Project Title: STOCK MARKET PREDICTIONS USING MACHINE LEARNING

Submitted by: Kodchasan Sproll

Student ID: 190005809

Advisor: Hannah Worthington

Table of Content

Abstraction

This dissertation is aimed at readers who have a background in finance, machine learning, or related fields and are interested in exploring the potential of machine learning techniques for predicting stock prices. The study presents a comprehensive analysis of historical stock data from a select group of companies and evaluates the effectiveness of various machine learning models in predicting stock price movements.

The companies chosen for this study are AAPL, AMZN, GOOGL, JNJ, KO, META, MSFT, PG, V, and WMT. The selection criteria included market capitalization, volatility, and diversity across different industries. AAPL, AMZN, and MSFT are among the largest technology companies with significant influence on the overall stock market. JNJ, PG, and WMT are well-established companies in the healthcare, consumer goods, and retail industries respectively, providing a wide range of products and services. KO is a consumer goods company known for its global brand and marketing campaigns, while V is a financial services company that has experienced substantial growth in recent years. Finally, META and GOOGL represent emerging industries in the metaverse and technology sectors, respectively, offering unique opportunities for analysis and prediction.

By evaluating the performance of machine learning models on these diverse stocks, this study provides valuable insights into the applicability and generalizability of the models across different industries and market conditions.

GitHub: https://github.com/overdeadright/stock-prediction-with-ML

Chapter 1

Introduction

I certify that this project report has been written by me, is a record of work carried out by me, and is essentially different from work undertaken for any other purpose or assessment.

1.1 Motivation

The stock market is a complex and dynamic system, with stock prices being influenced by various factors such as economic indicators, company performance, global events, investor sentiment, and numerous independent variables. This results in inherent volatility and unpredictability in stock prices. The stock market plays a crucial role in securing retirement funds and providing inflation protection for a large number of individuals, while also serving as a foundation for day traders and major financial institutions seeking long-term profits. The enigmatic nature of stock prices has attracted the attention of investors, statisticians, and data analysts in their pursuit of predicting the unpredictable.

Accurate stock prediction offers financial advantages to individual investors, large investment institutions, and the companies themselves. The importance of stock predictions has encouraged the development of innovative methodologies and techniques, with machine learning and neural networks emerging as natural progressions. These models often function as contributing factors for traders' decision-making processes rather than standalone tools.

The current project aims to create a simulation of a machine engaging in day trading, a prevalent investment strategy encompassing short-term buying and selling of stocks to capitalize on price fluctuations. Despite the inherent unpredictability of the stock market, the project's objective is to consistently yield a net positive result within the simulation for a broad array of stocks. Contrary to previously mentioned methods, the algorithm will autonomously determine whether to buy or sell stocks instead of merely assisting a trader.

1.2 Goals and Research Questions

To accomplish this objective, the project will investigate various machine learning algorithms and their associated hyperparameters.

The primary goal of this project is to predict stock market trends and successfully generate profits through day trading. In pursuit of this goal, several research questions will be posed and addressed throughout the project:

1. Which data sources and timeframes are most appropriate?

Accurate stock price predictions necessitate high-quality historical data. This project will rely on trustworthy historical data sources and incorporate additional relevant features as needed. Factors to consider include the duration of the simulation and the extent of the data utilized to inform predictions within the simulation.

2. What machine learning techniques should be employed?

Selecting suitable machine learning models is vital to the project's success. This project will explore a range of machine-learning techniques and evaluate their hyperparameters to ensure their effectiveness and appropriateness.

3. Can the project's outcomes be applied to predict future stock prices accurately?

The ultimate goal of this project is to develop a predictive model capable of forecasting stock prices in the real world. The project's success will be gauged by the model's consistent profitability in day trading. It is essential to recognize that effective strategies today may not guarantee success tomorrow, which necessitates the development of a model resistant to overfitting and universally applicable to a diverse array of stocks, ensuring a high probability of positive outcomes.

Chapter 2

Stocks

This section explores all relevant stock concepts.

2.1 Stocks

A stock, also known as a share or equity, represents ownership in a company. When a company goes public, it offers shares of its ownership to the public, enabling individual investors and institutions to become shareholders. As shareholders, individuals earn a portion of the company's profits in the form of dividends. These stocks are transferable and can be bought or sold by other investors [1].

2.2 Stock Market Prediction

Stock market prediction involves estimating a company's future stock value based on historical data and current events. Stocks are traded on stock exchanges daily, with prices continually fluctuating due to market forces. Numerous factors influence stock prices, including the overall economy, news, rumors, company performance, competition, and investor behavior, among others [2]. Accurate stock price forecasting could lead to substantial financial gains for investors, thereby driving research in this domain.

Traditionally, stock prices were predicted through fundamental analysis, which involved examining a company's financial and economic fundamentals such as revenue, earnings, and risk. This process often required calculating various ratios, including debt-to-equity and price-to-earnings ratios, to facilitate informed predictions. With the increased availability of historical data, technical analysis has emerged as another method for examining charts and past data to identify patterns and trends that inform stock price predictions. Neither method is entirely reliable on its own, and investors often rely on multiple indicators to make informed decisions [3].

With the advent of machine learning technology, stock market prediction has entered a new era. Various machine-learning techniques can be applied to historical data, while language-based machine learning can analyze current news articles to gain insights into stock or market opinions. Although these approaches have contributed to stock market prediction, their reliability, particularly when used independently, has not been consistently accurate [4].

2.3 Day Trading

Day trading is an attractive investment strategy for many individuals, particularly those seeking potential profits from short-term price movements in the stock market. This approach entails buying and selling stocks within the same trading day, typically relying on technical analysis and chart patterns for decision-making. However, day trading is a high-risk, high-reward strategy that demands substantial time, skill, and discipline to succeed [5].

Day trading can be challenging and stressful, particularly for novices in the stock market. Success requires a comprehensive understanding of market dynamics and trading psychology, as well as access to real-time market data and specialized trading software. Day traders must also rigorously manage risk, as minor losses can rapidly accumulate and offset potential profits. Nevertheless, those willing to invest the necessary time and effort can find day trading to be a potentially lucrative investment strategy. Through diligent research, market education, and the development of effective trading strategies, day traders can enhance their likelihood of success and achieve their financial objectives.

Chapter 3

Machine learning

This section explores all relevant machine learning concepts.

3.1 Data Preprocessing

Data preprocessing is a crucial step in the machine learning pipeline, as it involves transforming raw data into a suitable format for analysis and modeling. This stage typically involves several tasks, such as data cleaning, handling missing values, normalization, feature scaling, and feature engineering, among others. Proper data preprocessing can significantly improve the performance of machine learning models and enhance the accuracy of predictions.

3.1.1 Data Cleaning

Data cleaning refers to the process of identifying and correcting errors, inconsistencies, and inaccuracies in datasets. This step is essential, as unclean data can introduce noise, bias, and errors into the model, reducing its overall performance. Common data cleaning tasks include removing duplicate records, correcting spelling mistakes, and converting data types to a consistent format [6].

3.1.2 Handling Missing Values

Missing values are a common issue in real-world datasets and can arise due to various reasons, such as data entry errors, equipment malfunctions, or the unavailability of information. Incomplete data can lead to biased or incorrect estimates, and therefore, it is essential to address missing values before model training. There are several strategies for handling missing values, including:

1. Deletion: Removing rows or columns with missing values.
2. Imputation: Estimating missing values using statistical techniques or machine learning algorithms.
3. Interpolation: Estimating missing values based on the values of neighboring data points.

The choice of strategy depends on the nature of the data, the amount of missing data, and the specific requirements of the analysis [7].

3.1.3 Normalization and Feature Scaling

Normalization and feature scaling are techniques used to transform features to a common scale, ensuring that no single feature dominates the learning process. This is particularly important for algorithms that rely on distance metrics or gradient-based optimization, as large variations in the scale of features can result in suboptimal model performance. Common normalization and feature scaling techniques include:

1. Min-Max Scaling: Rescales features to a specified range, typically [0, 1], by subtracting the minimum value and dividing by the range.
2. Z-score Normalization: Centers features around zero with unit variance by subtracting the mean and dividing by the standard deviation.
3. Log Transformation: Applies a logarithmic transformation to reduce the impact of outliers and achieve a more Gaussian-like distribution.

3.1.4 Feature Engineering

Feature engineering involves creating new features or transforming existing features to improve model performance. This process can include techniques such as feature extraction, feature selection, and feature transformation. Proper feature engineering can help reveal hidden patterns within the data, reduce the dimensionality of the dataset, and improve model interpretability [8].

1. Feature Extraction: Deriving new features from existing features, often by combining or aggregating them in a meaningful way.
2. Feature Selection: Identifying the most relevant features for a given problem by evaluating their importance, redundancy, or correlation with the target variable.
3. Feature Transformation: Applying mathematical transformations to features to improve their distribution, reduce skewness, or highlight non-linear relationships.

3.2 Support Vector Machines

Support Vector Machines (SVM) represent a class of supervised machine learning algorithms, primarily employed for classification and regression tasks. The primary objective of SVM is to identify the optimal separating hyperplane, which maximizes the margin between the two classes in a given dataset. In this section, we will discuss the fundamental concepts underpinning the SVM algorithm, including hyperplanes, maximum margin classification, and kernel functions [9].

3.2.1 Hyperplanes and Maximum Margin Classification

A hyperplane is a subspace of one dimension less than the surrounding space. In a two-dimensional space, a hyperplane is a line, while in three-dimensional space, it is a plane. In general, a hyperplane in an n-dimensional space can be defined by the following equation:

$$w\char`^T * x + b = 0,$$

where w is a weight vector, x is an input vector, and b is a bias term. The primary goal of SVM is to find the optimal hyperplane that separates the data points belonging to different classes.

The concept of maximum margin classification is central to SVMs. The margin is defined as the distance between the hyperplane and the closest data points from the two classes. These closest data points are referred to as support vectors, which are crucial in determining the optimal hyperplane. Mathematically, the margin is computed as the distance between the support vectors and the hyperplane, which can be expressed as:

$$margin = 2 / ||w||,$$

where ǁwǁ represents the norm of the weight vector w. The optimal hyperplane maximizes the margin, ensuring the best separation between the classes.

3.2.2 Soft Margin and the C Parameter

In real-world scenarios, datasets are often non-linearly separable, and perfect classification may not be feasible. To address this issue, SVM introduces the concept of soft margin, allowing some misclassifications to achieve a better overall fit. The trade-off between maximizing the margin and minimizing the classification error is regulated by the C parameter, which controls the extent of misclassification allowed. A small value of C permits a larger margin at the expense of more misclassifications, while a larger value of C results in a narrower margin with fewer misclassifications.

3.2.3 Kernel Functions and the Kernel Trick

SVMs can be extended by employing kernel functions to tackle non-linear classification problems. Kernel functions implicitly map the input data to a higher-dimensional feature space, where a linear hyperplane can better separate the classes. The kernel trick allows for the computation of dot products in the feature space without explicitly computing the mapping, leading to computational efficiency [10].

The Radial Basis Function (RBF) kernel is a widely used kernel function in machine learning. It measures the similarity between two data points using a Gaussian function and is defined as$K(x, y) = exp(- \gamma||x - y||\hat{}2)$, where γ is a hyperparameter that controls the width of the Gaussian function. The RBF kernel is capable of modeling complex nonlinear relationships between variables and is often preferred over other kernel functions.

3.2.4 Support Vector Regression (SVR)

Support Vector Machines can also be applied to regression tasks, known as Support Vector Regression (SVR). The objective of SVR is to find a function that approximates the relationship between input features and continuous output values while minimizing the error. Similar to classification, the key concept in SVR is the notion of a margin, which is referred to as an ε-insensitive loss function in the regression context [12].

The ε-insensitive loss function allows for errors up to ε without incurring any penalty, whereas errors exceeding ε are penalized linearly. This leads to the construction of an ε-tube around the regression function, with support vectors being the data points lying outside the ε-tube. The regularization parameter C, along with the kernel function, plays a crucial role in determining the performance of the SVR model.

3.2.5 Model Selection and Parameter Tuning

Model selection and parameter tuning are critical steps in achieving optimal performance with SVM classifiers and regressors. The primary components that require tuning include the choice of the kernel function, kernel parameters, and the regularization parameter C.

Cross-validation is a common technique employed for model selection and parameter tuning, where the dataset is divided into multiple subsets (folds). The model is trained and evaluated iteratively on different combinations of these subsets. Grid search and random search are two popular methods for searching the hyperparameter space, allowing for the identification of the best combination of parameters that yields the highest performance.

3.2.6 Linear Discriminant Analysis for Dimensionality Reduction

In high-dimensional datasets, the performance of SVMs can be affected by the curse of dimensionality, which refers to the exponential increase in computational complexity and potential overfitting as the number of features grows. One approach to mitigate this issue is to employ dimensionality reduction techniques prior to applying the SVM algorithm. Linear Discriminant Analysis (LDA) is a popular technique for this purpose, particularly in the context of classification tasks [13].

LDA aims to find a linear transformation of the original feature space that maximizes the separation between classes while minimizing the within-class scatter. This is achieved by projecting the data onto a lower-dimensional subspace, which retains most of the discriminatory information present in the original space. In practice, LDA can help improve the computational efficiency and performance of SVM classifiers by reducing the complexity of the feature space while preserving class separability.

3.2.7 Grid Search for Hyperparameter Optimization

As discussed earlier, model selection and parameter tuning are critical steps in achieving optimal performance with SVM classifiers and regressors. Grid search is a widely-used method for searching the hyperparameter space, allowing for the identification of the best combination of parameters that yields the highest performance [14].

Grid search involves specifying a discrete set of values for each hyperparameter and systematically evaluating all possible combinations of these values through cross-validation. For instance, in the case of an SVM classifier with an RBF kernel, one would perform a grid search over a range of values for both the regularization parameter C and the kernel parameter γ. The combination of C and γ values that results in the best cross-validation performance is selected as the optimal set of hyperparameters for the SVM model.

Although grid search can be computationally expensive, especially for large datasets and high-dimensional feature spaces, it ensures an exhaustive search of the hyperparameter space, potentially leading to improved model performance. In practice, grid search can be combined with dimensionality reduction techniques such as LDA to reduce the computational burden and improve the efficiency of the hyperparameter optimization process.

3.3 Autoregressive Integrated Moving Average (ARIMA) Models

Autoregressive Integrated Moving Average (ARIMA) models are a class of linear time series models widely used for forecasting univariate time series data. The primary objective of ARIMA models is to capture the underlying patterns and structures in the time series data, such as trends, seasonality, and autocorrelation, to make accurate predictions of future observations. In this section, we will discuss the fundamental concepts underpinning the ARIMA model, including its components, model identification, and parameter estimation.

3.3.1 Components of the ARIMA Model

An ARIMA model is defined by three key components: autoregression (AR), differencing (I), and moving average (MA). The ARIMA model is typically denoted as ARIMA(p, d, q), where p represents the order of the autoregressive component, d represents the degree of differencing, and q represents the order of the moving average component. [15]

1. Autoregression (AR): The autoregressive component captures the dependency of the current observation on its previous observations. An AR(p) model can be expressed as:

$$Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots + \varphi_p Y_{t-p} + \varepsilon_t$$

   where Y_t denotes the observation at time t, c is a constant, φ_i represents the autoregressive coefficients, and ε_t is the error term at time t.
2. Differencing (I): The differencing component aims to achieve stationarity in the time series by subtracting the previous observation from the current observation. A time series is considered stationary if its statistical properties, such as mean and variance, remain constant over time. The differenced series is represented as:

$$\Delta Y_t = Y_t - Y_{t-1}$$

   where ΔY_t denotes the difference between observations at time t and t-1. The degree of differencing d indicates the number of times the differencing operation is applied to the original time series.
3. Moving Average (MA): The moving average component captures the dependency of the current error term on its previous error terms. An MA(q) model can be expressed as:

$$Y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

   where θ_i represents the moving average coefficients.

3.3.2 Model Identification and Parameter Estimation

The process of selecting the appropriate ARIMA model order (p, d, q) is called model identification. Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots are commonly used for this purpose. The ACF plot displays the autocorrelation coefficients for different lags, while the PACF plot displays the partial autocorrelation coefficients, which capture the direct relationship between observations separated by a specific lag.

Once the model order is identified, the ARIMA model parameters can be estimated using maximum likelihood estimation (MLE) or other estimation methods, such as the method of moments or least squares. The model's goodness-of-fit can be assessed using criteria such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC).

3.3.3 AutoARIMA for Automatic Model Selection

Instead of manually identifying the appropriate ARIMA model order (p, d, q) using ACF and PACF plots, a more efficient approach is to use an automatic model selection algorithm, such as AutoARIMA. [16] AutoARIMA is a stepwise procedure that iteratively searches for the best-fitting ARIMA model by evaluating different combinations of (p, d, q) and selecting the one that minimizes the model selection criterion, such as the AIC or BIC.

AutoARIMA begins by testing different values of d to identify the optimal degree of differencing that achieves stationarity in the time series. Subsequently, the algorithm explores various combinations of p and q, fitting ARIMA models and comparing their AIC or BIC scores. The model with the lowest AIC or BIC score is chosen as the optimal ARIMA model.

3.4 Multi-Armed Bandit Problem

The multi-armed bandit problem is a classical problem in probability theory and statistics that models the exploration-exploitation dilemma faced in decision-making processes. In this problem, an agent is presented with multiple slot machines, or "bandits," each with an unknown probability distribution of rewards. The agent's objective is to maximize its total reward over a finite number of attempts, or "pulls," by strategically choosing which bandit to play at each step.

The multi-armed bandit problem exemplifies the trade-off between exploration and exploitation. Exploration involves gathering information about the bandits by sampling each one to estimate their reward probabilities, while exploitation focuses on playing the bandit with the highest estimated reward probability to maximize the total reward. Balancing exploration and exploitation is essential, as overly focusing on either can result in suboptimal rewards.

Several algorithms have been proposed to solve the multi-armed bandit problem, with Upper Confidence Bound (UCB) and Thompson Sampling being two of the most widely used approaches. [17]

3.4.1 Upper Confidence Bound (UCB)

The Upper Confidence Bound (UCB) algorithm is a popular solution to the multi-armed bandit problem that balances exploration and exploitation by incorporating an optimism in the face of uncertainty principle. This algorithm selects the bandit with the highest upper confidence bound on its estimated reward probability.

The UCB algorithm assigns a confidence interval to each bandit's estimated reward probability, with the interval width typically decreasing as more samples are collected. The algorithm selects the bandit with the highest upper bound at each step, thus encouraging exploration of bandits with high uncertainty while also exploiting bandits with high reward probabilities.

The UCB1 algorithm, a specific variant of UCB, is defined as follows:

1. Initialize the play count and the total reward for each bandit.
2. For each round, select the bandit with the highest UCB value, calculated as:
   UCB(i) = mean_reward(i) + sqrt(2 * log(total_plays) / plays(i))
   where mean_reward(i) is the average reward of bandit i, total_plays is the total number of plays so far, and plays(i) is the number of times bandit i has been played.
3. Update the play count and total reward for the selected bandit.

3.4.2 Thompson Sampling

Thompson Sampling is a Bayesian approach to the multi-armed bandit problem that balances exploration and exploitation by sampling from the posterior distribution of each bandit's reward probability. This approach introduces randomness in the selection process, which encourages exploration while also exploiting the bandits with higher reward probabilities.

The Thompson Sampling algorithm can be described as follows:

1. Initialize a prior distribution for each bandit's reward probability, typically using a Normal distribution with parameters (1, 1).
2. For each round, sample a reward probability from the posterior distribution of each bandit and select the bandit with the highest sampled value.
3. Observe the reward from the selected bandit and update the posterior distribution accordingly.

Chapter 4

Methodology

This section explores the Methodology and the simulation.

4.1 Simulation Design

The simulation aims to optimize the prediction of stock price movements to facilitate more informed trading decisions and assess the effectiveness of the model during the COVID-19 pandemic. Using reinforcement learning techniques, specifically Thompson Sampling or Upper Confidence Bound (UCB) in combination with Support Vector Machines (SVM) or ARIMA models.

4.2 Data Collection

Historical stock data for a selected set of stocks is obtained from Yahoo Finance. This data includes daily stock prices and relevant financial metrics, which serve as the basis for training and testing the machine learning models. Yahoo's historical data comprises the following information:

- Date
- Open
- High
- Low
- Close
- Adjusted Close (will be removed)
- Volume

Additionally, several technical indicators are calculated and incorporated as features in the dataset:

- Moving Average Convergence Divergence (MACD)
- Signal Line
- Relative Strength Index (RSI)
- Average True Range (ATR)
- On-Balance Volume (OBV)
- Chaikin Money Flow (CMF)

These indicators provide a comprehensive view of the stock's historical performance and can help improve the predictive capabilities of the models.

4.3 Model Optimization and Training Size Selection

The hyperparameters of the SVM or ARIMA models are optimized using grid-search or autoarima techniques to achieve the best performance in predicting stock price movements. However, the training size cannot be optimized in the same manner. Therefore, a range of training sizes from 21 to 120 days (a total of 100 sizes) is considered.

For each trading day, the model is trained using data from 21, 22, ..., or 120 days before that day. The model then predicts whether the stock price will go up or down using the data of that day. This results in 100 decisions, one for each training size.

4.4 Reinforcement Learning with Thompson Sampling or UCB

To determine which decision to trust, Thompson Sampling or Upper Confidence Bound (UCB) algorithms are employed. These reinforcement learning techniques enable the model to learn from its past decisions and progressively trust the decisions associated with the most successful training sizes. Over time, the algorithm should gravitate towards the training size that consistently produces the best predictions.

4.5 Trading Strategy and Performance Evaluation

In this simulation, a trading strategy is implemented in which one stock for a company is either bought or shorted each day based on the model's predictions. Buying a stock indicates the expectation that the stock price will increase, while shorting a stock signifies the anticipation of a price decrease. The profit or loss for each transaction is calculated as the difference between tomorrow's closing price and today's closing price.

The performance of this trading strategy is tracked over a period of three years, from 2020 to 2023. This timeframe is particularly interesting as it encompasses the COVID-19 pandemic, which significantly impacted the global economy and financial markets. Assessing the model's performance during this period allows for the evaluation of its adaptability to unpredictable market conditions and its ability to maintain profitability amidst heightened market volatility.

Due to time constraints, 10 stocks were tested for SVM but only 5 were tested for ARIMA.

Chapter 5

Results

This section explores all results from 4 simulations.

5.1 SVM and UCB

Stock	Net gain
AAPL	6.2622380999999700
AMZN	10.764042900000000
GOOGL	3.3376110000000200
JNJ	6.6259923000000000
KO	-0.5940042999999950
META	-15.174981099999900
MSFT	28.878961899999900
PG	7.365010400000020
V	3.1539928000000100
WMT	-2.8719870999999800

Table 1: Result of SVM and UCB simulation

UCB (Upper Confidence Bound) is a deterministic algorithm, meaning that when provided with the same dataset and settings, it will arrive at the same result.

Table 1 shows the results of a 3-year stock prediction simulation are presented for 10 different stocks. The majority of the stocks (8 out of 10) show positive net gains, indicating that the model has generally performed well in predicting the direction of these stocks. Microsoft (MSFT) emerged as the top performer with a net gain of 28.88, followed by Amazon (AMZN) and Procter & Gamble (PG).

However, there are two stocks with negative net gains - Coca-Cola (KO) and Walmart (WMT), suggesting that the model's predictions were less accurate for these companies. Furthermore, the model significantly underperformed for Meta Platforms (META), which experienced a net loss of 15.17.

5.2 ARIMA and UCB

Stock	Net gain
AAPL	-0.6912609000000200
AMZN	-13.916441700000000
GOOGL	3.0493899999999900
JNJ	-9.746016099999960
KO	-0.28400130000000300

Table 2: Result of ARIMA and UCB simulation

Table 2 shows the results of a 3-year stock prediction simulation for a smaller sample of 5 different stocks. The majority of the stocks in this dataset show negative net gains. Only Alphabet Inc. (GOOGL) had a positive net gain of 3.05, while the other four stocks experienced net losses.

The model's performance is generally weak as it has failed to predict the positive performance of most stocks. Amazon (AMZN) and Johnson & Johnson (JNJ) are the worst-performing stocks with significant net losses.

5.3 SVM and Thompson Sampling

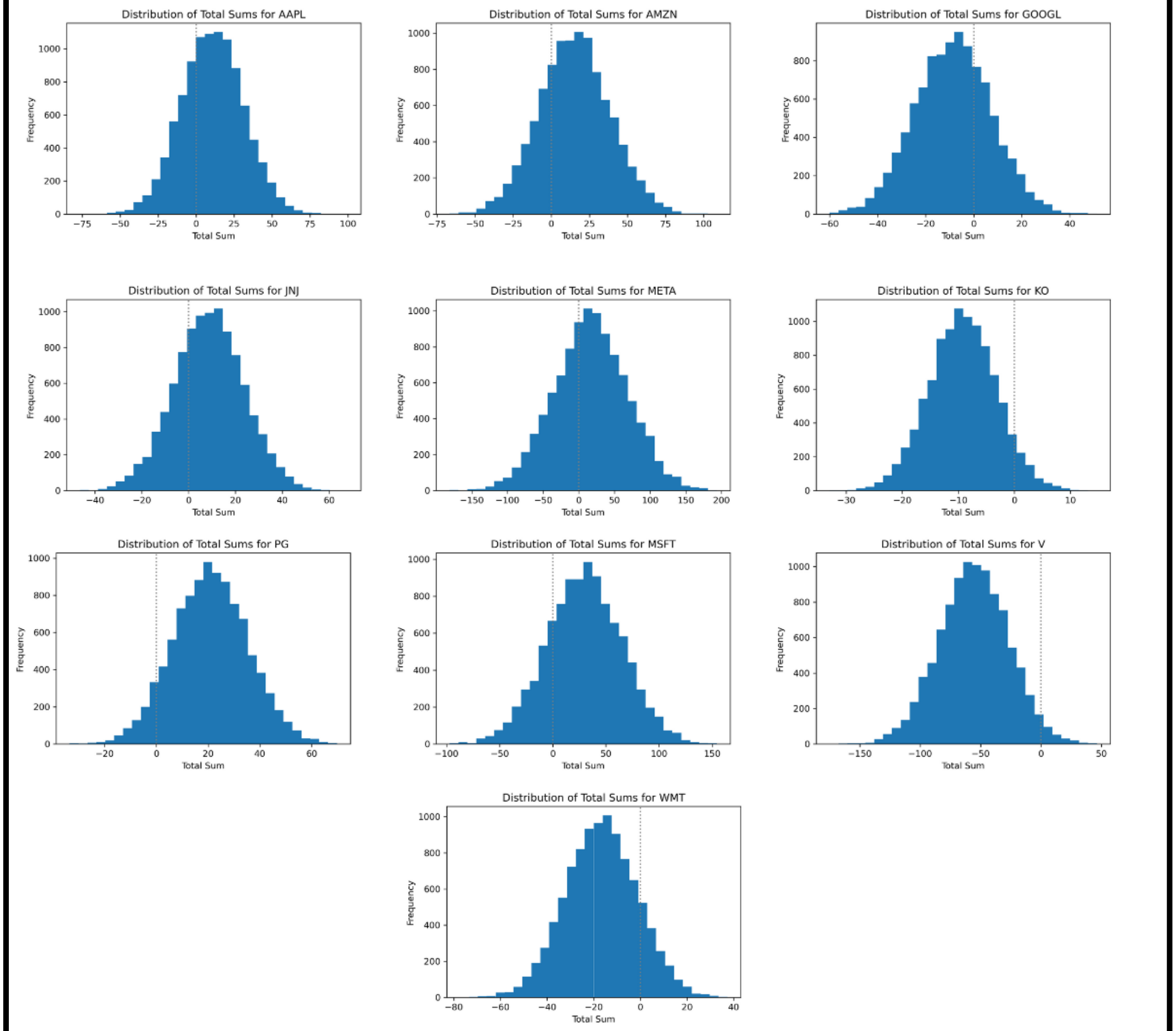

Image 1: Histogram SVM and Thompson Sampling simulation

Stock	**Mean Net Gain**	**Median**	**Standard deviation**	**Minimum value**	**Maximum value**	**25th percentile**	**75th percentile**	**Odds**
AAPL	10.72	10.74	20.39	-76.40	99.65	-3.05	24.59	0.6984
AMZN	15.22	15.18	23.41	-66.77	108.74	-0.50	30.68	0.7431
GOOGL	-8.61	-8.41	15.88	-60.03	51.25	-19.38	2.21	0.2958
JNJ	8.60	8.70	14.92	-46.37	67.85	-1.41	18.65	0.7187
KO	-9.22	-9.22	6.00	-32.96	14.71	-13.15	-5.20	0.0614
META	16.53	16.74	51.65	-181.07	193.60	-17.36	50.97	0.6315
MSFT	29.52	29.89	35.57	-97.17	154.40	5.53	53.63	0.796
PG	20.97	20.92	14.44	-33.55	69.80	11.09	30.68	0.9278
V	-55.56	-55.47	28.62	-174.99	46.25	-74.40	-35.95	0.026
WMT	-16.70	-16.66	15.39	-77.31	37.30	-27.10	-6.34	0.1416

Table 3: Result of SVM and Thompson Sampling simulation

Unlike the UCB method, Thompson Sampling utilizes a statistical approach. In this case, 10,000 trials were conducted, producing a variety of outcomes. This is then turned into a histogram and the table provided displays these results, where the "odds" column indicates the likelihood of achieving a profit, meaning a net gain greater than 0. A 0.6984 for Apple means there was a 69.84% chance of making a net gain, and 30.26% of losing money.

Image 1 shows the Histogram results of all 10,000 trials. Being above 0 is the goal, a graph that is skewed right will have higher odds of achieving profit. From Table 3, Apple (AAPL), Amazon (AMZN), Johnson & Johnson (JNJ), Meta Platforms (META), Microsoft (MSFT), and Procter & Gamble (PG) have positive mean net gains, indicating that they generally performed well during the simulation. Microsoft (MSFT) has the highest mean net gain, followed by Procter & Gamble (PG) and Amazon (AMZN).

On the other hand, Alphabet Inc. (GOOGL), Coca-Cola (KO), Visa (V), and Walmart (WMT) show negative mean net gains, implying that their performance was less favorable during the simulation. Visa (V) has the lowest mean net gain, followed by Walmart (WMT) and Alphabet Inc. (GOOGL).

5.4 ARIMA and Thompson Sampling

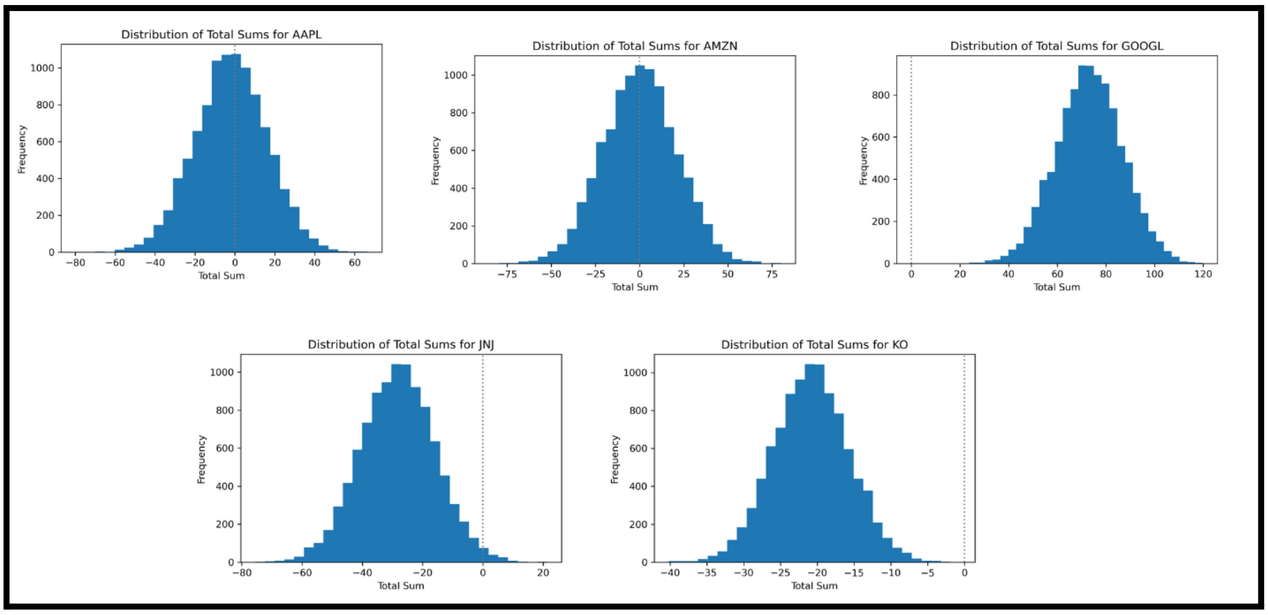

Image 2: Histogram ARIMA and Thompson Sampling simulation

Stock	Mean Net Gain	Median	Standard deviation	Minimum value	Maximu m value	25th percentile	75th percentile	Odds
AAPL	-2.03	-1.87	18.00	-79.73	66.38	-14.05	10.04	0.4584
AMZN	0.07	0.23	20.89	-85.14	80.02	-13.89	14.00	0.5052
GOOGL	73.14	73.17	13.63	23.81	119.85	64.03	82.28	1.0
JNJ	-28.23	-28.18	12.41	-75.48	21.26	-36.56	-19.92	0.0107
KO	-20.82	-20.82	5.14	-40.16	-0.55	-24.27	-17.39	0.0

Table 4: Result of ARIMA and Thompson Sampling simulation

Image 2 shows the Histograms results of all 10,000 trials. From Table 4, Alphabet Inc. (GOOGL) has the highest mean net gain of 73.14, and also the highest odds of 1.0, indicating that all its outcomes were positive. Apple (AAPL) and Amazon (AMZN) have slightly negative and positive mean net gains of -2.03 and 0.07, respectively. Johnson & Johnson (JNJ) and Coca-Cola (KO) show negative mean net gains, with values of -28.23 and -20.82, respectively.

The stock with the highest standard deviation is Amazon (AMZN) with a value of 20.89, followed closely by Apple (AAPL) with a value of 18.00, indicating that these stocks have experienced the most significant fluctuations in performance during the simulation.

The stocks with the lowest odds are Coca-Cola (KO) with a value of 0, and Johnson & Johnson (JNJ) with a value of 0.0107. The model's overall performance is terrible, except for a notable outlier observation for GOOGL which exhibits exceptional results with a 100 percent probability of generating profit. A little comment: I checked everything and cannot find an error for why this outlier occurred in a model I consider awful.

Chapter 6

Conclusion

This section explores the conclusions and critiques of the simulations.

6.1 Summary of Findings

In evaluating the overall performance of the prediction model, a positive net outcome is deemed successful. The Support Vector Machine (SVM) demonstrated better performance compared to the AutoRegressive Integrated Moving Average (ARIMA) model, primarily due to SVM's utilization of multiple parameters, whereas ARIMA relies solely on closing price data. It's worth noting that if someone had held onto stocks from 2020 to 2023, they would have made a bigger profit. So, in retrospect, it seems I didn't really do better than the overall market for these top companies.

6.2 Evaluation of Methodology and Limitations

The proposed approach offers several advantages, including adaptability to evolving market conditions, the employment of historical data from reliable sources such as Yahoo Finance, and the exploration of diverse training sizes. Nevertheless, the approach is not without its limitations, which have been considered throughout the course of this research:

1. Overfitting: The model may be susceptible to overfitting, leading to inadequate generalization to unseen data. This issue can be exacerbated when employing complex models or extensive training sizes that may capture data noise rather than underlying trends.
2. Computational Complexity: The utilization of grid-search or auto.arima for hyperparameter optimization, coupled with the evaluation of multiple training sizes, can be computationally intensive. This complexity may pose challenges in terms of processing time and computational resources, particularly when handling large or high-frequency datasets.
3. Market Noise and Unpredictability: Stock markets are influenced by myriad factors, including economic indicators, corporate performance, global events, investor sentiment, and various independent variables. As a result, stock prices exhibit volatility and unpredictability, complicating the development of accurate prediction models.
4. Variability in Training Size Performance: A particular training size may initially perform well but experience diminishing performance over time, or vice versa. This variability can introduce complications into the reinforcement learning aspect of the

approach, as it relies on learned weights from prior successes to make informed decisions.

6.3 Recommendations for Improvement and Future Research

To address these limitations and enhance the performance of the proposed approach, the following potential improvements are suggested:

1. Implement Weight Memory Duration: Integrate the concept of weight memory duration by allowing the weights assigned to different training sizes to decay over time. This method ensures that the reinforcement learning algorithm does not overemphasize past performance and can adapt more dynamically to changes in training size effectiveness.
2. Utilize Ensemble Methods: Apply ensemble methods to combine the strengths of multiple models, potentially mitigating issues related to overfitting, market noise, and unpredictability.
3. Explore Alternative Reinforcement Learning Techniques: Investigate the use of alternative reinforcement learning techniques that may offer improved adaptability and responsiveness to changes in training size performance.
4. Investigate Advanced Machine Learning Techniques: Conduct further research to explore the application of deep learning and other advanced machine learning techniques to address the challenges associated with the stock market prediction.

By implementing these improvements, the model could potentially achieve enhanced prediction accuracy and adapt more effectively to the dynamic and unpredictable nature of stock market data. I am keen on continuing to build on this model. The GitHub code is public but I gave up maintaining the code to a level of readability. The link is https://github.com/overdeadright/stock-prediction-with-ML

Bibliography

[1] Investopedia. (n.d.). Stock Basics. Retrieved from https://www.investopedia.com/terms/s/stock.asp

[2] Corporate Finance Institute. (n.d.). Stock Market Prediction. Retrieved from https://corporatefinanceinstitute.com/resources/capital-markets/prediction-market/

[3] Investopedia. (n.d.). Fundamental vs. Technical Analysis: What's the Difference?. Retrieved from https://www.investopedia.com/ask/answers/difference-between-fundamental-and-technical-analysis/

[4] Towards Data Science. (n.d.). Machine Learning in Stock Market Prediction. Retrieved from https://towardsdatascience.com/machine-learning-for-stock-prediction-a-quantitative-approach-4ca98c0bfb8c

[5] NerdWallet. (n.d.). Day Trading: What It Is and How It Works. Retrieved from https://www.nerdwallet.com/article/investing/day-trading-risks

[6] IBM. (n.d.). Data Cleaning. Retrieved from https://www.ibm.com/cloud/learn/data-cleaning

[7] Towards Data Science. (n.d.). Handling Missing Values in Data. Retrieved from https://towardsdatascience.com/handling-missing-values-in-data-9f652f32226

[8] Towards Data Science. (n.d.). When and Why Should You Normalize/Standardize/Rescale Your Data?. Retrieved from https://towardsdatascience.com/when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff

[9] KDnuggets. (n.d.). Feature Engineering in Machine Learning. Retrieved from https://www.kdnuggets.com/2018/12/feature-engineering-machine-learning.html

[10] SVM-Tutorial. (n.d.). Svm Understands Math Part 1. Retrieved from https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/

[11] Engati (n.d.). Kernel Method. Retrieved from https://www.engati.com/glossary/kernel-method

[12] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification. Retrieved from https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[13] Machinelearningmastery (n.d.). Cross-Validation, Grid Search and Random Search for Model Selection and Hyperparameter Tuning. Retrieved from https://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/

[14] Towardsdatascience (n.d.). Linear Discriminant Analysis for Dimensionality Reduction. Retrieved from https://towardsdatascience.com/linear-discriminant-analysis-in-python-76b8b17817c2

[15] Machinelearningmastery (n.d.). ARIMA for Time Series Forecasting with Python. Retrieved from https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

[16] RDocumentation (n.d.). auto.arima - Forecasting Using ARIMA. Retrieved from https://www.rdocumentation.org/packages/forecast/versions/8.21/topics/auto.arima

[17] Towardsdatascience (n.d.). A Comparison of Bandit Algorithms. Retrieved from https://towardsdatascience.com/a-comparison-of-bandit-algorithms-24b4adfcabb