Exploring Machine Learning Models in Stock Market Prediction Across Stock Types and Market Conditions

School of Computer Science and Informatics, Cardiff University

Computer Science (BSc)

Author: Joel Westcott

Supervisor: Yuhua Li

Moderator: Alexia Zoumpoulaki

15/05/2025

Acknowledgements

I would like to thank my family and friends for supporting me through the process of completing this project.

I would also like to express my gratitude to my project supervisor Yuhua Li, for his support and guidance throughout the project.

Abstract

This project aims to explore and compare the effectiveness of Machine Learning models in stock market prediction across different stock types and under varying market conditions. This exploration is achieved by implementing and comparing the predictive accuracy of four Machine Learning models – SVR, XGBoost, LSTM and RNN – to understand how well they perform across diverse scenarios and stock profiles. These models will then be evaluated using XAI techniques, to hopefully identify the most influential model features driving predictions.

Whilst there is an extensive volume of literature exploring and comparing a wide variety of models in their ability to predict the stock market, there is a limited amount of data exploring Machine Learning models with a specific focus on stock types and market conditions. This project addresses that gap by providing insights into which models are better suited for different stock profiles, and which models perform better in times of market volatility.

The findings from this project aim to help practitioners and researchers in making informed decisions on model selection based on market temperature and stock characteristics.

Table of Contents

Introduction

Stock market prediction using Machine Learning is an area that has been heavily researched since the 1980s, with increasingly sophisticated computational methods and financial indicators being employed over the years to accurately forecast financial trends. As such, there exists a large volume of literature comparing different Machine Learning models and their effectiveness in predicting the stock market. However, much of this existing work tends to focus on the overall accuracy of models on one given time period or stock type (Strader et al. 2020), without looking deeper into contextual factors that may influence the effectiveness of a Machine Learning model at making predictions, for example by exploring the category of stock which is being predicted and the market conditions at the time of the prediction. In doing so, these studies limit the generalisability of their results to real world situations.

This project attempts to fill this gap in research by exploring the accuracy and effectiveness of Machine Learning models across different stock profiles and under varying market conditions. As different stock types exhibit distinct behaviours and patterns, using one blanket accuracy value to measure a model’s performance across stock types is misleading and not applicable in a real-world financial context. The same can be said for model accuracy in varying market conditions, as the level of volatility during a model’s forecast period can have a significant impact on a model’s ability to make an accurate prediction. Therefore, a comparative analysis that does not factor in market conditions and stock types could potentially be considered unreliable and unapplicable to real-world financial forecasting tasks, where these factors need to be considered. By factoring in stock profiles and market conditions this project should provide a nuanced and applicable contribution to model selection.

This project will involve the implementation of 4 Machine Learning models: SVR, XGBoost, RNN and LSTM. These models will then be compared on their accuracy in forecasting stock price values across different types of stocks, and on their ability to maintain accuracy in times of market instability. Once the relative performance of each model is established, XAI techniques will be employed in order to assess why certain models outperform others under different conditions and circumstances, which will hopefully serve to highlight the areas in which these models can be improved to overcome any potential deficiencies.

The overall purpose of this project is to provide researchers, practitioners and stock investors with deeper insights into model selection and interpretability, so that they may apply Machine Learning more effectively in real life financial scenarios and gain a clearer understanding of why their models produce the results they do.

Background

Literature Review

Comparing Machine Learning models in the accuracy of their stock market prediction is not a novel topic, and as such there is a breadth of research in this area. One such study is Kumar et al. (2018), in which the authors performed a comparative analysis of traditional Machine Learning methods and determined the Random Forest algorithm to be the most effective model in terms of prediction accuracy. Nabipour et al. (2020) compares the prediction of both traditional and deep learning Machine Learning algorithms in predicting stock market trends to determine the deep learning algorithms RNN and LSTM to be the superior models in terms of predictive accuracy. Lakshminarayanan and McCrae (2019) compares SVM and LSTM models in stock market prediction and determines LSTM to be the superior model in all scenarios. There are many more examples of studies comparing Machine Learning models in stock price prediction, as can be seen in the literature review Mintarya et al. (2022) which outlines 30 studies relevant to the topic of Machine Learning approaches in stock market prediction.

However, several of the studies comparing Machine Learning models in stock prediction focus on either predictions made on only one stock (Shobayo et al, 2025),(Abunasser et al, 2023), or focus on a range of stocks and produce one average accuracy value per model without separating the stocks based on profile and behaviour (Kumar et al. 2018),(Chowdhury et al, 2024), therefore overlooking the key contextual differences between the stock types that need to be accounted for in real-world stock forecasting.

There are some examples of research that factor stock categories into their Machine Learning comparisons. Nabipour et al. (2020) explores four stock market groups in its comparative analysis – petroleum, diversified financials, basic metals and non-metallic minerals. Barua et al. (2024) compares deep learning model performance in predicting 5 different major companies in the Indian stock market to fill the same data gap identified in this project. This area of research is still however somewhat limited in the context of the US market.

There is some research into statistical modelling under varying market conditions, take for example Camska and Klecka (2020), which compares models across periods of economic recession and expanse. This paper highlights the fact that model predictions will produce different results in varying market conditions, which makes the limited research into comparing Machine Learning models specifically in varying market conditions somewhat surprising.

Evidently, there is somewhat of a gap to be filled in terms of comparing Machine Learning models across stock profiles and categories in the US market. A review of Machine Learning Stock Prediction studies conducted in 2020 (Strader et al, 2020), suggests that studies focus too much on using one market/time-period without consideration for the model's generalisability to other situations, with times of market downturn being mentioned specifically. This project fills that gap of studies by comparing each model's ability to make predictions in a period of economic downturn to their ability to make predictions in a relatively stable period.

Neural Networks

Neural networks are a family of Machine Learning algorithms that are designed in a way that is inspired by the structure of the human brain. Neural networks consist of a structure of interconnected nodes, where each connection between nodes has an associated weight that represents the strength of the connection. These weights determine how input data is transformed as it passes through the network, ultimately influencing the network's output. Neural networks can then adjust the weights of these edges during the training process to reduce the difference between its predictions and the actual outcomes, which captures patterns in the data. This adaptive learning process allows for the modelling of complex patterns and relationships in data, making Neural Networks ideal for stock market analysis.

Recurrent Neural Network (RNN):

RNN or Recurrent Neural Network is a form or Neural Network that is specifically designed to process sequential data, making it well suited to predict stock market data. RNNs differ from regular neural networks as they are structured as a recursive loop, meaning that information is maintained throughout iterations.

Below is a diagram of a RNN, where A is the model itself, X_t is some input and h_t is the output value.

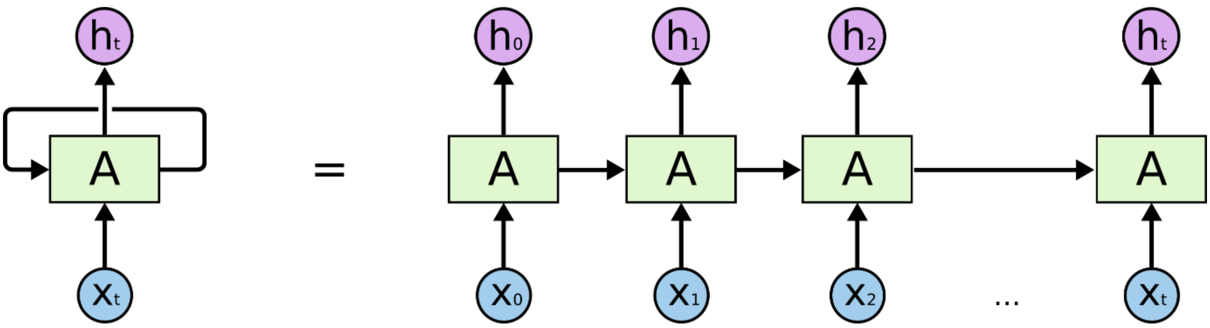

Figure 1: RNN Structural Diagram (Olah 2015)

On the left side of Figure 1 we can see a typical RNN loop – data X_t comes in, the RNN A recurses on itself and outputs value h_t. On the right side of the equals sign we can see this same RNN loop unravelled to show its inner structure and sequential nature. In this unravelled diagram, we can clearly see how each iteration of the RNN is fed into itself as an input along with the new input data. This allows for the RNN to maintain itself and its memory from each previous step, allowing for a strong inference of patterns over time.

Long Short-Term Memory (LSTM)

LSTM networks were first proposed in a research paper by Hochreiter (1997) as a solution to a major limitation of traditional recurrent networks. The limitation in question being that the traditional recurrent networks struggle with retaining information over long sequences. As a result, these networks struggle with learning long-term dependencies in sequential data – which can be a particular problem in predicting stock market data, where long-term trends are crucial to contributing to accurate predictions.

LSTM therefore builds on the standard RNN architecture by incorporating a more sophisticated memory system into its structure. LSTM uses memory cells, which store information from previous states, allowing the retention of relevant information over a long period of time. To prevent irrelevant data form being preserved or amplified, LSTM employs complex gating mechanisms which regulate what information is retained or discarded. This comes in the form of 3 gates that are present in each memory cell in the LSTM model - the Input gate, which determines the information that is added to the memory cell, the Forget gate, which controls what information is discarded from the memory cell and the Output gate which determines how much information from the current cell state is shown to the next layer in the network.

This complex architecture makes LSTM particularly effective at capturing long-term dependencies, making it a well-suited model for the learning of long-term market trends.

Support Vector Regression (SVR)

Support Vector Regression or SVR is a Machine Learning algorithm designed for regression tasks that was derived from the Support Vector Machine (SVM) framework, which was originally designed for classification (Drucker et al, 1997). SVR models attempt to find a function that best fits the training data by employing an error margin defined by a parameter epsilon (ε). Here, instead of minimising the error for every provided data point, SVR aims to keep predictions within this error margin, to ignore small deviations and focus on larger, more significant errors.

Figure 2 shows an example of SVR being applied to a sample of continuous numerical data. Each data point is represented by a circle, with each circle's shade of blue indicating its value – darker blue means higher values, lighter blue means lower values. The black solid line represents the SVR models prediction, being the SVR model's best estimate of the data trend. The dashed lines labelled "ε-tube" represent the margin of error defined by the parameter epsilon. Here we can see the points within the margin of error which receive no penalty are marked with a grey outline, and datapoints outside or on the boundary of the margin of error are marked with a black outline. These errors are known as support vectors, and as they are the datapoints that will contribute to defining and changing the regression function due to their large error contribution. Separating the errors in this manner allows for the SVR model to focus on minimising only the most important errors.

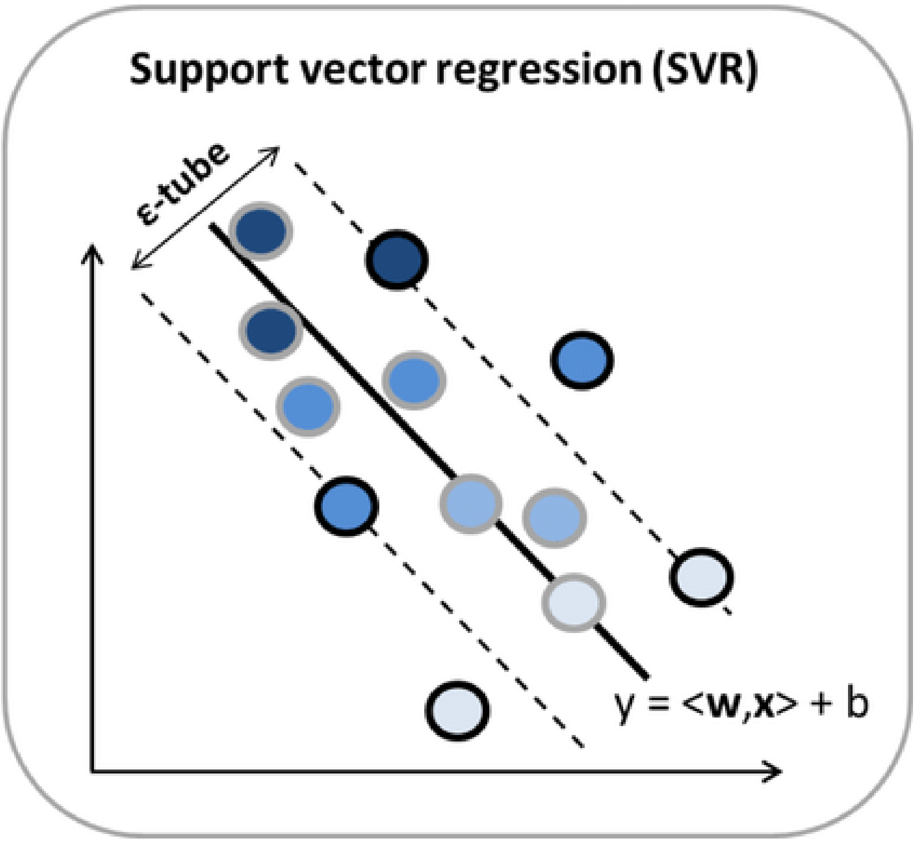

Figure 2: Diagram of SVR Prediction Training Process (Rodríguez-Pérez et al. 2022)

eXtreme Gradient Boosting (XGBoost)

The term Gradient Boosting can be defined as an ensemble Machine Learning method that essentially combines multiple weak predictors to create a single powerful model, whereby each new training model attempts to improve on the former (Friedman, 1999). XGBoost is an optimised implementation of gradient boosting which uses a number of techniques to improve upon the accuracy and speed of traditional gradient boosting model, most notably through advanced regularization and parallelisation:

Regularization is the process of adding a penalty to training data to prevent overfitting and enhance generalisation. In the context of XGBoost, regularization helps reduce the overfitting effects that come from combining a large number of decision trees, allowing the model to maintain an accurate prediction despite the complexity.

Parallelisation is the process of running multiple tasks simultaneously in order to complete each task quicker. XGBoost implements parallelisation during the construction of decision trees, allowing the models to process large datasets much faster than traditional gradient boosting methods.

Explainable Artificial Intelligence (XAI)

Explainable Artificial Intelligence is a field of AI that focuses on making AI systems more transparent and interpretable to humans. In this project XAI techniques are used to interpret models and provide an insight into where their predictions come from.

Shapley Values (SHAP)

SHAP values are a way of measuring the marginal contribution of a each feature to a prediction made by a Machine Learning model. This is calculated by removing a certain feature value from a prediction, and computing the difference of prediction accuracy with it removed vs the actual model accuracy. This can help show the relevancy and the extent of contribution of each feature to the model's prediction.

Methodology

Problem

The central problem for this project was to implement 4 separate Machine Learning models ready for subsequent investigation and analysis. The 4 Machine Learning models developed were: SVR, XGBoost, RNN and LSTM. These models were carefully selected to ensure a diverse and representative sample set of models for comparison, with a combination of both traditional Machine Learning algorithms (SVR and XGBoost) and deep learning algorithms (RNN and LSTM). The challenge in developing these models is that each model needs to be programmed in a different way, and each presents their own problems and constraints. Implementing all four within the project's limited timeframe made the technical side of this project an ambitious and demanding task.

Approach

Stock Selection

To explore model accuracy across different stock types, the stocks on which the models made predictions were categorised into three distinct groups: Growth Stocks, Declining Stocks and Stable Stocks. This allows for the evaluation of each model's ability to predict different types of stock characteristics and trends. Each stock type includes three representative companies, making a sample of 9 stocks that the results are produced from. The 3 categories are:

Growth Stocks: These are stocks that have demonstrated rapid recent growth. These stocks will test the ability of the Machine Learning algorithms to recognise and learn significant upwards trends. The 3 stocks selected in this category are:

- Nvidia (NVDA): Nvidia has seen large recent growth due to their leadership in GPU technology
- Tesla (TSLA): Tesla has seen a rapid growth due to the rising global popularity of electric vehicles
- Meta (META): Meta has seen a rapid growth in the past few years due to a substantial investment in AI products

Declining Stocks: These are stocks that have seen a significant decline in value in the last 3 years, challenging each models ability to effectively recognise downward trends. The 3 stocks selected in this category are:

- Intel (INTC): Intel has seen somewhat of a recent decline due to manufacturing delays and increased competition
- GlobalFoundries (GFS): Semi-conductor manufacturing company that has seen significant stock decline in recent times due to drops in revenue
- Walgreens Boots Alliance (WBA): Multi-national pharmaceutical company that has seen rapid decline over the last decade, somewhat due to a shift to online retailers causing falling profits

Stable Stocks: Stocks that maintain relatively consistent stock prices regardless of market volatility, testing the ability of the models to make accurate predictions for stocks that see minimal fluctuations. The 3 stocks selected in this category are:

- Procter & Gamble (PG): Procter & Gamble provide essential consumer goods, such as household goods and cleaning products, making them a fairly stable stock option.
- Coca-Cola Co (KO): As one of the largest household names for a consumer good in the world, Coca-Cola sees steady levels of consumer demand regardless of market fluctuations.
- Johnson & Johnson (JNJ): As a global pharmaceutical and healthcare company, Johnson & Johnson is able to retain consumer demand throughout market diversions.

This range of data will provide a solid basis for the evaluation of each Machine Learning model across stock types and market conditions.

To aid this project's investigation three datasets are generated using the categories above, each using a different timespan.

The first dataset is a general dataset that will be used to compare each model's ability to make accurate stock predictions based on stock type. This set of stocks spans from May 23rd 2022 to the April 21st 2025, with 1 hour time intervals between price values.

The other two datasets are used to compare each model's ability to predict stock market data in varying market conditions. More specifically, to compare each model's ability to predict stock data in periods of market volatility/instability. To achieve this comparison two datasets are generated – one dataset with stock values generated in a timespan with relatively stable market conditions and one dataset in a timespan with instable market conditions:

The dataset Group A is used a control group, consisting of stocks that are situated in a period of low market volatility. The Group A stocks will span from January 23rd, 2025, to February 28th, 2025, with 1 hour time intervals between price values.

The dataset Group B is used to measure the ability of each model to predict data in turbulent market conditions, to be compared to the values in the control set Group A. As such this dataset consists of stocks that are situated in a period of high market volatility. The Group B stocks will span from March 1st, 2025 to April 7th 2025 and will also use 1 hour intervals.

Both datasets contain the same number of trading days within their timespan in order to keep the comparison as fair as possible – i.e. one group does not have significantly more data to train on than the other. Additionally, the data from Group B takes place directly after the data from Group A, ensuring that the two datasets are in a similar timespan, minimising external variables.

The market volatility observed in Group B can be attributed to a major market crash that occurred on the 2nd of April as a direct consequence of international trading tariffs imposed by the US government, causing widespread disruptions in global trade and a significant downturn of global stock markets (Partington, 2025). To illustrate the market conditions in this time frame we can observe a projection of the Dow Jones Industrial Average across the time span in which Group A and B are situated (Figure 3). The Dow Jones Industrial Average or DJIA is a combined measure of 30 prominent US companies, making it a strong indicator of the average temperature of the US stock market over the period which Group A and B are contained. In Figure 3 we can clearly observe the significant market downturn that occurs in the test set of Group B.

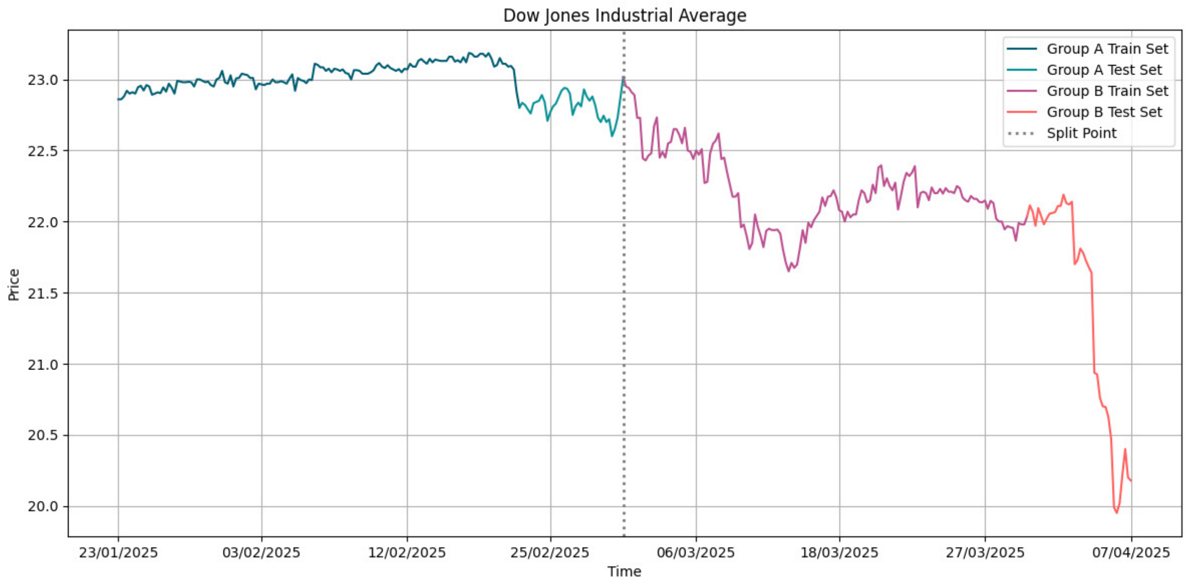

Figure 3: DJIA in timespan of Group A and Group B data

In this investigation a control group is used to allow for a baseline comparison between the accuracy of the models. This is especially important due to the smaller dataset used in the volatility data compared to the general dataset. As such, if we compared the Group B data in volatility to the predictions made on the general dataset, any observations of model differences would be confounded by the differing sizes of the datasets – i.e. any results would be indicative of each model's ability to make predictions with less data rather than their ability to make predictions in times of market volatility. By including Group A as a control group we isolate the market volatility as the dependent variable, allowing us to focus solely on the adaptability of the models in volatile conditions.

The Group A and Group B data should help serve to test the ability of the models to adapt to times of economic downturn and help to highlight which models are more robust in unstable market conditions and offer insights into each model's adaptability when faced with real-world financial instability.

Data Generation:

All stock market data will be generated using yfinance, a python API which extends from the Yahoo! finance stock data site

Data Preparation:

Once the data is generated any missing values need to be accounted for using the pandas bfill() method, which replaces any missing data with the data that is in the spot behind it. The columns will then need to be filtered to extract only the relevant data.

Feature Extraction

For SVR and XGBoost, only the Close price will be needed, as these models are designed to take in 2d data – (samples, time_steps). The LSTM and RNN models will use the following data:

Open: Stock value at the start of the time interval

Close: Stock value at the end of the time interval

High: Highest stock value reached during the time interval

Low: Lowest stock value reached during the time interval

Volume: The quantity of shares bought during time interval

Smoothed Volume – Noise reduced variation of the volume indicator

RSI (Relative Strength Index) – momentum indicator that measures the speed and magnitude of price movements. It is calculated by deriving the mean gains and mean losses for every time step over a given period - typically a time period of 14. These values are then put into the following formula to capture the Relative Strength or RS:

$$RS = \frac{Average\ Gains}{Average\ Losses}$$

The relative strength is then plugged into the formula below to generate the final RSI value:

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

Simple Moving Average (SMA): Shows stock price over a specified period. In the LSTM and RNN models 2 SMAs are used – one over a time period of 10 and one over a time period of 50. The SMA features help smooth out data and assists the models in assessing long term price trends.

$$SMA = \frac{Sum\ of\ n\ close\ prices\ over\ a\ time\ period}{n}$$

Exponential Moving Average (EMA): Shows the stock price over all price data up to a given point, with more weight being given to the most recent stock values, allowing faster reactions to recent price changes. The EMA is calculated over the entirety of the stock data, rather than over a specific time period, meaning significant smoothing is required to ensure the most recent data is given appropriate importance. This is achieved using the smoothing coefficient α, where a larger α means more weight is given to more recent prices, and a smaller α means less weight is given and more long-term trends are captured. The formula for the smoothing coefficient α is given below:

$$\alpha = \frac{2}{n + 1}$$

Where the “span” represents the number of time steps used to calculate the smoothing factor.

$$EMA = \alpha * Price_t + (1 - \alpha) * EMA_{t-1}$$

The RNN and LSTM models use two EMAs: One with a span of 10 and one with a span of 50

Moving Average Convergence Divergence (MACD): Analyses price trends by comparing 2 different EMAs – one derived from a 26 period span and one derived from a 12 period span. The MACD is then calculated by subtracting the 26 period EMA from the 12 period EMA. This calculation then allows for the indicator to capture both long term and short term trends in order to indicate shifts in momentum.

Data Splitting

Once the raw data has been cleaned and the features have been extracted where necessary, the data needs to be put into a format suitable to be trained by the ML models. The data is therefore split into 2 or 3 parts, depending on the type of model. For the regular Machine Learning models SVR and XGBoost, the data is split into a train set and a test set, with an 80:20 ratio, i.e. 80% of the data is allocated to training and 20% of the data is allocated to testing, as is the standard procedure in Machine Learning models (Joseph, 2022),(Gholamy et al, 2018)

The complexity of the LSTM and RNN models mean that there is a high risk of the models overfitting to the training data. Therefore, these models require a validation set, which is used to help monitor the performance of the models and stop the training early when the validation loss starts to stagnate or increase. The data for the models LSTM and RNN are split into a train set, a test set and a validation set with a 60:20:20 ratio – 60% allocated to training data, 20% allocated to test data and 20% allocated to validation data.

For time series models it is crucial that the data is split chronologically, and therefore that random shuffling is not used to ensure there is no data contamination. All data splitting was achieved using the train-test split Scikit-Learn data preparation algorithm.

From here, the data is transformed into a sliding window format. Sliding window is a way of transforming time series data by creating overlapping sub sequences or ‘windows’ of fixed length. These windows each consist of input features and a corresponding target value. The Machine Learning algorithm will then use the input features to predict the corresponding target value. This ensures that the models learn temporal dependencies and relationships in order to make informed predictions of stock prices.

Data Scaling

At this point the data needs to be scaled. This ensures that the relative contributions of each feature are treated equitably regardless of the relative size of their values. Here, the MinMaxScaler() is used from the sci-kit learn preprocessing library. This scaler uses minimum and maximum values to scale the data in the formula below:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

It is crucial that the scaler is fit only on train data, as transforming the train set based on test data would be considered data contamination and produce a false positive outcome.

Model Construction:

XGBoost

The XGBoost model will be constructed using the XGBRegressor from the XGBoost package. The XGBRegressor takes in 5 hyperparameters

booster – specifies type of booster used, i.e. linear or tree-based

n_estimators – number of trees used (ignored by linear booster)

learning_rate – scaler for the contribution of each new tree/step added to the model

reg_alpha: regularization parameter – uses L1 regularization to penalise high weights and prevent overfitting – achieved by effectively removing noisy data that show low relevance to the model's prediction

reg_lambda: regularization parameter – uses L2 regularization to penalise high weights and prevent overfitting – achieved by reducing the magnitude of weights, thereby smoothing out the data and encouraging smaller data values

SVR

The SVR model will be constructed using the SVR class from the scikit-learn library. The SVR class takes in 3 hyperparameters

kernel – similarity function used

epsilon – error tolerance margin, essentially if the error between actual and predicted values in the training sets falls within the margin the error is ignored, if the error falls outside the margin the model is adjusted.

C – determines how tolerant the model is of errors, whereby a higher C value results in a larger penalty being placed on errors outside the tolerance margin.

RNN

The RNN model will be constructed using Keras TensorFlow. Initially the model is instantiated as a base Sequential() model, which is a linear stack of layers in which each layer builds off of the last one.

The next step involves adding layers to the model. Figure 4 shows a diagram of the layers used in the RNN model and their various parameters.

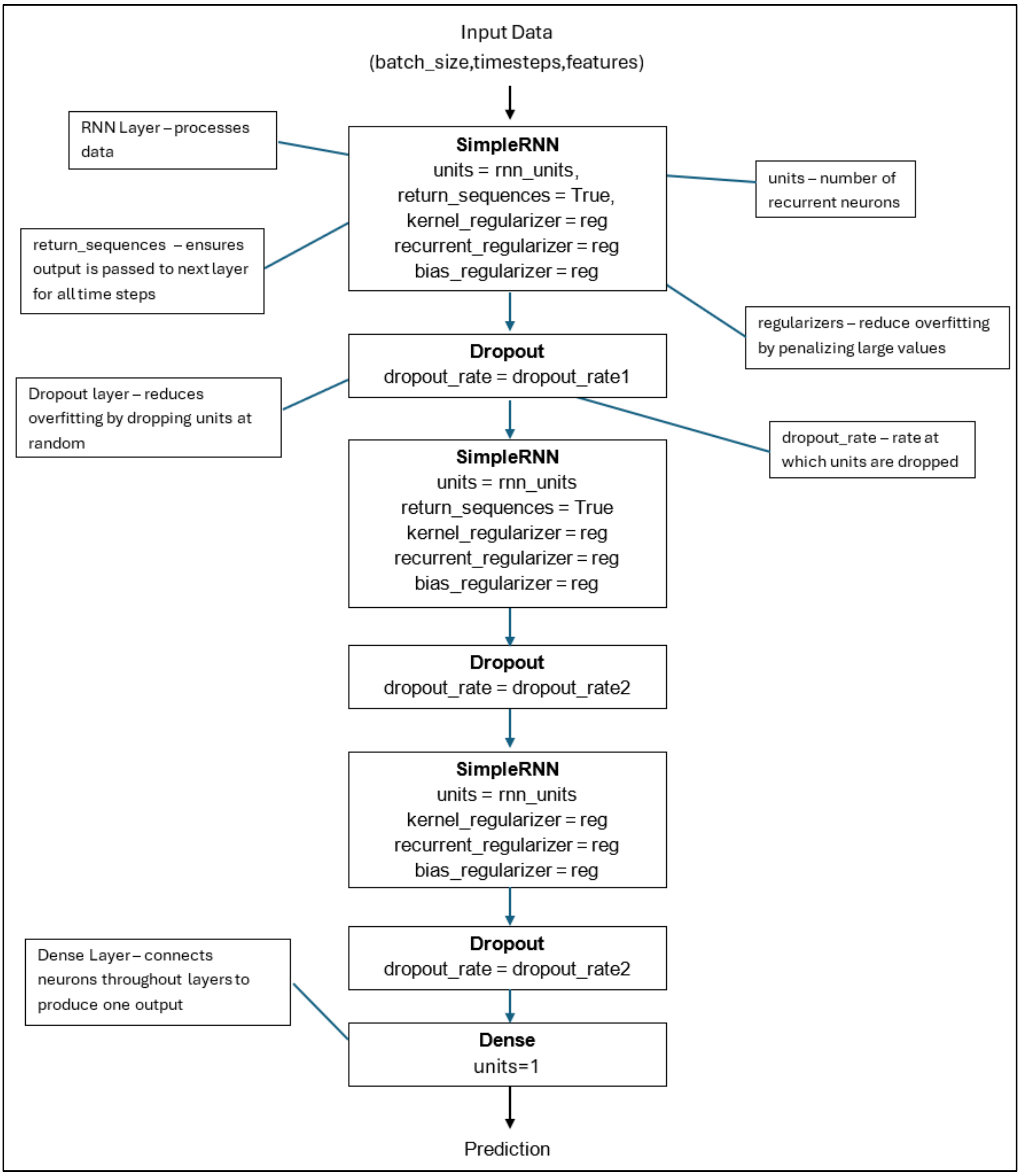

Figure 4: RNN Structural Diagram

In the SimpleRNN layers we can see regularizers are being employed in the RNN model. This is because during the validation phase models were producing jagged predictions, which is a clear sign of model overfitting. More specifically, the model was overfitting to fluctuations in the training data, producing jagged, erratic predictions. Regularizers help to mitigate issue by penalising excessively large weights during the training process, preventing the model from becoming overly sensitive to noise and producing smoother, more stable predictions.

Therefore the RNN model uses L2 regularization to reduce overfitting. L2 regularization is a type of regularization that adds a penalty to the model's loss function based on the square of the weights . The 3 regularizers used in the RNN model are:

- kernel_regularizer: applies a penalty to weight between the input and hidden layer connections
- bias_regularizer: applies a penalty to a layers bias terms, which is a constant that is added to the output of a node
- recurrent_regularizer: applies a penalty to weights that connect the recurrent units in the RNN network

As there are 3 regularizers being used it is important that the regularizer value is kept low to ensure that the data is not oversmoothed, as this would cause the data to become too simple and result in underfitting.

The RNN model uses an epoch value of 1000, where an epoch refers to one complete pass through the training set by the learning algorithm. An Early Stopping function is then employed if 5 consecutive epochs pass without an improvement in validation loss. This ensures that the model runs for as many iterations as it needs to produce the most accurate model possible, while preventing overfitting and unnecessary extension to the model's run time.

LSTM

The LSTM model will also be constructed using Keras TensorFlow. Similarly to the RNN model, the LSTM model is instantiated as a Sequential() model. From here layers are added to model in the structure seen in Figure 5 (diagram below)

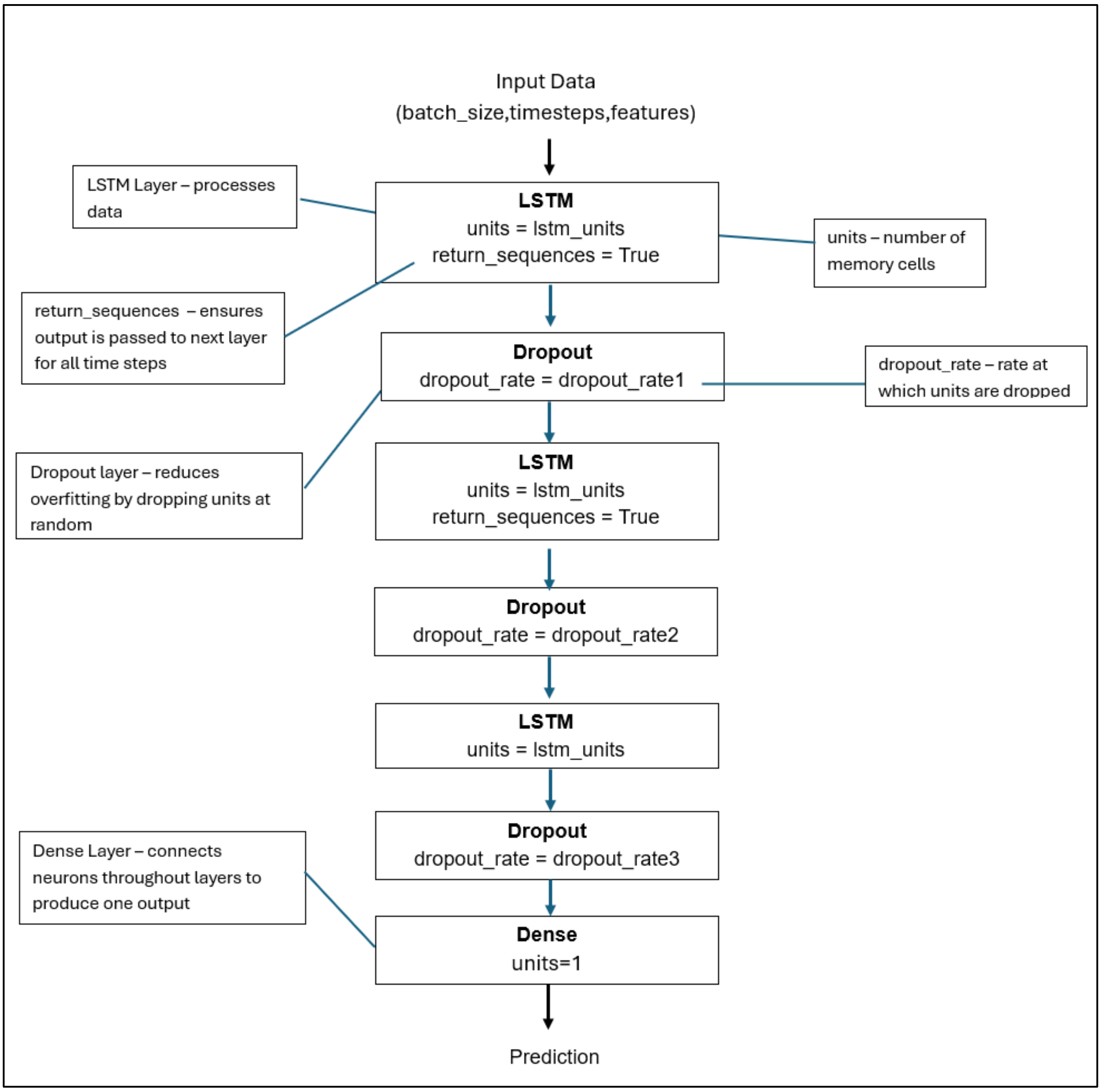

Figure 5: LSTM Structural Diagram

Notably, the LSTM model does not use regularizers like the RNN model does, due to the fact that during the validation phase the LSTM model's predictions had no issues with overfitting. This is most likely because the complex gating mechanism used in LSTM models makes the model less prone to overfitting than the RNN model.

Same as the RNN model, the LSTM model uses 1000 epochs and an early stopping function that halts the learning process after 5 consecutive epochs with no improvement in validation loss.

Hyperparameter Optimisation

Before predictions can be made each model's hyperparameters will need to be tuned in order to find the optimal values. Hyperparameters are the configuration values of a

Machine Learning model that control how the model learns from data. Hyperparameter tuning involves searching for the optimal combination of hyperparameter values that produces the best performance, based on some evaluation metric. Since the tuning process only uses the train and validation sets, optimal hyperparameters can be found before model predictions are made without risking any data contamination to the test set.

For the models SVR and XGBoost hyperparameter optimisation was performed using the GridSearchCV function from the Scikit-Learn library. This function performs a systematic search over a grid of possible hyperparameter values and employs cross validation to find the optimal combination of hyperparameters. The evaluation metric used for evaluating the performance of each model with different hyperparameter combinations was Negative Mean Squared Error (NMSE), which is a variant of MSE with negated values, such that the model aims to maximise the NMSE – i.e. a higher value means better accuracy. The time efficiency of the SVR and XGBoost models means that we will have sufficient time and computational power to tune the SVR and XGBoost to each individual stock from the dataset.

For the hyperparameter optimisation of models RNN and LSTM it is important to consider the time and computational constraints of the project. Due to the complexity and resource-intensive nature of these two models, performing hyperparameter tuning for each individual stock in the dataset will not be feasible. Therefore, the RNN and LSTM will be tuned using one representative stock from each stock category - growth, declining and stable - in order to reduce the computational load whilst still mitigating potential bias that would come with fitting hyperparameters to only one stock and using these values across the whole dataset.

For the RNN and LSTM hyperparameter optimisation process a Random Search algorithm will be used. Random search randomly samples different combinations of hyperparameters and returns the best performing set. Whilst this search function does not guarantee that the global optimum for the hyperparameter values will be found, it will ensure that near-optimal hyperparameters are found for each stock type in a time-efficient manner. Random Search is considered to be a more efficient method of hyperparameter optimisation than Grid Search according to some researchers (Bergstra and Bengio, 2012), especially in scenarios where a model has a high runtime like in the case of the RNN and LSTM models used here.

The Random Search will be performed using the RandomSearch() function from the KerasTuner API, which is a library that simplifies the process of tuning hyperparameters for Keras and Tensorflow models. 30 searches will be performed using different hyperparameter combinations.

Model Evaluation:

Each model was evaluated on the RMSE metric (Root Mean Squared Error). RMSE is used to measure the average difference between the actual stock values and the values predicted by the four algorithms. It shows how accurate a model is at predicting a target value, where an RMSE of 0 indicates a perfect match between actual and predicted values – therefore meaning a lower RMSE indicates a stronger accuracy. RMSE is calculated by the following formula:

$$RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (Predicted_i - Actual_i)^2}$$

In evaluating the accuracy of stock market data the case can be made that Mean Absolute Error (MAE) is the more effective metric. However, one common issue that arose during the validation phase is that some models were producing significant outliers and unexpected spikes/downturns in their predictions. Therefore, it was necessary to use an accuracy metric that is more effective at penalising large errors so that models with significant outliers were appropriately scored lower for accuracy. Therefore, as RMSE is more effective at penalising large errors than MAE (Chai and Draxler 2014), it ended up being the more appropriate metric to use.

Implementation

Programming Environment

All programming was performed using Python version 3.11.2

A number of external python packages were used in the process of completing this project Below is a table of the external packages used during the implementation stage – all packages were installed using pip (v 25.0.1).

Package	**Version**	**Usage**
Alibi	0.9.6	Python XAI library used to visualise relevance of models across time steps
Keras	3.9.2	High level deep learning API used with TensorFlow to build and run the LSTM and RNN models
KerasTuner	1.4.7	Hyperparameter optimisation framework used to tune hyperparameters for LSTM and RNN algorithms
Matplotlib	3.9.2	Python visualisation library used to plot predictions, evaluations and XAI graphs
NumPy	1.26.4	Array library used to store and work with financial data
Pandas	2.2.3	Data analysis library used to store and work with financial data
scikit-learn	1.6.1	Machine Learning library used in the preprocessing of data as well as the building and evaluation of the SVR model
Seaborn	0.13.2	Data visualisation library used for visualising feature importance
SHAP	0.46.0	XAI API used to explain ML algorithms
Tensorflow	2.19.0	Deep learning API used to build and run LSTM and RNN models

XGBoost	2.1.4	Library used to build and run the XGBoost model
yfinance	0.2.58	API used to retrieve stock data from Yahoo! Finance

Hyperparameter Optimisation Values

SVR

Hyperparameter	Array of Values	Chosen Value
epsilon	0.001,0.010.1,0.25,0.5,1,10,100	0.5
C	0.01,0.1,0.2,0.5	0.01

XGBoost

Hyperparameter	Array of Values	Chosen Value
booster	“gblinear”,”gbtree”,”dart”	gblinear
n_estimators	50,100,250,500,1000	1000
learning_rate	1e-3,1e-2,0.1,0.25,0.5	0.25
reg_alpha	1e-5,1e-4,1e-3,1e-2,0.1,1,10	1e-4
reg_lambda	1e-5,1e-4,1e-3,1e-2,0.1,1,10	1e-4

RNN

Hyperparameter	Array of Values	Growth	Declining	Stable
rnn_units	32,64,96,128	64	32	96
regularizer_val	1e-5,1e-4,1e-3,1e-2,1e-1	1e-3	1e-3	1e-4
dropout_rate	0.1,0.2,0.3,0.4,0.5	0.4	0.1	0.3
learning_rate	1e-5,1e-4,1e-3,1e-2,1e-1	1e-3	1e-3	1e-3

LSTM

Hyperparameter	Array of Values	Growth	Declining	Stable
lstm_units	32,64,96,128	128	128	128
dropout_rate	0.1,0.2,0.3,0.4,0.5	0.2	0.4	0.1
learning_rate	1e-5,1e-4,1e-3,1e-2,1e-1	1e-3	1e-3	1e-3

It should be noted that in the implementation of the hyperparameter tuning, the function used to create the model ready for tuning was taken from the KerasTuner webpage on the Tensorflow.org website (TensorFlow 2024)

Results

Each model was executed on each piece of stock data and the RMSE was calculated in each case. The RMSE values within each category were then averaged out using the mean to identify each model's ability to make predictions based on each stock type. The combined accuracy was then derived from these values to get the combined accuracy value for each model. These values can be seen in the table below:

Model	Growth Stock avg. RMSE	Declining Stock avg. RMSE	Stable Stock avg. RMSE	Combined avg. RMSE
SVR	8.07	0.379	0.727	3.059
XGBoost	6.472	0.475	2.464	2.604
RNN	17.792	0.521	2.503	6.939
LSTM	16.55	0.917	1.237	6.236

The combined accuracy values for each model were then plot into a bar chart to visualise the comparison between each model's general stock accuracy (Figure 6)

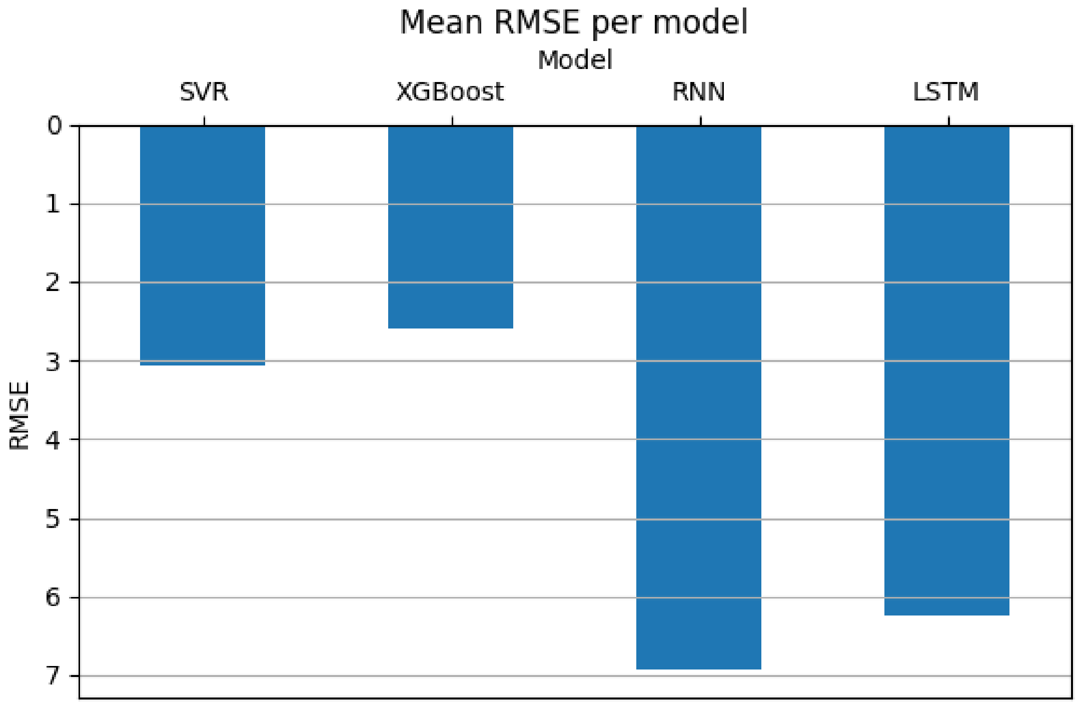

Figure 6: RMSE Bar Chart per Model

The accuracy of each model on each stock category is visualised in figure 7 below

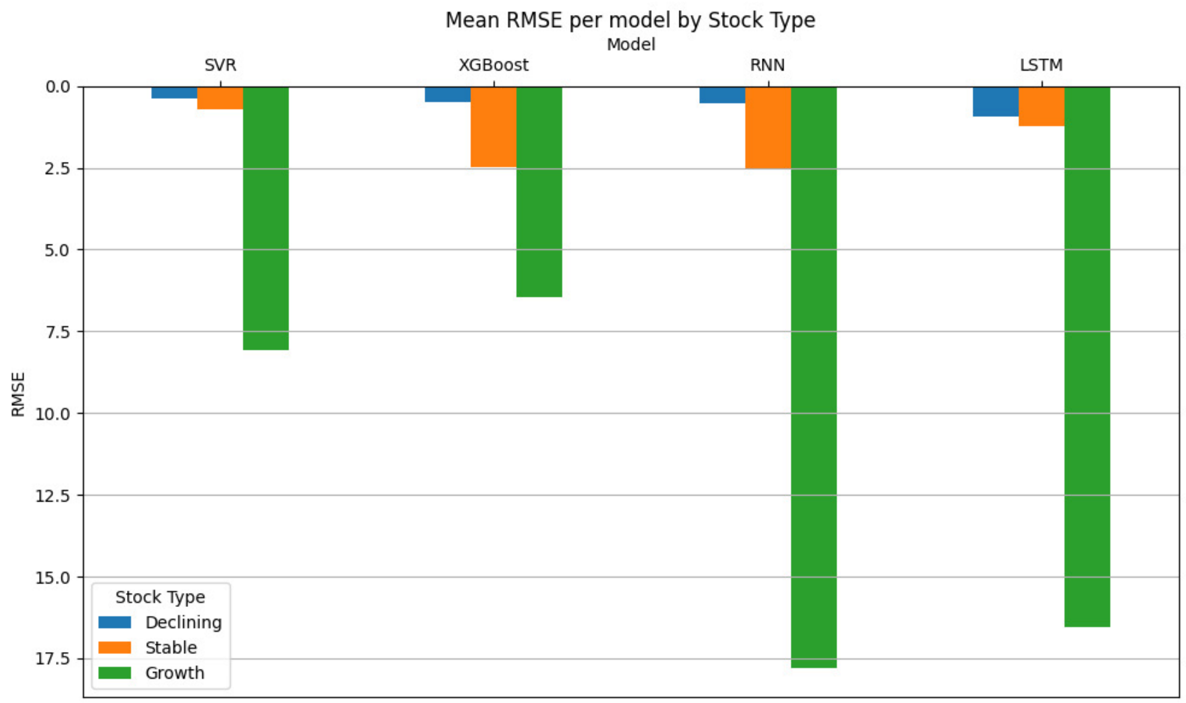

Figure 7: RMSE Bar Chart per Model by Stock Type

To show the ability of each model to predict a stock type relative to its combined accuracy, the contextual accuracy is calculated using the following formula:

Avg. RMSE for Stock Type / Combined Accuracy

Here, a contextual accuracy above 1 means that the model is above average at predicting that stock type and a contextual accuracy below 1 means that model is below average at predicting that stock type, with 1 meaning exactly average. This allows us to compare model performance of stock type independent from the overall performance of each model's implementation

Model	Growth stock contextual accuracy	Declining stock contextual accuracy	Stable stock contextual accuracy
SVR	2.638	0.124	0.238
XGBoost	2.485	0.182	0.946
RNN	2.564	0.075	0.361
LSTM	2.654	0.147	0.198

To compare the models on their ability to make predictions in varying market conditions each model was ran on both sets of volatility data – the control set Group A and the volatility data Group B. From these predictions we gathered the mean RMSE for each group across all stock predictions. The relative change is an indicator of how much the volatile market data affected each model's ability to make predictions compared to the control group, where a larger relative change indicates the and is calculated by the formula $\frac{Group\ A\ avg.\ RMSE}{Group\ B\ avg.\ RMSE}$. The volatility data can be seen in the table below and is visualised in a bar chart in figure 8

Model	Group A avg. RMSE	Group B avg. RMSE	Relative Change
SVR	5.521	11.06	2.00
XGBoost	4.642	7.086	1.53
RNN	6.386	11.699	1.83
LSTM	5.11	12.86	2.52

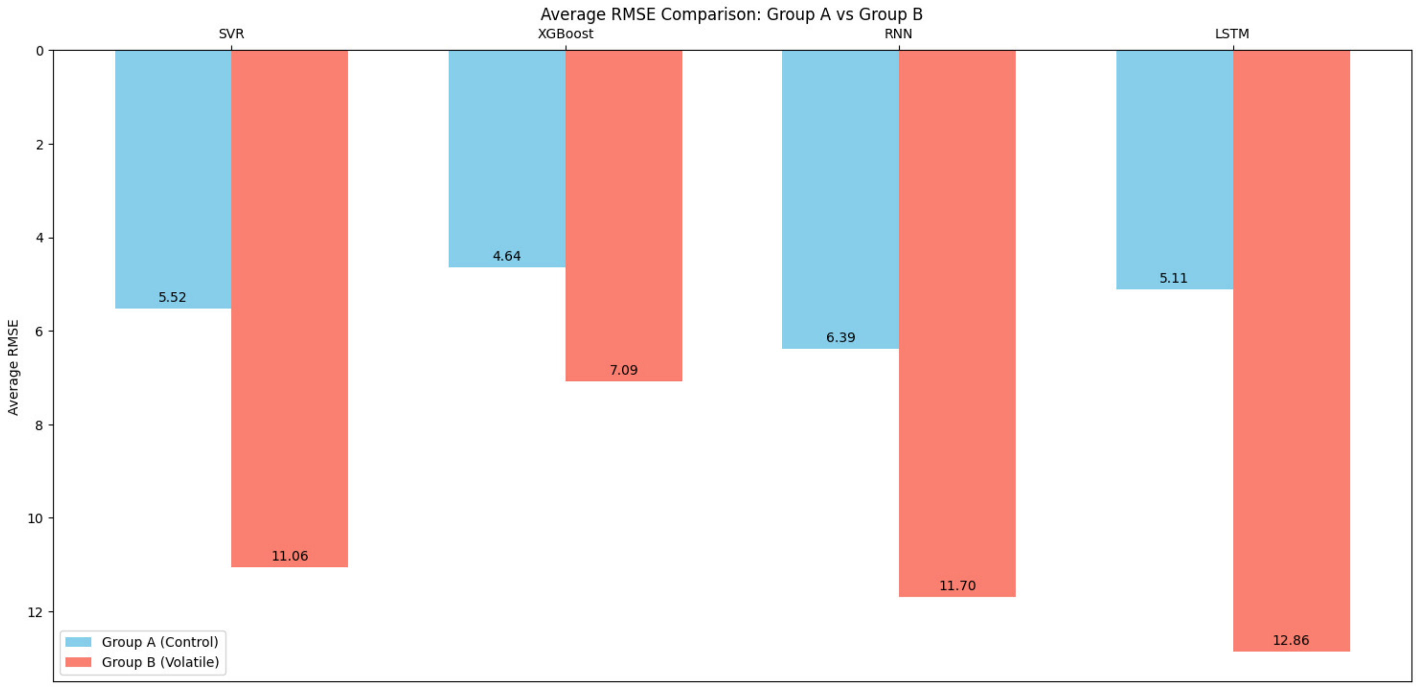

Figure 8: Bar Chart of Average RMSE for Group A vs Group B

Insights

From Figure 6 we can clearly see that the SVR and XGBoost models outperformed the RNN and LSTM models in terms of combined accuracy, most likely due to the smaller size of the data set, which lends itself to less complex algorithms. Additionally, Figure 7 shows us that across stock types XGBoost provided the most consistently accurate performance, despite being beaten in RMSE value by the SVR algorithm in the Stable stock category. Figure 7 shows us that the performance for the growth stocks was poor across all the models, most likely due to difficulties predicting stocks that show significant variance and upward trends.

Figure 6 suggests that the LSTM and RNN models have poor predictive power, due to their low RMSE values compared to the SVR and XGBoost models. However, the data in Figure 6 does not show the full picture, in that the LSTM and RNN models struggled in particular with predicting the Growth stocks, skewing their mean combined RMSE values, but stayed fairly competitive with the SVR and XGBoost models in the Stable and Declining stock categories. This highlights the need for stock category separation and context in comparative analysis of stock market prediction, as an analysis using purely the aggregated RMSE scores would identify the LSTM and RNN algorithms as poor algorithms, when in fact they are just better suited for different stock types and contexts.

The contextual accuracy highlights how well each model performed on stock types relative to its combined RMSE, allowing us to identify the algorithms that are better suited to predicting particular categories of stocks independent of their overall performance and implementation. From the contextual accuracy scores we can see that relative to their overall performance, XGBoost was the most effective algorithm for the prediction of growth stocks, RNN was the most effective model for the prediction of declining stocks and LSTM was the most effective model for the prediction of stable stocks. As many studies observe RNN and LSTM to be the most accurate Machine Learning methods in stock market prediction (Shobayo et al, 2023), (Lakshminarayanan and McCrae 2019), (Nabipour et al, 2019), the contextual accuracy suggests that with the right implementation RNN would be the most effective model for making predictions on declining stocks and LSTM would be the most effective model for making predictions on stable stocks.

These results for the volatility data in Figure 8 (volatility data) suggest that XGBoost is the most effective model for making stock predictions in volatile market conditions, indicated by XGBoost producing both the lowest average RMSE for the volatile dataset and the lowest relative change in accuracy. This is consistent with the per stock type accuracy shown in Figure 7 which determined XGBoost to be the most accurate model in predicting the Growth and Declining stock categories, which are the 2 most volatile stock types.

The volatility data in Figure 8 also shows us that the RNN model produced a lower relative change between Group A and Group B than SVR. This is most likely due to the simplicity of the SVR model, and the fact that it only uses Close prices to make predictions while the RNN model can make use of lag features, moving averages and volume to better anticipate sudden market downturn. The lower relative change seen in the RNN model suggests it is more robust and flexible than the SVR model. Therefore, if improvements are made to the RNN model to improve its general accuracy (See Future Improvements), the RNN model could be a better model to use for stock market forecasting than SVR in a real-life trading scenario, as the ability to anticipate and adapt to market shifts is crucial to real-world stock prediction where volatility and downturns are commonplace. This once again highlights the importance of using context when comparing models in stock market prediction.

Overall, the data suggests that XGBoost is the most robust and flexible model, especially in periods of market volatility.

Evaluation

It is important to note that the predictions made during testing are not reflective of real-time live forecasting. Each time step uses historical data as an input, meaning it does not necessarily rely on just its past predictions. These tests do however provide a fair estimate of each model's ability to learn from past data to make predictions, and of each model's next-day predictive power.

Many studies comparing Machine Learning models in stock market prediction have deep learning models such as RNN and LSTM producing higher predictive accuracy than the traditional Machine Learning methods such as SVR and XGBoost (Shobayo et al, 2023), (Lakshminarayanan and McCrae, 2019), (Nabipour et al, 2019). The fact that the deep learning models in this project have poorer accuracy than the traditional Machine Learning models suggests that there are potential deficiencies in the deep learning models that not been accounted for. It should be noted however that the abnormal results produced in this project compared to the relevant literature could simply be down to a difference in the data used. That being said, we can use XAI techniques to identify contribution and relevance and find exactly where any potential model deficiencies occur.

First we can look at plots of each model's predictions compared to the actual stock value. The Tesla stock was selected for analysis, as each model produced relatively poor predictions when applied to the Tesla stock data when compared to their combined accuracy. These projections can be seen in Figure 9

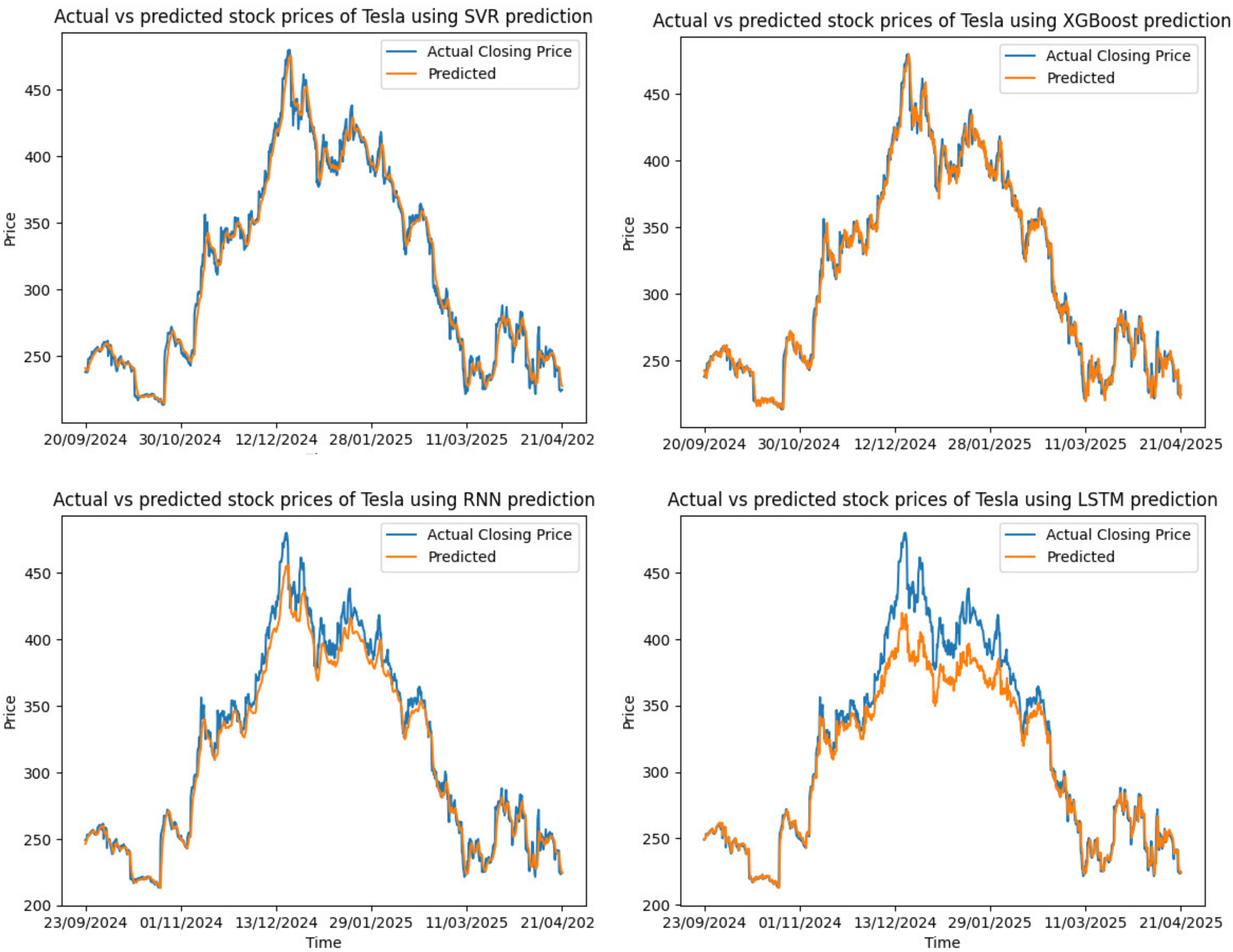

Figure 9: Tesla Price Prediction Graphs

From here we can see that the RNN and LSTM models are somewhat underpredicting the stock prices, especially in the time periods where the price values are high. Using XAI techniques we can investigate these models and see why this is occurring.

First, we can look at the average relevancy and prediction contribution of each time step, in the LSTM model's prediction of Tesla data. To generate this data, we use the IntegratedGradients explainer from the Python API alibi, which helps identify how much each part of the input influenced the model's decision. This can be seen in Figure 10 below:

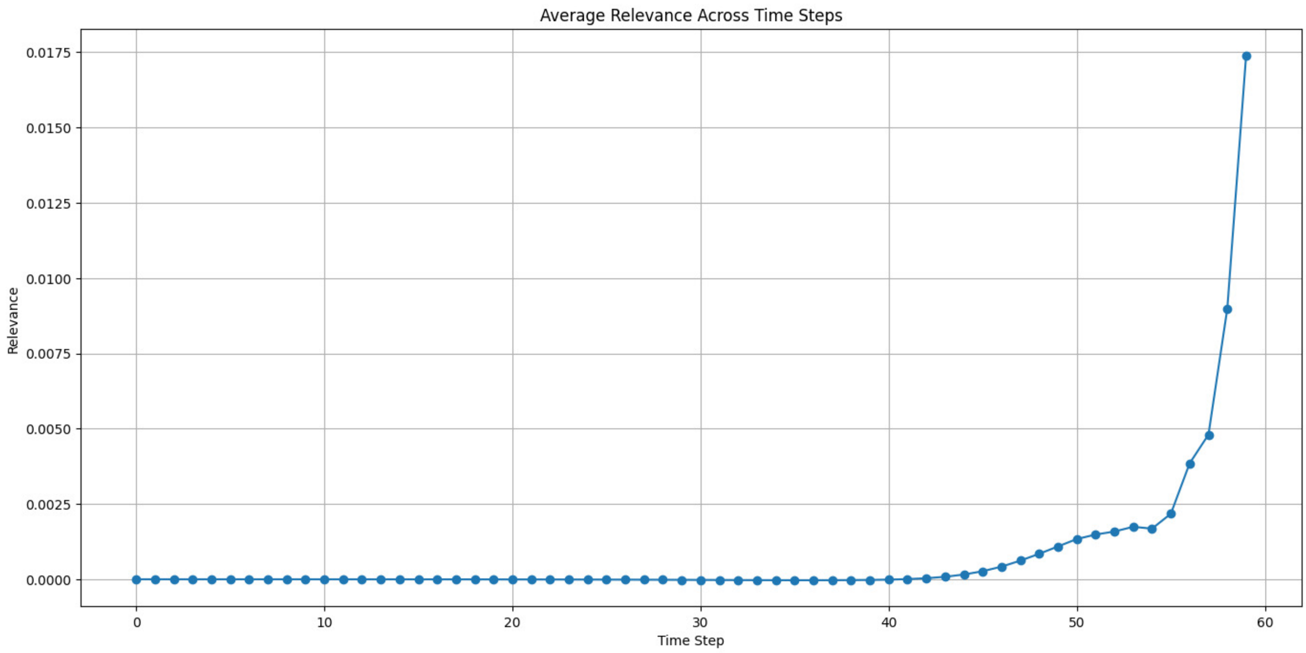

Figure 10: LSTM Average Relevance Across Time Steps for Tesla Data

We observe that the first 42 time steps contribute practically nothing to the model's prediction, with relevance values remaining close to zero until around step 35. This suggests that the use of a 60-day input window is unnecessarily large and could likely be reduced to around 20–25 time steps without any significant loss in predictive performance. The volume of unimportant time steps could be a reason for the poor prediction of the LSTM model in Figure 9, whereby the noise of irrelevant time steps dilutes the actual important information resulting in the blunted, conservative low predictions.

It is however also important to note that it's possible the increased relevance after time step 42 does not necessarily imply that earlier information is irrelevant and negligible, but could rather suggest that memory decay has occurred, which causes earlier inputs to be forgotten over time. In fact, some researchers have proposed that LSTM has no long-term memory at all in a time series scenario (Zhao et al, 2020), which would explain why the model only retains the relevant features from the last few time steps. If this is the case, there are mechanisms that could be put in place to overcome this deficiency (see Future Improvements).

Looking at feature contribution is also an important way of diagnosing underfitting. Here, a random subset of the training data is used to generate correlation scores between features, using the python API SHAP. These scores were then plotted into the heatmap below to provide insights into how the features contribute to the model's predictions (Figure 11)

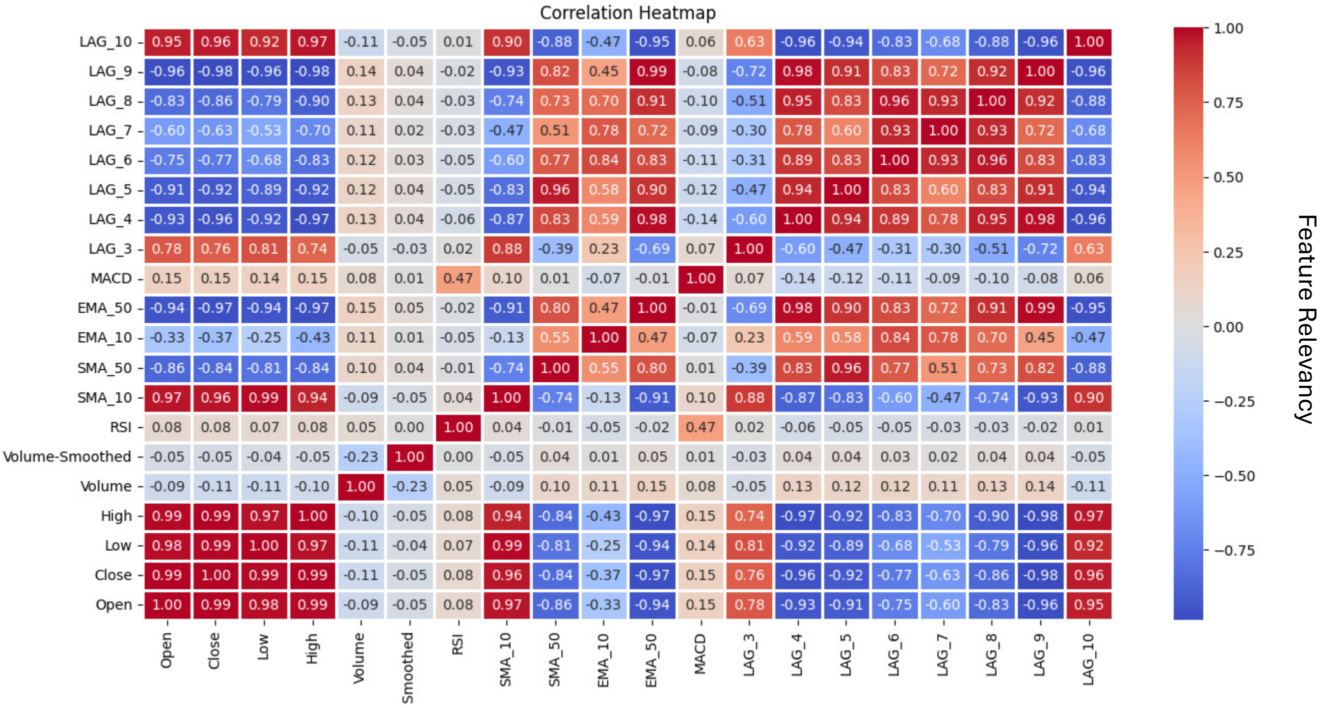

Figure 11: LSTM Feature Relevancy Heatmap

The important rows to focus on are the bottom 4, which show the correlation between the all the features and the stock prices - High, Low, Close and Open. A positive correlation between price and a feature indicates that as the value of the feature increases the price should increase, meaning that the feature in question contributes to the predictive power of the model. This is indicated by the boxes that are darker red in colour. A negative correlation between price and a feature suggests that as the feature decreases, the price of the stock increases, which also contributes to the predictive power of the model. These values can be seen in the darker blue colours.

We can see from this heatmap some possible sources of the underfitting seen in our LSTM prediction. The features Volume, Smoothed Volume, RSI and MACD all have neutral colours and near 0 correlation to the price values. This suggests that they have very low or even negligible contribution to the predictions made by the model. This means that these features only cause the input data to be noisier, which could be potentially resulting in the model struggling to identify meaningful patterns in the data, as the true signal is buried under the random fluctuations of these irrelevant features. This results in the broader trend we can see in the LSTM prediction in Figure 9.

The features RSI and MACD specifically having low contribution could also be an important indicator of why the model is making inaccurate predictions. This is because there is research supporting the idea that RSI and MACD should be effective indicators in stock market Machine Learning models (Vimalsubramanian et al, 2024),(Aguirre et al, 2020). Therefore, the lack of relevance contribution of these features suggests that the model architecture is not effectively incorporating them, potentially due to imbalanced scaling or poor feature modelling.

Another good way of investigating the accuracy of model prediction is using residual graphs. The residuals are the differences between actual and predicted values, and a residual graph can provide insight into the distribution of errors. Figure 12 shows the residuals of each model's prediction of the Tesla stock data. This plot is in the form of a histogram, whereby a desirable graph should be normally distributed in a bell curve shape with an equal number of values distributed either side of the residual value 0

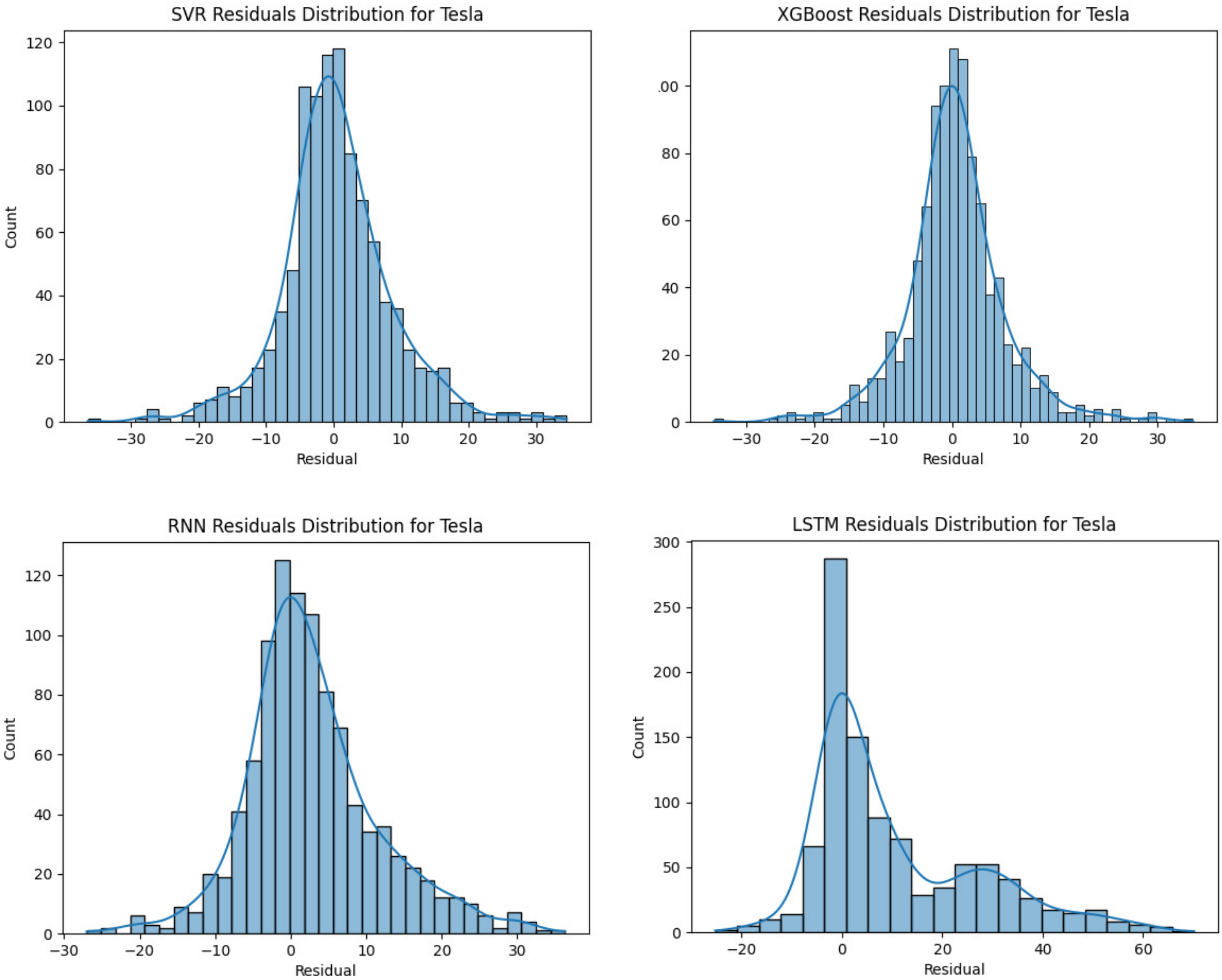

Figure 12: Residual Distribution of ML Model Predictions of Tesla Data

Figure 12 shows us that the residual distribution for XGBoost, SVR and RNN are close to normally distributed, with a slight lean towards the right of the zero. This means that there are an above average number of instances of a stock price having a higher value than the model predicted, i.e. the prediction is underestimating. This is most likely due to the significant price rise in the test data compared to the training data in the Tesla stock, leaving the model unprepared to predict the higher price values, which can be seen in the prediction plots in Figure 9.

The LSTM residual graph in Figure 12 further illustrates the issues the LSTM model has with underestimating values. We can see a significant volume of underestimated values to the right of the zero compared to values to the left of the zero. This further indicates that the LSTM model is suffering from overfitting, whereby the model has become too specialised in capturing patterns in the training data and is therefore unable to generalize to the rapid rise in price of the Tesla data. This suggests that the LSTM model is particularly poor for stocks that are experiencing rapid growth and would therefore be in a real-world scenario would be unable to capitalize on a stock's rapid growth for profit.

All this being said, regardless of some of the deficiencies in the LSTM and RNN models, the use of contextual accuracies and relative change allowed us to look past any issues with the implementation of the models and really get into each model's performance across different stock types and market conditions based on their different structures and behaviours.

Conclusions

This project aimed to compare Machine Learning models across market conditions and stock types to assist researchers and practitioners in selecting the optimal model to make stock predictions based on their specific needs. To achieve this, four models were chosen and implemented, and their predictions were analysed across stock types and different levels of market volatility. The results showed XGBoost to be the most adaptable and robust algorithm, as is evident from the model's predictive power in times of market volatility compared to the other models, as well as its ability to predict declining and growth stocks compared to the other models. For the stable stocks however the SVR and LSTM models proved more effective, suggesting that while the XGBoost model is the superior model for stock prediction in general, there are scenarios and situations in which the SVR and LSTM models are more appropriate.

While this project aimed to provide a resource that contributed to improved model selection for researchers and practitioners, the evaluation showed that some of the models had significant deficiencies that affected their predictive accuracy. Therefore, the results produced by this project should be taken with caution, as they are contingent on the models being further improved and optimized to their maximum predictive potential. It should be mentioned however that where applicable relative accuracy values were used alongside the actual accuracy values to show model favour and bias independent of any implementation issues.

Overall, this project did manage to achieve some of its goals in evaluating the performance of different machine models under varying market conditions and across stock categories. While the data produced from this project has some limitations, the findings contribute to a better understanding of model selection based on different market conditions and stock profiles.

Additionally, this project did succeed in demonstrating that there is a notable variance in the accuracy of model predictions depending on the stock type that the prediction is ran on and the market conditions at the time of prediction. These findings serve to highlight the importance of incorporating contextual factors into comparative analyses of Machine Learning models and should help to guide future researchers towards employing more nuanced approaches to evaluating their Machine Learning models that better reflect the complexity of real-world stock market prediction.

Future Work

The main limiting factor in the implementation of this project was the computing power used for running code. Ideally in future iterations of the project the code would be ran on a paid-tier cloud computing service like AWS or on a computer with a GPU. One area this limitation affected in particular was the hyperparameter optimisation process for the deep learning models RNN and LSTM. The hyperparameter tuning process for the deep learning algorithms was lengthy due to the processing power required of the models and the time constraints the project fell under. With more processing power, we would be able to tune more parameters – batch size, sliding window size, amount of LSTM layers etc. Additionally, the units of each RNN or LSTM layer could be tuned individually in each layer, and for the RNN model each regularizer function could be tuned to its own hyperparameter value. The allocation of more time and computational power could also allow for an exhaustive search to be performed over the RNN and LSTM hyperparameters more akin to the SVR and XGBoost searches. Finally, in future work the RNN and LSTM models could be tuned to each individual stock rather than one representative sample in each stock category. All these improvements could result in the predictive power of the RNN and LSTM models becoming more in line with the SVR and XGBoost models, allowing for a fairer investigation of model performance across stock types and market conditions.

One limiting factor of the results in this project is that due to time constraints with the deep learning algorithms, only 9 companies were incorporated into the analysis. Future iterations could use a larger range of stocks, providing a more representative array of data. In addition, future work should use a more scientific method for stock selection. In this project the 9 stocks were selected somewhat arbitrarily by searching for companies that have experienced economic growth/downturn or are in industries that experience low market volatility. Future work could use more scientific methods, for example by searching for Growth stocks using a stock screening application to find the 10 stocks that have seen the highest price growth in the last 3 years.

Future work could also investigate more stock categories and market conditions, in order to capture a broader range of market dynamics and stock profiles, therefore producing a richer analysis. For example, future work could incorporate stocks separated by sector (technology, healthcare, energy etc.) and investigate different market conditions other than just period of volatility (economic booms, periods of market recovery, bull vs bear markets etc.)

This project focuses on exploring Machine Learning using technical indicators, but there are more types of analysis that could be investigated. The main other type of indicator used by economists is the fundamental indicator, which is a long-term indicator that assesses the health, performance and value of a company. In fact, studies show that a combination of technical and fundamental indicators produces the most accurate stock market predictions – (Bettman et al, 2009). Originally this project planned to investigate both fundamental and technical indicators in predicting the stock market, but the fundamental indicators had to be cut out due to time constraints.

Another type of analysis that could be investigated in future work is sentimental analysis. In a stock market context sentimental analysis involves assessing the mood of investors and stakeholders towards a particular company or the market that a company occupies in order to provide an insight into the momentum of a particular stock. This is achieved by analysing social media data, financial news sites and other relevant content using techniques such as Natural Language Processing in order to understand the collective sentiment of the market and investors. This would provide another dimension to the Machine Learning models and would make the models particularly robust at identifying and adjusting predictions in times of market volatility and economic downturn.

In future work, long-term studies could be conducted that test each model's ability to make predictions on live stock market data. This would ensure that the models are generalizable to real life stock scenarios, improving the validity of the research.

In future work involving LSTM and RNN algorithms there are some solutions that could be implemented in order to address some of the deficiencies associated with these models that are illustrated in the evaluation. For example, attention mechanisms could be added to improve long-term memory retention, and the LSTM model could use a bidirectional approach to better capture long-term dependencies and prevent underfitting.

Reflection

In future projects I will be more realistic about the scope of the project. Initially, I planned on integrating fundamental and technical features together, but it quickly became clear that this was too ambitious, so I decided to focus on just technical features. I also dedicated time to developing a Random Forest algorithm which had to be scrapped due to time constraints halfway through the project, which ended up wasting time.

This project has developed my ability to learn new topics at a fast pace. Prior to taking this project I had limited Machine Learning knowledge and limited knowledge into economics and the stock market. However, I took this project as a challenge to myself to learn new topics that I had an interest in and in the end I managed to develop not only my knowledge of these topics but also my ability to push myself and learn new concepts and skills

This project has greatly expanded my knowledge of Machine Learning. Before undertaking this project, I had minimal Machine Learning knowledge, but now I have a breadth of knowledge into the theory of Machine Learning, how different Machine Learning models work and the practical skills to implement Machine Learning algorithms/models to solve a problem.

Overall, I'm happy I took on this project, despite the challenges it presented. It has allowed me to challenge myself and push my knowledge to new limits and has opened up a whole new exciting area of Computer Science to me that I can appreciate and explore.

Figures

References

Aguirre, A. Medina, R. Duque, N. 2020. Machine Learning applied in the stock market through the Moving Average Convergence Divergence (MACD) indicator. *Investment Management and Financial Innovations*. 17(40, pp. 44-60. doi: 10.21511/imfi.17(4).2020.05

Barua, M. Kumar, T. Raj, K. Roy, A. 2024. Comparative Analysis of Deep Learning Models for Stock Price Prediction in the Indian Market. *FinTech*. 3(4), pp. 551-568. doi: 10.3390/fintech3040029

Bergstra, J. Bengio, Y. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*. 13(10), pp. 281-305Chai, T and Draxler, R. 2014. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*. 7(3), pp. 1247–1250. doi: 10.5194/gmd-7-1247-2014

Camska, D. Klecka, J. 2020. Comparison of Prediction Models Applied in Economic Recession and Expansion. *Journal of Risk and Financial Management*. 13(3), 52. doi: 10.3390/jrfm13030052

Chowdhury, S. Nabi, N. Rana, N. Shaima, M. Esa, H. Mitra, A. Mozumder, A. Liza, I. Sweet, M. Naznin, R. 2024. *Journal of Business and Management Studies*. 6(2) pp. 95-99. doi: 10.32996/jbms.2024.6.2.9

Drucker, H. Burges, C. Kaufman, L. Smola, A. Vapnik, V. 1997. Support Vector Regression Machines. *Advances in Neural Information Processing Systems.* 28(7), pp. 779-784.

Friedman, J. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*. 29(5). doi: 10.1214/aos/1013203451

Gholamy, A. Kreinovich, V. Kosheleva, O. 2018. Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation.

Hochreiter, S and Schmidhuber J. 1997. Long Short-Term Memory. *Neural Computation*. 9(8), pp. 1735-1780. doi: 10.1162/neco.1997.9.8.1735

Joseph, V. 2022. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: An ASA Data Science Journal*. 15(4), pp. 531-538. doi: 10.1002/sam.11583

Kumar, I. Dogra, K. Utreja, C. Yadav, P. 2018. A Comparative Study of Supervised Machine Learning Algorithms for Stock Market Trend Prediction. *2018 Second International Conference on Inventive Communication and Computational Technologies*. pp. 1003-1007. doi: 10.1109/ICICCT.2018.8473214

Lakshminarayanan, S and McCrae, J. 2019. A Comparative Study of SVM and LSTM Deep Learning Algorithms for Stock Market Prediction. *Irish Conference on Artificial Intelligence and Cognitive Science*.

Mintarya, L. Halim, J. Angie, C. Achmad, S. Kurniawan, A. 2023. *Procedia Computer Science*. 126(1), pp.96-102. doi 10.1016/j.procs.2022.12.115

Nabipour, M. Nayyeri, P. Jabani, H. Band, S. Mosavi, A. 2020. Predicting stock market trends using Machine Learning and deep learning algorithms via continuous binary data; a comparative analysis on the Tehran stock exchange. *IEEE Access*. 1(1), pp. 99. doi: 10.1109/ACCESS.2020.3015966

Olah, C. 2015. Understanding LSTM Networks. *colah's blog* 27 August. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed: 28 April 2025].

Partington, R. 2025. Global markets in turmoil as Trump tariffs wipe $2.5tn off Wall Street. *The Guardian* 4 April. Available at: https://www.theguardian.com/business/2025/apr/03/global-markets-turmoil-trump-tariffs-wall-street-downturn [Accessed: 11 May 2025].

Rodríguez-Pérez, R. Bajorath, J. 2022. Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery. *Journal of Computer-Aided Molecular Design (2022)*. 36, pp. 355-362. doi: 10.1007/s10822-022-00442-9.

Shobayo, O. Adeyemi-Longe, S. Popoola, O. Okoyeigbo, O. 2023. A Comparative Analysis of Machine Learning and Deep Learning Techniques for Accurate Market Price Forecasting. *Analytics*. 4(1):5.

TensorFlow. 2024. *Introduction to the Keras Tuner*. Available at: https://www.tensorflow.org/tutorials/keras/keras_tuner [Accessed: 18 April 2025].

Vimalsubramanian, G. Alagusundar, N. Prasad, H. 2024. Stock Market Prediction through Ensemble of Deep Learning and Machine Learning Models along with Technical Indicators. *2023 Global Conference on Information Technologies and Communications (GCITC).* Bengaluru, India. 15-16 March 2024. IEEE.

Zhao, J. Huang, F. Lv, J. Duan, Y. Qin, Z. Li, G. Tian, G. 2020. Do RNN and LSTM have Long Memory?. *Proceedings of the 47th International Conference on Machine Learning*. doi: 10.48550/arXiv.2006.03860