**Introduction**

The movements of the stock market can be incredibly complex and seemingly random. How the stock market behaves has been a subject of extensive research that attempt to predict patterns that govern its movements. Researchers have turned to various methodologies, including but not limited to Natural Language Processing (NLP), machine learning (specifically neural networks), and statistical time series methods such as AutoRegressive Integrated Moving average (ARIMA) or Generalized AutoRegressive Conditional Heteroscedasticity (GARCH). This research paper will attempt to explore these three distinct approaches, seeking to evaluate their individual strengths and weaknesses in the domain of stock market prediction.

The unpredictability of financial markets is a challenge for both investors and researchers. Traditional models often fall short in capturing the intricate dynamics influenced by global events, sentiment shifts, and macroeconomic factors. As a result, there is a need to evaluate the effectiveness of cutting-edge techniques such as NLP, machine learning, time series, in enhancing the accuracy and reliability of stock market predictions. This research aims to address this gap in understanding, providing insights into the comparative performance of these methodologies and their potential implications for investment decision-making.

Accurate stock market prediction is not only an academic pursuit but also an important area of research with far-reaching consequences for investors, financial institutions, and the broader economy. Inaccurate forecasts can lead to financial losses, market instability, and economic downturns. By focusing on NLP, machine learning, time series, this research aims to contribute valuable knowledge that could reshape how analysts approach stock market forecasting, potentially paving the way for more informed and strategic investment decisions.

This research draws upon an extensive array of resources, including historical financial data, textual information from news articles and financial reports, and advanced machine learning algorithms. Leveraging these diverse sources allows for a holistic analysis that incorporates both quantitative and qualitative factors, providing a comprehensive understanding of the methodologies under scrutiny. Additionally, the study will utilize state-of-the-art tools and frameworks, ensuring the robustness and reproducibility of the findings. The combination of these resources forms the foundation for a rigorous and insightful investigation into the effectiveness of NLP, machine learning, and time series in predicting stock market trends.

Literature Review

For this comprehensive literature review, a variety of articles exploring the prediction of

future trends within the stock market were read. The review itself was structured around

three principal themes. The first theme delved into feature engineering concepts via the

use of technical indicators. The use of technical indicators serves to expand the amount of

information available for potential model development when they are used correctly.

The subsequent themes revolved around the development of predictive models. One

avenue involved the utilization of recurrent neural networks (RNNs) and long short-term

memory (LSTM) architectures, while the other avenue focused on autoregressive moving

averages (ARMA). Exploring robust feature engineering techniques within technical

indicators alongside the development of strong predictive frameworks such as LSTMs and

ARMA models serves as the foundation of this literature review, and it will provide the basis

of our methodologies and research going forward in this project.

**Technical Analysis of Stocks**

The literature reviews provided in the five papers offer a comprehensive understanding of

the role and effectiveness of technical indicators in stock market analysis and prediction.

They collectively emphasize the significance of accurate forecasting using the technical

indicators in financial decision-making of a retail investor for large cap companies which

have high liquidity in the market and highlight the challenges posed by market

unpredictability.

Paper: Application of Neural Network to Technical Analysis of Stock Market Prediction

delves into the application of neural networks, particularly LSTM, in forecasting stock market

trends. It introduces innovative models like the Deep LSTM Neural Network with Embedded

Layer and demonstrates promising results in predicting stock price movements. The

authors stress the potential benefits of using deep learning approaches for stock market

prediction and suggest avenues for further research.

Similarly, Paper: Enhancement of Stock Market Forecasting Using a Technical

Analysis-based Approach presents a systematic framework for enhancing stock market

forecasting through technical analysis. It highlights the practical importance of choosing

the time frame of investment as per the goal and need of retail investor for accurate

forecasting, particularly in regions like Taiwan. The proposed method, based on trend-based

classification and adaptive indicator selection, shows superior performance compared to

existing approaches, thereby facilitating better decision-making and increased profitability

for investors.

Paper: Stock Price Prediction with Golden Cross and Death Cross on Technical Analysis

Indicators Using Long Short-Term Memory focuses on the application of LSTM in predicting

stock prices using technical analysis indicators, specifically the golden cross and death

cross. It underscores the need for automated systems for objective and fast predictions and

demonstrates the enhanced prediction accuracy achieved by incorporating these features

with LSTM. The study suggests potential for further research in combining these features

with other prediction methods for even better results.

Finally, Paper: Technical Analysis of Three Stock Oscillators: Testing MACD, RSI and KDJ

Rules in SH & SZ Stock Markets explores the application of various technical analysis

indicators and machine learning techniques in predicting stock price movements. It

provides a comprehensive overview of previous studies and methodologies while

showcasing the effectiveness of indicators like RSI, MACD, decision trees, and neural

networks. The paper identifies areas for future research, including combining technical

analysis with news sentiment analysis and considering external events in predictive models,

to further improve forecasting accuracy and profitability.

In summary, these literature reviews provide valuable perspectives on the changing terrain

of stock market forecasting, offering researchers and investors new methodologies and

paths for deeper investigation in technical analysis and machine learning.

ARMA Model

The literature review on statistical methods in forecasting stock prices examines a study

conducted by Huanze Tang, focusing on predicting stock prices using the Autoregressive

Moving Average (ARMA) model. Tang's study explores the significance of stock price

forecasts for economic decision-making and highlights the ARMA model's role in capturing

the dynamics of financial time series. The review outlines the foundational principles of the

ARMA model, elucidating its integration of autoregressive and moving average components

to analyze stock price fluctuations.

Tang's analysis centers on Apple Inc.'s stock prices from January 2018 to January 2020. Tang

used the adjusted close price data, which was determined to be non-stationary. To solve for

this, the data was transformed by taking the log-differences of each price and the price one

period behind in order to make it stationary.

The ARMA(p,q) model was built by using the data to fit a model using all combinations of p

and q up to p = 5 and q = 5. The optimal model was chosen by comparing each model’s

Akaike information criterion (AIC) and choosing the model with the lowest score. The

optimal model in the case of the paper was an ARMA(4,4) model. The forecasting analysis

employs the ARMA(4,4) model to predict Apple Inc.'s adjusted closing prices for the next five

days. The results of the prediction demonstrate a low error rate, affirming the ARMA(4,4)

model's efficacy in short-term forecasting.

In discussing the implications of the study, Tang acknowledges the ARMA model's

limitations in capturing long-term trends amidst volatile market conditions. The review

suggests avenues for future research, including the exploration of hybrid models like

ARMA-GARCH and CARR to enhance predictive accuracy. Ultimately, Tang's study

contributes valuable insights to the field of financial time series forecasting, offering a

robust framework for informed decision-making in the realm of investment and finance.

RNN, LSTM

In recent years, using Long Short-Term Memory (LSTM) networks in stock prediction has

garnered substantial attention in financial forecasting. LSTM, a recurrent neural network

(RNN), has demonstrated remarkable efficacy in capturing and learning intricate temporal

dependencies, making it particularly well-suited for time-series prediction tasks. Within the

context of stock market prediction, LSTM models offer a promising avenue for investors,

analysts, and financial institutions seeking to exploit patterns and trends in historical

market data to anticipate future price movements. The below research utilized LSTM in

different ways

A research conducted by Naadun Sirimevan, I.G. U. H. Mamalgaha, Chandira Jayasekara, Y. S.

Mayuran, and Chandimal Jayawarden (ICAC, 2019)[1] involved a relatively comprehensive

approach to stock prediction methodology, encompassing data preprocessing, sentiment

analysis, and the utilization of machine learning models. Initially, data preprocessing tackles

null values in stock market data and missing values in web news datasets, employing

techniques like linear interpolation and sentiment analysis with TextBlob. Following this, the

discussion shifts towards employing Recurrent Neural Networks (RNNs), specifically LSTM,

for time series prediction, highlighting their efficacy. Input variables for forecasting models

include sentiment scores, web news sentiment, Google trends hit volume, and closed stock

price.

Furthermore, the integration and evaluation of various models for stock prediction are

detailed, focusing on Twitter sentiment, web news sentiment, and search engine query hits.

Ensemble methodology, specifically weighted average ensemble, integrates these models.

Most of the literature concludes by presenting correlation coefficients between sentiment

analysis data and stock market values and forecasting results for different time frames

using individual and ensemble models.

A research study in University of Massachusetts (ICBDCC, 2017)[2] Lowell involved collection

of data from four social media platforms in China - Weibo, Circle, Snowball, and Stock Bar -

over seven months in 2016, focusing on discussions related to stock and futures trading.

These platforms offer unique user behaviors and content styles. Circle exhibits the highest

activity level, especially during market trading hours, followed by Stock Bar, Snowball, and

Weibo. Weekday activity is higher than weekends across all platforms, with Circle showing

the closest correlation to market trading hours.

The stock price dataset covers 100 randomly chosen stocks from the Shanghai and

Shenzhen Stock Exchanges, capturing price movements within minutes, trading volume,

and price return. Preprocessing involved removing stopwords, URLs, and special characters,

while tokenization was adapted for Chinese text using the Jieba tool. Machine learning

algorithms such as logistic regression, SVM, NBSVM, LSTM, and ensemble methods were

explored for sentiment analysis, with the ensemble method achieving the highest accuracy.

Correlation analysis was used to study the correlation between post sentiment and trading

volume. Statistical hypothesis tests like Granger causality tests were performed to indicate

that post sentiment series Granger-cause stock price return and trading volume for most

stocks.

A prediction model utilizing post-sentiment features achieved an average prediction

accuracy of 0.573 for stock price movement without relying on traditional financial

indicators.

Research papers that highlight the models' adaptability to processing data from other

marketplaces, such as those that concentrate on the National Stock Exchange of India and

NASDAQ, demonstrate their worldwide application. The results of these studies

demonstrate how well LSTM and RNN models can capture intricate temporal relationships

in stock market data, which is important considering how volatile and time-sensitive

financial markets can be. Through examining different setups and approaches, the research

shows how to carefully maximize model performance. Experiments varying the number of

neurons, layers, and training epochs, for example, provide information about how the

architecture of the model affects prediction results. This meticulousness also applies to the

preprocessing and data collection phases, where the choice of pertinent features and

modification of input variables are critical to the effectiveness of the model. These studies'

results, which show accuracy levels of up to 97% and occasionally even close to 100%

(debatable), offer strong proof of machine learning's promise for financial forecasting.

Overall, With the recent AI advancements, the methodology to study sentiments can be

improved with the use of LLMs, and with the improvements in finding various data sources

and stock trading methodologies, day trading stocks can be made less volatile with the

addition of more indicators of different economic climates.

These researches serve as a fundamental baseline to the current research being performed.

ANN

The fusion of Artificial Neural Networks (ANNs) with macroeconomic indicators for stock

market forecasting marks a pivotal development in financial analysis. Incorporating

elements like inflation and interest rates into ANNs allows for a comprehensive

understanding of market forces, enhancing prediction precision. This methodology not only

sharpens forecasts but also deepens insights into economic influences on market trends.

Research underscores the efficacy of this integrated approach, suggesting it as a valuable

tool for investors and policy strategists alike. Furthermore, the intersection of ANNs with

financial technology (fintech) applications signals an exciting frontier for exploration.

Leveraging ANNs alongside big data analytics within fintech frameworks broadens the

scope of stock market analysis, marrying computational prowess with financial acumen.

This collaborative approach promises to refine investment strategies and enrich the domain

of market prediction research, ushering in a new era of informed decision-making.

In essence, the application of ANNs in predicting stock market directions showcases the

transformative potential of merging computational intelligence with financial analytics. By

adapting to complex data patterns through advanced learning algorithms, ANNs offer

unparalleled accuracy in forecasts. The research suggests optimization techniques like

genetic algorithms significantly bolster these predictions, presenting a paradigm shift in

how financial markets are analyzed and understood, and providing actionable insights for

market participants.

Methodology

Our objective is to rigorously examine a variety models and datasets with the aim of forecasting trends within the stock market. This will involve a comprehensive exploration of various predictive models and feature engineering techniques. We will outline step-by-step how we intend to build each model, with a detailed overview outlining its underlying principles and methodology. Similarly, we will outline the methods used for feature engineering, emphasizing how and why they are useful in order to strengthen our predictions.

The main goal is to assess how well classical machine learning methods, ARMA, LSTM, and RNN models can help predict the direction of stock prices. Based on the knowledge of stocks we know that the data that needs to be collected will be quantitative i.e., open price, close price, volume, average, etc. Further data can be added by calculating technical indicators that measure the momentum of price movements between periods. The project will include analyzing past patterns in stock prices and predict if the price will go in the positive direction or negative direction.

**Data Overview**

In order to employ our models to accurately forecast stock price direction, we concentrated on gathering quantitative, high-frequency data that accurately reflects the stock market's dynamic character.

This data was obtained from *poygon.io*, a reputable stock market API that can output data from various exchanges. Historical daily data was obtained from Yahoo Finance and Alpha Vantage, with the data gathered between a span of 2-years, from 2022-02-09 to 2024-01-09. With minute-level granularity available in this dataset, our trend analysis has a strong basis.

Several important financial factors that are necessary for assessing fluctuations in stock prices are included in our dataset. Among them are:

|  |  |
| --- | --- |
| **Column** | **Definition** |
| v | Volume, The trading volume of the symbol in the given time. |
| vw | Volume Weighted Average Price, an indicator of trading volume and price that provides the average price at which a stock has traded during the day. |
| c | The close price of the stock in the given time frame. |
| h | The highest price of the stock in the given time frame. |
| l | The lowest price of the stock in the given time frame. |
| n | The number of transactions in the time frame. |
| o | The open price of the stock at the beginning of the time period. |
| t | The Unix Msec timestamp for the start of the aggregate window. |
| datetime | Date and Time - The precise time and date the data point was captured, giving our study a clear sense of its temporal context. |

**Time Frame and Resolution:**

Taking into account data from a window of three to five days, the prediction model is intended to estimate the direction of stock prices over brief intervals. In order to accomplish this, we have chosen data that has a temporal resolution of between 30 and 1 hour, which is consistent with our goal of capturing the subtle, transient variations in stock prices. By using this method, we can be sure that our models are trained on comprehensive, high-quality data, which will allow us to anticipate stock price directions with greater accuracy.

**Feature Engineering – Technical Indicators**

Technical indicators are mathematical calculations based on historical price, volume or open interest data. These are mathematical tools used by traders or investors in financial markets to make informed decisions.

We will use following indicators:

1. Trend Indicators:
2. Exponential Moving Averages (EMA): This indictor helps in identifying trend of an asset’s price over a certain range of time.

The formula for calculating the Exponential Moving Average (EMA) involves using a smoothing factor and the previous day's EMA. The formula is as follows:

EMAt​=(1−α)×EMAt−1​+α×Pt

Where:

* EMAt is the EMA at time t,
* α is the smoothing factor, often calculated as 2 / (Period +1)
* EMAt−1 is the EMA from the previous day (or the initial value for the first calculation), and
* Pt is the closing price of the asset at time t.

To break it down further:

* ​(1−α)×EMAt−1 represents the previous day's EMA adjusted by the smoothing factor.
* α×Pt represents the current day's price adjusted by the smoothing factor.

The smoothing factor α determines the weight given to the most recent price relative to the historical EMA values. A smaller smoothing factor gives more weight to older prices, while a larger smoothing factor gives more weight to recent prices, making the EMA more responsive to recent price changes.

It's important to note that when calculating the initial EMA, there is no previous day's EMA, so the initial value is typically set to the first closing price. After the initial calculation, the formula is used iteratively for subsequent days.

1. Momentum Indicators:
   1. Moving Average Convergence Divergence (MACD): This indicator compares two moving averages of an asset's price to signal changes in momentum.

When two exponential moving averages of different periods of same asset crossover they indicate a convergence or divergence point and the trend beyond that point indicates the strength or weakness of that asset in that duration.

* 1. Relative Strength Index (RSI): Measures the speed and change of an asset’s price movements to identify overbought or oversold conditions.

The Relative Strength Index (RSI) is a momentum oscillator that measures the speed and change of price movements. The RSI is typically calculated using the following formula:

RSI = 100 – 100/{1 + RS}

Where:

* RS (Relative Strength) is the average of ‘n’ days' up closes divided by the average of ‘n’ days' down closes, where ‘n’ is the number of days used in the RSI calculation. Commonly, a 14-day period is used.

The steps to calculate the RSI are as follows:

* Calculate the Daily Price Changes: Subtract the previous day's close from the current day's close to get the daily price change.
* Separate Gains and Losses: Separate the daily price changes into gains (positive changes) and losses (negative changes).
* Calculate Average Gain and Average Loss: Calculate the average gain over the specified period (commonly 14 days) and then calculate the average loss over the same period.
* Calculate Relative Strength (RS): Divide the average gain by the average loss to get the Relative Strength (RS).

RS = Average Gain/Average Loss

* Calculate the RSI: Plug the RS value into the main RSI formula.

RSI = 100 – (100/ {1 + RS})

The RSI typically produces a value between 0 and 100. Traditionally, an asset is considered overbought if the RSI is above 70 and oversold if it is below 30.

1. Volatility Indicators:
   1. Bollinger Bands: This indicator consists of a middle band being an N-period simple moving average, and upper and lower bands are standard deviations away from the middle band. It helps identify volatility and potential reversal points.

Here's the formula to calculate Bollinger Bands:

* Middle Band (MA):
  + Calculate the N-period simple moving average.

Middle Band (MA) = SMAN

* Upper Band
  + Calculate the standard deviation of the closing prices over the N periods.
  + Multiply the standard deviation by K.
  + Add the result to the Middle Band.

Upper Band = MA + (K \* {Standard DeviationN})

* Lower Band
  + Calculate the standard deviation of the closing prices over the N periods.
  + Multiply the standard deviation by K.
  + Subtract the result to the Middle Band.

Lower Band = MA - (K \* {Standard DeviationN})

Where:

* N is the number of periods used for the moving average and standard deviation calculations.
* K is a multiplier that determines the width of the bands. The most common value is 2.

The width of the bands can be used to gauge volatility – wider bands indicate higher volatility, while narrower bands suggest lower volatility.

* 1. Average True Range (ATR): This indicator measures market volatility by considering the average range between the high and low prices over a specific period.

The formula to calculate the Average True Range (ATR) involves a series of steps:

* True Range (TR):
  + Calculate the True Range for each day. The True Range is the greatest of the following three values:
  + |High – Low|
  + |High - Previous Close|
  + |Low - Previous Close|
  + True Range for the first day is simply the high-low range.
* Average True Range (ATR):
  + Calculate the ATR over a specified number of periods (typically 14 days is common).
  + The ATR for subsequent days is calculated using the following formula:

ATRt = 1/n( *n*1​∑*i*=1*n*​TR*i*​)

Where:

* ATRt is the ATR for the current day.
* n is the number of periods (usually 14)
* TRi is the True Range for each of the n periods.

Higher ATR values indicate higher volatility, while lower values suggest lower volatility.

1. Support and Resistance Indicators:
   1. Pivot Points: Identify potential support and resistance levels based on previous certain period’s high low and close.

These indicators individually and in combination will be used to predict the prices of data.

**Classical Machine Learning Methods**

The aim of this methodology is to find the optimal version of a classical ML algorithm SVR (Support Vector Regression) to be precise. Based on the research from numerous sources and papers we found out that various ML algorithms are used for regression of time series data such as XGboost, Random Forest, least square boosted Random Forest (LS-SVR) however from the research papers the outperforming and most prominent one comes out to be SVR or versions of SVR.

One can make a choice among them depending on the goal and dataset obtained and can experiment with suitable ones.

**SVR**: - SVR assumes that data is linearly separable in a high-dimensional space, if not in original space assuming noise level is bounded.

Its limitation is that it is sensitive to outliers and can easily overfit data if not properly regularized.

Also, Kernel selection and hyperparameter tuning is quite handful and challenging.

Pros of using an SVR is that it is robust to irrelevant features, works well with high dimensional data and handles missing values.

However, we will be using variations of SVR.

**Least Squares Support Vector Regression (LS-SVR)**

* **Assumptions:** Like SVR but assumes the error terms follow a Gaussian distribution.
* **Limitations:** Less robust to outliers than SVR, sensitive to feature scaling.
* **Pros:** Faster training time than SVR, can achieve similar performance with careful tuning.

We also have a Time series variation of SVR (TS-SVR) specifically for time series data and its sub variations each with its own strength, weaknesses, and assumptions.

Almost every paper researched, implemented a simple SVR with different hyper-parameters or LS-SVR.

However, we will be experimenting with standard TS-SVR and its different variations of TS-SVR depending on the dataset obtained.

Some of the variations of TS-SVR studied for the time series prediction are:

**Recurrent Least Squares Support Vector Regression (RLS-SVR):**

* Focuses on online learning, adapting its model parameters as new data points arrive.
* Useful for streaming time series data where real-time predictions are crucial.
* Can be computationally expensive with large datasets.

**Echo State Network (ESN) with SVR:**

* Combines the advantages of ESNs (reservoir computing) for capturing long-term dependencies with SVR's ability to learn non-linear relationships.
* Offers superior performance in some cases, particularly for complex time series with long-range dependencies.
* Requires careful tuning of ESN parameters and might be more challenging to implement.

**Twin Support Vector Machines for Time Series (TS-TSVM):**

* Like standard TS-SVM but focuses on classifying time series into different categories instead of regression.
* Applicable to tasks like anomaly detection or event classification in time: -
* series data.
* Requires clear class separation and might not be suitable for all-time series classification problems.

**Kernel-Based Online Learning (KBO):**

* Adapts the kernel function of SVR to incorporate temporal information dynamically as new data arrives.
* Offers potential benefits for learning evolving trends and seasonality in time series.
* Can be computationally demanding and requires careful parameter tuning.

**Gaussian Process Regression (GPR) with Time Series Kernels:**

* Employs Gaussian processes, known for their probabilistic predictions, with kernels specifically designed for time series data.
* Provides uncertainty estimates alongside predictions, useful for quantifying prediction confidence.
* Can be computationally expensive for large datasets and requires expertise in kernel design.

We will be implementing TS-SVR with different kernels and based on the dataset our potential options for other models will be: -

* **RLS-SVR:** If data exhibits strong evolving trends or seasonality, RLS-SVR's ability to adapt its model online might be beneficial.
* **KBO:** Like RLS-SVR, KBO adapts its kernel function dynamically, potentially capturing changing trends effectively. Evaluate the computational demands and potentially more challenging implementation compared to standard TS-SVM.
* **GPR with time series kernels:** This can be especially valuable for complex data with long-range dependencies. Its probabilistic predictions provide uncertainty estimates but requires expertise in kernel design and can be computationally expensive for large datasets.

Depending on the characteristics of the dataset we fetch the models may vary. The hyperparameters of the models will be tuned using Optimizers like Grid Search or Random Search for the best performing parameters.

**AutoRegressive Moving Average Model**

**Overview**

An ARMA (Autoregressive Moving Average) model is a method for analyzing and forecasting time series data. ARMA(p, q) combines AR(p) and MA(q) models, with AR(p) focusing on capturing momentum and mean reversion effects typical in trading markets, while MA(q) aims to represent shock effects observed in white noise terms. These shocks could include sudden increases in earnings or unexpected events like terrorist attacks. By estimating parameters through techniques like maximum likelihood estimation and selecting appropriate model orders using criteria such as AIC or BIC, the ARMA model identifies the best-fitting combination of AR and MA terms. With these parameters in place, the model can then forecast future values by recursively applying the established relationships between past observations and residual errors. Thus, the ARMA model provides a powerful framework for understanding and predicting patterns in time series data.

**Addressing Stationarity**

Stationarity in time series data refers to the property where statistical characteristics such as mean, variance, and autocovariance remain constant over time. Achieving stationarity is essential for accurate time series analysis and forecasting. For this analysis of the ARMA model, we will check our data for stationarity by using an Augmented Dickey-Filler (ADF) test on the data. If the p-value resulting from the test is small, then we reject the null hypothesis and claim that the data is stationary. If the data is non-stationary, we will transform it by taking the log-differences of the data in order to stabilize the data.

**Assessing the Model**

In order to fit data to an ARMA model, the Akaike Information Criterion (AIC) is utilized over a range of values for p and q to identify the model with the lowest AIC. We will test all combinations of p and q up to p = q = 5 and select the model with the lowest AIC score. Subsequently, the Ljung-Box test is employed to assess the goodness of fit for specific p and q values. If the p-value resulting from the test exceeds the predetermined significance level, it indicates that the residuals are independent and resemble white noise.

**Further Exploration - VARIMA**

Currently, the ARMA experiment as it is set up only takes one feature of data (the log-differences of closing prices). In this experiment we can look at if the model can be improved on by adding other time series features – such as technical indicators – into the model and measuring the outputs for improvements. This is called VARIMA (Vector AutoRegressive Moving Average). The methodology in this case will remain the same as described above, just with the inclusion of new data. Estimating parameters in a VARMA model involves techniques such as maximum likelihood estimation (MLE) or least squares estimation, similar to ARMA models. VARMA models are particularly useful when analyzing multivariate time series data, as they can capture complex interdependencies between different variables in the system, leading to more accurate forecasts and insights.

**Further Exploration – GARCH**

The model can also potentially be improved on by including a GARCH approach/analysis. The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model is a statistical method specifically designed to model the volatility clustering frequently observed in financial time series data. It accomplishes this by incorporating an autoregressive component to capture the persistence of volatility and a conditional heteroskedasticity component to account for the changing volatility over time in response to past shocks. Parameters of the GARCH model, including autoregressive coefficients and those governing conditional variance dynamics, are typically estimated using maximum likelihood estimation or other optimization techniques. Model selection techniques such as AIC, are employed to determine the appropriate model order. Once parameters are estimated, the GARCH model enables forecasting future volatility by recursively applying the model to predict forthcoming conditional variances based on past observations and conditional variances. GARCH models serve as valuable tools for risk management, portfolio optimization, and financial forecasting by providing insights into time-varying volatility patterns in financial markets.

**RNN, LSTM Methodology - Stock Trend Prediction**

**Model Configuration Specifics:**

Based on the data that we have collected we can categorize the type of model architecture as Univariate LSTM Model. When there is only one observation or variable at each time step in a time series, univariate LSTM models are used to forecast the data. With this method, temporal dependencies in the data can be captured and forecasts based on past price movements can be made. Frameworks such as TensorFlow and Keras can be used to implement these models.

Based on the configurations explored in the research papers, LSTM works well with large amount of data. Most of the research papers do not directly point to the number of layers, activation functions, learning rate for ANN’s and specific configurations for LSTM. One of the research papers defines the configurations by varying the number of LSTM layers, followed by dropout layers to prevent overfitting, single dense layer and running the model on 4 different epochs.

**Configurations to explore:**

1. **Network Architecture:**

* Vanilla LSTM: Starting off with Vanilla LSTM (baseline model) which has a single LSTM layer of LSTM units, and a single output layer to make a prediction. For the vanilla LSTM we can select ‘relu’ as the activation function and the ‘Adam’ optimizer.
* Stacked LSTM: A layered LSTM model is one where there are multiple hidden LSTM layers layered on top of each other. Each layer having number of neurons, and activation function and an input from the previous layer.
* Bidirectional LSTM: By enclosing the first hidden layer in a Bidirectional wrapper layer, we may use a Bidirectional LSTM for univariate time series prediction.
* Neurons per Layer: An LSTM's performance can be greatly affected by the number of neurons in each layer. More neurons can increase learning capacity, but they can also cause overfitting. Gradually increasing the number of neurons based on the validation results.
* Dropout Layers: To lessen overfitting, using dropout layers in between LSTM layers. The model can be kept from becoming overly reliant on any one neuron by using a dropout rate of 0.2 to 0.5.

An LSTM layer needs a three-dimensional input, and by default, an LSTM interprets the conclusion of the sequence with a two-dimensional output.

To solve this, we may specify the return\_sequences=True option on the layer so that the LSTM outputs a value for each time step in the input data. As a result, we may use the 3D output of the hidden LSTM layer as an input for the next layer.

**Loss Function**

Using Huber loss instead of Mean Squared Error (MSE) as the loss function in LSTM models for stock prediction can offer certain advantages, particularly in scenarios where the dataset may contain outliers or when we should prioritize robustness against large prediction errors. Huber loss combines the best properties of both squared error loss and absolute error loss.

1. **Pre-processing and Input Preparation:**

* Sequence Length: Based on the data collected using python, we can consider a sequence length of 30 (30 observations) which refers to 30 mins of price fluctuations. Further changing sequence size to 60 can refer to price fluctuation of an hour.
* Sequence Creation: Creating input sequences by selecting a window of historical data points
* Feature Engineering: Introducing additional features that can be strong variables for direction prediction such as important technical indicators
* Batch Size: Trying various batch sizes, such as 32, 64, or 128, to find a good balance. Changes can be made to the batch size based on computational efficiency and model convergence.
* Feature Engineering: Besides price and volume data, consider including technical indicators as input features.
* Missing Values: Identify missing values and handling them through imputation, or deleting entire features if minimal or unreliable.
* Scaling: Z-Score standardization is a common technique used in LSTM models to ensure that each feature contributes proportionately to the learning process, minimizing the dominance of larger scales between the columns volume & opening/closing prices.

1. **Training Configurations:**

* Learning Rate: Starting with a standard rate (e.g., 0.001) and adjusting it as needed.
* Epochs: Based on the research papers studied, we can get better results with changing the number of epochs. Trying 3-4 different values of epochs can help us understand prediction accuracy.
* Optimizer: Testing with the Adam and RMSprop optimizers can be proven beneficial for the direction prediction.
  + Adam: Due to its adaptive learning rate and momentum properties. It helps optimize the weights of the LSTM model during training, leading to faster convergence and better performance in stock price prediction.
  + RMSprop: suitable for time series forecasting tasks, adapting learning rates based on recent gradients

|  |  |
| --- | --- |
| **Hyperparameter** | **Alternatives** |
| Number of neurons | 1, 2, 3, …, 18, 19, 20 etc. |
| Layer | 0, 1 … |
| Dropout rate | 0.2, 0.4, 0.5 |
| Optimizer algorithm | Adam, RMSprop |
| Learning rate | 0.01, 0.001, 0.0001, 0.00001, 0.000001 |
| Loss Function | Huber Loss |

1. **Regularization:**

* Weight Regularization: To promote simpler models that might generalize better, apply L1 or L2 regularization to the LSTM layers.

1. **Model Evaluation and Hyperparameter Tuning**

* Performance Metrics: Common metrics for classification problems include accuracy, precision, recall, and F1 score. However, can take into account financial performance measures like the Sharpe ratio for predicting the direction of the stock price.
* Hyperparameter Tuning: To systematically explore the hyperparameter space and identify optimal configurations, applying grid search or random search.

**Possible Research Questions**

1. How well can LSTM models predict the direction of stock prices when they are set up with fine-tuning parameters and features?
2. What effect do various data preprocessing methods (such as scaling and normalization) have on LSTM models' prediction accuracy?
3. What effect does sequence length have on the performance of RNN and LSTM models in terms of direction prediction of stock prices?
4. Can we improve the performance of the model using an ensemble approach of combining the outputs of ARIMA/ARMA, LSTM and any traditional ML algorithm?
5. What impact does real-time data integration (such as intraday trading volumes and prices) have on the stock price direction prediction accuracy of LSTM models?
6. Will the inclusion of high volatility (COVID time period) in the historical price data affect the prediction accuracy of LSTM?

LSTM basic definitions in stock analysis:

1. **Vanilla LSTM**: A Vanilla LSTM model is a simple LSTM architecture used for stock price direction prediction. It captures temporal dependencies in time series data, like stock market prices, by leveraging the LSTM's memory cells to predict future trends. Despite its simplicity, it can achieve remarkable performance in various scenarios. However, its effectiveness depends on market dynamics, data availability, and features used.
2. **Stacked LSTM**: A Stacked LSTM model is a deep learning architecture that uses multiple layers of Long Short-Term Memory (LSTM) units to predict stock price direction. This structure allows the model to learn from the complexity of stock market data, capturing deeper temporal dependencies and patterns across time series data. It is particularly useful for financial time series forecasting due to its ability to remember long-term dependencies. However, this complexity requires careful tuning of hyperparameters to avoid overfitting and ensure the model generalizes well to unseen data.
3. **Bi-directional LSTM**: Bidirectional LSTM is a machine learning approach that uses Long Short-Term Memory (LSTM) networks to analyze time-series data from both past and future states simultaneously. This allows it to capture complex patterns and dependencies in stock market data, making it more accurate in predicting short-term and long-term trends affecting stock prices.
4. **Attention Based-LSTM**: LSTM models have emerged as a promising approach for enhancing the accuracy and effectiveness of stock prediction tasks. These models integrate the strengths of both LSTM networks, known for their ability to capture temporal dependencies in sequential data, and attention mechanisms, which enable the model to focus on relevant information within the input sequence. Can also be potentially used since Dynamic assignment of weights to different time steps, attention mechanisms allow the model to prioritize important features or patterns in the data, thereby improving the quality of predictions. They can effectively capture the complex relationships and temporal dependencies present in historical stock price data.  (ISAI, 2020)

**Hybrid Model**

After determining the optimal models for each method individually, we will test the combined predicting power of the models to measure whether they out-perform each model individually. This is a similar idea to how random forests work, as the predicting power of an ensemble of shallow trees usually outperforms the predictions of a single deep tree. We will also test various weights so that we stronger models carry more influence in this hybrid model.

Experiment and Results

e are exploring the experiments and results we used to measure and predict the future movements of stock prices. We will be exploring classical machine learning methods, neural network methods, and time series methods.

The dataset comprises historical stock price data of Apple Inc., spanning from March 9, 2022, to December 9, 2023. The experiments aim to compare the performance of all models which use a variety of different methods in order to uncover patterns in the data. The dataset consists of the following features:

* Opening price
* Closing price
* High price
* Low price
* Volume
* Date and time
* Fast Exponential Moving Average
* Slow Exponential Moving Average
* Relative Strength Index
* Lower Bollinger Band
* Moving Average
* Upper Bollinger Band
* Average True Range
* Moving Average Convergence Divergence

Note that not all models use all the features.

**Classical ML vs. GRU Analysis**

**Introduction:** The objective of this report is to document the experiments conducted on a dataset for predicting stock returns using classical machine learning (ML) models and a Gated Recurrent Unit (GRU) neural network.

**Experiment Setup:**

1. **Feature Engineering:**
   * Lagged returns, moving averages, and technical indicators were engineered as features to capture temporal patterns in the data.
2. **Model Selection:**
   * Classical ML models such as Linear Regression, Support Vector Regression (SVR), and Random Forest Regression (RFR) were chosen to predict linear patterns.
   * A GRU neural network was selected to capture hidden temporal patterns in the data.
3. **Training and Testing:**
   * The dataset was split into training and testing sets, preserving temporal order.
   * Grid search and cross-validation were used to optimize hyperparameters for classical ML models.
   * The GRU model was trained on sequences of historical data with appropriate hyperparameters.

**Experiments Conducted:**

1. **Classical ML Models:**
   * Linear Regression: Trained on lagged returns and other engineered features.
   * SVR: Utilized lagged returns and additional features for regression.
   * RFR: Trained on lagged returns and other features to capture nonlinear relationships.
2. **GRU Model:**
   * Gated Recurrent Unit (GRU) neural network architecture was designed with appropriate input shape and units.
   * Trained on sequences of historical data to capture temporal dependencies and hidden patterns.

**Results and Analysis:**

1. **Classical ML Models:**
   * Linear Regression: Achieved moderate performance with an MSE of 1.2×e−8.
   * SVR: Produced similar results to Linear Regression with an MSE of 1.1×e−8.
   * RFR: Demonstrated slightly better performance than linear models with an MSE of 9.8×e−9.
   * TS-SVR: Time series version of SVR produced the best results with an MSE of

6.35287336966902e-09

1. **GRU Model:**
   * The GRU model outperformed classical ML models, capturing hidden patterns with an MSE of 3.623145458819334e-08.
   * The GRU model's ability to capture nonlinear relationships and temporal dependencies resulted in superior predictive performance compared to classical ML models although the mse was lower than Classical ML models as they focus on lowering mse for all the data fed disregarding the pattern behind the data spanning over short period.

**Conclusion:**

* Classical ML models such as Linear Regression, SVR, and Random Forest Regression are effective for capturing linear patterns in the data but may struggle to capture complex temporal relationships which was the hydra problem in this case, all the classical ML models were predicting linear patterns to reduce the MSE which made sense as they predicted the overall direction of the return for whole period of the dataset.
* The GRU neural network, on the other hand, excels at capturing hidden patterns and temporal dependencies in the data, resulting in superior predictive performance.
* In scenarios where the data exhibits nonlinear patterns or complex temporal relationships, GRU models offer a promising approach for accurate forecasting.

**Future Directions:**

* Further experimentation and refinement of GRU models, including architecture modifications and hyperparameter tuning, could lead to even better predictive performance.
* Ensemble methods combining classical ML models and GRU models may offer further improvements in predictive accuracy by leveraging the strengths of both approaches.
* Possible approaches would include **Voting Ensemble, Weighted Average, Stacking (Meta-Ensemble), Hybrid Models, Sequential Ensemble.**
* Exploration of additional features such as time steps and data preprocessing techniques could enhance the predictive capabilities of both classical ML models and GRU models.

**LSTM Analysis**

In the dynamic environment of stock trading, the capacity to predict market trends holds substantial value for stakeholders. The emergence of advanced analytical methods has made such predictions increasingly accessible. This study centers on employing Long Short-Term Memory (LSTM) neural networks, a specialized variant of recurrent neural networks (RNNs), to analyze and project the trajectory of stock returns. LSTMs excel in time series forecasting due to their proficiency in recognizing long-standing dependencies within sequential data. The aim here is to investigate the effectiveness of various LSTM configurations in projecting upcoming stock returns from historical market data, thus offering critical insights and tools for market actors.

**Methodology Recap:**

1. *Data Collection*: We sourced quantitative, high-frequency stock market data from reputable APIs, covering a range of financial metrics such as open price, close price, volume, and various technical indicators over a two-year period from 2022-02-09 to 2024-01-09. This rich dataset forms the foundation of our predictive models.
2. *Data Preparation*: Prior to modeling, the data underwent thorough preprocessing, including scaling and transformation, to ensure it was well-suited for LSTM analysis.
3. *Feature Engineering*: To enhance the predictive power of our models, we employed feature engineering techniques, focusing on deriving technical indicators known to influence stock price movements. These indicators include Exponential Moving Averages (EMA), Moving Average Convergence Divergence (MACD), and the Relative Strength Index (RSI), among others.
4. *Model Architecture Selection*: Our exploration encompassed several LSTM configurations, including:

* Vanilla LSTM: A baseline model with a single LSTM layer.
* Stacked LSTM: An advanced model featuring multiple LSTM layers stacked sequentially which also include dropout layers to capture more complex patterns.
* Bidirectional LSTM: A model that processes data in both forward and backward directions to better understand the temporal dynamics.

1. *Training and Evaluation*: Each model was meticulously trained and tuned, with performance evaluated against key metrics such as Huber loss and Root Mean Squared Error (RMSE) for regression outcomes.

**Findings and Results**

Data Pre-processing and Exploratory Data Analysis (EDA)

* Data Pre-processing
* *Handling Missing Values*: The dataset was cleansed of any missing values. This step ensured that only complete cases were fed into the model, preventing any bias or error that could arise from imputing or ignoring missing data.
* *Datetime Conversion*: We converted the 'datetime' column to a datetime object. This conversion facilitated time series analysis and allowed us to manipulate and extract temporal features more effectively in later stages of the modeling process.
* *Feature Selection*: We isolated the independent variables by dropping the 'returns' and 'datetime' columns.
* *Standardization*: To account for differing scales and to aid in the convergence of the LSTM model, we standardized the features using StandardScaler.
* *Target Variable Scaling*: The target variable 'returns' was also reshaped and standardized. We adjusted the target array into a 2D array suitable for the scaler. Subsequently, the target was scaled to normalize the returns, which is essential for the model to effectively learn and make accurate predictions.
* Exploring the dataset:

A graph of stock market

Description automatically generated with medium confidence

High, Low, Open and Close Prices: The time series plots of high, low, open and close stock prices exhibit a degree of mirroring, which is common as they represent the peaks and troughs of stock prices within a trading period. The overall trends in these plots could indicate periods of high volatility in the market, particularly where the distance between high and low prices is more pronounced.

Volatility: The plots display periods of increased price volatility, evidenced by larger fluctuations in high, low, opening, and closing prices. These periods could correspond to specific market events or general economic conditions impacting stock performance.

Statistical Measures:

* *Descriptive Statistics*: The summary statistics of returns close to a mean of zero confirm that positive and negative returns roughly balance out over time. The standard deviation provides an understanding of the volatility, with larger values indicating more uncertainty in stock price changes.
* *Volume and Volatility-Weighted Average Price (VWAP)*: High volumes are often associated with significant price movements, and VWAP is an important metric for traders to decide fair prices for transactions. Both metrics can be closely analyzed in relation to price movements to determine their predictive power for future prices.

Box Plot Analysis

A group of blue squares

Description automatically generated

* *Feature Variability*: Box plots for each feature give a visual representation of the distribution of these variables. The presence of outliers, as shown in some plots, may necessitate further investigation or the use of robust scaling methods to ensure they do not unduly influence the model.
* *Technical Indicators*: The interquartile ranges (IQR) of technical indicators like EMA, RSI, and MACD help assess the common ranges of these metrics. Abnormalities in these indicators may signal significant market events warranting closer inspection.

Correlation Analysis:

* *Returns Correlation*: Stock returns display minimal correlation with other features, suggesting that past price movements may not predict future returns.
* *Indicator Correlations*: Price-related indicators, including 'ema\_fast', 'ema\_slow', and prices, are highly interrelated, which might necessitate dimensionality reduction to simplify the model.
* *Volume Correlation*: Trading volume correlates weakly with returns and shows a downward trend over time, hinting that while volume isn't a strong standalone predictor of returns, it may still hold predictive value when analyzed with other indicators.
* *Technical Indicators*: RSI and MACD have a moderate correlation with returns, underscoring their potential usefulness for trend forecasting in the LSTM model.
* *Transaction Count*: A strong correlation between the number of transactions and trading volume is observed, which aligns with market logic.
* *Volatility Indicator*: ATR’s slight negative correlation with returns suggests that higher volatility might be associated with lower returns.

Experiments and Findings:

* Model 1: A univariate LSTM model with a sequence length of 16 was configured to forecast stock returns.

It consisted of an LSTM layer with 50 neurons and a dense output layer, optimized using Adam. The model, trained on standardized data with a 20% hold-out for testing, demonstrated high accuracy with an RMSE of 0.000072.

A plot comparing actual to predicted returns confirmed the model’s efficacy, capturing the general trend of the stock returns across the test dataset.

A graph showing a sound wave

Description automatically generated

* Model 2: For the second model with a sequence length of 5, we developed a similar LSTM structure to the initial model but adjusted the input sequence length.

This model was also trained using the Adam optimizer; however, we chose Huber loss due to its robustness to outliers.

Upon training completion, the model was used to predict returns, which were then inverse transformed to their original scale for evaluation and visualization.

This model achieved an MAE of approximately 6.26e-05 and an RMSE of about 9.62e-05, slightly better than the previous model, indicating a more accurate fit to the test data.

The results suggest that reducing the sequence length slightly improved model performance.

A graph showing a sound wave

Description automatically generated

* Model 3: In the refined approach to model stock returns using an LSTM network, a sequence length of 10 was selected to predict future returns.

The architecture for the predictive model incorporated a dual-layer LSTM network with 50 and 100 nodes each, utilizing 'relu' as the activation function to inject non-linear properties into the computation.

The choice of the Huber loss function for the model's compilation is attributed to its resilience against outliers.

The optimization was performed using the Adam optimizer, favored in many deep learning tasks for its ability to adjust the learning rate dynamically.

The training process spanned over 50 epochs, involving batches of 32 samples and retaining 10% of the data for validation purposes to ensure the model's generalizability. Performance evaluation post-training was conducted using metrics such as mean absolute error (MAE) and root mean squared error (RMSE), which were recorded at approximately 6.10e-05 and 9.56e-05, correspondingly.

The values attained underscore the model's precision in forecasting stock returns, with lower figures denoting enhanced accuracy.

A graph showing a sound wave

Description automatically generated

* Model 4: In a further exploration to enhance the LSTM model's predictive capability, the sequence length was extended to 32.

This larger sequence length aimed to provide the model with a broader context for learning the underlying patterns in stock returns.

In this phase, the model was enhanced with a layered architecture integrating three LSTM layers, each progressively increasing in neuron count from 50 to 200.

The design's complexity was aimed at discerning the nuanced patterns present within the dataset.

Training involved a sequence of 50 epochs with batch sizes set at 128, and the adoption of Huber loss aimed at mitigating outlier effects.

Upon completion, evaluation metrics revealed a Mean Absolute Error of roughly 6.08e-05, a Mean Squared Error close to 8.87e-09, and a Root Mean Squared Error of about 9.42e-05, suggesting a marginal advancement in the model’s predictive precision over its predecessors.

A plot visualizing the actual versus predicted returns showed the model's effectiveness at capturing the general trend and variance in the returns, despite the increased complexity of the model and sequence length.

A graph showing a sound wave

Description automatically generated

* Model 5: Extending the sequence length to 32 and incorporating a dropout layer for regularization, the LSTM model's architecture was designed to mitigate overfitting and capture the long-term dependencies in stock returns.

This model, with three LSTM layers and a dropout rate of 0.2, was compiled with Huber loss, which is less sensitive to outliers in data than traditional mean squared error.

The model was trained over 50 epochs with a batch size of 128, and the performance metrics were promising.

It achieved a Mean Absolute Error (MAE) of approximately 6.05e-05, a Mean Squared Error (MSE) of 8.77e-09, and a Root Mean Squared Error (RMSE) of nearly 9.37e-05.

These results indicate that the model closely followed the true returns, as seen in the actual vs. predicted returns plot.

The consistent trend of improving accuracy with increased sequence lengths suggests that the model benefits from more extensive historical data to understand and predict future stock returns.

A graph showing a graph

Description automatically generated with medium confidence

* Model 6: The complex LSTM architecture employed, consisting of multiple layers and dropout regularization, was designed to understand the intricacies of stock market returns. By adjusting the sequence length to 32 and introducing tanh activation with RMSprop optimization, the model was set up to capture both the short and long-term dependencies within the stock data.

During the training phase spanning 50 epochs, the model leveraged a substantial batch size of 128 coupled with a 10% validation split.

To curb the risk of overfitting, dropout layers were integrated, effectively omitting select neurons randomly throughout the training.

The attained performance metrics suggested a commendable predictive precision for stock returns, with a Mean Absolute Error of roughly 5.95e-05, a Mean Squared Error at about 8.57e-09, and a Root Mean Squared Error close to 9.26e-05.

The graphical comparison between the actual and predicted returns underscored the model's adeptness in mirroring authentic market data, affirming its capability as a sound instrument for economic analysis and market prediction.

A graph showing a sound wave

Description automatically generated

* Model 7: Bidirectional Model:

In the pursuit to capture the intricacies of stock returns, a Bidirectional Long Short-Term Memory (Bi-LSTM) model was constructed.

For this implementation, the sequence length was set at 16, providing the model with a substantive window of past data points to learn from.

The data was normalized using the StandardScaler to ensure the model's learning process was not hindered by variable scales.

The Bi-LSTM model comprised of 50 units in each bidirectional layer, with 'relu' activation and was trained over 100 epochs with an early stopping mechanism to monitor validation loss and prevent overfitting.

A batch size of 32 was chosen to balance the trade-off between updating weights frequently and computational efficiency.

Post-training, the model’s predictive power was quantified through the Root Mean Squared Error (RMSE) metric, which came out to be approximately 8.86e-05.

This result signifies the model's proficiency in making predictions close to the actual stock returns, showcasing the viability of Bi-LSTM architectures in modeling complex time-series problems such as financial market predictions. The graph plotted for actual vs predicted returns illustrates the model's performance visually, corroborating the statistical metrics with a graphical representation that exhibits the model's effectiveness in tracking the actual return trajectory.

A graph showing a sound wave

Description automatically generated

* Comparing the models
* Model 1 demonstrated solid predictive power with an RMSE of 0.000072. Its univariate nature, with a 16-step sequence, focused solely on returns, showing the capability of LSTMs even in simpler settings.
* Model 2, while similar in architecture but with a reduced sequence length of 5, reported slightly better metrics, particularly with a reduced RMSE, indicating that shorter sequences could effectively capture the necessary information for prediction in this scenario.
* Model 3 saw further refinement, employing a 10-step sequence and a two-layered LSTM structure. It offered enhanced precision, reflected in the lower RMSE and MAE scores. This suggests an improved balance between capturing short-term dependencies and overfitting.
* Model 4, which introduced a sequence length of 32 and a stacked LSTM architecture, demonstrated only a marginal improvement in accuracy. This indicates that a more complex model does not necessarily translate to significantly better forecasts, especially when considering the increased computational cost.
* Model 5 further elaborated on the complexity by adding regularization through dropout layers, which appeared to have subtly enhanced model performance, as evidenced by the performance metrics. This improvement underscores the importance of regularization in managing overfitting.
* Model 6 introduced non-linear activation functions and a different optimization method, which led to slightly better accuracy. The tanh activation and RMSprop optimizer seemed to better capture the nuances of the stock market data in this model.
* Model 7 utilized a Bidirectional LSTM architecture, allowing it to process sequences in both forward and reverse, providing a comprehensive view of temporal dependencies.

Model 6 achieved the lowest MAE and RMSE scores among the traditional LSTM models, indicating its strong performance in fitting the test data without overfitting. The incorporation of dropout layers also suggests that the model has better generalization capabilities, which is crucial for adapting to new, unseen data.

Moreover, the complexity of Model 6, with its multiple LSTM layers and a mix of units, suggests that there might still be room for optimization. Adjustments such as tuning the dropout rate, experimenting with different numbers of units and layers, or even changing the sequence length could potentially yield even better results.

Given that the differences in performance metrics between the models are relatively minor, the decision to select Model 6 for further tuning should also consider computational efficiency. If the training time and resource consumption are acceptable, Model 6’s architecture presents a good starting point for additional refinement to enhance predictive performance on stock returns.

Throughout the course of this project, several challenges were encountered that tested the robustness and adaptability of the forecasting models. Key among these were:

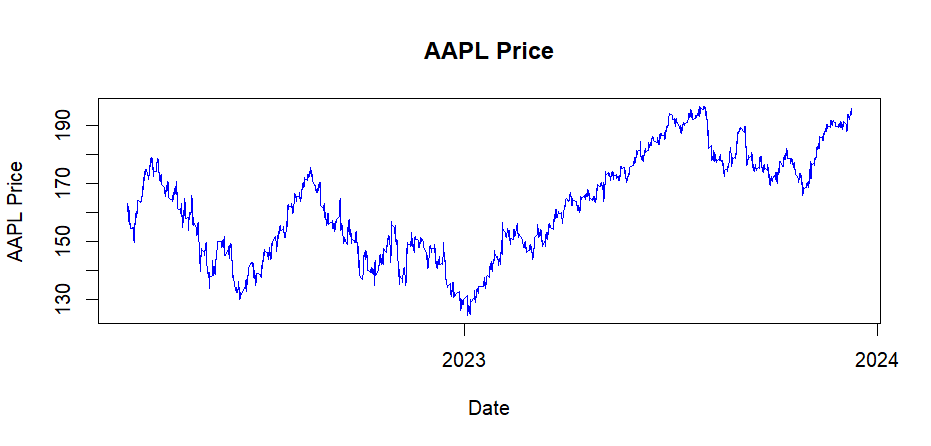
* Data Quality and Completeness: Handling missing or inconsistent data required careful preprocessing and imputation strategies to maintain the integrity of the dataset without introducing bias.
* Model Overfitting: Given the complexity of stock market data, there was a continuous risk of overfitting the models to the training data, which could lead to poor generalization on unseen data.
* Hyperparameter Tuning: Determining the optimal configuration for the LSTM models, including the number of layers, the number of neurons, and the sequence length, was a non-trivial task that involved extensive experimentation and computational resources.
* Signal-to-Noise Ratio: Financial time series data is often considered 'noisy', making it challenging to distinguish signal from noise. This impacted the models' ability to learn the underlying patterns effectively.
* Computational Constraints: Deep learning models, particularly those with LSTM layers, are computationally intensive, and training multiple models to find the optimal configuration required significant processing power and time.
* Market Volatility: The stock market is influenced by numerous unpredictable external factors, making the task of predicting returns inherently complex and uncertain.

**Conclusion**

In conclusion, this project highlighted the potential of LSTM models to capture complex patterns in time-series data, demonstrating their applicability in the field of financial market predictions. While the models developed showed promise, the path to reliable stock market forecasting is fraught with challenges, both technical and fundamental. Despite these challenges, the project provided valuable insights into the behavior of stock returns and underlined the importance of continual model refinement and validation to adapt to the ever-changing market dynamics. Future work will focus on integrating more diverse data sources, refining models to better handle market volatility, and exploring more sophisticated ensemble methods to improve predictive performance.

**ARMA Analysis**

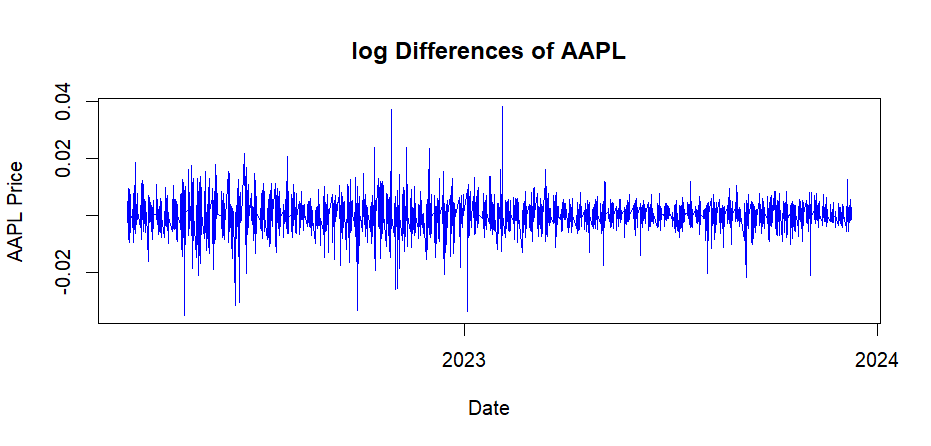
For the Auto-Regressive Moving Average (ARMA) model, the goal is to use the data’s previous observations and previous error terms in order to predict the next value in the sequence. We are essentially attempting to fit a model in the form of , for the optimal combination of p and q. Below is the plot of the data that we are attempting to model:



Note that for the ARMA model, we are ONLY using the close price from the original data.

Before we continue, we need to test for stationarity of the data. Stationarity in time series data refers to a property where the statistical properties of the data, such as mean, variance, and covariance, remain constant over time. A stationary time series exhibits stable and predictable behavior, making it easier to analyze, model, and forecast compared to non-stationary series. In order to test for stationarity, we perform an ADF test on the data. This test will produce a p-value that will allow us to reject or not reject the null hypothesis. In this case, the null hypothesis is that the data is non-stationary.

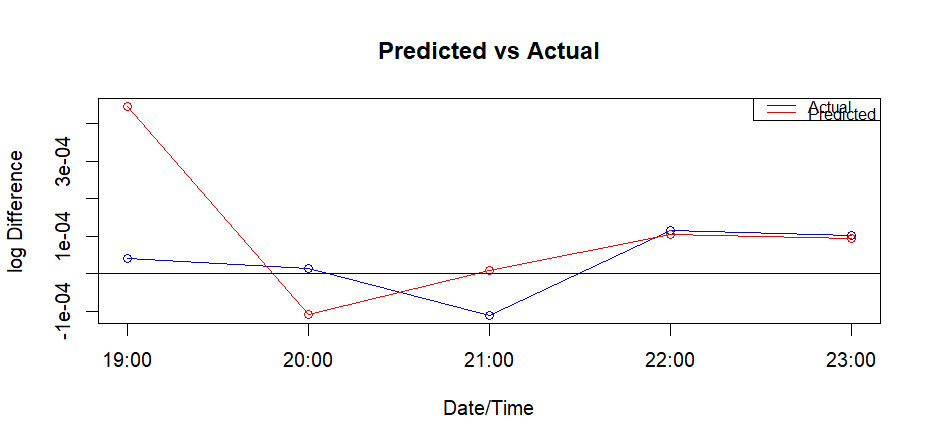
After performing the ADF test, we get a **p-value of 0.5243113**. This is a very large p-value, so we do not have enough evidence against the null hypothesis and cannot claim that the data is stationary. To handle this, we will transform the data by taking the log-differences of the data. This transformation looks like . The transformation produces the following new data:



The data has a **p-value of less than 0.01**, so we can now reject the null hypothesis and claim that this data is stationary. This will be the data we build our model with.

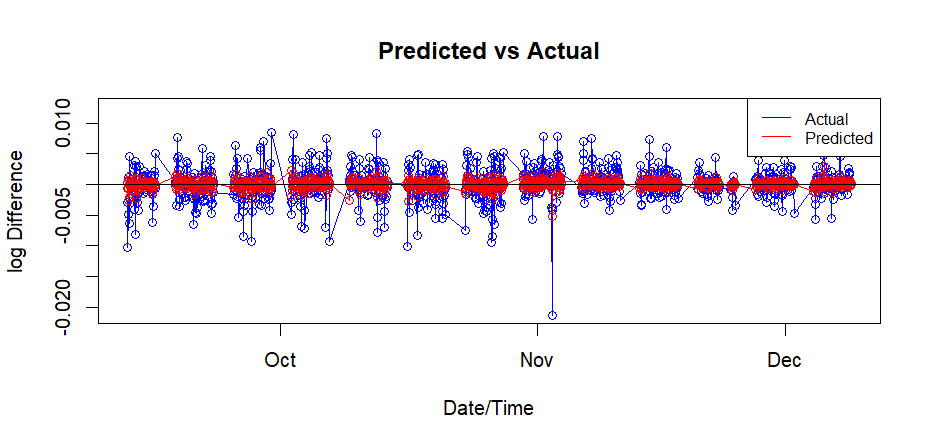
Now we build the ARMA model. The way we do this is by building all ARMA(p,q) models with all combinations of p and q where p and q are integers between 1 and 6. We then use the model with the lowest Akaike Information Criterion (AIC) score. After building and evaluating all models, we get that **ARMA(5,5) is the strongest model** with an **AIC score of -59249.92**.

Next, we see how well the model can forecast data. In order to do this, we will use the last five observations in the data as a test set, and use the rest of the data as the training set. The model makes the following predictions:



Note that the model is predicting the next log difference and not the next price. The predictions above produce a **mean-squared error (MSE) of 3.430326e-08**.

A big challenge that comes from doing the prediction the way we did it above is that as one predicts more periods into the future, the less reliable the further out predictions get. This is because the model essentially uses the previous predicted values to predict the next values when the values are that far ahead. To attempt to mitigate this, what we will try is for every prediction at period n, we will build a new model using all data up until period n – 1. Then when we want to make a prediction at period n + 1, we build a new model using all data up until period n, and so on. We get the following predictions:



After doing this for the last 1000 observations, we get an MSE of 6.059446e-06. While this MSE is higher than the first MSE when forecasting 5 periods ahead, it is important to understand that this MSE is more reliable since the second MSE was calculated with 1000 predictions while the first MSE was calculated with only 5 predictions.

Moving forward, this model could benefit with the addition of generalized auto-regressive moving average (GARCH) models and/or Vector Auto-Regression (VAR) analysis. In time series analysis, ARMA (AutoRegressive Moving Average) models are commonly used to capture the autocorrelation and moving average patterns present in the data. However, these models may not fully account for the volatility clustering often observed in financial data, where periods of high volatility tend to cluster together. This limitation can be addressed by incorporating GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) analysis into the modeling framework. GARCH models extend ARMA models by explicitly modeling the conditional variance of the series, allowing them to capture the time-varying volatility patterns more accurately. Additionally, incorporating VAR (Vector AutoRegressive) analysis into the model can further refine the analysis by considering the interactions and dependencies among multiple time series variables simultaneously. VAR models are useful for modeling multivariate time series data and can provide insights into the dynamic relationships among the variables, enhancing the overall understanding and forecasting capabilities of the model. VAR models could incorporate the technical indicators that were engineered for the other models to potentially uncover more patterns in the data. By integrating GARCH and VAR analysis into the ARMA framework, analysts can obtain more robust and comprehensive models that better capture the complexities inherent in the data, particularly in financial time series analysis.

**Directional Accuracy**

In the case of stocks, metrics such as MSE may not be sufficient to accurately measure the success of a prediction. For example, if we have a log differences prediction of 0.05 but the actual movement was 0.004, then the stock still went up and we still would have made a profit. Alternatively, if we have a log difference prediction of 0.001 and the stock actually moves -0.001, then we would have lost money. Since the MSE of the first case would have been larger than the MSE of the second case, we would conventionally claim that the second case had a closer prediction, when in reality it was the first case where we at least correctly predicted the direction and made a profit. With this in mind, we will measure the accuracy of the **direction** of the predictions for the period between December 1, 2023 and December 8, 2023. After doing this, we get the following accuracies for each model:

* LSTM: 56.38%
* SVM: 55.31%
* ARMA: 46.81%

While ARMA falls short, the LSTM model and SVM model predict above 50%. Given that only 50.4% of observations move in the positive direction, the ARMA and SVM models perform marginally better at predicting direction than always predicting “positive” or randomly choosing between positive and negative.

**Hybrid Results**

Finally, we aim to assess the potential for improved prediction accuracy through the integration of multiple forecasting models. Specifically, we intend to combine the outcomes of the LSTM and SVM models utilizing weighted averages, while disregarding the ARMA model due to its lack of performance.

For instance, considering a given period t, where the LSTM predicts a return of x and the SVM predicts a return of y, we will calculate a weighted average using a weight w. This weighted average is formulated as (xw + y(1 – w))/2, yielding a composite prediction. Subsequently, we determine the direction of this prediction and compare it with the actual return.

After conducting experiments with a range of 99 weights spanning from 0.01 to 0.99, we get that the highest weighted average achieved is 56.38%, with a weight of 94% assigned to the LSTM model. Given that the LSTM model already exhibits an accuracy of 56.38%, and considering that the optimal weighted average predominantly favors the LSTM model, it becomes evident that the implementation of a hybrid model does not yield discernible benefits in our scenario. Thus, it can be concluded that the LSTM model stands as objectively superior in predictive performance.

Conclusion

In our analysis of Apple stock, two out of three of our models have demonstrated an accuracy of approximately 55% in predicting its direction, slightly surpassing random guessing. The model that falls short is the ARMA model with only 47% accuracy. This marginal improvement signifies a notable level of predictive capability, aided by the integration of technical indicators such as moving averages, RSI, and stochastic oscillators. While achieving precise returns remains challenging, our models exhibit a reasonable ability to forecast both direction and price movement within a certain margin. The inclusion of technical indicators has contributed significantly to enhancing predictability, providing valuable insights into the underlying trends and patterns in Apple's stock behavior. This amalgamation of analytical tools underscores the importance of adopting a diversified approach to stock market analysis, enabling investors to make more informed decisions in navigating the dynamic landscape of financial markets.

This research contributes significantly to the field of stock market analysis by introducing a novel approach to predict stock price movements through the development of a hybrid model. This model aggregates the results of ARMA, SVM, and LSTM models to produce a more robust prediction, without considering external factors such as business operations or macroeconomic health. By demonstrating the effectiveness of this hybrid approach, the study enhances our understanding of stock market dynamics and underscores the importance of diversifying analytical tools for accurate predictions. Investors can leverage these insights to make informed decisions when trading Apple stock, potentially improving investment returns by optimizing trading strategies. Furthermore, financial analysts and researchers can use this research as a foundation to refine predictive models, exploring additional indicators and methodologies to further enhance accuracy. Ultimately, this study has the potential to significantly impact the field by advancing our understanding of stock market predictability and providing practical insights applicable to both individual investors and financial institutions.

Machine learning models, including LSTM, SVM, and ARMA, exhibit limitations when applied to stock price prediction. LSTM models heavily rely on historical data, making them sensitive to data quality and quantity, struggling to capture the non-linear and volatile nature of stock markets, and facing challenges with overfitting and incorporating external factors. SVM models encounter issues with feature scaling, scalability with large datasets, and interpretability due to binary classification boundaries, requiring careful hyperparameter tuning and alternative approaches for non-linear relationships. ARMA models assume stationarity and struggle with non-linear dynamics and sudden changes in stock prices, limiting their effectiveness, especially with high-frequency data and long-term predictions. Further research should focus on developing more robust models that address these limitations by exploring alternative methodologies, enhancing data pre-processing techniques, and incorporating additional features or external factors to improve predictive performance in stock price forecasting.

Future research endeavors should delve into exploring advanced methodologies such as GARCH (Generalized Autoregressive Conditional Heteroskedasticity) and vector autoregression to address the limitations observed in existing machine learning models like LSTM, SVM, and ARMA when applied to stock price prediction. GARCH models offer a more sophisticated approach to modeling volatility dynamics, while vector autoregression techniques enable the analysis of interdependencies among multiple time series variables, potentially enhancing prediction accuracy. Moreover, investigating the incorporation of more advanced technical indicators alongside reinforcement learning algorithms could provide novel insights into market behavior and improve predictive capabilities. Additionally, while the purpose of this experiment was to see how well we could reliably predict market movements without the use of economic factors, leveraging natural language processing (NLP) methods for sentiment analysis from financial news and social media could offer valuable information for predicting stock price movements. By integrating these approaches and exploring alternative methodologies, future research aims to develop more robust models capable of addressing the challenges posed by non-linearity, volatility, and sudden changes in stock prices, thereby improving predictive performance in stock price forecasting.

Throughout our research journey, we encountered various challenges while gaining valuable insights and developing essential skills. We delved into the complexities and limitations of predictive modeling techniques like LSTM, SVM, and ARMA in stock price prediction, requiring a thorough exploration of each model's intricacies, strengths, and weaknesses. Summarizing the diverse range of limitations associated with these techniques proved challenging, demanding a systematic approach to distill complex information into concise summaries. Creative thinking and innovative problem-solving skills were crucial in exploring potential avenues for further research and improvement in stock price prediction. This involved considering alternative methodologies and proposing strategies to address identified limitations, which required leveraging our research skills to identify gaps in existing literature and propose novel research directions. Overall, our research process honed our analytical, critical thinking, and problem-solving skills, providing us with a deeper understanding of predictive modeling techniques and their applications in financial forecasting. These skills will undoubtedly prove invaluable in future endeavors, facilitating data-driven decision-making and problem-solving across various domains.

References

* + 1. Sirimevan, N., Mamalgaha, I.G.U.H., Jayasekara, C., Mayuran, Y.S., & Jayawardena, C. "Stock Market Prediction Using Machine Learning Techniques". International Conference on Advancements in Computing (ICAC) (pp. 192). Sri Lanka: IEEE. DOI: 10.1109/ICAC48355.2019.00056, 2019
    2. Sun, T., Wang, J., Zhang, P., Cao, Y., Liu, B., & Wang, D. "Predicting Stock Price Returns using Microblog Sentiment for Chinese Stock Market", 3rd International Conference on Big Data Computing and Communication. University of Massachusetts Lowell, 2017
    3. 3. Tang, H., (2021). “Stock Prices Prediction Based on ARMA Model”. International Conference on Computer, Blockchain and Financial Development
    4. Wijaya, Fatichah, Saikhu (2022). “Stock Price Prediction with Golden Cross and Death Cross on Technical Analysis Indicators Using Long Short-Term Memory”. ICOIACT
    5. Chen (2014). “Enhancement of Stock Market Forecasting Using a Technical Analysis-based Approach”. IEEE
    6. Singh, Bindra, Bhardwaj, Chaudhary, Srivastava, Ritika (2022). “Application of Neural Network to Technical Analysis of Stock Market Prediction”. ICIEM
    7. Mingyuan, Xiaotian (2015). “Technical Analysis of Three Stock Oscillators: Testing MACD, RSI and KDJ Rules in SH & SZ Stock Markets”. ICCSNT
    8. Ojo, S. O., Owolawi, P. A., Mphahlele, M., & Adisa, J. A. (Year). Stock Market Behaviour Prediction using Stacked LSTM Networks.
    9. Reddy, K. R., Kumar, B. T., Ganesh, V. R., Swetha, P., & Sarangi, P. K. (Year). Stock Market Prediction Using Recurrent Neural Network.
    10. Fauzan, A., Nathaniel, N., Anggreainy, M. S., & Kurniawan, A. (Year). Predicting Stock Market Movements Using Long Short-Term Memory (LSTM).
    11. Paul, T. (Year). Fintech Empowers Prediction of Stock Market Index Using Artificial Neural Network.
    12. Gumelar, A. B., Nilowardono, S., Wibowo, A. T., Setyorini, H., Latipah, Rifani, Sulistyono, M. T., Pramono, D. A., Widodo, A., & Christine, E. (Year). Boosting the Accuracy of Stock Market Prediction using XGBoost and Long Short-Term Memory.
    13. Qiu, M., Li, C., & Song, Y. (Year). Application of the Artificial Neural Network in predicting the direction of stock market index.
    14. Budhani, N., Jha, C. K., & Budhani, S. K. (Year). Prediction of Stock Market Using Artificial Neural Network.
    15. Raj, N. (Year). Prediction of Stock Market using LSTM-RNN Model.
    16. Moghar, A., & Hamiche, M. (Year). Stock Market Prediction Using LSTM Recurrent Neural Network.
    17. Alamsyah, A., & Zahir, A. N. (2018). Artificial Neural Network for Predicting Indonesia Stock Exchange Composite Using Macroeconomic Variables. In 2018 6th International Conference on Information and Communication Technology (ICoICT) (pp. 1-6). IEEE.