Introduction

This report is a detailed documentation and guide to my personal predictive modeling project. Much of my inspiration in doing this project came with an interest in the dynamics of trading volume in the stock market, and how it may affect price movements. I wanted to see how a stock reacted to high and low trading volumes and if any patterns or relationships could be found. To delve deep into this analysis, I decided to use Tesla, one of the most volatile and dynamic stocks in recent years. Tesla's peculiar trading characteristics and its historic fluctuation in prices presented a great case study in testing predictive modeling techniques. Focusing on Tesla allowed me to delve into more meaningful insights regarding its behavior, particularly concerning trading volume, and whether such a relationship exists that is useful for making reliable price predictions. The relationships are analyzed using various methods, techniques, and approaches to data analysis, statistical modeling, and machine learning. This work tries to explore the predictive modeling process-step by step, starting from data pre-processing up to model evaluation. Meanwhile, visualizations and interpretations of the results also form part of the project in order to make such findings accessible and useful.

Before that, I had to find, format, and visualize my data. I retrieved a datasheet of Tesla's stock price every day from Jan. 2015 to April 2024, including the Open Price, Close Price, High, Low, Adjusted Close Price, and Volume. Next, I used python to clean up the data, remove any missing rows, and reformat the date to be compatible with the APIs I used later on. I also decided to look into SQL and gain some knowledge on how it works and how it is used. With some basic queries, I made a new sheet including a mix of insights, including busy trading days, greater than 5% price increases, and more. This allowed me to not only simplify my data into more tangible results, but use this data to research the real-world events that emerged and happened on some of these specific days and relate to why this happened and how. This, although time-consuming, helped me understand how the market reacts to specific things, like: predicted earnings, predicted sales, product launches and customer satisfaction, specific mandates such as the California Zero-Emission Mandate, inclusion in the S&P 500, etc. All this showed me how volatile and informative the stock/company and volume is respectively.

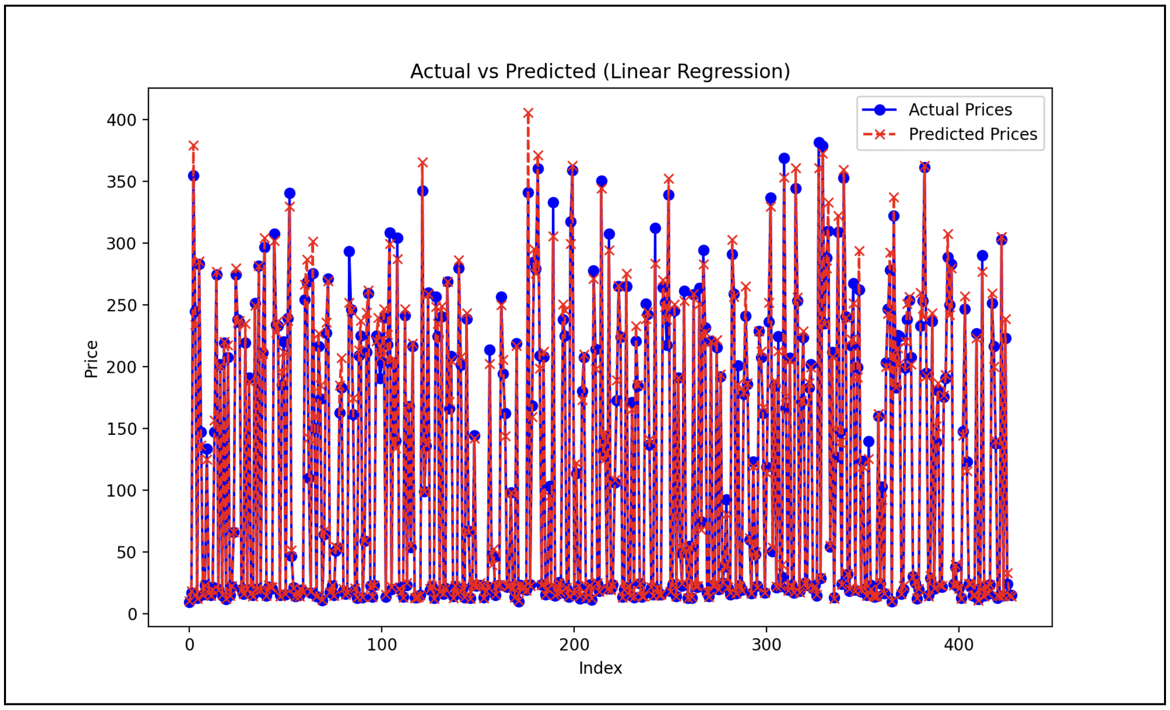
With that in mind, I got heavily interested in predictive modelling and if I can use volume to predict future price. I had to choose the best models. In Google Workspace, I created graphs, tables, and heat maps, to find patterns and trends. This would help in my strategy as to which type of modelling is best. These visualizations revealed that while volume alone might not predict price movement directly, periods of exceptionally high or low volume often coincided with significant price swings. This led to me picking models which were heavily invested in non-linear and temporal models. This obviously made sense, stock is not linear, the market is not linear, even if it may seem like that sometimes.

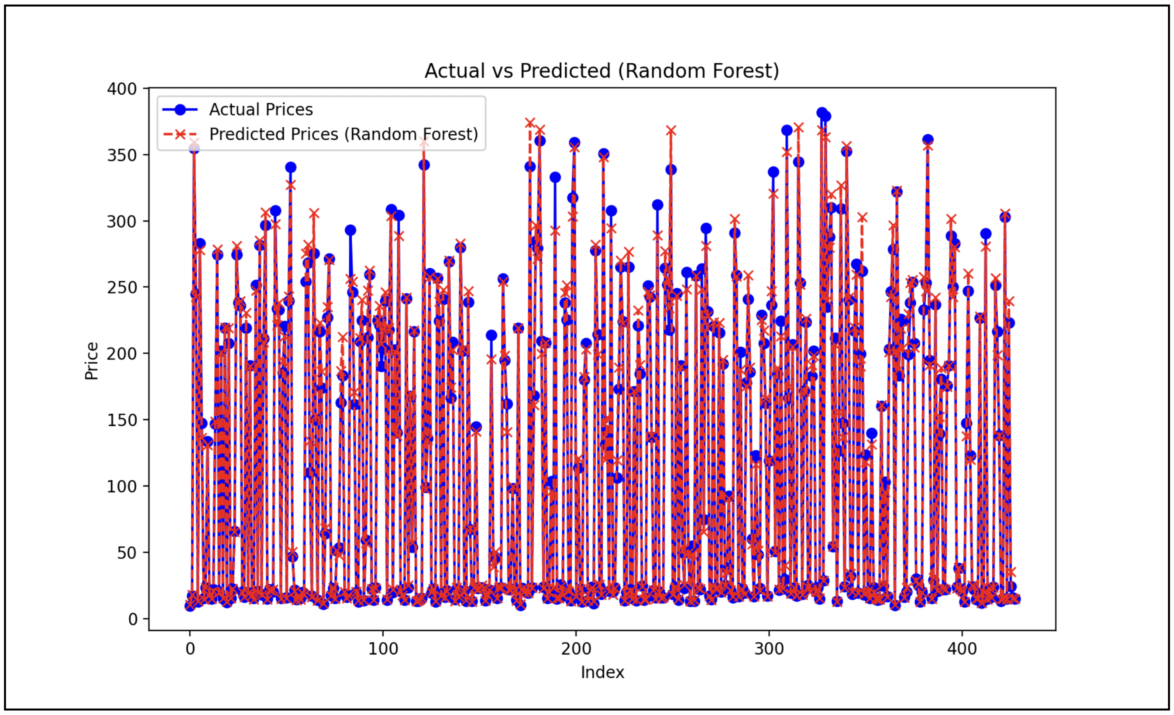
So finally, I selected the three modelling approaches which I believe were best fit for my skill and knowledge set, as well as for the stock:

1. Random Forest: An ensemble learning method based on decision trees. It works by building multiple decision trees during training and outputting the mean prediction, for regression tasks. This reduces overfitting and improves accuracy by averaging the results of individual trees. Because of this average result, it is best suited for the market, as the stock market is inherently noisy and full of ups-and-downs.
2. LSTM is a complex modelling network specifically designed for time-series data as it has memory cells which keep track of previous data and activity, ideal for modelling stock prices and trends. As Tesla has been volatile, LSTM is perfect for complex graphing and predicting.
3. XGBoost: A more complex and efficient implementation of Gradient Boosting, combining multiple decision trees, kind of like Random Forest, while correcting errors. This is ideal because it excels in non-linear modelling and ability for changeable hyperparameters, measures that optimize a model, which will be explained later.

Random Forest

All these models produced graphs, predictions, and comparative data and accuracy. Let's start with Random Forest:



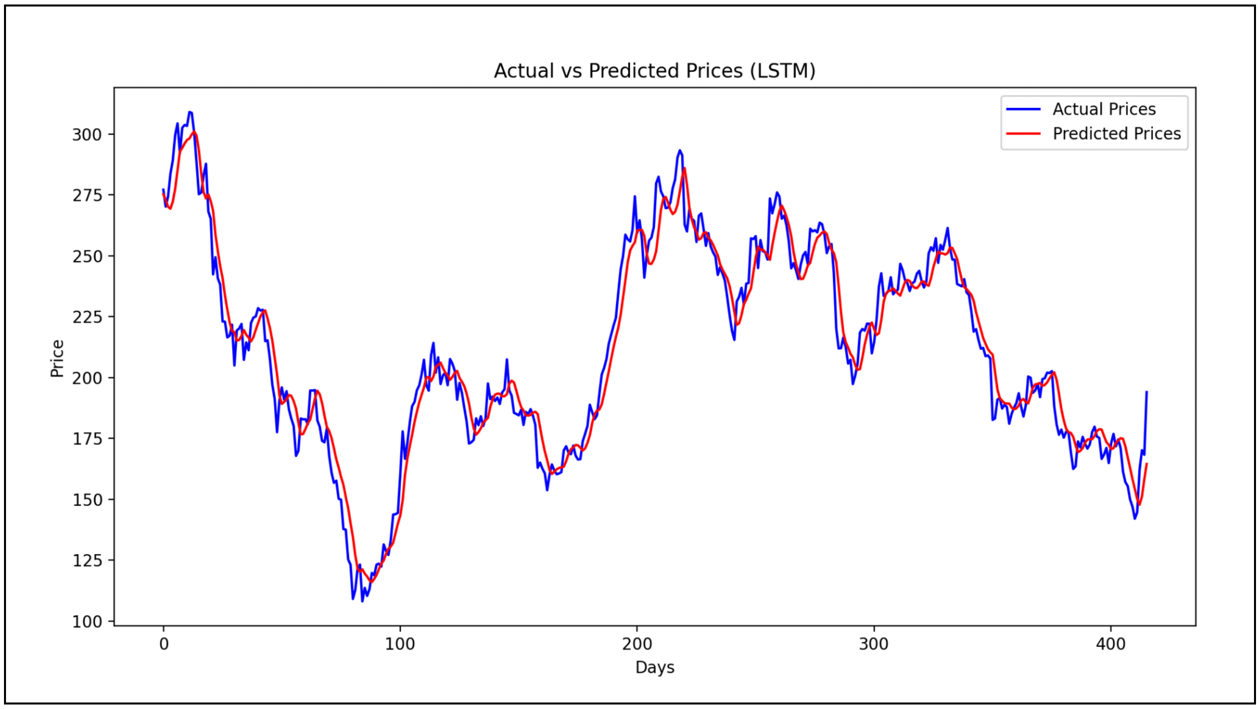


These two graphs are displays of the actual prices and the predicted prices. The first graph was obtained from linear regression and the second one was from Random Forest. I decided to use linear regression as a control. It is primarily used for less noisy and more linear datasets, not very ideal for a stock. But from a quick visual, most error-filled predictions were minimized when using Random Forest, proving my previous research and strategies.

Random Forest in simple terms, makes multiple decision trees, conditionals if you may (if this then that…). These trees are based on different features, these features are mainly financial. Some which I used are price movement, price movement percent, moving price average, and moving volume average. The model goes further, getting more technical, resulting in several decision trees. These trees are then compared and contrasted, optimized with previous data, and a final prediction is made. This is inevitably recursed, farther in time you go, more data there is to predict with.

The accuracy of Random Forest was 72.90%, a decent accuracy, with plenty more room to optimize and change.

LSTM (Long Short-Term Memory)

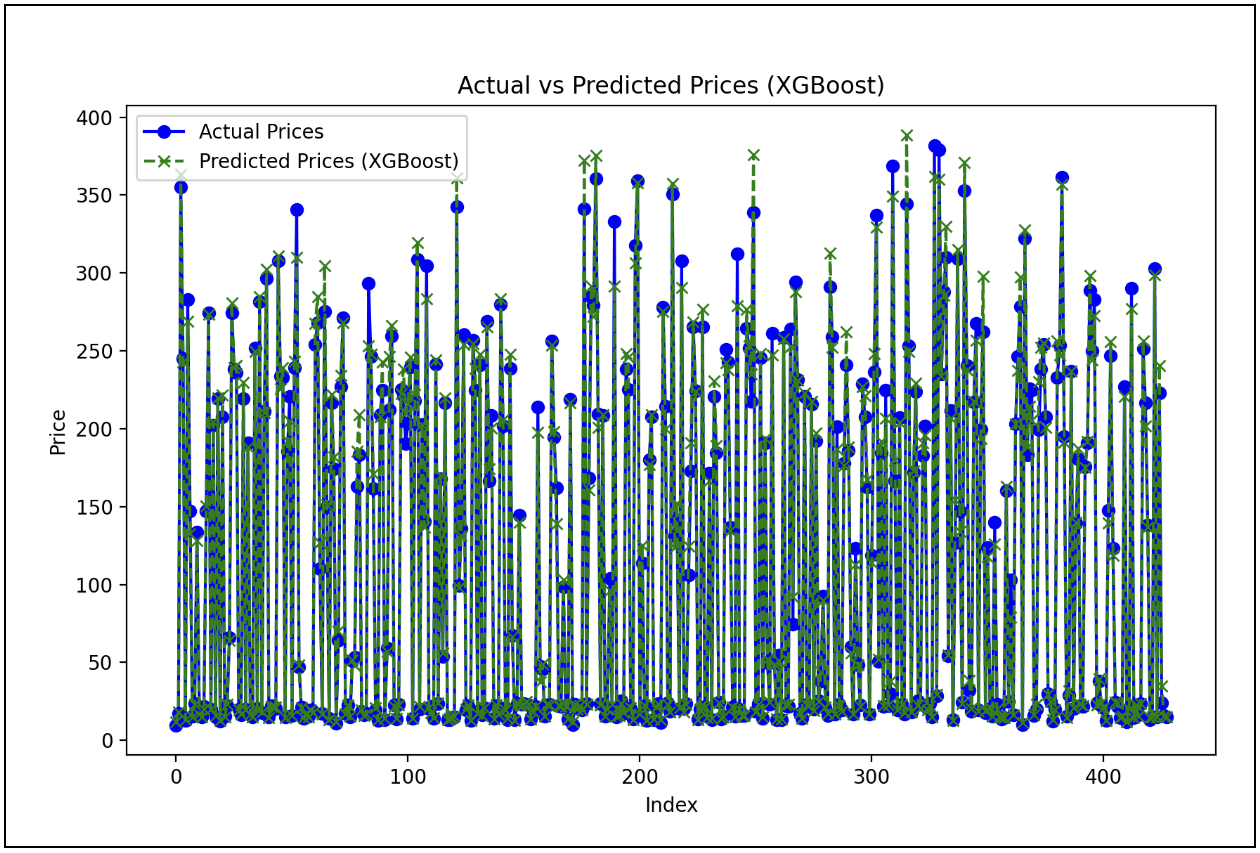


LSTM works kind of like taking notes, at least for some. Pretend you have a notebook, this is your data. This book gets filled with stuff on day one and on day two, you take a look at your previous data (notes) and remove any useless and keep everything important. Day 3 passes and you do the same thing, rinsing and repeating. Now, how the prediction works is in between each day, after the sorting is done, a prediction based on the notebook (data) is made.

This graph is another representation of the accuracy of the prediction. Why is the graph different? This mainly has to do with how the prediction is made. The Random Forest aims to predict specific prices which is why the vertical line graph is more suitable for showing the difference clearly. The LSTM predicts patterns, increasing and decreasing movements, not so much specific prices which is why a comparison to the movement of the stock like this is better.

The accuracy for LSTM was around 95%, a significantly more accurate model.

XGBoost



Finally, XGBoost. Quite plainly, it makes predictions, determines how off the actual price it was and predicts again, continuously doing this. It continuously makes models of itself until all the predictions are made.

One interesting thing is hyperparameters.

```
tesla_grid = {
   'n_estimators': [50, 100, 200],
   'learning_rate': [0.01, 0.1, 0.2],
   'max_depth': [3, 5, 7],
}
```

These are specific measures which are placed on how strong the model should be.

- "n_estimators" is the number of trees that will be used in the model to predict the price.
- "learning_rate" is how fast the model predicts, a higher rate tends to lead to a larger impact on the prediction and vice versa
- "max_depth" is the largest height of the trees. A higher depth, more insight, analysis, and accuracy and vice versa.

The accuracy of this model was around 70%.

With all these models however, there is something to keep in mind. Overfitting. It is when a model ends up memorizing the data instead of learning from it and this leads to overly accurate predictions. This is not ideal because for predicting future data, the model will have problems as it has not really understood the past data and has instead just memorized what has happened. Now, the threshold for what is considered overfitting is very ambiguous. For the majority of data analytics, if the difference between the test data accuracy and future prediction accuracy is significant, there is a high probability of overfitting. With these three models, it is difficult to determine this as I have not yet indulged enough into accuracy and overfitting in general. However, overfitting has other signs, which are not present in these models. This includes detailed changes in the graph and not a prediction of general trends, fitting hyperparameters which I have well chosen, and a large enough dataset that it cannot easily memorize.