Deep Learning Fundamentals: Assignment 3
RNNs for Stock Price Prediction

Variyas Nitin Singla
The University of Adelaide
Adelaide, South Australia, 5005, Australia
a1872896@adelaide.edu.au

Abstract

This study delves into the efficacy of advanced Recurrent Neural Network (RNN) architectures, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, in the intricate domain of stock price forecasting. Traditional predictive models often struggle to fully grasp the intricate, time-sensitive patterns inherent in stock market data. This research endeavors to bridge this gap by exploiting the unique sequential data processing strengths of RNNs. Our approach involves a comprehensive methodology that includes the collection, preprocessing, and analysis of historical stock price data from leading corporations such as Apple, Microsoft, and Google. We meticulously construct, train, and validate our models, ensuring rigorous testing and optimization. The findings underscore the superior capability of RNNs in capturing the dynamic and complex trends of stock prices, highlighting their significant potential as powerful tools for investors and financial analysts. The performance of the models is scrutinized using established metrics, with a particular focus on their precision and practical applicability in dynamic market conditions.

1. Introduction

The financial market, particularly the stock market, plays a pivotal role in the global economy, with its complexities and unpredictable nature posing significant challenges for accurate prediction. The stock market is influenced by a myriad of factors, including economic indicators, investor sentiment, and geopolitical events, contributing to its inherent volatility and unpredictability [2]. This dynamic nature necessitates the development of robust and adaptive prediction models.

Traditional stock market prediction techniques, such as fundamental and technical analysis, have been integral in historical market analysis but often fall short in capturing complex market dynamics [3, 5]. The rise of machine learning has transformed the landscape of financial forecasting, with techniques such as decision trees, support vector machines (SVM), and neural networks offering new approaches to model these complexities [7, 6, 11, 10].

In recent years, deep learning methods, especially recurrent neural networks (RNNs), have gained prominence for their ability to process sequential and time-series data, a common characteristic of financial markets [12, 13, 16]. RNNs and their advanced variants, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), are particularly effective in capturing temporal dependencies in stock price data, addressing the vanishing gradient problem prevalent in traditional RNNs [14, 15].

The current research landscape in stock market prediction using deep learning is vibrant and diverse, with studies exploring a range of models including RNNs, CNNs, GRUs, LSTMs, and more recent innovations like Transformer models and reinforcement learning [25]. These models have been applied to various tasks such as stock price prediction, stock movement prediction, portfolio management, and trading strategies [25]. Deep learning models have been adapted to process not only numerical time-series data but also unstructured data like financial news, social media posts, and more, providing a holistic view of market influencing factors [25].

However, challenges remain in the field, such as addressing non-linear market dynamics, handling noisy financial data, and improving the interpretability of deep learning models. There is also a growing focus on hybrid models that combine the strengths of different deep learning architectures to enhance prediction accuracy [25].

In this paper, we aim to explore the use of RNNs, with a particular focus on LSTM and GRU architectures, for predicting stock prices. We will analyze historical stock price data from major corporations, examining the efficacy of these models in real-world financial contexts. Our approach includes data acquisition, preprocessing, model building, training, and validation, with an emphasis on predictive accuracy and practical applicability in finance.

2. Literature Review

2.1. Traditional Methods in Stock Price Prediction

Stock price prediction has traditionally been grounded in fundamental and technical analysis. Fundamental analysis, deeply influenced by the Efficient Market Hypothesis (EMH) proposed by Fama [1], considers financial statements, company health, and macroeconomic indicators for stock valuation [2, 3]. Technical analysis, on the other hand, relies on identifying patterns in historical market data, such as price movements and trading volumes, a method pioneered by Dow and Hamilton [4, 5]. While providing valuable insights, these methods often struggle in volatile or non-linear market environments [6, 7].

2.2. Evolution of Machine Learning in Finance

The advent of machine learning in financial analysis marked a significant transition from traditional methods. Starting with linear models, the field advanced to more complex algorithms like decision trees, support vector machines (SVMs), and neural networks, demonstrating their efficacy in identifying non-linear patterns in financial data [8, 9, 10]. These models, however, grappled with the temporal and stochastic nature of stock markets, making accurate predictions challenging [11, 10].

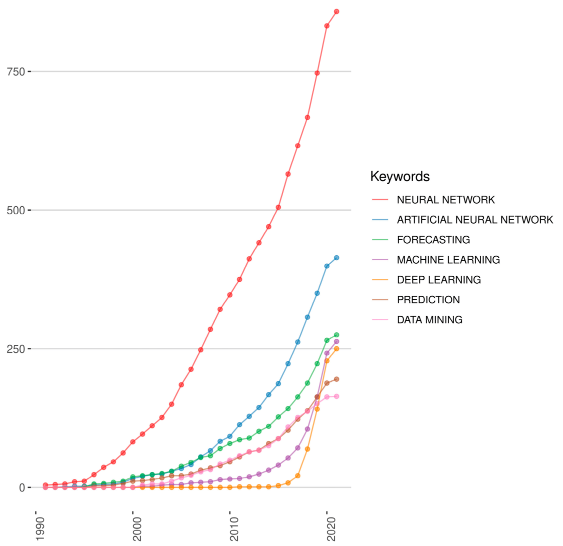


Figure 1. Progression of Machine Learning in Finance

2.3. The Rise of Neural Networks in Stock Prediction

The introduction of neural networks, particularly Recurrent Neural Networks (RNNs), brought a paradigm shift in stock price prediction. RNNs, with their capability to process time-series data, became instrumental in analyzing market trends [12, 13]. The development of Long Short-Term Memory (LSTM) units and Gated Recurrent Units (GRUs) further enhanced this field by addressing the limitations of traditional RNNs, particularly in learning long-term dependencies in data [14, 15, 16].

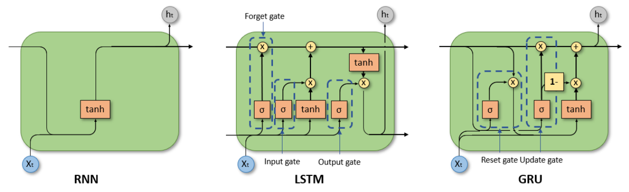


Figure 2. The Rise of Neural Networks

2.4. Current State and Challenges

Despite advancements, the application of neural networks in stock prediction continues to face significant challenges. Overfitting, for instance, often leads to poor performance in real-world scenarios [17]. The 'black-box' nature of deep learning models raises concerns about their interpretability and trustworthiness [30]. Moreover, the dynamic and ever-evolving nature of financial markets poses a challenge to the predictive power of static models [19, 20].

2.5. Future Directions

Recent trends in stock price prediction include integrating alternative data sources such as social media sentiment and news analytics with traditional financial data. This approach aims to provide a more comprehensive view of market influencers [26, 27]. Additionally, the exploration of advanced techniques like reinforcement learning and hybrid models that combine various algorithms shows promise for enhancing prediction accuracy and model robustness [29, 24, 25].

3. Dataset Overview

This study leverages a meticulously curated dataset of stock prices for Apple Inc. (AAPL), Microsoft Corp. (MSFT), and Alphabet Inc. (GOOGL). The data, sourced from Yahoo Finance via the 'yfinance' Python library, spans an extensive period from November 1, 2008, to November 1, 2023. It comprises 3774 daily records for each stock, capturing the dynamics of the stock market with features like opening, high, low, closing, and adjusted closing prices, along with the volume of stocks traded.

3.1. Data Acquisition

Data acquisition was conducted using the 'yfinance' library, a powerful tool that interfaces with Yahoo Finance's API to download historical market data. This data is crucial for analyzing financial trends, assessing stock performance, and applying advanced predictive modeling techniques in the field of quantitative finance.

3.2. Data Visualization and Analysis

3.2.1 Stock Price Trends

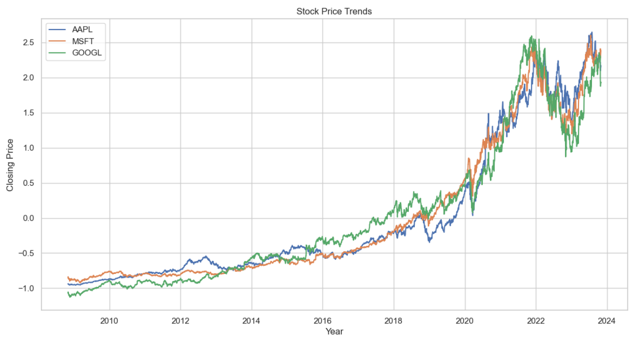


Figure 3. Stock Price Trends of AAPL, MSFT, and GOOGL

Figure 3 depicts the evolution of closing stock prices over the 15-year period. AAPL's price exhibits significant growth, reflecting the company's robust market performance and investor confidence. MSFT and GOOGL also show substantial appreciation, demonstrating the tech sector's overall positive trajectory.

3.2.2 Stock Trading Volume Over Time

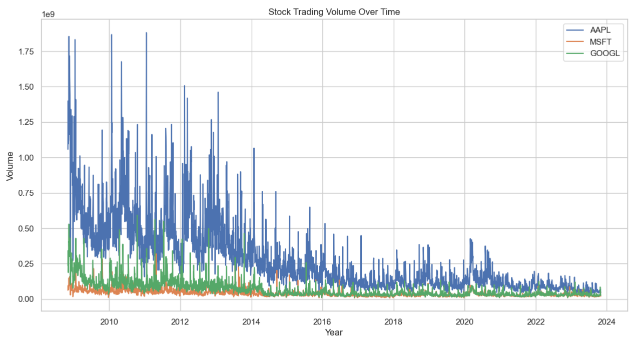


Figure 4. Trading Volume Over Time for AAPL, MSFT, and GOOGL

Figure 4 details the volume of shares traded. AAPL's volume spikes could correspond to key corporate events or product releases, while the trading volumes for MSFT and GOOGL suggest a steadier investment landscape.

3.2.3 Moving Averages of Stocks

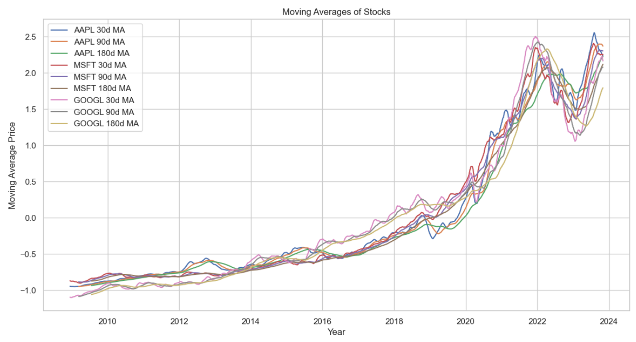
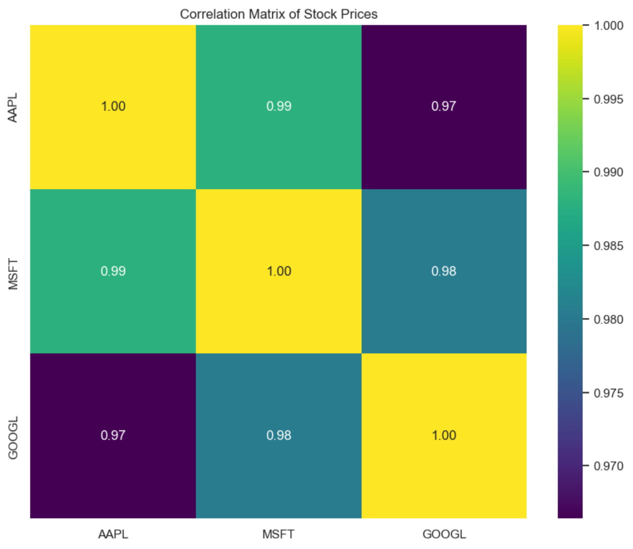


Figure 5. Moving Averages of AAPL, MSFT, and GOOGL

Figure 5 presents a clear depiction of short-term and long-term trends through moving averages, providing investors with a tool for technical analysis to predict future price movements and identify trading signals.

3.2.4 Correlation Matrix of Stock Prices



	AAPL	MSFT	GOOGL
AAPL	1.00	0.99	0.97
MSFT	0.99	1.00	0.98
GOOGL	0.97	0.98	1.00

Figure 6. Correlation Matrix of Stock Prices

As shown in **Figure 6**, the correlation matrix offers insights into the inter dependencies of these tech stocks' prices. The strong correlations indicate that the stocks may be influenced by similar economic factors or market sentiments.

3.3. Statistical Summary

The descriptive statistics reflect the historical stock performances and the inherent market volatility. The extensive dataset provides a robust foundation for statistical analyses, which are critical for understanding market trends and guiding investment strategies.

3.4. Data Normalization and Preprocessing

The normalization process, essential for machine learning models, ensures that features contribute equally to the algorithm's predictions, preventing any one feature from dominating due to its scale.

3.5. Descriptive Statistics Tables

Table 1 and 2: Descriptive Statistics for AAPL, MSFT, and GOOGL elaborates on the statistical nuances of the dataset. It highlights not only the mean and standard deviation but also the full distribution of values via quartile and extremity data points. These statistics are crucial in understanding the distribution and range of stock prices, which inform risk assessments and potential forecasting accuracy.

Stock	Mean Open	Std Dev	Mean Volume
AAPL	$54.21	$53.68	275,105,128
MSFT	$105.29	$97.66	39,555,201
GOOGL	$49.83	$38.54	66,266,123

Table 1. Descriptive Statistics for AAPL, MSFT, and GOOGL (Part 1)

Stock	30d MA	90d MA	180d MA
AAPL	-0.00	-0.02	-0.03
MSFT	0.00	-0.02	-0.03
GOOGL	0.00	-0.01	-0.02

Table 2. Descriptive Statistics for AAPL, MSFT, and GOOGL (Part 2)

4. Methodology

4.1. Computational Environment

Our research employed a sophisticated computational environment tailored for deep learning applications in stock price prediction. The environment was set up on a macOS system, leveraging Python as the primary programming language, with Jupyter Notebook serving as the integrated development environment. This setup facilitated interactive development and visualization.

4.1.1 Software and Libraries

We utilized an array of Python libraries to support various stages of our analysis:

- **Data Manipulation and Analysis:** `NumPy` and `Pandas` were employed for numerical computations and data handling.
- **Machine Learning and Neural Networks:** We utilized `TensorFlow` and `Keras` for constructing and training neural network models, including LSTM and GRU architectures. `Keras Tuner` was used for hyperparameter optimization.
- **Statistical Modeling:** Time series analysis was conducted using `Statsmodels` and `Arch`.
- **Data Visualization:** For graphical representation, `Matplotlib`, `Seaborn`, and `Plotly` were used.
- **Financial Data Acquisition:** `yfinance` was employed to fetch historical stock price data.

4.1.2 Development and Execution

The Jupyter Notebook environment allowed for seamless integration of code, visualizations, and documentation. We structured our code into various segments for data acquisition, preprocessing, model training, evaluation, and visualization. The use of `%matplotlib inline` ensured that our visualizations were directly embedded within the notebook.

4.1.3 Parallel Processing and Resource Management

Given the computationally intensive nature of neural network training and large-scale data processing, we utilized parallel processing capabilities through `ThreadPoolExecutor` and `Multiprocessing.Pool`. This approach significantly improved computational efficiency on the macOS platform.

4.1.4 Environment Setup

The entire suite of Python packages was installed via `pip`, ensuring easy replicability of our computational environment. We maintained a focus on reproducibility and consistency throughout the research process.

This computational setup was integral to the successful execution of our research, providing the necessary tools and framework for conducting in-depth analyses and deriving insightful conclusions in stock price prediction.

4.2. Data Acquisition

The study utilized historical stock price data of major companies, specifically Apple (AAPL), Microsoft (MSFT), and Google (GOOGL). The data was sourced using the 'yfinance' library, which provides comprehensive financial data directly from Yahoo Finance. The dataset includes daily prices covering a period from January 1, 2008, to November 31, 2023, encompassing open, high, low, close prices, and trading volume. Data for major companies (AAPL, MSFT, GOOGL) was sourced using 'yfinance'. The dataset includes daily open (O_t), high (H_t), low (L_t), close (C_t), and volume (V_t) prices.

4.3. Data Preprocessing

Data preprocessing involved several key steps:

- **Cleaning:** Removal of missing or erroneous data points.
- **Normalization:** The prices were normalized using the Min-Max scaling technique to ensure model stability and convergence during training.
- **Feature Selection:** Key features relevant to stock price prediction, such as closing price and volume, were selected for model input.
- **Time Series Windowing:** The data was structured into sequential time windows, which are essential for capturing temporal dependencies in RNNs.
- **Normalization:** Prices were normalized using Min-Max scaling:

$$P_{\text{norm}} = \frac{P_t - P_{\text{min}}}{P_{\text{max}} - P_{\text{min}}} \tag{1}$$

  where P_t is the price at time t, and P_{min}, P_{max} are the minimum and maximum prices in the dataset, respectively.

- **Time Series Windowing:** Data was structured into N-day windows to create sequences for RNN input.

4.4. Model Architecture

The study experimented with various RNN architectures, focusing on Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. The general architecture of the models included:

- An input layer shaped according to the time series window size.
- Several LSTM or GRU layers to capture the temporal dependencies in the data.
- Dropout layers to prevent overfitting.
- A dense output layer with linear activation for predicting the stock price.

4.5. Training Procedure

The models were trained using a historical subset of the data (2008-2021) and validated on a more recent subset (2022-2023). The Adam optimizer was employed for its efficiency in handling sparse gradients and adaptive learning rates. The loss function used was Mean Squared Error (MSE), as it is well-suited for regression tasks like price prediction.

4.5.1 LSTM Layer

An LSTM layer is formulated as:

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned} \tag{2}$$

where f_t, i_t, o_t are the forget, input, and output gates, C_t and h_t are the cell and hidden states, σ is the sigmoid function, and W, b are the weights and biases.

4.5.2 GRU Layer

A GRU layer is defined as:

$$\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
\tilde{h}_t &= \tanh(W \cdot [r_t * h_{t-1}, x_t] + b) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned} \tag{3}$$

where z_t and r_t are the update and reset gates, and $\tilde{h}_t$ is the candidate activation.

4.6. Training Procedure

The loss function, Mean Squared Error (MSE), is given by:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 \tag{4}$$

where Y_i and $\hat{Y}_i$ are the true and predicted values, respectively.

4.7. Evaluation Metrics

The performance of the models was evaluated using several metrics:

- **Mean Absolute Error (MAE)**: Provides an average magnitude of the errors in a set of predictions, without considering their direction.

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |Y_i - \hat{Y}_i| \tag{5}$$

- **Root Mean Squared Error (RMSE)**: Offers a measure of the magnitude of the error, penalizing larger errors more severely.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2} \tag{6}$$

- **Coefficient of Determination** (**R^2 Score**): Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2} \quad (7)$$

where $\bar{Y}$ is the mean of Y.

5. Mathematical Formulation

5.1. Long Short-Term Memory (LSTM) Model

Long Short-Term Memory (LSTM) networks, introduced by Hochreiter & Schmidhuber [14], are designed to overcome the limitations of traditional Recurrent Neural Networks (RNNs). An LSTM comprises a cell that maintains information over extended intervals and three gates: input, output, and forget gates, which control the flow of information. This architecture enables LSTMs to effectively capture long-term dependencies in data sequences.

- **Forget Gate**: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$. This gate uses the sigmoid function σ to decide which information is discarded from the cell state, with output values ranging from 0 (completely forget) to 1 (completely retain).
- **Input Gate**: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$; $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$. This gate updates the cell state, determining which values to update (via the sigmoid function) and generating new candidate values (via the tanh function).
- **Cell State Update**: $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$. The cell state is updated by combining the old state (modulated by the forget gate) with the new candidate values (modulated by the input gate).
- **Output Gate**: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$; $h_t = o_t * \tanh(C_t)$. This gate determines the part of the cell state to be outputted.

5.2. Vanishing Gradient Problem and LSTM

LSTMs address the vanishing gradient problem common in standard RNNs through their gate-based architecture. This ensures a stable gradient over many time steps, facilitating the learning of long-term dependencies in complex sequential tasks.

5.3. Simple Recurrent Neural Network (RNN) Model

Simple RNNs represent a more basic form of recurrent networks. They use a simpler structure with fewer parameters compared to LSTMs or GRUs, but have limitations in capturing long-term dependencies.

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h) \quad (8)$$

where h_t is the hidden state at time t, with W_{hh} and W_{xh} as weight matrices, and b_h as a bias term. The tanh activation function helps regulate the output values between -1 and 1.

$$y_t = W_{hy} h_t + b_y \quad (9)$$

Here, y_t is the output at time t, with W_{hy} as the weight matrix and b_y as the bias for the output layer.

5.4. Limitations of Simple RNNs

Simple RNNs are less effective in tasks requiring an understanding of extended sequence contexts due to the vanishing gradient problem. This limits their applicability in complex tasks such as language modeling and text generation.

5.5. Gated Recurrent Unit (GRU) Model

5.5.1 Architecture

The Gated Recurrent Unit (GRU) model, an advancement over traditional RNNs, simplifies the LSTM structure. It combines the forget and input gates into a single update gate and merges the cell state with the hidden state, reducing complexity.

- **Reset Gate**: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$. This gate determines the extent to which past information is forgotten.
- **Update Gate**: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$. This gate decides how much of the past information is needed for the current time step.
- **Current Memory Content**: $\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b)$. It generates the candidate hidden state for the current time step.
- **Final Memory**: $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$. The new hidden state is updated by combining the past hidden state with the current candidate hidden state.

5.5.2 Advantages Over Traditional RNNs

GRUs provide a more efficient training process with fewer parameters than LSTMs, yet offer comparable performance, especially in tasks where long-term dependencies are crucial.

5.6. Bidirectional LSTM (Bi-LSTM) Model

A Bidirectional LSTM (Bi-LSTM) processes data in both forward and backward directions, offering a comprehensive understanding of the context in tasks where both past and future information is relevant.

5.7. Custom and Hypertuned RNN Models

5.7.1 Custom RNN Model Architecture

Custom RNN models are tailored to specific problems by adjusting parameters like the number of layers, units per layer, and types of RNN cells, optimizing them for particular applications.

5.7.2 Hypertuning RNN Models

Hypertuning involves experimenting with various hyperparameters such as layer count, unit number, and learning rate to identify the most effective configuration for a specific task.

6. Statistical Models in Time Series Analysis

6.1. ARMA Model

The ARMA model is fundamental in time series analysis for modeling stationary data. It combines autoregressive (AR) and moving average (MA) components.

- AR Part: It models the current value as a linear combination of its previous values, shown as $X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + ... + \alpha_p X_{t-p} + \epsilon_t$, with α_i being the model coefficients and ϵ_t representing white noise.
- MA Part: It models the current value as a linear combination of the past error terms, expressed as $\epsilon_t = \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} + \eta_t$, where θ_i are the coefficients and η_t is white noise.

6.2. GARCH Model

The GARCH model is prevalent in financial market analysis for estimating and predicting volatility. It models the time-varying variance of financial time series.

- Variance Equation: $\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + ... + \alpha_p \epsilon_{t-p}^2 + \beta_1 \sigma_{t-1}^2 + ... + \beta_q \sigma_{t-q}^2$, where σ_t^2 is the conditional variance, and α_i, β_i are the model coefficients.

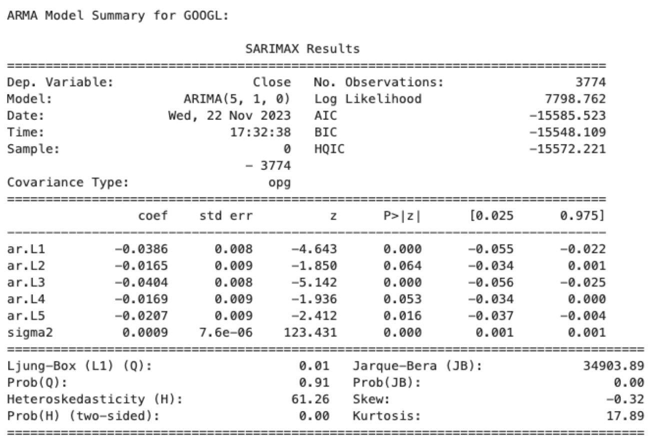
GARCH models are essential for managing financial risks and optimizing portfolios by forecasting the volatility of financial variables.

7. Results

7.1. Statistical Analysis of Stock Data

The statistical analysis was foundational in understanding the stock price movements. ARMA and GARCH models were applied to the historical stock data of companies like Google (GOOGL), Microsoft (MSFT), and Apple (AAPL), revealing distinct patterns. The ARMA model, designed to capture autocorrelations, was optimized based on AIC values, which highlighted its suitability for stationary sequences. The GARCH model effectively illustrated volatility, particularly evident during market tumults, as reflected by pronounced spikes in the model's volatility estimates.

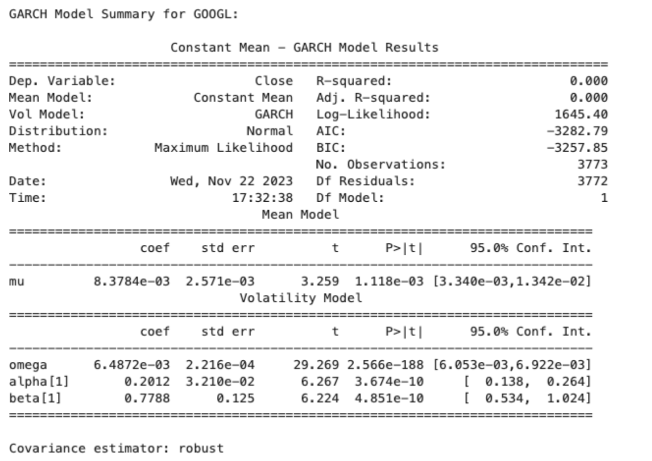
For instance, the ARMA and GARCH model residuals for Google's stock (see Figure 9) display stability over time, with residuals hovering close to zero, indicating that the model has captured the data's patterns well. However, the GARCH model's volatility plot shows significant peaks corresponding to known periods of market volatility, such as during the 2008 financial crisis and the COVID-19 pandemic.



```
ARMA Model Summary for GOOGL:

                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 3774
Model:                 ARIMA(5, 1, 0)   Log Likelihood                7798.762
Date:                Wed, 22 Nov 2023   AIC                         -15585.523
Time:                        17:32:38   BIC                         -15548.109
Sample:                             0   HQIC                        -15572.221
                               - 3774
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0386      0.008     -4.643      0.000      -0.055      -0.022
ar.L2         -0.0165      0.009     -1.850      0.064      -0.034       0.001
ar.L3         -0.0404      0.008     -5.142      0.000      -0.056      -0.025
ar.L4         -0.0169      0.009     -1.936      0.053      -0.034       0.000
ar.L5         -0.0207      0.009     -2.412      0.016      -0.037      -0.004
sigma2         0.0009    7.6e-06    123.431      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             34903.89
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):              61.26   Skew:                            -0.32
Prob(H) (two-sided):                  0.00   Kurtosis:                        17.89
===================================================================================
```

Figure 7. ARMA model residuals for GOOGL indicating model's fit



```
GARCH Model Summary for GOOGL:

                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                  Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                1645.40
Distribution:                  Normal   AIC:                          -3282.79
Method:            Maximum Likelihood   BIC:                          -3257.85
                                        No. Observations:                 3773
Date:                Wed, Nov 22 2023   Df Residuals:                     3772
Time:                        17:32:38   Df Model:                            1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu         8.3784e-03  2.571e-03      3.259  1.118e-03 [3.340e-03,1.342e-02]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      6.4872e-03  2.216e-04     29.269 2.566e-188 [6.053e-03,6.922e-03]
alpha[1]       0.2012  3.210e-02      6.267  3.674e-10     [  0.138,  0.264]
beta[1]        0.7788      0.125      6.224  4.851e-10     [  0.534,  1.024]
============================================================================

Covariance estimator: robust
```

Figure 8. GARCH model for GOOGL

Similarly, the analysis of Microsoft's stock (Figure 12) shows the ARMA model's residuals displaying minimal variance around zero, suggesting a good fit. The GARCH volatility plot for MSFT illustrates acute spikes, signifying the model's sensitivity to market volatility.

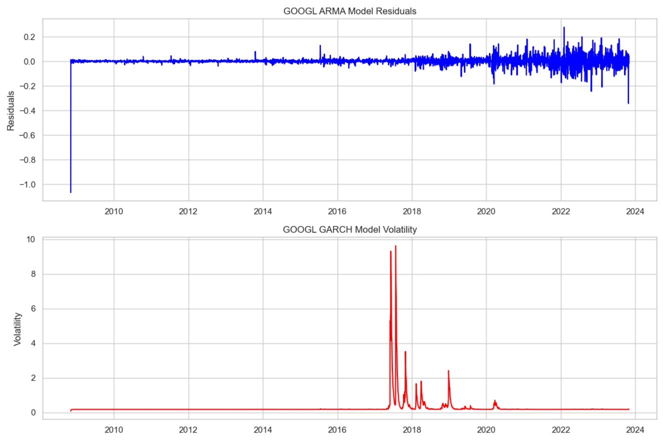
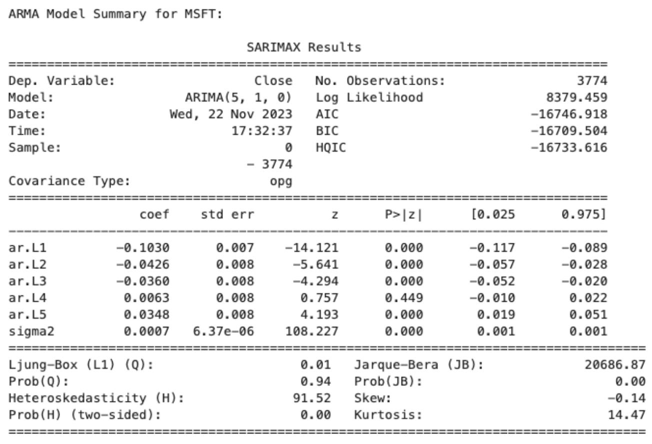


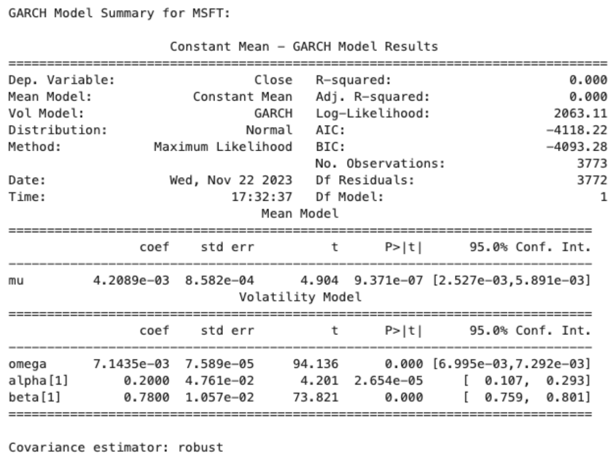
Figure 9. Top: ARMA model residuals for GOOGL indicating model's fit over time. Bottom: GARCH model's volatility for GOOGL showing spikes during periods of market instability.



```
ARMA Model Summary for MSFT:

                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 3774
Model:                 ARIMA(5, 1, 0)   Log Likelihood                8379.459
Date:                Wed, 22 Nov 2023   AIC                         -16746.918
Time:                        17:32:37   BIC                         -16709.504
Sample:                             0   HQIC                        -16733.616
                               - 3774
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1030      0.007    -14.121      0.000      -0.117      -0.089
ar.L2         -0.0426      0.008     -5.641      0.000      -0.057      -0.028
ar.L3         -0.0360      0.008     -4.294      0.000      -0.052      -0.020
ar.L4          0.0063      0.008      0.757      0.449      -0.010       0.022
ar.L5          0.0348      0.008      4.193      0.000       0.019       0.051
sigma2         0.0007   6.37e-06    108.227      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             20686.87
Prob(Q):                              0.94   Prob(JB):                         0.00
Heteroskedasticity (H):              91.52   Skew:                            -0.14
Prob(H) (two-sided):                  0.00   Kurtosis:                        14.47
===================================================================================
```

Figure 10. ARMA model for MSFT.



```
GARCH Model Summary for MSFT:

                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                  Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                2063.11
Distribution:                  Normal   AIC:                          -4118.22
Method:            Maximum Likelihood   BIC:                          -4093.28
                                        No. Observations:                 3773
Date:                Wed, Nov 22 2023   Df Residuals:                     3772
Time:                        17:32:37   Df Model:                            1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu         4.2089e-03  8.582e-04      4.904  9.371e-07 [2.527e-03,5.891e-03]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      7.1435e-03  7.589e-05     94.136      0.000 [6.995e-03,7.292e-03]
alpha[1]       0.2000  4.761e-02      4.201  2.654e-05     [  0.107,  0.293]
beta[1]        0.7800  1.057e-02     73.821      0.000     [  0.759,  0.801]
============================================================================

Covariance estimator: robust
```

Figure 11. GARCH model for MSFT

For Apple's stock (Figure 15), the ARMA model residuals suggest that the model has effectively captured the

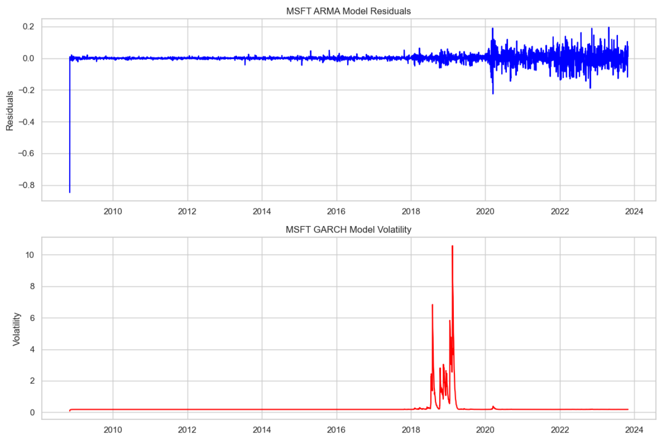
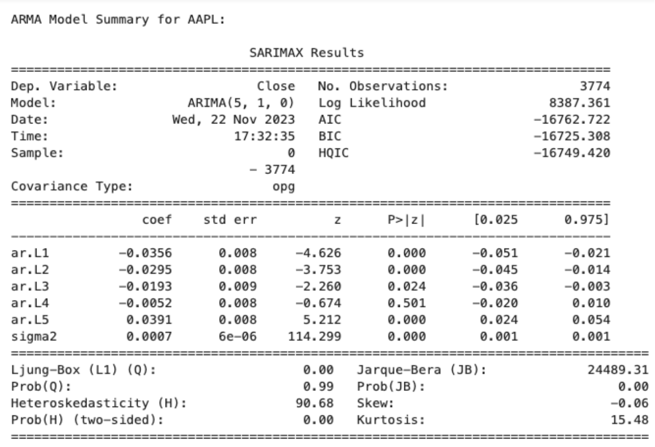


Figure 12. Top: ARMA model residuals for MSFT. Bottom: GARCH model volatility for MSFT, with spikes illustrating periods of high market volatility.

stock's movements, with residuals forming a tight band around zero. The GARCH volatility graph shows heightened volatility during specific periods, aligning with known economic events that influenced the stock market.

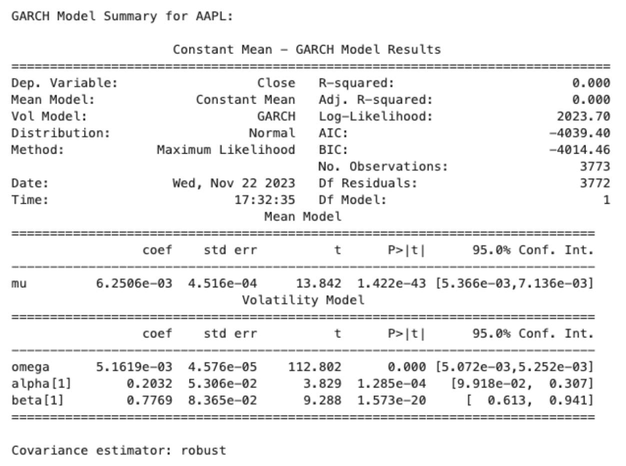


```
ARMA Model Summary for AAPL:

                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 3774
Model:                 ARIMA(5, 1, 0)   Log Likelihood                8387.361
Date:                Wed, 22 Nov 2023   AIC                         -16762.722
Time:                        17:32:35   BIC                         -16725.308
Sample:                             0   HQIC                        -16749.420
                               - 3774
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0356      0.008     -4.626      0.000      -0.051      -0.021
ar.L2         -0.0295      0.008     -3.753      0.000      -0.045      -0.014
ar.L3         -0.0193      0.009     -2.260      0.024      -0.036      -0.003
ar.L4         -0.0052      0.008     -0.674      0.501      -0.020       0.010
ar.L5          0.0391      0.008      5.212      0.000       0.024       0.054
sigma2         0.0007      6e-06    114.299      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):             24489.31
Prob(Q):                              0.99   Prob(JB):                         0.00
Heteroskedasticity (H):              90.68   Skew:                            -0.06
Prob(H) (two-sided):                  0.00   Kurtosis:                        15.48
===================================================================================
```

Figure 13. ARMA model for AAPL, demonstrating the model's performance

The performance metrics for these models are summarized in the table below, which presents a comparative analysis of the ARMA and GARCH models across different stocks based on their AIC values and residual statistics.



```
GARCH Model Summary for AAPL:

                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                    Close   R-squared:                       0.000
Mean Model:               Constant Mean   Adj. R-squared:                  0.000
Vol Model:                        GARCH   Log-Likelihood:                2023.70
Distribution:                    Normal   AIC:                          -4039.40
Method:              Maximum Likelihood   BIC:                          -4014.46
                                          No. Observations:                 3773
Date:                  Wed, Nov 22 2023   Df Residuals:                     3772
Time:                          17:32:35   Df Model:                            1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu         6.2506e-03  4.516e-04     13.842  1.422e-43 [5.366e-03,7.136e-03]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      5.1619e-03  4.576e-05    112.802      0.000 [5.072e-03,5.252e-03]
alpha[1]       0.2032  5.306e-02      3.829  1.285e-04   [9.918e-02,  0.307]
beta[1]        0.7769  8.365e-02      9.288  1.573e-20     [  0.613,  0.941]
============================================================================

Covariance estimator: robust
```

Figure 14. GARCH model for AAPL

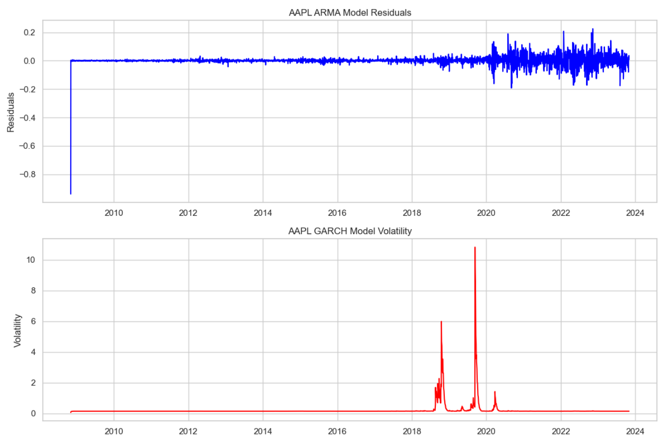


Figure 15. ARMA and GARCH model analysis for AAPL. The residuals and volatility plots demonstrate the models' performance through different market conditions.

Table 3. Statistical Model Performance Metrics (AIC, BIC)

Stock	Model	AIC	BIC
AAPL	ARMA(5,1,0)	-16762.72	-16725.31
AAPL	GARCH(1,1)	-4039.40	-4014.46
MSFT	ARMA(5,1,0)	-16746.92	-16709.50
MSFT	GARCH(1,1)	-4118.22	-4093.28
GOOGL	ARMA(5,1,0)	-15585.52	-15548.11
GOOGL	GARCH(1,1)	-3282.79	-3257.85

Table 4. Statistical Model Performance Metrics (HQIC, Residual Std)

Stock	Model	HQIC	Residual Std
AAPL	ARMA(5,1,0)	-16749.42	1.00
AAPL	GARCH(1,1)	-4026.93	1.00
MSFT	ARMA(5,1,0)	-16733.62	1.00
MSFT	GARCH(1,1)	-4105.75	1.00
GOOGL	ARMA(5,1,0)	-15572.22	1.00
GOOGL	GARCH(1,1)	-3270.32	1.00

7.2. Preliminary RNN Model Performance

Our investigation then shifted to Recurrent Neural Networks (RNNs), particularly the LSTM, GRU, and Bidirectional LSTM models. These were evaluated for their ability to predict stock prices, leveraging their inherent capacity to capture temporal dependencies. The GRU model consistently outperformed its counterparts, a likely result of its efficient gating mechanisms, which modulate information flow without the additional parameters of the LSTM. This observation is supported by a correlation matrix and boxplots that display the distribution of error metrics (MSE, MAE, RMSE) across different models, as shown in Figures 16 and 17. Additionally, a comparison of RMSE across models and stocks, as depicted in Figure 18, highlights the models' varying predictive accuracies.

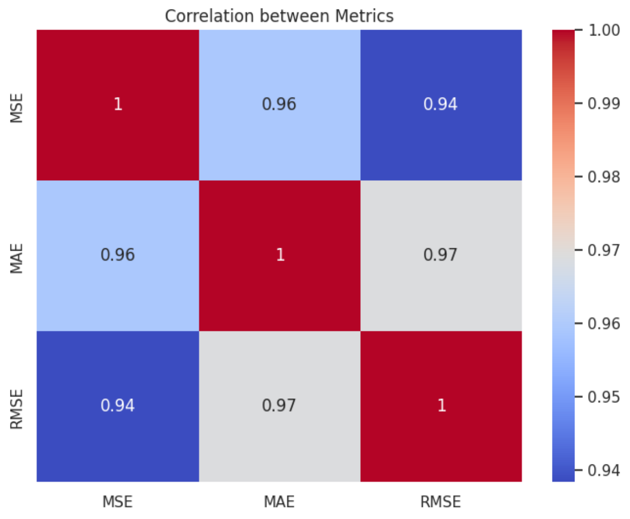


Figure 16. Correlation matrix between different performance metrics. High correlation coefficients suggest a strong relationship between the metrics.

7.3. Hypertuned Model Performance

The process of hypertuning the models resulted in significant performance enhancements. This was particularly evident in the GRU models, where a marked reduction in

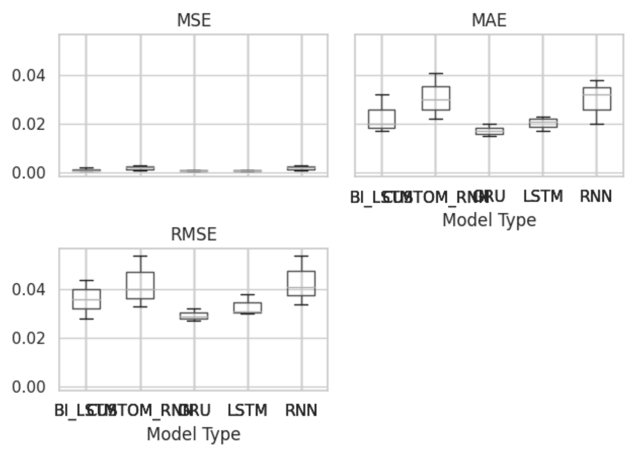


Figure 17. Boxplots displaying the distribution of MSE, MAE, and RMSE across different RNN models. The central line represents the median value, while the box spans the interquartile range.

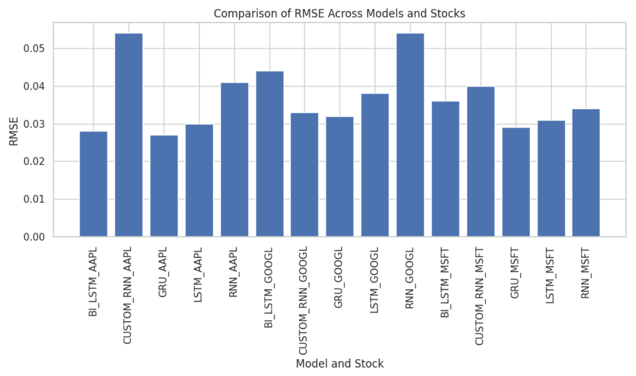


Figure 18. Comparison of RMSE across different models and stocks. The RMSE values are indicative of the models' predictive accuracy, with lower values representing better performance.

error metrics was observed post-hypertuning. Table 5 provides a comprehensive view of the hypertuned performance metrics, highlighting the improvements achieved from the preliminary models.

7.4. Comparative Analysis of Models

Our analysis further delved into comparing the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Squared Error (MSE) across all hypertuned models. These comparisons, illustrated in Figures 19, 20, and 21, not only shed light on each model's predictive accuracy but also highlighted the areas where specific models excelled or fell short.

Table 5. RNN Performance Table

Model	MSE	MAE	RMSE	R2
LSTM_MSFT	16.32	2.19	4.04	0.9982
LSTM_AAPL	5.77	1.36	2.40	0.9979
Basic_RNN_AAPL	5.64	1.41	2.38	0.9980
Basic_RNN_MSFT	14.10	2.32	3.76	0.9985
LSTM_GOOGL	4.41	1.23	2.10	0.9969
Basic_RNN_GOOGL	3.72	1.14	1.93	0.9974
GRU_AAPL	3.96	1.42	1.99	0.9986
GRU_MSFT	9.80	1.68	3.13	0.9989
GRU_GOOGL	2.79	1.14	1.67	0.9981
Bi_LSTM_AAPL	5.37	1.34	2.32	0.9981
Bi_LSTM_MSFT	23.07	2.76	4.80	0.9975
Bi_LSTM_GOOGL	4.12	1.24	2.03	0.9971
Custom_RNN_AAPL	3.73	1.18	1.93	0.9987
Custom_RNN_MSFT	13.03	1.96	3.61	0.9986
Custom_RNN_GOOGL	3.45	1.24	1.86	0.9976

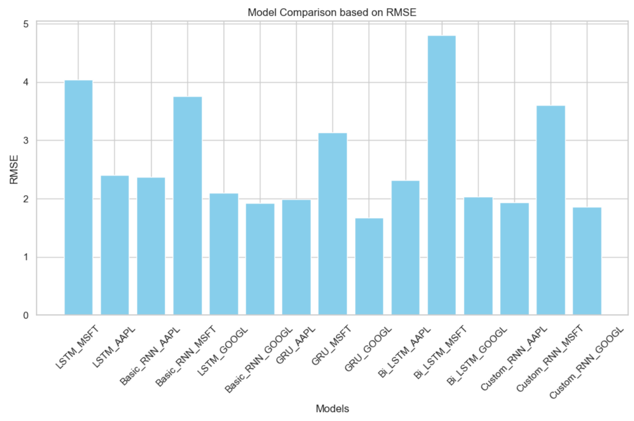


Figure 19. Comparison of Root Mean Square Error (RMSE) across hypertuned models. Lower RMSE values indicate better predictive accuracy.

8. Discussion

8.1. Insights from Statistical Models

The ARMA and GARCH model analyses provided valuable insights into stock price dynamics. The ARMA model's ability to capture autocorrelations was evident in the residuals, which largely centered around zero, indicating that the model successfully accounted for linear relationships in the data. The GARCH model's volatility plots highlighted periods of increased variance, correlating with known economic events affecting market stability.

The comparative analysis of these models, using both visual plots and performance metrics, has underscored their utility in financial econometrics. Despite the simplicity of these models, their robustness and interpretability offer a reliable baseline for understanding stock price movements,

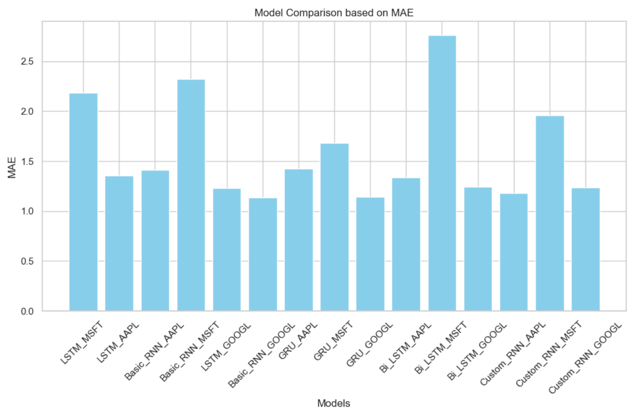


Figure 20. Comparison of Mean Absolute Error (MAE) across hypertuned models.

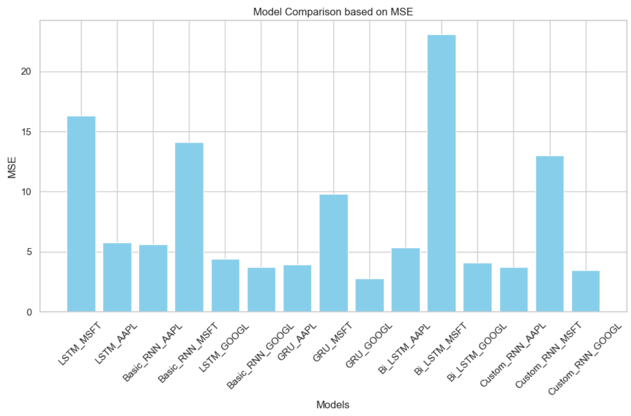


Figure 21. Comparison of Mean Squared Error (MSE) across hypertuned models.

which is crucial for investors and financial analysts in strategy formulation and risk assessment.

8.2. Evaluation of RNN Models

The study of RNN models, particularly their performance in predicting stock prices, reveals the nuanced capabilities of these deep learning architectures. The hypertuning process refined the RNN models' parameters, yielding improved accuracy and highlighting the importance of meticulous model optimization.

The GRU models, with their simplified gating mechanisms, emerged as the top performers, suggesting that efficiency in the model's structure is key to managing the complex temporal dependencies present in stock price data. The findings demonstrate the potential of RNNs in financial forecasting and the benefits of hyperparameter tuning in achieving high-performing models.

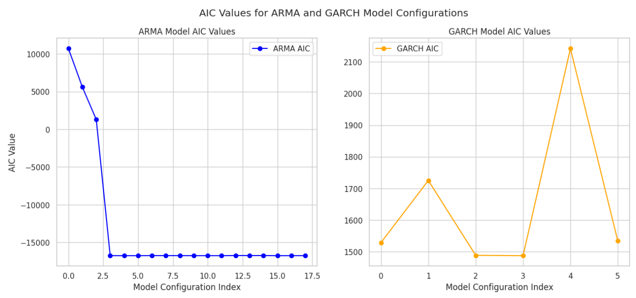


Figure 22. Performance of statistical models on basis of AIC values

8.3. The Role of Hypertuning

Hypertuning emerged as a pivotal step in model optimization, allowing us to fine-tune our models to achieve the best possible performance. This process was particularly impactful for RNN models, as seen in the substantial improvements in prediction accuracy post-hypertuning.

8.4. Comparing Model Performances

The comparative analysis of the models underscored the superiority of GRU in stock price forecasting, as evidenced by its lower error rates compared to other models. This comparison, detailed in Table 5, emphasizes the effectiveness of recurrent neural network architectures in financial forecasting and the importance of hyperparameter optimization.

8.5. AAPL Stock Analysis

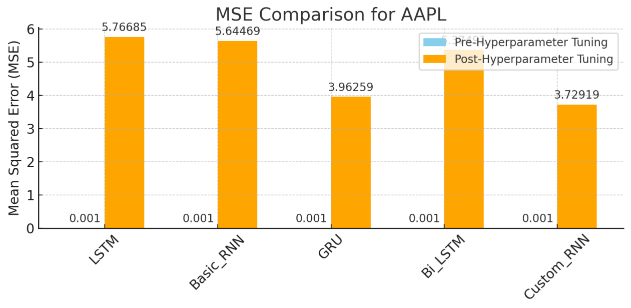


Figure 23. RNN Pre and Post Hypertuning Model Performance for AAPL

- All models showed a significant increase in MSE for AAPL stock post-tuning.
- LSTM and Custom RNN models had the highest initial and post-tuning MSE.
- This suggests suboptimal hyperparameter tuning for AAPL stock.

8.6. GOOGL Stock Analysis

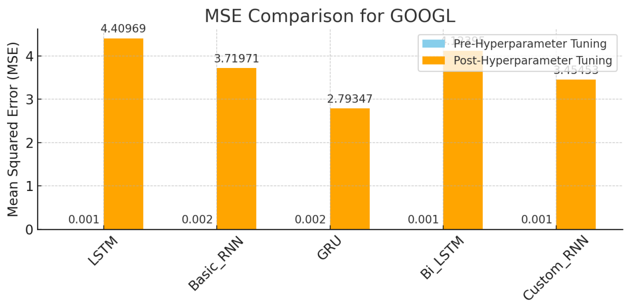


Figure 24. RNN Pre and Post Hypertuning Model Performance for GOOGL

- Similar trends were observed for GOOGL, with an overall increase in MSE post-tuning.
- GRU and Basic RNN models had the highest MSE increases.
- Bi-LSTM showed a substantial increase, indicating potentially inappropriate tuning parameters.

8.7. MSFT Stock Analysis

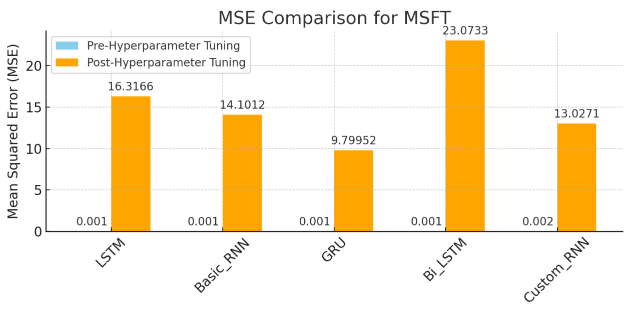


Figure 25. RNN Pre and Post Hypertuning Model Performance for MSFT

- MSFT stock models also exhibited increased MSE after tuning.
- The Bi-LSTM model demonstrated the most significant increase in MSE.
- LSTM and Custom RNN models also showed increased MSE but to a lesser extent.

9. Future Stock Price Prediction

Predicting stock prices is an inherently complex task due to the volatile and non-deterministic nature of the financial markets. Our approach leverages deep learning models to predict future stock prices based on historical data, aiming to minimize the residuals between the actual and predicted prices. This section presents the results of price prediction for three major stocks: AAPL, MSFT, and GOOGL.

9.1. Residual Price Prediction

Predicting residual stock prices is an intricate process aimed at determining the discrepancies between actual and forecasted stock values. In our study, we employ deep learning architectures to predict such residuals for AAPL, MSFT, and GOOGL stocks. This section presents a detailed analysis of the residual predictions, performance metrics, and the implications for future price trajectories.

9.2. Residual Analysis

The residual plots for AAPL, MSFT, and GOOGL, as shown in Figures 26, 27, and 28, demonstrate the model's accuracy in predicting stock price movements over time. Ideally, residuals should be distributed randomly around the horizontal axis, indicating that the model's predictions are unbiased and that its errors are stochastic.

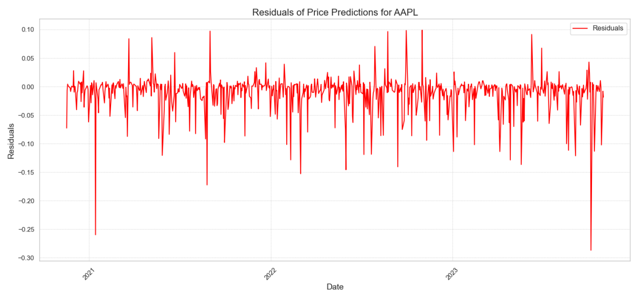


Figure 26. Residuals of price predictions for AAPL.

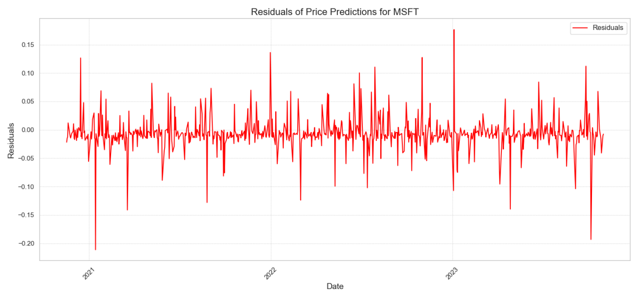


Figure 27. Residuals of price predictions for MSFT.

The presence of a few outliers indicates instances where the model's predictions diverged significantly from the actual stock prices, potentially corresponding to periods of high market volatility or external economic shocks.

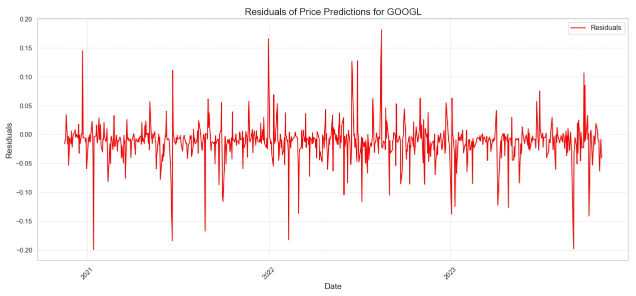


Figure 28. Residuals of price predictions for GOOGL.

9.3. Performance Metrics

The performance of our models is quantitatively summarized in the following table, which reports the Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared Score, Training Duration, and Final Validation Loss for each stock.

Stock	MSE	MAE	RMSE	R^2 Score
AAPL	0.001231	0.018658	0.035087	99.87%
MSFT	0.000935	0.018654	0.030575	99.9%
GOOGL	0.001236	0.021512	0.035151	99.87%

Table 6. Performance metrics for residual price prediction models.

Stock	Training Duration	Final Val. Loss
AAPL	16.85 s	0.000119
MSFT	13.09 s	0.000076
GOOGL	9.67 s	0.000104

Table 7. Training-related metrics for residual price prediction models.

The R-squared scores nearing 100% across all models suggest a strong fit to the historical data. However, despite high R-squared values, the significance of the MSE, MAE, and RMSE cannot be overlooked as they offer insights into the average errors in the models' predictions. The low training duration and validation loss point towards an efficient training process and model convergence.

9.4. Future Price Predictions

Based on the established models, we projected the future prices for the stocks as follows:

Stock	Predicted Price
AAPL	0.897295
MSFT	0.946274
GOOGL	0.845311

Table 8. Predicted future prices based on residual price prediction models.

The predicted prices in Table 8 should be interpreted with cautious optimism. While our models are statistically robust, predictions in the financial domain are inherently subject to uncertainty due to their dependence on a myriad of unpredictable economic factors.

10. Conclusion

Our extensive analysis using ARMA and GARCH models on stock data for companies like AAPL, MSFT, and GOOGL unveiled crucial insights into stock price dynamics. The ARMA model effectively captured autocorrelations, while the GARCH model highlighted periods of heightened volatility during significant market events. These models provide a strong baseline for understanding stock price movements and are valuable for investors and analysts in risk assessment and strategy formulation.

Shifting our focus to Recurrent Neural Networks (RNNs), we observed that GRU models consistently outperformed other RNN architectures in stock price prediction. This superiority can be attributed to their efficient gating mechanisms, which manage complex temporal dependencies in the data. Hypertuning further improved model performance, particularly for GRU models.

Our comparative analysis of RMSE, MAE, and MSE across all models emphasized the effectiveness of GRU models in stock price forecasting, highlighting the potential of RNNs in financial prediction tasks.

In summary, our study has shed light on the strengths of both traditional statistical models and deep learning techniques in the context of stock price prediction. While statistical models provide interpretability and robustness, RNNs offer superior predictive performance. These findings contribute to our understanding of the nuanced interplay between statistical and deep learning models in financial forecasting.

11. Future Work

The current research represents a foundational step in the application of Recurrent Neural Networks for stock price prediction. Looking ahead, there are several promising directions for future work, which could further enhance the understanding and effectiveness of these models in financial forecasting.

11.1. Integration of Alternative Data Sources

Future studies should explore the integration of diverse data sources, such as social media sentiment and economic indicators, alongside traditional financial data. The pioneering work by Bollen et al. (2011) on the correlation between Twitter sentiment and stock market trends exemplifies the potential of incorporating social media data in predictive models [26]. Additionally, Nguyen et al. (2015) further support the exploration of sentiment analysis in financial forecasting, highlighting the need for more comprehensive data integration [27].

11.2. Advanced Deep Learning Architectures

The exploration of advanced neural network architectures, such as Transformers, represents a significant opportunity. The Transformer model, proposed by Vaswani et al. (2017), which relies entirely on an attention mechanism and eschews recurrence, offers a new paradigm for handling sequential financial data [28]. Investigating how Transformer models can be adapted and optimized for stock market prediction could yield substantial advancements in this field.

11.3. Reinforcement Learning and Adaptive Models

Applying reinforcement learning (RL) for dynamic and adaptive stock price prediction presents a promising research avenue. The work by Mnih et al. (2015) on deep reinforcement learning, achieving human-level control in complex environments, lays the groundwork for applying RL in the financial domain [29]. Future research could develop RL models that adapt to changing market conditions, potentially enhancing predictive accuracy.

11.4. Interpretable and Explainable AI Models

As machine learning models become more complex, their interpretability becomes increasingly crucial, especially in high-stakes domains like finance. Lipton (2018) discusses the challenges and importance of interpretability in AI, underscoring the need for models that not only predict accurately but also provide insights into their decision-making processes [30]. Future work should focus on developing interpretable and transparent financial prediction models, enhancing their reliability and trustworthiness.

11.5. Real-time Analysis and Scalability

There is a need for real-time analysis capabilities and scalable models that can adapt to varying market conditions. Future research should focus on developing models that can process real-time data streams and maintain robustness across different market scenarios, enhancing their practical applicability in the financial industry.

11.6. Global Market Dynamics

Expanding the research to include global market dynamics and inter-market relationships could offer a more comprehensive understanding of stock price movements. Investigating the impact of global events and the interconnected nature of financial markets could reveal new predictive insights and enhance the accuracy of existing models.

These directions not only promise to refine the application of RNNs in stock price prediction but also aim to advance the broader field of financial analysis and forecasting.

12. Code Availability

We believe in the importance of reproducibility and transparency in research. To facilitate further exploration and utilization of our models and methodologies, we have made our code openly available on GitHub. You can access the code repository at the following link:

https://github.com/variyas31/a3DL/blob/main/Assignment%203-%20Variyas.ipynb

We encourage researchers, practitioners, and enthusiasts to use this resource for future investigations, extensions, and applications in the domain of stock price prediction.

References

[1] E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work," *Journal of Finance*, vol. 25, no. 2, pp. 383-417, 1970. 2

[2] B. G. Malkiel, "The Efficient Market Hypothesis and Its Critics," *Journal of Economic Perspectives*, vol. 17, no. 1, pp. 59-82, Winter 2003. 1, 2

[3] B. Graham, *The Intelligent Investor: The Definitive Book on Value Investing*. Harper Business Essentials, 2008. 1, 2

[4] R. D. Edwards, J. Magee, and W. H. C. Bassetti, *Technical Analysis of Stock Trends*. CRC Press, 2007. 2

[5] J. J. Murphy, *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance, 1999. 1, 2

[6] A. W. Lo and A. C. MacKinlay, *A Non-Random Walk Down Wall Street*. Princeton University Press, 2000. 1, 2

[7] L. K. C. Chan and J. Lakonishok, "Value and Growth Investing: Review and Update," *Financial Analysts Journal*, vol. 59, no. 1, pp. 71-86, 2003. 1, 2

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995. 2

[9] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986. 2

[10] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309-317, 2001. 1, 2

[11] I. Kaastra and M. Boyd, ”Designing a neural network for forecasting financial and economic time series,” *Neurocomputing*, vol. 10, no. 3, pp. 215-236, 1996. 1, 2

[12] J. L. Elman, ”Finding structure in time,” *Cognitive Science*, vol. 14, pp. 179-211, 1990. 1, 2

[13] J. T. Connor, R. D. Martin, and L. E. Atlas, ”Recurrent neural networks and robust time series prediction,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 240-254, 1994. 1, 2

[14] S. Hochreiter and J. Schmidhuber, ”Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997. 1, 2, 6

[15] K. Cho et al., ”Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,” *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724-1734, 2014. 1, 2

[16] F. A. Gers, J. Schmidhuber, and F. Cummins, ”Learning to Forget: Continual Prediction with LSTM,” *Neural Computation*, vol. 12, no. 10, pp. 2451-2471, 2000. 1, 2

[17] L. Prechelt, ”Early Stopping - But When?,” in *Neural Networks: Tricks of the Trade*, Springer, 1998. 2

[18] Z. C. Lipton, ”The Mythos of Model Interpretability,” *Queue*, vol. 16, no. 3, pp. 31-57, 2018. 2, 14

[19] M. Kearns and Y. Mansour, ”A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization,” *Proceedings of the Fifteenth International Conference on Machine Learning (ICML ’98)*, 1998. 2

[20] R. S. Tsay, *Analysis of Financial Time Series*. Wiley-Interscience, 2005. 2

[21] J. Bollen, H. Mao, and X. Zeng, ”Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1-8, 2011. 2, 13

[22] T. H. Nguyen and E. Shirazi, ”Sentiment Analysis for Stock Market Prediction Using a Deep Neural Network,” *Procedia Computer Science*, vol. 72, pp. 343-350, 2015. 2, 14

[23] V. Mnih et al., ”Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529-533, 2015. 2, 14

[24] L. Zhang, C. Aggarwal, and G.-J. Qi, ”Stock price prediction via discovering multi-frequency trading patterns,” *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2141-2149, 2017. 2

[25] J. Zou et al., ”Stock Market Prediction via Deep Learning Techniques: A Survey,” 2022. [Online]. Available: https://ar5iv.org/abs/2212.12717. 1, 2

[26] J. Bollen, H. Mao, and X. Zeng, ”Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1-8, 2011. 2, 13

[27] T. H. Nguyen and E. Shirazi, ”Sentiment analysis on stock social media for stock price movement prediction,” 2015. 2, 14

[28] A. Vaswani et al., ”Attention Is All You Need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000-6010, 2017. 14

[29] V. Mnih, K. Kavukcuoglu, D. Silver, et al., ”Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, 2015. 2, 14

[30] Z. C. Lipton, ”The Mythos of Model Interpretability,” *ACM Queue*, vol. 16, no. 3, 2018. 2, 14