6CCS3PRJ Final Year Project

Analysis of Machine Learning Algorithms and their Stock Market Prediction Capabilities

Final Project Report

Author: Zebin Liao

Supervisor: Peter McBurney

Student ID: k20074247

April 6, 2023

Abstract

Stock prediction and AI has always been a relevant topic in the field of finance and interesting topic in computer science. This project aims to understand and test the capabilities of different machine learning algorithms when used as a stock market prediction tool using historical stock data. The project will involve the implementation of data organisation and manipulation as well as scripting for the prediction algorithms to function in desired ways. The project will involve both next-day price value prediction(regression) and stock trend prediction(classification), as well as using different types of input data sets to test the output of the algorithms. The evaluation of these algorithms will involve mathematical derivations and graphical analysis. This project aims to evaluate the reasoning behind such data, how improvements can be made to the model as well as an analysis of their part to play in the future of stock prediction.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Zebin Liao

April 6, 2023

Acknowledgements

I would like to thank my supervisor, Professor Peter McBurney, who has kindly supported me throughout the project and provided me with feedback. As well as my family who have supported me throughout my studies.

Contents

Chapter 1

Introduction

This project aims to understand and test the capabilities of different machine learning algorithms when used as a stock market prediction tool using historical stock data.

The project will involve the implementation of data organisation and manipulation as well as scripting for the prediction algorithms to function in desired ways. There will mainly be 2 types of data used as the training set when predicting, that being closing price correlated data(open price, high, low, volume, etc...) as well as stock momentum data calculated from the data stated previously. The algorithms themselves will be borrowed from several python libraries(more on that in the design). The project will involve both next-day price value prediction(regression) and stock trend prediction(classification).

The evaluation of these algorithms will involve mathematical derivations and graphical analysis. This project is not an attempt at creating an algorithm that is capable of being used when trading independently(other than being used as references), it is simply a comparison of which algorithms are best suited and their corresponding positives and negatives, the reasoning behind such data, how improvements can be made to the model as well as an analysis of their part to play in the future of stock prediction.

The machine learning algorithms include:

- LSTM
- Random Forest

- Linear regression

Chapter 2

Background

Within the field of finances, predicting stock trends has always been a relevant topic. Currently, human brokers still prove to produce better prediction results than artificial intelligence, mainly because of the ability to predict human behaviour, as well as metric interpretation. However, The use of machine learning in the stock exchange has increased dramatically in recent years, as they are able to process historical data, utilising news and other relevant information at a faster speed.

This project plans to test the accuracy and efficiency of different machine-learning algorithms when predicting future stocks, this will help benefit the judgement of understanding which algorithms are better when they are used in similar topics in the future.

The project also hopes to dive into the reason for these results being produced in relation to their corresponding algorithm, finding a correlation between the results produced.

2.1 References

Payal Soni, Yogya Tewari and Prof. Deepa Krishnan —– Machine Learning Approaches in Stock Price Prediction: A Systematic Review, 2022
https://iopscience.iop.org/article/10.1088/1742-6596/2161/1/012065/pdf

Md. Mobin Akhtar, Abu Sarwar Zamani, Shakir Khanc, Abdallah Saleh Ali Shatat, Sara Dilshad, Faizan Samdani —– Stock market prediction based on statistical data using machine learning algorithms, March 2022
https://doi.org/10.1016/j.jksus.2022.101940

Suryoday Basak, Saibal Kar, Snehanshu Saha, Luckyson Khaidem, Sudeepa Roy Dey —– Predicting the direction of stock market prices using tree-based classifiers, January 2019
https://doi.org/10.1016/j.najef.2018.06.013

Casey Murphy —– What Technical Tools Can I Use to Measure Momentum? June 30, 2022
https://www.investopedia.com/ask/answers/05/measuringmomentum.asp

Adam Hayes —– On-Balance Volume (OBV): Definition, Formula, and Uses as Indicator, July 08, 2022
https://www.investopedia.com/terms/o/onbalancevolume.asp

Brian Dolan —– MACD Indicator Explained, with Formula, Examples, and Limitations, March 15, 2023
https://www.investopedia.com/terms/m/macd.asp: :text=The%20MACD%20line%20is%20calculated,for%20buy%20or%20sell%20signals.

Avijeet Biswal —– Stock Price Prediction Using Machine Learning: An Easy Guide!, March 5, 2023
https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning

Katherine (Yi) Li —– MLOps Blog - Predicting Stock Prices Using Machine Learning, 25 January 2023

https://neptune.ai/blog/predicting-stock-prices-using-machine-learning

Lazy Programmer — Predicting Stock Prices with LSTMs: One Mistake Everyone Makes (Episode 16) May 2021
https://www.youtube.com/watch?v=Vfx1L2jh2Ngt=184s

Greg Hogg — Stock Price Prediction Forecasting with LSTM Neural Networks in Python, June 2022
https://www.youtube.com/watch?v=CbTU92pbDKw

Chapter 3

Report Body

Understanding past stock data is crucial to the project, as to know what predicting historic stock data means, one has to first understand how stock prediction functions traditionally and how historic stock data is represented.

Historical stock data is usually presented as time series data having columns being:

- Closing price
- Opening price
- High
- Low
- Volume
- Time unit

(usually there are more attributes stored, however, these are the mains ones that this project is revolved around)

The traditional ways of forecasting stock data usually involve Fundamentals Analysis where a stock's True Value is calculated from the company's spending, profits and capital, etc... some techniques also involve news gathering. It also involves Technical Analysis, which tends to utilise time series forecasting using different techniques such as momentum indication data to predict the probabilities in trends of future stocks.

The project will use machine learning algorithms as well as Technical Analysis methods(more on this in Prediction using momentum data) to predict future stock prices, making calculations using data presented in the format above. Fundamental Analysis is not used as it requires separate data and/or machine learning techniques to execute which is out of the scope of this project.

The machine learning algorithms used are:

- LSTM(long short term memory)
- Random forest regression
- Linear regression

These algorithms are chosen because they include, a simple algorithm that gives us a baseline of prediction results(Linear Regression), an algorithm slightly more complex and has been a popular prediction method in the past(Random Forest), as well as an even more sophisticated model that utilises recurrent neural networks(LSTM), it will hopefully show the difference between the results when predicting data, as well as providing us information about how the base models can be improved upon with techniques such as Ensemble Learning.

Usually there are 2 types of prediction when it comes to stocks, those being next-day stock value predicting, which involves regression algorithms, as well as next-day stock trend predicting, which involves classification algorithms. This project explores both types of prediction, as both will provide insight into this topic. Therefore, the classification variants of the above algorithms will also be used when predicting trends:

- LSTM(modified for classification)
- Random forest classification
- Linear classification

3.1 Data Preperation

The historic stock data used in this project will be from Yahoo Finance in the Python Pandas library.

Data will be taken from a certain chosen stock and will fetch data from 2012-1-1 to the current

date. This data will be split into a training set(from 2012-1-1 to 2021-12-1) and a test set(2021-12-1 to the current date) which is roughly 80% to 20% split. Because this is time series data, a Scikit-Learn train test split would not work for the prediction as it disregards the time indices. Therefore the train test set is to be split manually.

Data will be pruned in case there are any invalid or NaN fields, it will then be normalised by putting it through a min max scaler where the feature range is 0 to 1.

There are 2 types of input data that will be evaluated upon in this project:

- Default price-related data(opening price, high, low... etc)
- Derived momentum data(RSI, MACD, price rate of change... etc)

This is to compare the difference between using price data as input and data derived using Technical Analysis techniques, and what that says about the prediction model and future improvements. Both of these data sets will be put through the models separately.

3.1.1 Price Related Input Data

Price-related data input would be taken directly from the original data set fetched from yahoo finances, the X Train set will consist of:

- opening price
- high
- low
- volume

The data would be ordered by the time they were recorded.

3.1.2 Momentum Indicator Input Data

A number of stock momentum indicators are used as separate input data:

- RSI
- Stochastic Oscillator
- Williams %R

- Moving Average Convergence Divergence (MACD)
- Price Rate Of Change
- On Balance Volume

RSI is a popular momentum indicator that determines whether the stock is overbought or oversold. A stock is said to be overbought when the demand is more than its true value. This usually means that the stock is overvalued, and the price is likely to go down. A stock is said to be oversold when the price goes down sharply to a level below its true value. This is a result caused due to panic selling. RSI ranges from 0 to 100, and generally, when RSI is above 70, it may indicate that the stock is overbought and when RSI is below 30, it may indicate the stock is oversold.

The formula used here is:

$$RSI = 100 - \frac{100}{1 + RS}$$

Where RS = Relative Strength, which can be calculated through Exponential Weighted Moving Average(more on this in Implementation).

Stochastic Oscillator follows the speed or the momentum of the price. It assumes that the momentum of a stock change before its closing price changes. It can be calculated from the [Low] and [High] columns of the stock data.

The formula used here is:

$$K = 100 * \frac{(ClosingPrice - Low)}{(High - Low)}$$

where Low and High are the lowest and highest prices with a period of time prior to the day being calculated(for this project it's 14 days).

Williams %R is an indicator quite similar to the Stochastic Oscillator as it is also calculated from the highs and lows of the price. It ranges from -100 to 0. When its value is above -20, it indicates a sell signal and when its value is below -80, it indicates a buy signal.

The formula used here is:

$$R = \frac{(High - ClosingPrice)}{(High - Low)} * (-100)$$

Where where Low and High are identical to Stochastic Oscillator.

MACD stands for Moving Average Convergence Divergence. When the MACD goes below a certain threshold, it indicates a sell signal. When it goes above the threshold, it indicates a buy signal.
The formula used here is:

$$MACD = EMA12(ClosingPrice) - EMA26(ClosingPrice)$$

Where EMA stands for Exponential moving average, which can be calculated from the closing price of the stock over several days.

Price Rate of Change is the most recent change in price with respect to the price n days ago.

The formula used here is:

$$PROC = \frac{ClosingPrice - ClosingPrice_N}{ClosingPrice_N}$$

Where ClosingPrice_N is the closing price of a stock N days ago.

On Balance Volume (OBV)is a technical indicator used to estimate buying and selling trends of a stock, by considering the cumulative volume, and utilizing changes in volume to estimate changes in stock prices. It cumulatively adds the volumes on days when the prices go up and subtracts the volume on the days when prices go down, compared to the prices of the previous day.

$$\text{if ClosingPrice} > \text{ClosingPrice}_Y, OBV = OBV_Y + Volume$$
$$\text{if ClosingPrice} < \text{ClosingPrice}_Y, OBV = OBV_Y - Volume$$
$$\text{if ClosingPrice} = \text{ClosingPrice}_Y, OBV = OBV_Y$$

Where OBV is On-Balance Volume, ClosingPriceY is the closing price yesterday, and OBVY is the On-Balance Volume yesterday.

all of these Momentum Indicators are then put inside a new data set and will later be used as input for the machine learning algorithms.

3.1.3 Output Data

Because this project involves both value prediction and trend prediction, there will be 2 types of output data, numerical and binary.

The output data for value prediction will be numerical, predicting the next day's stock closing price. Whereas the output for trend prediction will be binary, as it only needs to predict if the stock will go up or down the next day.
The input for the LSTM model will be slightly different compared to other models, which will be explained in the LSTM section.

3.2 Machine Learning Models

A general rundown of the algorithm settings and how each of them are used in this project.

3.2.1 LSTM

The LSTM model will consist of a sequential model from TensorFlow, with 3 layers of an output unit of 50, with the first layer having an input size the same as the input data.

The input data will be prepared in a way that 20 days of input data(X_train/test) will derive the output 1 day into the future(Y_train/test). (In comparison, the input and output are 1 to 1 when using other models.) This is deemed necessary for LSTM to function at its best, as LSTM requires a sequence of data for it to find patterns hidden within.

Both the value trend prediction will use the same LSTM model, however, the trend prediction's result will be modified so that its results can be evaluated, as LSTM does not make binary predictions, only percentages and continuous data.

3.2.2 Random Forest

The Random Forest model will come from the Scikit-learn library, with the number of estimators set to 150. The trend prediction will use Random Forest Classifier instead.

3.2.3 Linear Regression

The Linear Regression model will come from the Scikit-learn library, The trend prediction will use the SGDClassifier which is a linear regression classifier.

3.3 Evaluation methods

The results of the value prediction will first be plotted along with the goal price. Here the Y axis will be the price-quantity, while the X axis will be the time unit. Here the time unit is not in the format year-month-day in order to make the axis labels display better, it is switched out with integer numbers. the green line will represent the predicted price, while the black line will represent the actual price in that time period.

There will also be several evaluators measuring accuracy and precision:

- RMSE
- MAE
- r2

The results of trend prediction will be evaluated by calculating the accuracy score to the real data, which will tell us the percentage of predictions that were correct. The binary version of the real data(Y_test) will be prepared beforehand.

Chapter 4

Implementation

The software part of this project consists of several Python scripts that when executed, predict stock prices using the script's corresponding algorithm and/or input data set. The language Python is chosen due to its resourceful machine-learning libraries(Scikit and TensorFlow) and its suitability for manipulating and analyzing financial data(especially with pandas and numpy), which also has a built-in system for fetching financial data(pandas_datareader).

There are 2 algorithms used per script, one for value prediction and one for trend prediction. The price-related input and the momentum indicator input versions are also separated into separate python script files. Therefore there are a total of 6 independent files predicting each type of prediction and returning their individual evaluation. Each file can be executed separately and can be modified independently.

Those files are:

- LSTM_with_prices.py
- LSTM_with_indicators.py
- random_forest_with_prices.py
- random_forest_with_indicators.py
- linear_regression_with_prices.py
- linear_regression_with_indicators.py

This type of design is decided because each of the machine learning models requires its own slight adjustments to the input data(for example LSTM require an input size of 20:1 per day), adopting this type of design makes changing each model's input and output much easier and the process can be more closely monitored.

Another reason for this is that the algorithms are required to run several times over to obtain the data needed, this makes it so that data is much easier/cleaner to record for each model. There is also the need for consistent environment variables(execution time will be recorded), therefore making each model run independently will help prune out inconsistencies.

However, another design that would have worked would be to add a central script that fetches stock data of the assigned company name, goes through each of the models and gathers all the results and evaluate them all together. The current design is chosen due to its simplicity and flexibility when the input/output data of each model has to be adjusted.

4.1 Hardware/Software

Computer: ASUS ROG Zephyrus G15

CPU: AMD Ryzen 7 4800HS

RAM: 16GB

Software: Spyder IDE 5.1.5

Language: Python 3.9.7 64-bit

OS: Windows 10

Python packages used:

- numpy
- matplotlib
- pandas
- pandas_datareader
- yfinance
- sklearn
- TensorFlow

4.2 Input Data

Data is fetched from the pandas_datareader in combination with Yahoo Finances. There are other sources of financial data that display more detailed info(eg: adj_close, adj_open, etc...), however, Yahoo Finances would provide enough price attributes for the calculations made in this project.

```
38  #load data
39  company = "GOOG"
40
41  start = dt.datetime (2012,1,1)
42  end = dt.datetime.now()
43
44  train_start = dt.datetime (2012,1,1)
45  train_end = dt.datetime(2021,12,1)
46
47  data = pdr.get_data_yahoo(company, start, end)
48
49  data['change_in_price'] = data['Close'].diff()
```

The company stock symbol is chosen here(Google in this example). The start date and end date are specified, along with the start and end date of the training set. The total data set used including both the test and train set is fetched.

A new column change_in_price is also added to the data set fetched, which calculates the difference between the closing price of each day and the next day. (only used in momentum indicator data set)

Close	Adj Close	Volume	change_in_price
16.5731	16.5731	147611217	0
16.6446	16.6446	114989399	0.0714817
16.4137	16.4137	131808205	-0.230885
16.1898	16.1898	108119746	-0.223909
15.5034	15.5034	233776981	-0.686428
15.5203	15.5203	176483032	0.0169363
15.5906	15.5906	96359832	0.0702372
15.6822	15.6822	75289148	0.0916557
15.5664	15.5664	92637933	-0.115815
15.6558	15.6558	76658261	0.0894146
15.7637	15.7637	110882061	0.107846
15.9295	15.9295	253157352	0.165877
14.595	14.595	424637703	-1.3345
14.5833	14.5833	137027695	-0.0117064

4.2.1 Price Related

Here is what the dataset looks like after all the relevant values are prepared in a price input model.

Date	Open	High	Low	Close	Adj Close	Volume	value_prediction	Prediction
2012-01-03 00:00:00-05:00	16.2625	16.6414	16.2483	16.5731	16.5731	147611217	16.6446	1
2012-01-04 00:00:00-05:00	16.5637	16.6937	16.4538	16.6446	16.6446	114989399	16.4137	-1
2012-01-05 00:00:00-05:00	16.4914	16.5373	16.3445	16.4137	16.4137	131808205	16.1898	-1
2012-01-06 00:00:00-05:00	16.4172	16.4384	16.1841	16.1898	16.1898	108119746	15.5034	-1
2012-01-09 00:00:00-05:00	16.1021	16.1146	15.4728	15.5034	15.5034	233776981	15.5203	1
2012-01-10 00:00:00-05:00	15.685	15.7858	15.3652	15.5203	15.5203	176483032	15.5906	1
2012-01-11 00:00:00-05:00	15.5293	15.676	15.47	15.5906	15.5906	96359832	15.6822	1
2012-01-12 00:00:00-05:00	15.7216	15.7632	15.604	15.6822	15.6822	75289148	15.5664	-1
2012-01-13 00:00:00-05:00	15.598	15.6152	15.4685	15.5664	15.5664	92637933	15.6558	1
2012-01-17 00:00:00-05:00	15.7405	15.7405	15.5836	15.6558	15.6558	76658261	15.7637	1
2012-01-18 00:00:00-05:00	15.6073	15.7908	15.4949	15.7637	15.7637	110882061	15.9295	1
2012-01-19 00:00:00-05:00	15.9649	15.9649	15.7275	15.9295	15.9295	253157352	14.595	-1
2012-01-20 00:00:00-05:00	14.7081	14.7198	14.4882	14.595	14.595	424637703	14.5833	-1

The column value_prediction is simply the closing price column of the data set but shifted 1 row up. This is so that the prices that are being predicted are from the next day, not on the same day.

The column Prediction is a binary conversion of whether the change in price increased or decreased on the next day, if it increased, it is set to 1, if decreased, it is set to -1.

For the scripts using price-related input data, the 'Open', 'High', 'Low' and 'Volume' columns are put into X_train.

```
61  data_train = data.loc[:train_end,:]
62
63  x_train_data = data_train[['Open','High','Low','Volume']]
64  y_train_data = data_train ['value_prediction']
65
66  x_train_data = scaler_x.fit_transform(x_train_data)
67  y_train_data = scaler_y.fit_transform(y_train_data.values.reshape(-1,1))
68
69  x_train = []
70  y_train = []
71
72  x_train = np.array(x_train_data)
73  y_train = np.array(y_train_data)
```

Figure 4.1: MinMaxScaler

```
44  scaler_x = MinMaxScaler(feature_range = (0,1))
45  scaler_y = MinMaxScaler(feature_range = (0,1))
```

Figure 4.2:

The data sets are then put through a MinMaxScaler so that the data is normalised. The Y_train is also reshaped into a 2D array so that the respective models will accept it.

4.2.2 Momentum Indicator

Calculations of the momentum indicators will take place and replace the price-related input. The rest of the code will remain the same.

For explanations of the calculations made including the formulas used, please see 3.1.2 Momentum Indicator Input Data.

The RSI is calculated using the code down below, it first calculates the Exponential Weighted moving average, by making 2 copies of the change_in_price column and making one only keep the days where price increased, and one keeping the days where price decreased, then dividing the copies to calculate the relative strength, which is then converted into the Relative Strength Index using the equation provided.

```
#14 days
n=14
# copy change in price twice
up_df, down_df = data[['change_in_price']].copy(), data[['change_in_price']].copy()

# For up days, if the change is less than 0 set value to 0.
up_df.loc['change_in_price'] = up_df.loc[(up_df['change_in_price'] < 0), 'change_in_price'] = 0

# For down days, if the change is greater than 0 set value to 0.
down_df.loc['change_in_price'] = down_df.loc[(down_df['change_in_price'] > 0), 'change_in_price'] = 0

down_df['change_in_price'] = down_df['change_in_price'].abs()

# for both up and down days, calculate the Exponential Weighted Moving Average
ewma_up = up_df['change_in_price'].transform(lambda x: x.ewm(span = n).mean())
ewma_down = down_df['change_in_price'].transform(lambda x: x.ewm(span = n).mean())

relative_strength = ewma_up / ewma_down

relative_strength_index = 100.0 - (100.0 / (1.0 + relative_strength))

data['down_days'] = down_df['change_in_price']
data['up_days'] = up_df['change_in_price']
data['RSI'] = relative_strength_index
```

The Stochastic Oscillator and Williams %R are calculated quite similarly, they both require the highest and lowest closing price within a 14-day period(the lambda function does just that). Therefore only the equation needs to change when calculating the Williams %R.

```
low_14, high_14 = data[['Low']].copy(), data[['High']].copy()

# Apply the rolling function and grab the Min and Max.
low_14 = low_14['Low'].transform(lambda x: x.rolling(window = n).min())
high_14 = high_14['High'].transform(lambda x: x.rolling(window = n).max())

# Calculate the Stochastic Oscillator.
k_percent = 100 * ((data['Close'] - low_14) / (high_14 - low_14))

# Add the info to the data frame.
data['low_14'] = low_14
data['high_14'] = high_14
data['k_percent'] = k_percent

r_percent = ((high_14 - data['Close']) / (high_14 - low_14)) * - 100
data['r_percent'] = r_percent
```

The MACD and Price Rate Of Change calculations are quite simple and are done exactly like how they were specified in 3.1.2 using the equations. Where the first lambda function calculates the EMA of the previous 26 days and the second lambda function calculates the EMA of the previous 12 days.

```
# Calculate the MACD
ema_26 = data['Close'].transform(lambda x: x.ewm(span = 26).mean())
ema_12 = data['Close'].transform(lambda x: x.ewm(span = 12).mean())
macd = ema_12 - ema_26

# Calculate the EMA
ema_9_macd = macd.ewm(span = 9).mean()

# Store the data in the data frame.
data['MACD'] = macd
data['MACD_EMA'] = ema_9_macd
```

Figure 4.3:

```
# Calculate the Price Rate of Change
n = 9

# Calculate the Rate of Change in the Price, and store it in the Data Frame.
data['Price_Rate_Of_Change'] = data['Close'].transform(lambda x: x.pct_change(periods = n))
```

Figure 4.4:

```
def obv(group):

    # Grab the volume and close column.
    change = data['Close'].diff()
    volume = data['Volume']

    # intialize the previous OBV
    prev_obv = 0
    obv_values = []

    # calculate the On Balance Volume
    for i, j in zip(change, volume):

        if i > 0:
            current_obv = prev_obv + j
        elif i < 0:
            current_obv = prev_obv - j
        else:
            current_obv = prev_obv

        # OBV.append(current_OBV)
        prev_obv = current_obv
        obv_values.append(current_obv)

    # Return a panda series.
    return pd.Series(obv_values, index = data.index)


# apply the function to each group
obv_groups = data.apply(obv)['Open']

# add to the data frame, but drop the old index, before adding it.
data['On Balance Volume'] = obv_groups
```

Figure 4.5:

On Balance Volume is calculated with a function obv(), it decided what the OBV value is depending on if the closing price went up, went down or stays the same.

All the values calculated above are then put into the original data set, here is what the data set looks like after all the momentum indicators are in place:

Date	Open	High	Low	Close	Adj Close	Volume	change_in_price	down_days	up_days	RSI
2012-01-03 00:00:00-05:00	6.2625	16.6414	16.2483	16.5731	16.5731	147611217	0	0	0	0
2012-01-04 00:00:00-05:00	6.5637	16.6937	16.4538	16.6446	16.6446	114989399	0.0714817	0	0.0714817	25.3
2012-01-05 00:00:00-05:00	6.4914	16.5373	16.3445	16.4137	16.4137	131808205	-0.230885	0.230885	0	21.1555
2012-01-06 00:00:00-05:00	6.4172	16.4384	16.1841	16.1898	16.1898	108119746	-0.223909	0.223909	0	11.2394
2012-01-09 00:00:00-05:00	6.1021	16.1146	15.4728	15.5034	15.5034	233776981	-0.686428	0.686428	0	4.22851
2012-01-10 00:00:00-05:00	6.685	15.7858	15.3652	15.5203	15.5203	176483032	0.0169363	0	0.0169363	5.89957
2012-01-11 00:00:00-05:00	6.5293	15.676	15.47	15.5906	15.5906	96359832	0.0702372	0	0.0702372	13.1509
2012-01-12 00:00:00-05:00	6.7216	15.7632	15.604	15.6822	15.6822	75289148	0.0916557	0	0.0916557	22.1803
2012-01-13 00:00:00-05:00	6.598	15.6152	15.4685	15.5664	15.5664	92637933	-0.115815	0.115815	0	19.2607
2012-01-17 00:00:00-05:00	6.7405	15.7405	15.5836	15.6558	15.6558	76658261	0.0894146	0	0.0894146	27.7344
2012-01-18 00:00:00-05:00	6.6073	15.7908	15.4949	15.7637	15.7637	110882061	0.107846	0	0.107846	36.9444
2012-01-19 00:00:00-05:00	6.9649	15.9649	15.7275	15.9295	15.9295	253157352	0.165877	0	0.165877	48.5756
2012-01-20 00:00:00-05:00	6.7081	14.7198	14.4882	14.595	14.595	424637703	-1.3345	1.3345	0	17.9094
2012-01-23 00:00:00-05:00	6.5053	14.6615	14.5346	14.5833	14.5833	137027695	-0.0117064	0.0117064	0	17.7057

low_14	high_14	k_percent	r_percent	MACD	MACD_EMA	rice_Rate_Of_Chang	On Balance Volume	value_prediction	Prediction
0	0	0	0	0	0	0	0	16.6446	1
0	0	0	0	0.00160376	0.000890976	0	114989399	16.4137	-1
0	0	0	0	-0.00511621	-0.00157099	0	-16818806	16.1898	-1
0	0	0	0	-0.0161314	-0.00650336	0	-124938552	15.5034	-1
0	0	0	0	-0.0490918	-0.0191725	0	-358715533	15.5203	1
0	0	0	0	-0.0672585	-0.0322064	0	-182232501	15.5906	1
0	0	0	0	-0.0745686	-0.0429272	0	-85872669	15.6822	1
0	0	0	0	-0.0739122	-0.0503735	0	-10583521	15.5664	-1
0	0	0	0	-0.078079	-0.0567736	0	-103221454	15.6558	1
0	0	0	0	-0.0752932	-0.0609231	-0.0553493	-26563193	15.7637	1
0	0	0	0	-0.0664787	-0.0621386	-0.0529269	84318868	15.9295	1
0	0	0	0	-0.05	-0.0595317	-0.0294988	337476220	14.595	-1
0	0	0	0	-0.112499	-0.0707415	-0.0985047	-87161483	14.5833	-1
14.4882	16.6937	4.2130	95.6861	0.158408	0.0891002	0.0524453	224180178	14.469	1

The 'RSI', 'k_percent', 'r_percent', 'MACD', 'On Balance Volume', 'Price_Rate_Of_Change' are then put into the input dataset, rather than price values.

```
187   x_train_data = data_train[['RSI','k_percent','r_percent','MACD','On Balance Volume','Price_Rate_Of_Change']]
188   y_train_data = data_train['value_prediction']
```

4.3 Machine Learning Models

The machine learning models that will be used will come from:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDClassifier
```

Models are not written from scratch but imported because TensorFlow and Scikit-Learn already provide models good enough for the purpose of this project.

4.3.1 Value Prediction

For the LSTM model, it will come from tensorflow.keras.models. The Sequential model below will be used as explained in 3.2.1 LSTM.

```
#build the model
model = Sequential()

model.add(LSTM(units = 50, return_sequences = True, input_shape = (x_train.shape[1],6)))
model.add(Dropout(0.2))
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(units = 50))
model.add(Dropout(0.2))
model.add(Dense(units = 1))

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, epochs = 40, batch_size = 20)
```

Figure 4.6:

However the input and output of the models will be made so that 20 prior days of data will correspond to 1 day of output, and it will be done every day of the data set.

```
prediction_days = 20

x_train = []
y_train = []

for x in range (prediction_days, len(x_train_data)):
    x_train.append(x_train_data[x - prediction_days: x])
    y_train.append(scaled_data[x , 0])
```

Figure 4.7:

The Linear Regression and Random Forest models will come from sklearn.ensemble and sklearn.linear_model, as specified in 3.2.2 and 3.2.3.

4.3.2 Trend Prediction

The input data will stay the same as value prediction during trend prediction. The output data have to be changed to binary data in order for the evaluation to work.

The LSTM model will also stay the same for trend prediction. This is slightly problematic

as it can only produce continuous output rather than binary. Therefore it is solved by making it produce data from the range -1 to 1 by using a min-max scaler on the output data. from here, when the result is positive or 0, it will be classified as 1, but when the result is negative, it is -1.

```
377    y_pred = list(map(lambda x: -1 if x<0 else 1, y_pred))
```

Figure 4.8:

For Random Forest and Linear Regression, the models will be changed to RandomForest-Classifier and SGDClassifier.

4.4 Evaluation

The predicted data will be plotted on a line chart, using matplotlib.pyplot. It will then be used to calculate the MAE, MSE and R2 values using functions from sklearn.metrics. There will also be a timer that times the execution time of how long each algorithm ran for.

```
236    mse = mean_squared_error(actual_prices, predicted)
237    mae = mean_absolute_error(actual_prices, predicted)
238    r2 = r2_score(actual_prices, predicted)
239
240    rmse = math.sqrt(mse)
241    print(f'RMSE: {rmse:.3f}')
242    print(f'MAE: {mae:.3f}')
243    print(f'r2: {r2:.3f}')
```

Chapter 5

Results/Evaluation

5.1 Results

Each stock predictor(6 total) ran 3 times for each chosen stock, while the average numerical evaluation of those 3 runs is recorded, as well as the most fitting graph out of the 3 runs. 4 chosen stocks are evaluated for each predictor, those being:

- GOOG(Google)
- MSFT(Microsoft)
- AMZN(Amazon)
- NVDA(Nvidia)

The Results of the evaluation are shown below:

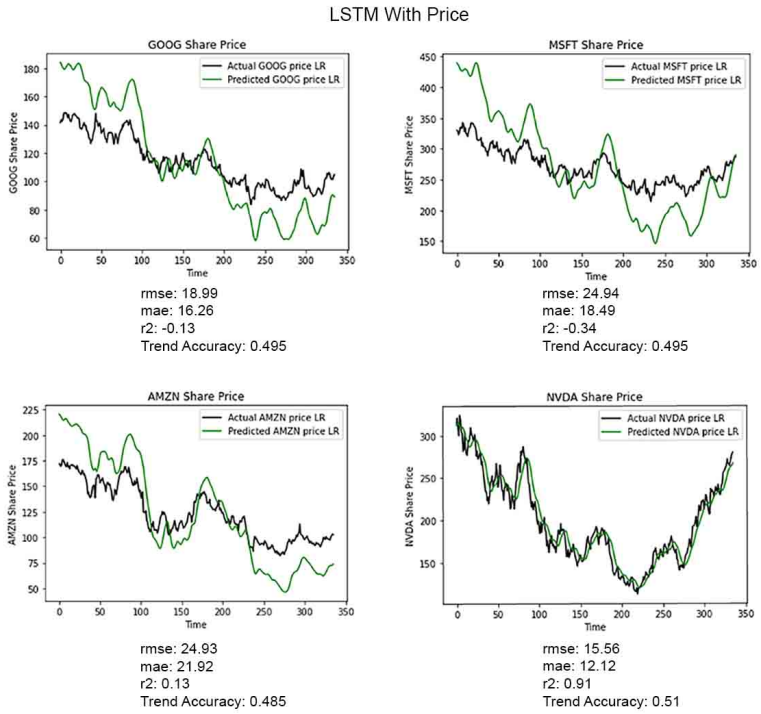

Figure 5.1:

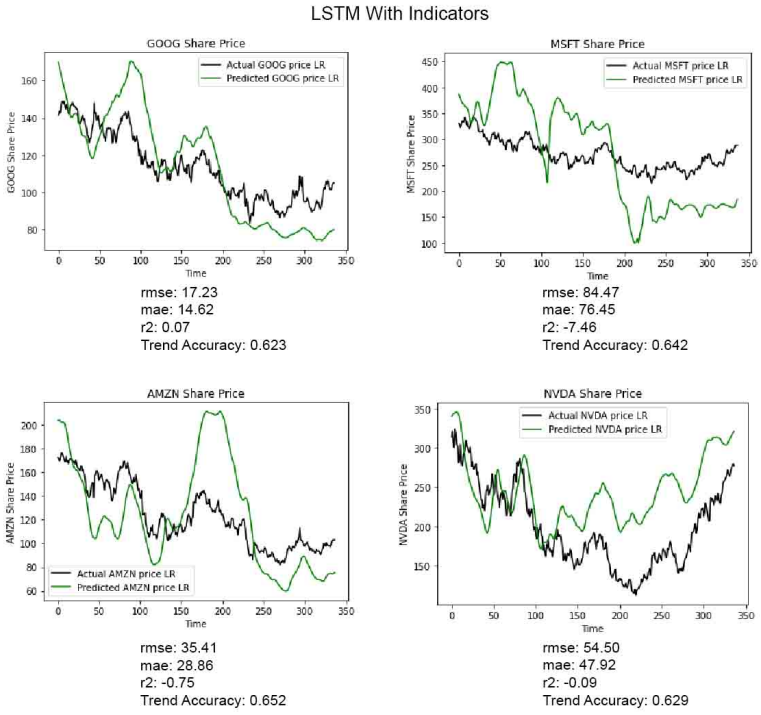

Figure 5.2:

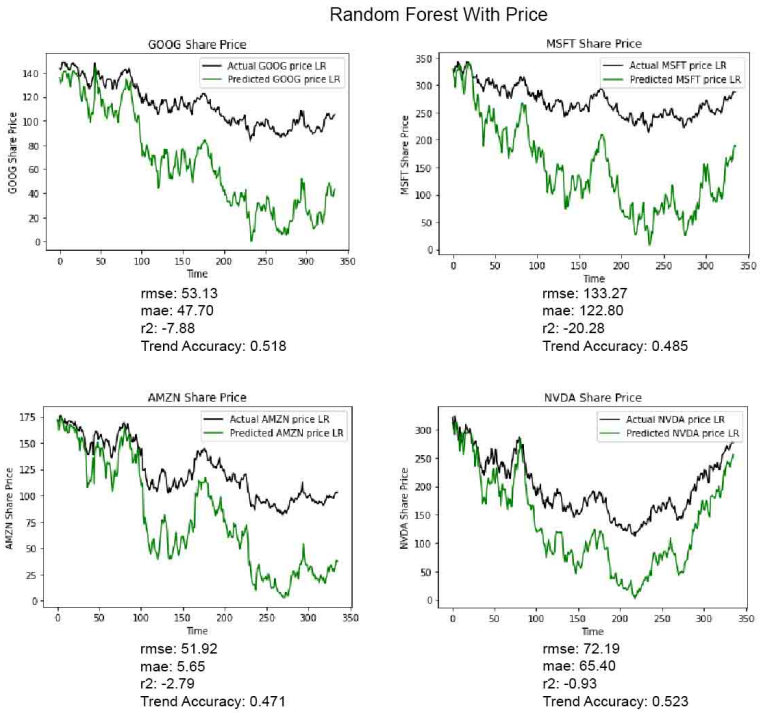

Figure 5.3:

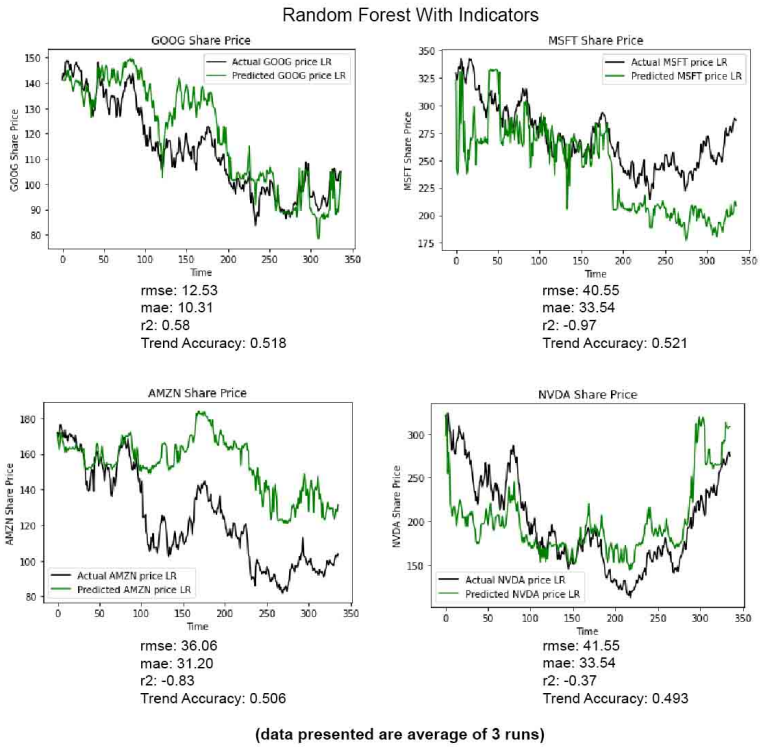

Figure 5.4:

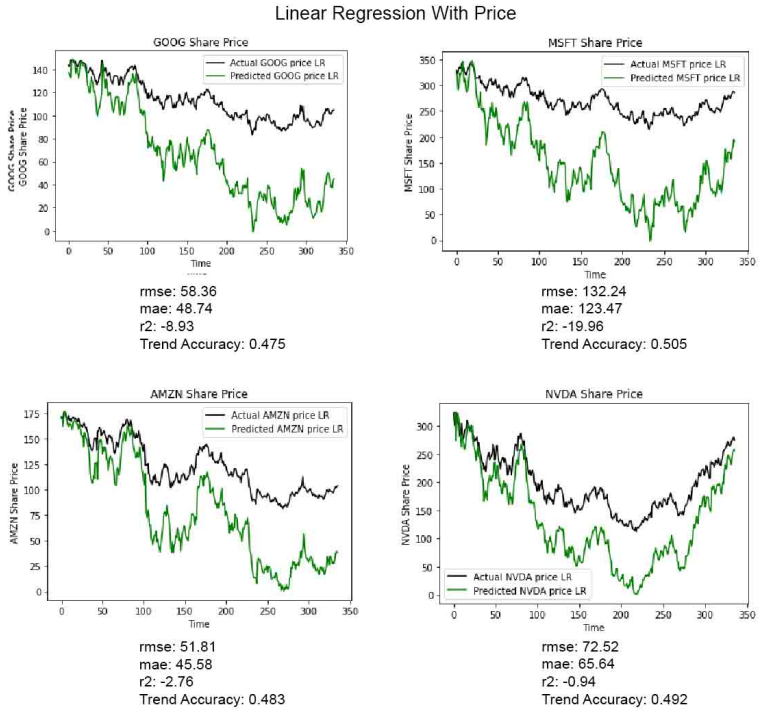

Figure 5.5:

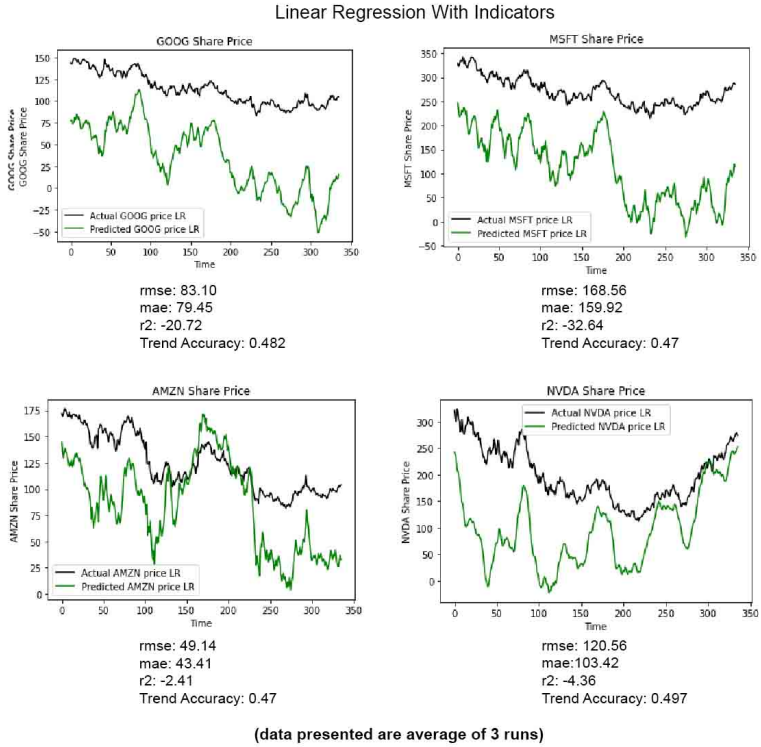

Figure 5.6:

5.2 analysis

5.2.1 LSTM with Prices

LSTM with price-related data as input show promising results for value prediction, Averaging a RMSE of around 21.10 and MAE of 17.2, which is the lowest RMSE and MAE scores out of all the algorithms. This is expected as training with price data that closely resembles the closing price will give a good numerical estimation(despite the closing price of the previous day is not part of the input data set).

The R^2 value shows how well the fit is between the predicted data and actual data, which is quite high compared to the other models.

Looking at the graphs, it would seem like the overall fluctuation of the values predicted has been correct(as in when the stocks go up or down). However, the scale of the data is inconsistent and a lot of the time are over scaled. This could mean that the model overestimates the increase and decrease in closing price. This can be fixed post-prediction, as the results get scaled back to regular data using the min-max scaler, and the amount scaled back can be changed so that the most fitting result is returned.

Data gathered for NVDA shows highly promising results, with an R^2 score of 0.91. As seen in the graph, the values predicted are an extremely close fit to the actual data. This shows what the model is capable of when the scaling is correct.

It can also be seen that the predicted data are slightly delayed on the graph compared to the actual data, by around 1-2 time units(especially obvious in NVDA stock). This could also be fixed post-prediction, by shifting the predicted graph a few units to the left.

Despite the accuracy of value prediction, the trend prediction is quite poor when using this model. The average accuracy of predictions does not exceed 50%, which means that it does not beat random predictions in this regard.
The algorithm ran for around 2 minutes and 43 seconds, which is very long compared to other algorithms in this project.

5.2.2 LSTM with Indicators

The value prediction isn't as accurate in this model as using prices to predict, averaging an RMSE of 47.90, and an MAE of 41.96, however, this is expected as there are only momentum data and no price data acting as input. It can however be seen that the points where there are increases and decreases in price are still recognised, and when a price is supposed to go up or down, it does change accordingly, it is only the magnitude at which it changes that is inconsistent. For example, in the stock NVDA, after time unit 100, the reflection points of the predicted curve match the actual prices, however, the placement and magnitude are off.

Despite this, the accuracy of the trend prediction has increased dramatically compared to predicting prices with an average accuracy of 63.7% across the 4 stocks. This is also expected as momentum data helps the prediction of trends dramatically, as more calculations are made surrounding the momentum, it makes it easier for the LSTM model to deduce the pattern within the data and the direction of where the next day's data will lie.

5.2.3 Random Forest with Prices

The Random Forest model when paired with price data doesn't seem very helpful when used for prediction. an average RMSE of 77.62, which is rather weak compared to the previous 2 models. the R^2 score is especially low with an average of -7.97. A negative R^2 score usually means that the predicted data barely fits the actual data. Therefore it is safe to say that it does not seem like the model has recognised many patterns from the price data.

The Accuracy of the trends is quite similar to the results LSTM with prices gave, with an average of 49.9% accuracy, just below 50%. It appears that both of those models do not seem to give any promising trend results.

5.2.4 Random Forest with Indicators

Surprisingly, the price value prediction using this model seems to be quite accurate, with an average RMSE of 32.67, and MAE of 27.14, ranking it second in all 6 algorithms, just below LSTM using prices.

Looking at the graph, the predicted curve seems to be more jagged than the lines in LSTM. This is probably because the input data in LSTM has been modified so that 20 days of data is used to predict 1 day into the future. This is not done for Random Forest as it is easily overfitted.

The GOOG stock make the best prediction out of the 4 stocks with an R^2 score of 0.58. This indicates that the model has potential and can be modified using techniques such as ensemble learning techniques, or making post-prediction changes to the data set.

Trend prediction using this model produces an average accuracy of 0.52, just above the random predictions, making this model quite well-rounded in both value and trend predictions.

Considering the algorithm only ran for around 0.8 seconds, the results are surprisingly positive.

5.2.5 Linear Regression with Prices

This model closely resembles the Random Forest with Prices model in both numerical evaluation and graph, with almost identical graphs produced. This is probably because the price data provided has values that are quite close to the actual prices, so weaker models tend to find a correlation between those values and the actual closing price, which leads to predictions that follow similar trends as the actual data but have values that are off.

This also means that both of these models are quite weak when finding a pattern within the data set.

5.2.6 Linear Regression with Indicators

This model does seem to find some correlation from the data set, judging from the pattern of the predicted curve which resembles the actual data at times. However, the values are quite far off, producing the highest RMSE and MAE in all the models.

This is expected, however, due to the fact that Linear Regression is quite a simple model when compared to the other 2.

This model also has the lowest accuracy out of all the models in trend prediction.

5.3 Overall Evaluation

The model that produced the best result for value prediction is undoubtedly LSTM using price data. Producing results with the lowest RMSE and MAE scores on its's own. It can easily be improved upon with post-prediction adjustments as mentioned above.

The time taken for the algorithm to run is dramatically longer than the other models. This is closely related to the batch size and the number of epochs, which is how many times the data is passed through the model. Right now the Batch size is set to 20 and Epoch is set to 40, however, if these were to be adjusted, the execution time might increase even further.

The plus side is, in real-world applications, as long as the model is not using real-time data, or predicting anything that requires fast decision times(eg. day trading), this amount of execution time is acceptable.

LSTM using momentum indicators produced the best trend prediction data, which on its own achieved an accuracy of 67.2%(in one of the MSFT runs).

The current model only consists of inputting all of the calculated momentum data into a data set and feeding it into the model, however, the data set can be improved further with each column having different weightings, where other types of analysis can be used to determine the weightings to hopefully improve the model further.

The current output of this model can also be adjusted, as LSTM cannot produce binary results, an estimate of binary results is produced from the continuous data outputted. This can be fixed using ensemble learning, where a second algorithm can be used to determine the final prediction using the percentages, as well as the rest of the input data. (for example, after getting an output of continuous data from the LSTM model, A random forest model uses that output and the previous momentum data to determine the final output.)

Random forest using momentum indicators is also a promising model, achieving both a decent value and trend prediction score with just 0.8 seconds of execution time. although there are better algorithms that can be used to make predictions, like LSTM, the random forest can be used as a supporting algorithm as described above, or can be used as a data pre-processing

tool, for example, separating the types of days in a stock and grouping them accordingly, so that more information can be extracted from the original data set.

However, one thing to note, due to the way random forest functions, it cannot predict exceeding the highest, or going below the lowest price point when predicting. This can be worked around by using a different algorithm when the stock value exceeds a certain price point and goes back to random forest when it returns below.

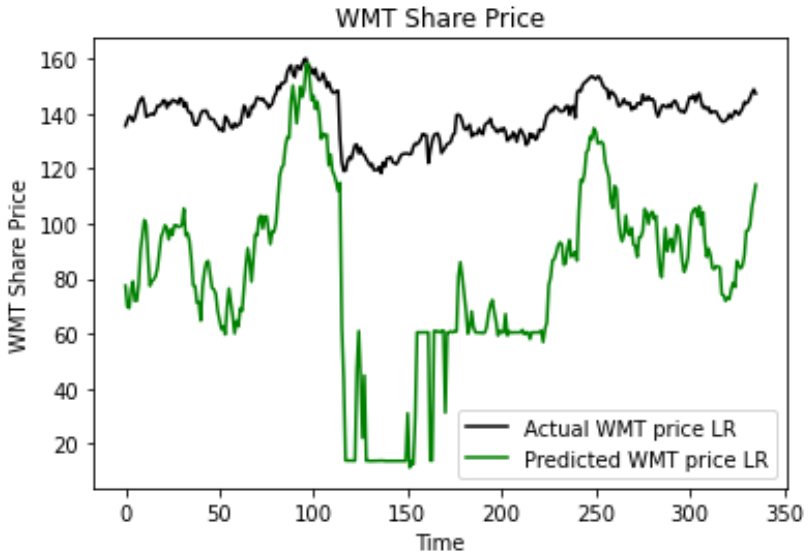

Figure 5.7: An example of the stock WMT, where random forest does not predict below around 15 price value.

(this is also why certain stocks are chosen in this project, as to obtain a fair assessment of the algorithms, it must be made sure that within the time period of the test set, the stock must not exceed the highest price point in the test set.)

Linear regression does not seem suitable for stock prediction, as stock data are too complex for the algorithm to find any patterns.

Chapter 6

Specification and Design

6.1 Functional Requirements

- The system will filter out data that are unusable and use filtered data when executing.
- The system must be able to execute given algorithms on historical stock data.
- the algorithms will provide prediction results for the different algorithms.
- The system will require different implementations for each algorithm, whether they be existing algorithms in packages or self-coded.
- The system will have to provide graphical visualisation of stock market data before and after the execution of algorithms.
- The specific algorithms needed are LSTM, random forest and linear regression.
- As the project develops, there may be more algorithms depending on the pace of development.
- The system will use the test set of the data to find out the MAE RMSE and R^2 of the algorithms.
- the system will test the execution time of each algorithm.
- further comparison and evaluation is to be analysed in the report.

6.2 Non-Functional Requirements

- The historical data being analysed is not real-time stock data, therefore there is no requirement for the system to have a fast execution time.
- The algorithms should be able to run independently, for the ease of changing certain model or running one model several times.
- The algorithms should be able to be applied to multiple stocks.
- Each algorithm should be adjusted to provide the most promising result possible.
- The data set should be split into a train and test set of appropriate proportions.
- LSTM will require running a number of epochs. The number of iterations should be dependent on when the data converge, and be decided appropriately.

6.3 General Specifications

The stock prediction tool will be written in Python, There are many reasons for this decision:

- Python is already a popular language in quantitative financial analysis, due to it being an object-oriented language and users are able to develop rapidly.
- This project plans to use the Python Pandas framework for fetching and filtering historical financial data. As well as Numpy for data analysis.

The project will use Anaconda Spyder as its IDE.

6.4 AI Specifications

Most of the algorithms will use the versions implemented in the Scikit-learn package in python. This includes

- Random Forest
- Linear regression

For LSTM, it will be using Tensorflow for the neural network model.

6.5 Evaluation Specifications

The data will initially be split into a 75% train and 25% test set.

For each algorithm, the result data will be compared to the test set and several scores will be calculated:

- MAE
- RMSE
- Scikit built-in score for testing accuracy
- Scikit built-in score for testing precision
- Execution time

The result of the prediction is plotted on a graph using the matplotlib package.

After all the algorithms ran, the result of the evaluation will be shown too on bar charts, as well as the evaluation times.

From there further evaluation is to be made in the report.

Chapter 7

Legal, Social, Ethical and Professional Issues

Your report should include a chapter with a reasoned discussion about legal, social ethical and professional issues within the context of your project problem. You should also demonstrate that you are aware of the regulations governing your project area and the Code of Conduct & Code of Good Practice issued by the British Computer Society, and that you have applied their principles, where appropriate, as you carried out your project.

7.1 Issues Surrounding the Subject

Many issues arises with the use of machine learning within the field of finance, and especially in stock prediction.

Data privacy has always been an arising issue surrounding data driven machine learning. For a stock prediction algorithm, the data involved must either be publicly available or legally obtained. GDPR laws have regulations against the collection, processing and storage of personal or illegal data. This is also to do with insider trading, market manipulation and fraud, as there are strict laws governing this area of finances.

The issues with social inequality when using machine learning is especially reflected in stock prediction, as there are chances that the technology would increase social inequality, as it does not benefit the users/creators equally. People of different financial status will also not be ben-

efited equally by the existence of this technology.

A machine learning algorithm can be nontransparent as sometimes it is difficult to understand its decision making. This raises concerns about who or what is accountable when issues arise. This is closely related to machine learning algorithms, as there are monetary risks to be taken if it were to be used in real life scenarios. instances of this issue may be a result to an underestimate of financial illiteracy, and a lack of understanding of investment risks and returns.

These issues are inevitable when working in this field, however, they can be minimised when abiding by the laws, Code of Conduct and Code of Good Practice issued by the British Computer Society, as well as careful,critical and ethical thinking when working with these subjects.

7.2 In Relation to Regulations

The issues described above are closely related to several principles issued by the British Computer Society. Despite the code of conduct not addressing machine learning or stock prediction specifically, there are still many general principles that are related to the field.

Members must develop their professional competence within the field of computing, which involves staying up to date with the latest development and best practices. This is Extremely important as machine learning and finances are rapidly developing and changing landscape where new technologies and regulations appear rapidly, it is key to stay up to date and act accordingly to everything that comes out.

Members must respect privacy when collecting/using data and must abide by the relevant laws and ethical principles. This is also closely related to a previous paragraph, as this project only uses publicly available stock data.

Developers must use their skills and knowledge to benefit society and minimise any negative impact of computing on society and the environment. This is closely related to a previous paragraph talking about social inequality, as this project will be for evaluation and research only.

Chapter 8

Conclusion and Future Work

Machine learning in the field of stock prediction has seen dramatic potential with LSTM, for both regression and classification. It produced promising results with little to no additions to the core model and can be explored further in ensemble learning.

The input data set is just as important as the algorithms themselves with different types of input deduced from the same data set achieving vastly different goals. Where price data is suitable for value prediction while momentum data derived from price data is better for stock trend prediction, capable of easily achieving more than an accuracy of 65%.

Other algorithms such as random forest also show promising results but are more suitable when used in combination with other algorithms, as well as linear regression which cannot seem to produce any satisfying result for this problem alone.

One future addition to this project would be to use ensemble learning to build a model achieving higher accuracy in value/trend prediction, using methods stated in 5.3 Overall Evaluation. From the data analysed, LSTM and random forest would be a key part of this, while other supporting models can be introduced too, as the size of the problem will increase the more input variables are introduced.

This project does not involve any Fundamentals Analysis techniques, however, that is one aspect of further research, as utilising further information such as financial news or reports is key to making more accurate predictions. One direction to look into could be text recognition

and evaluation algorithms which could help automate the process of news gathering, as well as ways of determining the true value of a stock with calculations.