Student Number: 11253444

Abstract

This report is the continuation of the course work 1 assignment title: “Identify the most accurate machine learning model for stock price prediction: A comparative study”. It consists of the implementation and evaluation phase of the project as well as the author’s critical reflection of the challenges, lessons and professional issues faced in the development of this report. Additionally, it contains the heuristic evaluation done by the author with regards to the project’s usability.

Individual computing science project

Assignment Title: Identifying the most accurate machine learning model for stock price prediction: A comparative analysis



Contents

[Introduction 4](#_Toc162883713)

[Implementation 4](#_Toc162883714)

[Linear regression – LR 7](#_Toc162883715)

[K – Nearest Neighbour – kNN 9](#_Toc162883716)

[Support vector machine – SVM 10](#_Toc162883717)

[Random forest – RF 12](#_Toc162883718)

[Long Short – term Memory – LSTM 13](#_Toc162883719)

[Evaluation 16](#_Toc162883720)

[Security measures 16](#_Toc162883721)

[Data security 16](#_Toc162883722)

[Performance 16](#_Toc162883723)

[Linear Regression – LR 17](#_Toc162883724)

[K – Nearest Neighbour 22](#_Toc162883725)

[Support Vector Machine 24](#_Toc162883726)

[Random Forest 27](#_Toc162883727)

[Long Short-Term Memory 29](#_Toc162883728)

[Test data and Unseen data results 32](#_Toc162883729)

[Limitations 32](#_Toc162883730)

[Critical Reflection 32](#_Toc162883731)

[Challenges 32](#_Toc162883732)

[Successes 33](#_Toc162883733)

[Lessons 33](#_Toc162883734)

[Professional and Ethical issues 33](#_Toc162883735)

[Conclusion 33](#_Toc162883736)

[Reference 35](#_Toc162883737)

[Appendix 36](#_Toc162883738)

[Linear regression 36](#_Toc162883739)

[K – Nearest Neighbour 36](#_Toc162883740)

[Support Vector Machine 37](#_Toc162883741)

[Random Forest 38](#_Toc162883742)

[Long Short – Term Memory 39](#_Toc162883743)

[Heuristic Usability 40](#_Toc162883744)

[Figure 1 Imported libraries 4](#_Toc162883745)

[Figure 2 Downloaded NVIDIA dataframe 5](#_Toc162883746)

[Figure 3 Dataframe info and assigning dataframe to df 6](#_Toc162883747)

[Figure 4 Dataframe informetaion using .head() 6](#_Toc162883748)

[Figure 5 df visualisation 7](#_Toc162883749)

[Figure 6 Creating rolling means 20-day and 50-day 8](#_Toc162883750)

[Figure 7 20-day rolling means data fitting 8](#_Toc162883751)

[Figure 8 50-day rolling means data fitting 9](#_Toc162883752)

[Figure 9 Creating 'Change\_ot' column 9](#_Toc162883753)

[Figure 10 Setting feature and target for k-NN 10](#_Toc162883754)

[Figure 11 Using ‘StratifiedKFold’ with k-NN 10](#_Toc162883755)

[Figure 12 Creating feature and target for SVM 11](#_Toc162883756)

[Figure 13 Using ‘StratifiedKFold’ with SVM 11](#_Toc162883757)

[Figure 14 Creating feature and target for RF 12](#_Toc162883758)

[Figure 15 Using ‘StratifiedKFold’ on RF 13](#_Toc162883759)

[Figure 16 Setting 'output\_var' as target on LSTM 13](#_Toc162883760)

[Figure 17 Setting features for LSTM and scaling 14](#_Toc162883761)

[Figure 18 Using ‘TimeSeriesSplit’ on LSTM 14](#_Toc162883762)

[Figure 19 Reshaping features into 3-dimensional array 15](#_Toc162883763)

[Figure 20 Setting LSTM parameters 15](#_Toc162883764)

[Figure 21 Fitting the features into LSTM model 16](#_Toc162883765)

[Figure 22 MSE and R² result for 20-day LR test data 17](#_Toc162883766)

[Figure 23 Visualising 20-day LR model prediction with actual data 18](#_Toc162883767)

[Figure 24 MSE and R² result for 50-day LR test data 18](#_Toc162883768)

[Figure 25 Visualising 50-day LR model prediction with actual data 19](#_Toc162883769)

[Figure 26 MSE and R² result for 20-day LR unseen data 19](#_Toc162883770)

[Figure 27 Visualising 20-day LR model prediction with unseen data 20](#_Toc162883771)

[Figure 28 MSE and R² result for 50-day LR unseen data 20](#_Toc162883772)

[Figure 29 Visualising 50-day LR model prediction with unseen data 21](#_Toc162883773)

[Figure 30 k-NN cross-validation score 22](#_Toc162883774)

[Figure 31 k-NN confusion matrix 23](#_Toc162883775)

[Figure 32 k-NN classification report 23](#_Toc162883776)

[Figure 33 k-NN confusion matrix with unseen data 24](#_Toc162883777)

[Figure 34 k-NN classification report with unseen data 24](#_Toc162883778)

[Figure 35 SVM cross-validation score 25](#_Toc162883779)

[Figure 36 SVM confusion matrix 25](#_Toc162883780)

[Figure 37 SVM classification report 26](#_Toc162883781)

[Figure 38 SVM confusion matrix with unseen data 26](#_Toc162883782)

[Figure 39 SVM classification report with unseen data 27](#_Toc162883783)

[Figure 40 RF cross-validation score 27](#_Toc162883784)

[Figure 41 RF confusion matrix 28](#_Toc162883785)

[Figure 42 RF classification report 28](#_Toc162883786)

[Figure 43 RF confusion matrix with unseen data 29](#_Toc162883787)

[Figure 44 RF classification report with unseen data 29](#_Toc162883788)

[Figure 45 LSTM model fitting with 100 epochs 30](#_Toc162883789)

[Figure 46 Visualised prediction of LSTM test data 30](#_Toc162883790)

[Figure 47 RMSE and MAPE of LSTM test data 31](#_Toc162883791)

[Figure 48 Visualised prediction of LSTM unseen data 31](#_Toc162883792)

[Figure 49 RMSE and MAPE of LSTM unseen data 32](#_Toc162883793)

[Figure 50 Machine learning model results 32](#_Toc162883794)

[Figure 51 Linear Regression 36](#_Toc162883795)

[Figure 52 k-NN algorithm visualisation 37](#_Toc162883796)

[Figure 54 SVM on separable data 38](#_Toc162883797)

[Figure 55 Random Forest algorithm 38](#_Toc162883798)

[Figure 56 LSTM neuron 39](#_Toc162883799)

[Figure 57 (Chen & Zhou, 2020) 39](#_Toc162883800)

[Figure 58 Heuristic evaluation table 40](#_Toc162883801)

# Introduction

This report builds upon the authors coursework 1 with the project title: “Identify the most accurate machine learning model for stock price prediction: A comparative study”. The report covers the implementation and evaluation phase of the project (Software Development Life Cycle) SDLC as well as includes my critical reflection of the project’s challenges, lessons and professional issues. Furthermore, an appendix is added to provide short definitions of each model’s function.

# Implementation

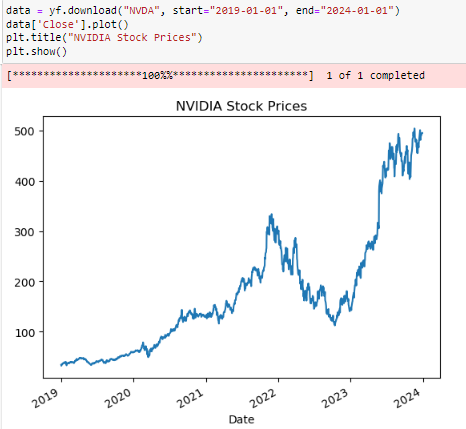
In the implementation phase, the necessary libraries were utilised (Figure 1).

Figure 1 Imported libraries



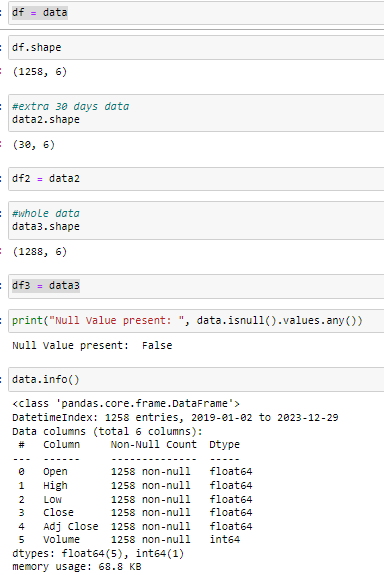
The datasets were then downloaded using ‘yfinance’ API. Subsequently, the datasets were then converted to CSV files then into pandas dataframes (Figure 2).

Figure 2 Downloaded NVIDIA dataframe



This was done to train and test the model with ‘df’ and to further evaluate the model’s predictive capabilities with unseen data from ‘df2’ (Figure 3).

Figure 3 Dataframe info and assigning dataframe to df



Additionally, data exploration was done to visualise the dataset. OHLC (Open, High, Low, Close) data was used as it is a common instrument in the financial sector to see the price movement of a given stock price (Figure 4).

Data cleaning is performed to assess the structure of the data, and null values were then dropped.

Figure 4 Dataframe informetaion using .head()

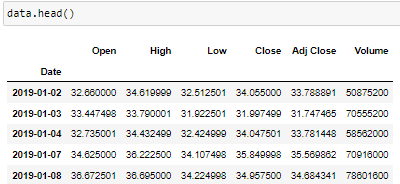
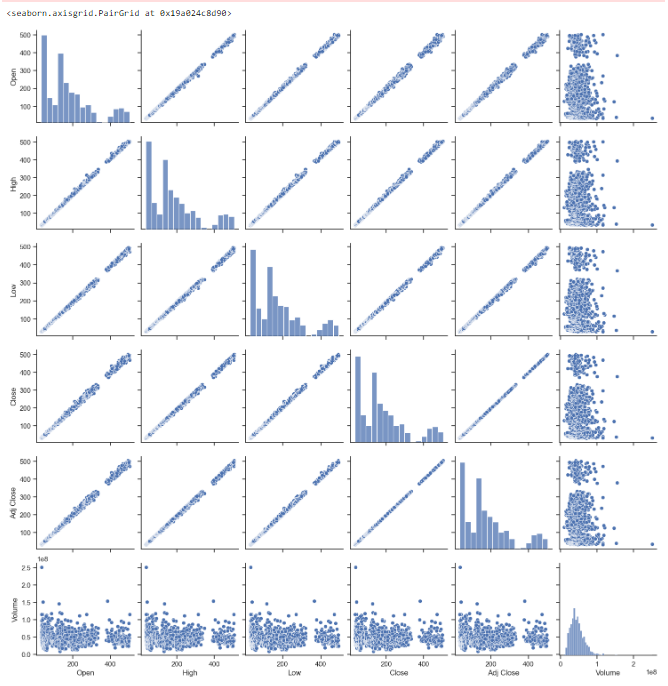


Figure 5 df visualisation



Furthermore, this project utilises the “Adj Close” of the following day (1 row below) as the target data, while the features of the current day are used for prediction. To do this the “Adj Close” column is shifted up by 1 row, and the last row in the dataset is dropped, as it will not have any target data.

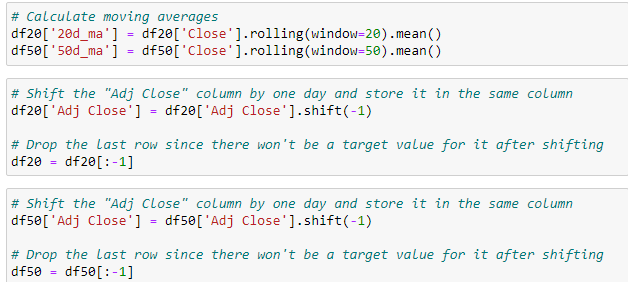
The ‘train\_test\_split’ function is utilised to split the data into 80% training and 20% testing sets. The implementation for each model is as follows:

## Linear regression – LR

In this model, a new column “20d\_ma” was added. The data within this column is generated using a rolling means function, which calculates the mean of the “Close” column over a 20-day window (Figure 6).

Furthermore, another dataframe is created using a rolling means function with a 50-day window (Figure 6).

Figure 6 Creating rolling means 20-day and 50-day



The features are then fitted using ‘train\_test\_split’ and scores are obtained (Figure 7 – 8).

Figure 7 20-day rolling means data fitting

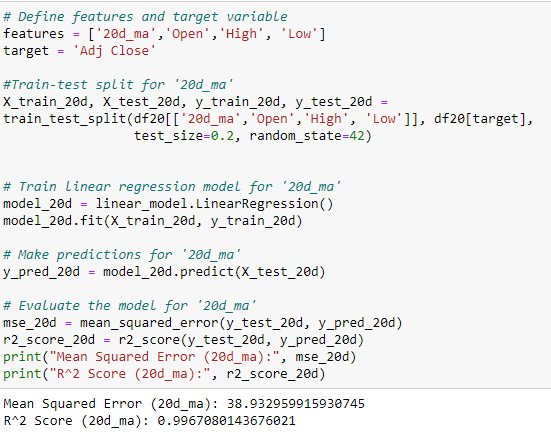
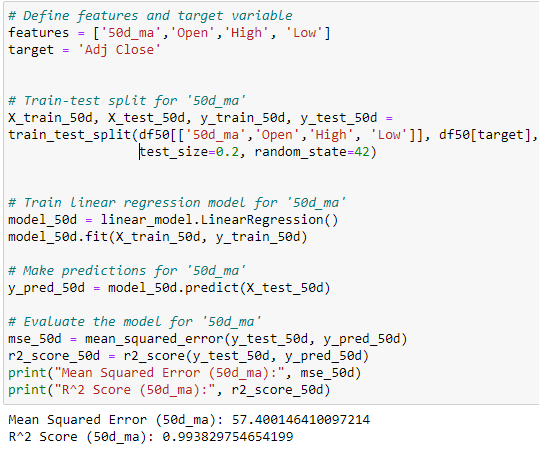


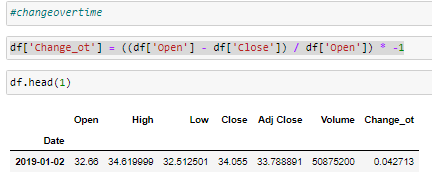
Figure 8 50-day rolling means data fitting



## K – Nearest Neighbour – kNN

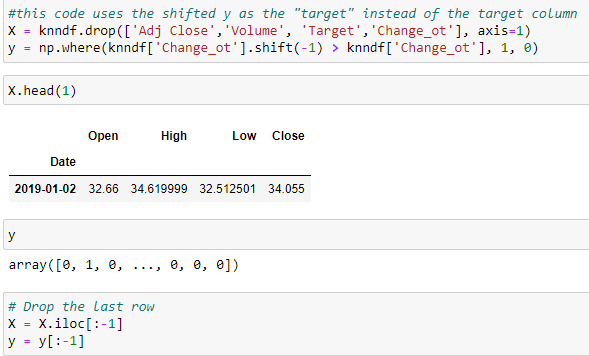
In this model, a new column named “Change\_it” is added. It utilises the “Open” and ”Close” values to compute the percentage change between the prices. A positive value indicates an increase in price, whereas a negative value indicates a decrease (Figure 9).

Figure 9 Creating 'Change\_ot' column



This column is then shifted 1 day ahead and a Boolean mask is performed. 1 if the value of the shifted “Change\_ot” is greater than that of the current day and 0 if it otherwise. This will be the target variable (Figure 10).

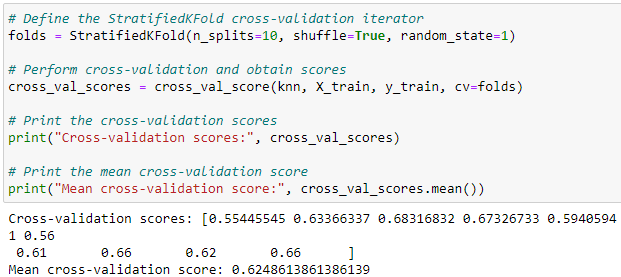
Figure 10 Setting feature and target for k-NN



Additionally, the features selected will be the “Open”, “High”, “Low”, and “Close” column data. These variables are then fitted using the ‘train\_test\_split’ method with 3 n\_neighbours.

Furthermore, the train data will also be split using cross\_val\_score for cross-validation, using StratifiedKFold which divides the data into subsets. The training data is then fitted with the model and results are obtained (Figure 11).

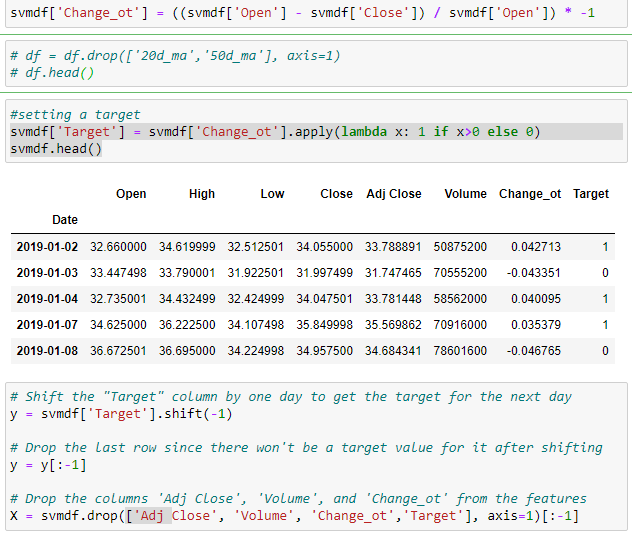
Figure 11 Using ‘StratifiedKFold’ with k-NN



## Support vector machine – SVM

Similar to the k-NN implementation, the “Change\_ot” is utilised. However, in this model a “Target” column is created and a lambda function is implemented. If the value of “Change\_ot” is greater than 0, then it will be converted to 1; otherwise, it set to 0. Subsequently, the “Target” column is also shifted to 1 day ahead (Figure 12).

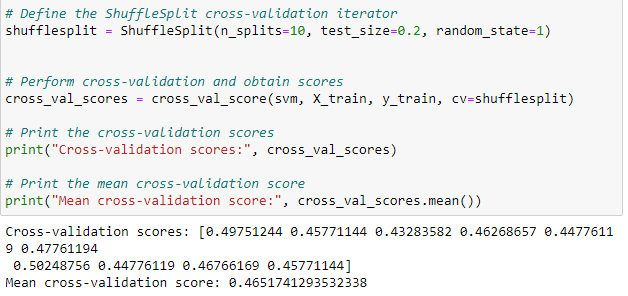
Figure 12 Creating feature and target for SVM



The target and features are then split using the ‘train\_test\_split’ and the training data further split using ‘StratifiedKFold’.

The training data is the fitted with the model and scores are obtained (Figure 13).

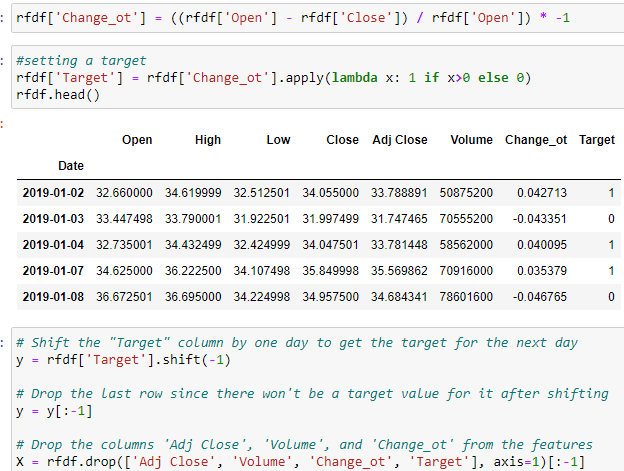
Figure 13 Using ‘StratifiedKFold’ with SVM



## Random forest – RF

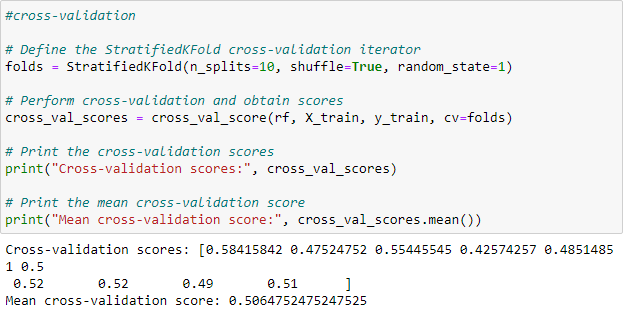
Comparably, the implementation of the model is similar to SVM. The shifted “Target” column is used as the dependent variable and the remaining “Open”, “High”, “Low” and “Close” columns act as independent variables (Figure 14).

Figure 14 Creating feature and target for RF



The data are split using the ‘train\_test\_split’ method and the training data is further split using ‘StratifiedKFold’ (Figure 15).

Figure 15 Using ‘StratifiedKFold’ on RF

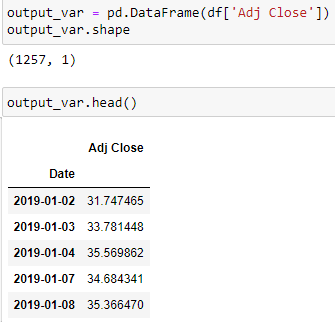


## Long Short – term Memory – LSTM

In this model the data is similarly divided into the target and feature variables wherein the “Adj Close” of the following day will be the dependent variable (Figure 16 – 17).

However, the features have to be pre-processed first to fit the LTSM model.

Figure 16 Setting 'output\_var' as target on LSTM



As the data is dealing with large numbers, ‘scaler.fit\_transform’ is used to scale down the data into smaller values for to prevent one single feature from dominating the distance calculations. This will help improve the model’s performance (Figure 17 – 18).

Figure 17 Setting features for LSTM and scaling

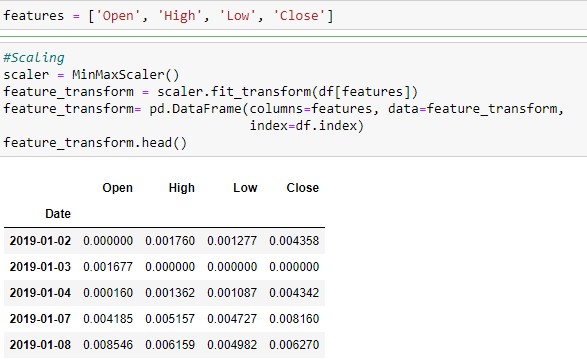
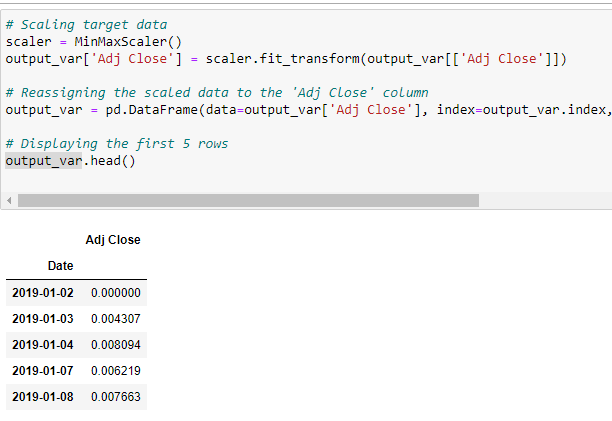
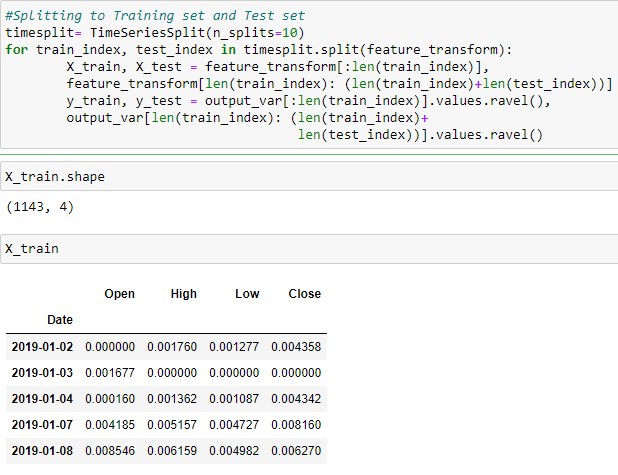


Figure 18 Scaling target variable



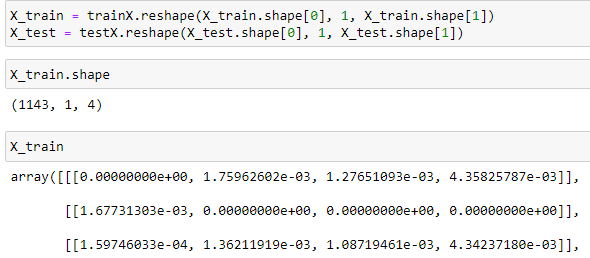
‘TimeSeriesSplit(n\_splits=10)’ is then used to cross-validate the features in 10 splits (Figure 18).

Figure 19 Using ‘TimeSeriesSplit’ on LSTM



Additionally, the features are then converted into a 2-dimensional array then into a 3-dimensional array. The final output would have sample size, number of time steps, and number of features for its dimensions. This is then fitted into the model (Figure 19).

Figure 20 Reshaping features into 3-dimensional array

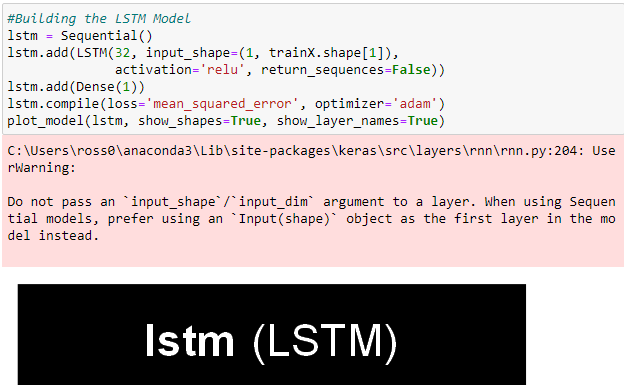


The sequential model is then used as the input and output of stock prices are sequence data. 32 neurons are specified with 1 time step (Figure 20).

‘relu’ or rectified linear unit is used as the activation function as it addresses the vanishing gradient issue. This problem arises when the gradient calculated for the weights become too small resulting in slow learning.

Additionally, it addresses sparsity. If the inputs are zero or less, ‘relu’ turns them into 0 which helps the model focus on the relevant features and discard noise.

Figure 21 Setting LSTM parameters

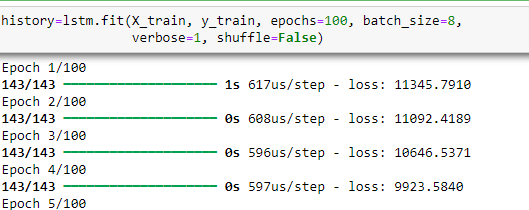


Adam is then used as the optimiser as it is an extension of stochastic gradient descent.

However, it is more computationally efficient, requires little memory and require little hyper-parameter tuning.

100 epochs are then used for the model to go through the whole training data (Figure 21).

Figure 22 Fitting the features into LSTM model



# Evaluation

## Security measures

All facets of a machine learning system can be compromised in confidentiality, integrity and availability. Therefore, proper understanding of mitigating features is needed to assure investors. This can include the use of expert systems that use “if-then” rules and subjective logic which factors in uncertainty and source into the decision (*Principles for the Security of Machine Learning*, 2022).

## Data security

Machine learning models are particularly vulnerable to training set poisoning which can lead to wrong predictions. To address this, data validation and verification is needed before using it to train the model. Using model ensembles can also be done to reduce the impact of data poisoning attacks (*OWASP Machine Learning Security Top Ten 2023 | ML02:2023 Data Poisoning Attack | OWASP Foundation*, n.d.).

## Performance

Specific tests are conducted to evaluate the performance of models. For regression models, Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE) and R² (coefficient of determination) are used.

MSE measures how near the calculated regression line is the set data points. MAPE measures the percentage difference between the actual and predicted values. The nearer it is to 0 the more accurate the model. R² (coefficient of determination) represents how good the regression model fits to actual data, with 1 as perfect fit and 0 poor fit (*Sklearn.metrics.r2\_score*, n.d.).

For classification models, ‘cross\_val\_scores’ and ‘classfication\_report()’ methods are used. ‘cross\_val\_scores’ returns an array of scores obtained from the ‘StratifiedKFold’ method, along with their mean. ‘classfication\_report()’ provides metrics such as precision, recall, accuracy, and F1-score based on the input y\_test and y\_pred (*13. Evaluation — Data Science 0.1 Documentation*, n.d.).

Precision is the ratio of True positives to all the Positive predictions, indicating the model’s ability to identify positive instances. Recall also known as sensitivity, measures of how correctly the model identifies True Positives out of all Positives. F1-score is the harmonic mean of precision and recall. Accuracy is the ratio of the total number of correct predictions with the total number of predictions, providing an overall measure of model performance.

## Linear Regression – LR

There are 2 linear regression models trained, the 20-day and 50-day “Close” rolling means. Both performed really well with 0.99 score in R².

Additionally, both have relatively low MSE scores with the 20-day model scoring 38.93 and the 50-day with 57.40. This indicates the model can accurately predict the actual value (Figure 22 – 25).

Figure 23 MSE and R² result for 20-day LR test data



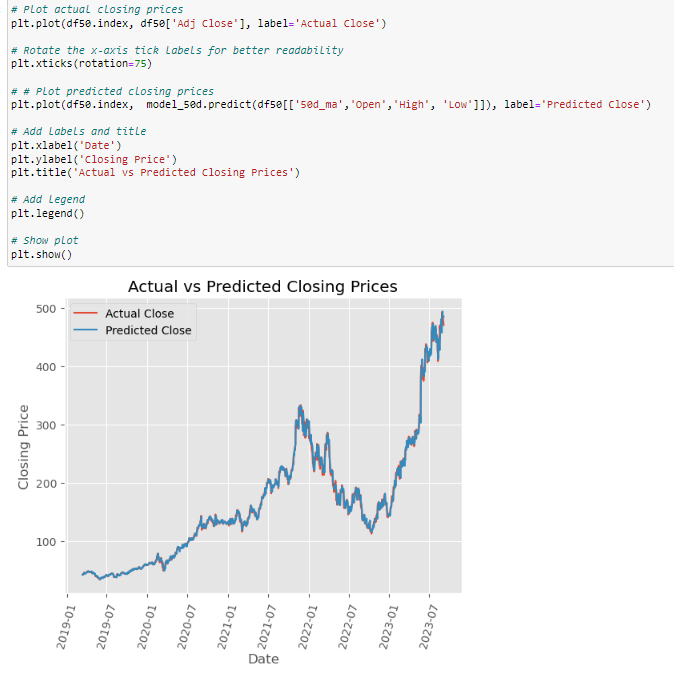
Figure 24 Visualising 20-day LR model prediction with actual data



Figure 25 MSE and R² result for 50-day LR test data



Figure 26 Visualising 50-day LR model prediction with actual data



Using df2 (unseen data), the R² of the 20-day model dropped to 63% with an MSE of 64.42. Similarly, the 50-day model had an R² of 63% with an MSE of 63.09. This means the models were still able to predict some values but there is a marked dropped in accuracy (Figure 26 – 29).

Figure 27 MSE and R² result for 20-day LR unseen data

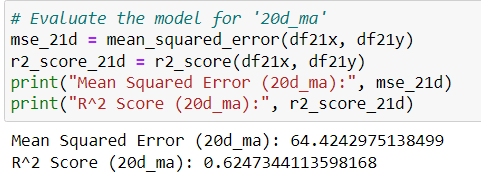


Figure 28 Visualising 20-day LR model prediction with unseen data



Figure 29 MSE and R² result for 50-day LR unseen data

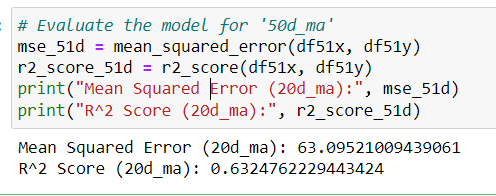
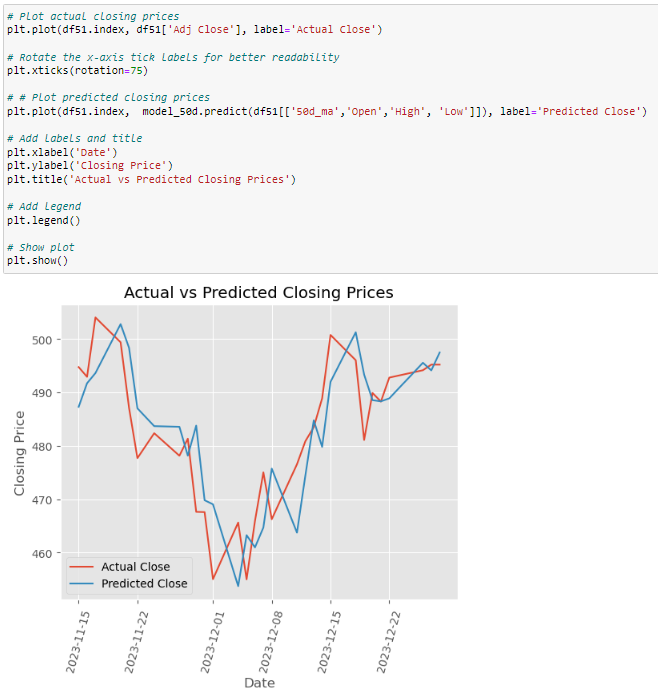


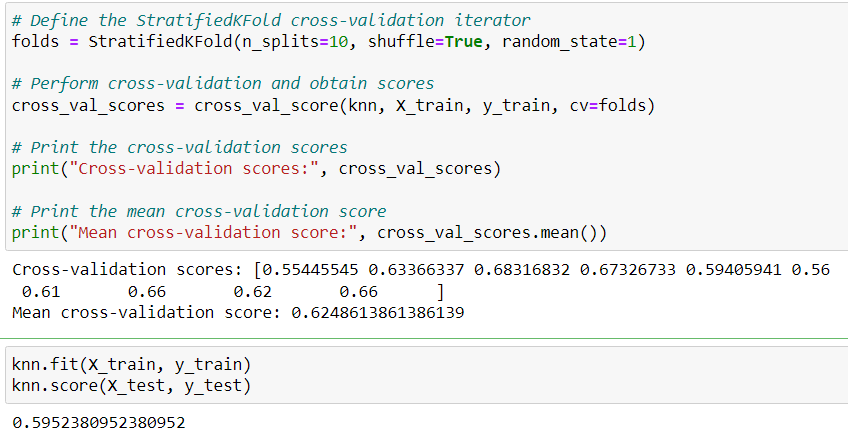
Figure 30 Visualising 50-day LR model prediction with unseen data



## K – Nearest Neighbour

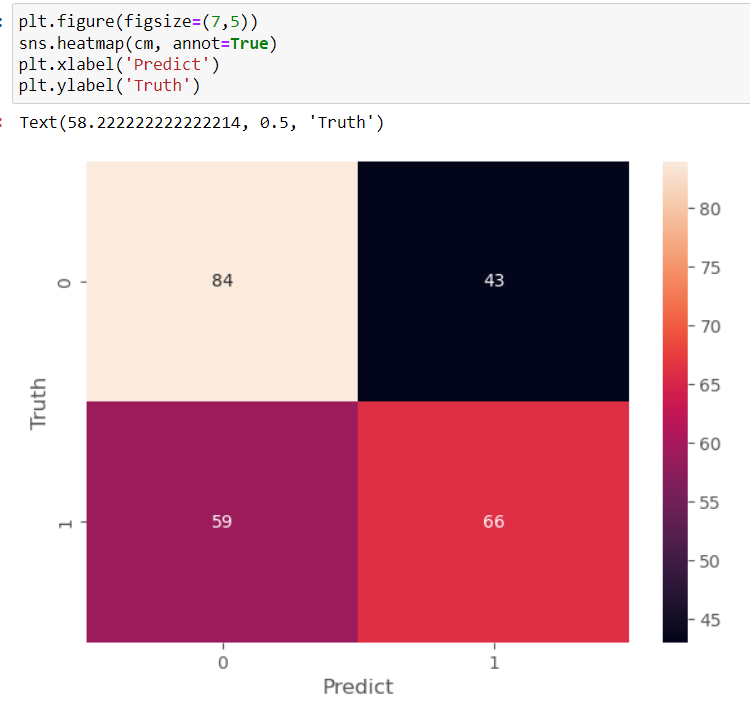
The KNN training model scored a mean of 62% in its cross-validation and a 59% score in its test data. This indicates that the model can predict the “Adj Close” of the next day if it will go up or down, with an accuracy of 60% (Figure 30).

Figure 31 k-NN cross-validation score



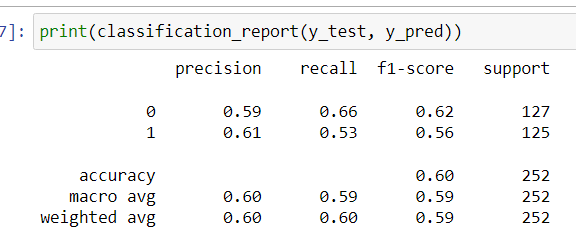
In the confusion matrix, it can be seen that the model predicted the 0, 59 times when it should be 1. Also, it predicted 1 when it should be 0, 43 times (Figure 31).

Figure 32 k-NN confusion matrix



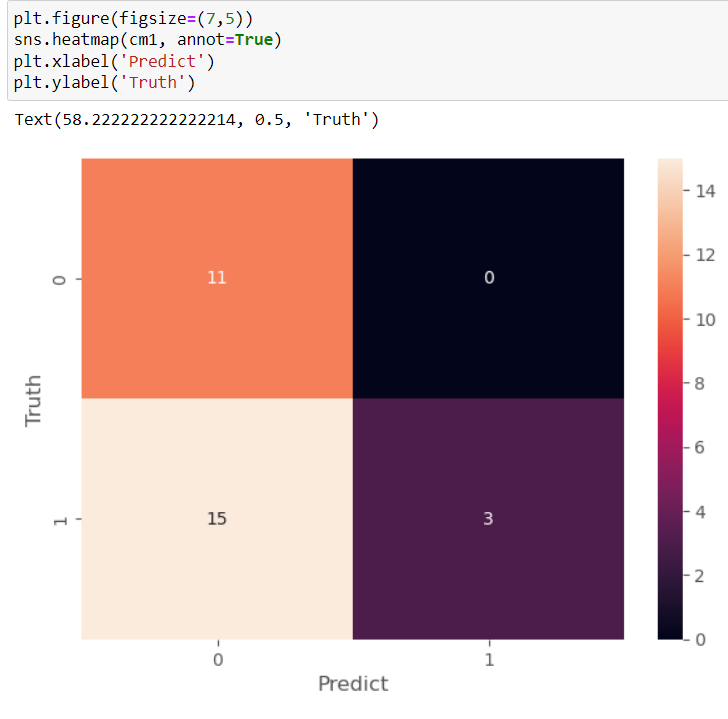
Further analysis can be seen using the ‘classification\_report’ (Figure 32).

Figure 33 k-NN classification report



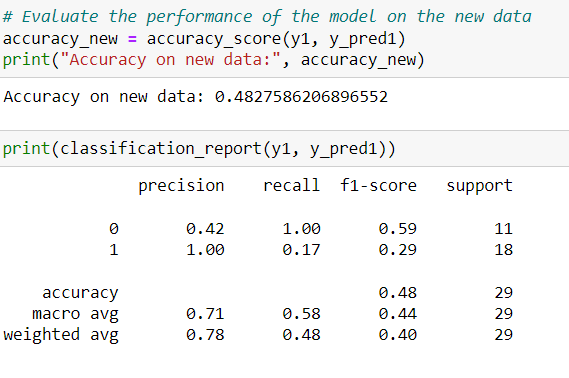
Testing the model with df2, it was able to predict 0 correctly 11 times. However, it also predicted it to be 0 when it should be 1, 15 times (Figure 33).

Figure 34 k-NN confusion matrix with unseen data



The model scored poorly with a 12% drop in accuracy, it was only able to predict 48% of the actual data (Figure 34).

Figure 35 k-NN classification report with unseen data



## Support Vector Machine

SVM performed poorly, scoring a mean cross-validation score of 46% on its training dataset and 53% on its testing dataset (Figure 35 – 37).

Figure 36 SVM cross-validation score

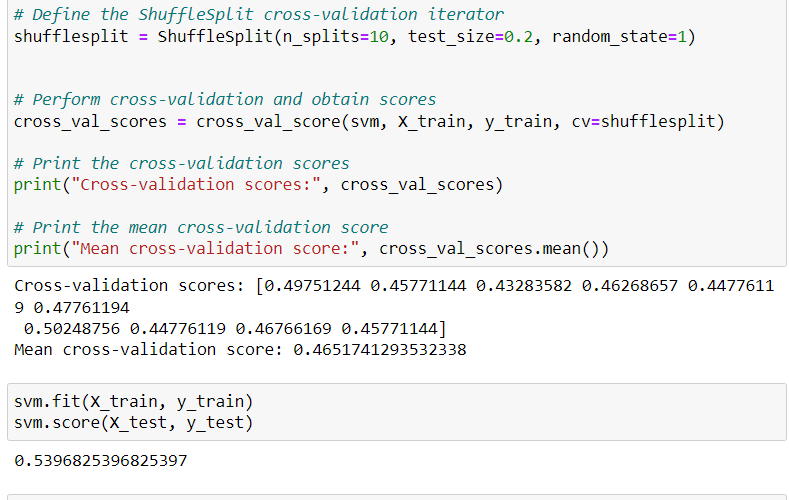


Figure 37 SVM confusion matrix

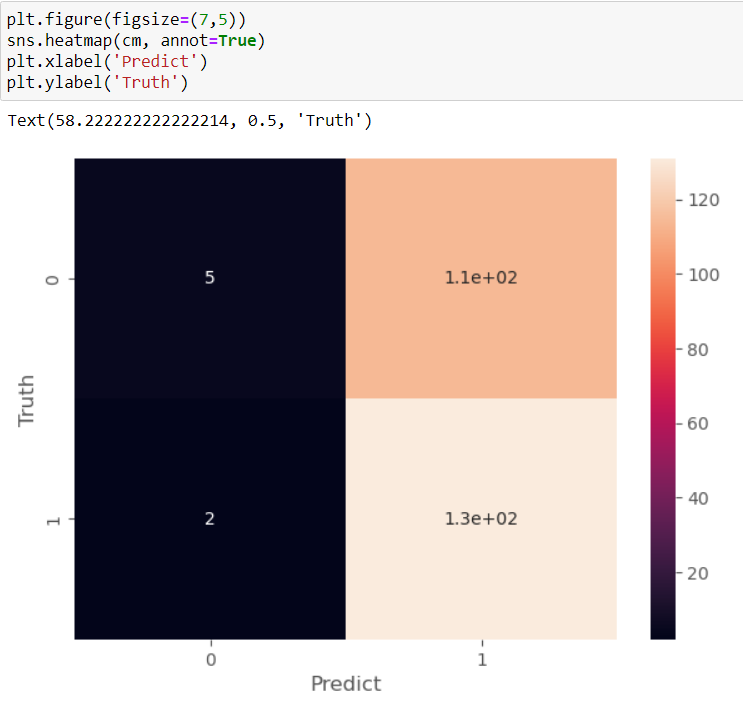
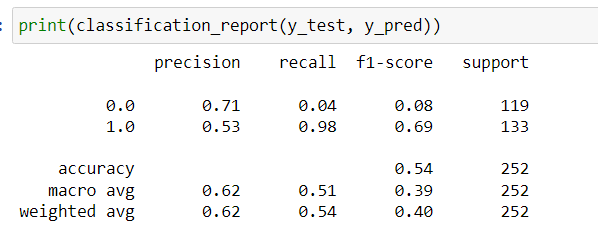


Figure 38 SVM classification report



Furthermore, its accuracy sharply dropped with unseen data as it scored 34% in prediction accuracy (Figure 38 – 39).

Figure 39 SVM confusion matrix with unseen data

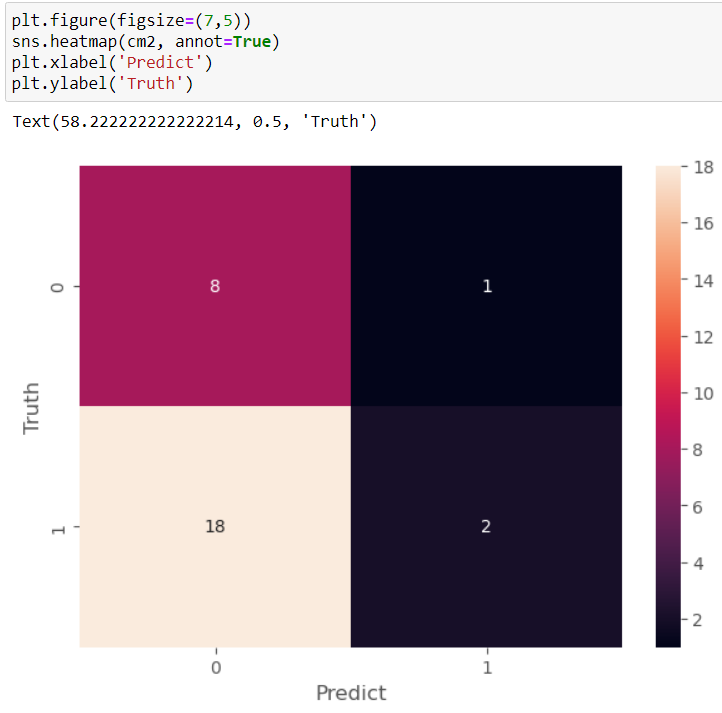
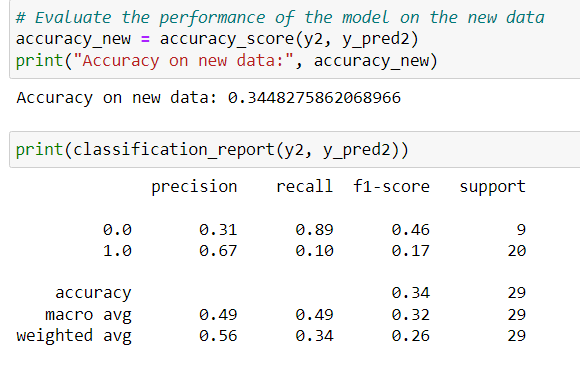


Figure 40 SVM classification report with unseen data



## Random Forest

RF had an even score of 50% for both its training and testing data set. This suggests that it can predict half of the instances of dataset (Figure 40 – 42).

Figure 41 RF cross-validation score

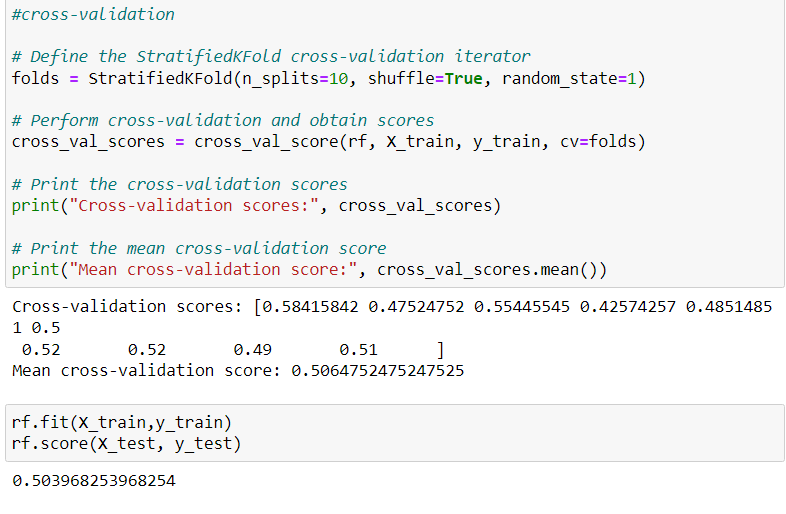


Figure 42 RF confusion matrix

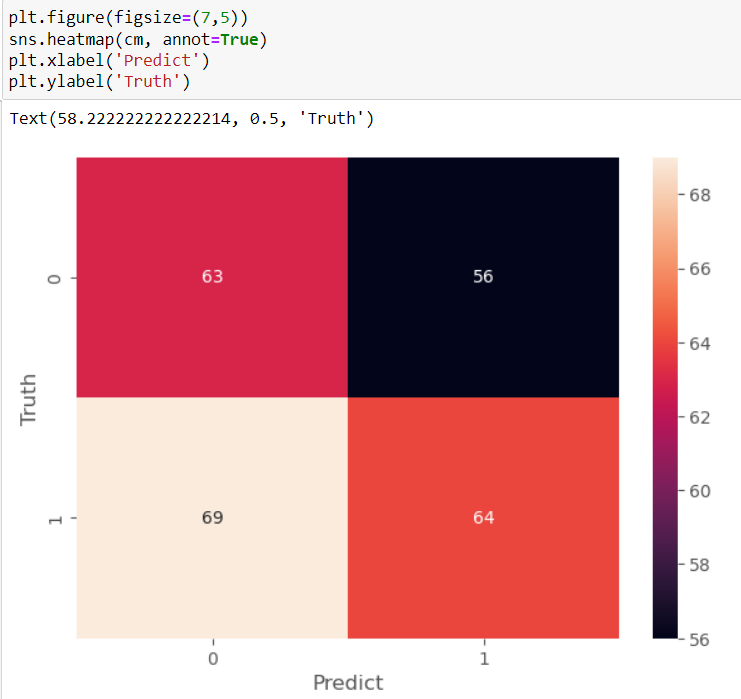
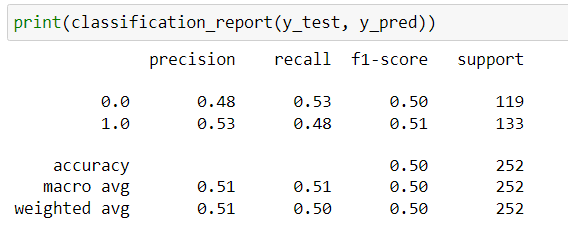


Figure 43 RF classification report



Furthermore, it performed poorly on unseen data, scoring only 34% in accuracy (Figure 43 – 44).

Figure 44 RF confusion matrix with unseen data

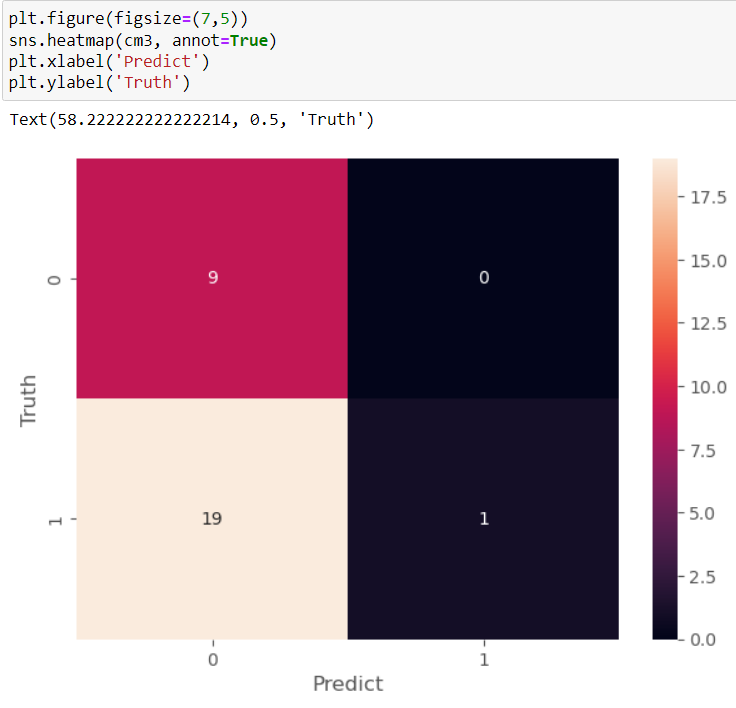
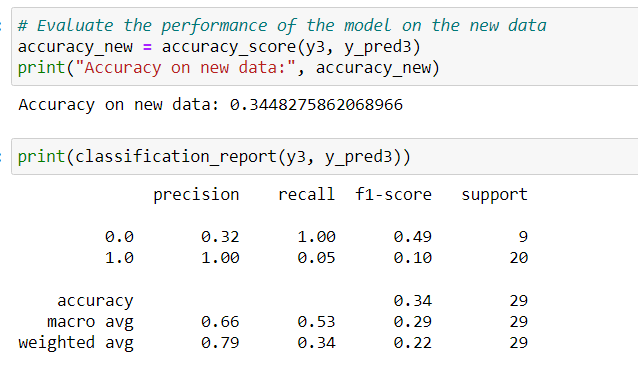


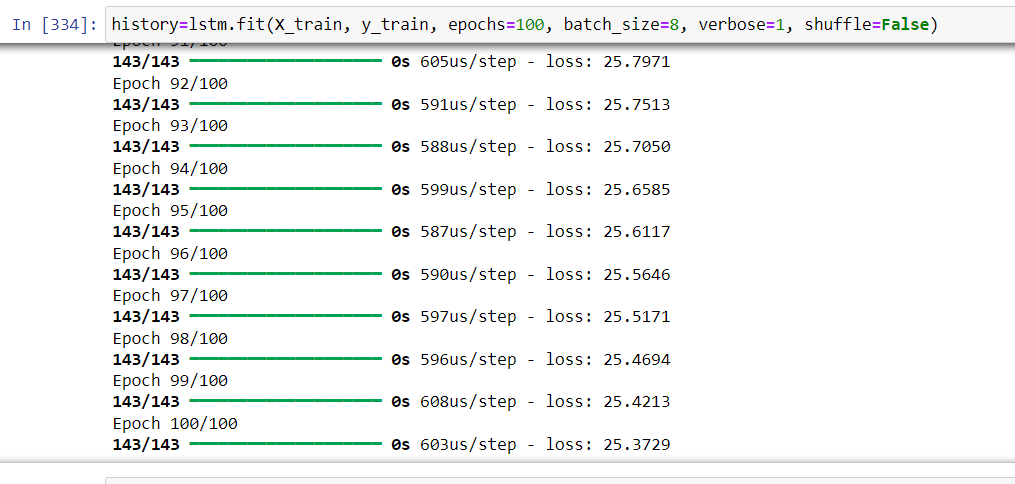
Figure 45 RF classification report with unseen data



## Long Short-Term Memory

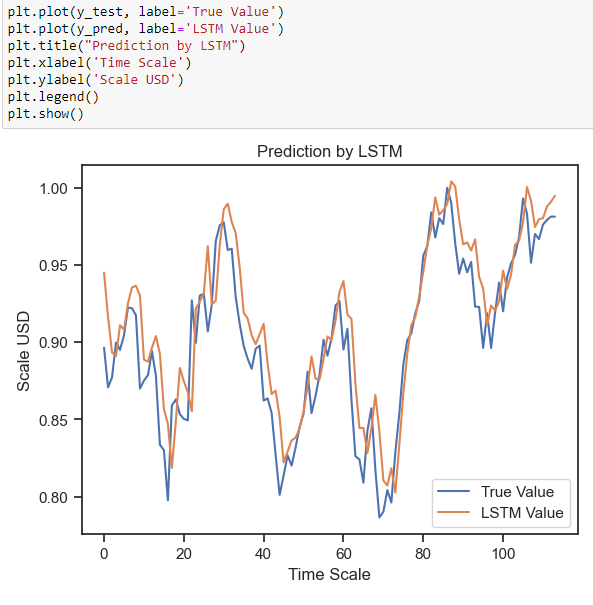
LSTM performed very well in the training and testing of its dataset. Using 100 epochs, it can be seen that the loss decreased gradually per epoch (Figure 42).

Figure 46 LSTM model fitting with 100 epochs



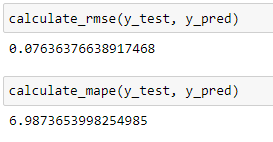
Additionally, Figure 46 illustrates how well it was able to predict the target “Adj Close” of the following day.

Figure 47 Visualised prediction of LSTM test data



Furthermore, its root MSE (RMSE) remained low, indicating a good performance of the model. Subsequently, MAPE is also done to test the data. The model yielded a 6.98% error, suggesting 94% accuracy in its predictions (Figure 47).

Figure 48 RMSE and MAPE of LSTM test data



Moreover, when tested with df2, the LSTM model’s RMSE slightly went up to 16.5, accompanied by a MAPE of 2.15%. This suggests that, on average, the absolute percentage difference between the actual and predicted value is 2.15%, indicating that the model’s accuracy, on average is approximately 97.85% (Figure 48 – 49).

Figure 49 Visualised prediction of LSTM unseen data

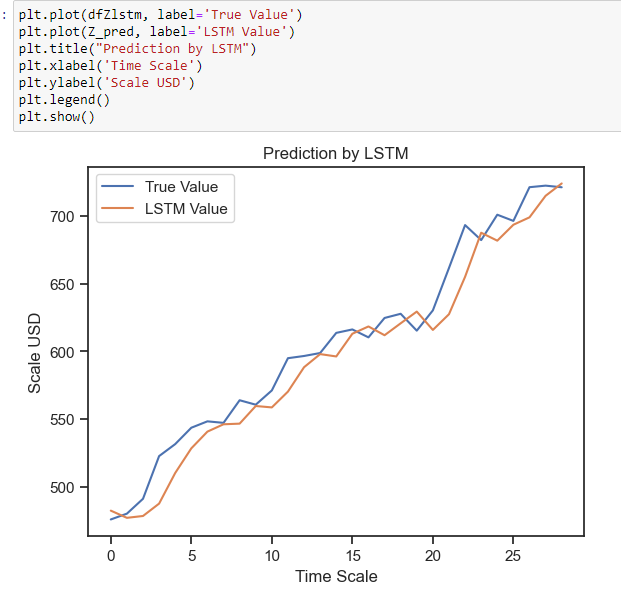
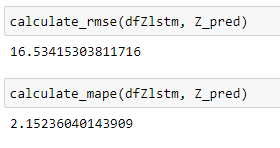


Figure 50 RMSE and MAPE of LSTM unseen data



## Test data and Unseen data results

Figure 51 Machine learning model results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 20-day rolling means LR | 50-day rolling means LR | k-NN | SVM | RF | LSTM |
| Test data Score | R²= 0.99 | R²=0.99 | Accuracy= 0.60 | Accuracy= 0.54 | Accuracy= 0.50 | MAPE= 6.98 |
| Unseen data Score | R²=0.62 | R²=0.63 | Accuracy= 0.48 | Accuracy= 0.34 | Accuracy= 0.34 | MAPE= 2.15 |

## Limitations

Due to the limited timeframe of the project hyper-parameter tuning has been omitted. Additionally, the project only uses data from a single stock, which is NVIDIA.

# Critical Reflection

## Challenges

The challenges met in this project were the time constraint, the amount of machine learning information and issues with preprocessing the data to match the model. Given a total of 6 weeks in this module, with the first half dedicated to the completion of coursework 1, I had to manage my time strictly as to meet the deadlines of each phase in the SDLC. Moreover, even though I was able to meet the planned timeframe, it did not give me extra time to do hyper-parameter tuning to improve the models, hence it was left out.

Additionally, even though the machine learning code is not lengthy in itself, I had to learn about each model’s functionality and the underlying mathematics that supports its system. I have observed that the LSTM model was the hardest to learn as it involved multiple parameters in its code.

Furthermore, I had issues with preprocessing the data as regression models have a different requirement compared to classification models.

## Successes

Despite the challenges encountered, I was able to meet the planned timeframe deadlines, learn all the information required to preprocess the data and implement the machine learning models correctly.

## Lessons

I have learned that much of the data science work involves preprocessing the data to fit the models. I figured out that in order to make predictions in stock prices, I not only needed a target data but I had to shift it to the following day. This is due to the “Close” feature being the last traded price at the end of regular market hours and regular investors will not be able to traded after that.

Additionally, I have learned about the different ways to evaluate machine learning models, each requiring a different approach. These methods include MSE and R² for linear regression, cross-validation for classification and RMSE, MAPE for the LSTM model.

## Professional and Ethical issues

The project uses yfinance API to obtain the dataframe for stock prediction. However, it is an open-source tool that uses Yahoo’s publicly available stock information. While valuable for educational and research purposes, it is best to consider relying on official APIs for professional use.

# Conclusion

In summary, the project implementation primarily involved preprocessing of the data prior to fitting it into the model. The models were trained and tested with ‘df’ and further tested with unseen data from ‘df2’. The LSTM model showed the best performance, achieving a 6.98% MAPE on the test data and 2.15% on unseen data. Conversely, both SVM and RF models showed poor performance, with an accuracy score of 34% on unseen data. It is worth noting, that the 50-day rolling mean LR scored an R² of 0.63 with unseen data, and slightly mirrored the trend of the original data.

Furthermore, critical reflection of this project revealed challenges, lessons and professional issues related to machine learning stock price prediction. Moving forward, it is recommended to conduct further studies in testing the models with larger sample sizes and across different stock tickers.

# Reference

Chen, S., & Zhou, C. (2020, December 24). *Stock Prediction Based on Genetic Algorithm Feature Selection and Long Short-Term Memory Neural Network* [Review of *Stock Prediction Based on Genetic Algorithm Feature Selection and Long Short-Term Memory Neural Network*]. Ieeexplore.ieee.org; IEEE. https://ieeexplore.ieee.org/abstract/document/9306838/metrics#metrics

Cunningham, P., & Delany, S. J. (2021). k-Nearest Neighbour Classifiers - A Tutorial. *ACM Computing Surveys*, *54*(6), 1–25. https://doi.org/10.1145/3459665

‌ GeeksforGeeks. (2018, November 13). *K-Nearest Neighbours - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/k-nearest-neighbours/

Joshi, A. V. (2022). Support Vector Machines. *Springer EBooks*, 89–99. https://doi.org/10.1007/978-3-031-12282-8\_8

Lederer, J. (2021). Linear Regression. *Springer Texts in Statistics*, 37–79. https://doi.org/10.1007/978-3-030-73792-4\_2

*Machine Learning Tutorial Python - 11 Random Forest*. (n.d.). Www.youtube.com. Retrieved March 27, 2024, from https://www.youtube.com/watch?v=ok2s1vV9XW0&list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw&index=12

*Machine Learning Tutorial Python - 2: Linear Regression Single Variable*. (n.d.). Www.youtube.com. Retrieved March 27, 2024, from <https://www.youtube.com/watch?v=8jazNUpO3lQ&list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw&index=2>

*OWASP Machine Learning Security Top Ten 2023 | ML02:2023 Data Poisoning Attack | OWASP Foundation*. (n.d.). Owasp.org. https://owasp.org/www-project-machine-learning-security-top-10/docs/ML02\_2023-Data\_Poisoning\_Attack

*Principles for the security of machine learning*. (2022). https://www.ncsc.gov.uk/files/Principles-for-the-security-of-machine-learning.pdf

Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal: Promoting Communications on Statistics and Stata*, *20*(1), 3–29. https://doi.org/10.1177/1536867x20909688

‌ *sklearn.metrics.r2\_score*. (n.d.). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2\_score.html#sklearn.metrics.r2\_score

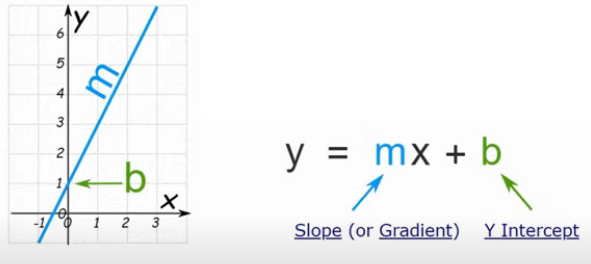
*13. Evaluation — Data Science 0.1 documentation*. (n.d.). Python-Data-Science.readthedocs.io. Retrieved March 28, 2024, from https://python-data-science.readthedocs.io/en/latest/evaluation.html

‌

# Appendix

## Linear regression

Figure 52 Linear Regression

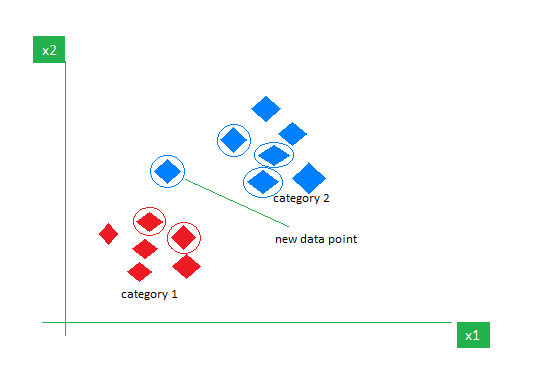


(Machine Learning Tutorial Python - 2: Linear Regression Single Variable, n.d.)

Linear regression relates the independent variable and the dependent variables. This model uses linear equation: y = mx +b, where m is the slope and b is the intercept. Assuming there is linearity in the relationship, it makes the model computationally and mathematically convenient. However, it can also be used effectively with non linear relationships by expanding the independent variable (x) such as x², x³ and so on. It becomes easier to use with increasing sample size but becomes more challenging with increased noise, which is the random variation in the relationship of dependent and independent variables (Lederer, 2021).

## K – Nearest Neighbour

Figure 53 k-NN algorithm visualisation



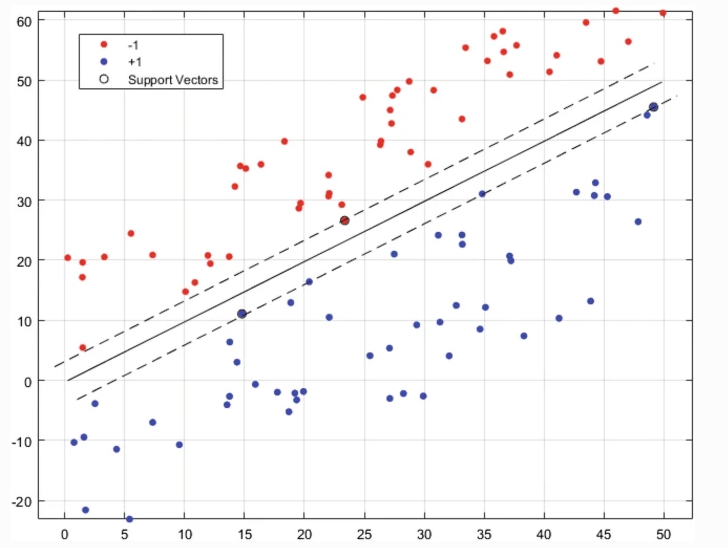
(GeeksforGeeks, 2018)

K – Nearest Neighbour is the classification of data based on their nearest neighbours. It involves 2 stages, first is determining the nearest neighbour and the second is determining the class using that neighbour. If the neighbours have different classes and are near to each other, distance weighted voting or simple majority voting is used to resolve the issue. Euclidean and Manhattan distances are the most popular k-NN distance measure. With larger number of decisions, hyper-parameter tuning and model selection is an important factor in this model. Moreover, using cross-validation is an invaluable strategy to evaluate the model’s effectiveness (Cunningham & Delany, 2021).

Additionally, k-NN is an easy to implement and understand model and should be considered in classification problems. However, because k-NN’s work is done in run time, problems can be encountered if the training set is too large. Furthermore, the model is sensitive to redundant or irrelevant features since all the features contribute to the similarity and therefore to the classification (Cunningham & Delany, 2021).

## Support Vector Machine

Figure 54 SVM on separable data

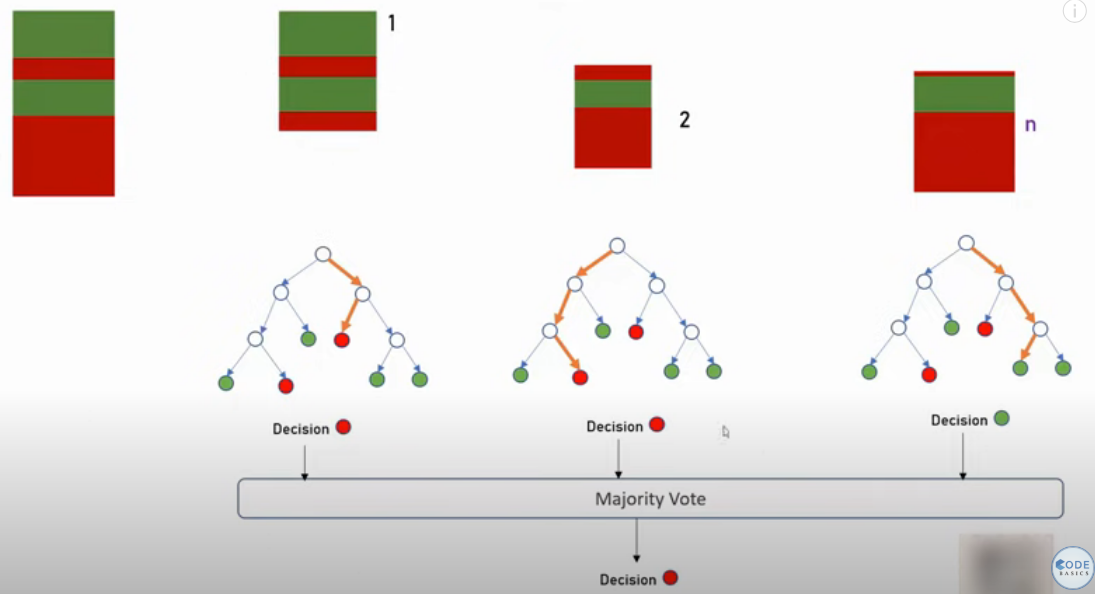


(Joshi, 2022)

Support vector machine (SVM) separates two classes using select number of data points called support vectors. A hyperplane is used to optimise the separation between the two classes. Additionally, even if the classes are not fully separable the algorithm can still locate the optimal hyperplane and support vectors. The advantage of this algorithm is the reduction of amount if information needed for classification. Moreover, this approach effectively balances performance optimisation with reduction of overfitting (Joshi, 2022).

## Random Forest

Figure 55 Random Forest algorithm



(Machine Learning Tutorial Python - 11 Random Forest, n.d.)

The random forest algorithm is a tree-based ensemble learning method where decision trees act as the foundation of the random forest model. A decision tree recursively partitions a given dataset into subsets based on certain criteria, such as entropy. It continuous this process until a condition is met. Random forest can be applied to both classification and regression tasks (Schonlau & Zou, 2020).

Using random forest is advantageous in nonlinear relationships and out performs linear regression models. They are well suited to datasets that are medium to large range. However, it is susceptible to overfitting as it can capture outliers and noise in a dataset, leading to perform generalisation performance poorly on new unseen data (Schonlau & Zou, 2020).

## Long Short – Term Memory

Figure 56 LSTM neuron

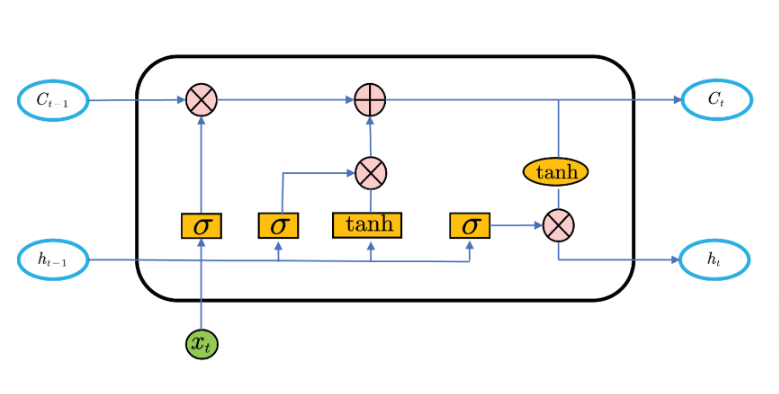


Figure 57 (Chen & Zhou, 2020)

Recurrent neural network (RNN) is a recursive neural network which uses sequence data as input. Subsequently, it executes recursion in sequence evolution direction wherein a chain connects all nodes. Due to its ability to remember previous time steps, it can perform well with short sequence models.

Long Short-Term Memory is a deep RNN which has greater memory capacity due to its special gate mechanism in the neural unit structure. Additionally, it solves the vanishing gradient issue cause by too much input sequence in the learning process.

Figure 57 illustrates the LSTM neuron. All information is saved pre-time step wherein each neuron is controlled by the input, output and forgetting gate (Chen & Zhou, 2020).

## Heuristic Usability

The following usability criteria has been completed by the author regarding the project’s usability.

Figure 58 Heuristic evaluation table

|  |  |  |
| --- | --- | --- |
| Usability Heuristic | Severity 1 - 10 | Evaluation |
| Visibility of system status | 5 | Each model’s output is clear and visible. |
| Match between system and the real world | 10 | The data used and visualised is taken from yahoo finance’s current stock prices. |
| User control and freedom | 10 | The codes implemented, makes it easy to manipulate and use on other stock tickers. |
| Consistency and standards | 10 | The used of OHLC chart has been consistent all through out the models. |
| Error prevention | 1 | The user can not use use data from different columns when querying under a specific name. However, users can still go around this by using different querying commands such ‘.iloc’. |
| Recognition rather than recall | 7 | The shape and info of the data frame can easily be known by using .shape, .info and .head(). |
| Flexibility and efficiency of use | 7 | The models are easy to use and visualise once preprocessing is done. |
| Aesthetic and minimalist design | 5 | The project is minimalistic as it is all in jupyter notebook but could benefit from having a web app to make it more aesthetically pleasing. |
| Help users recognise, diagnose, and recover from errors | 3 | As the project has a limited timeframe, error prevention and help for diagnosing or locating where the problem stems in the code has not been applied. However, the original data frame is saved and stored in case of file corruption. |
| Help and documentation | 8 | Proper documentation has been kept, regarding the dataframe and model’s information. |