

SmartStock

Stock Price Predictions using Machine Learning

BSc (Hons) Computer Science

Ronan Hayes

17830028

Supervisor: Dr Roger Evans

Second Reader: Dr Andrew Fish

Source Code:

<https://github.com/ro-stack/SmartStock>

Demonstration:

<https://web.microsoftstream.com/video/d1675b5b-b07e-44ae-b481-4d6f6a6a9bee>

Abstract

Technology has revolutionized the world of finance and banking, with machine learning being applied numerous ways, every day. This project aims to find a solution for reliably predicting the future of a stock's movement using supervised learning. I will be focusing on developing various regression algorithms for predicting future stock prices using a combination of technical indicators, as well as how classification algorithms can predict buy/sell signals. These models will then be integrated into a user-friendly application created using the Dash framework, the user will be able to test the models as well as plot various technical indicators.

Table of Contents

Abstract	2
1. Introduction	5
1.1. Goals.....	5
1.2. Inspiration.....	5
1.3. Adjustments.....	5
1.4. Related Modules.....	6
2. Background Research	6
2.1. Industry	6
2.2. Research Papers	7
2.3. Research Influence	7
3. Tools	7
3.1. Python	7
3.2. Python Libraries	8
3.3. Dash.....	8
3.4. Development Tools.....	9
4. Requirements Analysis	9
5. Project Management	9
5.1. Methodology	9
5.2. Timeline.....	10
6. Machine Learning Models.....	11
6.1. Machine Learning.....	11
6.2. Supervised Learning	11
6.3. Data.....	12
6.4. Feature Engineering	13
6.5. Model Structure	14
6.6. XGBoost Regression.....	15
6.7. Support Vector Regression (SVR)	15
6.8. Linear Regression	16
6.9. Decision Tree Regression.....	16
6.10. LightGBM Regression	16
6.11. Random Forest Regression	17
6.12. Results	17
6.13. Testing.....	18

6.14.	Buy/Sell Classifiers	19
7.	Dash Application.....	19
7.1.	Index.....	19
7.2.	Technical Analysis	20
7.3.	Machine Learning.....	22
7.4.	Design	23
7.5.	Testing.....	23
7.6.	Ethics.....	25
8.	Problems Encountered	25
9.	Improvements	27
10.	Successful Aspects	27
11.	Future Developments.....	28
12.	Conclusion	28
	References	29
	Appendix 1.....	30
	Appendix 2.....	31
	Appendix 3.....	31

1. Introduction

1.1. Goals

The main goal of the project is to develop a variety of regression models to predict the next day price of any stock. Using Open, High, Low, Close historical data for a given stock I will create technical indicators as features for the models. I can then train, test and evaluate their performance. Alongside the regression models I will investigate how classification can be used to predict whether a stock will go up or down in price the next day – returning a signal to either buy or sell. The final project goal is to develop a web application using Dash whereby users can interact with my models, as well as other functionalities such as plot various technical indicators or view the latest news. The resulting application resembling an investment dashboard.

In terms of personal goals, I will enhance my Python skills. My knowledge on machine learning will increase, and I can gain an insight into a career field I would possibly like to have a career in. Lastly, I will be able to learn a new modern web framework – Dash – which I could use for future data visualization projects.

1.2. Inspiration

Trading has always fascinated me, and the way financial markets work, and it is something I would like to learn truly one day. I learnt that 70% of trades everyday are made by algorithms as of 2017, and I am sure this has grown since then. [\[1\]](#) This made me even more intrigued by the topic. This led me to start exploring how technology plays a role in trading algorithms and learnt that machine learning can be a huge strength to succeeding in making profit. While large banks are using state of the art computers to develop high-frequency algorithms I thought I would base my project on a simpler way machine learning can be used within trading.

I also have a keen eye for detail and love for design. This led me to want to demonstrate these models in some form of data visualisation. Creating a web application allows me to generate something aesthetically pleasing while expressing my models visually.

1.3. Adjustments

Much of my project has stayed within the plan of my proposal however the one area that has changed was including sentiment as a feature for my models. After searching for various data sources, the only solutions were to use a very limited amount of sentiment or web scrape for data which I had not done before. Sentiment sources such as Twitter or News API limit you to the amount you can retrieve, which meant the sentiment would not match the amount of historical data I was training with. Apart from eliminating sentiment data, the original goals have all been met.

1.4. Related Modules

I believe all modules I have studied have had some impact on my project. The two most impactful modules being Intelligent Systems and Web Development.

CI213 Intelligent Systems allowed me to begin to learn Python which has been the core language within my project, this meant I was already aware of certain aspects of the language and did not have to learn it from scratch. It also developed my knowledge of various Artificial Intelligence topics, from search algorithms to machine learning. I also got to develop a classification algorithm. The insight into machine learning intrigued me and reinforced my desire to want to delve into the subject more.

CI135 Web Development is the second most influential as it taught me how to develop a website using HTML5, CSS3 and JavaScript. Being an artistic person myself I found the ability to develop something aesthetically pleasing with the ability to add various functionalities was fun. Developing my Dash application required a somewhat good understanding of HTML and CSS, this allowed me to take my Dash application further and make it more attractive and style it how I liked, rather than using the built in style.

While Python is somewhat a new language to me, I felt comfortable picking it up, thanks to modules such as CI101 Programming, CI228 Object Oriented Software Architecture, Design and Implementation, CI284 Data Structures and Algorithms and lastly CI285 Functional Programming. These have all helped me gain a true understanding of areas within programming, which could be transferred to Python.

Closely related to Web Development is CI141 Human Computer Interaction which strengthened my skills on how to make a website/application suitable for users. I was able to apply the user experience principles into my application to make the design user-friendly.

I based my individual report for CI153 Perspectives on Computing on ‘Future and Ethics of Artificial Intelligence in Banking’. This allowed me to investigate how AI is being used within Banking, specifically Natural Language Processing for various back-office jobs, Fraud Detection and Algorithmic Trading.

CI222 Project Planning, CI236 Integrated Group Project, CI143 Requirements Analysis have all made me aware of the steps that go into developing a product. They all played a part in different sections of my project from deciding the requirements of the product and how these needed to be planned out. They have allowed me to construct my product from start to finish in a structured format.

2. Background Research

2.1. Industry

Most of the research was conducted within my interim report however there were still elements I wanted to look at. I wanted to see what technologies companies were using to develop their machine learning models. Obviously, I would not be able to replicate systems

as developed as theirs but to gain an insight would be good. Man Group has been using machine learning to increase their returns and as of 2017 they had increased their assets under management by 77% over three years. [\[2\]](#) Another company I come across was Renaissance Technologies, a company based on developing statistical and quantitative models. They rely heavily on AI and have annual returns of 85.8% to 98.2% annually. [\[3\]](#)

2.2. Research Papers

A new paper I looked at was based on using Random Forest Regression to predict the closing price of a stock. I had not come across a paper on random forest regression yet so was intrigued by it. It seemed promising and achieved good results. I decided that I wanted to include a random forest regression model after this in the hopes of improving it as this paper stated they did not use any additional features. [\[4\]](#) I also investigated some more classification papers and a good example I found was on using XGBoost classifier. This was the first time I had seen someone use XGBoost for classification, so I wanted to look further into it. This example applied a combination of technical indicators and achieved some good results, reaching up to 74% accuracy. [\[5\]](#)

2.3. Research Influence

Much of the research I conducted led me to classification-based investigations for stock movement predictions, therefore I wanted to look more into the regression side as there were less papers based on this. My previous research had made me aware of the possibility's linear regression and support vector regression have for price predictions, but after reading the paper about I wanted to investigate random forest regression too. I also had read that XGBoost and LightGBM were extremely good and that XGBoost dominated winning Kaggle contests, this made me want to investigate these two as well. Other areas of research I looked at such as what language is most suitable and the best web framework to use helped me to decide which to progress with.

3. Tools

3.1. Python

The main chosen language for this project is Python. I have some prior experience in Python which was helpful. I decided to choose Python over other languages such as MATLAB or R due to the endless libraries it has to offer, including for machine learning. It can be easy to understand, which made it suitable as I would need to learn various new libraries. It also integrates with Dash which makes it the most appropriate language.

3.2. Python Libraries

I have used a vast amount of python libraries throughout this project, but I will cover the most influential.

Pandas is a vital tool for dealing with data within Python. It allows us to manipulate and analyse our data. Throughout my models I consistently have used some of its functions from reading in our data, removing specific columns from our dataframe, shifting a column back and so on. Pandas Datareader has been used to access the historical data via Yahoo. It allows us to create a dataframe from a specified Internet data source, in our case Yahoo.

Scikit-learn/sklearn has been extremely useful during this project and I have used many of their modules. Most of the models used are available via sklearn such as Linear Regression. I have also used their GridSearchCV package for ensuring my parameters were optimized. For evaluating my models, I used sklearn.metrics and was able to calculate RMSE, MAE and R^2 for each model.

Other model libraries include XGBoost and LightGBM which have allowed me to build two powerful models.

TA-Lib is a widely used library which allows you to perform technical analysis of historical data. It offers over 150 different technical indicators as well as candlestick pattern recognition. Using our data, we can create new features using TA-Lib's easy to function API, an example would be calculating the Simple Moving Average of the closing price.

To scale and normalize my data in the hope of achieving better results I used MinMaxScaler so each value within my features would be within the range -1 and 1. For each value in a feature, MinMaxScaler allows us to subtract the minimum value in the feature and divides it by the range between the original minimum and maximum.

I began using Matplotlib for visualizing different graphs and charts while developing my models, however I realized it would make much more sense to begin using Plotly for visualizing. This would save time when it comes to developing my application as the graphs used there would have been made using Plotly. Plotly graphs are much more interactive and have many functionalities. Rather than Matplotlib's static graphs, Plotly's you can zoom in/out of the graph, pan around the graph, and see data upon hovering.

3.3. Dash

Dash is a Python framework which can be used to generate web applications, specifically for data visualisation where you can include highly customisable user interfaces. It is written on top of Flask, Plotly.js and React.js and requires the user to develop the application using Python, HTML and CSS – eliminating the need to use JavaScript. [\[6\]](#)

After looking at various example applications I decided this was the best framework to develop my application on as it allowed me to implement a visually attractive application, with live updates based on user interaction. [\[7\]](#) By using 'callbacks' we can update graphs based on user input, this allows us to make our models interactive.

3.4. Development Tools

For developing my model's, I used Jupyter Notebook. It is very easy to use and allows you create and execute indivual snippets of code, this was a great benefit as it allowed for quick, effective changes to be made to my models. Once I had achieved my final models, Jupyter Notebook allows you to download the notebook as a Python file which I could then test in PyCharm. I used PyCharm for developing my application as I was familiar with other JetBrains IDE's. PyCharm allowed me to easily run my application locally.

4. Requirements Analysis

Many of the requirements have been reused from the original requirements, however I have been making changes constantly which required me to update the requirements. These come from ideas I figured I needed to implement, errors I found and feedback on the application.

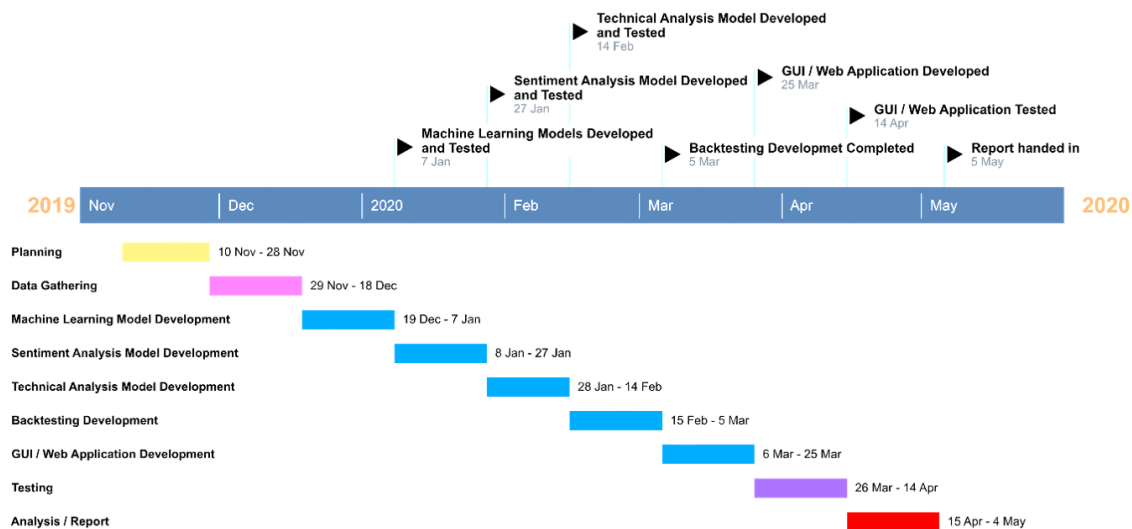
Requirements		
Requirement	User Story / Use	Priority
Retrieve historical data	Historical data is required to train the models	Must
Add technical features	Apply technical indicators as feature columns	Should
Clean the data	Remove any null values from data	Should
Normalize data	Scale the data suitably to enhance the models	Should
Split model into training, validation, testing sets	The data needs to be split correctly in a good proportion so that overfitting is eliminated.	Must
Perform grid search on parameter	To optimize the models use grid search to find the best out of a series of values	Should
Train / Test models	Train the models and test them so we can analyse the results.	Must
Evaluation metrics	Calculate evaluation metrics from the models	Should
Predict future results	Predict tomorrows closing price using todays data	Should
Create other models	Once one works well, use model as boilerplate code for new one	Could
Create a home page	Create an initial index page for application	Should
Plot chart for given stock	Allow user to search for a stock and plot graph of some sort	Must
Plot technical indicators	Be able to display multiple indicators on different graphs	Should
Update graphs via input	Be able to change graphs via symbol name or date	Should
Make notes on findings	Have a notes section where a user can input text based on their views/findings from an indicator	Would
Add descriptions of indicators	Add a small brief description about what the indicator does/measures for the user to understand it more.	Could
Give examples of symbols	Different types of symbols have different formats so should give a few examples	Should
Train models live within application	Upon user input train and test a model within the application	Should
User input for models	Allow user to select different models or input different stock symbols	Would
Print tomorrows prediction	Display tomorrows prediction as the graph only includes test data	Could
Inform user of training time wait	Some models can take a while to train, especially if I use GridSearch so inform them it might take a while rather than the page being static.	Should
Style each element	Edit CSS to style elements to my liking	Could
Separate all input/links from graphs	Clearly separate all the user input/interaction from the graphs	Could
Print evaluation metrics	Use metrics from model and display below predicted graph after training complete	Would

5. Project Management

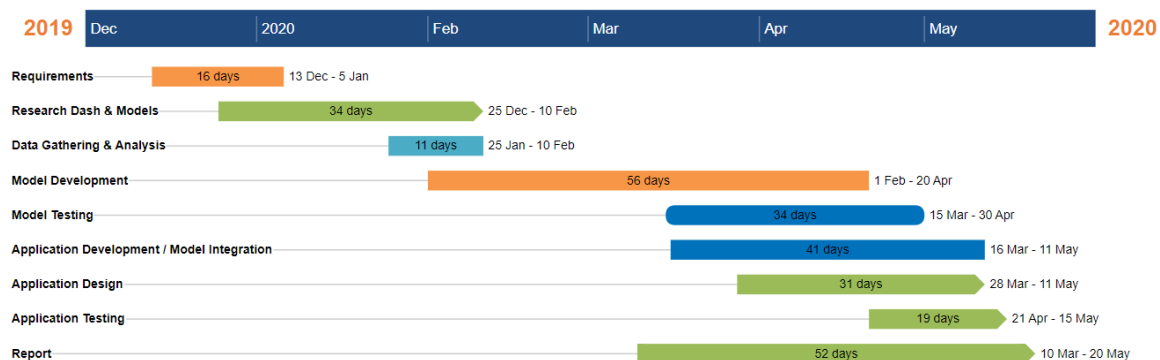
5.1. Methodology

I continued to use the agile approach as my project management methodology due to its flexibility and room for change. I found that my requirements were constantly changing and agile allowed me to update my sprints upon changes being made. I believe if I followed the original plan structure, I would have failed to implement new changes into the application and models as it followed a waterfall structure and did not consider room for problems and errors.

Below is my original plan for the project although it turned out to follow the waterfall methodology more than agile. Therefore, I needed to re-create my timeline based on any new changes and for it to follow the agile approach.



The updated plan below follows the agile methodology better and allows me to room to adjust requirements, as well as have room for any issues. I found I followed this timeline well, and it represented a good structure for developing my models and application.



5.2. Timeline

I also kept track of key dates throughout the project's progression, this included milestones as well as any important changes. This allowed me to ensure I was on track to complete my individual tasks and to implement the changes into my requirements.

Progression	
Date	Update
15/12/2019	Alpha Vantage Tested
17/12/2019	Dash tutorial app working, tried using AlphaVantage API didn't work due to API calls
01/01/2020	Changed data source to Yahoo via pandas web data reader
14/01/2020	SVR model implementation begins
18/01/2020	Draft SVR completed
22/01/2020	Added more technical indicators, performance improved
29/01/2020	Used GridSearchCV to find best parameters, again improved model
05/02/2020	Started using Plotly graphs to visualize predictions
10/02/2020	Implemented way to predict tomorrows price – copied dataframe last row
14/02/2020	Added evaluation metrics to understand model better
18/02/2020	Introduced normalization to data, improved model greatly
22/02/2020	Another change was to make the index the range and it improved slightly
01/03/2020	Working on making app multi paged
07/03/2020	Managed to eventually get multi pages to work
10/03/2020	Basic layouts for pages created, working on index page
15/03/2020	SVR model was completed, saved as Python file, removed grid search and tested
17/03/2020	Began working/researching on XGBoost model
19/03/2020	Majority was boilerplate code with a few changes, testing different parameters
24/03/2020	XGBoost model completed
28/03/2020	Linear Regression and LightGBM models added and completed
02/04/2020	Random Forest and Decision Tree models completed
03/04/2020	Working on Page1 – completed user input callback to update scatter of close price
08/04/2020	Changed to candlestick graph, added date range and began working on styling
16/04/2020	Added dropdown for graphs, plots one currently correctly.
22/04/2020	Changed to multi drop down, can plot multiple graphs/overlays
29/04/2020	Styling of CSS for Page1 looks good.
01/05/2020	Added final elements to Page1 such as graph descriptions, graph labels, example symbols.
02/05/2020	Working on Page2, created desired dropdown and search bar
12/05/2020	Implemented one model into app, few problems to fix
14/05/2020	Made dropdown work and added another model, tested and works when switching between.
18/05/2020	All models implemented via dropdown, input and dropdown respond correctly
23/05/2020	Styling for Page2 completed
24/05/2020	Added simple news page, worked apart from styling
26/05/2020	No fix to CSS news page problem so removed page
28/05/2020	Final test of app, all elements working correctly

6. Machine Learning Models

6.1. Machine Learning

Machine Learning is a subset of Artificial Intelligence. It is the process of teaching a computer system how to make accurate predictions when fed data. It enables computers to tackle tasks carried out previously by humans.

6.2. Supervised Learning

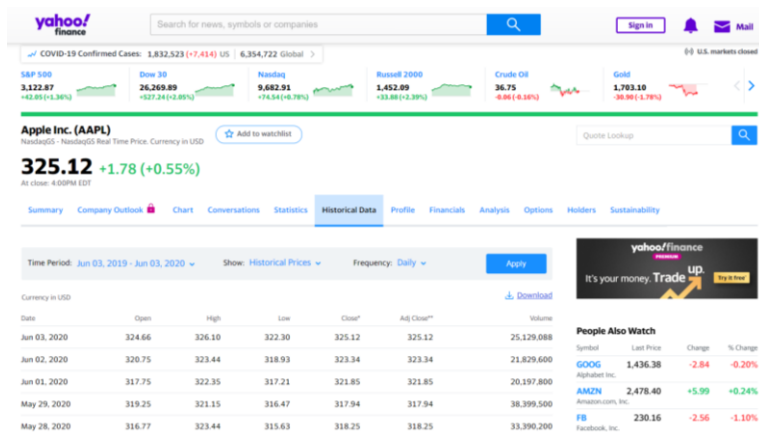
Supervised learning use algorithms to learn the mapping function between input variables (x) and output variables (Y). They use training data which acts as a teacher supervising the learning process, the algorithm can then iteratively make predictions which are corrected by the teacher until the results are acceptable and high performing. Supervised learning is broken down into two groups, regression, and classification. I will be focusing on regression problems but also look at classification.

Regression problems output a variable with a real numerical value or continuous, in our case it will be the closing price in dollars.

Classification on the other hand is where the output variable is categorical such as spam or not spam for an email spam detection classifier, in our case it will be does the closing price increase(1) or decrease(-1).

6.3. Data

The data I am using for my models is daily historical price data of a selected stock. The data I set out to retrieve was Open, High, Low, Close prices. I began using AlphaVantage who are a reliable data source outlet for financial data however the calls for their API were very limited. I then come across Yahoo Finance where you could download a .csv file of Open, High, Low, Close, Adjusted Close and Volume, this was good however it would require downloading a new .csv every time I wanted the most up-to-date data. However, using pandas datareader we can retrieve this same data with a simple function. If we set the end date to 'today()' using the datetime library, we will always have to most recent data when we run our models. I found the earliest their data dated back to was 1970 therefore if we set the start date to 1970, 1, 1 we will always have the earliest data available. There are also no limits meaning we can continuously use this. Below is an image of the data we are retrieving; you can also see the download button which can be used to download the .csv.

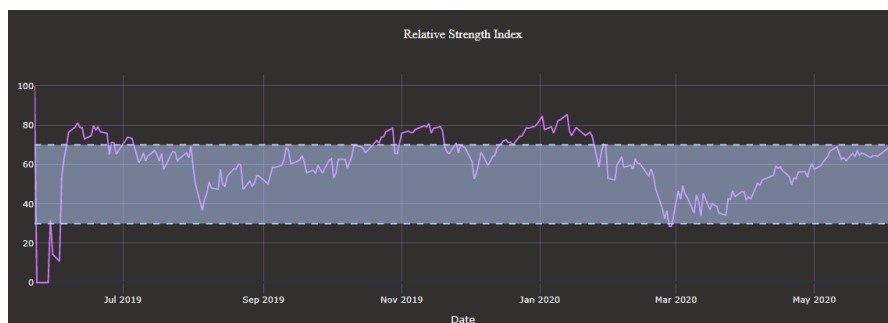


This chart below is a candlestick chart made using Plotly, it is a common way which you can visualize the OHLC data daily. The chart next to it explains how you interpret a candlestick. When a candlestick is red, it indicates the closing price was lower than the previous days, and vice versa if its green it means it was higher.



6.4. Feature Engineering

Technical analysis is an integral part of how traders make decisions on when to buy or sell a stock based on historical data. Therefore, I have decided to see if using only technical indicators as features will produce good results for predicting stock prices. There are patterns within technical indicators traders commonly use to determine their buying or selling actions. An example would be Relative Strength Index (RSI), this is an oscillator which ranges between 0 and 100, when the reading is above 70 it indicates the stock is overbought, versus if it is below 30 then it is oversold. By applying multiple different indicators hopefully our algorithm can pick up these patterns. Below is a chart representing RSI.



TA-Lib standing for Technical Analysis Library offers an easy way to calculate these indicators. It offers us the possibility of over 150 indicators under different categories. I will be focusing on two categories it offers: Overlap Studies and Momentum Indicators. Overlap studies refer to the movement of price. Many of these are different versions of a stock's moving average. Such as Bollinger Bands, which consists of 3 lines/bands. The middle being the Simple Moving Average (SMA) – the addition of recent prices and dividing by time-period. The upper and lower bands then represent the two standard deviations \pm of the SMA. The other being Momentum, this is the measurement of the speed or velocity of price changes. RSI is an example of a momentum indicator and like RSI many of these generate whether signals as to whether a stock is overbought or oversold. Another is Williams Percent Range / %R which shows where the last closing price is relative to the highest and lowest prices within a given time period, when the reading is above -20 it indicates the stock's overbought and below -80 is oversold. A different kind of momentum indicator is Average

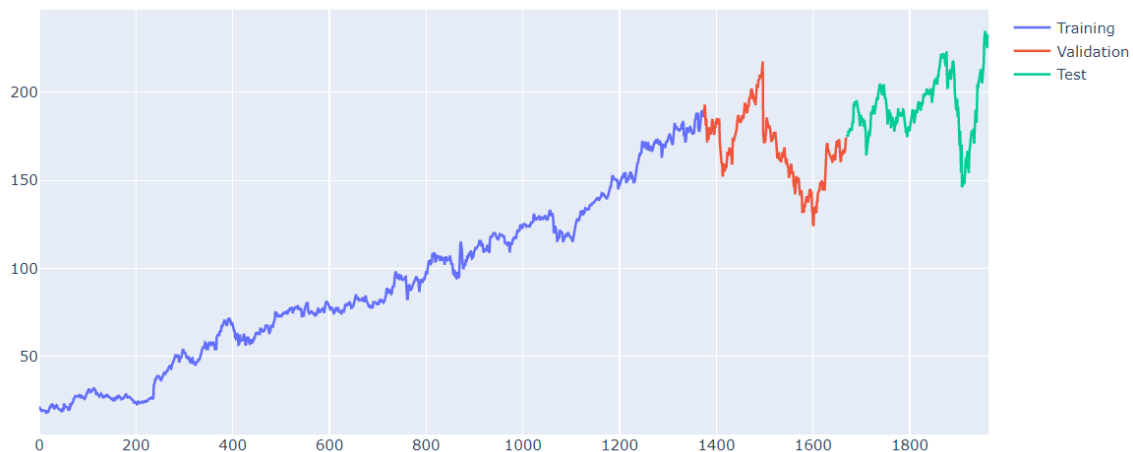
Directional Movement Index (ADX), which measures the overall strength of a trend. When the ADX value is above 25 it shows a strong trend, and there is no trend when less than 20.

In total I use 24 overlap and momentum indicators for my models. Below is an example of how I add a new feature to the dataframe. We simply need to call the indicator we want and set the correct parameters which we can find using their documentation. [\[8\]](#)

```
df['ADX'] = talib.ADX(high, low, close, timeperiod=14)
```

6.5. Model Structure

I will run through the general structure that my models follow. First, I import all the necessary libraries that I will be using. I then set the start and end date variables, along with the company name. I then use the data reader to retrieve the historical data for the company and transform it into a dataframe. I then apply all the technical indicators and create each as a new feature column. I then drop any columns that contain NaN values, many of the technical indicator first rows will be null as there is no data to fill them, i.e. the moving average of 14 days will be null for the first 14 rows. I then shift all the Close values back 1. This is, so that when we are predicting we predict the next day's closing price. I drop the last row as now we have shifted the Close values back 1, the last row will contain a NaN for the Close column. I found that rather than using the Date as the index, the range worked better, therefore I modify the index to be the length of the dataframe. I split the data into Training, Validation and Test sets in the format 70/15/15.

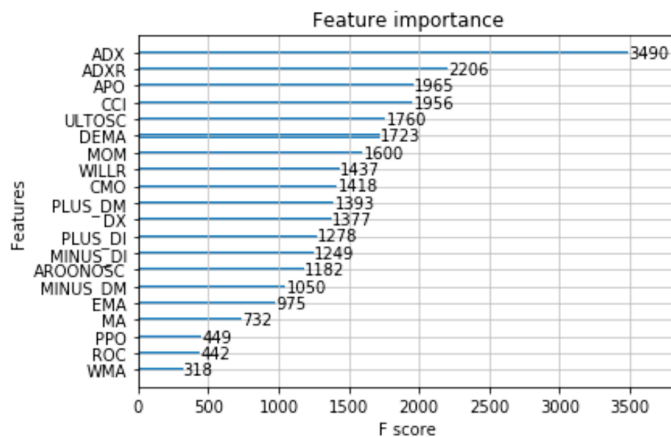


To enhance the model, I decided to normalize the data using MinMaxScaler. I fit all the values between -1 and 1 for every feature. I then drop the Volume, Open, Low, High and Adj Close. I now can create the y_train, X_train, y_valid, X_valid, y_test and X_test sets. The X sets containing only the features and the y sets containing the Close values. I can now begin training my models and I create a list of different values for some parameters. I create an evaluation set of the training data and validation data and perform a grid search on the multiple parameter values. Using best_params I can find what the best parameters were and now use them as the parameters for the actual model. I define the models name and fit it with the training and validation sets. Then I predict X_test and evaluate the results against

the `y_test` data. Multiple graphs are used throughout the see the split of the data and the predictions vs actual data.

I then created another model which I called ‘Final’ for each that only has training and test sets. I made these as otherwise it could take up to 40 minutes for a model to train within the application. I used the same best parameters as within the grid search for each model. The other change is I make a copy of the last row of the dataframe before removing it from the prediction set. This is so we can use it later to predict tomorrows closing price as due to shifting it we can only predict up to today’s closing price. So once the models trained, we can fit the model with the copied dataframe and return the value of tomorrows predicted price.

Below is an example within the XGBoost model that displays the importance of each feature. Every time I added new features the model improved, however less features may have improved it even more but finding which combinations worked would have taken an untold amount of time.



6.6. XGBoost Regression

XGBoost or eXtreme Gradient Boosting is an implementation of gradient boosted decision trees designed for speed and performance. It is seen as being very good at predicting structured tabular datasets for both regression and classification. It was developed for speed and performance.

6.7. Support Vector Regression (SVR)

It uses the Support Vector Machine algorithm to predict a continuous variable. SVR tries to fit the best line within a predefined or threshold error value. It tries to classify all the prediction lines in two types, ones that pass through the error boundary and ones that do not. Those lines which do not pass the error boundary are not considered as the difference between the predicted value and the actual value has exceeded the error threshold, epsilon. The lines that pass, are considered for a potential support vector to predict the value of an unknown.

6.8. Linear Regression

One of the most well known and understood machine learning algorithm is linear regression. It quantifies the relationship between one or more predictor variables and one outcome.

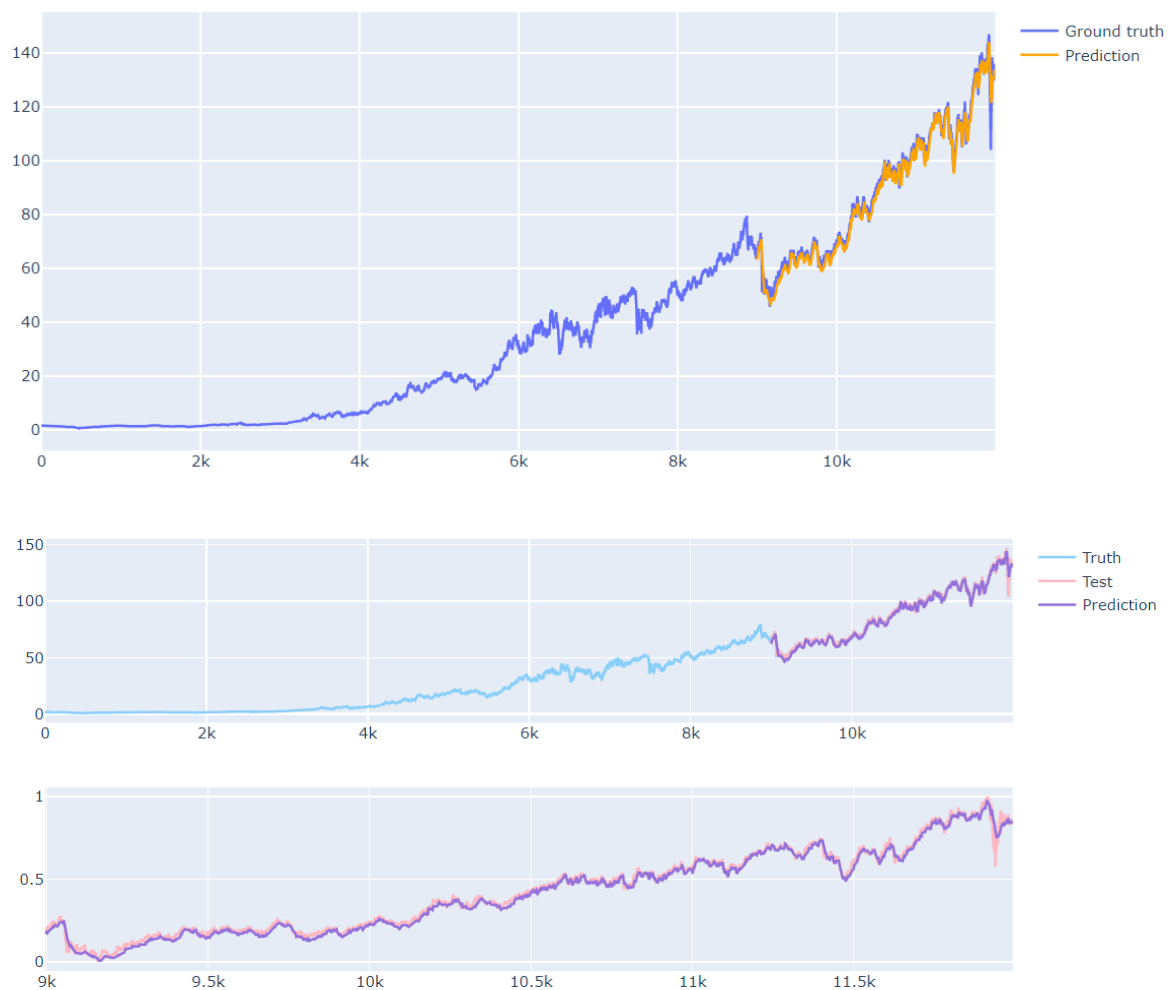
6.9. Decision Tree Regression

Decision Tree models are built in the form of a tree structure. It breaks the dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes. The top decision node within a tree corresponds to the best predictor. [\[9\]](#)

6.10. LightGBM Regression

LightGBM is a gradient boosting framework that uses tree-based learning. It grows trees vertically compared to other algorithms growing horizontally. It chooses the leaf with max delta loss to grow.

Below are the outputs of a LightGBM model on Pepsi Co.



6.11. Random Forest Regression

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and used averaging to improve the predictions accuracy.

6.12. Results

For analysing the results of my models and testing them I have chosen 3 different stocks at random, these are: PEP (PepsiCo, Inc), BABA (Alibaba Group Holding Limited), FB (Facebook, Inc).

I have used 3 different evaluation metrics, the first being Root Mean Squared Error (RMSE). RMSE is the standard deviation of the residuals – residuals being a measure of how far from the regression line data points are. Ultimately it measures how concentrated the datapoints are around the line of best fit. The closer to 0 the RMSE, the more concentrated the predictions are around the true values, making a lower score better.

Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions without considering their directions. Again, the lower the score the better.

R^2 is the statistical measure of how close the data is to the fitted regression line. Also known as the coefficient of determination. The closer the score is to 1 the better, a score of 1 represents the points are exactly on the trend line.

From my results table below I can infer that PEP's results are much greater than BABA and FB due to the larger amount of historical data used.

	PEP			BABA			FB		
Model:	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2
SVR	5.01	4.19	0.96	9.58	8.01	0.74	8.88	6.27	0.85
XGBoost	2.30	1.82	0.99	7.12	5.74	0.85	10.65	7.72	0.78
LightGBM	2.29	1.79	0.99	6.99	5.54	0.86	10.74	7.89	0.78
Random Forest	2.34	1.79	0.99	7.73	6.01	0.83	12.39	9.08	0.71
Linear Regression	3.27	2.95	0.98	8.16	6.23	0.81	12.39	9.44	0.71
Decision Tree	2.59	1.99	0.99	7.61	5.89	0.83	12.58	9.34	0.70

Both LightGBM and XGBoost proved to be the greatest models throughout the results. With SVR being the worst for two and best for one. I believe that BABA performed better than FB due to its Closing price fluctuating much more. The second highest peak of the price is within the training data, whereas FB is gradually increasing before it drops within the test data. Below on the left is BABA and on the right is FB. It is evident that the sudden drop in FB at the start of the test data could be the reason it performs the worst, even when it has more historical data than BABA.



6.13. Testing

To test my models properly I ran each model on 28th May 2020 once the markets had closed, meaning the prices would all be stable and not fluctuating when running the algorithms. This would then return the predicted closing price for May 29th. I could then create a table of the actual closing price versus what was predicted. As you can see from the table below the predictions for PEP were very accurate with 4/6 correctly predicting \$131. As we move to BABA, we can see again they are accurate with all the predictions only being a maximum of \$4 off the true price. Lastly, FB predictions were relatively accurate with the maximum difference being \$6 off the true price.

These tests correlate extremely well with our table of results from above. For example, PEP's test we see that every prediction is within \$1 off the true price except for SVR. The RMSE for SVR's prediction of PEP is also highly different from the rest of the models, who all score an RMSE beginning with 2. However, for BABA the SVR model had the closest prediction yet achieved the highest RMSE and MAE and lowest R^2 , like SVR's prediction of PEP the metrics suggest that SVR's prediction of BABA should have been the least accurate.

Again, we can see from our test results that PEP had the most accurate predictions, highly due to the greater scale of data used.

	PEP		BABA		FB	
Model:	True	Predicted	True	Predicted	True	Predicted
SVR	131.55	125.39	207.39	208.73	225.09	220.49
XGBoost	131.55	131.57	207.39	203.90	225.09	230.93
LightGBM	131.55	131.27	207.39	203.41	225.09	231.29
Random Forest	131.55	131.58	207.39	205.40	225.09	228.55
Linear Regression	131.55	131.06	207.39	209.32	225.09	227.40
Decision Tree	131.55	132.48	207.39	201.52	225.09	225.02

6.14. Buy/Sell Classifiers

Once I had developed my regression models, I thought it would be interesting to see how classification would perform for stock predictions. Therefore, I decided to create some simple models which would predict whether the next day's price would either increase or decrease. Upon further development these models could simulate buying/selling a stock based on the classifier's predictions.

Following the general structure of the regression models we read the data in, apply the technical indicator features but add another column called percentage change which calculates the change in percentage of the closing price from day to day. This can then generate a column called direction, using the `numpy.sign` function we can label each column with -1 or 1 representing if the percentage change went up or down. I then began testing and adjusting multiple models including: KNN, Logistic Regression, SVC, Random Forest, Naive Bayes, Decision Tree, XGBoost, Stochastic Gradient Descent and LightGBM. I did not spend much time optimizing the parameters but managed to achieve some promising results.

Accuracy of XBG model: 0.731	Accuracy of LightGBM model: 0.720
Confusion matrix:	Confusion matrix:
<code>[[548 8 211]</code>	<code>[[542 9 224]</code>
<code>[0 0 0]</code>	<code>[0 0 0]</code>
<code>[234 8 705]]</code>	<code>[240 7 692]]</code>
Accuracy of RF model: 0.718	Accuracy of Logistic Regression model: 0.702
Confusion matrix:	Confusion matrix:
<code>[[524 7 210]</code>	<code>[[390 3 102]</code>
<code>[0 0 0]</code>	<code>[0 0 0]</code>
<code>[258 9 706]]</code>	<code>[392 13 814]]</code>

All the above models achieved over 70% accuracy. For the Random Forest and Logistic Regression models I optimized the parameters which improved them slightly, however for XGBoost and LightGBM I did not, which suggests they could potentially be improved after optimizing.

7. Dash Application

7.1. Index

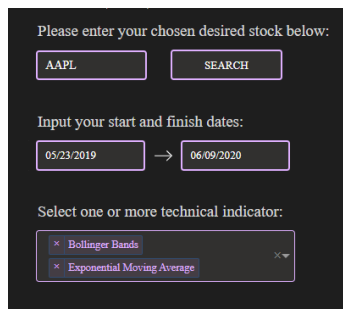
I wanted to make my application multi paged to separate technical analysis and machine learning, this would make each page less clustered. This proved more challenging than it sounded and after following a tutorial it did not work. I managed to make it work but at the expense of it being easier to navigate when coding, as I had to combine each page into one python file.

To create the multiple pages I had to create a layout for each page. Each of these layouts had a unique pathname. I had to use a callback to output the correct page based on the input of the URL and pathname. For example, when a link is clicked for the pathname `'/page-1'` it will return the `'page_1_layout'`.

I kept the index page simple and simple included the links to both the pages large and clearly underneath the title of the application. I also added a message saying do not use the application as investment advice to ensure users understand that any results are purely predictions and not to be used as investment ideas.

7.2. Technical Analysis

This page is where I had to begin conducting research into Dash more and viewing example application code became extremely beneficial. I found a great tutorial which got me started and it even was using the data reader to retrieve stock data. [\[10\]](#) This let me understand how callbacks worked and I began altering it so I could apply a date range too, and that the chart would be a candlestick chart. I now had the very basic backbone of the page and could plot a candlestick graph based on the stock symbol and date range. Then came the challenge of implementing the technical indicators. I began using a single dropdown to only plot one at a time before realizing multiple would be better. Below show the options for updating the graphs or adding new ones.



Please enter your chosen desired stock below:

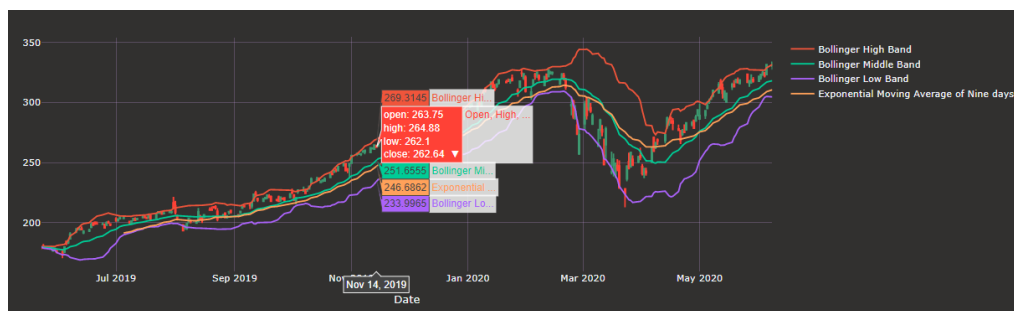
Input your start and finish dates:

→

Select one or more technical indicator:

☒ Bollinger Bands ☒ Exponential Moving Average

I began with the overlays as they seemed the easiest to do. The callback works by updating the figure of the candlestick chart when a dropdown has been selected. Below we can see the final product, this was the easier part as it did not require me creating new graphs.



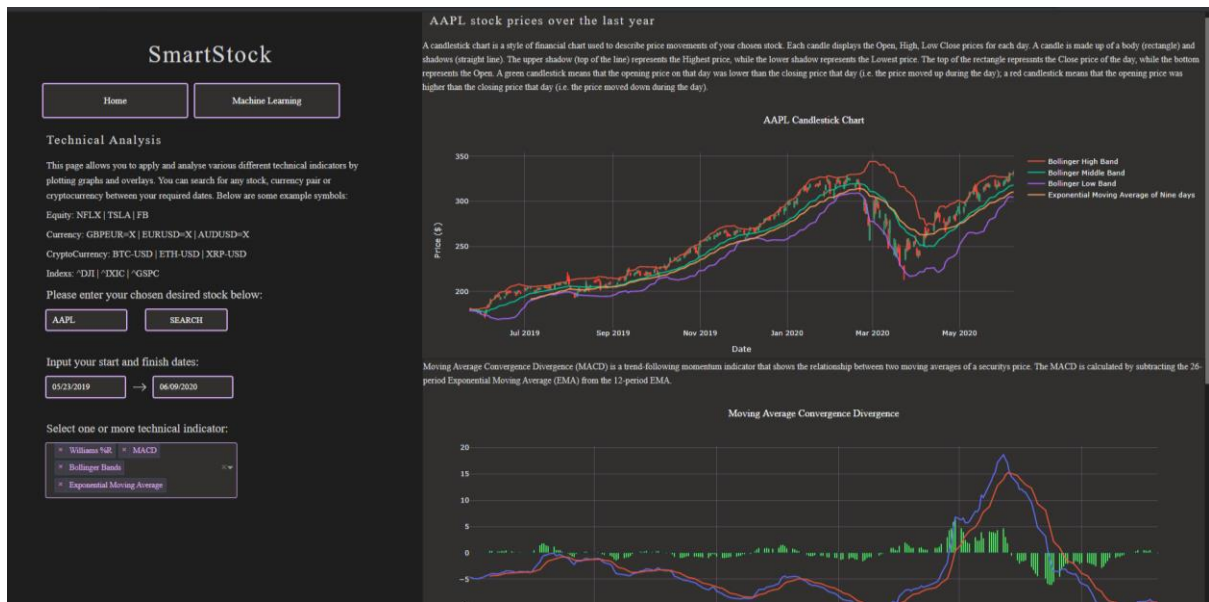
Then came the challenge of adding new graphs based on the selected technical indicator. I did this by creating Div elements for each graph. Within the Divs I included a unique id for the div and a graph with a unique id. Then using the callback, when a user selects one from the dropdown it toggles the div on and makes it display. It also creates the specific graph. For each graph I used the website TradingView to ensure I was creating them correctly. [\[11\]](#)

From using Plotly within my models I was comfortable in creating different plots such as Bars or Scatters. But TradingView came in handy to ensure that they were made correctly. The below images show TradingViews plots of Williams %R and Aroon Oscillator and my

version and as you can see, they are very alike. For example, I did not know that Williams %R used -20 and -80 and guidelines for when a stock is overbought or oversold until I investigated further and analysed the chart on TradingView. I could then simply add these lines in just like TradingView has.

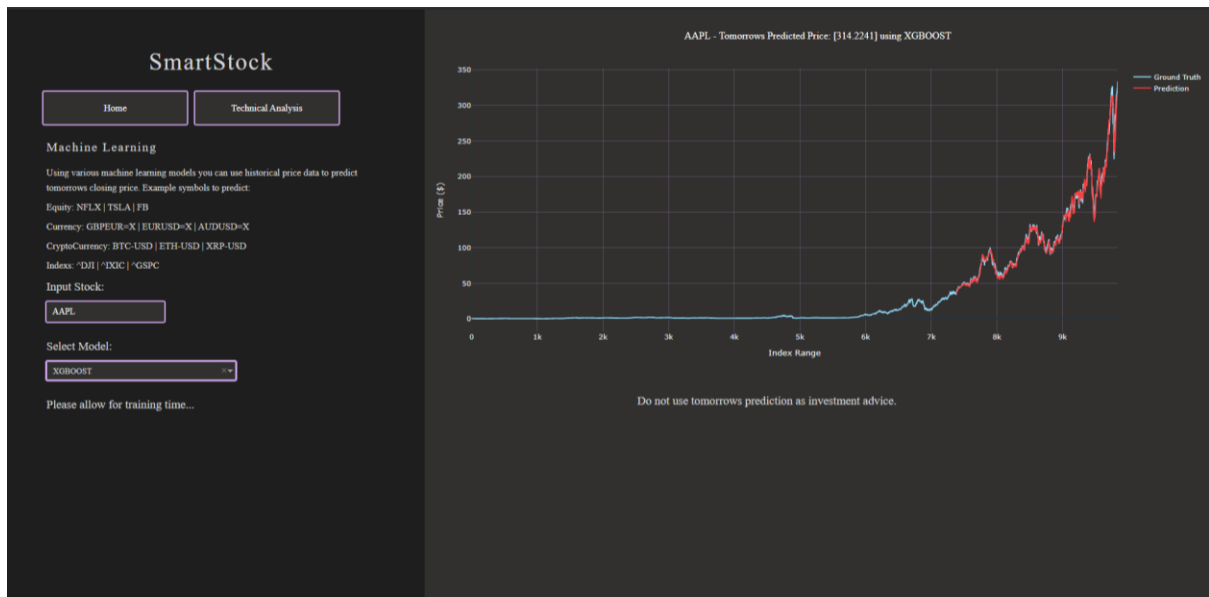


I could then start on adding other details such as the descriptions and working on the design. The finished Technical Analysis page looks like this:



7.3. Machine Learning

Moving onto the Machine Learning page I was sceptical on how to go about it as I could not find many examples using live models within the applications. I started to investigate saving the results of the models into an SQLite database and then reading them in based on user input. However, this would limit it to only a few stock symbols as I would never be able to run the models and save predictions for every single symbol. Another downside is they would not be current predictions unless every morning I updated the models and saved the new predictions. Therefore, I stuck to my original idea of training the models live within the application. This would allow the user to test it with any symbol and have the most up-to-date predictions. It turned out to be simpler than I thought. Once the Jupyter Notebook models were finished I could save them as Python files. Similarly, to the technical analysis page I use a callback to output the updated graph based on the dropdown list. When a user selects a model, it will begin running the complete algorithm. It works by outputting only the true values and predicted values and plotting them onto the graph. I created a variable 'prediction' which also updates and represents the price for tomorrows prediction. Below is the final output.

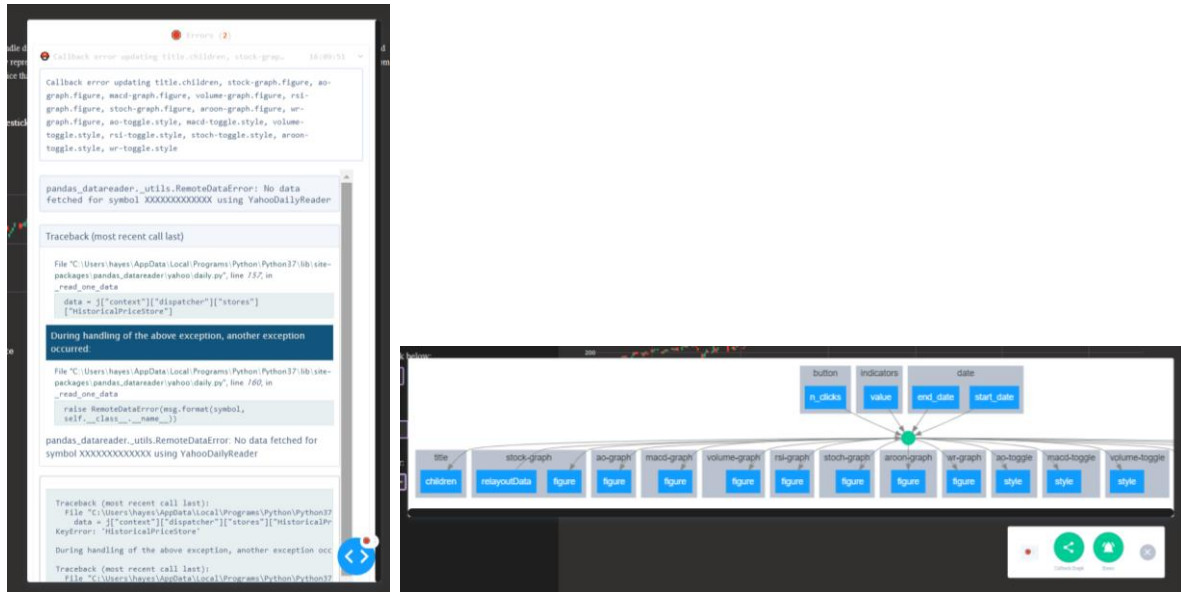


7.4. Design

So many dash applications just use external stylesheets such as the default one. But I wanted to add my own design to it and achieve a look I liked. Therefore, I added the external stylesheet within the assets folder and began editing it. Some of it simply required me to change the pre-defined code such as the background colour, but most of it required me inspecting the CSS within the browser to find what element needed changing. As a lot of these elements were built in, I had to override them and add them to the CSS file. For the graph designs I used the go.Layout design and edited them with this. I come across a few problems when trying to edit graphs with go.Layout but the Plotly community was very helpful and most of the time my problem had already been answered. [\[12\]](#) I kept the design basic and clear to ensure it focuses on the data visualization. I prefer dark themes to light ones so went with using darker colours for the background and fills, while the outlines and text would be a light colour. I made sure the graphs were various colours and stood out clearly so each could be distinguished distinctly.

7.5. Testing

Dash by Plotly allows us to debug our applications in a much more effective way. When we set 'debug=True' we can see detailed error messages and callback graphs within the application. This allowed me to understand errors more thoroughly and pick up problems within the application while testing it myself. Here we can see an error message within the application and a callback graph to visualize the structure of the code clearer.



Due to unforeseen circumstances I was unable to carry out testing how I intended. I would have liked to carry out a survey of feedback on a group or participants early on to gather improvements to the application which could be implemented. However, I managed to use 3 participants to test the application once I believed it was finished to ensure there was no errors.

Participant	Positives	Improvements
1	Interactive elements are good, allows you to see different outputs of the data.	Do not know what to enter as a symbol section. Maybe some examples or a dropdown box like the technical indicators.
2	Clear layout and structure to the pages, made it easy to navigate.	What does each graph mean? Or what does each represent. A description would be good.
3	The graphs were informative, and you can zoom in/out of them to look at the values closer, or hover over and see even more details.	The graph on the second page is not very clear and the two colours do not complement each other well.

Using the feedback, I could make some adjustments to the application. I decided to add a little section including examples of symbols you can enter in the search box. I also added small descriptions of what each of the technical indicators are above their graphs to explain them better. The prediction graph which shows the actual vs predicted values used two colours which were quite similar, this meant the predictions didn't stand out very well, I changed them to two colours which stood out from each other – red and blue.

Once I had implemented the feedback, I could finalize the application and push it to GitHub.

7.6. Ethics

As my application is related to investments and trading some users may be unaware of the dangers and risk that trading brings. I wanted to ensure that the information produced by the application is not used in the wrong way. For instance a prediction for tomorrows closing price may be greater than the current price, this could be seen as a opportunity to buy shares in that stock, however there is always the chance the price could decrease and they lose money. Therefore, I ensured that I make it clear than the application is not investment advice by stating it on the index page and after tomorrows price prediction on the machine learning page.

8. Problems Encountered

The early problem I encountered was gathering sentiment data to enhance my models, this proved quite a difficult task as many data outlets either charge a fee or do not offer enough data. As my model's historical data dated back a minimum of 10 years it would have been difficult to match it. Twitter and NewsAPI were two sources I looked at and both were limited, NewsAPI only allowed for 30 days of headlines and 100 headlines in total. Therefore, I decided to remove sentiment as a feature from my models. Maybe if I had access to minute OHLC data rather than daily I could have trained my models to predict small movements minutely. Below shows where I got to regarding creating sentiment as a feature, I used TextBlob to calculate the polarity and subjectivity over the 30 days of news headlines.

	polarity	subjectivity
date		
2020, 04, 29	-0.033854	0.272396
2020, 04, 30	-0.027024	0.369643
2020, 05, 01	-0.034061	0.468909
2020, 05, 02	-0.025000	0.175000
2020, 05, 04	0.160000	0.540000
2020, 05, 05	0.083468	0.337239
2020, 05, 06	0.117083	0.369167
2020, 05, 07	0.215278	0.583333
2020, 05, 08	0.111111	0.250000
2020, 05, 09	0.092857	0.223810
2020, 05, 11	-0.040079	0.523413
2020, 05, 12	0.008333	0.093519
2020, 05, 13	0.086212	0.538485
2020, 05, 14	0.329545	0.594318
2020, 05, 16	0.100000	0.050000
2020, 05, 18	0.000000	0.000000
2020, 05, 19	0.028472	0.160648
2020, 05, 22	0.000000	0.000000
2020, 05, 23	0.083333	0.111111
2020, 05, 24	0.312500	0.537500
2020, 05, 26	-0.020455	0.131818
2020, 05, 27	0.043750	0.459375

When developing my Dash application and reviewing example Dash applications I found many of them were used only one page and squeezed multiple graphs and data onto one page. This led to me wanting to create a Dash application with multiple pages and I found you could. I began using the tutorials supplied by Dash. [13] I wanted to have multiple python files for each page to make it easier to structure and follow, however the tutorial for this would not work and returned an error every time I ran the application. As I could not find a way to fix this, I had to create one python file and have each page within the one file which in return created a somewhat large, complex file.

The last main problem I had was introducing a third page into my application. As I had worked on using the NewsAPI and gained a polarity and subjectivity score for each day I thought it would be a good addition to the application. One of the NewsAPI parameters is 'search query' so the idea was the user would be able to input a search term i.e. Facebook. It also has a date range parameter so to make the application current and live I set it from today to today-1 so it returns all the headlines over the past day that include this term within the headline title or summary. Lastly, it has a parameter for specific news outlets you want to retrieve the headlines from, so I input the main financial ones such as Bloomberg, Reuters etc. and created a dropdown to select from. As you can see below, I implemented the page, it worked and retrieved headlines based on the input term and from the selected news outlet.

Title	Date	Source	Summary
Twitter's ex-CEO stepped up the Silicon Valley beef and attacked Facebook	2020-05-29	Business Insider	cul> cul>There's a big spat between Twitter and Facebook right now and
The irony is, if Section 230 were to die like Trump says he wants, Twitter	2020-05-29	Business Insider	cul> cul>1506 President Trump's anger at Twitter escalates, his so-called
10 things you need to know before the opening bell	2020-05-29	Business Insider	cul> cul>Here's what you need to know before the markets open. 1. The coronavirus
A partner who attended the now-infamous Lake of the Ozarks pool party	2020-05-29	Business Insider	cul> cul>1514 Missouri resident who attended a crowded pool party at the
US billionaires have increased their net worth by \$405 billion during	2020-05-29	Business Insider	cul> cul>1505 billionaires added almost half a trillion dollars to their
BANK OF AMERICA: Funds are falling out of love with financial stocks	2020-05-29	Business Insider	cul> cul>151Large-cap long-only funds are the most underweight in finance
A Wall Street fire studied every crash over the past 100 years - and	2020-05-29	Business Insider	cul> cul>151The stock market's resurgence from its recent crash has large
The anti-vax movement is using growing hesitation around the coronavirus	2020-05-29	Business Insider	cul> cul>Associated Press cul> cul>15Some recent surveys suggest Americans, espe
ADU's no-refrigeration-needed kombucha is the perfect light summer drink	2020-05-29	Business Insider	cul> cul>151Kombucha is a tasty beverage touted for its potential health
Shipping Gears: Meet Amazon's most important man who's not Jeff Bezos	2020-05-29	Business Insider	cul> cul>Happy Friday and welcome to another issue of Shipping Gears, Business
Vaccine and coronavirus skeptics packed into a hotel for a conference	2020-05-29	Business Insider	cul> cul>Andie Res cul> cul>151About 200 vaccine and coronavirus skeptics gathere
Don't make social media tech bro billionaires the arbiters of truth	2020-05-29	Business Insider	cul> cul>151It's hard to believe that in an election year, during a pande
Google's potential stake in Vodafone Idea would accelerate US tech giant's	2020-05-29	Business Insider	cul> cul>151Business Insider Intelligence and elsewhere are now Insider s
Horses who have the coronavirus are fighting their employers to get a	2020-05-29	Business Insider	cul> cul>151Horses are on the frontlines of the coronavirus pandemic, and
Trump's feud with Twitter, explained in 30 seconds	2020-05-29	Business Insider	cul> cul>151President Trump and Twitter have been embroiled in a feud th
Tesla's cars aren't perfect - here are all their most disappointing features	2020-05-29	Business Insider	cul> cul>151Tesla's vehicles are generally impressive. cul> cul>151But the
10 things in tech you need to know today	2020-05-29	Business Insider	cul> cul>Good morning! This is the tech news you need to know this Friday, cul>
Mark Zuckerberg says he had a 'visceral negative reaction' to Trump's	2020-05-29	Business Insider	cul> cul>Mark Zuckerberg has broken his silence on Donald Trump's scolding about
President Trump's clout on social media could hurt startups and companies	2020-05-29	Business Insider	cul> cul>151President Donald Trump just signed an executive order that co
A political strategist turned VC thinks Trump's war with Twitter and	2020-05-29	Business Insider	cul> cul>151Bradley Turk, founder and CEO of Turk Venture Partners, told
Trump is going to war with social media companies like Facebook and Twitter	2020-05-29	Business Insider	cul> cul>151President Donald Trump is calling for the repeal of a law th
The \$155,000 Mercedes-AMG GLS 63 is an incredibly luxurious large SUV	2020-05-29	Business Insider	cul> cul>151I tested a \$155,000 Mercedes-AMG GLS 63, a high-performance
Microsoft News just cut dozens of editorial workers as it shifts to a	2020-05-29	Business Insider	cul> cul>151Microsoft News has just shed dozens of contractors as it mov
US billionaires have increased their net worth by \$405 billion during	2020-05-29	Business Insider	cul> cul>1505 billionaires added almost half a trillion dollars to their
BANK OF AMERICA: Funds are falling out of love with financial stocks	2020-05-29	Business Insider	cul> cul>151Large-cap long-only funds are the most underweight in finance
Facebook has done nothing with a Trump post that threatens shooting M	2020-05-29	Business Insider	cul> cul>151On Monday, a 46-year-old black man named George Floyd died i
Twitter just slapped a 'glorifying violence' label on a Trump tweet	2020-05-29	Business Insider	cul> cul>151Twitter just placed a click-through block on a tweet from Do
Trump and Biden both want to revoke Section 230, but for different reasons	2020-05-29	Business Insider	cul> cul>151President Donald Trump and likely Democratic pres cul> cul>151
Trump has officially declared war on Twitter and Facebook. Here's the	2020-05-29	Business Insider	cul> cul>151Donald Trump signed an executive order on Thursday, cul>
Microsoft asks staff each year if they think their compensation is co	2020-05-29	Business Insider	cul> cul>151Internal Microsoft poll results reviewed by Business Insider

However, for some reason it ruined the design for the other pages, particularly the buttons. I spent hours trying to figure out why this affected the other pages and messed up the layouts and assumed it was the CSS of the buttons conflicting. After removing the buttons, it still affected the rest of the application, so the problem was the data table. As Dash uses a pre-defined, built in style sheet, I had to inspect the web page while it was running and after days editing the code and overriding the built in style there was no solution so I had to remove the page entirely to save the other two pages looking poorly designed.

9. Improvements

Leading on from the last problem encountered I would have liked to include this page as news can be a great tool for investors along with technical analysis. As the data table was the problem introducing a pie chart of positive/negative sentiment news over the past day could have been a solution.

Another application improvement would be to include evaluation metrics below the actual versus predicted graph on the machine learning page. I include the metrics within my models so it would be beneficial to a user to see them to gain a better insight into the model they have selected. On the technical analysis page, I include a brief description of the technical indicator that has been selected, I could implement this into the machine learning page to display a description of the model the user has selected. It could be done using a simple div below the graph which is toggled on and off when the user selects the specific model.

There is always room for improvement within my models and enhancing them to be more reliable and accurate can be done. I could try to optimize the parameters better by adding extra values within the grid search. As well as maybe adding new ones. I could focus more on the classifiers too, making them more detailed. This could lead to a classification page being introduced to the application which predicts whether to buy or sell their input symbol.

10. Successful Aspects

I am proud that I managed to successfully create multiple regression models which can predict the next day's price relatively well. I never expected the predictions to be perfect but from my models they all seem to achieve predicting the correct movement of price. I found implementing the models into the application a daunting task as I did not know how to get them to train live within the application. Once I had developed the first page and understood how callbacks worked it became clearer on how it would work. Successfully implementing the models so they can train and predict any symbol live was a great achievement. I found it enjoyable to add my own twist on the application by editing the CSS rather than using the default stylesheet, this brought it a bit more to life and allowed me to express my design skills. The default stylesheet was quite plain and had a simple linear structure. I like the way the application is very interactive and allows for various user inputs to make it dynamic. Being able to search for any symbol you like is intriguing as it does not limit you to a few selected stocks as well as having a date range which can make analysis much more detailed.

Other personal successes of the project are I am now more comfortable using Python and have learnt a variety of new tools and libraries which can enhance my development of Python projects in the future. Dash is a powerful and great web application framework which I am glad I found and is certainly something I will use again as I now feel confident with it. Lastly, I have felt my skills within creating machine learning models has increased

and I have understood new aspects of modelling. For example, how normalization can greatly increase the precision of a models results.

11. Future Developments

I began looking into developing a Long Short-Term Memory (LSTM) model for the regression task and found a good example. A future goal would be to investigate into neural networks and develop a model myself using technical indicators. I found many papers which have been used for predicting the movement of a stock price and few on predicting the actual price. I would also include technical indicators as features as not many previous investigations do this. I could then implement this into model into application.

I set out to develop a ‘trading algorithm’ and while my models somewhat fit this, they are not automated and able to make trades. A certain future goal is to begin developing simple algorithms which are connected to broker APIs such as Alpaca [\[14\]](#) for placing automated trades. Now I have a better understanding of machine learning my models could be implemented into a trading algorithm.

Dash is a great tool and I want to begin developing more applications using this framework to expand my data science and visualization skills. In regards to my SmartStock application I would like to deploy it to a server such as Heroku so it is available to anyone with the URL, rather than only being able to run it locally, this would be beneficial to future employers.

12. Conclusion

To conclude I have successfully created 6 regression models which have been optimized using grid searches to predict the next day’s closing price of any given stock. I have also developed multiple classification models to predict the increase or decrease of a stock’s price. I then developed a web application using Dash by Plotly whereby a user can search for any symbol, enter a date range, and select technical indicators to analyse their input symbol using multiple graphs and overlays. It also demonstrates my regression models and allows a user to train and test any symbol they like. Working on a project like this has been challenging as well as rewarding and I have learnt many skills from it including technical ones such as creating a web application which trains and predicts models in real time, to industry skills such as requirements analysis and project management. This project has broadened my future aspirations and has made me consider entering the world of data.

References

- [1] European Central Bank, Algorithmic trading: trends and existing regulation, 03/02/2020 https://www.bankingsupervision.europa.eu/press/publications/newsletter/2019/html/ssm.nl190213_5.en.html
- [2] Adam Satariano and Nishant Kumar, 2017, The Massive Hedge Fund Betting on AI, Bloomberg Markets, 03/03/2020, <https://www.bloomberg.com/news/features/2017-09-27/the-massive-hedge-fund-betting-on-ai>
- [3] Slava Kurilyak, 2017, Artificial Intelligence (AI) in Hedge Funds, Produvia, 08/03/2020, <https://blog.produvia.com/artificial-intelligence-ai-in-hedge-funds-c4c10f95f903>
- [4] N P Samarth, Gowtham V Bhat, Hema N, 2019, Stock Price Prediction, 14/03/2020, <https://www.google.com/search?q=Stock+Price+Prediction+N+P+Samarth%2C+Gowtham+V+Bhat%2C+Hema+N&oq=Stock+Price+Prediction+N+P+Samarth%2C+Gowtham+V+Bhat%2C+Hema+N&aqs=chrome.69i59j0j8&sourceid=chrome&ie=UTF-8#>
- [5] Harsh Sahu, 2019, Stock Prediction with XGBoost: A Technical Indicators' approach, Medium, 20/03/2020, <https://medium.com/@hsahu/stock-prediction-with-xgboost-a-technical-indicators-approach-5f7e5940e9e3>
- [6] Dash by Plotly, Introduction to Dash, 10/04/2020, <https://dash.plotly.com/introduction>
- [7] Dash by Plotly, Dash App Gallery, 14/04/2020, <https://dash-gallery.plotly.host/Portal/>
- [8] mrjlbq7, TA-Lib, GitHub, 02/04/2020, <http://mrjlbq7.github.io/ta-lib/>
- [9] Saed Sayad, Decision Tree – Regression, 15/04/2020, https://saedsayad.com/decision_tree_reg.htm#:~:text=Decision%20Tree%20%2D%20Regression,decision%20nodes%20and%20leaf%20nodes.
- [10] Plotly, 2017, Dash in 5 Minutes, YouTube, 03/02/2020, <https://www.youtube.com/watch?v=e4ti2fCpXMI>
- [11] TradingView, 20/04/2020, <https://uk.tradingview.com/>
- [12] TraceLD, 2019, How to change the color of those lines?, Plotly Community, 15/05/2020, <https://community.plotly.com/t/how-to-change-the-color-of-those-lines/24814>
- [13] Dash by Plotly, Multi-Page Apps and URL Support, 04/03/2020, <https://dash.plotly.com/urls>
- [14] Alpaca, API for Stock Trading, 20/05/2020, <https://alpaca.markets/>

Appendix 1

Date: 08/10/2019

Initial email regarding the project, planned to discuss it further and the ethics form.

Date: 10/10/2019

Meeting regarding the proposal of the project and the brief for it. Also discussed about the project's final outcome and the process of the project. Lastly, we spoke on the problems with data as well as the ethics of it.

Date: 08/11/2019

Meeting on various project topics: The project planning elements of the project. What the outcome and main deliverable, a GUI application with user interactivity, ability to select different models. How to evaluate/analyse machine learning models.

Date: 15/11/2019

Project management methodology and requirements gathering techniques discussed.

Date: 21/11/2019

Viva meeting regarding the project proposal. Tips on minimizing the scope of the project as it was broad and may need to be focused on one element. The project management methodology was more waterfall than agile.

Date: 24/04/2020

Teams video call. How to test/evaluate models. Hosting the project or running locally. How to go about demonstration.

Date: 22/05/2020

Check on project development, any issues.

Date: 06/06/2020

Submission requirements check.

Appendix 2

Jupyter Notebook folder contains all the developments of the models including the grid search models and final models. Other notebooks include data analysis and gathering sentiment using NewsAPI and the classification models.

Models folder contains the finalised models within Python.

SmartStockApp folder contains the dash application files including the Python app file and CSS file.

Link to source code files:

<https://github.com/ro-stack/SmartStock>

Appendix 3

The demonstration can be seen within the PowerPoint presentation file which is uploaded to the SmartStock GitHub repository on the slide 'Demonstration' as an .mp4 video or via the link below on Microsoft Stream.

Link to project demonstration:

<https://web.microsoftstream.com/video/d1675b5b-b07e-44ae-b481-4d6f6a6a9bee>