IT1244 Report

Jamekorn Likitwattananurak (A0287346R), Fransiska Aurelia Giovanny Susanto (A0276096R), Muhammad Salman Al Farisi (A0276128Y), Muhammad Amadeus Ihsan Candra (A0276114J)

Introduction

Stocks are financial assets with high risk and high return (Zhong-y, 2014). For years, people have tried to mitigate stock market risk through prediction techniques. However, stock market predictability remains debated. The random walk theory posits that stock prices cannot be predicted using historical data (Usmani, 2016), while the chartist theory argues that price trends can indeed be forecasted (Usmani, 2016).

Since the 1980s, chartists like Keim and Stambaugh (1986) demonstrated successful stock market predictions using classic statistical methods, such as weighted least squares. As technology advanced, Sezer et al. (2020) showed that traditional machine learning often outperforms statistical methods, and, more recently, deep learning has surpassed traditional machine learning.

Inspired by these insights and the work of Vijha et al. (2019), our group selected a diverse range of models. To assess each model's effectiveness, we first established a random walk baseline. Any approach that surpasses this baseline will indicate that, for our dataset, the chosen prediction method is preferable. We pursued three analysis paths: classic statistical methods (ARIMA, Simple Moving Average, and linear regression), traditional machine learning models (lasso regression, random forest, XGBoost, CATBoost), and deep learning models (Single Layer Perceptron, ANN, GRU, RNN, LSTM, and MFNN).

Literature Review

Long et al. (2018) noted that stock price prediction involves more factors than other tasks which makes feature selection and engineering essential to enhancing dataset quality for model inputs.

Our feature selection methods are based on Anwar et al. (2020), who implemented a multi-filter feature selection (MFFS) method for stock prediction. Anwar et al. argued that single feature-selection results in suboptimal feature sets. They combined Random Forest, SVM, and Logistic Regression and refined top features with exemplar-based clustering, using linear correlation and information gain to eliminate redundancy, our approach substitutes logistic regression with lasso regression, aligning with our regression-focused goals. Additionally, inspired by Long et al. (2018), we experimented with a fully deep learning-based feature selection and prediction method using a multi-filter neural network (MFNN) with convolutional and recurrent layers, as Long et al. found that MFNN outperformed traditional and machine learning models.

Dataset

The dataset consists of two files covering S&P 500 companies from October 2020 to July 2022. The first file includes the company's symbol, GICS sector, headquarters location, and founding year. The second file provides each company's symbol, stock close prices, trade volumes, 14 news volume categories, and exchange dates. All columns, except dates and symbols, are used as features to predict the close price based on data from the past five dates rather than the current date.

Data Cleaning – First, we merged the files using the 'Symbol' column. We checked for missing values, removing two rows with missing dates, close prices, and volumes. For the 522 rows with missing news volumes, we replaced these with 0, assuming no news. Duplicates were also removed.

Train-Test Data Preparation – The data was split into train and test sets in an 80-20 ratio. Dates and symbols were excluded as they do not contribute to close price prediction, as were categorical columns (GICS sector, headquarters location, founded year) to avoid complications with one-hot encoding. We created lagged features based on each company's past five dates. As a result, the first five dates for each company contained NaN values, so we removed these rows. The current date's features, except the close price, were also removed. column into test data and the rest of the columns as train data. Lastly, we standardize the data.

Feature Selection

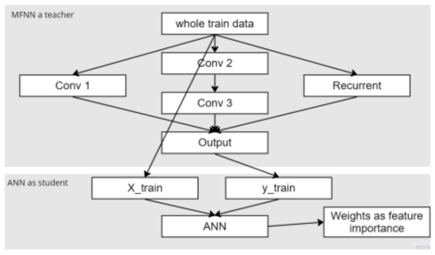


Figure 1. *MFNN structure*

Following Anwar et al. (2020), we applied the MFFS method for feature selection. Initially, we ranked features based on intrinsic importance (weights) from lasso regression, support vector regression (SVR), and random forest (RF), selecting the top 20 features from each. We chose these methods due to their different data assumptions: lasso regression assumes linearity, SVR handles non-linearity, and RF captures complex relationships with its tree-based structure. We then combined features from all methods and applied affinity propagation to further filter them. Affinity propagation selects features that best represent the data based on high similarity indices with other features, using linear correlation for linearity and information gain for non-linearity. The resulting features form the final MFFS selection.

Inspired by Long et al. (2018), we also used MFNN for feature selection. MFNN processes the training data through three paths: a single convolutional network, a stacked convolutional network, and a recurrent network. The predicted close price from MFNN is used as the target (y-value) to train a simple ANN with two dense layers (Mirzaei, 2019). The weights from the first ANN layer provide feature importance, from which we select the top 20 features.

In summary, we obtain seven feature combinations: all features, close price only, SVR, RF, Lasso, MFFS, and MFNN. Each feature combination will be evaluated to determine the best-performing set for our dataset.

Visualization - We visualized all features to close the price using a scatter plot. However, we did not gain any meaningful insight that affected our feature selection. In our opinion, objective data-driven feature selection techniques are more reliable than eye-relying techniques.

Method

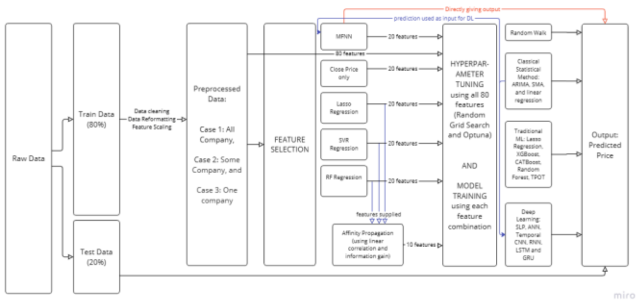

Directly giving output
prediction used as input for DL
Raw Data
Train Data (80%)
Test Data (20%)
Data cleaning
Data Reformatting
Feature Scaling
Preprocessed Data:
Case 1: All Company,
Case 2: Some Company, and
Case 3: One company
FEATURE SELECTION
MFNN
80 features
Close Price only
Lasso Regression
SVR Regression
RF Regression
20 features
features supplied
Affinity Propagation (using linear correlation and information gain)
10 features
HYPERPARAMETER TUNING using all 80 features (Random Grid Search and Optuna)
AND
MODEL TRAINING using each feature combination
Random Walk
Classical Statistical Method: ARIMA, SMA, and linear regression
Traditional ML: Lasso Regression, XGBoost, CATBoost, Random Forest, TPOT
Deep Learning: SLP, ANN, Temporal CNN, RNN, LSTM and GRU
Output: Predicted Price
miro

Figure 2. *Model Workflow*

In this section, we address the challenge of predicting a stock's closing price using data from the past five days. The figure above illustrates our model workflow. After performing train-test splitting and data preprocessing, we organise the data into three cases: using all companies, using some companies absent from the test set, and using data from only the company targeted for prediction in the test set. We then apply feature selection, creating seven different feature combinations for model training.

Before training, we tune hyperparameters using all features to simplify the process. We apply expanding window time-series split with four splits and use two tuning methods. The first is Random Grid Search, exploring 50 random hyperparameter combinations, and the second is Optuna, which frames tuning as an optimisation problem to minimise MSE.

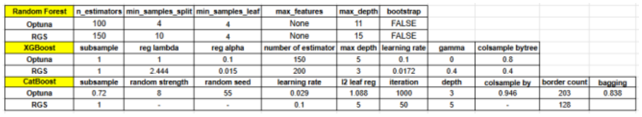
With the best hyperparameters identified, we train each model to predict stock prices. Our models fall into three categories: classic statistical models, traditional machine learning models, and deep learning models. Following this, we will justify each model choice and outline the setup for each.

Random Walk - We use random walk as a baseline to test the claim made by the chartist that the stock market is predictable using historical data (Usmani, 2016). The prediction algorithm for a random walk is the stock's last date's price plus a gaussian noise with mean = 0 and standard deviation = one percent of the standard deviation of y_train.

Classical Statistical Method - According to Anwar et al. (2020), complex models are not always significantly better than simple models as complex models depend on the optimal selection of hyperparameters and input features. It is demonstrated in Adebiyi et al. (2014) that the stock prediction difference between the two is insignificant statistically. Hence, we decided to include simple models such as SMA, ARIMA, and linear regression in our project as a baseline.

We hyperparameter-tuned ARIMA using AUTO ARIMA and got ARIMA (0, 1, 0) as the best model, which means that the prediction is only the previous date's price. The prediction algorithm for SMA is the simple mean of the previous 5 date's stock price. As linear regression has no hyperparameters, we directly train it and use it for prediction. It assigns weight to every feature, which will then be multiplied by the testing data features and summed together to be the prediction.

Traditional Machine Learning



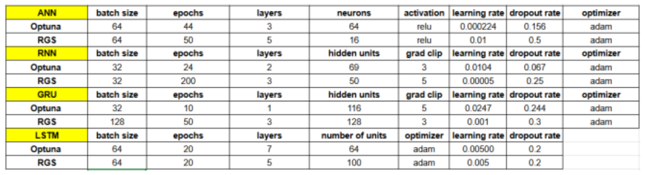
Random Forest	n_estimators	min_samples_split	min_samples_leaf	max_features	max_depth	bootstrap				
Optuna	100	4	4	None	11	FALSE				
RGS	150	10	4	None	15	FALSE				
XGBoost	subsample	reg lambda	reg alpha	number of estimator	max depth	learning rate	gamma	colsample bytree		
Optuna	1	1	0.1	150	5	0.1	0	0.8		
RGS	1	2.444	0.015	200	3	0.0172	0.4	0.4		
CatBoost	subsample	random strength	random seed	learning rate	l2 leaf reg	iteration	depth	colsample by	border count	bagging
Optuna	0.72	8	55	0.029	1.088	1000	3	0.946	203	0.838
RGS	1	-	-	0.1	5	50	5	-	128	

Figure 3. *ML Models Hyperparameters*

Our selection of traditional machine learning methods is based on Vijha et al. (2019), who mentioned that traditional machine learning such as random forest can be employed for stock price prediction. We expanded our model selection by researching ChatGPT, which advised us to use lasso regression, XGBoost, and CatBoost. The hyperparameter tuning method used is the one explained in the second paragraph of the "Method" section.

Random forest consists of multiple decision trees, each trained using a randomly selected subset. The close price prediction is the average of individual tree predictions. Its strengths that are relevant for stock price prediction are its ability to avoid underfitting due to flexibility to handle nonlinear relationships and to avoid overfitting and noise effects because of using multiple trees. Lasso Regression is just linear regression with a penalty term, alpha, which penalizes high coefficients, reducing overfitting while increasing the risk of underfitting. We obtain the hyperparameter alpha=0.0625 from LassoCV. XGBoost is a tree-based model with extreme gradient boosting, where the tree is trained to correct the previous tree version to minimize the loss function while also penalizing too complex trees to prevent overfitting. Catboost, similar to XGboost, is also a tree-based model, where each new tree boosts the gradients and adds on to the previous trees. And similar to other models in traditional ML, it utilizes many weak learners (simple models) to build a robust model. Catboost is proficient in handling categorical data by using ordered boosting and prevents information leakage. This way of handling categorical data is also beneficial in computational efficiency, as traditional one-hot encoding is computationally expensive. Our group uses CatBoost due to its ability to handle categorical data more Seffectively as well as potentially regularizing the model better by preventing data leakage.

Deep Learning



ANN	batch size	epochs	layers	neurons	activation	learning rate	dropout rate	optimizer
Optuna	64	44	3	64	relu	0.000224	0.156	adam
RGS	64	50	5	16	relu	0.01	0.5	adam
RNN	batch size	epochs	layers	hidden units	grad clip	learning rate	dropout rate	optimizer
Optuna	32	24	2	69	3	0.0104	0.067	adam
RGS	32	200	3	50	5	0.00005	0.25	adam
GRU	batch size	epochs	layers	hidden units	grad clip	learning rate	dropout rate	optimizer
Optuna	32	10	1	116	5	0.0247	0.244	adam
RGS	128	50	3	128	3	0.001	0.3	adam
LSTM	batch size	epochs	layers	number of units	optimizer	learning rate	dropout rate	
Optuna	64	20	7	64	adam	0.00500	0.2	
RGS	64	20	5	100	adam	0.005	0.2	

Figure 4. *DL Models Hyperparameters*

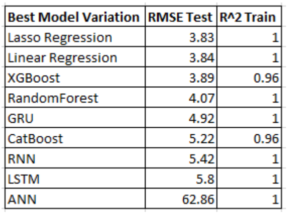
Inspired by Usmani et al. (2016), we implemented artificial neural networks (ANN). Selvin et al. (2017) guided us towards recurrent neural networks (RNN) and long short-term memory (LSTM), while GRU was selected based on ChatGPT research, and Long et al. (2018) influenced our choice of multi-filter neural networks (MFNN). Building on Usmani et al. (2016), we used predictions from traditional statistical and machine learning methods as inputs for the deep learning models. A detailed explanation of hyperparameter tuning follows. Recurrent neural networks (RNNs) outperform ANNs by retaining information from previous steps through feedback connections. RNNs process data sequentially, where each output depends on both the current input and past outputs. With feedback loops, RNNs leverage short-term historical data effectively, though less extensively than LSTMs. RNN's success would indicate the utility of recent past data for current stock predictions. Long Short-Term Memory (LSTM) builds on RNN by using three gates—forget, input, and output—to control information flow. The forget gate discards unnecessary data, the input gate selects new information to integrate, and the output gate determines which parts of the hidden state influence future predictions. LSTM also has cell states, allowing it to retain data over longer periods, making it a preferred choice for time series tasks like stock prediction. We selected LSTM for its capacity to use distant past data for stock prices, though our 5-day focus may limit its effectiveness.

Gated Recurrent Units (GRUs) are similar to LSTM but simplify the gating process. GRU's update gate manages incoming information, while the reset gate determines data to discard, resembling LSTM's forget and input gates. We chose GRU as a more efficient alternative to LSTM.

Results and Discussion

Deep learning's poor performance in stock prediction is likely due to the noisy, unpredictable nature of stock data, which often leads to overfitting. Simpler models like ARIMA and Random Walk handle this volatility better due to their robustness. Additionally, deep learning's need for large datasets and precise tuning can be limited by practical constraints, making simpler models more effective here.

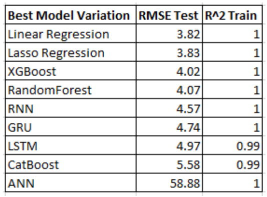
All Companies in Train and Test



Best Model Variation	RMSE Test	R^2 Train
Lasso Regression	3.83	1
Linear Regression	3.84	1
XGBoost	3.89	0.96
RandomForest	4.07	1
GRU	4.92	1
CatBoost	5.22	0.96
RNN	5.42	1
LSTM	5.8	1
ANN	62.86	1

The best-performing model is Lasso Regression, using the last five days' Close Price as features. Among deep learning models, GRU performs best, followed by other models most likely because of overfitting. Lasso's regularization enables it to focus on key features, effectively capturing recent price patterns. GRU also captures sequential patterns efficiently, learning recent trends and reducing prediction error without excessive complexity.

Some Companies in Train and Others in Test



Best Model Variation	RMSE Test	R^2 Train
Linear Regression	3.82	1
Lasso Regression	3.83	1
XGBoost	4.02	1
RandomForest	4.07	1
RNN	4.57	1
GRU	4.74	1
LSTM	4.97	0.99
CatBoost	5.58	0.99
ANN	58.88	1

The best technique is Linear Regression Model with close price as features. Overall, classic statistical methods perform better than traditional machine learning, which performs better than deep learning. Linear regression might perform better because of simplicity and reduced overfitting. Close price as the best features means that the other features possibly add more noise than explanatory power. From the 2 table above, the high R^2 values indicate that the models fit the training data almost perfectly, capturing nearly all variance. This suggests a potential risk of overfitting, where the models may struggle to generalize well to new data, as seen in the higher test RMSE values for some models

One Company in Train and Test

The best-performing model is ARIMA, using the previous day's close price as the prediction, with an RMSE of 3.66 and MAE of 2.99. Random Walk ranks second, with an RMSE of 3.69 and MAE of 3.01, closely following ARIMA. The next 12 best models are linear and lasso regressions, all classic statistical methods, with RMSE values ranging from 3.69 to 3.93. Traditional machine learning generally outperforms deep learning, with RMSE values between 4.17 and 7.93. Among deep learning models, GRU performs best with an RMSE of 4.625, followed by RNN at 5.51, LSTM at 7.159, and finally ANN with 45.97. Notably, ARIMA's top result aligns closely with the Random Walk, as both its autoregressive and moving average terms are zero, meaning it simply returns the previous day's close price. This suggests that, for our data, historical predictions cannot improve on Random Walk. Deep learning's poor performance in stock prediction is likely due to several factors. Stock market data is noisy, making it hard for deep learning models to find meaningful patterns. Although suited to complex, time-dependent data, deep learning models often overfit limited, unpredictable datasets like stock prices. Simpler models like ARIMA and Random Walk manage these challenges better due to their robustness. Deep learning also needs large datasets and careful hyperparameter tuning, but practical limits on iterations hinder optimization. Given stock data's non-stationary nature, simpler models may be more effective for this task. Stock market prediction merges quantitative analysis with external socio-political and economic factors. Models capture historical data trends well but often miss unexpected external shifts, where human analysts excel. However, models process vast data objectively, uncovering patterns humans may miss, making a blend of model precision and human judgment ideal. Our data does not confirm that historical data can predict stock prices, as Random Walk performed better. However, past research suggests this result is not universal. if prediction is feasible, large institutions with computational power could reduce human traders, impacting jobs and potentially widening inequality. High speculation may also risk economic bubbles, and an economy overly reliant on stock gains could face inflation due to reduced production and increased

References

1. Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction. *Journal of Applied Mathematics*, *2014*, 1–7. https://doi.org/10.1155/2014/614342

2. Keim, D. B., & Stambaugh, R. F. (1986). Predicting returns in the stock and bond markets. *Journal of Financial Economics*, *17*(2), 357–390. https://doi.org/10.1016/0304-405x(86)90070-x

3. Long, W., Lu, Z., & Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, *164*, 163–173. https://doi.org/10.1016/j.knosys.2018.10.034

4. Mirzaei, A. (2019, April 8). *How to use Deep-Learning for Feature-Selection, Python, Keras*. Medium. https://medium.com/@a.mirzaei69/how-to-use-deep-learning-for-feature-selection-python-keras-24a68bef1e33

5. Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. https://doi.org/10.1109/icacci.2017.8126078

6. Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181. https://doi.org/10.1016/j.asoc.2020.106181

7. Usmani, M., Adil, S. H., Raza, K., & Ali, S. S. A. (2016). Stock market prediction using machine learning techniques. *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*. https://doi.org/10.1109/iccoins.2016.7783235

8. Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A. (2020). Stock Closing Price Prediction using Machine Learning Techniques. *Procedia Computer Science*, *167*(167), 599–606. https://doi.org/10.1016/j.procs.2020.03.326

9. Zhong-y, X. (2014). Empirical Research on Market Risk and Return of Stock Portfolio. *Journal of Guizhou University of Finance and Economics*.