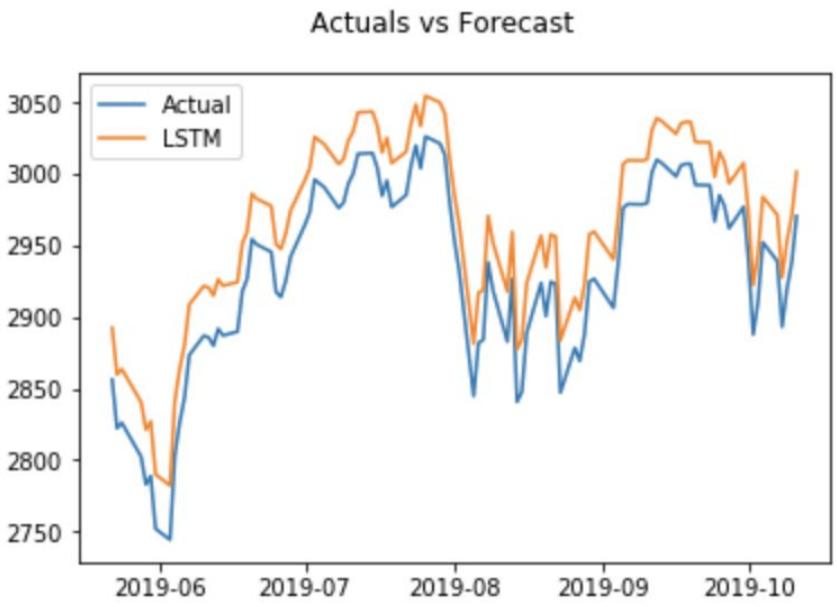
- Dropout: 0.2
- Activation: Sigmoid

From the metrics using Mean squared error and mean absolute percentage error, It is seen that LSTM performed well.
LSTM aligns well with the solution expected and performs better with the benchmark.

Even though the model metric is better than the benchmark, I do not believe the model is robust enough to be used in a real-life scenario as more data (sentiment) needs to be taken into consideration to ascertain its robustness. Also, the data has not been tested with various inputs and unusual inputs so, I do not think the model is robust enough. More update will be made to the project through to improve it and make it more robust.

See the chart below.

Benchmark/Model	MSE	MAE
Benchmark	2135.71	33.97
Model	416.98	19.98



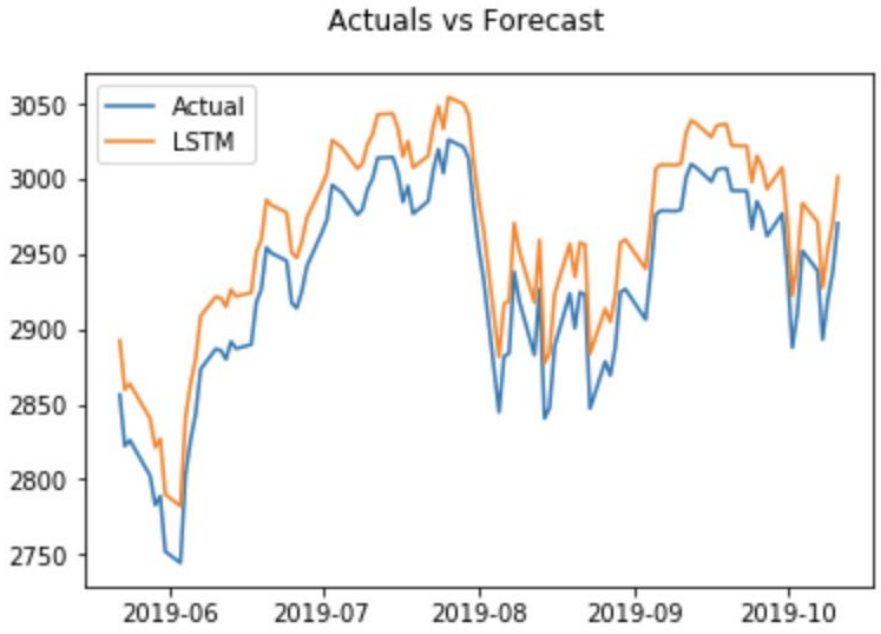
Justification

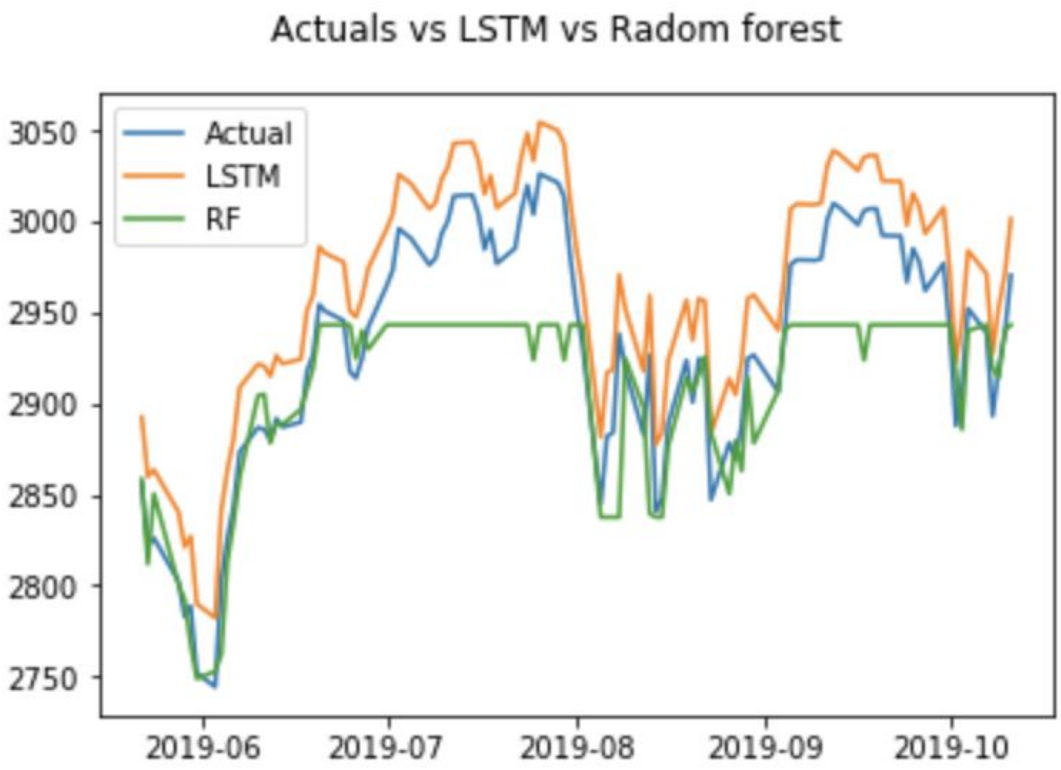
The model outperformed the benchmark and the other model used. Although the solution outperformed the benchmark, I do not believe that this solution is significant enough to have solved the problem this is because there are various factors that determine the dimensionality of the stock price apart from historical data. Hence to make the solution more robust, I need to take into consideration the other factors.

V. Conclusion

Free-Form Visualization

The chart below shows the forecast in comparison with the actual chart and the benchmark. The model predicts the upward and downward trends and it is not accurate, but the dimensionality is often predicted.





Reflection

The project was an interesting one for me as I was able to try out some ML techniques like LSTM and Arima.
In the project, I checked if there are null/NAN values in the project, Adj close was used in place of close. Then, important features for the model was harvested and used.
The data were also normalized to be used by the model. I experimented with different layers, hyperparameters and activation functions.
I found all aspect of the project in general interesting as I was learning new techniques and having hands-on experience with the techniques.
The difficult aspect of the project was using Arima. I had to leverage some resources to be able to understand Arima and implement it op the project.
It is important to note that the stock price cannot only be predicted using historical data as a lot more factors affect the price. The solution fits the problem to some extent, but not entirely as the stock price is affected by other factors.

Improvement

As I said before, the stock price is affected by other factors aside historical data hence, to improve this project I will use sentiment analysis on news site and twitter to get real-time information that can affect the price and use it in the project accordingly.
I can also use reinforcement learning to train an agent to predict the price.

References

https://en.wikipedia.org/wiki/S%26P_500_Index
https://en.wikipedia.org/wiki/Stock_market_prediction
https://www.fool.com/knowledge-center/what-is-the-sp-500.aspx
https://finance.yahoo.com/quote/%5EGSPC/
https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/
https://towardsdatascience.com/machine-learning-techniques-applied-to-stock-price-prediction-6c1994da8001
https://machinelearningmastery.com/time-series-forecasting-performance-measures-with-python/
https://otexts.com/fpp2/accuracy.html https://en.wikipedia.org/wiki/Mean_squared_error

https://www.kaggle.com/myonin/bitcoin-price-prediction-by-arima

https://towardsdatascience.com/machine-learning-part-19-time-series-and-autoregressive-integrated-moving-average-model-arima-c1005347b0d7

https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

https://ademos.people.uic.edu/Chapter23.html

https://www.kaggle.com/someadityamandal/bitcoin-time-series-forecasting

https://www.kaggle.com/myonin/bitcoin-price-prediction-by-arima

https://machinelearningmastery.com/time-series-forecasting-python-mini-course/

https://www.youtube.com/watch?v=7vunJlqLZok

https://www.kaggle.com/kp4920/s-p-500-stock-data-time-series-analysis

https://stackabuse.com/time-series-analysis-with-lstm-using-pythons-keras-library/

https://www.datacamp.com/commt_unity/tutorials/lstm-python-stock-market

https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average