CSC413H1 - Final Project - Time Series Analysis

Amina Shmanova **Ashwin Karthikeyan** **Varun Girish Vaze**

Abstract

Predicting stock market trends remains an ongoing challenge, with various machine learning (ML) models attempting to address the problem. This study investigates the potential of three ML models - k-nearest neighbors (k-NN), Long Short-Term Memory (LSTM), and Transformers - in predicting stock prices, comparing their performance against simple regression algorithms, random walk, ARIMA, and an ensemble of kNN and ARIMA. Aligning with the Efficient Market Hypothesis, the results indicate that even advanced non-linear models struggle to consistently achieve excess returns, as the arrival of new information is unpredictable. The ensemble of kNN and ARIMA demonstrated the best performance on the 20MICRONS dataset, further emphasizing that more complex models such as LSTMs and Transformers are unable to outperform relatively simple models, and successful predictions are likely achieved by chance.

1 Introduction:

The domain of predicting stock market trends has become a popular area of research. Despite numerous attempts to predict stock prices through various studies, no model has consistently demonstrated accurate stock market forecasting. Analysing the behaviour of a stock market is a challenging task since stocks often introduce volatility, i.e. random fluctuations over time.

Machine Learning (ML) models are often used to make predictions through technical aspects of stock price movement, such as trends, patterns, peak and lowest prices (Saini & Sharma, 2019). Regardless of the data type or the observed phenomenon's specifics, network models represent a more abstract approach than traditional methods. Unlike manually done analysis, neural networks do not impose restrictions on the nature of input information and appear to be more adaptive and time-efficient (Bugorskij & Sergienko, 2008).

However, this approach has been criticized for ignoring subjective factors, such as sentiments expressed on social media and in news sources (Saini & Sharma, 2019), as well as economic and social factors (Thakkar & Chaudhari, 2021). Despite these drawbacks, ML techniques continue to be a favored technique for market forecasting, leading to numerous models and publications released annually. As such, different methods have been proposed with varying degrees of success.

The aim of this study was to investigate the potential of using machine learning methods for predicting stock prices. Three ML models, namely k-nearest neighbors (k-NN), Long Short-Term Memory (LSTM), and Transformers, were selected and compared based on their predictive performance. Additionally, the performance of these models was compared against that of simple regression algorithms and random walk. The study was guided by the hypothesis that, in the absence of any underlying patterns in the stock market, the models would exhibit a few instances of successful predictions by chance, which could not be replicated. This hypothesis aligns with the Efficient Market Hypothesis (Fama, 1970), which proposes that prices are solely driven by new information. Therefore, since the arrival of new information is unpredictable, market prices appear to be generated randomly. Therefore, it is not possible to consistently achieve excess returns using available information, and even advanced non-linear models such as NN would not be able to

Preprint. Under review.

effectively execute this task.

2 Methods

In this section, we provide a concise overview of the chosen machine learning techniques and associated parameters.

2.1 Machine Learning Models

2.1.1 k-Nearest Neighbour.

The nearest neighbor (NN) classifier is one of the simplest machine learning algorithms that classifies a new pattern based on its similarity to the available training patterns (Ersan et al., 2020). The similarity between the new pattern and training patterns is measured using the Euclidean distance, and the class of the nearest training pattern is assigned to the new pattern. The k-nearest neighbor (k-NN) classifier is similar to the NN, but assigns a class based on the most common class among its k-nearest neighbors. k-NN was chosen as it is efficient for analyzing many stocks each day, and as a baseline for comparison with MLP neural networks, which often yield better classification accuracy.

2.2 Auto-Regressive Integrated Moving Average (ARIMA)

Auto-Regressive Integrated Moving Average (ARIMA) model is a commonly used statistical model for forecasting in time-series data (Hayes, 2022). They are widely used due to their ability to predict future values in the short-term quite accurately. The ARIMA model is a combination of 3 different components: an autoregressive model, that assumes that current values are affected by past values, a moving average model, that assumes current values depend on the error terms of past values, and a differencing component, that makes a time-series stationary (constant mean in the long-term) (Maklin, 2019).

2.2.1 LSTM.

LSTM is a type of RNN model that maintains a memory vector c_t in addition to a hidden state vector h_t, allowing the model to additively modify memory contents through gating mechanisms. This design enables gradients to flow backward through time uninterrupted for long periods of time. As such, the model is able to remember large sequences of data, which is important for stock market predictions. Compared to kNN and ARIMA models, papers on LSTMs for stock market predictions have demonstrated superior performance (Moghara & Hamiche, 2020).

In this project, we utilized the word embeddings model provided by Loye (2023). As the model was not originally designed for time series data, certain modifications were made. These included adjusting weight initialization and reshaping the hidden and cell states to accommodate time series data. Data extraction and visualization techniques were adopted from Nayak (2020).

Our LSTM model consisted of a single LSTM layer, a dropout layer, and followed by two activation functions: ReLU and Sigmoid. The model's parameters and their respective descriptions can be found in Table 1.

2.2.2 Transformer

First introduced in the paper "Attention is All You Need" by Vaswani et al. (2017), the MultiHeadAttention transformer is a sophisticated deep learning architecture that has exhibited remarkable efficacy in a variety of tasks, including time series prediction. We expect the model to perform well with time series prediction because it can effectively capture long-range dependencies and complex relationships within the data by processing input sequences in parallel using multiple attention heads. An expected challenge for this model stems from the considerable quantity of trainable parameters within it. This may result in overfitting, especially when confronted with limited or noisy data sets.

The table (Table 2) below gives a precise summary of the model architecture used for our time series prediction task. The model performs the following task: On the input - 'lookback' number

Table 1: LSTM parameters

Parameters		
Name	Description	Size
embedding_dim	The number of expected features in the input	1
hidden_size	Number of hidden features	512
n_layers	Number of recurrent layers	2
bias	Bias weight	0
drop_prob	Dropout Layer	0.5
batch size	Batch size	4
epochs	Number of epochs	1000
lr	Learning rate	0.001

of time seires entries, the model predicts the "Close" or "OT" for the next time series entry. Note that the number of parameters are not listed. This is because the parameters vary with hyperparameters like lookback, embedding dimension, and feed forward dimensions.

Table 2: Transformer Layers

Transformer Layers in Top-Down Order
Input Layer
TransformerBlock
SlicingOpLambda
Dense
Dropout
Dense

2.3 Datasets

In this project, we utilized two distinct datasets. The first one, referred to as the **Exchange** dataset, was obtained from Liu et al. (2022) and consisted of panel data on daily exchange rates from 8 different countries spanning from 1990 to 2016.

The total number of entries after cleaning: 7584

In order to address the challenges associated with processing long sequences in our RNN model, we employed a smaller dataset from Nayak (2020) focusing on the Indian Stock Market. Specifically, we used the **20MICRONS** dataset, which covers a one-year period at a daily sampling rate and contains open-high-low-close index levels of the 20 Microns company. Additionally, the RNN model was evaluated on various other Indian companies, such as 5Paisa and SBI.

The total number of entries after cleaning: 203

2.4 Performance Measures

The forecasting performance of each of the model was evaluated across four different metrics

2.4.1 Mean Square Error Loss (MSE)

MSE loss measures the mean squared error (squared L2 norm) between each element in the input x and target y. The loss can be represented by formula:

$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_n - y_n)^2$

2.4.2 Directional Symmetry (DS).

The directional symmetry (DS) measure determines the fraction of correctly predicted changes in the direction of the rate of change (ROC) from t_{i-1} to t_i from the total number of predictions as defined:

$DS = \frac{100}{n} \sum_{t=1}^{n} d_t$ with $dt = 1$ for $(y_t - y_{t-1})(\hat{y}_t - \hat{y}_{t-1}) \geq 0$ and 0 otherwise

2.4.3 Spearman Correlation (ρ)

To estimate the correlation between predicted values and true values, we utilized a Spearman rank correlation:

$\rho = \frac{cov(R(X)R(Y))}{\sigma_{R(X)}\sigma_{R(Y)}}$

2.4.4 Grid Search

To evaluate the impact of hyperparameters and tune them accordingly we performed the Grid Search on respective data. Grid search is a parameter tuning technique that aims to determine the optimal values of hyperparameters. This method involves an exhaustive search conducted on a predefined set of parameter values for a given model.

3 Analysis

To analyse the performance of these models, we train them on the Exchange and 20microns datasets. The implementation and analysis of these models are given below.

3.1 kNN

We implemented kNN to forecast future value. The 20microns dataset is modified to consist of datapoints with 4 features: Open, Close, High and low and their targets which corresponds to the next day's Closing price. The Exchange dataset is, instead, modified to consist of 4 features: 0, 1, 3 and OT. We kept the maximum number of features to be considered to 4, so that the model does not suffer from the Curse of Dimensionality resulting from a high number of features. The choice of these 4 features for either dataset was taken after trying multiple sets of features and taking the set of features that gave the least MSE loss. All rows containing missing values are dropped. The dataset is then split into test data, which consists of the last datapoint of the csv, and the train data, which consists of every other datapoint except the last datapoint. This is done to check whether this model can correctly forecast the future value, given the test data. When making predictions, the model chooses the k-nearest training datapoints to the test datapoint.

To find the optimal value of k, the model was trained on values of k ranging from 2 to 5 and selecting the model with the least loss. The optimal k value was found to be 3. For the 20microns dataset, the MSE loss is 0.000278 and the MAE loss is 0.0166667 and for the Exchange dataset, the MSE loss is 3.35×10^{-6} and the MAE loss is 0.001833. The great losses obtained could be because the datasets don't have much variance, making the losses smaller.

3.2 ARIMA

Both the 20microns and Exchange datasets are modified to consist of just the Closing price to model it as a time series. All rows containing missing values are dropped. The last datapoint is taken as the test datapoint to check whether the model correctly forecasts the last value by using all other datapoints for training.

The p,d,q parameters for the model were taken to be 2,1,1, respectively. For the 20microns dataset, the MSE loss is 0.000312 and the MAE loss is 0.017666 and for the Exchange dataset, the MSE loss is 7.4436×10^{-17} and the MAE loss is 8.6276×10^{-9}. As mentioned above, ARIMA models are known to be great at short-term forecasting. Hence, the impressive loss achieved by this model could be because it has to predict a single datapoint in the future.

We made an ensemble with the kNN and ARIMA models. We averaged the predictions of the two models and computed MSE and MAE losses which were 2.4958×10^{-7} and 0.0005, respectively for the 20microns dataset and 8.3966×10^{-7} and 0.0009, respectively for the Exchange dataset.

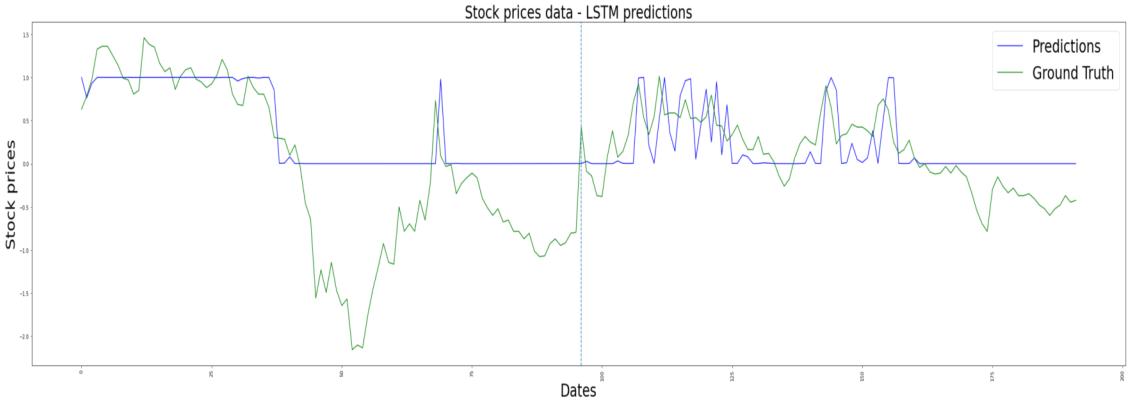

Figure 1: LSTM Predictions.

3.3 LSTM

3.3.1 Performance

In our analysis, we tested the trained LSTM model on half of the given dataset. The loss on the test data exhibited significant fluctuations, with values ranging from 0.182 to 0.92. The most common and smallest loss across batches was 0.182. Figure 1 presents the results of a model that used data corresponding to the 20MICRONS stock from January 1st, 2020 to October 23rd, 2020 with a lag of 1. It is evident that the model has captured the overall trend of the stock prices, but it exhibits high instability. This is demonstrated by the model either overestimating the prices by a considerable margin or remaining constant. Furthermore, in some instances, the model failed to accurately capture or was delayed in capturing actual events. For example, if there was a sudden price jump today, the predictions might only show the jump the following day or fail to show it at all. This is also indicated by the directional similarity $DS = 0.4736$. The blue dashed line in the figure demonstrates the training and validation datasets respectively. Notably, the validation and training predictions did not exhibit similar behavior, which indicates inconsistency in the prediction pattern across both seen and unseen data. Finally, despite its limitations the model was able to successfully predict the majority of the data. The LSTM model achieved a rank correlation of 0.86 ($p < 0.01$)

3.3.2 Hyperparameter Sensitivity

Although the model demonstrated good performance on the chosen data, it was highly sensitive to hyperparameters. Grid Search revealed that the model was most sensitive to the learning rate. Specifically, when the learning rate α exceeded 0.01, the gradient experienced an explosion. Grid Search indicated no difference in batch sizes and epoch sizes; however, this was not observed during manual tuning. In fact, increasing the batch size resulted in the model's inability to learn patterns effectively. A similar behavior was observed when the model was trained on different datasets without fine-tuning the hyperparameters. For the purpose of training, we chose the conventional hyperparameters for epochs = 1000, however, the batch size and the learning rate had to be significantly reduced. Table 3 represents all the results of the performed analysis on 20MICRONS dataset

Table 3: LSTM Model Performance (approx.)

Metric	20MICRONS 2
MSE	0.64
Spearman correlation	0.86
Directional Symmetry	0.474

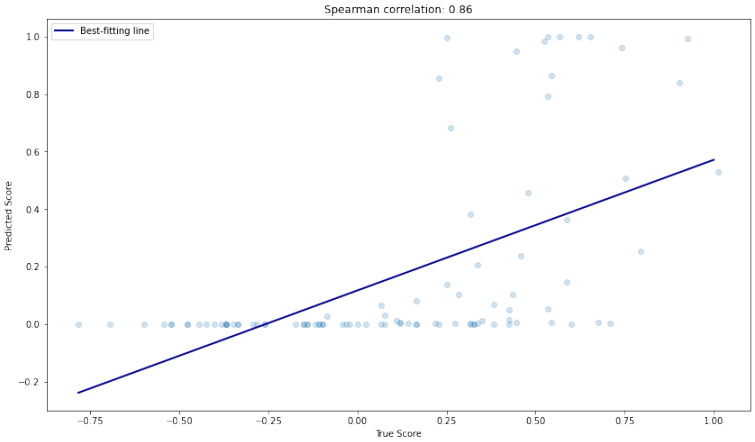

Figure 2: LSTM Spearman rank correlation.

```
In [102]: val_per_param
Out[102]: array([[[0.15851811, 0.15851827, 0.        ],
                  [0.15851723, 0.15851852, 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ]],

                 [[0.15851389, 0.15851386, 0.        ],
                  [0.15851387, 0.15851384, 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ]],

                 [[0.15851384, 0.15851384, 0.        ],
                  [0.15851384, 0.15851384, 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ]],

                 [[0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ]],

                 [[0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ],
                  [0.        , 0.        , 0.        ]],
```

Figure 3: LSTM Grid Search.

3.4 Transformer

3.4.1 Performance:

The model performs better with the exchange rate dataset as expected because of the significantly limited size of the 20MICRONS dataset. The model performance by datasets is summarized in table 4.

Table 4: Transformer Model Performance (approx.)

Metric	20MICRONS 2	exchange_rate
MSE	2.7071	0.0010
MAE	1.2992	0.0311
Spearman correlation	0.7170	0.6419
p-value 10	0	0
Directional Symmetry 13	50.5263	48.4210

The prediction and actual values are plotted vs. time as in figure 4 and figure 5 for the two datasets.

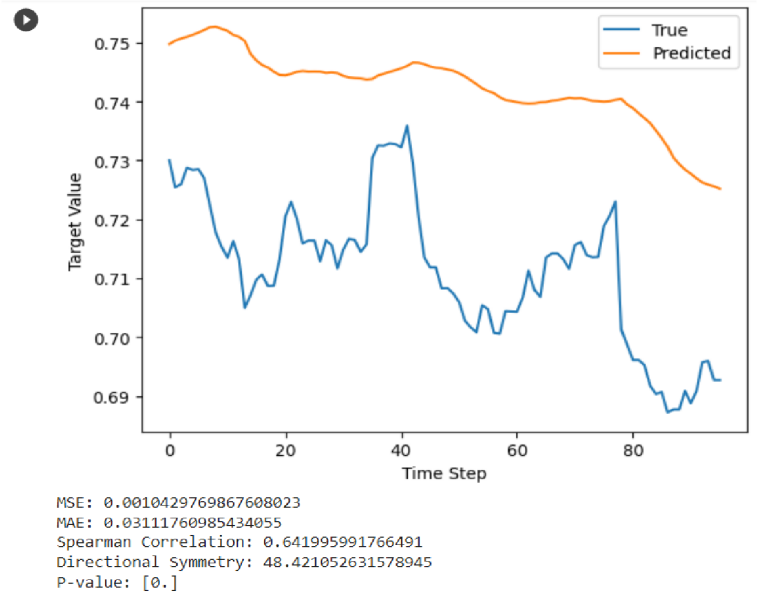

Figure 4: Transformer exchange_rate plot

3.4.2 Hyperparameter Sensitivity:

The hyperparameter choice for the 20MICRONS dataset can be summarized by the following output (figure 6):

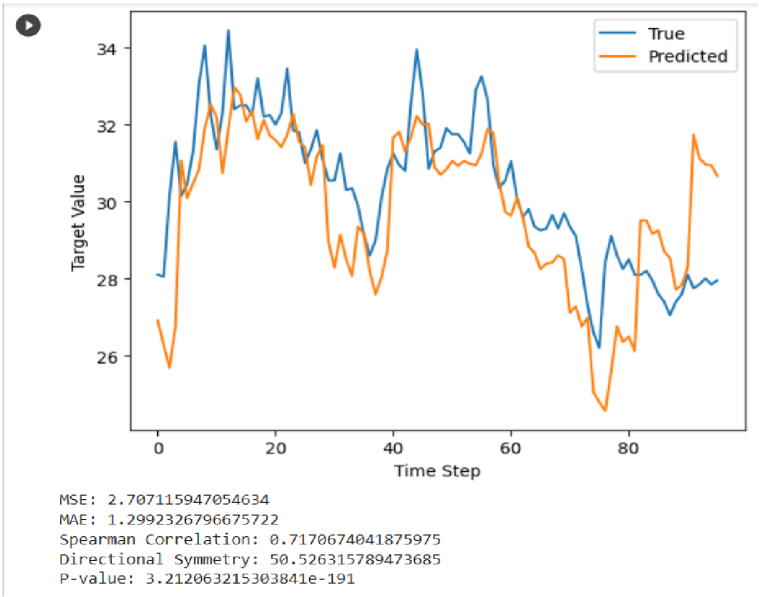

Figure 5: Transformer 20MICRONS plot

	embed_dim	num_heads	ff_dim	lookback	validation_loss
0	128	5	256	20	0.004736
1	128	5	256	25	0.011741
2	128	5	512	20	0.005417
3	128	5	512	25	0.048410
4	128	10	256	20	0.010969
5	128	10	256	25	0.029178
6	128	10	512	20	0.014135
7	128	10	512	25	0.040241
8	256	5	256	20	0.009272
9	256	5	256	25	0.007095
10	256	5	512	20	0.004817
11	256	5	512	25	0.015773
12	256	10	256	20	0.008964
13	256	10	256	25	0.007516
14	256	10	512	20	0.006576
15	256	10	512	25	0.024396

Figure 6: Transformer 20MICRONS Grid Search

3.5 Random Walk

To further assess the performance of the neural network models, we conducted a random walk evaluation, which is independent of past history. Surprisingly, the random walk demonstrated strong performance, with an average MSE loss of 0.1448832938034525.

4 Conclusion

This study succesfully investigated the potential of using machine learning methods for predicting stock prices. The three ML models, namely k-nearest neighbors (k-NN), Long Short-Term Memory

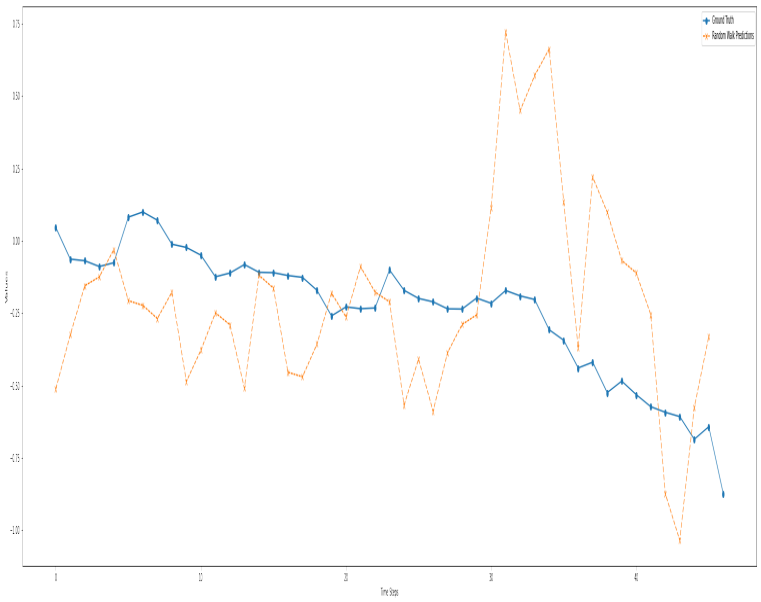

Figure 7: RandomWalk Prediction Plot

(LSTM), and Transformers were compared based on their predictive performance. Additionally, the performance of these models was compared against that of simple regression algorithms, random walk, ARIMA and an ensemble of kNN and ARIMA. The best model performance we have seen on the metric of MSE for the 20MICRONS dataset is the ensemble of kNN and ARIMA. This is not very surprising even in the presence more expressive and complex models like LSTMs (MSE = 0.64 approx.) and Transformers (MSE = 0.001 approx.) because the time series data doesn't encompass all information, so the arrival of new information is practically unpredictable. This gives the illusion that market prices are generated randomly. Therefore, more complex models such as LSTMs and Transformers are not able to outperform the relatively simple models. Furthermore, it is probable that the models display a few instances of successful predictions primarily by chance, as evidenced by the performance of random models.

5 Supplementary Material:

The .ipynb, .py, and .csv (code and dataset) files we wrote/used to obtain the results above can be accessed through the followin link: https://github.com/amiensh/CSC413.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Brownlee, J. (2017, March 24). *How to make out-of-sample forecasts with Arima in python*. MachineLearningMastery.com. Retrieved April 19, 2023, from https://machinelearningmastery.com/make-sample-forecasts-arima-python/

[4]V. Bugorskij, A. Sergienko (2008) Ispol'zovanie nejronnyh setej dlya modelirovaniya prognoza kotirovok cennyh bumag. *Prikladnaya informatika*,3(15). (in Russian)

[5]Ersan, D., Nishioka, C., &; Scherp, A. (2020). Comparison of machine learning methods for financial time series forecasting at the examples of over 10 years of daily and hourly data of Dax 30 and S&;P 500. Journal of Computational Social Science, 3(1), 103–133. doi: 10.1007s4200101900057-5

[6]Fama, E. F. (1970). Efficient Capital Markets: A review of theory and empirical work. The Journal of Finance, 25(2), 383. doi: 10.23072325486

[7] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

[8] Hayes, A. (2022). *Autoregressive integrated moving average (ARIMA) prediction model*. Investopedia. Retrieved April 19, 2023, from https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

[9]Loye, G. (2023). Long short-term memory: From zero to hero with pytorch. FloydHub Blog. Retrieved April 19, 2023, from https:blog.floydhub.comlong-shortterm-memoryfromzerotoherowithpytorch

[10] Maklin, C. (2019). *Arima model python exampletime series forecasting*. Medium. Retrieved April 17, 2023, from https:towardsdatascience.commachinelearningpart-19timeseriesandautoregressiveintegratedmoving-averagemodelarima-c1005347b0d7

[11]Moghar, A., & Hamiche, M. (2020). *Stock market prediction using LSTM recurrent neural network*. Procedia Computer Science, 170, 1168–1173. doi: 10.1016j.procs.2020.03.049

[12]Nayak, V. (2020, October 31). *Pytorch lstms for time series forecasting of Indian stocks*. Medium. Retrieved April 19, 2023, from https://medium.com/analytics-vidhya/pytorch-lstms-for-time-series-forecasting-of-indian-stocks-8a49157da8b9

[13]Saini, A., &; Sharma, A. (2019). *Predicting the unpredictable: An application of machine learning algorithms in Indian Stock Market*. Annals of Data Science, 9(4), 791–799. https:doi.org10.1007s40745019-002307

[14]Thakkar, A., &; Chaudhari, K. (2021). *A comprehensive survey on Deep Neural Networks for stock market: The need, challenges, and future directions*. Expert Systems with Applications, 177, 114800. https://doi.org/10.1016/j.eswa.2021.114800

[15] Vaswani,, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., &; Polosukhin, I. (2017). Attention Is All You Need. ArXiv, 1706.

[16]10mohi6. (2021, April 25). Super Easy Python stock price forecasting (using K-nearest neighbor) machine learning. Medium. Retrieved April 19, 2023, from https://10mohi6.medium.com/super-easy-python-stock-price-forecasting-using-k-nearest-neighbor-machine-learning-ab6f037f0077

Contributions:

Amina Shmanova worked on everything related to the LSTM model and RandomWalk model. Additionally, she also wrote the introduction for the project. Ashwin Karthikeyan worked on everything related the Transformer model. Additionally, he wrote the conclusion for the project. Varun Girish Vaze worked on kNN, ARIMA and ensemble models. Additionally, he wrote the subsections on kNN in Section 3 and ARIMA in Section 2 and 3. We all helped each other with finding datasets, papers, and sometimes debugging code.