Introduction

Statistical models on the predictability of the stock markets performance have been used for a long period of time in economics (von Szeliski, 1935). While opinions differ on the efficiency of market predictions, many empirical studies show that financial markets are to some extent predictable (Bollerslev, Marrone, Xu, & Zhou, 2014). Among different methods for stock price prediction such as regression, statistical methods using the analysis of market data through machine learning have been widely applied (Chong, Han, & Park, 2017). These approaches employ a variety of methods like linear regression in random forests and neural networks to predict stock price values.
Neural networks and artificial intelligence have recently attracted a lot of attention in the press which may be due in large part to its success in image classification, natural language processing, or solving time-series problems (Lee, Pham, Largman & Ng, 2009). The combination of increased computational ability and availability of large datasets have caused a rapid increase in the success of multi-layer neural networks (Schmidt et al. 2019). Where there has been growing interest in whether these methods can be effectively applied to problems in finance. With the less than satisfactory performance of existing models, multiple institutions put efforts into research that objectively examine the possibility of using deep learning methods to predict stock market performance (Sarker 2021).

The purpose of this project is to extract historical stock price data using the yfinance API and tweet data scraped from Twitter to be analysed for sentiment. This will be used to investigate the predictability of a stock's performance using supervised learning methods including; Long Short-Term Memory (LSTM), a form of Recurrent Neural Network (RNN); Random Forest Regression (RF), a bagging technique of multiple Decision Tree; and a Bagging Model, a form of ensemble learning which combines the LSTM and RF models. Each model will be used to predict the following day's close price using the previous 5, 10, and 20 days of open, high, low, and close values, using data over an interval of two years. The performance of the models will be scrutinised and compared using statistical scores measuring and comparing the error of each model. We have also chosen to test each model's ability to predict high volatility stocks against low volatility stocks, we hypothesise that the model will perform worse when a stock's price is erratic (high volatility).

Literature Review

There are numerous techniques for forecasting stock prices. One can use statistical or artificial intelligence methodologies. According to the existing literature, deep learning approaches, particularly LSTM, are more accurate at predicting stock prices than other more

standard machine learning techniques (Sen, 2022). This is primarily owing to the time-series nature of the data, in which prior values affect the current prediction. This study examines the use of LSTM, RF, bagging and sentiment analysis.

LSTM

Long Short-term memories (LSTM) are a type of recurrent network that uses the previous phase's output as an input for the next step. Like other neural networks, the node calculates inputs and outputs. This output value is subsequently used as an input for the next phase, and so on (Brownlee, 2021). An LSTM's recurrent nodes have an internal state. The internal state serves as the node's working memory, storing and retrieving data over time (colah's blog, 2015). Calculations use input, past output, and internal state.

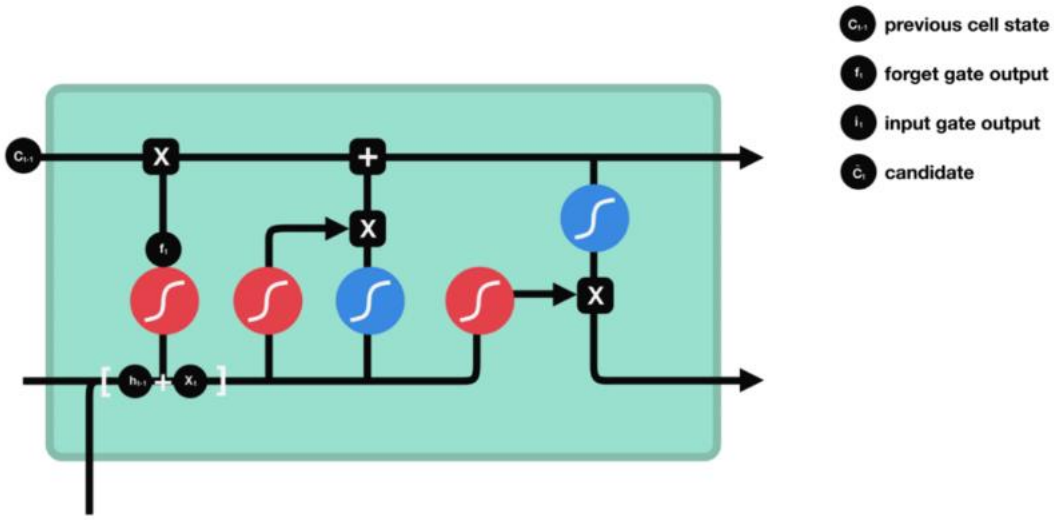

Figure: LSTM diagram (Phi, 2018)

Several authors have stated that the performance of LSTM models is significantly superior in predicting stock prices than other machine learning models (Mehtab, Sen, and Dutta, 2021). This is mostly because deep learning models are better at extracting and learning the characteristics of time series data than their machine learning equivalents. Furthermore, because univariate models are more precise and faster, multivariate analysis is incompatible with LSTM-based regression. Sen (2018) conducted another study in which two Convolutional Neural Network (CNN) models and three LSTM models were used. While all models forecasted with high accuracy, the univariate CNN, rather than the LSTM, was determined to be the most accurate and fastest in its implementation.

Zhigang and Yang (2020) propose using an LSTM model with a sentiment index to account for investors' emotional tendencies in stock market prediction. The results demonstrate that the LSTM model improves prediction accuracy while significantly decreasing time delay. Additionally, it beats the Empirical Model Decomposition (EDM)-enhanced LSTM model, which decomposes the complex stock pricing sequence into simpler, more predictive sequences. Overall LSTM does have some limitations such as the time lag of the prediction (Wei, 2019) however, it has the best accuracy out of all the other algorithms.

Random Forest

A random forest is a supervised machine learning technique based on decision tree algorithms. It makes use of ensemble learning, a technique for resolving complex issues through the employment of several classifiers (Brownlee, 2020). A random forest algorithm is made up of several decision trees. The algorithm creates a 'forest' that is trained using either bagging or bootstrap aggregation. The method generates the output based on the prediction made by the decision trees. Forecasting is accomplished by averaging or summing the output of several trees. Increasing the number of trees increases the accuracy of the outcome.

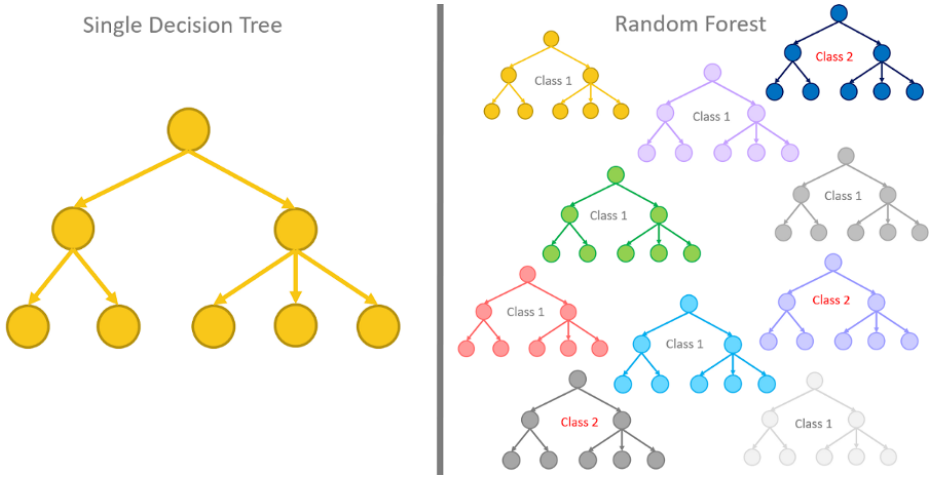

Figure: The random forest technique employs numerous decision trees, each of which has been trained slightly differently; each of them is considered for the final classification. (Rosaria Silipo, 2019)

The disadvantages of the decision tree algorithm are overcome by the random forest method. It reduces overfitting and increases the precision of datasets (Silipo, 2019).

There is a substantial body of literature on the use of random forests for stock price prediction. Some authors strongly advocate for the use of random forests (Khaidem, Saha and Dey, 2022). Ballings and Van den Poel Nathalie concur, stating that random forest outperforms other classifiers in their investigation, including Support Vector Machines, neural networks, and logistic regression. Additionally, it discovered that three of the top four algorithms were ensembles, emphasising the importance of ensembles in the domain of stock price prediction. This contrasts with Kumar and Thenmozhi's (2006) study, which found that SVM outperformed random forest but significantly exceeded other classic methods such as neural networks. When LSTM is compared to the random forest, research indicates that LSTM is the best model for forecasting stock market prices, however, both tree-based and deep learning algorithms demonstrated remarkable ability to forecast future stock exchange values (Nabipour, M 2020).

Bagging Model

Bagging is an ensemble learning approach for improving the accuracy and performance of machine learning systems (Biswal 2021). It is suitable for regression and classification models but excels in decision tree approaches. To perform bagging, a random sample is taken from the training dataset, which has n observations and m features.

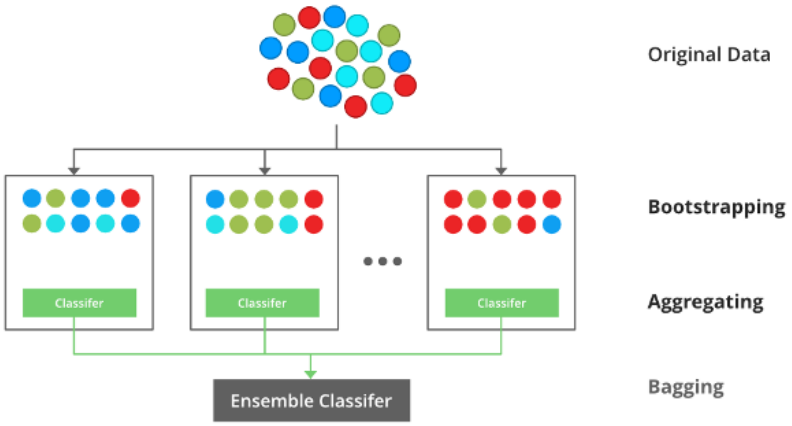

Figure: An illustration of the concept of bagging (Bagging vs Boosting in Machine Learning - GeeksforGeeks, 2021)

A subset of m features is randomly chosen from sample observations, and the feature that yields the best split is then used to split the nodes. Following that, the tree is grown, and the process is repeated an infinite number of times. It optimises forecasting by combining the output of individual decision trees (Brownlee 2020). Among the advantages of bagging is that it reduces data overfitting while enhancing model accuracy. It is also capable of handling multidimensional data successfully, and while it can be used in conjunction with other algorithms (Biswal 2021).

Exploratory Analysis

Stock price data were obtained using the yfinance Python package. The data obtained was composed of the daily Open, High, Low, Close, Adjusted Close, and Volume values for the chosen stocks. Respectively, these are the price of a stock at the start and end of a day, the lowest and highest values during a day, the close price after adjustment for corporate actions such as paying dividends, and the units of stock that were traded during the day.

The stock data was inspected for general quality and null values and was found to be complete and of appropriate format from the outset.

Dataset Statistics

Correlation among stock data was investigated to understand relationships between variables for a stock. Price data for the chosen stocks were found to have a generally strong positive correlation. Within individual stocks, price variables were all negatively correlated with the traded volume, indicating that higher trading volume occurs when the stock price is lower.

Investigation of Volatility

Stocks were selected from The Standard and Poor's 500 (S&P 500) to be used in the stock price prediction model. The S&P 500 is an index composed of 500 leading publicly traded U.S. companies and is regarded as a good gauge for the overall health of the stock market (Kenton 2001).

All stocks in the S&P 500 were assessed for volatility using standard deviation. Standard deviation is commonly used as a measure of volatility and stocks with higher volatility are usually regarded as riskier (Hayes 2021). The two stocks with the highest and lowest standard deviation, NVR (NVR) and Amcor (AMCR) respectively, were selected for use in the model.

Data Pre-Processing

Sentiment Analysis

By using the snscrape library, stock-related tweets were scraped and used for sentiment analysis. The scraped data was cleaned by removing unnecessary symbols, words, and punctuation. After cleaning all the tweets, a Vader Intensity Analyser was applied to find out the compound score for each tweet post**.** This score is the total of positive, negative, and neutral scores, which are then adjusted between -1 and +1. The extreme negative is -1, the extreme positive is +1, and 0 is neutral.

For each day, the positive, negative, and neutral values are calculated, and the probability is taken for the positive and negative values per day. Equations (1) and (2) will be used to calculate positive and negative sentiment probabilities for tweet postings. Here P_{Pos} denotes the likelihood of positive posts per day in N_{Total}**,** and P_{Neg} denotes the probability of negative posts per day in N_{Total} (Ko and Chang 2021).

N_{Total} = total number of posts per day.
N_{Pos} = total number of positive posts per day.
N_{Neg} = total number of negative posts per day.
P_{Pos} = probability of positive posts per day.
P_{Neg} = probability of negative posts per day.

$$P_{Pos} = N_{Pos} / N_{Total} \quad (1)$$

$$P_{Neg} = N_{Neg} / N_{Total} \quad (2)$$

Variable Selection

The total time period of the data and intervals between observations was chosen to be a length of two years, from April 20 2020 to April 19 2022. The intervals were set to 1 day; shorter options such as 1 hour or 5 minutes are available but models would easily overfit with so much noise.

The variables that were selected as independent variables are daily Open, High, Low and Close stock prices, in order to predict Close at the next day; generally, investors are concerned about prices at the end of trading days, and having all 4 variables should result in a more predictive model.

A separate set of LSTM models were also fitted using an additional variable, positive, which is the positive score from the Sentiment Analysis preprocessing, making it 5 independent variables in total. It is possible that positive opinions about a stock on the Internet would drive more trading and have an effect on the stock price.

Training-Test Split

To avoid overfitting and make sure the model generalises well to new, unseen data, the original data for the two stocks were split into a training set and test set, consisting of the first 80% and last 20% of the data by date respectively.

Numerical Variable Scaling
Numerical variables must be scaled in order to remove higher order significance, so the model does not value these greater values more than others. Normalisation was implemented to both the training and test sets in the form of 'MinMaxScaler' as it yields better results than standardisation 'StandardScaler', this is due to the former being vulnerable to outliers that have importance (Unterthiner et al. 2022).

Data for LSTM and Random Forest
Since this is a time series, we need to represent Open, High, Low, Close and positive (if applicable) at each time lag in days. This is done through the series_to_supervised function (Brownlee 2017), which can frame a standardised time series as a supervised learning dataset, creating new variables for times t, t-1, up to t-k. After the conversion, columns at time t were removed so that only the target variable, Close, remained at time t.

To test a number of scenarios, lags of 5, 10 and 20 trading days should be used, corresponding to 1 week, 2 weeks and approximately 1 month of previous stock prices. All subsequent work would be performed on these 3 lags.

The next step was to further split the data values into explanatory and target variables; for LSTM, the explanatory variables need to be further reshaped into a 3D array. The final set of inputs for LSTM model fitting were named differently depending on lag, stock, and whether sentiment analysis was factored in.

Model Implementation

LSTM
The LSTM models are built using the keras module. To optimize models for each stock and lag, we applied the KerasTuner framework. The first step was to define a function build_model, which contains the search space for different hyperparameters and even the presence or absence of additional layers. The search space is described below:

- Number of neurons for each layer: integers from 10 to 100 with step 5
- Learning rate of optimizer: any floating point from 0.0001 to 0.1, with log sampling such that equal probabilities are assigned to each order of magnitude.
- Dropout rate in dropout layers: any floating point from 0.1 to 0.5 with linear sampling
- By default, there is one LSTM layer and one dense layer only. Boolean methods determine whether to include a second and a third LSTM layer.
- Another boolean method determines whether dropout layers should be added. They must either immediately follow all LSTM layers, or not be added at all.

The model also uses Mean Absolute Error for the loss, the Adam optimizer, and the tanh activation function, which are default for LSTM.

Next a tuner class was selected to run the search. Subject to the search space, it would fit many models and find the one that minimises the loss on validation data. We decided on the BayesianOptimization algorithm, which chooses the first few combinations randomly, and then makes iterative improvements by taking performance history into account (Wadekar 2021). 10 trials of 50 epochs were run, with each trial having executions for robustness. After

the search was completed, the best model could be extracted and stored; the best hyperparameters, model structure and score could also be viewed.

Finally, the tuned model was fitted once again and evaluated. Early stopping was implemented in this step. The optimal hyperparameters of each lag and stock are shown in the tables below. Note that Keras does not allow reproducible results, so different models will be generated for each run.

AMCR

Lags	**5 day lag**	**10 day lag**	**20 day lag**
No. of neurons	10	100	35
Learning Rate	0.1	0.003214	0.008425
Dropout Rate	0.1	0.3924	0.1
No. of LSTM layers	1	1	1
Dropout layers	Yes	No	No

NVR

Lags	**5 day lag**	**10 day lag**	**20 day lag**
No. of neurons	25	30	100
Learning Rate	0.03943	0.001357	0.0001
Dropout Rate	0.1	0.1525	0.1
No. of LSTM layers	1	2	1
Dropout layers	No	No	Yes

LSTM Model with Sentiment Analysis

This follows that exact same procedure as LSTM except with a different input that includes sentiment analysis results.

AMCR

Lags	**5 day lag**	**10 day lag**	**20 day lag**
No. of neurons	15	35	60
Learning Rate	0.02578	0.005994	0.001262
Dropout Rate	0.1300	0.1640	0.1
No. of LSTM layers	3	1	1

Dropout layers	Yes	Yes	Yes

NVR

Lags	**5 day lag**	**10 day lag**	**20 day lag**
No. of neurons	100	10	10
Learning Rate	0.006770	0.002434	0.008449
Dropout Rate	0.2875	0.1	0.5
No. of LSTM layers	1	2	1
Dropout layers	No	No	No

RF Model

In implementing the RF Model, we used a RandomForestRegressor, and the following parameters were used for both stocks (AMCR and NVR) and lags (5, 10, and 20-day lag). See below:

- n_estimators — number of trees in the forest
- max_depth — maximum depth in a tree.
- min_samples_split — minimum number of data points before the sample is split.
- min_samples_leaf — minimum number of leaf nodes that are required to be sampled.
- bootstrap — sampling for data points, true or false
- random_state — generated random numbers for the random forest.

RF parameters before Hyperparameter for each stock and lags: {'n_estimators': [300], 'max_depth': [300], 'max_features': [4], 'random_state': [42]}.

For the RF model, hyperparameter-tuning with RandomizedSearchCV was done to increase the accuracy of each model. The best method to determine the optimal settings is to try different combinations and evaluate the performance of each model. However, evaluating each model only on the training set can lead to overfitting. Hyperparameter optimization accounts for overfitting through CrossValidation.

The parameters used for hyperparameter tuning can be shown below.

{'n_estimators': [20, 50, 100, 500, 1000], 'max_depth': np.arange[1, 15, 1], 'min_samples_split': [2, 10, 9], 'min_samples_leaf': np.arange[1, 15, 2], bootstrap: [True, False], 'random_state': [1, 2, 30, 42]}.

Based on the parameters above, the best parameters after hyperparameter tuning, we then use them to fit our model and improve the model's performance. See the tables below for the best parameters used for each stock and each lag.

AMCR

Lags	**5 day lag**	**10 day lag**	**20 day lag**
n_estimators	20	20	20
max_depth	13	5	8

min_samples_leaf	9	3	7
min_samples_split	10	9	9
random_state	1	2	1
bootstrap	False	True	False

NVR

Lags	**5 day lag**	**10 day lag**	**20 day lag**
n_estimators	50	100	50
max_depth	8	30	5
min_samples_leaf	3	5	7
min_samples_split	9	9	9
random_state	30	10	2
bootstrap	True	True	False

The use of plot_tree and graphviz was used to visualise every decision tree with each node. It shows various samples with each depth, and predicted price. See below:

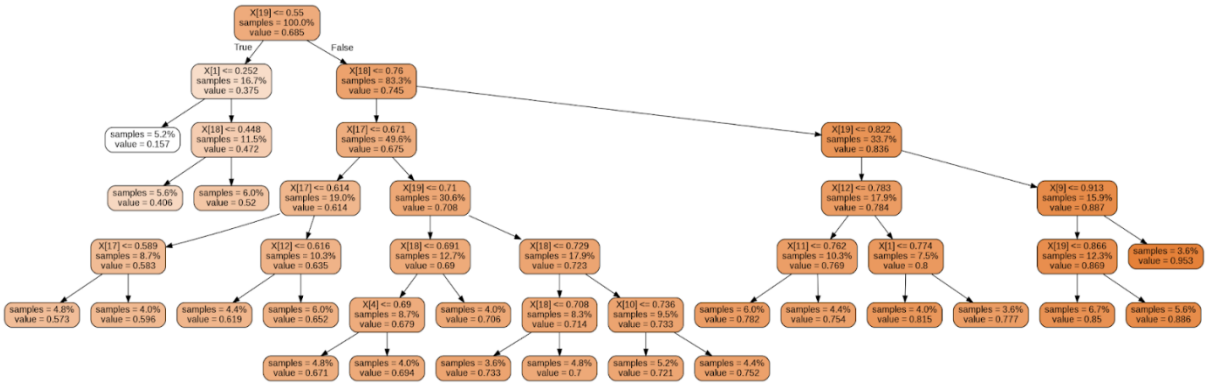

Figure: Plot of decision tree from random forest model

Bagging Model

We wanted to see whether combining the LSTM model and the RF model with a bagging algorithm would improve overall performance. To test this we implemented a bagging algorithm.

The way a bagging model works is by bootstrapping, a process of randomly choosing samples of training data and training existing models using those samples. In our case, we choose two samples, one for LSTM and the other for RF. Then, fitted the LSTM and the RF model with their respective samples, and took the average of their predictions.

A Bagging class was created with a constructor, a fit() method, and a predict() method. The constructor stores a list of all models and the sample proportion. The fit() method fits the

model by fitting all models to their respective samples. The predict() method gives predictions by having each model predict a value and taking the average of all predictions. An important note here is that the shapes of the data used in LSTM and RF models are different, therefore, both pre-processed datasets must be submitted to the bagging model.

Optimising the model involves finding the best sampling proportion, we did this by searching for the best proportion from a range between 0.1 and 0.9, step 0.1. Here are some results from one test.

Stock Name	lag (in days)	Best Sampling proportion
AMCR	5	0.6
AMCR	10	0.7
AMCR	20	0.9
NVR	5	0.1
NVR	10	0.1
NVR	20	0.3

We were able to observe that the best sampling proportion for AMCR is significantly higher than that of NVR. This result may suggest that a low sampling proportion is more suited for high volatility stocks, while a high sampling proportion is better for low volatility stocks. However, such a hypothesis is by no means conclusive because of low sample size.

Model Evaluation

Comparison Plot

For a qualitative evaluation of the model's performance, it is useful to see how the model's predictions performed against the true values it was aiming to forecast. For this the normalisation previously implemented would need to be reversed in order to gain insight to the true values of stock price. After this a simple plot of the predicted close price and original close price against epoch can be produced.

Loss Function Plot (LSTM)

When building the LSTM model the loss (error) function of mean absolute error (MAE) was chosen to configure the neural network as RMSE is more sensitive to outliers (Chai and Draxler 2014), its calculation being the absolute difference between the actual and predicted values. It is advantageous then to plot the error values at each epoch of the training and testing sequence for direct comparison, in order to determine useful insights such as overfitting and underfitting of the model to the training dataset. However, its primary use is to visualise when/if the model converges (Jose 2019).

Decision Tree Plot (RF)

When building the RF model, it may be useful to see the final decision tree being used to gain insight into the model's underlying structure, in an attempt to solve the black box problem

inherent to deep learning (Knight 2022). For this the graphviz API was used to plot each cell of the decision tree depicting the percentage of samples retaining to a given value.

Performance Metrics

It is necessary to calculate a cohort of scores and errors as metrics to quantitatively compare the performance of our models. The following metrics were chosen as the best way to evaluate a regression model (Hiregoudar 2020), where $\hat{y}$ is the predicted value and y being the true value. The scores retain positive orientation such that greater values mean greater model performance, however for errors the inverse is true.

Explained Variance Score - Used to describe the dispersion of errors made by the model. Where the variance of prediction errors and variance of actual values form a ratio by the following formula.

$$EVR(y, \hat{y}) = 1 - \frac{Var(y - \hat{y})}{Var(y)}$$

Coefficient of Determination - Otherwise known as the R^2 score is a measure of understanding a model's goodness of fit. Representative of the proportion of variance of the dependent (target) variable that can be explained by the independent variables in the model. Where the ratio of sum of squares error (SSE) of the regression model, by the SSE of the baseline model is used in the following formula.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

Another use for this metric is to recognise data leakage from the training dataset to the testing dataset as scores above 0.9 indicates the model may be too accurate (Bock 2017).

Mean Absolute Error - The error in prediction without consideration of direction over all instances. The error itself is simply the measure of difference between the true value and predicted value or euclidean distance, given by the following formula.

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1} y_i - \hat{y}_i$$

Root Mean Square Error - RMSE is considered the most important metric for predictive models (Grace-Martin 2013) as it has the benefit of penalising large errors more than smaller errors. It does so by calculating the square root of mean square error (MSE), which gives the variance of the residuals of true value against predicted, as seen in the following formula.

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}$$

RMSE can be described as the standard deviation of the unexplained variances between true and predicted values, or the distribution of our errors.

Median Absolute Error - Calculated as the median of all absolute differences between true values and predicted values as shown in the following formula.

$$MedAE(y, \hat{y}) = median(|y_1 - \hat{y}_1|, \dots, |y_n - \hat{y}_n|)$$

The MedAE score gives a different insight to the model's performance as it is robust to outliers, due to the median of the residuals being used rather than mean.

Maximum Residual Error - This metric simply gives the maximum residual calculated and as such produces the paramount error made by the model.

Mean Absolute Percentage Error - The easiest to interpret as a percentage which is scale independent, MAPE is calculated by

$$MAPE(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} \frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)}$$

Where ϵ is a very small but strictly positive number, to avoid undefined results when y is zero. Due to the absolute error being taken MAPE avoids the negation of negative and positive errors. Although it does acquire limitations when predictions are higher than the true value, due to percentage error not exceeding 100% for predictions below the true value. As a result, this metric will favour models that under-predict rather than over-predict.

Results and Discussion

During the construction and finalisation of each model's design all of the metrics and plots covered in model evaluation were used. However, it would be too exhaustive to compare on all of the metrics therefore Explained Variance Score, Coefficient of Determination, Root Mean Squared Error, and Mean Absolute Percentage Error were chosen due to being the best for evaluating regression models (Wu 2020), and for ease of interpretability. EVR, CoD, and MAPE can be used to compare between models, however RMSE can only be used to compare lags of the same stock. The results are as follows:

LSTM	EVR	CoD	RMSE	MAPE
AMCR, 5 day lag	0.7516	0.7229	0.1656	0.0110
AMCR, 10 day lag	0.7779	0.7772	0.1488	0.0099
AMCR, 20 day lag	0.7926	0.7900	0.1512	0.0100
NVR, 5 day lag	0.9447	0.9447	115.4025	0.0184
NVR, 10 day lag	0.9445	0.9434	117.7659	0.0191
NVR, 20 day lag	0.8991	0.8965	148.6965	0.0245

LSTM + Sentiment Analysis	EVR	CoD	RMSE	MAPE
AMCR, 5 day lag	0.7388	0.7358	0.1607	0.0109
AMCR, 10 day lag	0.7573	0.7463	0.1604	0.0106
AMCR, 20 day lag	0.7348	0.7257	0.1748	0.0120
NVR, 5 day lag	0.9469	0.9465	112.8024	0.0178
NVR, 10 day lag	0.9336	0.9101	144.5210	0.0240
NVR, 20 day lag	0.9428	0.9407	108.8639	0.0177

Random Forest	EVR	CoD	RMSE	MAPE
AMCR, 5 day lag	0.7612	0.7612	0.1528	0.0101
AMCR, 10 day lag	0.7352	0.7352	0.1622	0.0110
AMCR, 20 day lag	0.7303	0.7303	0.1713	0.0122
NVR, 5 day lag	0.7146	0.5962	311.9117	0.0414
NVR, 10 day lag	0.6987	0.5852	318.7407	0.0417
NVR, 20 day lag	0.7262	0.6683	266.1368	0.0351

Bagging Model	EVR	CoD	RMSE	MAPE
AMCR, 5 day lag	0.7662	0.7576	0.1540	0.01020
AMCR, 10 day lag	0.7286	0.7264	0.1649	0.01141
AMCR, 20 day lag	0.7652	0.7652	0.1598	0.01099
NVR, 5 day lag	0.8338	0.7908	224.5168	0.03195
NVR, 10 day lag	0.8394	0.7889	227.3885	0.03211
NVR, 20 day lag	0.8736	0.8697	166.7934	0.02538

Most models' prediction accuracy is greater when the underlying volatility of the stock is high. Which disagrees with our initial hypothesis of low volatility stocks yielding better results. In contrast to this however is the RF model which performs better with low volatility stocks, this may be due to greater variation in stock price with high volatility leading to greater variation in prediction, giving less accurate results.

Inference on models' performance using different lags is difficult as each model shows different results. Intuitively the more data the model receives the better its prediction may be, although this conclusion cannot be drawn as the results are inconclusive. The LSTM model shows greater prediction accuracy when greater lags are used for low volatility stocks but the inverse is true for high volatility stocks. In contradiction to this again is the RF model where greater lags increase the accuracy of prediction for high volatility stocks, but slightly decrease with low volatility stocks. The LSTM+Sentiment model and Bagging model show no inherent correlation.

When including sentiment analysis scores to the standalone LSTM model it is seen to increase the model's overall performance, this may be due to added dimensionality that the neural network can use to draw correlations and thus make better predictions (Daradkeh 2022). Some have suggested a stock's valuation should be used for long term predictions

although it is the markets sentiment on the stock that predicts its short-term price (Investments 2022). Which holds true as our model only implements short-term predictions.

It was expected that the bagging model would return the best results due to the prosperous nature of ensemble learning, particularly in performance and robustness (Brownlee 2020). However, this is not the case, primarily due to the standalone RF model being worse. Thus, once combined with the comparatively good LSTM model it decreases the overall accuracy of prediction.

Conclusion

This project was an exploration of using machine learning methods for stock price prediction. Price and twitter sentiment data for one high volatility and one low volatility stock were used to train two LSTM models, an RF model, and a bagging model that included an LSTM and an RF model. The group had hypothesised prior that low volatility would lend itself to better model performance.

The volatility hypothesis may be correct for the RF model, however the LSTM and bagged models performed better on the high volatility stock. A range of three different lags were also used for the models: 5, 10, and 20 days.

As with volatility, different models' performance was affected conversely. Higher lags seemed to improve performance in the LSTM model for the high volatility stock, whilst lower lags seemed to improve performance in the RF model for high volatility stocks. There was no conclusion to be made about the effect of lags on performance in the LSTM/Sentiment Analysis or Bagged models.

LSTM performance improved when sentiment analysis scores were included. This may suggest that sentiment has an effect on short term stock valuation.

Bagging multiple models did not yield the improved overall performance that was expected.

To summarise, each model's performance appears to be equal when predicting a low volatility stock. However, LSTM+Sentiment Analysis returns the greatest accuracy of all models when predicting a high volatility stock, shortly followed by the standalone LSTM model.

There are some areas where future work could be carried out to continue this work. Firstly, the models were only fitted for two stocks that were found to be at the extreme ends of volatility. Other stocks, funds or indices that do not have very high or low volatility could be investigated in the future. Similarly, shorter (2-3 days) or longer (50, 100 days) lags could be explored for improved effects on performance. Other options to explore include stocks with obvious increasing/decreasing trends or seasonality.

Another possibility for future work could be the application of k-fold cross validation which may improve accuracy when working with large sample sizes (Jiang and Wang, 2017).

One of the challenges encountered was the runtime of model fitting, especially on LSTM models. Since running hyperparameters optimization for multiple stocks, lags, and input data can take up to several hours. In the end the number of trials, epochs, and execution during

search were limited to decrease run time but this made the models subject to more randomness and less stable. If time allows, these settings can be increased in the future, along with testing more layers in order to further improve model performance. Furthermore, we could look into ways to obtain more processing resources that would be more effective than Google colab.

Another challenge was the interpretability of models. LSTM or bagging do not yield results that provide insight into the inner workings of the model such as a different model, e.g. linear regression, might provide. Random forest is easier to communicate with a decision tree plot but the whole Random Forest is difficult to visualise concisely.

Bibliography

Ballings, M., Van den Poel, D., Hespeels, N. and Gryp, R., 2015. *Evaluating multiple classifiers for stock price direction prediction.* Expert systems with Applications, 42(20), pp.7046-7056. doi: 10.1016/j.eswa.2015.05.013

Biswal, A., 2021. *Bagging in Machine Learning: Step to Perform And Its Advantages.* Available at: https://www.simplilearn.com/tutorials/machine-learning-tutorial/bagging-in-machine-learning [Accessed: 23 April 2022].

Bock, T., 2017. *8 Tips for Interpreting R-Squared*. Available at: https://www.displayr.com/8-tips-for-interpreting-r-squared/ [Accessed: 17 Apr 2022]

Bollerslev, T., Marrone, J., Xu, L., & Zhou, H. 2014. *Stock return predictability and variance risk premia: statistical inference and international evidence.* Journal of Financial and Quantitative Analysis, 49(3), 633-661. doi: 10.1017/S0022109014000453
.

Brownlee, J. 2019. *How to Convert a Time Series to a Supervised Learning Problem in Python.* Available at: https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python [Accessed: 01 April 2022]

Brownlee, J., 2020. *Bagging and Random Forest Ensemble Algorithms for Machine Learning*. Available at: https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/ [Accessed: 23 April 2022].

Brownlee, J., 2020. *How to Develop a Random Forest Ensemble in Python*. Available at: https://machinelearningmastery.com/random-forest-ensemble-in-python/ [Accessed: 24 April 2022].

Brownlee, J. 2020. *Why Use Ensemble Learning?.* Available at: https://machinelearningmastery.com/why-use-ensemble-learning/#:~:text=There%20are%20two%20main%20reasons,the%20predictions%20and%20model%20performance. [Accessed: 24 April 2022].

Brownlee, J., 2021. *A Gentle Introduction to Long Short-Term Memory Networks by the Experts*. Available at: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ [Accessed: 24 April 2022].

Chai, T. and Draxler, R.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

Chen, J. 2021. *Index.* Available at: https://www.investopedia.com/terms/i/index.asp [Accessed: 29 March 2022].

Chong, E., Han, C., & Park, F. C. 2017. *Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies.* Expert Systems with Applications, 83, 187-205. doi: 10.1016/j.eswa.2017.04.030

Colah.github.io. 2015. *Understanding LSTM Networks -- colah's blog.* Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed: 24 April 2022].

Daradkeh, M. 2022. *A Hybrid Data Analytics Framework with Sentiment Convergence and Multi-Feature Fusion for Stock Trend Prediction.* Electronics 11(2), p. 250. doi: 10.3390/electronics11020250.

GeeksforGeeks. 2021. *Bagging vs Boosting in Machine Learning - GeeksforGeeks.* Available at: https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/ [Accessed: 24 April 2022].

Grace-Marin, K. 2013. *Assessing the Fit of Regression Models.* Available at: https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/#:~:text=RMSE%20is%20a%20good%20measure,than%20one%20are%20often%20useful. [Accessed: 03 Apr 2022]

Hayes, A. 2021. *Volatility.* Available at: https://www.investopedia.com/terms/v/volatility.asp [Accessed: 05 April 2022].

Hiregoudar, S. 2020. *Ways to Evaluate Regression Models.* Available at: https://towardsdatascience.com/ways-to-evaluate-regression-models-77a3ff45ba70 [Accessed: 24 April 2022].

Investments, L. 2022. *Sentiment Analysis - What is market sentiment and how does it affect the stock market?.* Available at: https://www.lehnerinvestments.com/en/sentiment-analysis-stock-market-sentiment [Accessed: 24 April 2022].

Jiang, G. and Wang, W. 2017. *Error estimation based on variance analysis of k -fold cross-validation.* Pattern Recognition 69, pp. 94-106. doi: 10.1016/j.patcog.2017.03.025.

Jose, G.V., 2019, *Useful Plots to Diagnose your Neural Network.* Available at: https://towardsdatascience.com/useful-plots-to-diagnose-your-neural-network-521907fa2f45 [Accessed: 23 April 2022]

Kenton, W. 2022. *The S&P 500 Index: Standard & Poor's 500 Index.* Available at: https://www.investopedia.com/terms/s/sp500.asp [Accessed: 18 April 2022].

Khaidem, L., Saha, S. and Dey, S.R., 2016. *Predicting the direction of stock market prices using random forest.* arXiv preprint arXiv:1605.00003. doi:10.48550/arXiv.1605.00003

Knight, W. 2022. *The Dark Secret at the Heart of AI.* Available at: https://www.technologyreview.com/2017/04/11/5113/the-dark-secret-at-the-heart-of-ai/ [Accessed: 24 April 2022].

Ko, C. and Chang, H., 2021. *LSTM-based sentiment analysis for stock price forecast.* PeerJ Computer Science, 7, p.e408. doi: 10.7717/peerj-cs.408

Kumar, M. and Thenmozhi, M., 2006. *Forecasting stock index movement: A comparison of support vector machines and random forest.* Indian Institute of Capital Markets 9th Capital Markets Conference Paper.

Lee, H., Pham, P., Largman, Y. and Ng, A., 2009. *Unsupervised feature learning for audio classification using convolutional deep belief networks.* Advances in neural information processing systems, 22.

Mehtab, S., Sen, J. and Dutta, A., 2020.. *Stock price prediction using machine learning and LSTM-based deep learning models*. In Symposium on Machine Learning and Metaheuristics Algorithms, and Applications (pp. 88-106). Springer, Singapore.

Phi, M., 2018. *Illustrated Guide to LSTM's and GRU's: A step by step explanation.* [online] Medium. Available at: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 [Accessed: 24 April 2022].

Sarker, I. 2021. *Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions.* SN Computer Science 2(6), p. p1. doi: 10.1007/s42979-021-00815-1.

Schmidt, J. et al. 2019. R*ecent advances and applications of machine learning in solid-state materials science.* npj Computational Materials 5(1). doi: 10.1038/s41524-019-0221-0.

Sen, J. and Chaudhuri, T.D., 2018. *Stock price prediction using machine learning and deep learning frameworks*. In Proceedings of the 6th International Conference on Business Analytics and Intelligence, Bangalore, India (pp. 20-22).
Silipo, R., 2019. *From a Single Decision Tree to a Random Forest.* Available at: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147 [Accessed: 24 April 2022].

Unterthiner, T. et al. 2022. *Compare the effect of different scalers on data with outliers.* Available at: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html [Accessed: 24 April 2022].

von Szeliski, V. 1935. *The Statistical Analysis of Stock Prices*. Econometrica 3(4), p. 435. doi: 10.2307/1905635.

Wadekar, S. 2021. *Hyperparameter Tuning in Keras: TensorFlow 2: With Keras Tuner: RandomSearch, Hyperband, BayesianOptimization.* Available at: https://medium.com/swlh/hyperparameter-tuning-in-keras-tensorflow-2-with-keras-tuner-randomsearch-hyperband-3e212647778f [Accessed: 12 April 2022]

Wu, S. 2020. *What are the best metrics to evaluate your regression model?.* Available at: Available at: https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b [Accessed: 24 April 2022].

Jin, Z., Yang, Y. and Liu, Y., 2020. *Stock closing price prediction based on sentiment analysis and LSTM.* Neural Computing and Applications, 32(13), pp.9713-9729. doi: 10.1007/s00521-019-04504-2