# Capstone Project Report:

# Stock Price Predictor

Raj Dholakia

October 31, 2020

# Table of Contents

## Table of Figures

# I. Definition

## Project Overview

A real-world application of machine learning is in the world of trading and investing. The industry functions on predicting on a daily, even hourly basis. Estimations of the price of a stock vary from a few minutes to a few years. Naturally, with the aim to **predict** and presence of **large amounts of data**, the field has laid out the foundation to apply some machine learning algorithms.

Having said that, the role of machine learning in the field is yet part of active research. The uncertain nature of stock prices and the time-dependent aspect of the problem makes it a particularly difficult one to solve. Then again, considering the fact that the industry breathes money, a lot of research is well-funded and progress has been made. Even a decent prediction model, attracts a lot of attention in the market (Asadi *et al.*, 2012; Agarwal and Sabitha, 2017).

There are two broad ways one can go about predicting the price of a stock. The first is through **technical analysis**. The historical price data of a stock is used to predict its future price. The second being **fundamental analysis**, which uses unstructured textural information to understand the market sentiment and predict what will happen to the price of a stock. Information sources for the latter approach include news and social media (Subhi Alzazah and Cheng, 2020).

I decided to take on this problem due to my interest in finance and modest internship experience at a start-up that focuses on machine learning applications in intraday trading. I believe working on this project will give me a better understanding of the financial markets (finance in general) and time-series machine learning problems. As I am new to the world of machine learning and finance, I have decided to carry out technical analysis using the historical data of stocks.

I started with the ambitious goal of predicting prices of three stocks and later came down to **one**. The project proposal tries to encompass everything possible, but realistically not everything is possible.

> **I have narrowed the project down to training two different models and comparing their performance on `AAPL` stock data (in particular it's daily adjusted close price).**

I have learnt a lot from the many mistakes I have made during this journey and I intend to work on this project post submission as the Capstone Project for Udacity’s Machine Learning Nanodegree.

The project is broken down into three parts:

```
1. Exploratory Data Analysis
2. Data Preparation
3. Model Training and Testing
```

My focus has been maintained on ensuring the code is clean and modular, graphs are clear and notebooks, documents and directories are neat. Everything went smoothly till I reached the Model training part. When I realised what my first mistake of setting ambitious goals.

It is good to have ambitious goals, but it is better to ensure they are realistic. In the effort to cover everything I found interesting; I did not ration time to finish the project. Hence, I am going to focus on passing the project and after the submission continue building on the project.

## Problem Statement

**The aim of this project is to predict the long-term price trend of one stock with at least 90% accuracy.**[1] Two models' performances will be compared: `ARIMA` and `DeepAR`. The ability of the DeepAR model to predict ten months in the future will be observed. More importantly, the model's ability to predict how the stocks performed in 2020 (the year of the pandemic) will be observed. There are multiple options when it comes to time series forecasting and I have decided to do a univariate prediction. This is to bring in some simplicity to the project. Hence, `open, high, low, close prices` will not be predicted, only `adjusted close price` will used for the analysis.

In the end, the model should be able to able to predict a general trend of the time series. One particular graph that will give a visual indication of the solution being reached is the quantiles graph. The aim is to use 30-70% quantiles of the predictions to encompass the true time series.

## Evaluation Metrics

I will be using two main evaluation metrics to understand the model's performance.

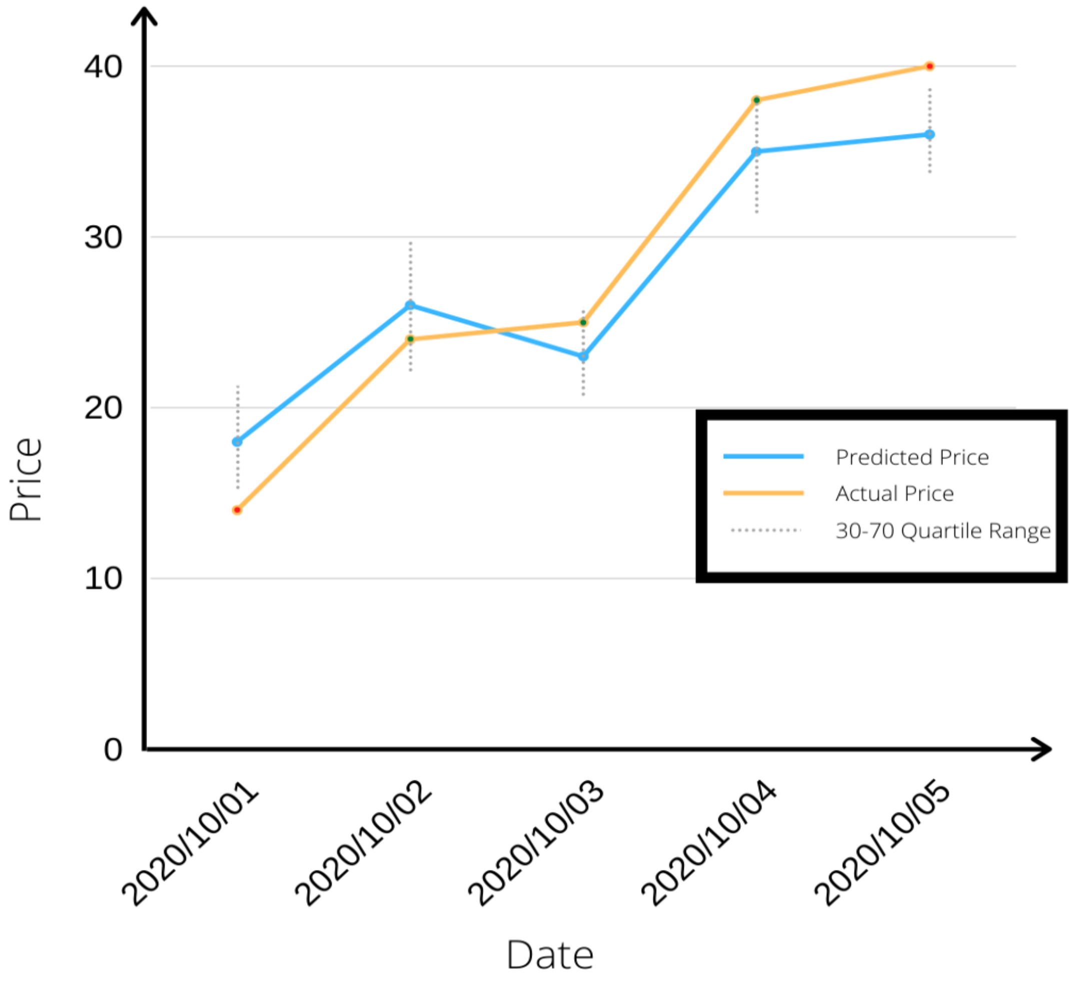
[1] *90% accuracy can be taken as 10% mean absolute percentage error.*

```
One will be quantitative and the other will be visual (can be converted to quantitati
ve):
```

1. **Mean Absolute Percentage Error (MAPE)**: It is the mean of percentage of absolute errors of the predictions. The following formula (code) explains how it is calculated ('MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)', 2006; Glen, 2011).

```
MAPE = np.mean(np.abs((y_preds - y_true)/y_true))
```

*Figure 1: Example Stock Price vs Time graph with predicted values*



2. **Percentage Points Correctly Predicted (PPCP)**: This is more of a visual indicator of how the model is doing. It is the percentage of actual points that lie in the 30-70% quantile range of the predictions.

In the example graph above (Figure 2), it is clear that 3 out of 5 points fall in the 30-70% quantile range. Hence, `PPCP = 3/5 = 60%.`

I came up with this metric as a solution to the problem of predicting for larger intervals. I intend to use this to understand if the model can make accurate predictions on long-term trends. However, some weakness of the metric would be its inability to give great results for predictions that have high variability (standard deviation). As a higher standard deviation would mean a larger area is covered by the predictions, the probability of the actual value to land within the 30-70% quantile range is higher. However, if the standard deviation is high, the model is not following any specific trend (up or down) but is just spreading in both directions, leading to an inaccurate measure of what is actually happening.

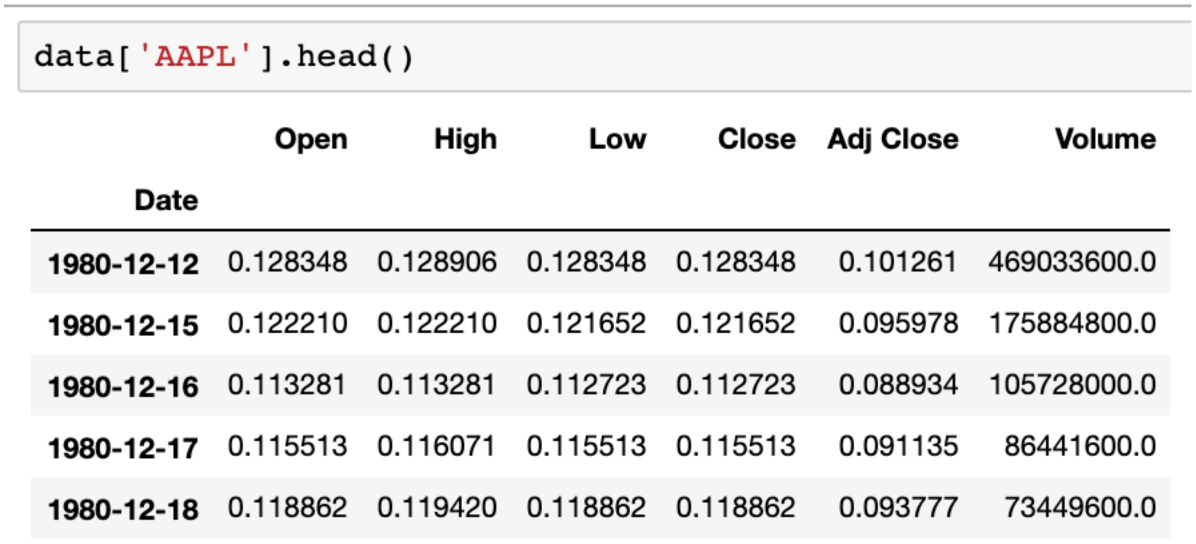
Hence, a combination of MAPE and Percentage Points will give a better understanding of how the model is performing.

Other metrics like the `mean squared error (MSE)` and `root mean squared error (RMSE)` can be used to further support MAPE. As MAPE is a percentage value it will be easier to compare it with other stocks of different prices. Additionally, as the stock price starts with low values (in the 20th Century) and starts rising over time, it is important to use a metric that is independent of price.

# II. Analysis

## Data Exploration and Visualisation

> Notebook 1_Exploratory_Data_Analysis



```
data['AAPL'].head()
```

| Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| 1980-12-12 | 0.128348 | 0.128906 | 0.128348 | 0.128348 | 0.101261 | 469033600.0 |
| 1980-12-15 | 0.122210 | 0.122210 | 0.121652 | 0.121652 | 0.095978 | 175884800.0 |
| 1980-12-16 | 0.113281 | 0.113281 | 0.112723 | 0.112723 | 0.088934 | 105728000.0 |
| 1980-12-17 | 0.115513 | 0.116071 | 0.115513 | 0.115513 | 0.091135 | 86441600.0 |
| 1980-12-18 | 0.118862 | 0.119420 | 0.118862 | 0.118862 | 0.093777 | 73449600.0 |

*Figure 2: Example Stock Price Data Image*

The data was first loaded and features of the data understood and explained in the notebook. Figure 2 contains the first five rows of `AAPL` stock price data. This includes meaning of `OHLC` prices and `Adj Close` price. All these prices are very strongly correlated to each other for `AAPL`. The data for the three stocks and/or indices were then visually inspected to determine the length of time series data to be taken. One can notice that most of the data before `2002` was almost constant when considering the changes post `2002` for Apple Inc. Furthermore, the decision to use `Adj Close` for project was taken after understanding different types of time series forecasts. To predict multiple prices using one model, multi-variate time series prediction is to be utilised, which is beyond the scope of this project.

One of the major anomalies of the data is the missing values for weekends and bank holidays. As it can be seen in the example above, data is missing for days and those are the times the stock market was closed. This is a concern in time series data analysis as it creates a time gap. Bank holidays especially are a problem as it becomes difficult to take into account all the holidays when considering over 20 years of data. This specific characteristic of the data needs to be managed carefully.

## Algorithms and Techniques

> Notebook 2_Data_Preparation
>
> Two algorithms are used in this project:
>
> 1. Auto-Regression Integrated Moving Average (ARIMA)
> 2. AWS DeepAR

A section in the notebook is dedicated to discussing the insides of each of the models. Not only was the working of the algorithm be understood, but also the varying formats in which both accept data were to be understood well. Data is then prepared for each of the models.

The train-test split for both the models is different. The train-test split being created for DeepAR will contain multiple time series, while for the ARIMA model one large time series will be split into train and test dataset. If the prediction length is maintained for both the models, the data split percent will be quite different. ARIMA model will be provided with a single training time series of around 218 months long and test time would include the last 10 months of 2019. Both the models will be tested on the first 10 months of 2020 (Jan 2020 - Oct 2020). This will ensure a more accurate comparison between the two.

The time series is to be made stationary to hold all assumptions made for the ARIMA algorithm. The test for stationarity (`Augmented Dickey-Fuller Test`) is used to determine if the time series can be assumed to be stationary. This discussion is done in Notebook `3_Model_Train_Test`.

On the other hand, DeepAR requires multiple time series in a `JSON Lines` format. The algorithm takes train and test data and trains a neural network to predict the `prediction_length` using a given `context_length`. Once the model is trained, we can also make predictions from a given timestamp (we will use this feature to get predictions from January 1, 2020).

The parameters taken by both the models are of great importance to how the model performs. The discussion on parameter selection can be found in Notebook `3_Model_Train_Test`.

### Data Anomalies

One of the major concerns in the data is the missing data for bank holidays and weekends. This is common as stock markets will not be open on that day. As we have daily data, it is necessary that it is taken care of. In notebook 2_Data_Preparation, I had decided to keep missing data and let DeepAR algorithm handle it. However, when training a DeepAR model, an error kept preventing the data from being read. After numerous attempts at ensuring the data is in the correct `JSON Line` format, I decided to remove all the `Nan` values. After removal of all of these, the model trained. I realised the hard way that one of the sources might have misguided me.

> Flunkert, V. et al. (2018) Amazon SageMaker DeepAR now supports missing values, categorical and time series features, and generalized frequencies | AWS Machine Learning Blog, Amazon SageMaker, Artificial Intelligence. Available at: https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-deepar-now-supports-missing-values-categorical-and-time-series-features-and-generalized-frequencies/ (Accessed: 29 October 2020).

## Benchmark Model

The benchmark models are the results obtained by Nagesh Singh Chauhan in his analysis of Altaba Inc. stock from *1996–04–12* till *2017–11–10* (Chauhan, no date). He managed to get a MAPE value of **3.5%**, which can be said to be **96.5% accuracy**, using a well-tuned **ARIMA model**. In this project, the goal will be to get the MAPE value to be less than 90%.

This is a very particular example and it could turn out of that the results obtained are not as expected. I will be identifying the shortcomings of the analysis.

# III. Methodology

## Data Pre-processing

Time series data usually do not require pre-processing, apart from management of missing values. In our case, we need to manage missing values and ensure the indexing of the values is done correctly. While, the latter has been a recent discovery, the former has been addressed in Notebook `2_Data_Preparation`.

> Deeper discussion in Data Anomalies

First all the missing dates were added to the series with `Nan` values. Then these missing values were filled with interpolated values. As we are dealing with a time series problem, interpolation is the best option. Further investigation can be done on the effects of higher order interpolation on the performance of the models, but that is beyond the scope of this project.

## Implementation

I have clearly explained every step taken in all the three notebooks. The second metric decided upon is slightly complicated as I have not been able to find resources that teach how to get quantiles from an ARIMA model. Hence, it is now limited to a graphical representation. There are numerous helper functions created throughout the notebooks. These functions can be found in `helper_functions.py` python script. They all contain a docstring and can be imported into a Jupyter notebook and used.

## Refinement

Many modifications have been made to the initial plan of action. The solution to the problem did not come easy. The process of improving performance of a model is a long process and I have just scratched the surface by creating one and bringing small improvements to it. I intend to work on improving the DeepAR model's performance post-submission of this capstone project. I aim to ensure I am able to solve the problem statement and achieve at least 90% accuracy. Currently, I have refined the following two (one technique and one model performance):

1. Data processing: Data processing has been a long process of ups and downs. It started with preparing the first set or type of data for ARIMA model. As I got to know more about the model, my dataset kept changing. I had started with the expectation of using the same data for the ARIMA modelling process as use in DeepAR. However, after the first obstacle, it seemed like a better idea as I will be able to train and test the model on multiple time series. However, ARIMA seems to throw errors and going around to change it back into a pandas series, I would have to create one more function. Instead, the process was shorter if the data for ARIMA model were saved locally and loaded in the whichever notebook it is needed. Data processing can be further refined as there are places where I have to recreate the same data again. This can be avoided saving the data locally in an easily readable format.

2. ARIMA model performance: The MAPE value was improved from `19.1%` to `15.3%` by just manual hyperparameter tuning. Can be improved by trying different sets of parameters and using different ways to make the time series stationary.

# IV. Results

## Model Evaluation and Validation

Model validation is done with the test data saved with the training data. DeepAR model takes the test data and uses it to train the neural network, while we use the test data for the ARIMA model to train and measure the performance of the model. Once we have models performing well (around `10% MAPE Value` or 90% accuracy), the models will be tested on the `275 days` or `10 months` of data of 2020. The metrics remain the same as models are validated and tested.

Currently, the ARIMA model has an accuracy of `85%` and the DeepAR model has of `80.1%` for the same test data. Better results can be achieved with more hyperparameter tuning.

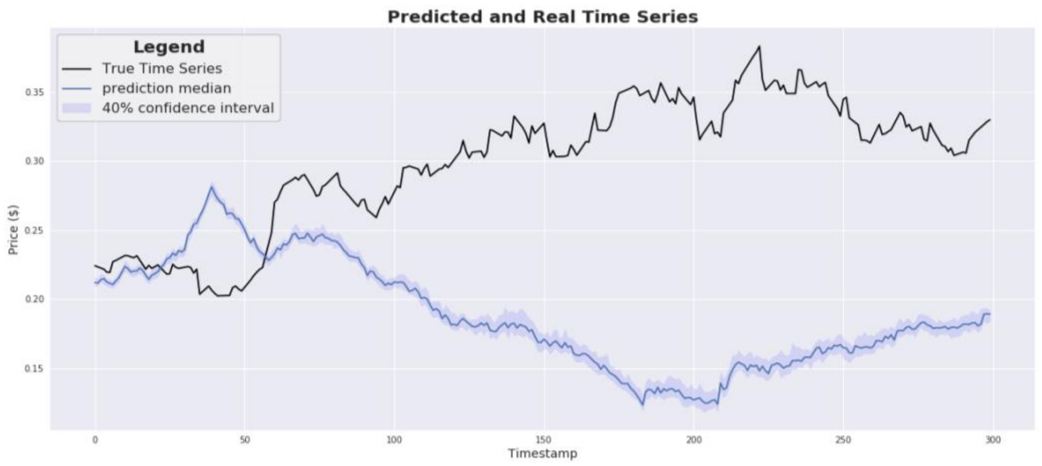
The ARIMA model has higher chances of running into a bias as it has only one test set. Comparatively, the DeepAR model has `five` different time series sets with each having a test time series. Hence, an average MAPE value of how the model performs in each of those training series will give a more accurate understanding of DeepAR's performance. On the other hand, the ARIMA model will need to be tested on completely unseen data (of year 2020) to gauge its performance accurately after hyperparameter tuning.

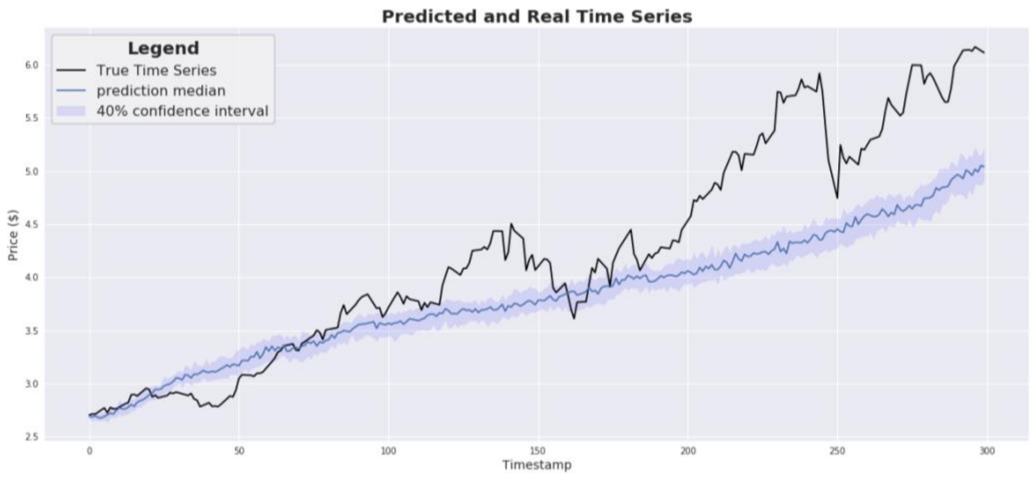
## Justification

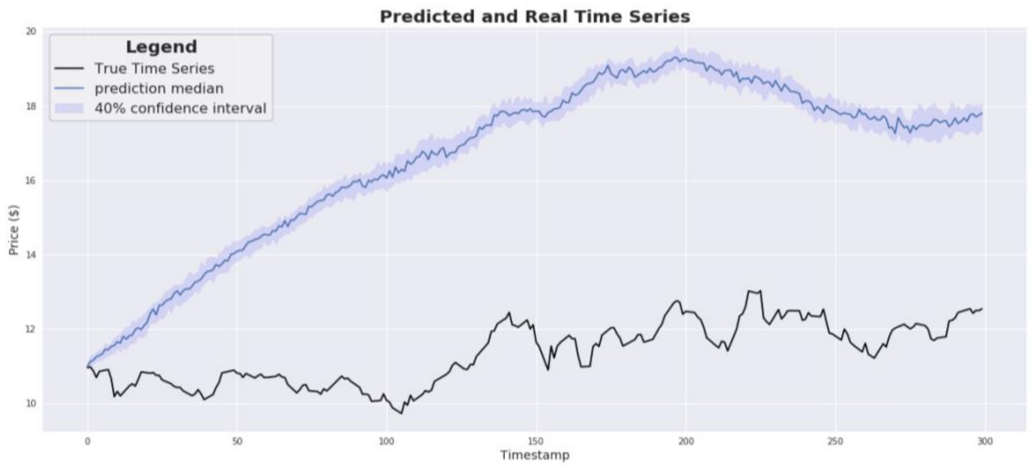
Comparing the results obtained to the benchmark model, there is 10-15% difference in the MAPE. The hyperparameters of the benchmark model have been tuned. Firstly, the stock is not very widely traded and doesn't have a volatile nature. `AAPL` is a reliable but volatile stock, and it can be difficult to predict the price compared to Altaba Inc. It might as well be the case that Mr. Chauhan found a stock that works best for the model he has created, which led to such a low MAPE. There could be a bias that we might not be aware of. Hence, it is important to look at other models that have attempted to predict stock prices.

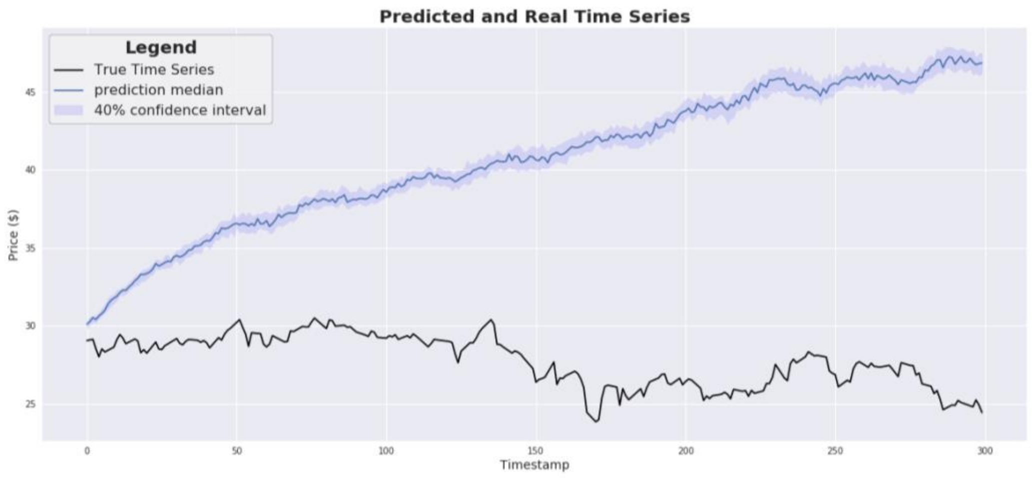
## V. Conclusion

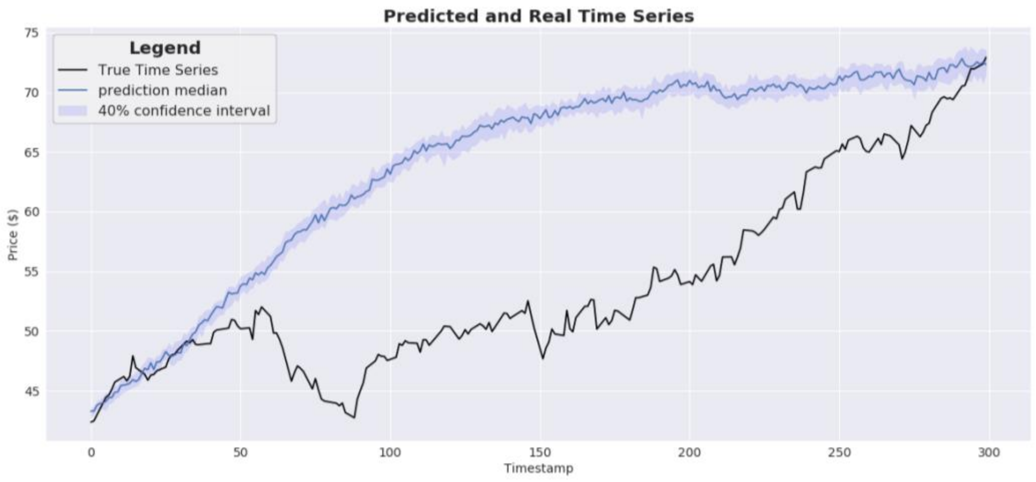
### Free-Form Visualisation











The graphs above were generated from the predictions made by Baseline DeepAR model.

## <u>**Reflection**</u>

I believe my goal had been ambitious from the beginning. However, I have worked hard to implement everything to the best of my ability for the time I had in my hand. I will continue giving time to this project to improve it and make it something presentable to other Aspiring Machine Learning Engineers. Working on this project has given me a better understanding of what kind of problems a Machine Learning project can face (though at a smaller level). It all starts by defining a problem statement and coming back to it and trying to improve it. Some exploratory data analysis can provide some fresh ideas and the problem statement can be tuned to account for changes. Setting a methodology will ensure the focus on the project remains the same throughout. Results are the best place to learn about what could be improved in the data processing and implementation process (even if they are incomplete). Machine Learning is all about jumping multiple obstacles, reaching the finish line, restarting behind the obstacle and evaluating till we understand what is the best way to cross an obstacle.

## Improvement

There is a lot of work needed to **completely** finish the project. From better data handling technique to improved model tuning, all aspects of the implementation need to be worked on. I have created multiple functions to prepare the data to be fed into the algorithms, to plot graphs to understand performance, evaluate a model's performance with the predictions.

First improvement that can be made should be to indexing of the data. The data received from a predictor uses integers as an index and the indices in the training data are in DateTimeIndex format. If indices need to be in the same format, it will enable better understanding and smoother integration of real and predicted values.

Second improvement that can be made would be to find a way to manage missing data. One of the ways I have tried to get around it is by linear interpolation of the missing values. I could try to instead remove the missing data and observe how the model performs. As we are looking at a time series which does not have a very seasonal trend, the latter could work better.

Thirdly, find an efficient way to do hyperparameter tuning for the ARIMA model. For the ARIMA model, I started by trying to find the optimal value for the lag, which requires analysis of the data **and** training and testing the model. All the analysis methods used provide a tentative value of `p` and `q`, which can be leveraged to find the optimal values by modelling and testing.

Finally, more examples of time series analysis of stock prices needed to be looked at to understand if the current benchmark model provides a realistic goal. This is to be ensure that the benchmark model provides a robust and accurate representation of an algorithm's performance. Similar results can be obtained for other stocks or indices.

> Another note, all the functions can be neatly packed in a `helper_functions.py` file and accessed in the notebooks to make the notebooks more presentable.

# References

Agarwal, U. and Sabitha, A. S. (2017) 'Time series forecasting of stock market index', in *India International Conference on Information Processing, IICIP 2016 - Proceedings*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/IICIP.2016.7975381.

Asadi, S. *et al.* (2012) 'Hybridization of evolutionary Levenberg-Marquardt neural networks and data pre-processing for stock market prediction', *Knowledge-Based Systems*. Elsevier, 35, pp. 245–258. doi: 10.1016/j.knosys.2012.05.003.

AWS (2019) *Machine Learning with Amazon SageMaker*, *Amazon Web Services, Inc.* Available at: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-mlconcepts.html (Accessed: 7 October 2020).

Banton, C. (2019) *An Introduction to U.S. Stock Market Indexes*, *Investopedia*. Available at: https://www.investopedia.com/insights/introduction-to-stock-market-indices/ (Accessed: 4 October 2020).

Bourke, D. and Neagoie, A. (2020) *Complete Machine Learning and Data Science: Zero to Mastery | Udemy*, *Udemy*. Available at: https://www.udemy.com/course/complete-machine-learning-and-data-science-zero-to-mastery/ (Accessed: 7 October 2020).

Chauhan, N. S. (no date) *Stock Market Forecasting Using Time Series Analysis*. Available at: https://www.kdnuggets.com/2020/01/stock-market-forecasting-time-series-analysis.html (Accessed: 8 October 2020).

Glen, S. (2011) 'MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)', in *SpringerReference*. doi: 10.1007/springerreference_6919.

'MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)' (2006) in *Encyclopedia of Production and Manufacturing Management*. Springer US, pp. 462–462. doi: 10.1007/1-4020-0612-8_580.

*Stock Market Index: Meaning, Importance, NSE & BSE and more* (2020) *Defmacro Software Pvt. Ltd.* Available at: https://cleartax.in/s/stock-market-index (Accessed: 4 October 2020).

Subhi Alzazah, F. and Cheng, X. (2020) 'Recent Advances in Stock Market Prediction Using Text Mining: A Survey', in *E-Business [Working Title]*. IntechOpen. doi: 10.5772/intechopen.92253.

*Yahoo Finance – stock market live, quotes, business & finance news* (no date). Available at: https://in.finance.yahoo.com/ (Accessed: 2 October 2020).