Empirical Review of Models used for Predicting Financial Market Crashes Using Market Data

COMP 451 Final Project: Final Report

By Ping-Chieh Tu, Adrien Bélanger, Inigo Torres

December 13th 2024

Introduction

The volatility and complexity of financial markets have always been a significant challenge for the management of modern economies. For instance, the abrupt declines in the markets, often known in finance as crashes, can lead to widespread financial losses, economic recessions, and a loss of confidence in the stability of financial systems.

The idea of being able to anticipate and predict such market fluctuations is not new and has been widely studied from a mathematical point of view. However, conventional methods used by econometricians for time series analysis, such as Linear Trend Projection or Weighted Moving Average, may be effective for markets with general stationary trends but lack effectiveness for highly volatile ones. Modern machine learning techniques, particularly those involving neural networks and attention-based architectures, may provide a new pathway to surpassing these traditional techniques.

In this project, we propose to make an extensive comparative study of the effectiveness of 3 of the most widely used methods for Time series forecasting today: Recurrent Neural Networks (RNNs), Transformer-based architectures, and the Autoregressive Integrated Moving Average (ARIMA) model.

Each of these models is developed with a distinct methodological perspective. ARIMA models, based on classic statistics and linear algebra concepts, have long been a go-to tool for time series forecasting. RNNs, specifically Long Short-Term Memory (LSTM) networks, offer a non-linear and data-driven alternative that can capture long-range dependencies while reducing issues of vanishing or exploiting gradients. Transformers, which introduce attention mechanisms, aim to further improve predictive capabilities by focusing selectively on critical segments of past information, often achieving state-of-the-art performance in various sequential prediction tasks.

Our main objective is then to develop each of these models and subject them to a series of tests using real historical financial data to determine their effectiveness in predicting financial crashes.

Although the world of finance is changing, and there are no specific parameters since each financial market is governed by its own rules, we will test each of these models under the same conditions and thus establish an empirical review to provide clear insights into the strengths and limitations of each methodology. We aim to guide researchers and practitioners in selecting suitable models for short-term market crash prediction, ultimately contributing to more robust risk management strategies.

Background & Litterature Review

We are not the first to attempt to compare ML model's performance on their prediction of market crashes using SP500 historical data. Multiple approaches have been tried and tested. Time series analysis has used ARIMA and RNN based models, with the more recent addition of Transformers boosting and improving their performance [1]–[3]. Reviews of these models have been done before, but the comparison of these three models on short-term prediction using very recent market data is lacking in the literature.

One article compared Linear Regression and Autoregressive Moving Average (ARIMA) to predict the volatility and trend of SP500. [4]. Key-findings show that ARIMA struggled on short term predictions, particularly during the 1930s and 2020 volatile markets.

Zhou et al. demonstrated that transformers handle long-term dependencies well with the "Informer model," which efficiently handles long sequences and improves trend forecasting accuracy using a "self-attention" mechanism [5]. Moreover, Lim et al. were even able to outperform traditional methods (like ARIMA) by combining the Transformer architecture with recurrent layers [6].
Zhen Zeng studied the applicability of Transformers combined with RNNs in trend detection, specifically for financial time series, in his paper [7]. He concluded that combining these two architectures significantly improved forecasting accuracy for intraday stock price movements of the SP500.

Market Crashes do not hold a single definition. While historical data tags specific periods as depressions and bubbles, there are no specific metrics that are universally defined. Some have defined it as a rapid decline of 20% or more from a recent peak over a short period [8], [9]. Others still have defined them as a drawdown of 99.5% quantile [10].

Time series in the stock market often exhibit complex trends and non-linear patterns. Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN) are designed to learn long-term trends in sequential data, making them very good at modelling and forecasting financial market time series [11], [12].
LSTM networks address the vanishing gradient problem encountered in standard RNNs on this type of data by introducing specialized LSTM cells. These cells incorporate gates to control flow, allowing LSTMs to forget and remember information over long sequences [11].
The LSTM cell can be defined as in Appendix entry 5 [11]:
The *forget* gate, f_t, determines how much of the previous cell c_{t-1} is retained. The *input* gate i_t updates the cell with new information. The *output* gate o_t controls the hidden state, which is the LSTM output at time t [11].

Time series in the stock market are non-stationary, which means that their statistical properties (Avg, median) change over time. ARIMA models can be adapted to capture the behavior of non-stationary time-series [13]. A methodology formalized by Box et Al. in 2015 to apply the ARIMA model with a moving average component [14]. This makes ARIMA suitable for predicting non-stationary time series, such as financial markets, as demonstrated by M. K. Ho et al. in their paper [13]. The ARIMA model's Mathematical definition can be found in Appendix Entry 6[14]:
The use of ACF (autocorrelation function) and PACF (partial ACF) will help us decide the hyperparameters p, d, q for building our model [15], and performing Grid-Search on the specific hyperparameters will help us pinpoint the exact best accuracy we can get.

Transformers are based on self-attention mechanisms. They are able to weigh the importance of input elements by computing *attention scores*. While transformers were originally developed for natural language processing tasks, they have become popular for time-series forecasting due to their ability to handle long-range dependencies [16].
Unlike RNNs, Transformers process all steps at the same time, enabling parallel computation and improved efficiency for long-range dependency learning [16].
The core attention scoring by scaled dot product's definition can be found in Appendix entry 6 [16].

Transformers eliminate the need for recurrent connections by relying entirely on attention mechanisms. This allows them to model long-range dependencies more effectively than RNNs, which sometimes struggle with sequential bottlenecks and vanishing gradients [16]. While originally developed for natural language processing tasks, Transformers have been increasingly applied to time-series forecasting [7]. Given the volatility and complexity of financial markets, their ability to capture complex patterns makes them an ideal choice for financial market forecasting [17].

Drastic market crashes are rare. Models can achieve extremely high accuracy by simply predicting no market crash for every datapoint [18]. Other methods are thus needed to evaluate the models. Others in literature have used many methods, such as evaluating true positives and true negatives [19]. Others have used mean absolute error (MAE) to assess prediction accuracy [15]. Finally, some have used runtime and resource usage for practical feasibility to assess their performance [6].

Methodology

Note: All code can be found in the gitHub repository linked in the Appendix.

Definitions and Metrics used in our project

For our project, we use an LSTM-based RNN, Transformers and ARIMA to predict whether there will be a crash in the next x days based on the past y days. We defined a crash as a 99.5% quantile drawdown within a day. This gave us around 26 crashes in the test data.
To evaluate our models, we will measure the number of Predicted crashes against the real number of crashes. We will also check the number of false alarms and missed crashes. Finally, we will evaluate the training and resource usage time of all three models for each experiment.

Database, preprocessing and Models

We pulled the code from the SP500 public data available on the Yahoo Finance financial data library. From there, we preprocessed the code by labelling crashes in the database.
Our code made use of a sliding window for the sequence length. Then, we created batches of data points of size 20. We then tried to predict if there would be a crash in the following shift days, which was a hyperparameter. For example, we could have a 15-day sliding window to predict if there would be a crash in the 3 days after those 15 days. This made the problem one of binary classification, which was simple and efficient to process and made for easy comparison.
Our RNN code was based on the *Keras* Sequential model, using the pre-made LSTM layers [20], [21]. This was used since our project aimed to be a comparison and not one of implementation. The code used two LSTM layers of 50 units each, then a dense layer with a sigmoid activation layer for binary classification. We then used the Adam optimizer with a binary cross-entropy loss function to train our binary model.
Our ARIMA code uses the default ARIMA from **statsmodels**[22] without using trend nor seasonal features of our data. We first use the ADF test to determine the stationarity of our data. It shows that our data is non-stationary, and therefore, the differencing term d is needed. Then, we looked at the PACF for the decision of p and ACF for the decision of q.

From PACF, we can choose either $p = 0$ or $p = 1$. From ACF, it is hard to tell what should be used as q. Therefore, we run the grid search to have the hyperparameters to ensure we have a high accuracy.
For our Transformer-based neural network architecture, we defined a positional encoding function (`positional_encoding`), introducing a temporal order understanding that Transformers inherently lack. The `transformer_model` function then constructs a model configured to accept time-series data (i.e. normalized closing prices or market indicators over a specified look-back window). Each input sequence is passed through a dense embedding layer (`Dense(embed_dim)(x)`) and augmented with positional encodings. This enables the model to capture temporal trends more effectively. Next, a Multi-Head Attention layer selectively focuses on different parts of the input sequence, while skip connections and `LayerNormalization()` stabilize learning. The attended information is refined by a feed-forward network (Dense layers), then flattened and passed through a final sigmoid-activated layer for binary crash predictions. Finally, the model is compiled with a binary cross-entropy loss and accuracy metric to evaluate its performance on historical financial data.

Experiments

1. 30 days sliding window, 14 days shift days
   Hypothesis: Here, we expect Transformers to profit from the larger sliding window, surpassing the performance of RNNs [4], [12]. We expect ARIMA to have the best performance of all three experiments in this one, as it has the most context and does better with more data [13].

2. 14 days sliding window, 7 days shift days Hypothesis: We expect both Transformers and RNNs to perform comparably by utilizing their strength in capturing medium-range dependencies. Meanwhile, ARIMA may also deliver reasonable results, as the window provides sufficient context for traditional statistical modelling.

3. 7 days sliding window, 3 days shift days
   Hypothesis: Our belief in this experiment is that LSTM (Long Short Term Memory) based RNN models will excel due to their capability to handle shorter sequences and identify immediate dependencies effectively, as Hewamalage et al. demonstrated [23]: RNN architectures, and specially LSTMs can adapt well to limited sequential data, and still be able to learn patterns from it. Thus, we expect a superior performance of RNNs, followed by Transformer and ARIMA respectively.

Overall, we expect ARIMA to do good on longer sequences [13]. We expect RNNs to do the best with the low context of the third experiment [23]. Transformers should excel in higher context windows and fall slightly behind RNNs in the third experiment. Because of the short-term nature of the experiments, we expect ARIMA to be behind the other two models in all three experiments, given the low context.

Results

From the results of the experiments, the ARIMA model has the worst performance, despite still having some correct predictions, it has the highest amount of false alarms in all 3 experiments compared to RNN and transformers, which have only a few. It would be better to exclude ARIMA from the comparison and talk about it later.

Discussion

The results of Experiment 1, in Figure 1, present a highly skewed report. In comparison, the RNN and the Transformers performed relatively similarly, with maybe slightly better overall performance on the part of the Transformers.

However, ARIMA performed very much underwhelmingly, with a prediction of over 3682 crashes. This makes 3087 false alarms, compared to 5 for RNNs and 3 for transformers. This shows that ARIMA struggled deeply with this experiment. This could be because of the way the experiment was set up: as mentioned, ARIMA is a statistical model that needs more data to predict correctly.

While these results do follow our hypothesis, it would be unfair to say we can draw positive conclusions from this experiment. The ARIMA model performed much under our expectations. The RNN and Transformers performed as expected, with Transformers having slightly better results.

In terms of training and resource usage, the Transformers, because of their architecture, clearly beat all three with their training time and disk and RAM usage. RNNs were slightly slower and consumed more disk space. ARIMA, because of our grid-search implementation, took 480 mins to compute. This is compared to the 2-3 minute training time of the other two models. ARIMA once again fell very much under our hypothesis.

In Experiment 2, RNN and Transformers still perform similarly. While Transformers has more correct predictions, it also has one more false alarm than RNN. The results of RNN and Transformers go without a hypothesis. They both have a near 100% prediction. As in Experiment 1, Transformers performed better in training time and memory usage.

In the 3rd Experiment, where we used a shorter 7-day sliding window and a 3-day prediction horizon to test our models, the results closely aligned with our predictions. We can see that the RNN model performed exceptionally well, confirming the hypothesis that LSTM-based networks are still able to capture immediate dependencies in short sequences [23]. What's more, the Transformers model also maintained a high really close prediction accuracy. However, it did not surpass the RNN in this reduced timeframe. Finally, as expected, the ARIMA model continued generating several false alarms despite reducing the time frame. Overall, these results suggest that, even on short time frames, neural network models are stronger than traditional linear models like ARIMA in quickly identifying abrupt fluctuations in financial time series (i.e. such as crashes).

Conclusion

This empirical review project demonstrated that Recurrent Neural Networks (RNNs) and Transformer-based architectures consistently outperformed the ARIMA model in predicting financial crashes. Based on these results, we can affirm that RNNs and Transformers are better suited to capture the complex, non-linear relationships and long-range dependencies of financial time series, resulting in higher prediction accuracy of the crashes. In contrast, the ARIMA model, inherently based on linear assumptions, was proved to be less effective in handling the volatile and non-linear nature of financial markets. Mathematically, a differenced series $(1 - B)^d X_t$ is modelled as in Appendix Entry 7. However, the ARIMA model, restricted by its linearity, assumed that the time series X_t could be expressed as a linear function of its past values and past error terms and could not properly capture the sudden chaotic shifts and non-linear patterns of our financial data. Consequently, ARIMA's linear structure leads to a less effective approximation compared to models better designed to handle non-linear patterns like RNNs or Transformers.

Future Directions

While the superiority of Transformers and RNNs over ARIMA in predicting financial crashes is more than evident, ARIMA models could still be useful as a baseline, particularly in periods of relative market stability where linear dynamics are more pronounced. A potential avenue of improvement for this project could be to investigate the combination of RNNs/Transformers with ARIMA. For instance, during less turbulent intervals, the linear patterns captured by ARIMA can complement the non-linear adaptability of RNNs and Transformers. The linear and more conservative trend-following nature of ARIMA may be used to balance the overreaction of non-linear methods to short-term noise, decreasing false crash alarms. By intelligently combining these models, we could leverage ARIMA's strengths for stable trend-like segments of the data while allowing more sophisticated architectures to handle the abrupt non-linear fluctuations. Regarding a more general improvement of the predictive power of our models, incorporating more macroeconomic indicators specific to each financial market and studying alternative data preprocessing techniques, such as feature engineering or advanced normalization schemes, may potentially lead to better results.

Self-Assessment

From the result of this project, the RNN and Transformer models produced promising results, demonstrating their capability to capture complex temporal dependencies and effectively model market patterns leading to crashes. However, due to the lack of knowledge in data science, the performance of the ARIMA model fell short of expectations. With more knowledge of the fundamentals of data science of preparing data and statistics, we can possibly transform the data to better fit the ARIMA model with different techniques and get better results.

Contributions

1. Adrien - For the report, my main contributions were for writing the Litterature Review and parts of the Methodology, Discussion and Conclusion. I've also contributed to the code by implementing the RNN model, consolidating the three models in a notebook and helping design the experiments.

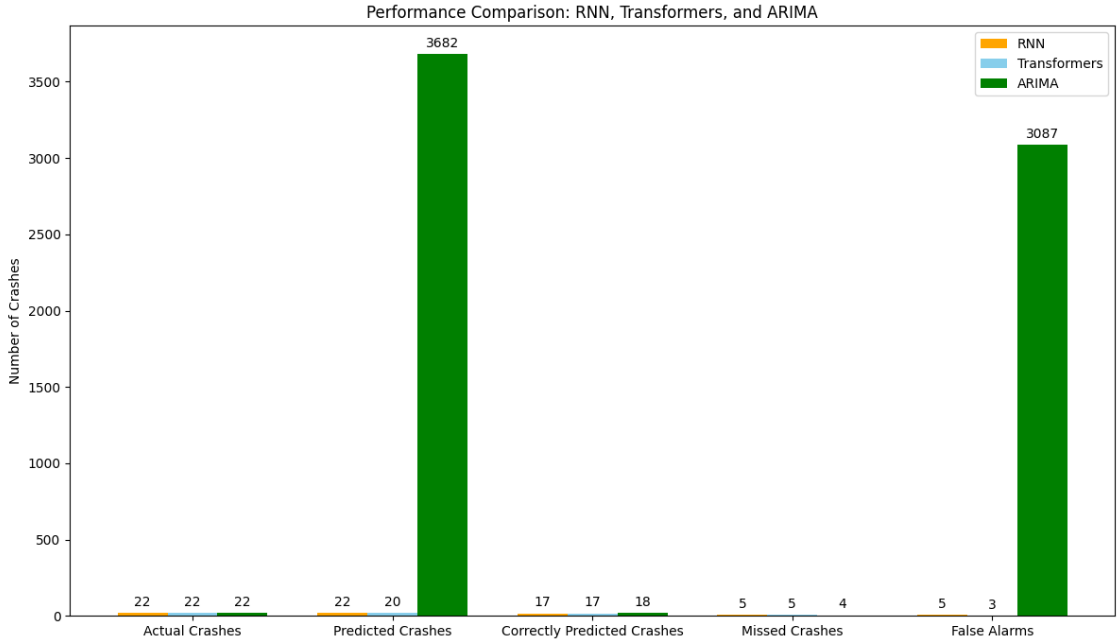
2. Ping-Chieh - For the report, I mainly contributed to the empirical evaluations and the self-assessment sections. I also provided some suggestions on the conclusion of the report. I've also contributed to the code of the ARIMA model.

3. Inigo Torres, for this report, I implemented the code for the Transformers model. In addition, I was responsible for writing the introduction, conclusion, and future directions sections, plus, I also contributed to the discussion and interpretation of the results of the experiments, primarily those related to the Transformers model.

Appendix

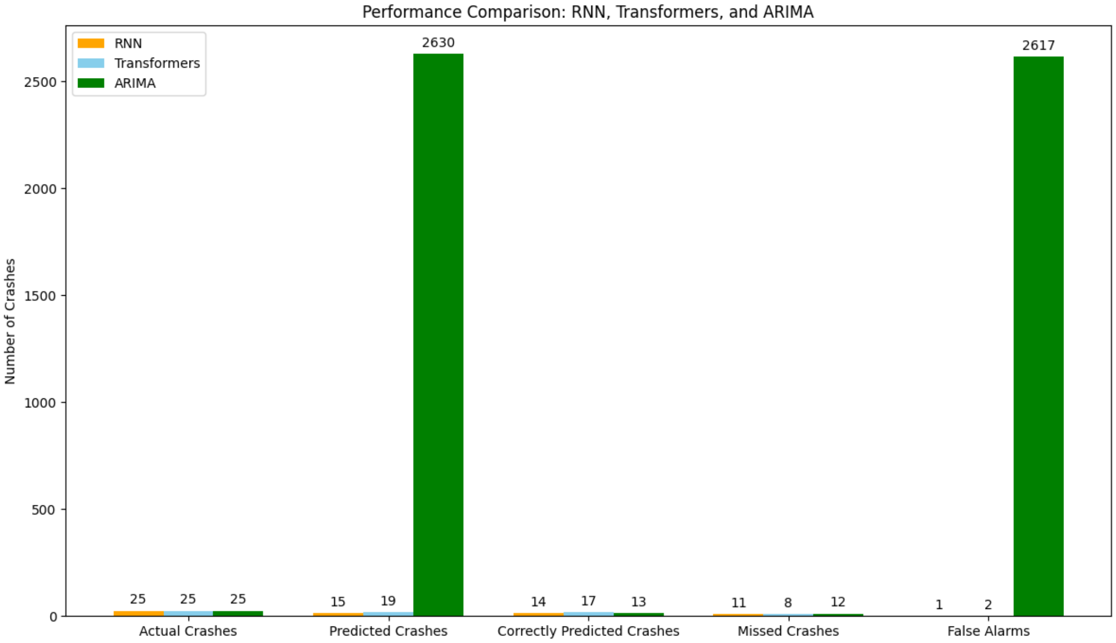
1. Link to the gitHub repository with the code: `https://github.com/AdrienBelanger/451-Project`

2. Figure 1: Experiment 1 Result plots



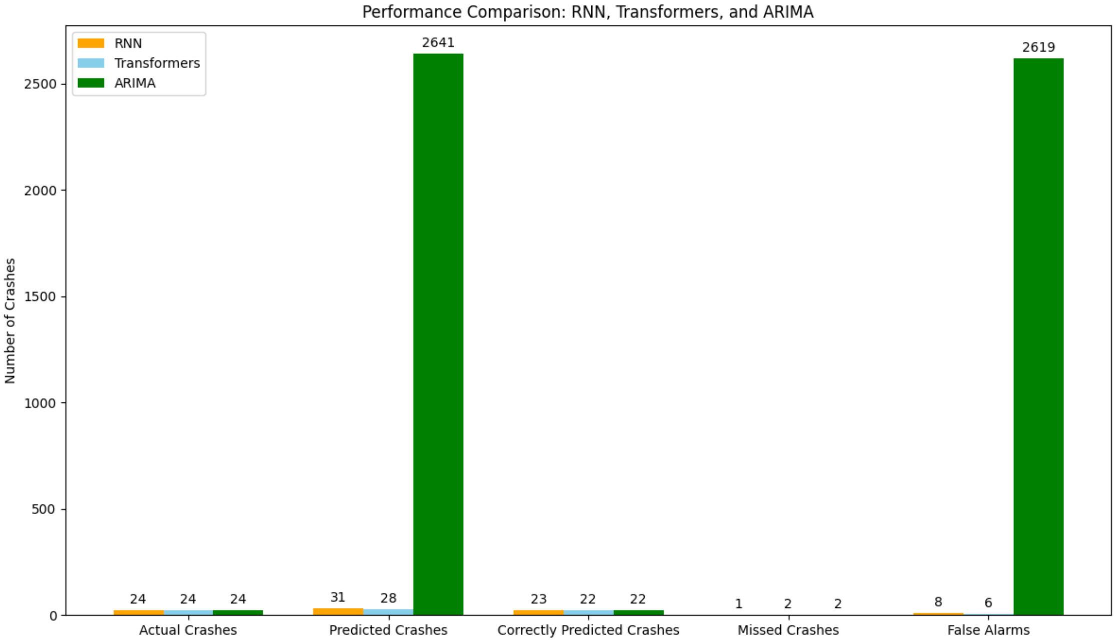
(a) Figure 1: Experiment 1 Comparison Plot

3. Figure 2: Experiment 2 Result plots



(a) Figure 2: Experiment 2 Comparison Plot

4. Figure 3: Experiment 3 Result plots



(a) Figure 3: Experiment 3 Comparison Plot

5. LSTM Cell Equation

$$\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i),\\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f),\\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o),\\
\tilde{c}t &= \tanh(W_c x_t + U_c ht - 1 + b_c),\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t,\\
h_t &= o_t \cdot \tanh(c_t),
\end{aligned}$$

   where

   - x_t is the input at time t, representing market data (e.g., stock price, volume),
   - h_{t-1} is the hidden state (output) of the previous time step,
   - i_t, f_t, o_t are the input gate, forget gate, and output gate
   - $\tilde{c}_t$ is the candidate memory cell state,
   - c_t is the cell state at time t, capturing long-term dependencies,
   - W and U are the weight matrices and b is the bias vector,
   - σ is the sigmoid activation function

6. ARIMA model definition: The ARIMA model is defined mathematically as [14]:

$$\varphi(B)z_t = \phi(B)\nabla^d z_t = \theta_0 + \theta(B)a_t$$

   where

   - $\phi(B)$ is the autoregressive part with p degrees,
   - $\theta(B)$ is the moving average part with q degrees,
   - ∇ is the integrated (degree of differencing) part with d degrees,
   - and θ_0 is the constant term.

7. Transformers Attention Scoring function

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

   where:

   - Q (queries), K (keys), and V (values) are linear projections of the input data,
   - is the dimension of the key vectors,
   - The softmax function ensures that attention scores are normalized.

8. Differenced Series

$$(1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p)(1 - B)^d X_t = (1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q)\varepsilon_t,$$

   where B is the backshift operator, φ_i and θ_j are parameters, and ε_t is white noise.

References

[1] P. Okpeke, P. O. Paul, and T. V. Iyelolu, "Predicting stock market crashes with machine learning: A review and methodological proposal," *Open Access Research Journal of Science and Technology*, 2024.

[2] S. Ahmed, I. E. Nielsen, A. Tripathi, S. Siddiqui, R. P. Ramachandran, and G. Rasool, "Transformers in time-series analysis: A tutorial," *Circuits Syst. Signal Process.*, vol. 42, no. 12, pp. 7433–7466, 2023, ISSN: 0278-081X. DOI: 10.1007/s00034-023-02454-8. [Online]. Available: https://doi.org/10.1007/s00034-023-02454-8.

[3] K. E. ArunKumar, D. V. Kalaga, C. Mohan Sai Kumar, M. Kawaji, and T. M. Brenza, "Comparative analysis of gated recurrent units (gru), long short-term memory (lstm) cells, autoregressive integrated moving average (arima), seasonal autoregressive integrated moving average (sarima) for forecasting covid-19 trends," *Alexandria Engineering Journal*, vol. 61, no. 10, pp. 7585–7603, 2022, ISSN: 1110-0168. DOI: https://doi.org/10.1016/j.aej.2022.01.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016822000138.

[4] K. Guo, Z. Jiang, and Y. Zhang, "Prediction of sp500 stock index using arim and linear regression," *Highlights in Science, Engineering and Technology*, vol. 38, pp. 399–407, Mar. 2023. DOI: 10.54097/hset.v38i.5848. [Online]. Available: https://drpress.org/ojs/index.php/HSET/article/view/5848.

[5] H. Zhou, S. Zhang, J. Peng, *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11 106–11 115, 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17325.

[6] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021. DOI: 10.1016/j.ijforecast.2021.03.012. [Online]. Available: https://doi.org/10.1016/j.ijforecast.2021.03.012.

[7] Z. Zeng *et al.*, "Financial time series forecasting using cnn and transformer," *arXiv preprint arXiv:2304.04912*, 2023. [Online]. Available: https://arxiv.org/abs/2304.04912.

[8] M. Fonville, "Understanding stock market corrections and crashes (2024)," 2024. [Online]. Available: https://www.covenantwealthadvisors.com/post/understanding-stock-market-corrections-and-crashes?utm_source=chatgpt.com.

[9] J. Chen, "Guide to stock market crash," 2022. [Online]. Available: https://www.investopedia.com/terms/s/stock-market-crash.asp.

[10] E. Jacobsson, "How to predict crashes in financial markets with the log-periodic power law," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:201929664.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[12] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, ISSN: 2162-2388. DOI: 10.1109/tnnls.2016.2582924. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2016.2582924.

[13] M. K. Ho *et al.*, "Application of arima models for non-stationary time series," *Journal of Physics: Conference Series*, vol. 1988, no. 1, p. 012 041, 2021, See also Box et al. [2015] and Hyndman et al. [2018]. DOI: 10.1088/1742-6596/1988/1/012041. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1988/1/012041/pdf.

[14] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th. Wiley, 2015. [Online]. Available: http://repo.darmajaya.ac.id/4781/1/Time%20Series%20Analysis_%20Forecasting%20and%20Control%20%28PDFDrive%29.pdf.

[15] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd. OTexts, 2018. [Online]. Available: https://otexts.com/fpp2/.

[16] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762.

[17] N. Wu, B. Green, X. Ben, and S. O'Banion, *Deep transformer models for time series forecasting: The influenza prevalence case*, 2020. arXiv: 2001.08317 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2001.08317.

[18] Anonymous, *Financial market crash prediction: Challenges and solutions*, Accessed: 2024-06-07, 2021. [Online]. Available: https://cs230.stanford.edu/projects_winter_2021/reports/70767949.pdf.

[19] H. Dichtl, W. Drobetz, and T. Otto, "Forecasting stock market crashes via machine learning," *Journal of Financial Stability*, vol. 65, p. 101 099, 2023, ISSN: 1572-3089. DOI: https://doi.org/10.1016/j.jfs.2022.101099. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572308922001206.

[20] K. Team, *Keras lstm layer*, 2024. [Online]. Available: https://keras.io/api/layers/recurrent_layers/lstm/.

[21] K. Team, *Keras sequential model*, 2024. [Online]. Available: https://keras.io/guides/sequential_model/.

[22] statsmodels, *Statsmodels arima.*

[23] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2020.06.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207020300996.