MACHINE LEARNING APPLICATIONS IN PORTFOLIO OPTIMIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FİRDEVS NUR UYKUN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
FINANCIAL MATHEMATICS

SEPTEMBER 2024

Approval of the thesis:

MACHINE LEARNING APPLICATIONS IN PORTFOLIO OPTIMIZATION

submitted by **FİRDEVS NUR UYKUN** in partial fulfillment of the requirements for the degree of **Master of Science in Financial Mathematics Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Kestel
Dean, Graduate School of **Applied Mathematics** ________________

Prof. Dr. A. Sevtap Kestel
Head of Department, **Financial Mathematics** ________________

Assist. Prof. Dr. Büşra Zeynep Temoçin
Supervisor, **Actuarial Sciences, METU** ________________

Examining Committee Members:

Prof. Dr. Ali Devin Sezer
Financial Mathematics, METU ________________

Assist.Prof.Dr. Büşra Zeynep Temoçin
Actuarial Sciences, METU ________________

Prof. Dr. Furkan Başer
Actuarial Sciences, Ankara Uni. ________________

Date: ________________

Name, Last Name:	FİRDEVS NUR UYKUN
Signature :	

ABSTRACT

MACHINE LEARNING APPLICATIONS IN PORTFOLIO OPTIMIZATION

UYKUN, FİRDEVS NUR
M.S., Department of Financial Mathematics
Supervisor : Assist. Prof. Dr. Büşra Zeynep Temoçin

September 2024, 121 pages

This study examines the evaluation of S&P 500 trend movements and their impact on portfolio optimization methodologies. Through the meticulous construction of risk aversion-adjusted portfolios applicable to both single and multiple period analyses, the research employs variance and Conditional Value at Risk (CVaR) for single periods, while using Mean Absolute Deviation (MAD) for multiple periods. The optimization process benefits from the synergistic use of Python for computational modeling and AMPL for executing complex mathematical formulations. To accurately predict risk aversion, the study utilizes six classification models integrated with 29 indicators derived from technical analysis: Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Decision Trees (DT), Random Forest (RF), and eXtreme Gradient Boosting (XGBoost). Simultaneously, return forecasting leverages the predictive capabilities of four regression frameworks—Linear Regression, LSTM, XGBoost, and LightGBM—based on various technical indicators. Explainable AI (XAI) techniques, particularly LIME and SHAP, facilitate a deeper understanding of feature importance in the decision-making processes of machine learning algorithms. The findings of this thesis demonstrate that the methods used perform differently across various optimization problems. While DT achieves the highest Sharpe ratio for Mean-Variance portfolios, LR performs best in the Mean-CVaR portfolios, and SVC excels for the Mean-MAD portfolios. Additionally, the KNN, SVC, and LR have the lowest Sharpe ratios, respectively. In the process of predicting returns, Linear Regression produces the best outcomes based on the applied comparison metric. Lastly, the XAI methods highlight the importance of incorporating the Average Directional Index (ADX) with a ten-period setting in the feature design of the KNN and LR models.

Keywords: Portfolio Optimization, Technical Analysis, Machine Learning, Explainable Artificial Intelligence

ÖZ

PORTFÖY OPTİMİZASYONUNDA MAKİNE ÖĞRENMESİ UYGULAMALARI

UYKUN, FİRDEVS NUR
Yüksek Lisans, Finansal Matematik Bölümü
Tez Yöneticisi : Dr. Öğr. Üyesi Büşra Zeynep Temoçin

Eylül 2024, 121 sayfa

Bu çalışma, S&P 500 eğilimlerinin değerlendirilmesi ve bu eğilimlerin portföy optimizasyonunda kullanılan metodolojiler üzerindeki etkilerini incelemektedir. Hem tekil hem de çoklu dönem analizlerine uygulanabilir riskten kaçınma ayarlı portföylerin titizlikle oluşturulması yoluyla, araştırma, tekil dönemlerin değerlendirilmesi için varyans ve Koşullu Riske Maruz Değer (CVaR) kullanırken, çoklu dönemler için Ortalama Mutlak Sapma (MAD) kullanmaktadır. Optimizasyon süreci, hesaplamalı modelleme için Python'un ve karmaşık matematiksel formülasyonların gerçekleştirilmesi için AMPL dilinin sinerjik kullanımından faydalanmaktadır. Riskten kaçınma davranışlarını doğru bir şekilde tahmin etmek amacıyla, çalışma, teknik analizden türetilen 29 gösterge ile entegre edilen altı sınıflandırma modelinden yararlanmaktadır: Lojistik Regresyon (LR), K-En Yakın Komşu (KNN), Destek Vektör Sınıflandırıcı (SVC), Karar Ağaçları (DT), Rastgele Orman (RF) ve Aşırı Gradient Artırma (XGBoost). Eşzamanlı olarak, getiri tahmini, çeşitli teknik göstergelerin analiziyle temellendirilmiş dört regresyon çerçevesinin: Doğrusal Regresyon, LSTM, XGBoost ve LightGBM öngörücü yeteneklerinden yararlanmaktadır. XAI teknikleri, özellikle LIME ve SHAP, makine öğrenimi algoritmalarının karar verme süreçlerindeki özellik önemini daha derinlemesine anlamayı kolaylaştırmaktadır. Bu tezin bulguları, kullanılan yöntemlerin çeşitli optimizasyon problemlerinde farklı performans gösterdiğini ortaya koymaktadır. DT, Ortalama-Varyans portföylerinde en yüksek Sharpe oranına ulaşırken, LR, Ortalama-CVaR portföylerinde en iyi performansı sergiler ve

SVC, Ortalama-MAD portföylerinde üstünlük sağlamaktadır. Ek olarak, KNN, SVC ve LR sırasıyla en düşük Sharpe oranlarına sahiptir. Getirileri tahmin etme sürecinde, uygulanan karşılaştırma metriğine göre Doğrusal Regresyon en iyi sonuçları üretmektedir. Son olarak, XAI yöntemleri, KNN ve LR modellerinin özellik tasarımında on-dönemlik Ortalama Yön Endeksi'nin (ADX) dahil edilmesinin önemini vurgulamaktadır.

Anahtar Kelimeler: Portföy Optimizasyonu, Teknik Analiz, Makine Öğrenmesi, Açıklanabilir Yapay Zeka

To my Mom & Dad

ACKNOWLEDGMENTS

I would like to express my profound gratitude towards my thesis advisor, Assistant Professor Dr. Büşra Zeynep Temoçin, for her meticulous guidance, boundless enthusiasm, and insightful recommendations throughout the development and preparation stages of this thesis. Her readiness to impart her knowledge and experiences has significantly contributed to my growth and illuminated my academic journey.

I would like to express my gratitude to the members of the examining committee, Prof. Dr. Ali Devin Sezer and Prof. Dr. Furkan Başer, for their suggestions and insightful comments.

Throughout my life, the unwavering support influence of my parents, Şengül Uykun and Ali Uykun, have constituted the cornerstone of my development and success. Their unconditional love, belief in my potential, and guidance have been instrumental in shaping the individual I have become. I am eternally thankful for their endless support and contributions to my personal and academic growth.

I also hold a special place of gratitude in my heart for my dear relatives, including Mustafa Uykun, Serpil Okumuş and Ekrem Doğanay, whose precious memories remind me of the importance of love and support. To Aysun Pancar, Zehra Demiray, and Nurhan Terkan—your encouragement and presence in my life are treasures that I will always cherish.

During the challenging periods of my journey, my friends İrem Akgönül, Eda Nur Ünal, Ece Akansel, Rabia Çakır, Nesli Burkankulu, Betül Polat, Yeşim Girgin, Nesrin Ergin, Ezgi Pehlivanlı, Ezgi Canıgüzel, Fatma Zehra Erdoğan, and Harun Doğan stood by me like pillars of strength. Their support, love, and belief in me were vital in keeping me motivated and focused. They were the torchbearers of hope and camaraderie, making the arduous journey a bit easier to tread. To them, I owe a debt of gratitude for their sincere companionship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

x	Vector
X	Matrix
t	Generic Time
T	Time Horizon
R	Return of Portfolio
σ^2	Variance of Portfolio
ω_i	Relative weight of asset i
ω	Vector of Portfolio weights
ω_*	Optimal vector of all portfolio weights
Σ	Covariance Matrix
$\mathbf{1}$	Vector of ones
MPT	Modern Portfolio Theory
MVO	Mean Variance Optimization
EMH	Efficient Market Hypothesis
KFCV	K-Fold Cross Validation
OHLCV	Open High Low Close Volume
MA	Moving Average
SMA	Simple Moving Average
WMA	Weighted Moving Average
EMA	Exponential Moving Average
ADX	Average Directional Movement Index
RSI	Relative Strength Index
CCI	Commodity Channel Index
EMV	Ease of Movement
BB	Bollinger Bands
SAR	Parabolic Stop-and-Reverse
MFI	Money Flow Index
WVAP	Volume Weighted Average Price

S&P 500	S&P 500 Index
LR	Logistic Regression
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
SVR	Support Vector Regression
SVC	Support Vector Classification
DT	Decision Trees
RF	Random Forest
XGBoost	eXtreme Gradient Boosting
ANN	Artificial Neutral Network
RNNs	Recurrent Neural Networks
LSTM	Long Short Term Memory
LightGBM	Light Gradient Boosting Machine
XAI	eXplainable artificial intelligence
LIME	Local Interpretable Model-agnostic Explanations
SHAP	SHapley Additive exPlanations
RMSE	Root Mean Square Error

CHAPTER 1

INTRODUCTION

1.1 Problem description

Time series is an ordered temporal sequence from past to present. Studies on time series analysis cover a wide range of areas, from statistics to medicine. Practitioners collect data in business, economics, engineering, and other scientific fields which include both observable and unobservable components. Observable variables consist of the past of current data, and unobservable variables make time series challenging to evaluate the data. Time series modeling issue is especially crucial for forecasting financial time series data. However, there is no perfect solution to address the forecasting challenges, particularly when considering time series data characterized by varying intervals.

The common use of machine learning for financial market prediction has diverted researchers' attention away from conventional time series techniques. Different machine learning models used for predicting financial time series have gain importance. For example, tree-based models have been used for trend and price prediction. Also, deep learning which is a subfield of machine learning models, has recently gained popularity and produced excellent results in multiple domains thanks to the opportunity given by the big data era and computational capacity. Recurrent Neural Networks and Convolutional Neural Networks, examples of deep learning models, have also been used in forecasting literature. Long Short Term Memory (LSTM), a recurrent neural network model, has been used to predict the price in time sequential form.

In addition to forecasting studies, machine learning can also be used to facilitate portfolio construction. Portfolio management and finding optimal investment weight of each asset are crucial for investors. Two sources utilized for portfolio analysis are past returns performance and securities forecasts for the future [62]. Efficient portfolios have been achieved by solving three types of problem which are minimizing risk for given expected return, maximizing return for given risk and maximizing risk-adjusted mean return. Risk-adjusted mean return equation uses risk aversion, which is investors' response to market fluctuations. Risk aversion is influenced by market's bullish and bearish signals.

The literature reviewed for the study showed that there were two problems regarding stock index forecasting: the first is predicting the precise price, and the second is predicting the direction movement of the stock index. This study aims to forecast the direction of the stock index's movement in order to calculate the risk aversion coefficient, solve various optimization questions associated with risk-adjusted mean return, and predict stock returns to improve portfolio evaluation.

1.2 Proposed Method & Contributions

Our research workflow is organized into four distinct phases. The initial phase centers on predicting risk aversion coefficient using historical S&P 500 data. This entails generating technical analysis variables to compile an extensive set of features. Next, we utilize six different classifiers to forecast the S&P 500 trend, evaluating their performance via the accuracy metric. The study investigates how stock index movement predictions can enhance portfolio construction solutions, explicitly defining the probability of predicting an upward movement in the S&P 500 as a risk aversion.

The study scrutinizes three pivotal elements in the formation of investment portfolios: asset allocation, diversification, and portfolio rebalancing. Regarding asset allocation, the methodology encompasses the selection of 24 stocks and a risk-free asset. To attain diversification, stocks from various sectors is meticulously chosen, ensuring a broad representation of industry types. For portfolio rebalancing, a rolling window technique is employed, offering a systematic approach to adjust portfolio weights over time.

In the second phase, the research integrates predictions of the risk aversion coefficient, which is found in the initial phase, into the mean-risk models, facilitating the computation of optimal portfolio weights within both single and multiple-period scenarios. This approach allows for a nuanced understanding of how risk preferences impact portfolio construction and adjustment strategies, thereby contributing to the literature on efficient portfolio management under uncertainty.

In the third phase of the workflow, the research concentrates on the analysis of single-period portfolios by forecasting asset prices over a one-week horizon. This involves the application of four distinct regression models: Linear Regression, Long Short-Term Memory (LSTM), XGBoost, and Light Gradient Boosting Machine (LightGBM). The primary objective within this phase is to derive predictions on returns by leveraging these modeling techniques.

In the workflow's final stage, we apply the most accurate trend classifiers, specifically Logistic Regression and K-Nearest Neighbors. The justification and implications of these models are further explored through Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive explanations (SHAP), enhancing our understanding of the model's predictions, as detailed in Section 2. This methodical approach not only enables us to predict market trends with notable accuracy but also to delve into the factors underlying these predictions, significantly enriching the field of financial forecasting and portfolio management.

The outcomes of this workflow demonstrate that the effectiveness of different algorithms in portfolio evaluation, as measured by the Sharpe ratio, results in varying performance. Notably, the DT algorithm exhibits superior performance in terms of the Sharpe ratio within the context of Mean-Variance portfolios, while the KNN algorithm demonstrates the lowest Sharpe ratio. Furthermore, with respect to the Mean-CVaR portfolios, the LR algorithm achieves the highest Sharpe ratio, whereas the

SVC manifests the lowest ratio value. In the analysis of Mean-MAD portfolios, the SVC algorithm stands out by achieving the highest Sharpe ratio, inversely, the LR algorithm is marked by the lowest Sharpe ratio.

During the empirical analysis, which evaluates the efficacy of various predictive models in computing financial returns, Linear Regression is observed to yield superior results. This conclusion is drawn based on the performance of these models against the established comparison metric. Following Linear Regression, LightGBM demonstrates strong results but does not surpass the performance of Linear Regression. LSTM, while typically effective for sequential data, also performs worse than Linear Regression. XGBoost, despite its advanced capabilities, delivers the worst performance among the models tested. These findings highlight that simpler models like Linear Regression can sometimes outperform more complex ones, emphasizing the importance of model selection and feature engineering in financial forecasting.

Additionally, the application of XAI methods underscores the significance of incorporating the Average Directional Index (ADX), with a designated period of ten, in the feature design of both KNN and LR models, highlighting the critical role of this indicator in enhancing model performance and interpretability.

1.3 Literature Review

Three related topics will be covered in the literature review. The first part of the review will discuss research on portfolio formation problems. Research studies on stock market trend predicting technical analysis indicators and stock price predictions will be the main topic of the second part. The third and last portion will cover studies on explainable artificial intelligence.

1.3.1 Porfolio Formation

The well-known economist Harry Markowitz was awarded a Nobel Prize in 1990 for his contributions to financial economics through his article Portfolio Selection (1952).

He founded Modern Portfolio Theory (MPT) in 1952, a mathematical modelling approach based on statistics and theoretical presumptions. He defined future return of an asset as random variable and defined portfolio risk measure as variance. He used geometric analysis and the return-risk principle to demonstrate the nature of efficient portfolio design. His work pointed out the importance of diversification in portfolios by selecting assets in different industries [61]. Although Markowitz‘s mean variance model is used by industry and academia, it has some limitations, some practitioners want to calculate only down deviation and Markowitz also suggest semi-variance in his book [62]. Using the mean return variance model's assumptions, such as the normal distribution of returns and the quadratic utility function, becomes increasingly difficult in finance.

Different risk measures which construct different mean-risk portfolios were also investigated in the literature. Mean Absolute Deviation (MAD), a piecewise linear risk measure, was proposed by Konno and Yamazaki [53] in 1991 as a potential substitute for Markowitz's model's drawbacks, including its inability to solve large asset portfolios. In 1999, Rockafellar and Uryasev introduce Conditional Value at Risk (CVaR) as a new risk measure for constructing efficient portfolios and shared how optimization of CVaR with finite scenarios [82].

Some studies focused only on specific risk measures, while others compared multiple risk measures. A study conducted by Byrne et al. [16] analyzed five essential risk measures, such as semi variance, minimax, and MAD rebalance portfolios, on a quarterly basis. The objective was to determine minimum risk optimization portfolios and evaluate the asset allocation of risk measures. The results demonstrated that while MAD and semi variance's asset allocation aligns with mean variance, minimax deviates from this approach. In their study, Hundjra et al.[44] analyzed four different risk measures - mean variance, semi variance, MAD, and CVaR - in the context of minimum-risk portfolio optimization models in South Asian stock markets from 2003 to 2015. Their findings suggest that, particularly during various economic scenarios, CVaR with a 0.95 confidence level is a dependable risk measure for optimizing single period portfolios. As such, it is recommended that investors in the region consider incorporating CVaR into their portfolio optimization models during these times for improved results.

Research on financial investments has changed with advancements in modelling techniques and financial instruments. Artificial intelligence applications evolved the portfolio formation with many ways, machine learning and deep learning models are used for evaluation of models and new optimization techniques are used such as neural network based approach is used in Ban et al.'s [8] study for mean-variance and mean-CVaR models.

There has been a lack of investigation into the relationship between the trading signals of the market and the adjustment of risk aversion in portfolio optimization despite the various types of artificial intelligence applications that have changed portfolio formation. A study was carried out by Ji et al. [46] in 2019 which delved into multi-period risk aversion adjusted mean-Gini portfolios. The study utilized four distinct machine learning models, namely Logistic Regression, Neural Networks, Random Forest, and Support Vector Machine. Over a 17-year period, 942 US companies within and outside the S&P 500 index were analyzed. The researchers found that logistic regression with regularization achieved the greatest accuracy throughout the entire period. They also considered four shorter periods to account for trend movement. To compare market movement predictions, they used a known risk-aversion set and evaluated the short periods based on various metrics such as out of sample weekly average return and cumulative return. With the exception of one period, the use of dynamic changes in risk aversion led to better results compared to the use of constant risk aversion. In an extensive study, Ji et al. [47] explored the integration of risk aversion with machine learning predictions for the S&P 500 market. Their research employed Logistic Regression and XGBoost models, with the latter proving to have superior accuracy. The study analyzed a portfolio of 25 companies traded in the S&P 500 market, constructing various benchmark portfolios and comparing out-of-sample successes. While machine learning models produced comparable results, XGBoost stood out with a higher return.

In 2021, Dubach [30] utilized CVXPY to model optimization problems and Python modeling languages to compare five portfolio optimization problems: minimum variance, maximum return, maximum Sharpe ratio, risk aversion, and minimum CVaR. He then identified the challenge of implementing risk aversion. He compared the portfolios using only the Swiss Exchange index and the 85 listed companies on the Swiss Equity Market. His thesis compares specific problems with and without a short selling constraint for a daily in-sample period spanning the entire eigth year, with 2018 and 2019 utilized for out-of-sample periods.

Return prediction is an another improvement for machine learning and deep learning models in portfolio researches. Return predictions are made using two methods: before optimization for modelling assessment and after optimization for comparison of out of samples. For the first time, Ma et al. [59] in 2020 looked into return prediction for the preselection of stocks with a mean-variance model, using the classical time series price prediction model, which is autoregressive integrated moving average as a comparison benchmark, along with machine learning models consisting of Random Forest and Support Vector Regression and deep learning models such as Long Short Term Memory. They evaluated the performance of different models using 49 stocks from the Chinese Securities 100 Index daily returns and 60-day historical returns as features with sliding window approach. The mean-variance model with random forest outperformed other models in terms of excess returns.

1.3.2 Forecasting in Stock Market Index

The Random Walk Theory is an extension of the Efficient Market Hypothesis (EMH), which asserts that the market is capricious, and prices are arbitrary [33]. The Efficient Market Hypothesis (EMH) posits that stock prices reflect all available information. It comes in three variants: weak, semi-strong, and strong, which respectively encompass past trading data, all public information, and all information including insider knowledge [34]. Anomalies and the high instability of stocks refute the efficacy of the semi-strong form [26].

Researchers use three basic assumptions when employing technical analysis methodologies for forecasting: market activity discounts everything; prices move in trends; and history repeats itself [70]. The stochastic nature of the EMH precluded technical analysts from accepting it.

Investors who seek stock index investment typically adhere to either a buy-and-hold or random purchase strategy, in accordance with the principles of the weak form of the EMH. On the other hand, technical analysts often question the applicability of the weak form and may tend to support the validity of the semi-strong form efficiency in financial markets [55]. They leverage technical analysis as a strategic tool to attain superior returns. To stay ahead in the competitive market, market participants must keep themselves informed about the stock market index trend.

In Sezer et al.'s [89] review, trend prediction is studied with three ways, the first method of trend prediction uses only historical pricing, the second method adds technical and fundamental analysis, and the third method uses text mining algorithms. The literature is replete with studies that endeavor to devise more effective trading strategies using technical analysis, machine learning and evaluate them against the buy-and-hold approach. In their study, Chen et al. [22] employed a combination of feature-weighted KNN and SVM models to accurately predict and classify trends in the Shanghai Stock Exchange Composite Index and Shenzhen Stock Exchange Component Index. By analyzing nine technical analysis variables and various time frames, their findings offer valuable insights into the application of technical analysis in stock market analysis. Using machine learning, Ayala et al. [7] successfully predicted stock market indexes for Germany, Spain, and the US. After employing four different models, they discovered that the Linear Regression and Artificial Neural Network models proved to be the most effective. Their study centered on generating clear and concise buy or sell orders based on estimated price movements that were easily comprehensible for investors.

Despite the abundance of financial data available, there is a lack of research on the application of technical analysis in the management of investment portfolios. A study by Zhu et al. [101] found that no previous studies had realistically used technical analysis and optimization models with data; they used a moving average signal in the asset allocation problem of two assets. The study compared an all-or-nothing approach, where the moving average suggests a buy signal and the investor buys an asset with all their available funds. The study considered different utility functions, such as log-utility and power-utility, to theoretically research the moving average effect and its lag adjustment. Through a simulation of 78 years of monthly observations on the S&P 500, the study provided compelling evidence that utilizing the moving average with optimization was a lucrative approach for investing.

1.3.3 Forecasting in Stocks

After forecasting the stock market index, improving forecasting of the stock prices is another vital field for portfolio optimization. May [63] reports that researchers are actively exploring ways to replace past returns with anticipated returns and leverage machine learning to formulate investment portfolios based on these projections. This field is a heavily studied aspect of machine learning, with a focus on prediction and portfolio creation. May conducted research about XGBoost price prediction with using nine technical analysis variables and select optimal portfolio using Monte Carlo simulations. May used two test periods and XGBoost performed better in predicting the top 40 stocks traded on the Johannesburg Stock Exchange in 2019 than in 2020, according to average RMSE. May compared an equal-weighted portfolio, a recommended portfolio, and the Johannesburg Stock Exchange index. She suggested that the recommended portfolio had a higher return for both test periods.

In their study, Hartanto et al. [40] compared different boosting models for the purpose of predicting stock price. The models they assessed were XGBoost, AdaBoost, CatBoost and Light Gradient Boosting Machine (LightGBM). The study examined various time frames ranging from 1992 to 2022, all for the purpose of predicting the stock price of Apple. According to their findings, LightGBM was the best-performing model among the four, with the lowest RMSE value.

Kobets and Savchenko [52] investigated Markowitz portfolio models profit prediction with using eleven S&P 500 companies and Vanguard 500 Index Fund and they did close price predictions with Linear regression and LSTM models. They suggest

LSTM since it gives better portfolio result. They compared three portfolios: one using only historical data and two combining historical data with monthly predictions. They found that using machine learning predictions in portfolio optimization is helpful for advising.

Additionally, there exist studies that uses machine learning into the stock price prediction. Monikasri and Varshini[68] predicted the future of stocks with using open, high, low and close prices from 2003 to 2021 using LSTM and they suggest LSTM is the best for stock price prediction. Nikou et al. [71] compared four machine learning models, including ANN, SVR, RF, and LSTM, to predict the price of an exchange traded fund. According to their findings, LSTM achieved the best result in terms of RMSE.

Dhokane et al. [29] conducted an empirical investigation into the predictive capabilities of the LSTM model on stock prices, utilizing a dataset encompassing eight distinct stocks. The study was distinct in its approach by integrating two technical analysis variables, namely the Exponential Moving Average (EMA) with varying periods and the Moving Average Convergence Divergence (MACD), into the predictive model. Their findings substantiate the hypothesis that incorporating these technical analysis variables significantly enhances the accuracy of stock price predictions. This is a noteworthy contribution, as the prevailing literature predominantly emphasizes the use of open, high, low, and close values as primary features for prediction. By demonstrating the augmented predictive power through the inclusion of EMA and MACD, Dhokane et al. provide a compelling argument for the reevaluation of feature selection in stock price prediction models. This research underscores the potential benefits of integrating technical analysis variables alongside traditional features to improve the precision of predictive models in financial markets.

In their study, Tasneem et al. [93] embarked on a comprehensive analysis of the stock prices for Acer Ltd. and Asustek Comp. Inc. This involved the application of various technical analysis variables, specifically the Relative Strength Index (RSI), Simple Moving Average (SMA), Exponential Moving Average (EMA), and Moving Average Convergence Divergence (MACD). The objective was to assess the predictive accuracy of these techniques in forecasting stock price movements. To achieve a thorough evaluation, the study compared several advanced machine learning and deep learning models including XGBoost, Random Forest, LightGBM, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and a hybrid approach combining LightGBM with deep learning strategies.

1.3.4 Explainability in time series

There are two widely used ways for predicting financial time series: one uses technical analysis, which provides high interpretability, and the other includes machine learning models, which provide high accuracy [3]. Model accuracy is crucial for scientists when analyzing financial time series. Hybridization of classifiers is used to improve prediction accuracy, although this method may make computation more complex. Another option is to build a single model with trustworthy attributes, increasing the model's confidence [19]. Building trust is crucial for the development of models using real data. There is a need for an explanation of the model's behavior to aid decision-making because merely assessing accuracy may not help us comprehend the model. Regulators should get explanations before making vital choices. Consequently, the research field of XAI (eXplainable Artificial Intelligence) approaches has emerged. They aid in the constructing trust in machine learning classifiers and give insight to researchers about trust in prediction and model issues [81].

Researchers are interested in the relationship between accuracy and explanation, and according to the Google Trends Popularity Index, the term "Explainable Artificial Intelligence" peaked in popularity in July 2023. Although XAI gained popularity after the introduction of various approaches, like LIME (Local Interpretable Model-agnostic Explanations) in 2016 and SHAP (SHapley Additive exPlanations) in 2017, XAI is not a novel idea; it has a history spanning back several decades to when artificial intelligence systems were developed [24].

Explainability and interpretability, in accordance with the literature review, are comparable terms. There is a need for clear distinction for them; Rudin [86]said that after the modelling process, the model explainability could be satisfied, and interpretability could be produced for the intrinsic behavior of the model. This study follows the Rudin viewpoint for the purpose of simplicity.

The literature on financial time series with XAI discusses two standard techniques, LIME and SHAP. The integration of LIME with Random Forest and eXtreme Gradient Boosting (XGBoost) models for trend prediction of six daily stock indices including S&P 500 was first presented in the literature by Çelik et al [19]. As evaluation metrics, accuracy, precision, recall, F1-score, and ROC-AUC are used, RF outperforms XGBoost. For illustration, seven random samples produced by the LIME algorithm are provided. In their study, Bandi et al. [9] employed technical analysis and sentiment analysis in conjunction with LIME to gain a deeper understanding of sentiment classification regarding the Indian stock market index. Carta et al. [18] employed RF and LIME in the feature selection process to increase the accuracy of the intra-day return estimates for 300 stocks from several countries. They found that LIME performed poorly and required it to be modified to work with financial time series data compared to other strategies for increasing model accuracy.

Deng et al. [27] compared XGBoost, SVM, RF, KNN, ANN and Ordinary Least Squares models for direction forecasting of the Shanghai and Shenzhen composite indexes. Index daily movements are categorized as increasing and decreasing, respectively, 1 and 0. Models are compared using performance criteria like accuracy and F1-score, and according to the average testing period, XGBoost is selected to be the best model. Three sentiment indexes are employed as features, and the SHAP technique is used to estimate their importance. Participants in the market are informed of SHAP results to gain an improved knowledge of directions. Mandeep et al. [3] forecast three S&P 500 listed companies' closing values, namely Adobe, Amazon, and Apple, using the Random Forest and XGBoost models. Instead of using metrics for model comparison, they focused on LIME and SHAP explanations. Goodell et al. [38] concentrated on XAI modelling on daily forecasts of bitcoin prices from 2016 through the end of June 2022. They examine the SHAP in the feature selection process and after the modelling. They consider Granger causality, Variational mode decomposition causality, and the proposed SHAP values algorithm for choosing features. After the process, they compared deep learning models, such as Deep Learning Neural Networks and LSTM models, with machine learning models, such as Linear Regression, SVM, RF, XGBoost, and Extra Trees Regression. The two evaluation metrics they employed out of sample R^2 and Root Mean Square Error. Among all models considering both metrics, LSTM with SHAP and LSTM with Granger causality models produce the best results. The best error for the Variational Mode Decomposition result is discovered in a different model called Extra Trees Regression. They employed SHAP for local and global explanation and LIME for local explanation with the Extra Trees model to further analyze the models and XAI.

1.4 The Outline of the Thesis

The study is composed of four distinct chapters. Chapter 1 includes a problem description, method and contributions and literature review. Literature review discusses related studies in four categories, namely portfolio formation, forecasting in stock market index, forecasting in stocks and explainability in time series. Chapter Two, methodology, delves into portfolio models, technical indicators and machine learning algorithms. In addition, XAI is included in this chapter. Chapter Three provides an exhaustive analysis of various machine learning models, specifically focusing on data and risk-aversion prediction and the comparative performance of portfolio optimization outcomes and return prediction. Additionally, this chapter delves into the interpretability of machine learning algorithms by examining the SHAP and LIME results, thereby contributing valuable insights into the underlying decision-making processes of selected models. Chapter Four offers a comprehensive summary of the findings and articulates succinct discussions for subsequent research endeavors.

CHAPTER 2

METHODOLOGY

This chapter consists of five parts: asset prices and returns, portfolio formation, selected technical indicators for market trend, mathematical concepts of machine learning models, and explainable artificial intelligence.

2.1 Asset Allocation

The exploration of return studies occupies a fundamental position within the domain of financial research. Simple net weekly return from holding an asset; in other words an investor buys an asset at its closing price on day t-1 and sells at its closing price on week t:

$$R_t = \frac{P_t}{P_{t-1}} - 1.$$

Simple gross return is defined as $1 + R_t$. Distribution of R_t is not symmetrical. For k periods, it can be written as multiplication of k single periods:

$$1 + R_t(k) = (1 + R_t)(1 + R_{t-1})...(1 + R_{t-1+k}) = \frac{P_t}{P_{t-k}}$$

Calculating return as logarithmic return is very common in the literature. The following equation defines the logarithmic return:

$$r_t \equiv ln(1 + R_t) = lnP_t - lnP_{t-1}$$

Logarithmic return also called as continuously compounded return is a special type of compounded return. Logarithmic return has some advantageous mathematical and empirical properties. First of all, it is time additive which means considering multi-period returns, the total logarithmic return can be expressed as summation of single period returns [17].

$$r_t(k) = ln(1 + R_t(k)) = r_t + r_{t-1}... + r_{t-k+1}$$

Secondly, if R_t is very small r_t can be decomposed into a Taylor series which implies:

$$r_t \approx R_t$$

The last benefit is that the price can not be negative if the logarithmic return has a normal distribution [48]. We will use simple net return in this study.

2.2 Portfolio Models

Maximizing return and minimizing risk are the two objectives of risk-averse investors. The three strategies that are most effective for bi-objective mean-risk portfolios are, first, reduce risk while accounting for a pre-specified level of expected return; second, maximize return while preserving a reasonable level of risk; and third, maximize risk-adjusted mean return while justifying for adjusted risk aversion coefficient. Let us assume we have n assets, $r = (r_1, \ldots, r_n)$ denote the vector of historical return of assets, $\omega = (\omega_1, \ldots, \omega_n)$ is the vector of weights in the portfolio and r_{it} denote the historical return of asset i in period t, $i = 1, ..., n$ and $t = 1, ...T$. Our aim is construct Mean-risk portfolios with varying risk measures and risk aversion parameters which set a trade-off between return and risk. Aiming the unbiasedness we transform all objective functions into normalized version. Portfolio return with corresponding weights calculated as:

$$R_t = \sum_{i=1}^{n} r_{it}\omega_i$$

The mean return of an asset, denoted by μ_i, represents the average return that the asset is expected to generate over a specified period. Furthermore, the mean return of a portfolio is calculated by taking into account the mean returns of all the assets contained within the portfolio:

$$R = \sum_{i=1}^{n} \mu_i\omega_i$$

Calculating the mean return of a portfolio allows investors to estimate the expected performance of their investment portfolio, thereby facilitating more informed decision-making concerning portfolio composition and risk management strategies.
The model that we propose in this thesis is given as follows [47]:

$$max \quad \rho\frac{R - R_{min}}{R_{max} - R_{min}} - (1-\rho)\frac{M - M_{min}}{M_{max} - M_{min}} \tag{2.1}$$

$$s.\ t.\ R = \sum_{i=1}^{n} \mu_i \omega_i \tag{2.2}$$

$$\sum_{i=1}^{n} \omega_i = 1 \tag{2.3}$$

$$\omega_i \geq 0 \tag{2.4}$$

Within equation (2.1), the variable ρ denotes the risk aversion coefficient, which spans between 0 and 1. If ρ is set to 0, the model prioritizes minimizing the risk measure, operating under the assumption that the market will not increase. Conversely, if ρ is set to 1, the model prioritizes maximizing the return, operating under the assumption that the market will increase. To ensure consistency, both the return and risk measure are normalized within the [0, 1] range. Constraints (2.3) and (2.4) state that all the available money must be used and no short selling is allowed. M_{min} and R_{min} refer to the minimum risk measure and expected return, respectively, obtained by minimizing the risk measure without any constraint on the expected return. Similarly, M_{max} and R_{max} refer to the maximum risk measure and expected return, respectively, obtained by maximizing the expected return without any constraint on the risk measure. Next, we give the mean-risk portfolio models that we consider in our analyses.

2.2.1 Mean-Variance Portfolio

Using the assets in the portfolio, investors create a feasible set of portfolios; an efficient portfolio offers the highest expected return of all feasible portfolios with the specified risk [32]. About 70 years ago, Markowitz [61] presented Mean-Variance structure in his paper to obtain efficient portfolios. The Markowitz model, fundamental in constructing efficient portfolios, posits that the optimization of a portfolio can be achieved through the mean-variance framework, which utilizes expected returns

and standard deviation as its core parameters. Investors who are risk averse means they select minimum risk portfolio if their expected returns are equal. Financial assets are homogenous for all investors. It is possible to estimate the mean and standard deviation of financial asset returns, which follow a normal distribution [32] .
The return of the portfolio is equal to:

$$R(\omega) = \omega^\top R$$

The expected return of the portfolio:

$$\mathbb{E}[R(\omega)] = \mathbb{E}\left[\omega^\top R\right] = \omega^\top \mathbb{E}[R] = \omega^\top \mu$$

Variance of portfolio:

$$\begin{aligned}
\sigma^2(\omega) &= \mathbb{E}\left[(R(\omega) - \mathbb{E}[R])(R(\omega) - \mathbb{E}[R])^\top\right] \\
&= \mathbb{E}\left[\left(\omega^\top R - \omega^\top \mu\right)\left(\omega^\top R - \omega^\top \mu\right)^\top\right] \\
&= \mathbb{E}\left[\omega^\top (R-\mu)(R-\mu)^\top \omega\right] \\
&= \omega^\top \mathbb{E}\left[(R-\mu)(R-\mu)^\top\right]\omega \\
&= \omega^\top \Sigma \omega
\end{aligned}$$

Standard deviation:

$$\sigma(\omega) = \sqrt{\omega^\top \Sigma \omega}$$

Markowitz describes how to construct optimizations for efficient portfolios by maximizing the expected return of the portfolio subject to a volatility constraint and minimizing the variance of the portfolio subject to a return constraint. Problems are the same if strong duality holds. The efficient frontier portfolio starts with the minimum variance portfolio, and ends with the maximum return portfolio. Among these portfolios, there exist many optimal portfolios. Moreover, optimization problem can be written as a standard quadratic programming problem optimization problem with $\mathbf{1} = (1_1, ...1_n)$ vector of ones, subject to sum of shares invested in portfolio must be 1 is written as:

$$\begin{aligned}
\omega_\star(\gamma) &= \arg\min \frac{1}{2}\omega^\top \Sigma \omega - \gamma \omega^\top \mu \\
&\text{s.t.} \quad \mathbf{1}_n^\top \omega = 1
\end{aligned}$$

where $\gamma = \phi^{-1}$ is the risk and return trade-off parameter. Solution of all these problems are identical if μ is not a multiple of 1 [12]. This problem with unlimited short

selling allowed can be solved analytically using Lagrange multipliers.
The Lagrange function is:

$$\mathcal{L}\left(\omega;\lambda_0\right) = \omega^\top \mu - \frac{\phi}{2}\omega^\top \Sigma \omega + \lambda_0\left(\mathbf{1}_n^\top \omega - 1\right)$$

The first-order conditions are:

$$\begin{aligned}\partial_\omega \mathcal{L}\left(\omega;\lambda_0\right) &= \mu - \phi\Sigma\omega + \lambda_0 \mathbf{1}_n = \mathbf{0}_n \\ \partial_{\lambda_0} \mathcal{L}\left(\omega;\lambda_0\right) &= \mathbf{1}_n^\top \omega - 1 = 0\end{aligned}$$

From first equation:

$$\omega = \phi^{-1}\Sigma^{-1}\left(\mu + \lambda_0 \mathbf{1}_n\right)$$

Putting ω into second equation we have:

$$\mathbf{1}_n^\top \phi^{-1}\Sigma^{-1}\mu + \lambda_0\left(\mathbf{1}_n^\top \phi^{-1}\Sigma^{-1}\mathbf{1}_n\right) = 1$$

λ_0 calculated as:

$$\lambda_0 = \frac{1 - \mathbf{1}_n^\top \phi^{-1}\Sigma^{-1}\mu}{\mathbf{1}_n^\top \phi^{-1}\Sigma^{-1}\mathbf{1}_n}$$

The solution is then:

$$\omega_\star\left(\phi\right) = \frac{\Sigma^{-1}\mathbf{1}_n}{\mathbf{1}_n^\top \Sigma^{-1}\mathbf{1}_n} + \frac{1}{\phi}\cdot\frac{\left(\mathbf{1}_n^\top \Sigma^{-1}\mathbf{1}_n\right)\Sigma^{-1}\mu - \left(\mathbf{1}_n^\top \Sigma^{-1}\mu\right)\Sigma^{-1}\mathbf{1}_n}{\mathbf{1}_n^\top \Sigma^{-1}\mathbf{1}_n}$$

The set of optimum weights of global minimum variance portfolio is:

$$\omega_\star\left(\infty\right) = \frac{\Sigma^{-1}\mathbf{1}_n}{\mathbf{1}_n^\top \Sigma^{-1}\mathbf{1}_n}$$

If we add risk free asset to our risky portfolio,we get single optimum portfolio, namely tangency portfolio. Our problem is the quadratic programming problem and which forbids short selling, it has not analytical closed-form solution:

$$\begin{aligned}\omega_\star\left(\gamma\right) &= \arg\min \frac{1}{2}\omega^\top \Sigma\omega - \gamma(\omega^\top \mu) \\ &\text{s.t. } \mathbf{1}_n^\top \omega = 1 \\ &\text{s.t. } \omega \geq 0\end{aligned}$$

The Lagrange function is:

$$\mathcal{L}\left(\omega;\lambda_0,\lambda\right) = \omega^\top \mu - \frac{\phi}{2}\omega^\top \Sigma \omega + \lambda_0\left(\mathbf{1}_n^\top \omega - 1\right) + \lambda^\top \omega$$

where $\lambda = (\lambda_1, \ldots, \lambda_n) \geq \mathbf{0}_n$ is the vector of Lagrange coefficients associated with the constraints $\omega_i \geq 0$ [84]. Quadratic solver must be used to compute optimal weights of portfolio [100]. Without normalization, the following equation depicts our optimization issue:

$$\begin{aligned} \omega_\star(\rho) &= \arg\max \rho\, \omega^\top \mu - (1-\rho)\omega^\top \Sigma \omega \\ &\text{s.t. } \mathbf{1}_n^\top \omega = 1 \\ &\text{s.t. } \omega \geq 0 \end{aligned}$$

2.2.2 Mean-CVaR Portfolio

Some scholars propose that risk metrics solely assess the negative aspects of return. Markowitz also suggests using semivariance as a risk measure but due to computational difficulty, he builds modern portfolio theory with variance [32]. The mean-variance model computes above and below returns dispersion in the same way. Due to the limitations of the traditional model, additional risk measures are required. Value at Risk (VaR) which is a percentile of a loss distribution is proposed by J.P.Morgan [57] in 1996, it is coherent only when the underlying distribution is normal and does not give information above the described loss level. Optimization of VaR is difficult when historical return values are used [82]. The concept of Conditional Value at Risk (CVaR), which aims to measure the expected value of losses that exceed the VaR, was introduced by Rockafellar and Uryasev [82] in 2000. VaR is lower bound for CVaR, CVaR is always greater than VaR.

For each weight, loss function represented as $f(\omega, r)$, is conceptualized as a random variable characterized by a distribution within the real number domain, $\mathbb{R}$. This conceptualization stems from the consideration of the random nature of returns. Probability density function of random returns r is denoted as $p(r)$. For fixed ω, cumulative distribution function (CDF) of the loss:

$$\Psi(\omega, \gamma) = \int_{f(\omega,r)\leq\gamma} p(r)dr$$

VaR is a minimum loss value with given confidence level α:

$$VaR_\alpha(\omega) = min\{\gamma \in \mathbb{R} : \Psi(\omega, \gamma) \geq \alpha\}$$

Then, CVaR:

$$CVaR_\alpha(\omega) = \frac{1}{1-\alpha}\int_{f(\omega,r)\geq VaR_\alpha(\omega)} f(\omega,r)p(r)dr$$

It is difficult to optimize this definition so by defining auxiliary function, we want to write CDF as follows:

$$F_\alpha(\omega,\gamma) = \gamma + \frac{1}{1-\alpha}\int (f(\omega,r)-\gamma)^+ p(r)dr$$

CDF has a VaR minimizer convex function and minimum value is equal to CVaR. Since it is hard to work with random returns, historical values of returns are used. Let us assume $r = (r_1, ..., r_T)$ denote the return vector, approximation of the CDF is given as :

$$\hat{F}_\alpha(\omega,\gamma) = \gamma + \frac{1}{(1-\alpha)T}\sum_{t=1}^{T}(f(\omega,r_t)-\gamma)^+ p(r)dr$$

The function $\hat{F}_\alpha(\omega,\gamma)$ is convex and piecewise linear with respect to γ [82]. To solve minimum $CVaR_\alpha(\omega)$ optimization problem with change of variables $(f(\omega,r_t)-\gamma)^+$ with z_t with defining constraints $z_t \geq f(\omega,r_t)-\gamma$ and $z_t \geq 0$. Our optimization problem as without normalization is given as:

$$\begin{aligned}
\omega_\star(\rho) &= \arg\max \rho\, \omega^\top \mu - (1-\rho)\left(\gamma + \frac{1}{(1-\alpha)T}\sum_{t=1}^{T} z_t\right)\\
&\text{s.t. } \mathbf{1}_n^\top \omega = 1\\
&\text{s.t. } \omega \geq 0\\
&\text{s.t. } z_t \geq f(\omega,r_t)-\gamma\\
&\text{s.t. } z_t \geq 0
\end{aligned}$$

In our case since loss function is linear and our problem turns up as a simple linear programming problem, solved with linear solvers [100].

2.2.3 Mean-MAD Portfolio

The third risk measure we consider is the Mean Absolute Deviation. In 1991, Konno and Yamazaki [53] proposed a prominent risk measure which is called mean absolute deviation. As stated earlier, the weekly return of asset i on trading week t computed as:

$$r_{i,t} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}}$$

Mean return of asset i:

$$\bar{r_i} = \frac{1}{T}\sum_{t=1}^{T} r_{i,t}$$

The mean absolute deviation (MAD) for asset i is:

$$MAD_i = \frac{1}{T}\sum_{t=1}^{T} \mid r_{i,t} - \bar{r_i} \mid$$

The calculation of the initial and final values of portfolios is described by the following two equations. Let W_t denote the initial portfolio value:

$$W_t = \sum_{i=1}^{n} \nu_{i,t} P_{i,t}$$

$$W_{t+s} = \sum_{i=1}^{n} \nu_{i,t+s} P_{i,t+s}$$

For the period $[t, t+s)$, assume $\nu_{i,t}$ denote the shares of asset i, then the portfolio return is given as:

$$\begin{aligned} r_{t+s} &= \frac{\sum_{i=1}^{n} \nu_{i,t} P_{i,t+s} - \sum_{i=1}^{n} \nu_{i,t} P_{i,t}}{W_t} \\ &= \sum_{i=1}^{n} \frac{\nu_{i,t} P_{i,t}}{W_t} r_{i,t+s} \\ &= \sum_{i=1}^{n} \omega_{i,t} r_{i,t+s} \end{aligned}$$

where $r_{i,t+s}$ is return on asset i at time $t+s$, and $\omega_{i,t}$ is proportion of total wealth.

Assume we can update weights over a fixed time horizon $t = 1, ..., T$ and have n assets. The MAD for portfolio returns over T interval is calculated as:

$$\frac{1}{T}\sum_{t=1}^{T} \left| \sum_{i=1}^{n} \omega_i (r_{t,i} - \bar{r_i}) \right|$$

where $r_{t,i}$ is the return on asset i at time t, $\bar{r_i}$ is the mean return for asset i, and ω_i is the fraction of the total portfolio that is invested in asset i. Defining two sets of auxiliary variables u_t and v_t , decomposition of absolute deviation can be written as:

$$\left|\sum_{i=1}^{n} \omega_i (r_{t,i} - \bar{r}_i)\right| = u_t + v_t$$

This decomposition allows expressing absolute deviation linearly, making it easier to include in a linear optimization problem, where u_t represents the excess of the portfolio return over the mean return in period t. Mathematically expressed as:

$$u_t \geq \sum_{i=1}^{n} \omega_i (r_{t,i} - \bar{r}_i)$$

v_t represents the amount by which the mean return exceeds the portfolio return in period t. Mathematically expressed as:

$$v_t \geq -\sum_{i=1}^{n} \omega_i (r_{t,i} - \bar{r}_i)$$

Without normalization, our optimization problem—which is a linear optimization problem—can be stated as follows:

$$\begin{aligned} \omega_{\star}(\rho) \quad &= \quad \arg\max \rho\, \omega^{\top} \mu - (1-\rho) \frac{1}{T} \sum_{t=1}^{T} (u_t + v_t) \\ &\text{s.t.} \quad \mathbf{1}_n^{\top} \omega = 1 \\ &\text{s.t.} \quad \omega \geq 0 \\ &\text{s.t.} \quad u_t - v_t = \sum_{i=1}^{n} \omega_i (r_{t,i} - \bar{r}_i) \\ &\text{s.t.} \quad u_t, v_t \geq 0 \end{aligned}$$

Deviation constraint allows us to express the deviation as the difference between two non-negative variables. This is particularly useful in linear programming because it linearizes the absolute value of the deviation [77, 60].

In the following section, we provide important technical indicators that will be utilized in our analysis for feature construction.

2.3 Technical Indicators

To enhance the understanding of stock index trends and individual stock price movements, and to facilitate improved decision-making in financial markets, the application of technical analysis is deemed essential. Technical analysis encompasses a variety of indicators that can be categorized based on their primary function, such as trend analysis, momentum measurement, and volatility assessment [83, 73]. Among these, eleven technical indicators stand out for their efficacy in predicting future movements of the S&P 500 index. These indicators include three different types of moving averages and the Average Directional Movement Index (ADX), which are instrumental in identifying trends. Additionally, the Commodity Channel Index (CCI) and the Parabolic Stop And Reverse (SAR) are also utilized for trend analysis. To gauge momentum, the Relative Strength Index (RSI) is employed. For the analysis of volume, the Ease of Movement (EMV), Money Flow Index (MFI), and the Volume Weighted Average Price (VWAP) are utilized. Lastly, to assess volatility, Bollinger Bands (BB) are applied. This comprehensive set of technical indicators provides a robust framework for analyzing and forecasting the movements of the S&P 500 and stocks, playing a crucial role in the decision-making processes of market participants.

2.3.1 Moving Averages(MA)

Moving averages can help analyze data trends effectively, Simple Moving Average (SMA) is the most used type [70].

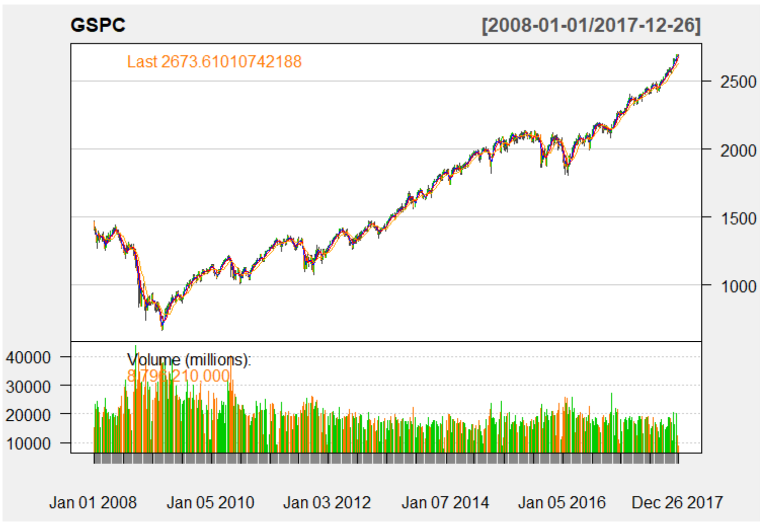


Figure 2.1: SMA Plot of S&P 500

In Figure 2.1, each SMA is represented by a distinctive color for clarity in visualization: blue for the 3-period SMA, red for the 5-period SMA, and orange for the

10-period SMA. SMA can be formulated as a simple average formula for past n periods,and N is final period.

$$SMA = \frac{\sum_{i=0}^{n-1} \ Close \ Price_{N-i}}{N}$$

Instead of equally weighted approach, some technical analysts use linearly weighted moving average (WMA) giving more weight into recent price:

$$WMA = \frac{\sum_{i=0}^{n-1} (n-i) Close \ Price_{N-i}}{\sum_{i=0}^{n-1} (n-i)}$$

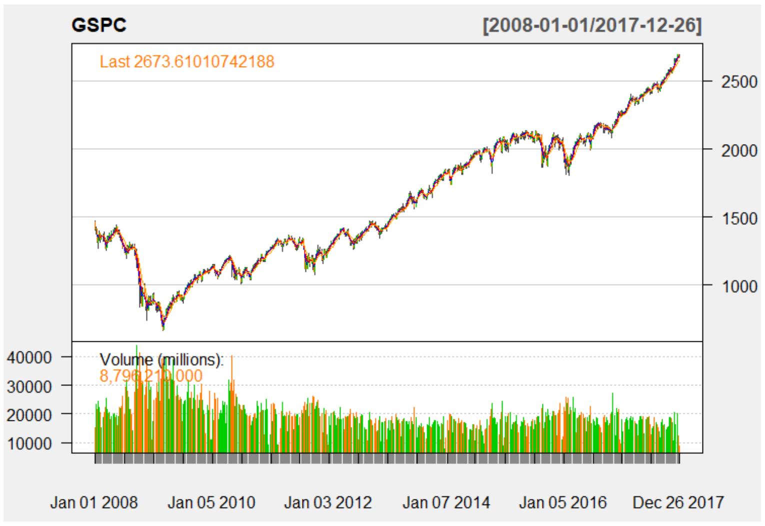


Figure 2.2: WMA Plot of S&P 500

In Figure 2.2, each WMA is represented by a distinctive color for clarity in visualization: blue for the 3-period WMA, red for the 5-period WMA, and orange for the 10-period WMA. Instead of WMA, some technical analysts use Exponential Moving Average (EMA) decreasing weight exponentially:

$$EMA = \sum_{i=0}^{n-1} [\alpha(1-\alpha)^i Close \ Price_{N-i}]$$

where α is smoothing constant, equals $2/(n+1)$ [23].

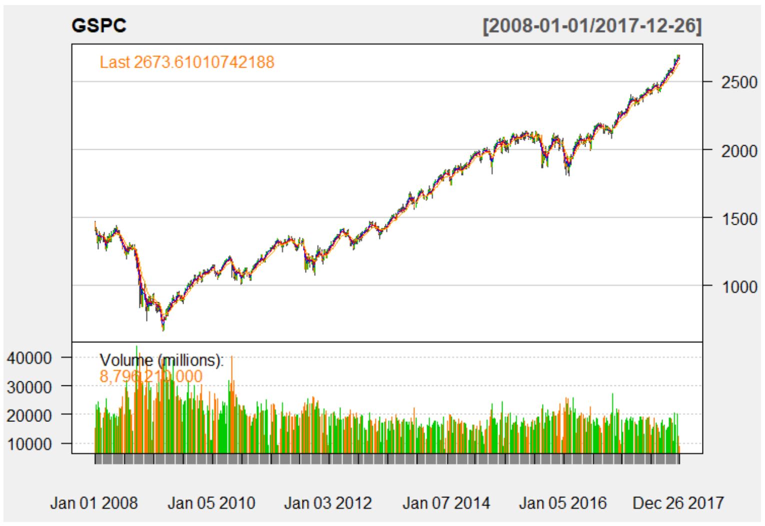


Figure 2.3: EMA Plot of S&P 500

In Figure 2.3, each EMA is represented by a distinctive color for clarity in visualization: blue for the 3-period EMA, red for the 5-period EMA, and orange for the 10-period EMA.

2.3.2 Average Directional Movement Index(ADX)

ADX was developed by J.Welles Wilder [98], it is calculated as exponential moving average of Directional Movement Index(DX) over timeperiod. The DX is calculated as combining a positive directional indicator (DI$^+$) and negative directional indicator (DI$^-$) which are calculated based on directional movements(DM) . ADX can be formulated as step by step [91]:

$$DM_t^+ = \begin{cases} max\ (High_t - High_{t-1}, 0) & \text{if} \quad High_t - High_{t-1} \geq Low_t - Low_{t-1} \\ 0 & \text{if} \quad otherwise \end{cases}$$

$$DM_t^- = \begin{cases} max\ (Low_t - Low_{t-1}, 0) & \text{if} \quad High_t - High_{t-1} \leq Low_t - Low_{t-1} \\ 0 & \text{if} \quad otherwise \end{cases}$$

$$DI_t^+ = \frac{DM_t^+}{TR_N}$$

$$DI_t^- = \frac{DM_t^-}{TR_N}$$

TR_N is true range prior N time period calculated as max of difference of below prices or 0.

- HP$_t$-LP$_t$
- HP$_t$-CP$_{t-1}$
- LP$_t$-CP$_{t-1}$
  where HP: High price and LP: Low price, CP: Close Price.

Then DI_t^+ and DI_t^- smoothed over N time period:

$$DI_N^+ = \Sigma \frac{DM_N^+}{TR_N}$$

$$DI_N^- = \Sigma \frac{DM_N^-}{TR_N}$$

Then DX can be calculated as:

$$DX_t = \frac{DI_t^+ - DI_t^-}{DI_t^+ + DX_t^-}$$

Then using EMA, ADX calculated as:

$$ADX_N = 100 * EMA\,(DX, N)$$

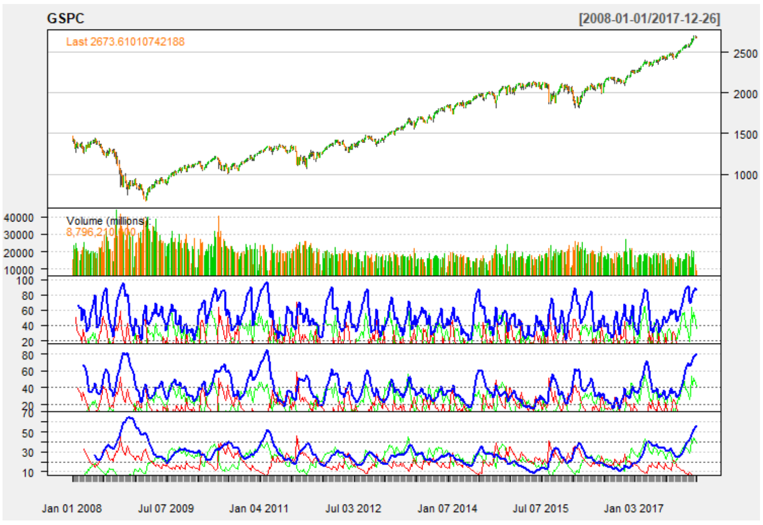


Figure 2.4: ADX Plot of S&P 500 with 3,5,10 periods

In Figure 2.4, each ADX period is represented by distinctive boxes for clarity in visualization: the first for the 3-period ADX, the second for the 5-period ADX, and the third for the 10-period ADX.

2.3.3 Relative Strength Index(RSI)

RSI is one of the most popular momentum indicator and it was developed by J.Welles Wilder [70]. The momentum line is smoothed by RSI, and it aids in the emergence of boundaries ranging from 0 to 100. The term "relative strength(RS)" refers to the proportion of average upward and downward movements [70]. For RSI formula,first consider RS [83]:

$$RS = \frac{Mean\ of\ Close - Previous\ Close\ on\ Up\ Movement}{Mean\ of\ Previous\ Close - Current\ Close\ on\ Down\ Movement}$$

$$RSI = 100 - \frac{100}{1 + RS}$$

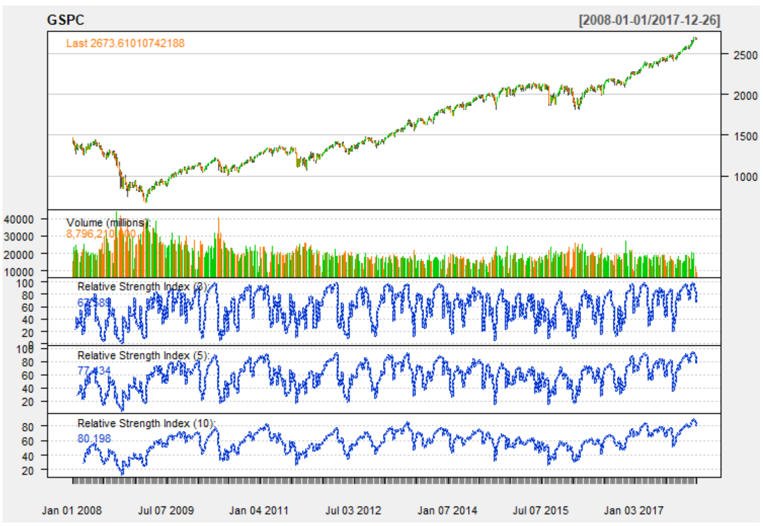


Figure 2.5: RSI Plot of S&P 500 with 3,5,10 periods

In Figure 2.5, each RSI period is represented by distinctive boxes for clarity in visualization: the first for the 3-period RSI, the second for the 5-period RSI, and the third for the 10-period RSI.

2.3.4 Commodity Channel Index(CCI)

CCI is one of the momentum indicator and it was developed for commodities by Donald, R [70]. It can be represented as a four-step calculation [66].

$$Typical\ Price = \sum_{n=1}^{N} \frac{High + Low + Close}{3}$$

$$Moving\ Average\ (MA) = \frac{Typical\ Price}{N}$$

$$Mean\ Deviation = \frac{\sum_{n=1}^{N} |Typical\ Price - MA|}{N}$$

$$CCI = \frac{Typical\ Price - MA}{L * Mean\ Deviation}$$

where N is number of periods and L is Lambert coefficient which is equal to 0.015.

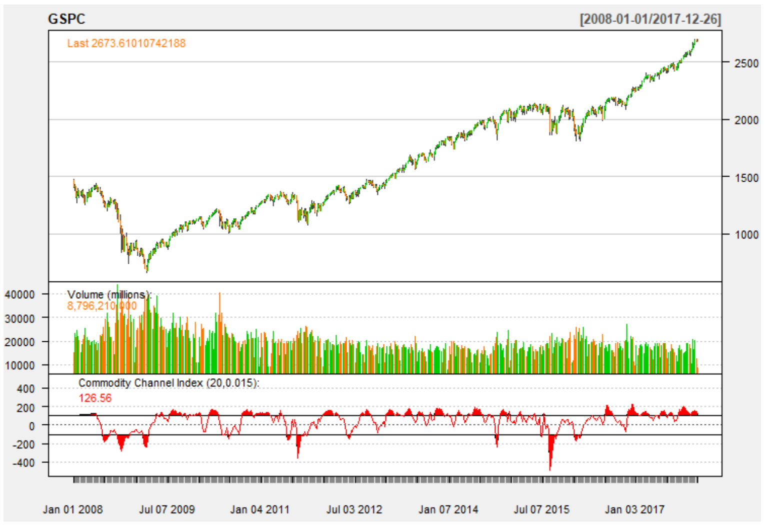


Figure 2.6: CCI Plot of S&P 500 with 20 period

In Figure 2.6, the 20-period CCI is depicted in red at the bottom of the figure.

2.3.5 Ease of Movement(EMV)

This momentum indicator assesses volume momentum by examining the midpoint values of successive periods, and then dividing this by the volume of trades divided by the high-low price difference [23].

$$Midpoints\ of\ range = \frac{HP_t + LP_t}{2} - \frac{HP_{t-1} + LP_{t-1}}{2}$$

$$EMV = \frac{Midpoints\ of\ range}{Volume_t/(HP_t - LP_t)}$$

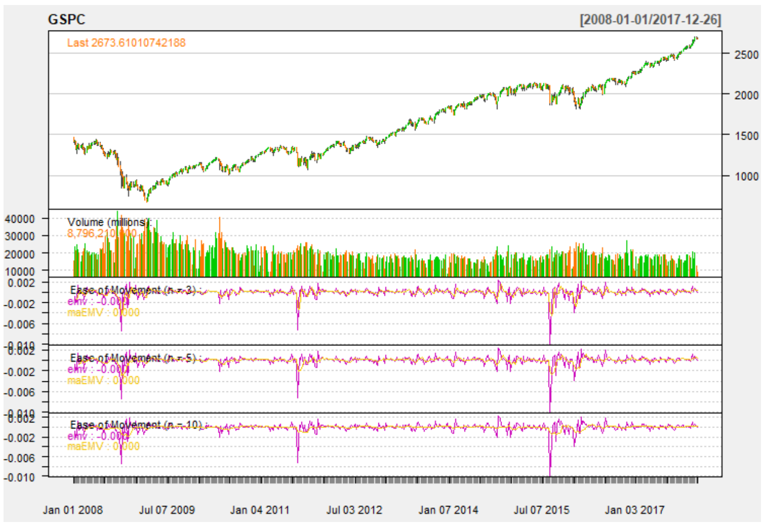


Figure 2.7: EMV Plot of S&P 500 with 3,5,10 periods

In Figure 2.7, each EMV period is represented by distinctive boxes for clarity in visualization: the first for the 3-period EMV, the second for the 5-period EMV, and the third for the 10-period EMV.

2.3.6 Bollinger Bands(BB)

BB uses standard deviation in its calculations and this makes them a volatility indicator. As a mean value, BB employ the moving average. BB are made up of three bands, known as the upper, middle, and lower bands.σ is standard deviation, x_i is data points, $\bar{x}$ is average of data points and N is period number. BB calculation as follows:

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})^2}{N}}$$

$$Middle\ Band = SMA\ (N)$$

$$Upper\ Band = Middle\ Band + 2\sigma$$

$$Lower\ Band = Middle\ Band - 2\sigma$$

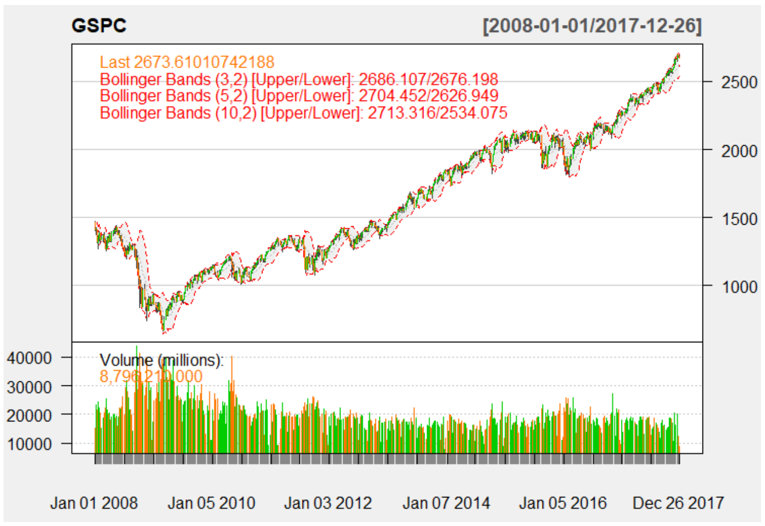


Figure 2.8: BB Plot of S&P 500 with 3,5,10 periods

In Figure 2.8, each "BB" period is shown on the time series plot.

2.3.7 Parabolic Stop-and-Reverse(SAR)

J. Welles Wilder's innovative tool has the ability to generate fresh trading signals by analyzing market trends. With the power of two formulas, it's a game-changing solution for investors.
For new buy signals:

$$RSAR = Prior\ SAR + Prior\ AF(Prior\ EP - Prior\ SAR)$$

For new sell signals:

$$FSAR = Prior\ SAR - Prior\ AF(Prior\ SAR - Prior\ EP)$$

where AF is Acceleration Factor ranges between 0.02 to maximum 0.2 with the increment 0.02 and EP is Extreme Points [23].

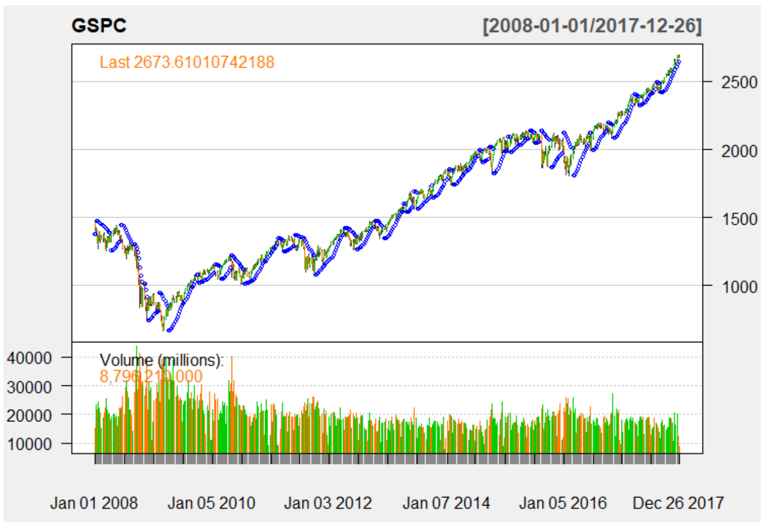


Figure 2.9: SAR Plot of S&P 500

In Figure 2.9, SAR is shown on the time series plot.

2.3.8 Money Flow Index (MFI)

MFI is a momentum indicator calculated with four steps as follows, firsly selected period typical price is calculated:

$$Typical\ Price = \frac{High + Low + Close}{3}$$

Next, Money Flow is calculated as follows:

$$Money\ flow = Typical\ Price * Volume$$

$$Money\ flow = \begin{cases} Positive & \text{if} \quad Typical\ Price_t > Typical\ Price_{t-1} \\ Negative & \text{if} \quad otherwise \end{cases}$$

Positive money flow is the total positive period, negative money flow is the total negative period, and the money ratio is the division of the two[2].

$$Money\ Ratio = \frac{Positive\ Money\ flow}{Negative\ Money\ Flow}$$

Finally, MFI calculated as:

$$MFI = 100 - \frac{100}{1 - Money\ Ratio}$$

Utilizing the MFI indicator can assist in making wise investment decisions. When the MFI surpasses 80, it demonstrates overbought conditions. Therefore, it is advisable to purchase before it reaches that threshold. Similarly, if the MFI dips below 20, it indicates oversold conditions. Hence, it would be prudent to sell before it falls further [75].

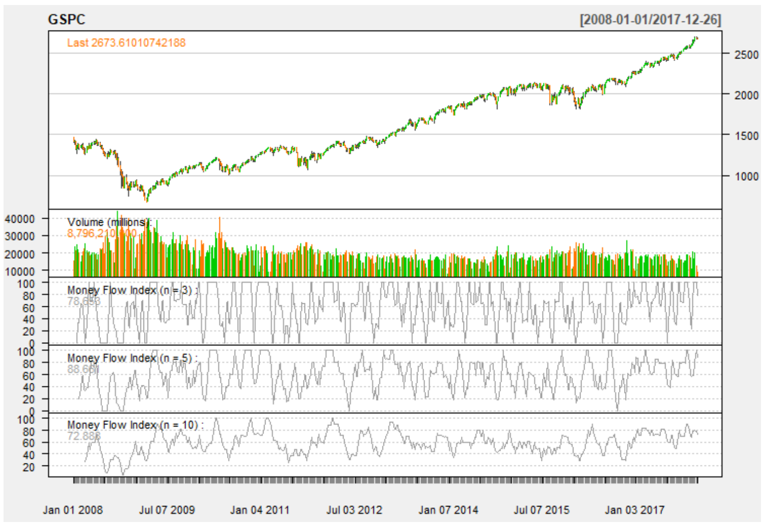


Figure 2.10: MFI Plot of S&P 500 with 3,5,10 periods

In Figure 2.10, each MFI period is represented by distinctive boxes for clarity in visualization: the first for the 3-period MFI, the second for the 5-period MFI, and the third for the 10-period MFI.

2.3.9 Volume Weighted Average Price(WVAP)

The calculation of WVAP involves dividing the overall trade value by its volume, as measured over a specified timeframe [36].

$$Typical\ Price = \frac{High + Low + Close}{3}$$

$$WVAP = \frac{\sum_{i=0}^{n-1}(Volume_{N-i} * Typical\ Price_{N-i})}{\sum_{i=0}^{n-1} Volume_{N-i}}$$

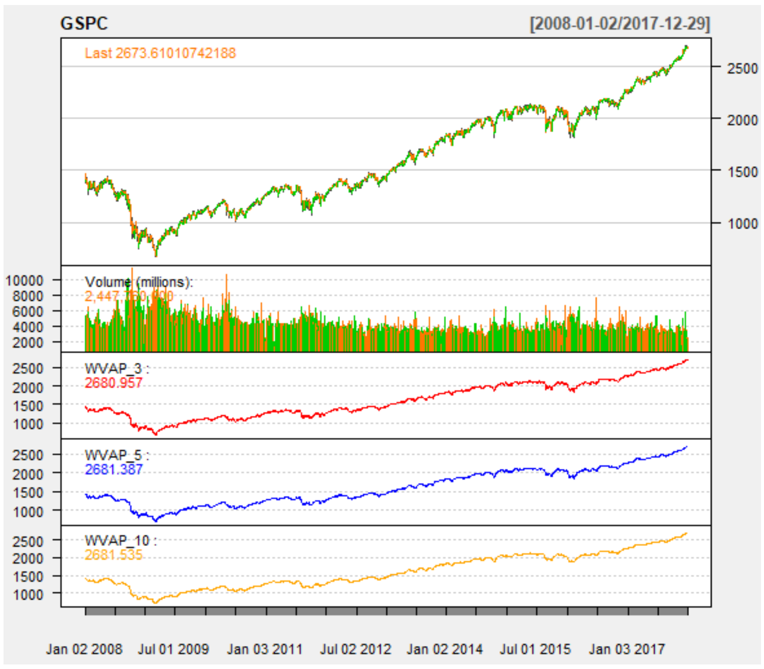


Figure 2.11: WVAP Plot of S&P 500 with 3,5,10 periods

In Figure 2.11, each WVAP period is represented by distinctive boxes for clarity in visualization: the first for the 3-period WVAP, the second for the 5-period WVAP, and the third for the 10-period WVAP. Following the development of our technical analysis variables, it is apparent that machine learning plays a vital role in predictive analytics. Consequently, we elucidate the mathematical underpinnings of selected models designed for forecasting trends in the S&P 500 in the subsequent section.

2.4 Machine Learning Algorithms

Learning is an integral component of a researcher's journey. In earlier times, with the help of mathematics, scientists tried to learn data with the induction method, which was a theoretical approach. Mathematicians want to construct rules and generalize the rule for a statement. In statistics, it was crucial to make inferences from a sample; they wonder how they model the world. The world has many data; understanding it, making usable models, and estimating the model parameters are attractive for statisticians. Thanks to the invention of computers, mathematicians and statisticians can automate the learning process. The mathematician Alan Turing asked the question "Can machines think?" and discussed programming of machines in his article in 1950

[96]. Like Turing, scientists work to teach machines how to learn using algorithms. Thanks to advances in machine learning, the prediction of price direction is a research topic in the fields of finance, statistics, and computer science.

Today, with much data, we live in a significant data era. Access to data is faster than ten years ago; every minute, new algorithms which are designed to predict the learning process has better accuracy than old algorithms.

Three categories of machine learning algorithms can be distinguished: reinforcement learning, unsupervised learning, and supervised learning. Classification and regression algorithms are the two categories that constitute supervised algorithms.

- Classification: It is called as discriminative analysis in statistics [65, 41] and called as pattern recognition in computer engineering [31].
- Regression: Its objective is to forecast numerical value such as weekly price of a stock.

Learning the class of the data is the aim of classification models. One problem that this thesis considers is understanding the S&P 500's up and down movements. Let us assume two variables that affect S&P 500 trend prediction are RSI and SAR. Thus, two features, x_1 and x_2, can represent each move.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

and **x**'s label is given as:

$$\mathbf{g} = \begin{cases} 1 & \text{if} \quad \mathbf{x} \; is \; UP \\ 0 & \text{if} \quad \mathbf{x} \; is \; DOWN \end{cases}$$

If the training set has N examples, it can be represented as $\mathbf{X}=\{\mathbf{x}^t,\mathbf{g}^t\}_{t=1}^{N}$, t is different samples in the dataset.

After further insight from finance experts if the RSI and SAR are between certain ranges, then the movement is considered as "up", these certain ranges are given as:

$$IF \; O_1 < RSI < O_2 \; AND \; S_1 < SAR < S_2 \; then \; UP \tag{2.5}$$

In our class, **C** represents upward movement with an equation of 2.5, assuming it is a rectangle taken from the hypothesis class in the RSI-SAR price space. The learning algorithm aims to find the most appropriate hypothesis, $h(\mathbf{x})$, which is closest to **C**.

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad h\ classifies\ \mathbf{x}\ as\ UP \\ 0 & \text{if} \quad h\ classifies\ \mathbf{x}\ as\ DOWN \end{cases}$$

The error of $h(\mathbf{x})$ given **X**:

$$E(h|\mathbf{X}) = \sum_{n=1}^{N} 1(h(\mathbf{x}^t) \neq \mathbf{g}^t)$$

where $1(h(\mathbf{x}^t) \neq \mathbf{g}^t)$ is 1 if $h(\mathbf{x}^t) \neq \mathbf{g}^t$ and is 0 if $h(\mathbf{x}^t) = \mathbf{g}^t$.

Each (0_1^h,0_2^h,S_1^h,S_2^h) construct a hypothesis and we search for the best values of these parameters for the hypothesis given **X**, which only includes upward movements. Since RSI and SAR are real valued, there exist infinitely many hypotheses, and some hypotheses have correct predictions for **X**. For any hypothesis with boundaries, the upper boundary is called the most general hypothesis set, and the lower boundary is called the most specific hypothesis set; these hypotheses are called consistent. **X** is assumed to be large enough for unique the most specific and general hypothesis and h is chosen for increasing margin, which is the distance between boundary and the nearest samples to it. Margin gives insight for classification for the boundary and data samples, there exist a need for a new version of the error function which can also check the margin. A hypothesis with the largest margin gives the most significant classification.

Up to this point, we have examined a basic hypothesis class. However, the presence of unobservable variables as noise would indicate that our class cannot remain simple; it must be more complex in order to account for this. Complex hypotheses need complex models that fit the data. The set of assumptions for the classification are called as inductive bias, and hypothesis class is one of the inductive bias. For the approximation of $\mathbf{g}^t$, we need to choose three things: inductive bias (model), loss function, and optimization procedure. They will be shared in the following sections.

Apart from classification models, regression models aim to learn real-valued outcomes. This study focuses on predicting asset prices, specifically the price of stocks for regression models prediction.

$$\mathbf{g} = Price$$

where $Price \in \mathbb{R}$ denotes that $Price$ belongs to the set of real numbers [5].

2.4.1 Logistic Regression (LR)

One discriminant-based classification approach, LR, makes assumptions regarding the discriminant between class labels. The LR, which uses the term odds, is a probabilistic model. The odds related to a predicted event can be stated as the ratio of the probability that the event will occur to the probability that it will not occur :

$$\frac{p}{1-p}$$

For brevity, assuming we have two classes, C_1 and C_2, we want to predict the conditional probability of class 1 given training data as $\mathbf{X}$=$\{\mathbf{x}^t$,$\mathbf{g}^t\}_{t=1}^{m}$ and probability $p := P(C_1|\mathbf{x})$.

$$logit(P(C_1|\mathbf{x})) = ln\frac{P(C_1|\mathbf{x})}{1-P(C_1|\mathbf{x})} = ln\frac{P(\mathbf{x}|C_1)}{P(\mathbf{x}|C_2)} + ln\frac{P(C_1)}{P(C_2)} = z$$

Logit function equals z, according to Bayes' rule usage of the equation and $[0, 1]$, $\mathbb{R}$ are the domain and codomain of the logit function, respectively.

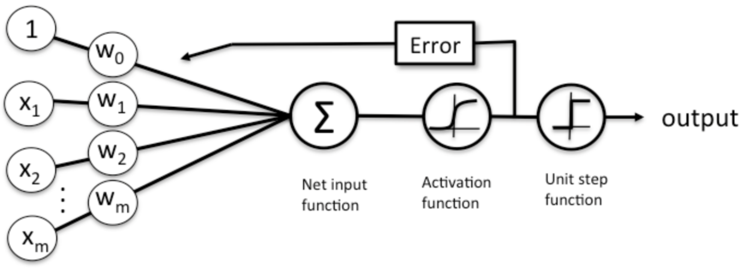


Figure 2.12: Logistic Regression Structure [79]

In Figure 2.12, the structure of logistic regression is visualized. The mathematical expression of input value z is a linear combination of the features ($\mathbf{x}$) and the set of parameters (**w**).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_m \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_1 \\ . \\ . \\ . \\ w_m \end{bmatrix}$$

$$z = \mathbf{w}^T\mathbf{x} = w_0x_0 + w_1x_1 + ... + w_mx_m = \sum_{i=0}^{m} w_ix_i,$$

Given the significance of probability, we require the inverse of the logit function, frequently referred to as the sigmoid function and represented as $\phi(z)$, which is the activation function of logistic regression. Based on Figure 2.13, its S-shaped function indicates the likelihood of results [79].

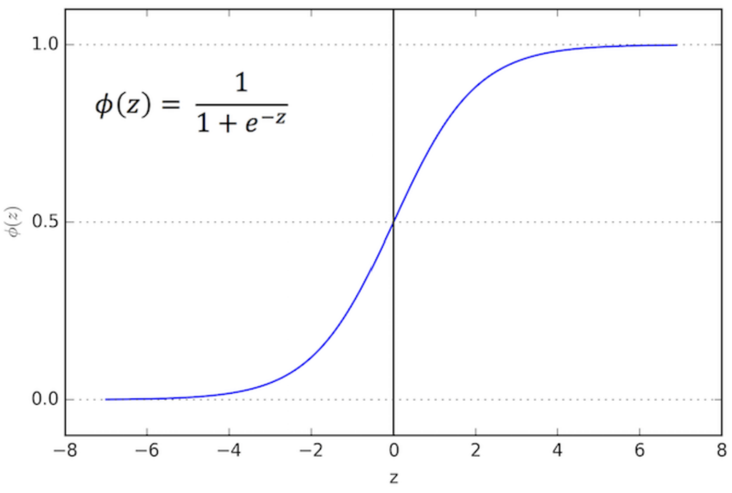


Figure 2.13: Logistic Sigmoid Function[79]

Predicted probability calculated with threshold function denoted as:

$$y = \widehat{P}(C_1|\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \phi(z) \geq 0.5 \\ 0 & \quad \text{otherwise} \end{cases}$$

2.4.1.1 Loss Function

This method operates by learning parameters, specifically, the weight vector denoted as $\mathbf{w}$ and the bias term w_0, which are pivotal in delineating the decision boundary between different classes in the feature space. For this, maximum likelihood estimation is utilized, with the following definition of the likelihood function, assuming the independence of sample data. Data as **X**=$\{\mathbf{x}^t$,$\mathbf{g}^t\}_{t=1}^m$, where $\mathbf{g}^t$=1 if x $\in C_1$ and $\mathbf{g}^t$=0 if x $\in C_2$ and assume $\mathbf{g}^t$, given $\mathbf{x}^t$ has Bernoulli distribution with probability y^t. So, the sample likelihood function is

$$\mathcal{L}(\mathbf{w}, w_0)|\mathbf{X}) = \prod_{t=1}^{m} P(y^t|x^t;\mathbf{w}) = \prod_{t=1}^{m}(y^t)^{(\mathbf{g}^t)}(1-y^t)^{(1-\mathbf{g}^t)}$$

The concept of log-likelihood is frequently employed in the realm of statistical analysis and machine learning due to its facilitation of maximization. However, there is an alternate approach aimed at achieving similar objectives, known as the cross-entropy error function. This function, which can be minimized, is mathematically represented as $E = -\log(\mathcal{L})$, where E denotes the cross-entropy error, and $\mathcal{L}$ symbolizes the log-likelihood of the model. We can explicitly indicate it as:

$$E(\mathbf{w}, w_0|\mathbf{X}) = -\sum_{t=1}^{m} \mathbf{g}^t logy^t + (1-\mathbf{g}^t)log(1-y^t)$$

The loss function does not have an analytical solution, so we require optimization methods to minimize it [5].

2.4.1.2 Optimization Methods

Gradient Descent

In the pursuit of optimizing convex error functions, a plethora of methodologies have been established. Specifically, gradient descent emerges as an efficacious strategy for resolving the cross-entropy problem. This technique involves the computation of weight coefficients through the application of the partial derivative of cross-entropy, as delineated below:

Derivative of cross-entropy error with respect to j th weight:

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \sum_{1}^{m} (\frac{\mathbf{g}^t}{y^t} - \frac{1-\mathbf{g}^t}{1-y^t}) y^t (1-y^t) x_j^t = \eta \sum_{1}^{m} (\mathbf{g}^t - y^t) x_j^t, j = 1, ...d$$

Derivative of cross-entropy error with respect to w_0:

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_{1}^{m} (\mathbf{g}^t - y^t)$$

Single update becomes:

$$w_j = w_j - \eta \nabla E(w_j)$$

where η is called stepsize [5].

Liblinear

It is an optimization method used for linear classifiers, particularly for solving unconstrained optimization problems in datasets with binary output. It is represented as:

$$\underset{\mathbf{w}}{min} \frac{1}{2} \mathbf{w}^t \mathbf{w} + C \sum_{1}^{m} E(\mathbf{w}, w_0 | \mathbf{X})$$

where C is penalty parameter. For Logistic Regression $log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})$ is error function. This method uses an additional dimension for the solution [35].

2.4.1.3 Regularization

In order to achieve optimal data fitting, regularization techniques can be used to adjust model complexity and prevent issues with overfitting or underfitting.

L2 Regularization

It uses the Euclidean norm, which lessens complexity by introducing a penalty term into the cross entropy. L2 penalty term denoted as:

$$\frac{\lambda}{2} ||\mathbf{w}||_2 = \frac{\lambda}{2} \sum_{1}^{m} w_j^2$$

The cross-entropy error equation transforms into the equation below:

$$E(\mathbf{w}, w_0|\mathbf{X}) = -\sum_{t=1}^{m} \mathbf{g}^t logy^t + (1-\mathbf{g}^t)log(1-y^t) + \frac{\lambda}{2}\sum_{1}^{m} w_j^2$$

Where λ is the tuning parameter that needs to be adjusted. As λ increases, the regularization strength also increases, but if λ is too high, it makes the weights approach 0.

L1 Regularization

This regularization uses L1 norm and L1 penalty term, The L1 norm of the weight vector **w** is given by:

$$\|\mathbf{w}\|_1 = \sum_{1}^{m} |w_j|$$

This term is added to the loss function to penalize large coefficients in a linear model [80]:

$$E(\mathbf{w}, w_0|\mathbf{X}) = -\sum_{t=1}^{m} \mathbf{g}^t logy^t + (1-\mathbf{g}^t)log(1-y^t) + \lambda\sum_{1}^{m} |w_j|$$

This makes L1 regularization especially valuable for performing feature selection in high-dimensional data.

2.4.2 K Nearest Neighbour (KNN)

KNN is a memory-based learning algorithm introduced in the 1960s for classification [99]. It makes no assumptions about the dataset for modelling purposes and assumes that similar inputs have similar outputs. This algorithm leverages the principle of data memory to categorize newly introduced data, represented as x_{new}, by applying a similarity metric to ascertain the most analogous data points to x_{new}. Since KNN is an instance-based technique computing a new instance requires O(N) memory. When the calculation of large instances takes a long time, and the complexity of KNN relies on the training set , in most situations, N>d, where N is the number of samples and d is the number of features, computational complexity is a concern for KNN [5]. A KNN classifier with Minkowski distance takes computing time O(Nd) [25].

2.4.2.1 Distance metrics

This metrics serve as a fundamental determinant of similarity or dissimilarity between data points within a given feature space. The distance metric, denoted as d, between a training sample x and a new observation x_{new}, can be rigorously defined by the following axioms which ensure its mathematical consistency and applicability:
(1)d$(x, x_{new}) \geq 0$
(2)d(x, x_{new})=0 only if x=x_{new}
(3)d(x, x_{new})=d(x_{new}, x)
(4)d(x, z)≥d(x, x_{new})+d(x_{new}, z)
Fulfilling these criterias allow for the effective functioning of KNN by accurately modeling the distances or dissimilarities within the feature space of the dataset. The common metrics used in the KNN model include:

Minkowski Distance

Let us call feature set as F. General formula for Minkowski Distance is denoted as MD:

$$MD_p(x_{new}, x_i) = (\sum_{f \in F} |x_{new_f} - x_{if}|^p)^{1/p}$$

When p=1, it is called as Manhattan Distance and p=2 called as Euclidean distance. When p= ∞, this special form of Minkowski distance called as Chebyshev distance:

$$MD_\infty(x_{new}, x_i) = \underset{f \in F}{max} |x_{new_f} - x_{if}|$$

Mahalanobis Distance

$$d_M = \sqrt{(x - x_{new})^T \mathbf{S} (x - x_{new})}$$

where **S** is covariance matrix of the data [25].

2.4.2.2 Search Strategies

The KNN model uses various strategies to determine the k nearest neighbors for its utilization.

Brute Force Approach

The methodology under discussion is commonly identified as an exhaustive search technique, pivotal in computing similarity distances between each query point as x_{new} and various data sample points. This process involves a meticulous sorting of the computed distances to accurately identify the k-nearest neighbours. Despite the straightforward nature of this search mechanism, it is imperative to acknowledge its considerable computational complexity. To elucidate, consider a scenario where the dataset comprises n query points represented as x_{new}, m sample points, and d distinctive features. In this context, the computation of distance measurements necessitates a temporal complexity of O(mnd). Concurrently, the sorting procedure is anticipated to demand O(nm$\log$m) time. Thus, the cumulative time required by the brute force approach to conclude the computation effectively aggregates to O(mnd) + O(nm$\log$m). Interestingly, when each data point functions simultaneously as both the query and reference point—thus setting n equal to m—the computational complexity morphs into O(n^{2}d) + O(n^2 $\log$n). This particular analysis underscores the challenges inherent in managing substantial data volumes, especially when such calculations are necessitated individually for each new instance. However, it also highlights the potential of parallel processing as a viable strategy to mitigate computing time. Furthermore, the exploration of alternate methodologies continues to be an area of significant interest and research within the field [6].

kD-Trees

kD-Trees refers to the k features that constitutes k dimensional space, it was first used in the 1990s [99]. The prolonged search technique for KNN is kD-Trees [11]. The architectural design of kD-Trees is predicated on the binary tree model, wherein the division of training data is meticulously arranged into hierarchical branches, culminating in leaves. This structure facilitates an efficient partitioning of space, making it adept at managing high-dimensional data. An illustrative example of this mechanism can be observed in a bi-dimensional dataset scenario; here, kD-Trees proficiently bisect the dataset through hyperplanes that are orthogonal to either the X or Y axis. This partitioning strategy hinges on the analysis of feature variance within the dataset, selecting the feature with the maximal dispersion as the axis for division. The optimal position for the hyperplane is determined by computing the mean value of the selected feature, ensuring a balanced distribution of data points across the tree. This process is recursively applied, with each iteration refining the search space until the most proximate neighbors are delineated. The search time is $O(\mathrm{d}log\mathrm{k})$ which is increasing when k is so large, and kD-Trees passes the time of the Brute Force search [11]. A critical examination of kD-Trees in the context of nearest neighbor searches reveals a nuanced limitation: the employment of hyperrectangles for bounding nearest neighbors, though effective, is not universally optimal.

Ball Trees

In n-dimensional Euclidean space, a ball is a region surrounded by a hypersphere. In the KNN approach, a ball tree generates k-dimensional hyperspheres for training data, and the balls are sorted as leaves of the tree. The implementation can be done top-down or bottom-up, as it is similar to that of the hierarchical clustering problem. Unlike kD-Trees, there is no need to partition the entire space, and the sibling regions of ball trees can intersect [72]. These characteristics give ball trees their strength. Ball trees are a sophisticated method that, unlike kD-trees, addresses the curse of dimensionality problem by building metric trees to provide a solution for high-dimensional data. In comparison to kD-Trees, ball trees emphasize the construction of superior metric to the feature space. However, this great property for Ball trees restricts the applications of Dynamic Time Warping, a similarity metric for time series data [25].

Distance metrics and the number of similar data points must be specified in the KNN implementation. KNN obtains the final class prediction of the query by a simple majority vote of k data points. The equation provided:

$$Vote(y_j) = \sum_{j=1}^{k} \frac{1}{d(x_{new}, x_c)^p} 1(y_j, y_c)$$

y_j is prediction of neighbour x_c and p is usually selected as 1 [25]. The function $1(y_j, y_c)$ is an indicator that returns 1 if the class label y_c same with prediction, and 0 otherwise:

$$1(y_j, y_c) = \begin{cases} 1 & \text{if} \quad y_j = y_c \\ 0 & \quad \text{otherwise} \end{cases}$$

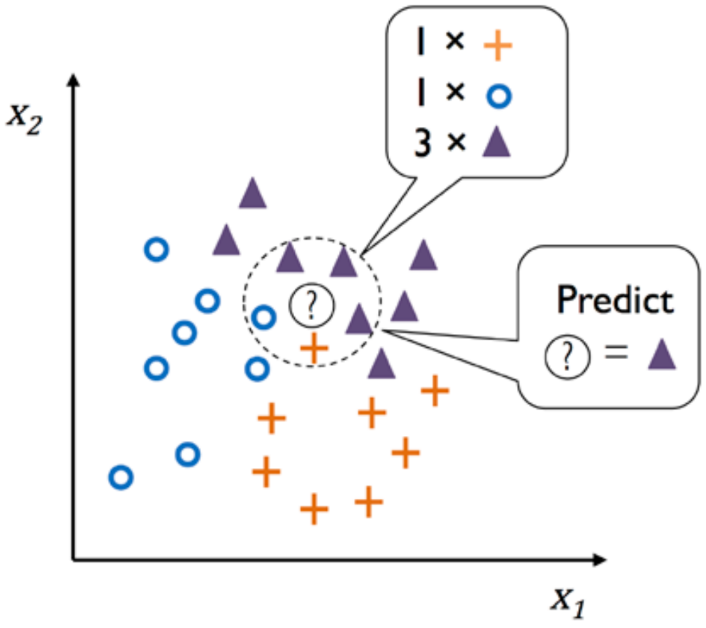


Figure 2.14: KNN with two features[79]

In Figure 2.14, the visualization of KNN classification with two features is provided.

2.4.3 Support Vector Machine (SVM)

SVM can be used to solve both linear and nonlinear classification problems. If SVM is used for a classification problem, it's known as a Support Vector Classification (SVC) [43]. Classes are not always linearly separable in feature space; hence, SVM uses feature transformation to achieve linear separability to deal with nonlinearity. SVM's main goal is to identify the hyperplane as the decision boundary that best separates the classes. A hyperplane for each class establishes the margin, which is maximized during training. Support vectors are training data points close to the hyperplanes, and they are crucial in determining the position of the decision boundary. The sample is $\mathbf{X}$=\{$\mathbf{x}^t$,$\mathbf{g}^t$\}, where $\mathbf{g}^t$=1 if x $\in C_1$ and $\mathbf{g}^t$=-1 if x $\in C_2$. If our classes can be linearly separated and there are two features, the SVM visualization appears as shown in the following figure.

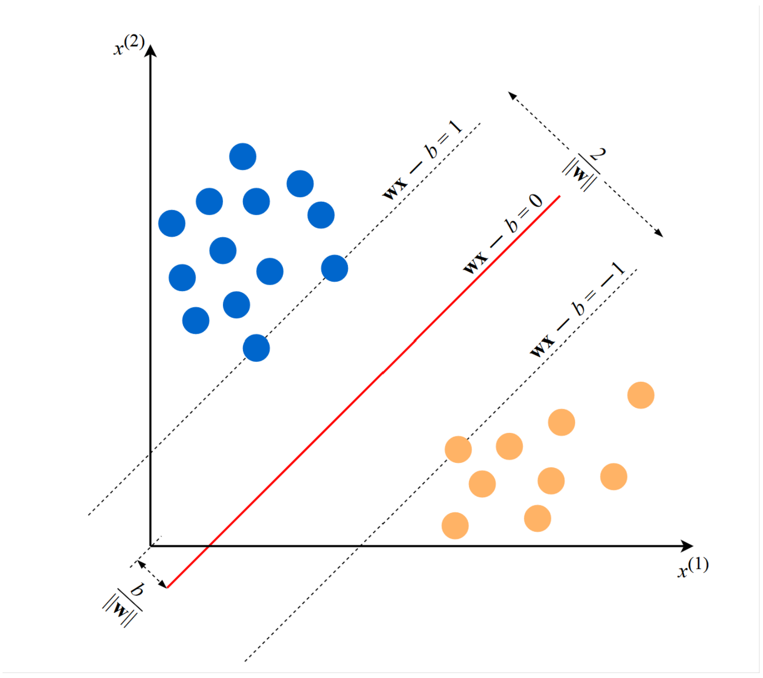


Figure 2.15: SVM[15]

In Figure 2.15, the visualization of SVM classification with two features is provided. There are numerous hyperplane options, and SVMs are well-known for their ability to learn new data by selecting a greater distance between the decision boundary and the data point. Here is the decision boundary equation:

$$\mathbf{w}^T\mathbf{x}^t + b = 0$$

where $\mathbf{w}$ is the weight vector, b is a real number.

To discover the optimal hyperplane that maximizes the margin between the two classes, SVMs enforce the following constraints:

- Hyperplane 1: $\mathbf{w}^T\mathbf{x}^t$+b$\geq$+1 for $\mathbf{g}^t$=+1

- Hyperplane 2: $\mathbf{w}^T\mathbf{x}^t$+b$\leq$-1 for $\mathbf{g}^t$=-1

Therefore, data above hyperplane one and below hyperplane two belongs to C_1 and C_2, respectively. For convenience, these two inequalities can be expressed as:

$$\mathbf{g}^t(\mathbf{w}^T\mathbf{x}^t + b) \geq 1$$

The distance between the hyperplanes is given by $\frac{2}{||\mathbf{w}||}$. Maximizing this distance is equivalent to minimizing the value of $\frac{1}{2}||\mathbf{w}||^2$.

$$min\ \frac{1}{2}||\mathbf{w}||^2\ subject\ to\ \mathbf{g}^t(\mathbf{w}^T\mathbf{x}^t + b) \geq 1. \tag{2.6}$$

In solving for the optimal hyperplane in a quadratic programming problem, it is important to reconsider the problem using Lagrange multipliers and Karush-Kuhn-Tucker conditions. The final formulation can be solved using programming methods with a set of support vectors $\mathbf{x}^t$, where $\mathbf{w}$ is expressed as the weighted sum of these selected training instances acting as the support vectors. Support vector machine, or SVM, is a term for calculating support vectors on average [5]. After training SVM, the hyperplane with the largest margin is reformulated as the decision boundary:

$$d(\mathbf{x}^T) = \sum_{t=1}^{l} \mathbf{g}^t \alpha^t \mathbf{x}^t \mathbf{x}^T + b_0$$

where $\mathbf{x}^T$ is a test data; α_i and b_0 are numeric parameters that were determined automatically by the optimization; and l is the number of support vectors. Sign of this equation determines the class of test data [39].

If our classes can not be linearly divided, there exist nonlinear separation problem. Noise and inherent nonlinearity are causes of non separable classes. If there exist noise, SVM try to find decision boundary with some misclassification error. There exist two misclassification cases, an instance correctly classified but in the margin area and lie in the wrong class. Equation (2.6) with penalty term:

$$min\ \frac{1}{2}||\mathbf{w}||^2 + C\sum_t \zeta^t\ subject\ to\ \mathbf{g}^t(\mathbf{w}^T\mathbf{x}^t + b) \geq 1 - \zeta^t, \zeta^t \geq 0 \tag{2.7}$$

C is regularization parameter between margin maximization and error minimization, must be tuned.

When there is inherent nonlinearity, Support Vector Machines (SVM) transform the samples into a higher dimensional space, denoted as $x_i \rightarrow \phi(x_i)$, in order to create a linear separation problem in this new dimension. This approach addresses the same issue observed with noise, but the constraints are defined in the new space.

$$\mathbf{g}^t(\mathbf{w}^T\phi(\mathbf{x}^t) + b) \geq 1 - \zeta^t$$

Equation (2.7) can be rewritten as a quadratic programming problem, transforming the decision boundary:

$$d(\mathbf{x}) = \sum_t \mathbf{g}^t \alpha^t K(\mathbf{x}^t, \mathbf{x})$$

Where $K(\mathbf{x}^t, \mathbf{x})$ is a kernel function that simplifies the decision-making process for nonlinear functions and can operate in the original feature space [5].

2.4.3.1 Kernel Functions

There are different types of kernels for classifications, known as similarity measures. When $\mathbf{x}^t$ and $\mathbf{x}$ are more similar, their value is higher [5].

Polynomials of degree q

When q = 1, it turns a linear kernel.

$$K(\mathbf{x}^t, \mathbf{x}) = (\mathbf{x}^T, \mathbf{x}^t + 1)^q$$

Radial Basis Function

It defines a spherical kernel.

$$K(\mathbf{x}^t, \mathbf{x}) = e^{[-\frac{||\mathbf{x}^t - \mathbf{x}||^2}{2s^2}]}$$

where $\mathbf{x}^t$ is center and s is radius, $\gamma = \frac{1}{s^2}$ must be tuned [69], $||\mathbf{x}^t - \mathbf{x}||$ is Euclidean distance between two feature vectors. The polynomial and sigmoid kernels aren't as adaptable as Radial Basis Function [69].

Sigmoidal functions

It is also known as Multi Layer Perception Kernel.

$$K(\mathbf{x}^t, \mathbf{x}) = tanh(\kappa \mathbf{x}^T \mathbf{x}^t - \delta)$$

where κ, δ are adjustable.

2.4.4 Decision Trees (DT)

Decision trees are a type of classification learning algorithm [50, 5, 80, 69]. We have data represented as **X**={$\mathbf{x}^t$,$\mathbf{g}^t$} for a binary classification problem. A decision tree (DT) consists of three parts: the node, branch, and leaf. The DT grows its structure recursively from the root node to the final node as a leaf by making decisions. A branch includes the main root node and all its subsequent nodes. Each internal node, denoted as m, implements the test function $f_m(\mathbf{x})$, which defines a discriminant in the input space. Since $x_j \in \mathbf{X}$ is numeric, the test function is:

$$f_m(\mathbf{x}) : x_j > h$$

where h is a threshold value. The decision node's binary split is represented by the left and right areas $L_m = \mathbf{x}|x_j > b$ and $R_m = \mathbf{x}|x_j \leq b$, respectively. This process continues by creating splits that are orthogonal to one another. In the feature space, the leaf node defines a hyperrectangle. With the provided data, decision trees can segment the training set into multiple subsets, each of which may include more than one class.

2.4.4.1 Splitting criteria

When creating a decision tree, there is an approach to identifying the most suitable splitting features: minimizing impurity measures or maximizing gain for each node. This involves assessing potential features and choosing the one that yields the greatest information gain or the least impurity. Decision trees are a crucial model in machine learning. The induction of decision trees seeks to produce the best discriminant function and ends with all instances having the same label, called pure. A node's purity or impurity is determined using impurity metrics. An estimate of the likelihood of label C_i provided that an instance has reached node m is also calculated.

$$\hat{P}(C_i|\mathbf{x}, m) = p^i_m = \frac{N^m_i}{N^m}$$

A node is termed pure if the quantity p^i_m is equal either to 0 or 1. The entropy measurement of impurity can indeed be expressed using the formula [78]:

$$I_m = -\sum_{i=1}^{K} p^i_m log_2 p^i_m$$

In two class scenarios, where $p^1 = p$ and $p^2 = 1 - p$, a function $\phi(p, 1-p) \geq 0$ is considered a measure of impurity if the following conditions are met [28]:

1. $\phi(\frac{1}{2}, \frac{1}{2}) \geq \phi(p, 1-p)$ for any $p \in [0, 1]$
2. $\phi(0, 1) = \phi(1, 0) = 0$
3. $\phi(p, 1-p)$ is increasing in p on [0,1/2] and decreasing in p on [1/2, 1]

Standard impurity functions for binary classification are as follows:

1. Entropy:
   $\phi(p, 1-p) = -plog_2p - (1-p)log_2(1-p)$
2. Gini Index [14]:
   $\phi(p, 1-p) = 2p(1-p)$
3. Misclassification error:
   $\phi(p, 1-p) = 1 - max(p, 1-p)$

These three measurements are not significantly different from one another [5]. However, the classification error for tree growing models is not as good as for the others [45].

DT must identify features for each stratum that maximize information gain denoted as IG for attribute A :

$$IG(\mathbf{X}, A) = \underset{v \in values(A)}{I(\mathbf{X})} - \sum (|\mathbf{X}_v|/|\mathbf{X}|) I(\mathbf{X}_v)$$

where $\mathbf{X}_v = \{x \in \mathbf{X} | A(x) = v\}$, I is any impurity function [54].
For binary classification,

$$IG(\mathbf{X}, A) = I(\mathbf{X}) - \frac{N_{left}}{N_p} I(\mathbf{X}_{left}) - \frac{N_{right}}{N_p} I(\mathbf{X}_{right})$$

DT orders the features by calculating the information gain.

2.4.4.2 Pruning

The decision tree (DT) has an issue with overfitting if it performs well on the training data but poorly on unobserved data. To address this problem, we can employ data preprocessing or pruning techniques. In data preprocessing, decision trees simplify sequential processes indirectly. In pruning, decision trees identify subtrees that cause overfitting and remove them [54].

There are two methods for pruning decision trees: pre-pruning and post-pruning. Pre-pruning involves stopping the decision tree (DT) before it is fully grown, while post-pruning involves pruning the DT after it has achieved zero training error [5]. DT performs post-pruning using a minimal cost-complexity algorithm. When pruning, it is important to aim for the optimal tree size. This is because exceeding a certain point in tree size, and limiting it too much, does not significantly reduce misclassification errors. The process for DT pruning starts from the bottom and goes to the top. The DT begins by constructing the largest tree for the given training data as T_{max}. For any subtree T, , the complexity, or the final number of nodes in the subtree, is denoted as $|\tilde{T}|$. Assuming $\alpha \geq 0$ is a real-valued complexity parameter, the cost-complexity measure is denoted as:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|$$

$R(T)$ represents the total misclassification error in the final nodes. The variable α is adjusted to penalize complexity. If α is large, T_{max} will also be large. There are finite subtrees, and in each iteration, the complexity reduces until pruning is complete. The cost complexity for a single node can be denoted as:

$$R_\alpha(t) = R(t) + \alpha$$

In general, the cost complexity of a single node is higher than the cost complexity of a branch T_t, but there are specific values of α where they become equal. This means that the subbranch is smaller than the subtree, which is desirable when adjusting the size of the tree and the critical value of α :.

$$\alpha < \frac{R(t) - R(T_t)}{|\tilde{T}| - 1}$$

By defining a function g_1 and $t \in \tilde{T}_1$:

$$g_1(t) = \begin{cases} \frac{R(t)-R(T_t)}{|\tilde{T}|-1} & t \notin \tilde{T}_1 \\ +\infty & \in \tilde{T}_1 \end{cases}$$

Minimal cost-complexity pruning prune the weakest link by looking for smallest value of $g_1(t)$. This algorithm terminates when $g_1(t)$ is greater than α [14].

To enhance Decision Tree (DT) performance, ensemble learning algorithms have been developed. Notably, Bagging and Boosting represent two distinct strategies within ensemble methodologies.

2.4.5 Random Forest (RF)

The bagging algorithm is the foundation of the random forest algorithm. If the training set **X** consists of N data points, denoted as **X**=$\{\mathbf{x}^t,\mathbf{g}^t\}$, let's consider a predictor φ with a sequence of learning sets represented as $\{\mathbf{X}_k\}$, each containing N independent samples from the same underlying distribution as **X**. A subset of the sample, including replacement, is taken by each predictor to form a bootstrap sample. This technique, known as bagging, aims to improve predictor performance. However, it has a limitation of only working with a sequence of predictors $\varphi(\mathbf{X}_k)$.

In classification problems, aggregation is done by taking B bootstrap samples (for each b=1,.., B) and using a majority vote method $\varphi_B(\mathbf{x})$ =majority vote $\varphi_B(\mathbf{x}, \mathbf{X}^{B})$. This process is known as "bootstrap aggregation" or "bagging", where predictions are collected in a bag before the final decision is reached through a majority vote [13]. Bagging is successful in improving accuracy if φ is unstable, meaning small changes in the input data (**X**) lead to large changes in the predictions. It enhances the accuracy and stability of the predictor while reducing its final variance. Classification and regression trees tend to be unstable and are thus suitable candidates for bagging [5, 13].

The random forest (RF) is an ensemble of modified decision trees that uses a random subset of input features in each split. This approach is a particular case of bagging. [5, 80, 15]. The motivation behind selecting randomly chosen features is to create de-correlated trees. By using sample training data and sampling on features, the trees grow to be more uncorrelated [69]. To calculate the selected p features from d features, the default process value is $\sqrt{d}$ [41]. For classification, the RF algorithm works as follows:

1. For b = 1 to B:
   - Draw a bootstrap sample N* of size N from the training data.
   - Produce a random forest tree to the bootstrapped data by repeating the following steps for each leaf of the tree, until the minimum node size is founded.
     i.Randomly select p variables from d variables.
     ii.Determine the optimal split among p, then divide the nodes into binary.
2. Final decision of the ensemble of trees.

For a testing data prediction, assume $\hat{C}_b$ is the prediction of bth tree, then $\hat{C}_{rf}^{B}$=majority vote $\{\hat{C}_b\}$ [41].

2.4.6 eXtreme Gradient Boosting (XGBoost)

Boosting involves selecting random samples without replacement, evaluating each learner's mistake, and then incorporating low-accuracy learner implementations one after the other. Boosting differs from bagging in the absence of sample replacement. Gradient Boosting is a modified version of boosting that uses the gradient of loss in its calculation [80]. In 2016, Tianqi Chen and Carlos Guestrin developed the eXtreme Gradient Boosting method based on boosting, which uses decision trees to create strong learners. XGBoost brings scalability and speed to the gradient boosting tree technique in addition to being a newer version of the algorithm. The sparsity-aware XGBoost algorithm also features a regularized objective function and out-of-core processing. To address overfitting concerns, XGBoost uses column subsampling and shrinkage [21].

2.4.6.1 Regularized Learning Objective

In the context of a dataset $D = (x_i, y_i)$, where $|D| = n$, $x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$ uses K-additive functions are employed to predict the y_i values. This is represented by the equation:

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

where $\mathcal{F} = f(x) = w_{q(x)}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ denotes the space of regression trees. The function q represents the structure of each tree, while T represents the number of leaves in the tree. Each f_k corresponds to an independent tree structure q and leaf weights w. Notably, each leaf of each regression tree carries a continuous score, denoted by w_i for the i-th leaf. The function q is used as decision rules for the classification an example and the final prediction is calculated by summing up w_i in the corresponding leaves. The learning of the model's functions entails the minimization of the regularized objective.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y_i}, y_i) + \sum_k \Omega(f_k) \tag{2.8}$$

where

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

penalizes the complexity of the model and helps overfitting, l is a differentiable convex loss function [21].

2.4.6.2 Gradient Tree Boosting

Equation (2.8) cannot be optimized using traditional methods in Euclidean space. The following is how the additive training strategy helps as an alternative: It adds a new function to each sample throughout each iteration [20].

$$\hat{y_i}^0 = 0$$
$$\hat{y_i}^1 = f_1(x_1) = \hat{y_i}^0 + f_1(x_i)$$
$$\hat{y_i}^2 = f_1(x_i) + f_2(x_i) = \hat{y_i}^1 + f_2(x_i)$$
$$.$$
$$.$$
$$.$$
$$\hat{y_i}^t = \sum_{k=1}^{K} f_k(x_i) = \hat{y_i}^{(t-1)} + f_t(x_i)$$

Ultimately, the regularized objective function is transformed into:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{t-1} + f_t(x_i)) + \Omega(f_t)$$

By utilizing the second order Taylor expansion of the loss function, the objective function can be expressed as:

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y_i}^{t-1}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y_i}^{t-1})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y_i}^{t-1})$ are first and second order gradient of the loss function, respectively. Constant terms are omitted, resulting in the simplified objective function at step t :

$$\tilde{\mathcal{L}}^t = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

Define $I_j = \{i | q(x_i) = j\}$ as the sample set of leaf j. This is a reformulation of the equation above:

$$\begin{aligned} \tilde{\mathcal{L}}^t &= \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \end{aligned} \tag{2.9}$$

Optimal weight w_j^* of a leaf j can be calculated as follows:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

By substituting w_j^* into the equation (2.9), the optimal value is obtained as:

$$\tilde{\mathcal{L}}^t = -\frac{1}{2}\sum_{j=1}^{t}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T$$

For the purpose of evaluating the quality, this can be utilized as the scoring function of q.

2.4.6.3 Shrinkage and Feature Subsampling

Shrinkage and column subsampling are two methods that help prevent overfitting. Shrinkage involves scaling new weights by a small number (η) between 0.01 and 1 after each step of boosting. This adjustment improves the tree's structure. Feature subsampling, used in Random Forest [14] also helps in avoiding overfitting in comparison with traditional row sub-sampling [20].

2.4.6.4 Split Finding Algorithms

One of the challenges in tree learning is figuring out the right split. The sets I_L and I_R represent the instances in the left and right nodes after the split, and $I = I_L \cup I_R$ represents the combined instances.

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

In practice, the above formula is commonly used to evaluate potential splits and calculate the loss reduction, as described in [20].

Basic Exact Greedy Algorithm

Basic Exact Greedy Algorithm is a split-finding algorithm that enumerates all possible splits on all the columns. It sorts the data and calculates h_i and g_i in $\mathcal{L}_{split}$ for continuous features. [20].

Approximation Algorithm

The Basic Exact Greedy Algorithm is not efficient when the data doesn't fit well into memory and distribution. In such cases, an Approximate Algorithm is needed. The algorithm first suggests candidate splitting points based on percentiles of feature distribution and then places continuous features into buckets split by these candidate points. After that, it aggregates the statistics and selects the best option based on the aggregated statistics. There are two types of the algorithm: the global variant and the local variant. In the global type, all the candidate splits are proposed during the initial phase of tree construction, and the same choices are used for split finding at all levels. The local variant, on the other hand, reconsiders the choices after each split [20].

Weighted Quantile Sketch

In the approximate method, it is essential to suggest potential split settings. Let multi-set $\mathcal{D} = \{(x_{1k}, h_1), (x_{2k}, h_2)(x_{nk}, h_n)\}$, represent the k-th feature values and second order gradient statistics of each training instance. We can define a rank function $r_k : \mathbb{R} \to [0, +\infty)$ as the ratio of instances whose feature value k is smaller than z:

$$r_k(z) = \frac{1}{\sum_{(x,h)\in\mathcal{D}}} \sum_{(x,h)\in\mathcal{D}_k, x<z} h$$

The goal is to find candidate split points $\{s_{k1}, s_{k2}, s_{kl}\}$, such that:

$$|r_k(s_{k,j}) - r_k(s_{k,j+1})| < \epsilon, s_{k1} = minx_{ik}, s_{kl} = maxx_{ik}$$

where ϵ is an approximation factor. Weighted squared loss with labels g_i/h_i and weights h_i:

$$\sum_{i=1}^{n} \frac{1}{2} h_i(f_i(x_i) - g_i/h_i)^2 + \Omega(f_t) + constant,$$

For large datasets, finding candidate splits that satisfy the criteria is a challenge. The weighted quantile sketch approach uses a data structure that enables merge and prune operations and is intended for weighted datasets [20].

Sparsity-aware Split Finding

The input variable is often sparse in real-world applications for several reasons, such as missing values, frequent zero entries, and one-hot encoding. Creating a sparsity-aware algorithm is crucial for identifying patterns in the data. One solution is to assign a default direction to each tree node. Each branch has two alternatives, and the best default direction is learned from the data.[20].

2.4.7 Light Gradient Boosting Machine (LightGBM)

After XGBoost was introduced to practitioners and scholars, Ke et al. [49] proposed another well-known gradient boosting model called the Light Gradient Boosting Machine in 2017. This model is a fast and innovative tree-based model. When implementing the decision tree algorithm, an issue arises in finding possible splits and evaluating them for information gain. As an alternative solution, LightGBM adds two new attributes to the gradient boosting algorithm: Gradient-based One-Side Sampling and Exclusive Feature Bundling.

2.4.7.1 Gradient-based One-Side Sampling

In the GOSS (Gradient-based One-Side Sampling) sampling strategy, instances are selected based on their gradients. More significant gradients are selected, while small gradients are randomly dropped. This method has been proven to be more effective than uniform sampling. The gradient-boosting decision tree mechanism uses these gradients to weight data instances. By identifying and discarding instances with small gradients, which indicate minor errors, the approach can unintentionally affect the distribution. To address this issue, GOSS offers a solution. Consider a set of n i.i.d. training instances $(x_1, ..., x_n)$. In each iteration of gradient boosting, the residual errors of the loss function with respect to the model's output are denoted as $(g_1, ..., g_n)$. For gradient boosting decision trees, the variance gain of splitting feature j at pint d defined as $V_j(d)$. GOSS organizes the training instances in descending order based on the absolute values of their gradients and retains $a \times 100\%$ of the largest instances in a set called A. Then, GOSS randomly samples a subset B with a size of $b \times |A^c|$. Finally, GOSS splits by considering the estimated variance gain, $\tilde{V}_j(d)$, over the subset $A \cup B$.

$$\tilde{V}_j(d) = \frac{1}{n} \frac{(\sum_{x_i \in A_l} g_i + \frac{1-a}{b} \sum_{x_i \in B_l} g_i)^2}{n_l^j(d)} + \frac{(\sum_{x_i \in B_r} g_i + \frac{1-a}{b} \sum_{x_i \in B_r} g_i)^2}{n_r^j(d)}$$

where $A_l = \{x_i \in A : x_{ij} \leq d\}$, $A_r = \{x_i \in A : x_{ij} > d\}$, $B_l = \{x_i \in B : x_{ij} \leq d\}$, $B_r = \{x_i \in B : x_{ij} > d\}$ and $\frac{1-a}{b}$ used for normalize the sum of gradients over B. The GOSS estimation process helps decrease computation costs, and for training accuracy, GOSS is superior to random sampling.

2.4.7.2 Exclusive Feature Bundling (EFB)

In real-world large datasets, a common problem is data sparsity, where many features rarely have nonzero values at the same time. When certain features are mutually exclusive and never have nonzero values simultaneously, Ke et al. proposes bundling these features into a single feature called an exclusive feature bundle (EFB). The selection and construction of features for bundling are two important issues for EFB. EFB introduces an algorithm to approximately optimize bundling within polynomial time to choose features for bundling. This algorithm converts the optimal bundling problem into the graph coloring problem and, with the help of a greedy algorithm, creates bundles. This algorithm strategically assigns each feature to an existing bundle by minimizing conflict rates when capable of including non-mutually exclusive features, or initiates the creation of a new bundle when necessary. This approach results in a reduced number of bundles, significantly enhancing computational efficiency. For constructing the bundle, the methodology involves adding offsets to the original values of the features, allowing the original features to be visible within the bundle.

2.4.8 Linear Regression

Linear regression is a statistical technique used to model the relationship between features and a dependent variable. For a feature vector denoted as $\mathbf{x}^T = (x_1, ..., x_N)$, the regression model:

$$f(x) = \beta_0 + \sum_{i=1}^{N} x_i \beta_i$$

The model includes unknown parameters represented by 'β_i' and an intercept represented by 'β_0'. When a linear regression model is trained, it learns to fit a linear function to the data. The vertical distance between the training data and the estimated regression model is known as the residuals, which indicate the degree of prediction error [80]. The model aims to find the best fit, so it needs to optimize its parameters using the well-known least squares method. With the best values of parameters, it aims to minimize error. For a training set $\mathbf{X} = \{\mathbf{x}^t, y^t\}_1^N$, the residual sum of squares for coefficients $\beta = (\beta_0, ..., \beta_N)^T$, denoted as [41]:

$$Residual\ sum\ of\ squares(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

2.4.9 Long Short Term Memory (LSTM)

In 1943, Warren McCulloch and Walter Pitts [64] published the calculus of a brain cell, known as the McCulloch-Pitts neuron. Neurons are connected with synapses, if a neuron surpasses its threshold it gives impulse. In 1957, Rosenblatt [85] presented the perceptron learning rule, which utilizes the McCulloch-Pitts neuron model to determine the weight of each input. This rule imitates the firing of neurons. It starts by initializing weights and bias, then updates the weight and bias for each sample [80]. In 1986, D.E. Rumelhart, G.E. Hinton, and R.J. Williams introduced the backpropagation algorithm, which revolutionized neural network training [87].

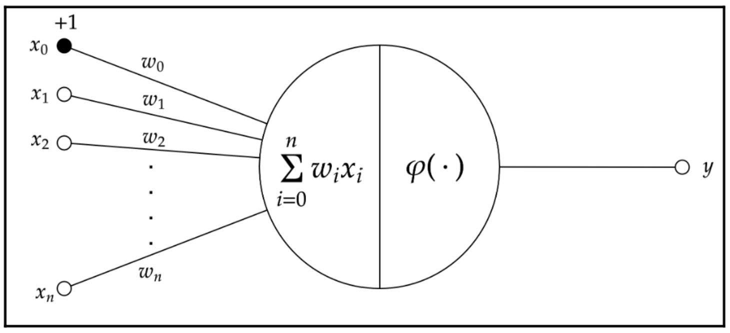


Figure 2.16: An artificial neuron[95]

Each neuron has following equation for n input:

$$y = \varphi(\sum_{i=1}^{n} w_i x_i - b)$$

where φ is an activation function, b is bias. Common activation functions used in deep learning include ReLU, piecewise linear, and Gaussian. During the training process, backpropagation and the Gradient Descent Algorithm are used to minimize the difference between predicted and actual outputs by using the gradient of the loss function. This leads to significant improvements in accuracy and performance. The choice of an appropriate loss function depends on the specific characteristics of the problem. For regression problems, it is typical to use metrics such as Mean Squared Error (MSE) to measure the difference between predicted and actual values:

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2$$

The weights were updated using backpropagation through time, which followed the formula for iteration:

$$w(k+1) = w(k) - \eta(\frac{\partial MSE}{\partial w})$$

where η is learning rate. In machine learning, an epoch refers to a single complete pass through the entire training dataset. During an epoch, the algorithm evaluates each instance in the dataset and may adjust its parameters to minimize errors in predictions. This is important for the learning process and helps improve the model's accuracy over time. Additionally, in batch learning, parameter updates are not made instantaneously with each data point, but are accumulated and applied collectively at the end of processing the entire batch [5].

A single-layer perceptron has a limitation in capturing nonlinear behaviors due to its linear operational framework. This limitation is addressed by feedforward networks with one or more hidden layers, which improve the network's ability to model nonlinear relationships effectively. However, feedforward networks have a unidirectional operation that may not be optimal for time series data. In such cases, recurrent neural networks (RNNs) are effective, as they can capture sequential dependencies. RNNs maintain a hidden state, serving as a memory that is updated at each time step and influenced by both current and previous input.

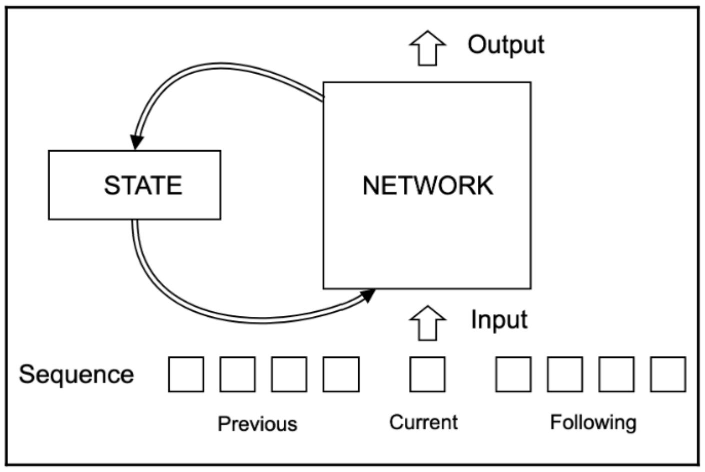
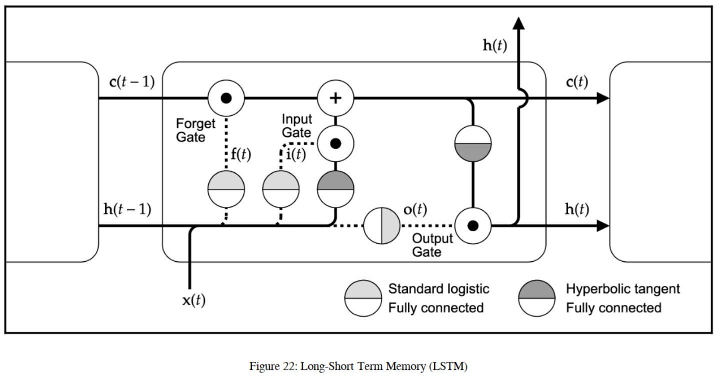


Figure 2.17: RNNs structure[95]

In finance, alternative models are necessary due to the challenges faced by RNNs in handling long-term dependencies. RNNs encounter two main issues during their training: the exploding and vanishing gradient problems. In 1997, Sepp Hochreiter and Juergen Schmidhuber proposed a solution to these problems with Long Short-Term Memory (LSTM) [42]. LSTM has had a significant impact on sequence modeling. Below is a visualization of the LSTM structure. It features a cell that is updated with the previous time step c_{t-1}, hidden units h_{t-1}, and the current step data x_t.



h(t)
c(t − 1)
c(t)
Forget Gate
f(t)
Input Gate
i(t)
h(t − 1)
h(t)
o(t)
Output Gate
x(t)
Standard logistic Fully connected
Hyperbolic tangent Fully connected
Figure 22: Long-Short Term Memory (LSTM)

Figure 2.18: LSTM[95]

LSTM (Long Short-Term Memory) has three gates: the forget gate (f_t), the input gate (i_t), and the output gate (o_t). These gates control the flow of information within the cell at time t by performing matrix multiplication of their inputs and using an activation function, typically the standard sigmoid function denoted as σ, which provides output values between 0 and 1.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Update the cell state by applying the hyperbolic tangent function to the input and candidate values, providing output between -1 and 1:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Cell state and hidden units at time t:

$$c_t = (f_t \odot c_{t-1}) \oplus (i_t \odot \tilde{c}_t)$$
$$h_t = o_t \odot \tanh(c_t)$$

where $\odot$ element-wise product and $\oplus$ element-wise summation [80, 95].

2.5 Evaluation Criteria of Classification Models

When it comes to predicting trends, there are several common strategies, but there isn't one specific method for testing the accuracy of the models' classifications. This study employs binary classification to determine the trend in the S&P 500: a value of 1 indicates that the S&P 500 is rising, while a value of 0 indicates that it remains the same or is decreasing. Test data is utilized to assess the model's performance. When dealing with binary classification, where class 1 is considered the positive class and class 0 is the negative class, there are four key terms used to calculate various metrics:

1. True Positive(TP): This refers to model predictions that are positive and are indeed true when compared to the actual values.

2. False Positive(FP): This refers to model predictions that are positive but are actually false when compared to the actual values.

3. True Negative(TN): This refers to model predictions that are negative and are indeed true when compared to the actual values.

4. False Negative(FN): This refers to model predictions that are negative but are actually false when compared to the actual values.

These terms are used in confusion matrix.

Table 2.1: Confusion matrix

		Predicted Class		
		1	0	Total
Actual Class	1	TP	FN	$TP + FN$
	0	FP	TN	$FP + TN$
	Total	$TP + FP$	$FN + TN$	N

This matrix illustrates the different forecast errors generated by the classification models and the performance evaluation table used for assessing the forecasts. The error is calculated as the sum of all false predictions divided by the total number of predictions.

$$Error = \frac{FP + FN}{FP + FN + TP + TN}$$

Accuracy is the ratio of correctly classified examples to the total number of classified examples [80].

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN} = 1 - Error$$

2.6 Evaluation Criteria of Regression Models

According to price prediction studies, there are common strategies, but no specific way to test prediction accuracy. Root mean squared error (RMSE) is used to compare regression model errors:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}$$

where $error = y_t - \hat{y}_t$, y_t is original data value and $\hat{y}_t$ is approximation.

2.7 Performance Metrics of Portfolios

The following metrics are used to evaluate and compare the performance of portfolio optimization weights and their corresponding returns in an out-of-sample setting. For simplicity, we assumed a risk-free rate of return of 0%. The metrics are used to compare single-period portfolios:

The out-of-sample weekly return at time $t+1$:

$$R_{t+1} = \sum_{i=1}^{I} \omega_{it} r_{i,t+1}$$

Sharpe ratio for a single period portfolio:

$$SR = \frac{R_p}{\sigma_p}$$

R_p is the average return of the portfolio over the period. σ_p is the standard deviation of the portfolio's returns over the period. Multi-period portfolios comparison metrics:

The out-of-sample average weekly return:

$$R_w = \frac{1}{T-\tau} \sum_{t=\tau}^{T-1} \sum_{i=1}^{I} \omega_{it} r_{i,t+1}$$

The out-of-sample Standard deviation:

$$\sigma_{\text{out}} = \sqrt{\frac{1}{T-\tau-1} \sum_{t=\tau}^{T-1} (R_{t+1} - R_w)^2}$$

The out-of-sample Sharpe ratio:

$$SR = \frac{R_w}{\sigma_{\text{out}}}$$

The out-of-sample cumulative return:

$$R_c = \prod_{t=\tau}^{T-1} (1 + R_{t+1}) - 1$$

The out-of-sample annualized return:

$$R_{anu} = (R_c + 1)^{\frac{1}{(T-\tau-1)/52}} - 1$$

2.8 Explanation Methods

In this work, the terms "interpreting" and "explaining" AI are used interchangeably. This section shares the scopes of interpretability methods and several preferred techniques

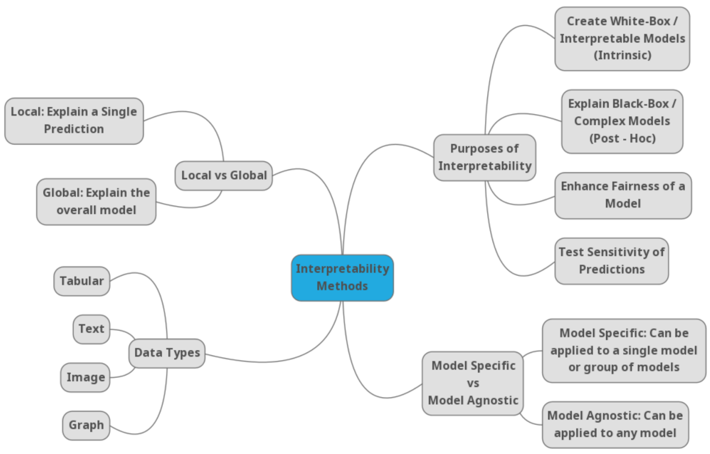

Create White-Box / Interpretable Models (Intrinsic)
Local: Explain a Single Prediction
Explain Black-Box / Complex Models (Post - Hoc)
Purposes of Interpretability
Local vs Global
Global: Explain the overall model
Enhance Fairness of a Model
Test Sensitivity of Predictions
Interpretability Methods
Tabular
Text
Data Types
Image
Graph
Model Specific: Can be applied to a single model or group of models
Model Specific vs Model Agnostic
Model Agnostic: Can be applied to any model

Figure 2.19: The classification of interpretability techniques[56]

In Figure 2.19, the classification map of interpretability methods is shown. The next phase will provide a brief overview of classifications:

- Local vs Global

  If a method explains a single instance, it is referred to as local; if it explains the entire model, it is referred to as global.

- Data Types

  There exist methods for text, image, graph, and tabular data types.

- Model Specific vs Model Agnostic

  If a method describes a specific algorithm, it is considered model-specific; if it applies to any algorithm, it is known as model-agnostic.

- Purposes of Interpretability

  Four categories are identified: methods for creating white-box models ,methods for explaining complex black-box models, methods that promote fairness and restrict the existence of discrimination, and methods for analysing the sensitivity of model predictions. Post-hoc approaches frequently focus on interpretability techniques linked to deep learning, and they have two subfields: one for deep learning research and the other for other black-box models [56].

LIME and SHAP are both local, model-agnostic, and additive feature attribution techniques. They work by using an explanation model represented by g, which sums up the impacts of the features. The explanation model is a linear function of binary variables.

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i^{'}$$

where $z^{'} \in \{0,1\}^M$, M is the number of simplified input features and $\phi_i \in \mathbb{R}$. Local models work like this: they aim to explain $f(x)$, which is the original classifier based on a single input x. The explanation model uses simplified inputs x' by mapping the original inputs through a function $h_x(x')$, denoted as $x = h_x(x')$. Ultimately, the local methods aim to ensure that $g(z')$ is approximately equal to $f(h_x(z'))$ when z' is approximately equal to x' [58]. Explanations should be clear and provide a qualitative understanding of the relationship between input variables and the response [81].

2.8.1 LIME

In 2016, LIME (Local Interpretable Model-agnostic Explanations) introduced by Marco Ribeiro, Sameer Singh and Carlos Guestrin [81], is an interpretability method that approximates any classifier. LIME selects an explanation model, represented as g, from a set of potentially interpretable models, denoted as G that can be visually presented to a decision-maker. The domain of g is $\{0,1\}^{d'}$, g where g operates based on the absence or presence of interpretable components. Since the interpretability of the explanation model is a concern for LIME implementation, complexity can be measured by $\Omega(g)$. In classification, a model $f : \mathbb{R}^d \rightarrow \mathbb{R}$, $f(x)$ is described where $f(x)$ represents the probability that x belongs to a certain class. The proximity measure π_x defines neighborhood of x. $\mathcal{L}$ measures how g is in approximating f in the locality defined by π_x. Interpretability and local fidelity are two desired features for an explanation. In order to ensure both features, we need to minimize $\mathcal{L}(f, g, \pi_x)$ while keeping $\Omega(g)$ low enough to be interpretable by humans. The explanation provided by LIME at a local point x can be expressed using the following formula:

$$explanation(x) = \underset{g \in G}{argmin}\ \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.10}$$

For problems requiring an explanation, this formula can be used with different values of G, $\mathcal{L}$, and Ω [81].

2.8.2 SHAP

The Shapley value was introduced by Lloyd Shapley in 1951 [90]. A Shapley value represents a player's contribution to the game. The "value" function, denoted as $value : 2^F \to \mathbb{R}$ is a characteristic function that maps subsets of players to nonnegative real numbers. In the context of machine learning, "games" refer to predictions, and "players" refer to features. The Shapley value is defined as the value function of players in subset of features denoted as S.

$$\phi_j(value) = \sum_{S \subseteq \{1,...,F\}\{j\}} \frac{|S|!(F - |S| - 1)!}{F!}(value(S \cup j) - value(S))$$

where F is the number of features, $S \subseteq \{1, ..., F\}\{j\}$ is all subsets exclude jth feature, $\frac{|S|!(F-|S|-1)!}{F!}$ is weight for each subset, $(value(S \cup j) - value(S))$ is marginal contribution of feature j to the subset S.

Let x feature values of an instance, $value(S)$ is prediction for feature values can be calculated as:

$$value_x(S) = \int \hat{f}(x_1, ..., x_p) d\mathbb{P}_{x \notin S} - E_X(\hat{f}(X))$$

where:

- $\hat{f}$ is the prediction function.
- $d\mathbb{P}_{x \notin S}$ represents the integration over the distribution of the features that are not included in the subset S.
- $\mathbb{E}_X(\hat{f}(X))$ is the expected prediction over the entire feature space, serving as a baseline or reference value.

Shapley values are the only additive attribution techniques that follow four axioms [37, 67]:

1. **Efficiency**

   The total payment for the game can be divided based on its features:

$$\sum_{j=1}^{N} \phi_j = value(F)$$

2. **Symmetry**

   If two features have equal importance in the game, their Shapley values must also be equal.

$$[(\forall S \backslash i, j) value(S \cup i) = value(S \cup j)] \implies \phi_i = \phi_j$$

3. **Dummy**

   The Shapley value of feature j is equal to zero if it contributes nothing to the coalition it is included in.

$$[(\forall S) value(S \cup j) = value(\{j\})] \implies \phi_j = 0$$

4. **Additivity**

   For two different characteristic functions, $value$ and $value^+$, the Shapley values of feature j preserve the addition of the games.

$$\phi_j(value + value^+) = \phi_j(value) + \phi_j(value^+)$$

In order to calculate the Shapley value, it's important to compute all potential combinations of features. As the number of features grows, this process becomes more difficult. To address this, Strumbelj et al. [92] suggest using Monte Carlo sampling for approximation, this approximation works as follows:

A specific instance x and a randomly selected instance from the dataset are used to estimate the Shapley value of the jth feature. In a process consisting of M iterations, a random order of features and a random instance z are generated. Two new instances are produced: x_{+j} after the jth feature takes on values from z, and x_{-j} after the $j-1$th feature takes on values from z. These new instances are then input into the model, and the difference between their outputs is calculated:

$$\phi_j^m = \hat{f}(x_{+j}^m) - \hat{f}(x_{-j}^m)$$

The Shapley value is calculated as average:

$$\phi_j(x) = \frac{1}{M}\sum_{m=1}^{M} \phi_j^m$$

Shapley Additive Explanations (SHAP) is an interpretability method that is not tied to any specific model. It is based on Shapley values, which were introduced by Lundberg and Lee in 2017 [58]. SHAP offers a new approach to Shapley values known as KernelSHAP, which is a kernel-based estimation method. SHAP also provides a global interpretation by aggregating Shapley values. It's important to note that Shapley value explanations are represented as an additive feature attribution method, which can be viewed as a linear model [67]. SHAP adheres to the axioms of Shapley values and adds three properties: local accuracy, missingness, and consistency. When estimating the original model f for an input x, consider the following axioms:

1. **Local accuracy**

   $$f(x) = g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j x_j^{'}$$

   The explanation model $g(x)$ matches the original model when $x = h_x(x)$, where $\phi_0 = f(h_x(0))$ represents the model output with all simplified inputs missing. By defining $\phi_0 = E_X(f(x))$ and setting all $x_j^{'}$ to 1, this is the same as Shapley efficiency axiom [67].

2. **Missingness**

   $$x_j^{'} = 0 \implies \phi_j = 0$$

   If missing features exist, they get a 0 Shapley value.

3. **Consistency**

   Let $f_x(z^{'}) = f(h_x(z^{'}))$ and $z_j^{'}$ indicates that $z_j^{'} = 0$. For any two models f and f', if

   $$f_x^{'}(z^{'}) - f_x^{'}(z_j^{'}) \geq f_x(z^{'}) - f_x(z_j^{'})$$

for all inputs $z' \in \{0,1\}^M$, then:

$$\phi_j(f', x) \geq \phi_j(f, x)$$

SHAP values are a unique additive feature importance method that possess the three properties mentioned above and use conditional expectations to define simplified inputs. They are solutions to the equation of Shapley value under the assumption:

$$f_x(z') = f(h_x(z')) = E[f(z)|z_S]$$

where S is set of non-zero indexes in z'.

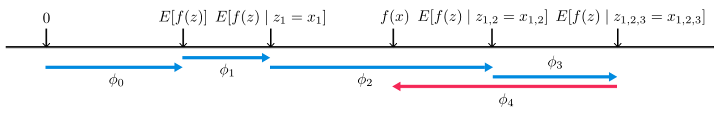


Figure 2.20: SHAP values[58]

In Figure 2.20, SHAP values depicts with a single ordering with the anticipated value of model updates taking feature values into account. Assuming the model is non-linear, or its features are not independent, SHAP values average ϕ_j for feature importance over all permutations. With various plotting techniques, SHAP values can be applied to local and global explanation analyses. The precise computation of SHAP values remains a complex challenge, as highlighted by Lundberg and Lee. To address this, they have proposed model-agnostic and model-specific approximation methodologies, with Kernel SHAP emerging as a notable example of the model-agnostic techniques. In these methods, feature independence and model linearity are optional. For further details on this subject, one is encouraged to consult the paper authored by Lundberg and Lee [58].

Kernel SHAP (Linear LIME + Shapley values)

Linear LIME uses a linear model as an explanation model; Shapley values are the solution of the LIME objective function and satisfy local accuracy, missingness and consistency properties. LIME made a heuristic choice of the loss function, weighting kernel, and a regularization term, but this approach is inappropriate for the Shapley values. Lundberg et al. propose the Shapley kernel for regression-based Shapley weighting:

$$\Omega(g) = 0$$
$$\pi_{x'}(z') = \frac{(M-1)}{(M choose|z'|)|z'|(M-|z'|)}$$
$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_{x'}(z')$$

where Z is the training data, $|z'|$ is the number of non-zero elements in z'. The estimated coefficients of the model, the ϕ_j's, are the model agnostic Shapley values [58, 67].

CHAPTER 3

IMPLEMENTATION AND EVALUATION

This chapter involved the implementation and results. Python 3.10.13 [97] was used to develop models for time series predictions and XAI methods. Jupyternotebook [51] downloaded, prepared and visualized the dataset for XAI explanations. Using the Scikit-Learn package [74], machine learning models were put into practice. SHAP [58] and LIME [81] were utilized to describe how machine learning models work. RStudio [76] and Quantmod [88] were powerful tools utilized for visualization of the technical analysis and S&P 500. The development of portfolio optimization models utilized a modeling language for mathematical programming called as AMPL [77], with mean-variance calculations executed through Gurobi solver and other models through Highs solver. These tools provided effective means of evaluating stock market performance, recognizing patterns, and guiding investment decisions. Extensive testing was performed on a robust 64-bit desktop, powered by an AMD Ryzen 9 6900HS processor running at 3.30 GHz, and equipped with 16GB of RAM.

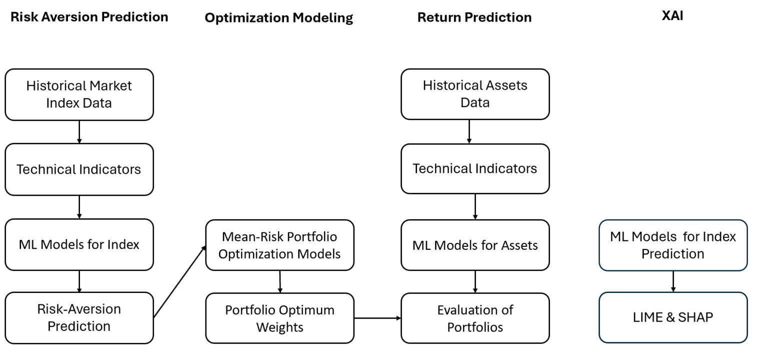


Figure 3.1: Workflow

Figure 3.1 illustrated a comprehensive workflow which was crucial for understanding the processes and procedures outlined in the study. The detailed depiction in this figure provided a sequential visualization, allowing for a clearer comprehension of the steps involved and their interconnections.

3.1 Data and Risk-aversion Prediction

The Standard & Poor's 500 (S&P 500), a stock market index that comprised weekly open, high, low, close prices, and trading volumes, was downloaded from Yahoo Finance. The prediction of upward movements of the S&P 500 was utilized as a risk aversion coefficient. Stocks from various industries were selected to explore the effects of diversification on portfolios. The adjusted close prices of 24 risky assets and the 13 Week Treasury Bill, serving as a risk-free asset, were employed for portfolio construction. The data spanned from January 2008 to January 2018, including 503 weekly returns.

Table 3.1: The description of the OHLCV

Columns	Explanation
Date	Time
Open	Weekly Open Price of S&P 500
High	Weekly Highest Price of S&P 500
Low	Weekly Lowest Price of S&P 500
Close	Weekly Close Price of S&P 500
Volume	Weekly Transaction Amount of S&P 500

In Table 3.1, the OHLCV (open, high, low, close, volume) of the S&P 500 were documented, along with their names and notations.

3.1.1 Descriptive Statistics of S&P 500

In Table 3.2, descriptive statistics were provided for the S&P 500. The higher mean compared to the median suggested that the price was right-skewed. A skewness value near zero indicated a right-tail support, but the kurtosis value far from three showed that we could not assume a normal distribution.

Table 3.2: The Descriptive Statistics of S&P 500

Statistic	Value	Statistic	Value
Observation	522	1st Qu.	1224.5
Mean	1613.7	3rd Qu.	2050.7
Median	1461.5	Max	2690.2
Std. Dev.	484.3	Skewness	0.243
Min	676.5	Kurtosis	-1.072

In Table 3.2, the skewness value indicated an asymmetric distribution of returns, and the kurtosis value suggested that the distribution had heavier tails than a normal distribution. In Figure 3.2, the return distribution of the S&P 500 was visualized with a histogram, and a theoretical normal distribution curve was also plotted on the same histogram for comparison.

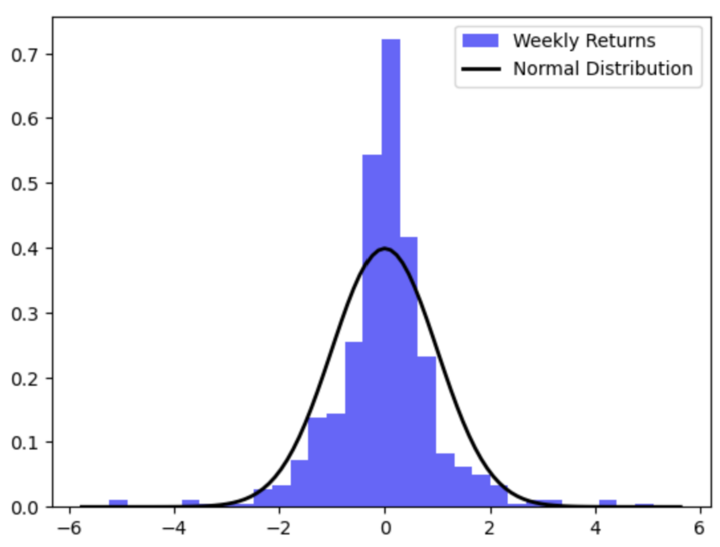


Figure 3.2: S&P 500 Weekly Returns

It was anticipated that the data had an approximately normal distribution. The normal distribution was considered appropriate for the actual return distribution, and the mean-variance model relied on the assumption of a normal distribution. For the purpose of that study, it was assumed that the S&P 500 and assets followed a normal distribution. The following were the weekly return distributions of selected stocks:

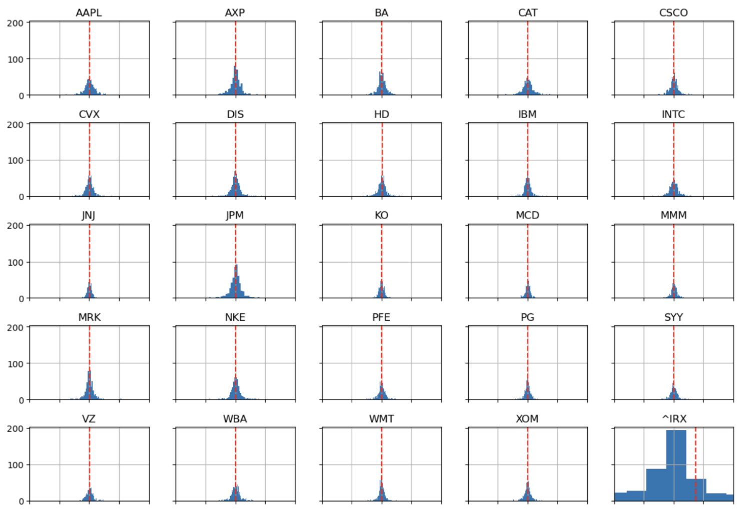


Figure 3.3: Weekly Returns of stocks

In Figure 3.3, we had also included the mean values for each asset.

3.1.2 Exploratory Data Analysis of S&P 500

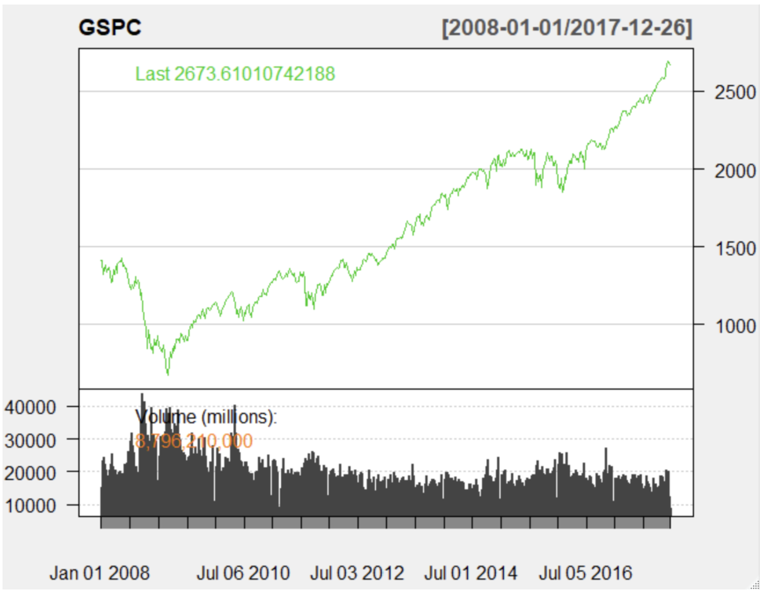


Figure 3.4: S&P 500 Weekly Prices

In Figure 3.4, a time series plot of the S&P 500 from January 1, 2008, to December 26, 2017, is shown. Several factors influenced the S&P 500, including fluctuations in global economic growth and the level of economic uncertainty following the financial crisis. The S&P 500 reached its lowest point on March 3, 2009, and, according to the timeline, its price peaked on December 12, 2017. The figure below depicted the selected assets and their performance over the same period, forming a portfolio.

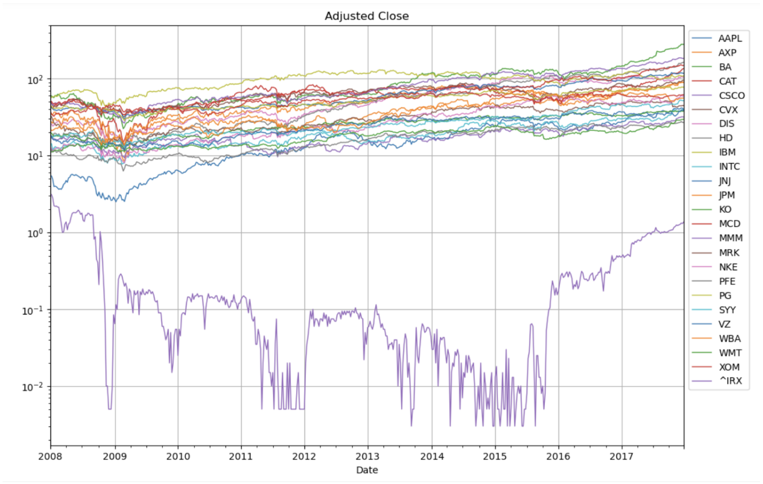


Figure 3.5: Weekly Prices of Assets

In Figure 3.5, time series plots of assets are shown from January 1, 2008, to December 26, 2017. The range of stock return volatility in our portfolio varied from 0.02 for

Johnson & Johnson (JNJ) to 0.057 for JPMorgan Chase & Co. (JPM), as shown in the figure referenced earlier. It's worth noting that out of the 25 assets analyzed, Treasury bills had the highest volatility at 0.93, indicating significantly greater fluctuation in their returns compared to other assets in the study.

3.1.3 Technical Indicators

Two Python libraries were used for technical analysis: EMV, BB, and VWAP were generated with [73] library, while the rest were generated using [10].

Table 3.3: Technical Indicators

Indicators Name	**Parameter Value**
SMA	Number of periods to average over: 3,5,10
WMA	Number of periods to average over: 3,5,10
EMA	Number of periods to average over: 3,5,10
ADX	Number of periods to use for DX calculation: 3, 5, 10
RSI	Number of periods for moving averages: 3, 5, 10
CCI	Number of periods for moving average: 20
EMV	Number of periods for moving average: 3, 5, 10
BB	Number of periods for moving average: 3, 5, 10
SAR	Acceleration factor: 0.02 & Maximum acceleration factor: 0.2
MFI	Number of periods to use: 3, 5, 10
VWAP	Number of periods to average over: 3,5,10

In Table 3.3, each indicator was characterized by distinct parameters, underscoring their utility and adaptability in various research and practical finance applications.

3.1.4 Preprocessing pipeline

$Trend$ was determined using the following procedure: Let's consider P_t as the weekly closing price and the target variable as $Trend$.

$$Trend = \begin{cases} 1 & \text{if} \quad P_t \geq P_{t-1} \\ 0 & \text{if} \quad P_t < P_{t-1} \end{cases}$$

If the value of $Trend$ was 1, it meant the market index would rise or stay the same, and if it was 0, the market index would decrease from P_{t-1} to P_t. Firstly, the data columns were checked for any missing values. Missing values were present after including technical analysis variables as variables, so missing values in the first 19 rows were deleted. Issues related to class imbalance in classification needed to be examined. Class imbalance occurs when one class has a majority, and models tend to favor that class [79]. There was no class imbalance in the dataset.

We employed same ten-year timeframe for both single-period and multi-period portfolio construction methodologies. Specifically, for the construction of single-period portfolios, we utilized a decade's worth of historical data for training purposes, with the final week of this dataset designated as the testing period. Conversely, in the approach towards multi-period portfolios, we implemented a rolling horizon methodology. This entailed generating distinct portfolios at biennial intervals throughout the ten-year span, with each two-year segment's ensuing week serving as the test period for its respective portfolio. This methodology allowed for a nuanced examination of portfolio performance over time, catering to both short-term and long-term investment strategies.

3.1.5 Data Transformation Methods

Some models made the assumption that the feature set could be scaled; hence, the statistical foundation of scaling strategies was explained.

1-Standardization

Some machine learning techniques required standardization, commonly referred to as z-score standardization, as demonstrated by the formula below.

$$x_{standardized} = \frac{x_{feature} - \mu}{\sigma}$$

μ and σ represented the mean and standard deviation of the corresponding column. Standardization was used for Logistic Regression, KNN, and SVM. There was no need for any feature transformation for tree-based models [41].

2-Normalization

Min-max scaling was a normalization technique that adjusted the features to fit within a specific range, typically between 0 and 1. Scaling was achieved by subtracting the minimum value of the feature and then dividing by the range [80].

$$x_{scaled} = \frac{x_{feature} - x_{min}}{x_{max} - x_{min}}$$

This method was intended for use in LSTM prediction only.

3.1.6 Model Optimization Method

There were two distinct types of parameters in machine learning: primary model variables that were learned from training data (such as the weights in LR), and tuning parameters that needed to be adjusted or built manually [4]. Grid search was a known and simple method for hyperparameter optimization. It methodically searched every possible combination of tuning parameters because the model did not optimize the tuning parameters [15]. It was recommended to use grid search with k-fold cross-validation for the exploration of the hyperparameter space [74].

K-Fold Cross-Validation (KFCV)

When evaluating and developing a machine learning model, there were more practical options than simply splitting the data into train and test samples. Using K-Fold Cross-Validation (KFCV) was an optimal choice. In KFCV, the training samples were randomly divided into k equal portions called folds. KFCV then iterated through each fold, using it as the test data while using the remaining components as training data. Each iteration generated an error, and the final error was determined by averaging the errors after the kth iteration [69].

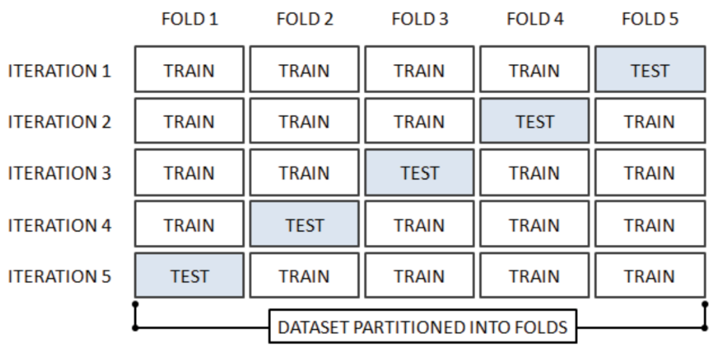

FOLD 1
FOLD 2
FOLD 3
FOLD 4
FOLD 5
ITERATION 1
TRAIN TRAIN TRAIN TRAIN TEST
ITERATION 2
TRAIN TRAIN TRAIN TEST TRAIN
ITERATION 3
TRAIN TRAIN TEST TRAIN TRAIN
ITERATION 4
TRAIN TEST TRAIN TRAIN TRAIN
ITERATION 5
TEST TRAIN TRAIN TRAIN TRAIN
DATASET PARTITIONED INTO FOLDS

Figure 3.6: KFCV Process[69]

In Figure 3.6, the KFCV process with 5-fold visualization was displayed. In order to properly handle time series data, it was important to use a time series cross-validator like Time Series Split. This method ensured that the data was not randomly divided because the order of the data points was crucial in time series analysis. Time Series Split worked by returning the first k folds as the training set and the (k+1)th fold as the test set in the kth split [74]. In other cases, the value of k was chosen as 10, with the exception of LSTM.

3.1.7 Risk-aversion estimation

The development of our investment portfolio strategy incorporated an intricate assessment of risk tolerance, culminating in the deliberate construction of diverse investment portfolios. Central to this strategic formulation was the derivation of a risk aversion coefficient, a pivotal metric aimed at gauging the propensity of investors to avoid risk. This coefficient was extracted through the application of six sophisticated machine learning classifiers, each selected for its proven efficacy in predicting the S&P 500 index. The classifiers employed in this analysis comprised Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Decision Tree (DT), Random Forest (RF), and XGBoost, representing a comprehensive spectrum of algorithmic approaches tailored to navigate the complex dynamics of financial markets.

Table 3.4: The Prediction of Single Period Models

Models	**Risk-Aversion**	**Models**	**Risk-Aversion**
LR	0.43	SVC	0.57
KNN	0.42	DT	0.56
RF	0.54	XGBoost	0.51

In Table 3.4, all models made predictions about the likelihood of the market going up. The prediction generated by the Support Vector Classifier (SVC) exhibited a tendency towards boldness in its forecasts, in contrast, the predictions made by the K-Nearest Neighbors (KNN) algorithm demonstrated a more cautious approach. To assess the accuracy of the models, a decision threshold was set at a probability value of 0.5. If the predicted probability was 0.5 or higher, the S&P 500 was expected to go up. Conversely, if the predicted probability was below 0.5, the market was expected to go down.

3.2 Portfolio Optimization

We conducted thorough research on the composition and performance of portfolios consisting of 24 stocks from eight different sectors in the S&P 500 index, along with a risk-free asset. Our research analyzed the performance of portfolios over a ten-year period, using both mean-variance and mean-Conditional Value at Risk (CVaR) strategies for single-period portfolios. We maintained portfolio weights for one week and then analyzed out-of-sample returns. For Sharpe ratio comparison, we held optimized weights for one week and then compared the models over four weeks. Additionally, we made stock price predictions for the same timeframe to help make better investment decisions.

During a ten-year period, we regularly adjusted our mean-Mean Absolute Deviation (MAD) portfolios every two years using a strategic rebalancing approach. We used a rolling window method with a window size of 104 weekly returns. As we moved forward, we included new weekly data and dropped the oldest data point, using the data following each week for out-of-sample testing. This process resulted in 399 out-of-sample portfolios. This method allowed us to capture the dynamic nature of the stock market and evaluate our portfolio strategy under real-world conditions. By continuously updating the portfolio with new data, we created a simulation of real-world conditions, enabling us to assess the effectiveness of our strategy.

The primary goal was to clarify the effectiveness of both single-period and multi-period portfolios, thereby making a significant contribution to the body of portfolio management literature.

3.2.1 Classification Models for Single Period Portfolios

Data was divided into 502 weeks for training and 1 week for testing. A grid search was implemented for each machine learning classifier using time series ten-fold cross-validation. The model's performance was evaluated using the test data after training. The optimized parameter results are the same for mean-variance and mean-CVaR problems.

1-LR

To find the best settings for regularization, C, and solver parameters, we thoroughly analyzed a range of values. Our analysis showed that the LR classifier performed best with L1 regularization and a regularization parameter of 0.1. We used the Liblinear algorithm for the optimization process.

Table 3.5: Grid Search Results for Logistic Regression with Liblinear Solver

Regularization Type	C Value Range	Solver	Best Performing Setting
None	10^{-5} to 100	Liblinear	-
L1	10^{-5} to 100	Liblinear	L1, $C = 0.1$
L2	10^{-5} to 100	Liblinear	-
ElasticNet	10^{-5} to 100	Liblinear	-

2-KNN

We carefully chose the best options for our model to achieve high accuracy and performance. When we used the k-nearest neighbors algorithm, it was important to consider factors such as the number of neighbors, the weight given to all points in the neighborhood, the search algorithms used, and the power of the Minkowski metric.

Table 3.6: Grid Search Results for K-Nearest Neighbors (KNN)

Parameter	Value Range Considered	Best Performing Setting
Number of Neighbors	1 to 30	8
Weight Function	Uniform, Distance	Distance
Search Algorithms	Auto, Ball Tree, KD Tree, Brute	Auto
Minkowski Power Parameter	1 (Manhattan), 2 (Euclidean)	2 (Euclidean)

3-SVC

We had identified the optimal parameter values to improve the accuracy of prediction for the SVC.

Table 3.7: Grid Search Results for Support Vector Classifier (SVC) with RBF Kernel

Parameter	Value Range Considered	Best Performing Setting
Regularization Parameter	0.1, 1, 10, 100, 1000	0.1
Kernel Type	Radial Basis Function (RBF)	RBF
RBF Kernel Coefficient	1, 0.1, 0.01, 0.001, 0.0001	1

4-DT

In conducting a decision tree analysis, it was necessary to identify the optimal values for features, pruning, depth, and split parameters to achieve the most effective results. These values were selected to ensure the most effective decision tree analysis.

Table 3.8: Grid Search Results for Decision Tree Classifier

Parameter	Value Range Considered	Best Performing Setting
Number of Features for Best Split	$\sqrt{\text{features}}$, $\log_2(\text{features})$	$\sqrt{\text{features}}$
Complexity Parameter for Pruning	0.1, 0.01, 0.001	0.1
Maximum Depth of the Tree	5 to 9	5
Impurity Measure	Gini, Entropy	Gini

5-RF

The random forest (RF) model provided a variety of hyperparameters that could be fine-tuned to improve model performance. These hyperparameters included the maximum tree depth, the number of features to consider for the best split, the maximum number of leaf nodes, and the number of trees in the forest. These hyperparameters had been identified after careful analysis and were recommended for optimal results.

Table 3.9: Grid Search Results for Random Forest Model

Parameter	Value Range Considered	Best Performing Setting
Maximum Tree Depth	3, 6, 9	6
Number of Features for Best Split	$\sqrt{\text{features}}$, $\log_2(\text{features})$, None	$\log_2(\text{features})$
Maximum Number of Leaf Nodes	3, 6, 9	9
Number of Trees in the Forest	25, 50, 100, 150	25

6-XGBoost

The XGBoost model provided various hyperparameters that could be optimized to improve model performance.

Table 3.10: Grid Search Results for XGBoost Model

Parameter	Value Range Considered	Best Performing Setting
Subsample Ratio of Columns	0.6 to 0.9	0.7
Minimum Loss Reduction (γ)	0.0 to 0.4	0.0
Maximum Tree Depth	6 (default)	6
Minimum Sum of Instance Weight Needed in a Child	4, 5, 6	5
Subsample Ratio of Training Instances	0.1	0.1

LR and KNN models accurately predicted test data, while others did not. LR and KNN had 100% accuracy, whereas the other models had 0%. Hyperparameter tuning and risk aversion estimation were the same for single-period models.

3.2.2 Results of Portfolios

In this subsection, we evaluated the returns generated by eight different portfolios for each of the three different risk measures. These portfolios were named 1-LR, 2-KNN, 3-SVC, 4-DT, 5-RF, 6-XGBoost, 7-ρ=0.5, and 8-Equal weight portfolio. Our analysis showed that each portfolio had unique performance characteristics, with varying degrees of outperformance compared to the others.

Our findings contributed to the broader discussion on portfolio management strategies, providing insights into the comparative performance of different investment approaches under various analytical frameworks. This analysis emphasized the importance of selecting an appropriate framework for portfolio evaluation, as the chosen methodology significantly impacted the perceived effectiveness and risk profile of investment portfolios.

3.2.2.1 Mean-Variance Portfolios

Among risk-aversion adjusted mean-variance portfolios with returns ranged from -0.5286% to 1.70%.

Table 3.11: Out of one week returns

Models	**Returns**
LR	-0.003863432361643219
KNN	-0.005285664424739152 5
SVC	0.0170301088270511
DT	0.015173588933214636
RF	0.011465664916599339
XGBoost	0.007112445773191464
$\rho = 0.5$	0.004902702745847146
Equal weights	0.00006342

In Table 3.11, the risk aversion adjusted mean-variance portfolio analysis was found to provide returns for investors who were willing to take risks, while it resulted in losses for those investors who were risk-averse. After analyzing different investment strategies, it was found that the support vector classification (SVC) model had the highest return, while the K-nearest neighbors (KNN) model had the largest loss. As a result, it became important to investigate how the SVC model is implemented, specifically focusing on its risk-aversion parameter. There was a need to further examine how this parameter affected the process of optimizing the mean-variance portfolio. The results from last week's assessment were summarized in the table below, showing the optimal distribution of weights across different holdings. This distribution was a direct outcome of the analytical processes mentioned, highlighting the usefulness of the SVC model in creating a portfolio.

Table 3.12: Weight Distribution of SVC

Assets	**Allocations**	**Assets**	**Allocations**
DIS	0.000000007779476	PG	0.000000004092733
VZ	0.000000005604903	SYY	0.000000006255391
HD	0.0000001052161	WMT	0.000000004940773
NKE	0.00000001423772	CVX	0.000000005155912
KO	0.000000005005611	JPM	0.00000001454095
MCD	0.00000001158054	XOM	0.000000003530744
AXP	0.00000001171231	JNJ	0.000000005215589
PFE	0.000000004751994	MRK	0.000000004980114
BA	0.00000001232032	WBA	0.000000005464575
CAT	0.000000007169119	MMM	0.000000006815777
AAPL	0.3494082	CSCO	0.000000004508817
IRX	0.6505915	IBM	0.000000004054405
INTC	0.00000000560477	**Total**	**1.00000**

In Table 3.12, the study presented a portfolio with minimal allocations to stocks except for one. This strategy was visually depicted as follows:

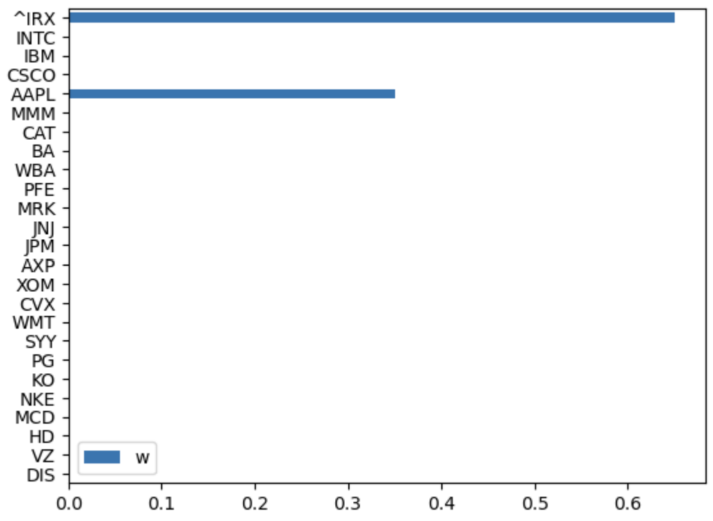


Figure 3.7: Weight distribution of stocks for last week

From Figure 3.7, the recommended allocation was to allocate 65% of the portfolio to a risk-free asset and 35% to Apple stock. The only considered risky asset was an information sector company.

In this study, the financial performance was investigated using the Sharpe ratio. The analysis focused on a specific four-week trading period from December 26, 2017, to January 16, 2018. The portfolio returns were calculated using optimized asset weights, which remained constant throughout this period and for each week under review.

Table 3.13: Sharpe Ratios of Mean-Variance Portfolios

Models	**Sharpe Ratios**
LR	0.9018486447556565
KNN	0.831808255665903
SVC	1.7157832748028983
DT	1.7488074992173797
RF	1.7345387582840919
XGBoost	1.562782580579704
$\rho = 0.5$	1.4311894392325304
Equal weights	1.4492115443467954

Table 3.13 showed a range of Sharpe ratios from 83.18% to 174.88%. It indicated that the portfolios optimized for risk aversion using DT methodology had the highest Sharpe ratio. On the other hand, the portfolio created using the KNN approach had the lowest ratio in the observed range. This difference highlighted the significant impact of predictive modeling techniques on the risk-adjusted returns of portfolios.

3.2.2.2 Mean-CVaR Portfolios

After the implementation of a confidence level as 95%, risk-aversion adjusted mean-CVaR portfolios ranged from 0.0063% to 0.2054%

Table 3.14: Out of one week returns

Models	**Returns**
LR	0.001604944587767839
KNN	0.002049988051708565
SVC	0.0020542292269545583
DT	0.0019870370975106234
RF	0.0020309732602152818
XGBoost	0.001972091506583323
$\rho = 0.5$	0.001977057562171938
Equal weights	0.00006342

Based on the Table 3.14 related to the evaluation of investment strategies, it had been observed that the SVC algorithm displayed the most favorable return among the models under consideration. Comparative analysis among various machine learning models demonstrated that the Logistic Regression (LR) model and portfolios balanced through equal-weight strategies yielded minimal financial returns over one week. The insights from the assessment conducted the last week were summarized in the following table, which outlined the optimal asset allocation among the different investment holdings.

Table 3.15: Weight Distribution of SVC

Assets	**Allocations**	**Assets**	**Allocations**
DIS	0	PG	0.114557
VZ	0.029191	SYY	0
HD	0	WMT	0.132165
NKE	0	CVX	0
KO	0.192442	JPM	0
MCD	0.250342	XOM	0
AXP	0	JNJ	0.198371
PFE	0	MRK	0
BA	0	WBA	0
CAT	0	MMM	0.040068
AAPL	0.024572	CSCO	0
IRX	0.018292	IBM	0
INTC	0	**Total**	**1.00000**

From Table 3.15, the mean-CVaR portfolio presented a diversified portfolio by allocating weights to eight different stocks. This strategy was depicted graphically as follows:

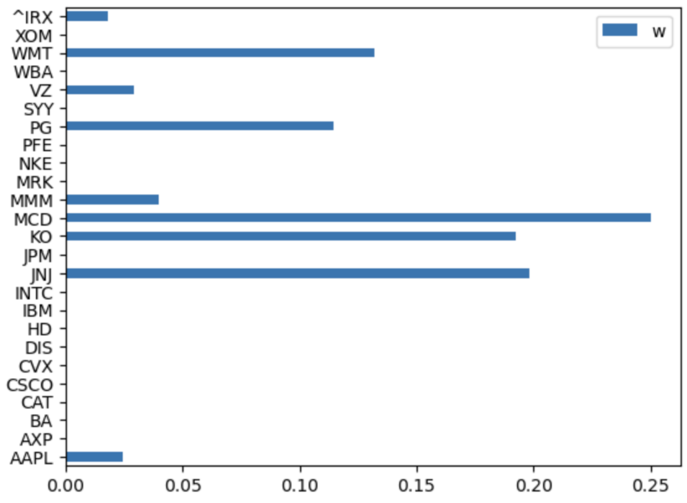


Figure 3.8: Weight distribution of stocks for last week

The recommended portfolio diversification suggested by the Mean-CVaR model was different from that of the Mean-Variance model, according to Figure 3.8. The Mean-

CVaR model suggested investing 19% in the healthcare industry, 45% in consumer staples companies, 2% in information technology companies, 25% in McDonald's Corporation, 4% in 3M Company, 3% in Verizon Communications Inc., and 2% in a risk-free asset. It was important to note that the recommended weightage for Apple Inc. and the risk-free asset was significantly less in the Mean-CVaR model compared to the Mean-Variance model. Additionally, the Mean-CVaR model did not include investments in the energy and financial sectors in its recommended portfolio diversification.

The Mean-CVaR model and sharpe ratio analysis were employed analogously to the Mean-variance approach. This exploration led to the generation of Table 3.16 delineating the Sharpe ratios, which serves as a comparative metric for assessing the risk-adjusted returns of different investment portfolios.

Table 3.16: Sharpe Ratios of Mean-CVaR Portfolios

Models	**Sharpe Ratios**
LR	1.0929695326016386
KNN	1.0545960361550237
SVC	1.0277092097811156
DT	1.0744360161385353
RF	1.0459181627048637
XGBoost	1.04105525283206
$\rho = 0.5$	1.0543797069508316
Equal weights	1.4492107223766362

The analysis showed a range of Sharpe ratios from 102.77% to 144.92%. It indicated that the portfolios optimized for risk aversion with equal weights had the highest Sharpe ratio and LR model among machine learning models. On the other hand, the portfolio created using the SVC approach exhibited the lowest ratio in the observed range. This difference highlighted the significant impact of predictive modeling techniques on the risk-adjusted returns of investment portfolios.

3.2.2.3 Mean-MAD Portfolios

We had 399 out-of-sample periods for machine learning predictions, and we aggregated the individual prediction results into a single accuracy score.

Table 3.17: Accuracy of models

Models	**Accuracy**
LR	0.418
KNN	0.611
SVC	0.418
DT	0.398
RF	0.626
XGBoost	0.365

Based on Table 3.17, the RF algorithm produced the highest overall accuracy, while SVC and LR had the same overall accuracy. XGBoost had the lowest accuracy in all periods. Table 3.18 displayed the performance comparisons of the eight portfolios with respect to the performance metrics including the out-of-sample average return, standard deviation, Sharpe ratio.

Table 3.18: Out of one week results

Models	**Average Return**	**Standard Deviation**	**Sharpe Ratios**
LR	1.26%	23.30%	5.40%
KNN	2.11%	30.35%	6.96%
SVC	10.61%	72.29%	14.68%
DT	6.94%	55.04%	12.61%
RF	7.37%	51.78%	14.23%
XGBoost	6.13%	53.75%	11.40%
$\rho = 0.5$	8.23%	59.85%	13.74 %
Equal weights	1.11%	4.16%	26.75%

In the analysis of risk-aversion adjusted mean-MAD portfolios, with returns spanning from 1.11% to 10.61%, it was observed that portfolios utilizing SVC algorithms yielded the highest average return. Conversely, equal weights portfolios had the lowest average returns throughout 399 one-week periods. The exploration of diverse portfolio configurations illuminated the predominant efficiency of the equal weights portfolio strategy. This approach prescribed an allocation of 4% to each asset within the portfolio, a method that was empirically demonstrated to outperform alternative strategies when assessed against the criterion of out-of-sample standard deviation. The quintessence of this configuration's superior performance was encapsulated by its minimization of standard deviation, which facilitated an enhancement in the Sharpe ratio, culminating in a remarkable figure of 26.75%. The attainment of such a Sharpe ratio not only signified the exceptional performance of the equal weights portfolio but also established it as the most favorable strategy in terms of risk-adjusted returns. When the panorama of portfolio configurations was considered, excluding the paramount equal weights portfolio, it was observed that the SVC portfolio emerged as the strategy with the subsequent highest Sharpe ratio. The LR portfolio was found to have the lowest Sharpe ratio among all the strategies. Given that SVC boasted the highest Sharpe ratio, an in-depth analysis was conducted on its weight distribution across 399 distinct portfolios. The research analyzed the distribution of weights across the initial and two arbitrarily chosen investment portfolios. This approach provided insightful perspectives into the strategic allocation strategies employed across varying portfolio constructs.

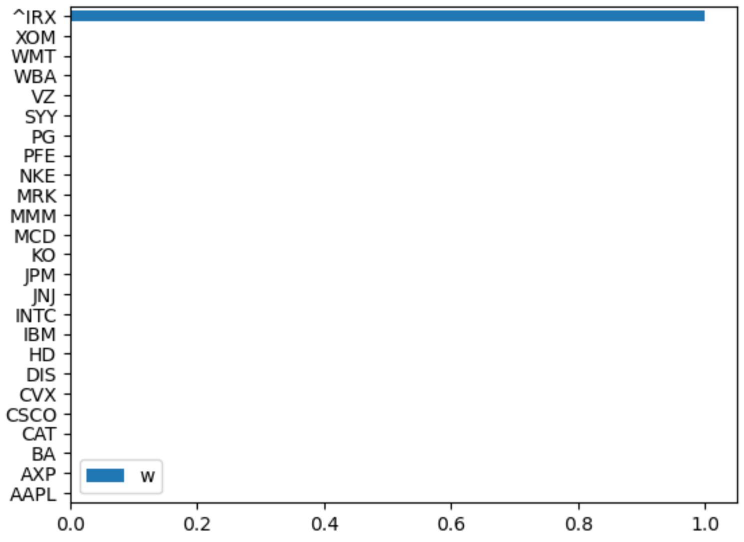


Figure 3.9: SVC-Weight distribution of stocks for first week

According to Figure 3.9, the SVC model suggested investing all capital in a risk-free asset only.

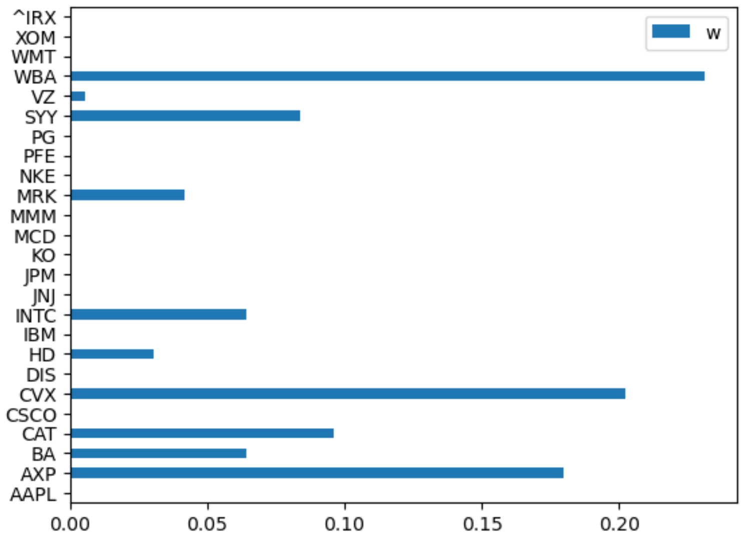


Figure 3.10: SVC-Weight distribution of stocks for a random week

According to Figure 3.10, the SVC model suggested investing all capital in ten different stocks. It gave the highest weight to Walgreens Boots Alliance, Inc. (WBA) stock.

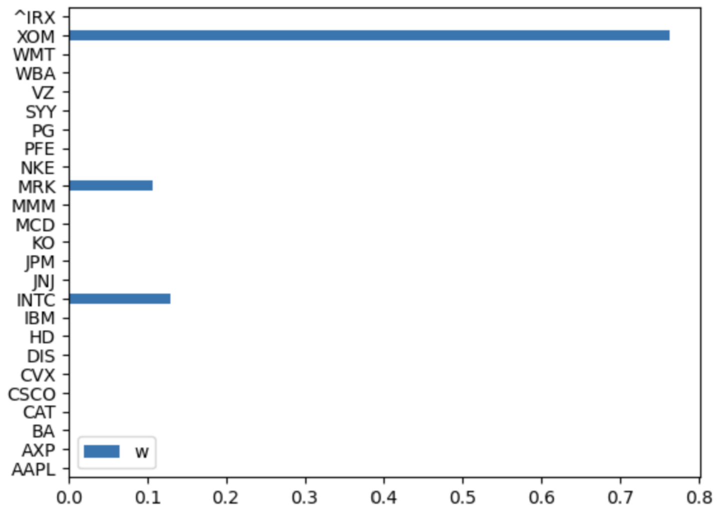


Figure 3.11: SVC-Weight distribution of stocks for a random week

According to Figure 3.11, the SVC model suggested investing all capital in three different stocks. It gave the highest weight to Exxon Mobil Corporation's (XOM) stock.

Table 3.19: Out of one week results

Models	**Cumulative Return**	**Annualized Return**
LR	13.98%	1.72%
KNN	-6.85 %	-0.92 %
SVC	-85.40%	-22.23%
DT	-97.13%	-37.11%
RF	2654.87%	54.22%
XGBoost	-98.02%	-40.09%
$\rho = 0.5$	171.71%	13.95%
Equal weights	5927.88%	70.84%

In Table 3.19, the performance of eight different portfolios was summarized across out-of-sample periods, specifically focusing on their cumulative and annualized returns. The data indicated that portfolios with equal weights outperformed others in terms of cumulative returns, demonstrating their superiority over variably-weighted strategies. Additionally, within the realm of machine learning models, the Random Forest (RF) approach stood out, offering superior portfolio returns. When examining annualized returns, the performance pattern remained the same; equal-weighted strategies across all models and the RF method within machine learning models exhibited superior performance. This consistent performance in both cumulative and annualized returns underscored the effectiveness of equal weighting and the RF technique in managing portfolio returns over the specified out-of-sample periods.

3.2.2.4 Return Predictions

Table 3.20: Asset Symbols and Corresponding Values for Last Week

Asset Symbol	Actual Value	Asset Symbol	Actual Value
DIS	105.34	BA	281.23
VZ	38.78	CAT	134.25
HD	162.65	MMM	183.94
MCD	148.08	AAPL	41.32
NKE	59.26	CSCO	31.90
KO	37.70	IBM	108.30
PG	77.92	PFE	26.94
SYY	51.87	WBA	55.98
WMT	29.43	CVX	95.53
CVX	95.53	XOM	61.86
MRK	44.77	AXP	90.32
JPM	89.71	JNJ	118.25
INTC	39.61	ÎRX	1.30

In Table 3.20, the adjusted closing prices from the final week, which was designated as the test week for machine learning models, were presented. This data corresponded to original asset price values. The study aimed to predict the out-of-sample returns for a single period of assets by employing four models, namely Linear Regression, LSTM, XGBoost, LightGBM. The comparison of the out-of-sample returns was made by applying a specific methodology. The adjusted close of the test week asset prices was first predicted and subsequently employed for return prediction. Machine learning models were developed and trained employing this methodology while excluding the use of the EMV and VWAP technical indicators. The rationale behind this exclusion lay in the observed computational inaccuracities arising during the calculation of these indicators when applied to a risk-free asset. The evaluation of all predictive models was conducted using a 10-fold cross-validation approach for time series data. The exception to this methodology was the implementation for Long Short-Term Memory (LSTM) networks, where a 3-fold cross-validation was utilized. For the optimization of hyperparameters across the various models, a comprehensive grid search strategy was employed, with the notable exception of the Light Gradient Boosting Machine (LightGBM). The utilization of grid search as a prevalent methodology for hyperparameter tuning in regression models was deemed unsuitable for LightGBM within the context of this study, primarily due to constraints associated with computational memory. Consequently, this necessitated the exploration of alternative techniques that are more conducive to optimizing hyperparameters specifically for LightGBM.

It was expected that this study would provide a comprehensive comparison of the models and their respective performance in predicting out-of-sample returns for a single period of portfolio optimization models.

Table 3.21: RMSE values for test week price predictions for the assets

Assets	**Linear Regression**	**LSTM**	**XGBoost**	**LightGBM**
Disney	0.059	2.211	2.869	**0.019**
Verizon	**0.012**	1.017	1.06	0.896
Home Depot	**0.048**	6.480	20.92	14.71
McDonald's	**0.026**	0.376	12.738	5.469
Nike	**0.031**	1.5822	0.547	0.845
Coca-Cola	**0.031**	0.553	0.651	0.150
Procter & Gamble	**0.043**	1.185	1.204	0.334
Sysco	**0.037**	0.808	4.458	3.111
Walmart	**0.013**	1.412	1.444	1.433
Chevron	**0.086**	5.422	5.635	5.119
ExxonMobil	**0.010**	0.439	0.415	0.045
American Express	**0.024**	0.158	2.669	2.447
JPMorgan Chase	**0.033**	0.815	8.056	6.303

Based on the data presented in the Table 3.21 and Table 3.22, the analysis of price prediction across various assets indicated that Linear Regression had superior performance for twenty one stocks and a risk-free asset. Conversely, LightGBM emerged as the optimal model for two specific stocks, namely Disney and Johnson & Johnson. Furthermore, the LSTM model was identified as the most effective for predicting the price of Walgreens Boots Alliance, distinguishing itself as the best choice for this single stock.

Table 3.22: RMSE values for test week price predictions for the assets

Assets	Linear Regression	LSTM	XGBoost	LightGBM
Johnson & Johnson	**0.015**	0.435	0.699	3.341
Merck Co.	0.055	0.582	0.965	**0.036**
Pfizer	**0.055**	0.550	1.331	0.345
WBA	0.010	**0.002**	4.090	0.222
Boeing	**0.143**	9.704	31.011	26.584
Caterpillar	**0.033**	9.994	22.458	14.322
3M	**0.034**	3.496	11.645	10.507
Apple	**0.019**	0.413	0.971	1.453
Cisco	**0.005**	0.219	2.632	1.802
IBM	**0.020**	0.747	0.985	0.276
Intel	**0.0009**	0.848	2.854	1.585
13-Week Treasury Bill	**0.0005**	0.054	0.150	0.090

This analysis underscored the predominance of Linear Regression in terms of overall efficacy in stock price predictions.

1-Linear Regression

According to the linear regression estimation of last week price prediction, the model predicted that the prices of nine out of twenty-five assets would be higher than their original price. Time series cross-validation with 10 folds and grid search were used for hyperparameter tuning. The grid search space was designed to determine whether to calculate the intercept for all stock prices and whether to set all coefficients to be positive. The RMSE ranged from 0.0005 to 0.143, with the treasury bill having the lowest RMSE. The model made the best prediction for the treasury bill. Risk-aversion adjusted mean-variance model comparisons according to different return estimation processes are as follows:

Table 3.23: Out of one week returns with last week prediction

Models	**Returns**
LR	-0.0037361484592123607
KNN	-0.005141642671989341
SVC	0.01691149989701578
DT	0.015076832241855334
RF	0.011412545613675483
XGBoost	0.00711055801663313
$\rho = 0.5$	0.004926820865999038
Equal weights	-0.0001812976

In Table 3.23, the outcomes suggested that the predicted returns oscillated within a delineated range, notably between -0.5142% and 1.6911%. It was observed that the KNN algorithm exhibited the largest loss, whereas the SVC demonstrated the highest return among the evaluated returns. Risk-aversion adjusted mean-CVaR model comparisons according to different return estimation processes were as follows:

Table 3.24: Out of one week returns with last week prediction

Models	**Returns**
LR	0.0014713128120319822
KNN	0.0019141552165056555
SVC	0.001931769100308993
DT	0.0018643844536794995
RF	0.0019067477729840063
XGBoost	0.0018464424336787934
$\rho = 0.5$	0.0018493155034222614
Equal weights	-0.0001812976

In Table 3.24, the outcomes suggested that the predicted returns oscillated within a delineated range, notably between -0.018% and 0.1931%. Comparative analysis among various machine learning models demonstrated that the Logistic Regression (LR) model had lowest return and portfolio balanced through equal-weight strategy had largest loss over one week. Support Vector Classifier (SVC) demonstrated the highest return among the evaluated returns.

2-LSTM

Keras [94] was a profound deep-learning library dependent on TensorFlow to execute its functions. TensorFlow [1] was an open-source platform that empowered researchers to design and develop machine-learning models using Python. So these two libraries were used for construct LSTM.

LSTM framework commenced with the application of min-max normalization on the dataset as a fundamental preprocessing step. Subsequently, a sliding window methodology waas employed to encapsulate the temporal correlation inherent in pricing data, a pivotal aspect of our analysis. Cross-validation could be employed to determine optimal window lengths, or computation of errors across various window dimensions could be used. For the purposes of our experiment, an optimal window size of four weeks was identified. This decision was informed by the hypothesis that such a window size was adept at reflecting the temporal dynamics commonly encountered in monthly analytical frameworks.

Progressing to subsequent stages of our methodology, we delineated a function capable of processing the dataset alongside the predefined window size, thereby producing input and target data. The window size herein is indicative of the number of preceding time steps utilized for the prediction of the subsequent time step. By adopting a strategy of advancing one step at a time, we constructed four-week length windows. Prior to model training, the dataset was partitioned into training and test subsets. The training data was then employed to fit the model, post which the model's performance was appraised using the test data. To align with the preconditions for LSTM model training, the dataset was reshaped into a 3-dimensional array encapsulating 4 timestamps and 23 features at each step, adhering to the LSTM model's requirement of the input shape [samples, time steps, features].

In the initial phase of the fitting process, weights were arbitrarily assigned, setting the stage for the establishment of a sequential model. This model's optimization was rigorously pursued through the application of grid search techniques within the specific framework of time-series 3-fold cross-validation.

A critical component of the model's architecture was the incorporation of a fifty-node LSTM layer, chosen for its proficiency in handling sequential data. The model compilation phase was characterized by the adoption of the Adam optimizer. This choice was underpinned by the optimizer's robust performance in regression-based problems, further complemented by the selection of mean squared error as the preferred loss function. Such methodological decisions were instrumental in refining the model's predictive accuracy.

The subsequent model fitting procedure was meticulously designed, incorporating grid search parameters that specified a training duration of 20 epochs. Additionally, the model's parameters are finely tuned, with a batch size established at 100, alongside exploratory variations in the learning rate (0.001, 0.002) and the number of hidden units (16,32). This phase was pivotal in calibrating the model's responsiveness to the intricacies of the data it processed.

The predicted outputs underwent a transformation process, wherein their values were inverted to facilitate a more intuitive interpretation. The evaluation of the model's predictive precision was conducted through the computation of the root mean squared errors, providing a quantitative measure of the model's performance. This methodological approach underscored the meticulous and iterative process of model development and optimization, reflecting a commitment to achieving a high degree of accuracy in predictive analytics within the realm of time-series analysis. According to the LSTM estimations from last week, the RMSE ranged from 0.002 to 9.994. It was determined that the 13-Week Treasury Bill had the smallest estimation error, while the Caterpillar stock had the highest RMSE. Risk-aversion adjusted mean-variance model comparisons according to different return estimation processes were as follows:

Table 3.25: Out of one week returns with last week prediction

Models	**Returns**
LR	-0.013457085176366838
KNN	-0.013923279223326496
SVC	-0.006608388108209022
DT	-0.0072169336040906265
RF	-0.008432354915360776
XGBoost	-0.009859299521260885
$\rho = 0.5$	-0.0105836295756067
Equal weights	-0.0191789964

In Table 3.25, the outcomes suggested that the predicted returns oscillated within a delineated range, notably between -1.917% and -0.6608%. Equal weight strategy was identified as the one incurring the greatest loss amongst all portfolios evaluated. It was noted that the KNN algorithm exhibited the highest degree of loss relative to other machine learning models under consideration. Conversely, the SVC emerged as the model demonstrating the lowest loss. Risk-aversion adjusted mean-CVaR model comparisons according to different return estimation processes were as follows:

Table 3.26: Out of one week returns with last week prediction

Models	**Returns**
LR	-0.011960549689161151
KNN	-0.012711143785358725
SVC	-0.01256778138458701
DT	-0.011303682539524995
RF	-0.011371819065359688
XGBoost	-0.011295219080157064
$\rho = 0.5$	-0.011282633895633052
Equal weights	-0.0191789964

In Table 3.26, the outcomes suggested that the predicted returns oscillated within a delineated range, notably between -1.91% and -1.128%. It was observed that the equal weight portfolio showed the largest loss, and among the machine learning models, KNN exhibited the largest loss, whereas portfolio with constant risk aversion demonstrated the smallest loss among the evaluated models. It was observed that out of all the machine learning models, XGBoost showed the lowest amount of loss.

3-XGBoost

Apart from predicting risk aversion, XGBoost was used to predict prices. The hyperparameters were chosen to align with those utilized in the classification task employing the XGBoost methodology. The RMSE ranged from 0.150 to 31.011, with the treasury bill having the lowest RMSE. The model delivered its most accurate forecast for the treasury bill relative to other assets. Risk-aversion adjusted mean-variance model comparisons according to different return estimation processes were as follows:

Table 3.27: Out of one week returns with last week prediction

Models	**Returns**
LR	0.025350393910891764
KNN	0.021355244997715774
SVC	0.0840418182158452
DT	0.07882675418776922
RF	0.06841093297236035
XGBoost	0.056182416218239856
$\rho = 0.5$	0.04997510613734968
Equal weights	-0.0406623456

In Table 3.27, the outcomes suggested that the predicted returns oscillated within a delineated range, notably between -4.0662% and 8.4042%. It was observed that the equal weights showed the largest loss, and among the machine learning models, KNN exhibited the lowest return, whereas SVC demonstrated the highest return among the evaluated models. Risk-aversion adjusted mean-CVaR model comparisons according to different return estimation processes were as follows:

Table 3.28: Out of one week returns with last week prediction

Models	**Returns**
LR	-0.03344939190330786
KNN	-0.034870592322850225
SVC	-0.03475759145328796
DT	-0.03379573089709645
RF	-0.03468419730221972
XGBoost	-0.034783334198867207
$\rho = 0.5$	-0.03414452213446066
Equal weights	-0.0406623456

In Table 3.28, the outcomes suggested that the predicted returns oscillated within a delineated range, notably between -4.0662% and -3.3449%. It was observed that the equal weights showed the largest loss, and among the machine learning models, KNN exhibited the largest loss, whereas LR demonstrated the smallest loss among the evaluated models.

4-LightGBM

In the process of model development, initial values for hyperparameters had been employed as defaults. The RMSE observed varied between 0.0198 to 26.584, with the Disney stock having exhibited the lowest RMSE. Out of all the assets we analyzed, LightGBM showed the highest accuracy in predicting the performance of Disney stock. Risk-aversion adjusted mean-variance model comparisons according to different return estimation processes were as follows:

Table 3.29: Out of one week returns with last week prediction

Models	**Returns**
LR	0.0007419498002562211
KNN	-0.002620004123480686
SVC	0.05013129978426262
DT	0.045742769535535736
RF	0.03697776483748613
XGBoost	0.02668736494929491
$\rho = 0.5$	0.02146385563899879
Equal weights	-0.0312013812

In Table 3.29, the outcomes suggested that the predicted returns oscillated within a delineated range, notably between -3.1201% and 5.0131%. It was observed that the equal weights showed the largest loss, and among the machine learning models, KNN exhibited the largest loss, whereas SVC demonstrated the highest return among the evaluated models.

Risk-aversion adjusted mean-CVaR model comparisons for different return estimation processes are shown in Table 3.30. The outcomes suggested that the predicted returns oscillated within a delineated range, notably between -3.1201% and -2.3252%. It was observed that the equal weights showed the largest loss, and among the machine learning models, KNN exhibited the largest loss, whereas DT demonstrated the least loss among the evaluated models.

Table 3.30: Out of one week returns with last week prediction

Models	**Returns**
LR	-0.024631316536776384
KNN	-0.024822132872309444
SVC	-0.02367457154879119
DT	-0.023252547756465295
RF	-0.02346457503386626
XGBoost	-0.023575408267797773
$\rho = 0.5$	-0.023336143178965547
Equal weights	-0.0312013812

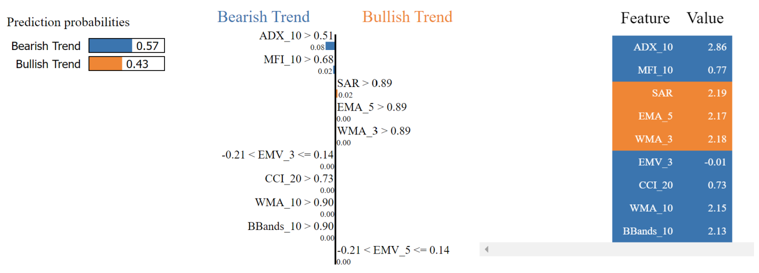
3.2.2.5 Feature Importance

It was crucial for a portfolio manager to have a good understanding of the behavior of machine learning models. This understanding helped explain to investors how their savings were managed. When constructing portfolios, portfolio managers could use machine learning models and explain why they had diversified their portfolios in a certain way. This new role of the portfolio manager was to act as an explainer using machine learning models in portfolio management. There were different types of explainers, but the SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) explainers were ones that could be used to explain the movement of the S&P 500. The SHAP explainer and the LIME explainer, both of which were accessible within the Python programming environment, were utilized to obtain interpretive insights into model predictions. Within the scope of the current research, SHAP and LIME were employed to elucidate the rationale behind the local test instance prediction. Additionally, SHAP was utilized to furnish global explanations pertaining to the training dataset. These methodologies were applied to models demonstrating the highest accuracy, specifically Logistic Regression (LR) and K-Nearest Neighbors (KNN), for predicting the movement of the S&P 500 index.

LIME (Local Interpretable Model-agnostic Explanations) Results

The classification of market trends was named into two distinct categories: bullish and bearish trends. To make the analysis and interpretation of tabular data more comprehensible, a function known as the tabular data explainer was developed. This innovatively designed function was crafted to accept the nomenclature of class labels—specifically "bullish" and "bearish" trends—as parameters, thus enhancing the readability and interpretability of the output. Additionally, it demanded the input of the training dataset along with a predefined list of features to operate optimally. LIME, a technique that perturbed features by drawing samples from a standard normal distribution, was utilized. This was followed by the inverse operation of mean-centering and scaling, which made use of the mean and standard deviation values derived from the training dataset. Through this permutation strategy, LIME was set to generate a sampling size of approximately 5,000 instances by default. The samples were then adjusted in value using an exponential kernel, wherein the kernel's width was, by default, set at 75% of the square root of the total number of features used. This adjustment aimed at transforming the distance metric into a measure of similarity.

The culmination of this process was the development of an explanation model that highlighted a maximum of ten features, providing insight into the test set. For the construction of this model, Ridge regression was chosen as the default explanation model. Logistic regression (LR) prediction with LIME was presented in the given visualization:



Feature	Value
ADX_10	2.86
MFI_10	0.77
SAR	2.19
EMA_5	2.17
WMA_3	2.18
EMV_3	-0.01
CCI_20	0.73
WMA_10	2.15
BBands_10	2.13

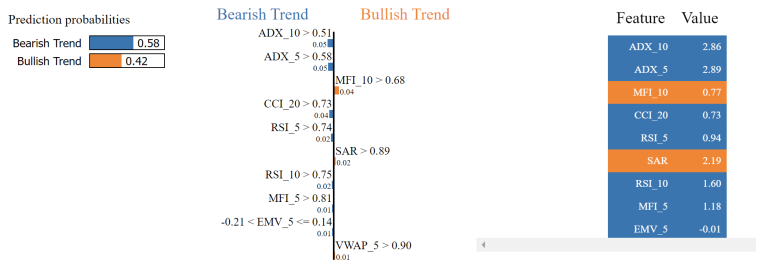
Figure 3.12: LIME explanation for test sample as a "Bearish Trend" by LR Model

Figure 3.12 provided information about test prediction. LR predicted a bullish trend of the S&P 500 with a 0.43 probability and a bearish trend with a 0.57 probability, and the final model estimation was a bearish trend. According to the plot, the Average Directional Index (ADX) with ten periods had the biggest effect. LIME focused on why the instance was classified as the more likely class. The middle of the figure delineated the top ten features which significantly influenced the decision-making process. These features were visually represented, with their contributions explicitly indicated. The employment of color coding, blue for technical analysis features suggesting a bearish trend, and orange for those signifying a bullish trend, was noted. Weights importance of prediction as follows:

Table 3.31: Technical Indicators and Their Corresponding Values

Indicators	**LIME Weights**
ADX_10 $>$ 0.51	-0.08250658482879819
MFI_10 $>$ 0.68	-0.018583373772887418
SAR $>$ 0.89	0.017677259216989864
EMA_5 $>$ 0.89	0.0018874384414025354
WMA_3 $>$ 0.89	0.0013308653358219727
-0.21 $<$ EMV_3 $\leq$ 0.14	-0.0012934602479181853
CCI_20 $>$ 0.73	-0.0012304407060790374
WMA_10 $>$ 0.90	-0.0010696718620313678
BBands_10 $>$ 0.90	-0.0010630833144003743
-0.21 $<$ EMV_5 $\leq$ 0.14	0.00102999773911544

Based on Table 3.31, it was noteworthy to mention that the employment of both ADX and the Money Flow Index (MFI), with their respective ten-period values of 2.86 and 0.77, significantly exceeded the threshold values of 0.51 and 0.68 utilized in the explanatory framework. When the ADX and the MFI with a ten-period setting were eliminated from the model, it was anticipated that the classifier would predict a bullish trend. This prediction adjustment correlated with a decrement of 0.1, which represented their cumulative weight as determined by the LIME. Furthermore, Stop and Reverse (SAR) had the highest weight for the bullish trend probability. The right feature table gave the original feature values. The LIME weights for each feature were corresponding coefficients values of explanation model in our case, ridge regression. The K-Nearest Neighbors (KNN) algorithm represented another model within our research that demonstrated a high level of accuracy. In order to further investigate its performance and interpretability, we employed LIME. The visualization of this analysis was provided in the subsequent figure:



Feature	Value
ADX_10	2.86
ADX_5	2.89
MFI_10	0.77
CCI_20	0.73
RSI_5	0.94
SAR	2.19
RSI_10	1.60
MFI_5	1.18
EMV_5	-0.01

Figure 3.13: LIME explanation for test sample as a "Bearish Trend" by KNN Model

The KNN algorithm was deployed to forecast the movement trends of the S&P 500, presenting a probability distribution wherein the likelihood of a bullish trend stood at 42% and that of a bearish trend at 58%. This prognostication aligned closely with the estimations derived from LR models. In further detail, Figure 3.13 elucidated that, consonant with the interpretations derived from LR analyses, the ADX over a span of ten periods exerted a paramount influence on the predictive outcomes. This finding underscored the significant role of the ADX in discerning the directional momentum of the S&P 500. Weights importance of prediction as follows:

Table 3.32: Technical Indicators and Their Corresponding Values

Indicators	**LIME Weights**
ADX_10 > 0.51	-0.052105842623155554
ADX_5 > 0.58	-0.04762622187351637
MFI_10 > 0.68	0.044948779444152494
CCI_20 > 0.73	-0.03786573352072073
RSI_5 > 0.74	-0.02368554837180563
SAR > 0.89	0.016402296156018962
RSI_10 > 0.75	-0.015340850332950882
MFI_5 > 0.81	-0.013978449164872966
-0.21 < EMV_5 <= 0.14	-0.013731656580025458
VWAP_5 > 0.90	0.011746843751864114

Based on Table 3.32, it was significant to highlight that the application of the ADX characterized by its ten-period and five-period values, which stood at 2.86 and 2.89 respectively, markedly surpassed the benchmark threshold values of 0.51 and 0.58 employed within the analytical model. Upon the removal of certain indicators from the analysis model—specifically the ADX, the Relative Strength Index (RSI) with a ten-period setting, the Commodity Channel Index (CCI) with a 20-period setting, and others including the MFI, and the Ease of Movement Value (EMV) adjusted to five periods— it was anticipated that the model's inclination would pivot towards favoring the bullish trend classification, attributing 0.2 as their collective weight value in the analytical process. Moreover, within the context of enhancing the model's accuracy in predicting bullish trend probabilities, the MFI with a ten-period was identified as the primary feature, followed by the SAR indicator which held secondary significance.

The LIME method significantly helped in understanding the decisions made by the KNN model. Specifically, when comparing the top 10 features influencing the model's output, the RSI, ADX, MFI, and VWAP stood out for their substantial contribution to predicting bearish trends. Their contribution's magnitude displayed a 1% difference from the LR model, highlighting their crucial role in driving such predictions.

SHAP Results

Similar to the approach taken with LIME, SHAP analysis was focused on identifying the bearish trend by selecting the model's target variable. This part focused on elucidating the outcomes of employing SHAP values to interpret the outputs generated by Logistic Regression (LR) and K-Nearest Neighbors (KNN) algorithms. Visualization of model-based SHAP values was important for the explanation. For single-period optimization mean-risk models, LR and KNN provided better accuracies among the models. The concentration was on understanding the technical indicators' importance for S&P 500 prediction using test data. In the study, the Kernel SHAP methodology was employed to elucidate the predictive outcomes generated by LR and KNN algorithms. SHAP local explanation of LR:

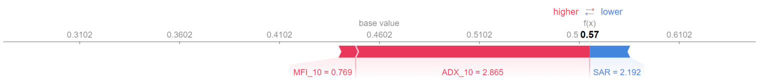


Figure 3.14: LR-SHAP force plot for test sample

Figure 3.14, the force plot, a notable visualization generated through SHAP, played a vital role in elucidating the contributions of individual predictors to a specific forecast. This mechanism of SHAP was notably crucial when examining singular predictions, where it benchmarked against a baseline – in this instance, the average predicted probability stood at 0.4602. Additionally, in the domain of technical analysis, indicators such as the Money Flow Index (MFI) and the Average Directional Index (ADX), particularly when considering ten-period features, had been identified to significantly augment the probability of a bearish trend emergence. Conversely, the Stop and Reverse (SAR) indicator of 2.192 showed limited effectiveness in mitigating this increased bearish trend. The influence of ADX and SAR on the outcome was notably significant, given their proximity to the crucial dividing boundary between the red and blue bars. Among the three features mentioned, the magnitude of the SHAP value of ADX was the biggest. SHAP local explanation of KNN:

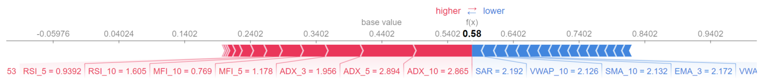


Figure 3.15: KNN-SHAP force plot for test sample

In Figure 3.15, K-Nearest Neighbors (KNN) with Shapley Additive Explanations (SHAP) force plot baseline prediction was calculated as 0.4402, the model's score was 1% bigger than LR's score. ADX's five and ten-period push prediction was bearish, while SAR and VWAP's ten-period push prediction was a bullish trend. In the KNN-SHAP analysis, it was observed that a broader set of features significantly contributed to the model's predictive outcome. Conversely, in the LR framework, SAR was identified as the singular feature that negatively impacted the model's outcome. Furthermore, within the scope of KNN, additional features such as Volume Weighted

Average Price (VWAP), Simple Moving Average (SMA), and Exponential Moving Average (EMA) were also found to diminish the model's outcome. This differential impact of features across models suggested a nuanced understanding of feature selection and its implications on model performance was crucial for optimizing predictive accuracy.

To gain a comprehensive understanding of the most significant features for a predictive model, it was recommended to visualize the SHAP values of every feature across all samples. This global approach served to underscore the criticality of integrating SHAP and the KNN model when assessing the training dataset. This methodology not only enhanced the interpretability of the model's decision-making process but also substantiated the relative importance of each feature within the predictive framework. The graphical depiction provided, it was noted that the preconfigured upper limit for the quantity of features exhibited stood at ten. Within this visual representation, individual points symbolized Shapley values, each corresponding to a distinct feature. The structure of the graph was characterized by its vertical axis, which quantified the variables. These variables were systematically arranged in a hierarchical sequence according to their average absolute SHAP values. Concurrently, the horizontal axis delineated the SHAP values themselves. Accompanying this graphical representation was a color bar situated on the right, serving the purpose of representing the numerical value attributed to the feature.

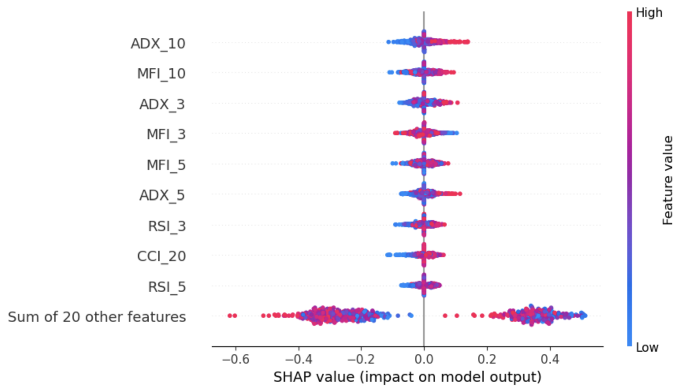


Figure 3.16: SHAP summary plot for training sample predictions as a "Bearish Trend" by the KNN model

Figure 3.16 suggested that the top five variables that were most significant were ADX (measured at two different time intervals) and MFI (measured at three different time intervals). The ADX over a period of 10 was the most essential feature, and a longer period of ADX and MFI were better than other periods for feature selection. In the figure presented, it was observed that elevated levels of the features, with the exception of the MFI across three periods, were associated with an increase in the bearish trend probability value by a maximum of 0.2. Conversely, diminished levels of these features corresponded with a decrease in the predictive output by up to 0.2. Additionally, in order to dissect the intricate interplay between feature values and SHAP values, the study presented dependence plots for the four paramount features—namely ADX and MFI with 10 periods, and ADX and MFI with 3 periods. These features had a significant impact on the predictive model for the S&P 500 index.

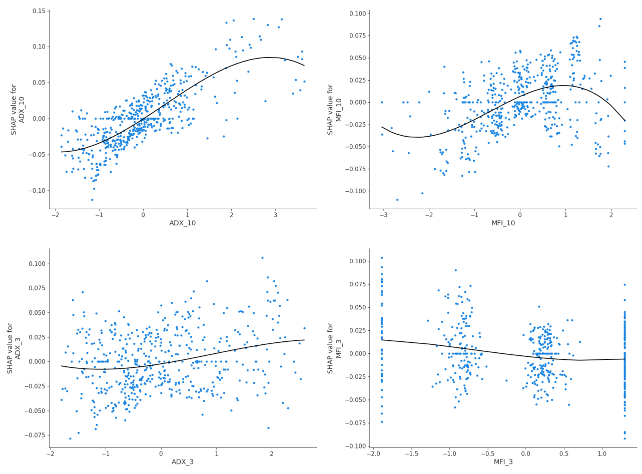


Figure 3.17: The SHAP dependence plot of the ADX_10, MFI_10, ADX_3, MFI_3, for the direction forecasting of the S&P 500 index

In an analysis that employed the ADX with 10 period dependence plot, particularly concerning the S&P 500 index, an intriguing dynamic was observed regarding the impact of features on market trend predictions. Specifically, an augmentation of the most critical feature from -2 to 3 was correlated with a rise in its SHAP value. This increment signified an enhanced predictive contribution towards anticipating index movement. Concurrently, as the ADX value escalated, a notable decrease in the SHAP value was observed, suggesting an amplification in its adverse impact on the predicted index trajectory's predictive accuracy. Thus, the probability of “falling” for the index price was decreased as the ADX value was increased.

As depicted in Figure 3.17, a notable observation was the behavior of the secondary but significant feature, denoted as MFI with 10 period. It was evidenced that this feature exhibited a gradual escalation within the specified ranges of (-3,-2) or (1, 2). This escalation inversely correlated with the likelihood of a decline in the index, suggesting a reduction in the probability of a "falling" scenario for the index within these ranges. Conversely, within the (-2,1) interval, there was an observable augmentation in the SHAP value, which was directly related to an increased probability of a decline in the index. This analysis suggested a nuanced interplay between the MFI feature's value ranges and the index's directional movements, highlighting the importance of considering such features in predictive models.

Furthermore, the analysis of the ADX with 3 period plot revealed a nuanced relationship with the SHAP values, where up until a -1 value of ADX, the SHAP value decreased alongside a reduced probability of index price decreases. Beyond this point, an escalation in the SHAP value was noted, which correlated with an increased likelihood of the index price decreasing. The relationship between MFI values and the probability of a decline in index prices demonstrated a noteworthy inverse correlation. Specifically, as the MFI value, denoted here as MFI with 3 period, increased, there was a concomitant decrease in the likelihood of a downturn in the index price. This observation suggested a pivotal role of the MFI in forecasting market trends, particularly downturns. This investigation underscored the intricate dynamics between ADX and MFI with 10 periods, and ADX and MFI with 3 periods and their SHAP values, and their collective impact on S&P 500 index price movements, providing substantial insights into predictive market analytics.

CHAPTER 4

CONCLUSION AND FUTURE WORK

The objective of this research was to analyze the performance trends of the S&P 500 and its impact on portfolio optimization methodologies. Our study primarily aimed at diversification, incorporating 24 stocks from eight distinct industry sectors. The focus was on crafting portfolios for both single and multiple periods that were adjusted for risk aversion, employing various risk metrics. For the construction of single-period portfolios, we utilized variance and Conditional Value at Risk (CVaR) as our chosen risk measures, while the Mean Absolute Deviation (MAD) metric was employed for the rebalancing of multi-period portfolios.

To refine our portfolio optimization process, we integrated six classification models: Logistic Regression, K Nearest Neighbors (KNN), Support Vector Classification (SVC), Decision Trees (DT), Random Forest (RF), and eXtreme Gradient Boosting (XGBoost). These were employed alongside 29 technical analysis indicators to predict risk aversion tendencies accurately.

Our findings indicated that the SVC model's prediction exhibited a propensity towards higher risk, whereas the KNN model's prediction was more conservative or risk-averse. When analyzing performance, portfolios assisted by SVC yielded the highest return for the risk-aversion adjusted mean-variance portfolio over a one-week period. On the other hand, portfolios assisted by KNN generated the largest loss. In the comparison of single period Sharpe ratios, we calculated the monthly Sharpe ratio. The DT model yielded the best result, while the portfolio derived using the KNN showed the lowest Sharpe ratio. Within the mean-Conditional Value at Risk (CVaR) portfolio framework, SVC achieved the highest return. Logistic Regression (LR) model and portfolios balanced through equal-weight strategies' portfolios had the lowest return. In comparing Sharpe ratios, the equal-weight portfolio outperformed all other portfolios, including the LR model which performed the best amongst other machine learning-based portfolios. Conversely, the SVC-derived portfolio exhibited the lowest Sharpe ratio.

In developing portfolios based on mean-MAD (Mean-Absolute Deviation) optimization, we employed a methodological framework characterized by the utilization of a rolling window approach. This technique involved setting the window size to encompass 104 weekly returns, a decision aimed at facilitating a comprehensive evaluation and optimization of mean-MAD portfolios. Such an approach fostered a deeper comprehension of our strategy's operational efficiency in real-world market conditions. For portfolios optimized using the MAD risk measure, the SVC model outperformed all others across an examined period, offering superior average return for 399 mean-MAD portfolios.

In the examination of superior portfolio construction methodologies, it is discerned, through rigorous analysis, that a portfolio with equitably distributed weights stands as the quintessential framework. Upon the extensive evaluation of a broad spectrum of portfolio strategies, with the exclusion of the previously mentioned equal weight portfolio, the portfolio configuration predicated on SVC methodology distinctly emerges as the contender with the second highest Sharpe ratio. This finding accentuates the predominance of the equal weights schema in the sphere of portfolio optimization, concurrently illuminating the promising capabilities of advanced analytical techniques, such as SVC, in refining investment strategies. The LR portfolio was found to have the lowest Sharpe ratio among all the strategies. When looking at cumulative and annualized returns, equal-weighted strategies across all models and the RF method within machine learning models show superior performance.

In the next phase of the study, we conducted an extensive examination of return forecasting through the deployment of four distinct regression models: Linear Regression, Long Short-Term Memory Networks (LSTM), XGBoost, and LightGBM within the framework of single-period portfolios. The variables utilized across all regression models were exclusively sourced from technical analysis, maintaining consistency with the variables selected for classification models. Our analysis primarily focused on the efficacy of these models in predicting the price for the test week of our dataset. The findings of the study revealed that Linear Regression emerged as the most efficacious predictive model, as it achieved the lowest Root Mean Square Error (RMSE) values for twenty-two assets. This was followed by LightGBM, which recorded the lowest RMSE values for two assets, and LSTM, which accomplished the lowest RMSE value for an asset, respectively. Notably, XGBoost did not secure the lowest RMSE value for any of the assets under investigation. Based on the calculations of returns, it was observed that the performance outcomes of models, in terms of both returns and losses, exhibit variability.

This finding was particularly noteworthy as LSTM, despite its popularity in numerous studies for stock price prediction, did not demonstrate superior performance in our investigation. The study suggested the possible overcomplexity associated with deploying deep learning methods, such as LSTM, for stock price prediction may not have justified its benefits. This assertion was further reinforced by the work of Kobets[52], which illustrated the potential of Linear Regression to outperform LSTM in similar contexts. Subsequent to the model evaluation, we analyzed the returns within these periods, supported by our optimized weights, to conduct a comparative analysis.

The investigation extended its focus to the examination of interpretation mechanisms for achieving optimum predictive performance, specifically analyzing the K-Nearest Neighbors (KNN) and Logistic Regression (LR) models within a defined temporal context. The incorporation of Explainable Artificial Intelligence (XAI) played a pivotal role in assisting decision-makers by elucidating the intricacies of sales and purchase activities via an in-depth understanding of feature relevance. For the purpose of furnishing localized elucidations during the evaluation phase, methodologies such as Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) were employed. Additionally, SHAP force plots were provided for both LR and KNN models.

Moreover, SHAP was utilized to determine the global significance of features within the KNN model's training phase, and the dependency plots of the four most influential features were also disclosed. The outcomes of this inquiry highlighted the Average Directional Index (ADX) with a period scope of ten as the most critical and influential feature for forecasting movements in the S&P 500 index. It was proposed that investors could benefit from the application of XAI to elucidate feature significances, based on the insights derived from their research findings.

In forthcoming scholarly endeavors, it is paramount to consider supplementary variables, including text mining methodologies and investor sentiment, to augment the predictive accuracy of trends within the S&P 500. The integration of lag variables alongside data on opening and minimum prices promises to enhance the richness of the dataset significantly. Consequently, expanded investigations into market index forecasting are warranted to elevate predictive precision. Moreover, the adoption of various risk assessment techniques for the construction of either single or multi-period investment portfolios represents a promising area for future research exploration.

REFERENCES

[1] M. Abadi and et al. tensorflow/tensorflow: An Open Source Machine Learning Framework for Everyone. `https://github.com/tensorflow/tensorflow`. [Accessed 01-03-2024].

[2] S. B. Achelis. *Technical Analysis from A to Z*. The MIT Press, 2001.

[3] A. Agarwal, A. Bhatia, A. Malhi, P. Kaler, H. S. Pannu, et al. Machine learning based explainable financial forecasting. In *2022 4th International Conference on Computer Communication and the Internet (ICCCI)*, pages 34–38. IEEE, 2022.

[4] C. C. Aggarwal. *Linear Algebra and optimization for Machine Learning: A textbook*. SPRINGER NATURE, 2021.

[5] E. Alpaydin. *Introduction to machine learning*. The MIT Press, 2020.

[6] A. S. Arefin, C. Riveros, R. Berretta, and P. Moscato. Gpu-fs-k nn: A software tool for fast and scalable k nn computation using gpus. pages 1–13, 2012.

[7] J. Ayala, M. García-Torres, J. L. V. Noguera, F. Gómez-Vela, and F. Divina. Technical analysis strategy optimization using a machine learning approach in stock market indices. *Knowledge-Based Systems*, 225:107119, 2021.

[8] G.-Y. Ban, N. El Karoui, and A. E. Lim. Machine learning and portfolio optimization. *Management Science*, 64(3):1136–1154, 2018.

[9] H. Bandi, S. Joshi, S. Bhagat, and D. Ambawade. Integrated technical and sentiment analysis tool for market index movement prediction, comprehensible using xai. In *2021 International Conference on Communication information and Computing Technology (ICCICT)*, pages 1–8. IEEE, 2021.

[10] J. Benediktsson. TA-Lib/ta-lib-python: Python wrapper for TA-Lib (http://ta-lib.org/). `https://github.com/TA-Lib/ta-lib-python/`. [Accessed 21-02-2024].

[11] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[12] T. Bourgeron, E. Lezmi, and T. Roncalli. Robust asset allocation for robo-advisors. pages 1–67, 2019.

[13] L. Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[14] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[15] A. Burkov. *The hundred-page machine learning book*. Eleven Languages, 2019.

[16] P. Byrne and S. Lee. Different risk measures: different portfolio compositions? *Journal of Property Investment & Finance*, 22(6):501–511, 2004.

[17] J. Y. Campbell, A. W. Lo, A. C. MacKinlay, and R. F. Whitelaw. The econometrics of financial markets. *Macroeconomic Dynamics*, 2(4):559–562, 1998.

[18] S. Carta, A. S. Podda, D. Reforgiato Recupero, and M. M. Stanciu. Explainable ai for financial forecasting. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 51–69. Springer, 2021.

[19] T. B. Çelik, Ö.İcan, and E. Bulut. Extending machine learning prediction capabilities by explainable ai in financial time series prediction. *Applied Soft Computing*, 132:109876, 2023.

[20] T. Chen. Introduction to boosted trees. *University of Washington Computer Science*, 22(115):14–40, 2014.

[21] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[22] Y. Chen and Y. Hao. A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, 80:340–355, 2017.

[23] R. W. Colby. *The encyclopedia of technical market indicators*. McGraw-Hill, 2003.

[24] R. Confalonieri, L. Coba, B. Wagner, and T. R. Besold. A historical perspective of explainable artificial intelligence. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(1):e1391, 2021.

[25] P. Cunningham and S. J. Delany. k-nearest neighbour classifiers-a tutorial. *ACM computing surveys (CSUR)*, 54(6):1–25, 2021.

[26] K. Daniel, D. Hirshleifer, and A. Subrahmanyam. Investor psychology and security market under-and overreactions. *the Journal of Finance*, 53(6):1839–1885, 1998.

[27] S. Deng, X. Huang, Y. Zhu, Z. Su, Z. Fu, and T. Shimada. Stock index direction forecasting using an explainable extreme gradient boosting and investor sentiments. *The North American Journal of Economics and Finance*, 64:101848, 2023.

[28] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[29] R. M. Dhokane and S. Agarwal. Enhancing stock price prediction with macd and ema features using lstm algorithm. In *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, pages 1–6. IEEE, 2024.

[30] P. Dubach. A python integration of practical asset allocation based on modern portfolio theory and its advancements. pages 1–95, 2021.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. Wiley, 2001.

[32] F. J. Fabozzi, H. M. Markowitz, and F. Gupta. Portfolio selection. *Handbook of finance*, 2, 2008.

[33] E. F. Fama. The behavior of stock-market prices. *The Journal of Business*, 38(1):34–105, 1965.

[34] E. F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970.

[35] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874, 2008.

[36] J. Fernando. Volume-Weighted Average Price (VWAP): Definition and Calculation. `https://www.investopedia.com/terms/v/vwap.asp#:~:text=VWAP%20is%20calculated%20by%20multiplying,number%20of%20periods%20(10).` [Accessed 14-02-2024].

[37] D. V. Fryer, I. Strümke, and H. D. Nguyen. Shapley values for feature selection: The good, the bad, and the axioms. *CoRR*, abs/2102.10936:1–8, 2021.

[38] J. W. Goodell, S. B. Jabeur, F. Saâdaoui, and M. A. Nasir. Explainable artificial intelligence modeling to forecast bitcoin prices. *International Review of Financial Analysis*, page 102702, 2023.

[39] J. Han, M. Kamber, and J. Pei. Data mining concepts and techniques third edition. *University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University*, 2012.

[40] A. D. Hartanto, Y. N. Kholik, and Y. Pristyanto. Stock price time series data forecasting using the light gradient boosting machine (lightgbm) model. *JOIV: International Journal on Informatics Visualization*, 7(4):2270–2279, 2023.

[41] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[42] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[43] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. pages 1–16, 2003.

[44] A. I. Hunjra, S. M. Alawi, S. Colombage, U. Sahito, and M. Hanif. Portfolio construction by using different risk models: A comparison among diverse economic scenarios. *Risks*, 8(4):126, 2020.

[45] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor. *An introduction to statistical learning: With applications in python*. Springer Nature, 2023.

[46] R. Ji, K. Chang, and p. Jiang. Risk-aversion adjusted portfolio optimization with predictive modeling. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2019.

[47] Z. Jiang, R. Ji, and K.-C. Chang. A machine learning integrated portfolio rebalance framework with risk-aversion adjustment. *Journal of Risk and Financial Management*, 13(7):155, 2020.

[48] P. Jorion. *Value at risk: the new benchmark for managing financial risk*. McGraw-Hill, 2007.

[49] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:1–9, 2017.

[50] C. Kingsford and S. L. Salzberg. What are decision trees? *Nature biotechnology*, 26(9):1011–1013, 2008.

[51] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. *Elpub*, 2016:87–90, 2016.

[52] V. Kobets and S. Savchenko. Building an optimal investment portfolio with python machine learning tools. pages 1–9, 2022.

[53] H. Konno and H. Yamazaki. Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market. *Management Science*, 37(5):519–531, 1991.

[54] S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.

[55] W. Leigh, R. Purvis, and J. M. Ragusa. Forecasting the nyse composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support. *Decision support systems*, 32(4):361–377, 2002.

[56] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[57] J. Longerstaey and M. Spencer. Riskmetricstm—technical document. *Morgan Guaranty Trust Company of New York: New York*, 51:54, 1996.

[58] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30:1–10, 2017.

[59] Y. Ma, R. Han, and W. Wang. Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, 165:113973, 2021.

[60] R. Mansini, W. ,odzimierz Ogryczak, M. G. Speranza, and E. T. A. of European Operational Research Societies. *Linear and mixed integer programming for portfolio optimization*, volume 21. Springer, 2015.

[61] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[62] H. M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. Yale University Press, 1959.

[63] K. May. *Forecast Based Portfolio Optimisation Using XGBoost*. PhD thesis, University of the Witwatersrand, Johannesburg, 2022.

[64] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[65] G. J. McLachlan. *Discriminant analysis and Statistical Pattern Recognition*. John Wiley & Sons, 2005.

[66] C. Mitchell. What is the commodity channel index (cci)? how to calculate. https://www.investopedia.com/terms/c/commoditychannelindex.asp. [Accessed 14-02-2024].

[67] C. Molnar. *Interpretable Machine Learning*. Christoph Molnar, 2 edition, 2022.

[68] M. Monikasri and S. Varshini. Stock market price prediction using machine learning. pages 1–6, 2021.

[69] J. P. Mueller. *Machine learning for dummies*. John Wiley Sons Inc, 2021.

[70] J. J. Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. New York Institute of Finance, 1999.

[71] M. Nikou, G. Mansourfar, and J. Bagherzadeh. Stock price prediction using deep learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, 26(4):164–174, 2019.

[72] S. M. Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[73] D. Padial. Technical Analysis Library using Pandas and Numpy, 2018. `https://github.com/bukosabino/ta/`. [Accessed 21-02-2024].

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[75] L. C. M. Phuong. Investor sentiment by money flow index and stock return. *International Journal of Financial Research*, 12(4):33–42, 2021.

[76] Posit team. *RStudio: Integrated Development Environment for R*. Posit Software, PBC, Boston, MA, 2023.

[77] K. Postek, A. Zocca, J. Gromicho, and J. Kantor. *Hands-On Mathematical Optimization with AMPL in Python*. Online, 2024.

[78] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[79] S. Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24), Apr. 2018.

[80] S. Raschka, Y. H. Liu, V. Mirjalili, and D. Dzhulgakov. *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing Ltd, 2022.

[81] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[82] R. T. Rockafellar, S. Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[83] B. Rockefeller and M. Tusing. *Technical analysis for dummies*. Tantor Media, 2019.

[84] T. Roncalli. *Introduction to risk parity and budgeting*. CRC Press, 2013.

[85] F. Rosenblatt and S. Papert. *Perceptron*, volume 9. April, 2021.

[86] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[87] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[88] J. A. Ryan. `https://cran.r-project.org/web/packages/quantmod/quantmod.pdf`. [Accessed 22-02-2024].

[89] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[90] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[91] M. D. Sheimo. *Cashing in on the dow: Using dow theory to trade and determine trends in today's markets*. John Magee, 1998.

[92] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.

[93] Z. Tasneem and M. M. Rahman. Machine learning based approaches for predicting stock closing price. In *2023 26th International Conference on Computer and Information Technology (ICCIT)*, pages 1–6. IEEE, 2023.

[94] K. Team. Keras: Deep Learning for humans — keras.io. `https://keras.io/`. [Accessed 01-03-2024].

[95] L. Troiano, A. Bhandari, and E. M. Villa. *Hands-On Deep Learning for Finance: Implement deep learning techniques and algorithms to create powerful trading strategies*. Packt Publishing Ltd, 2020.

[96] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[97] G. Van Rossum and F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[98] J. W. Wilder. *New Concepts in Technical Trading Systems*. Trend Research, 1978.

[99] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and techniques with Java implementations*. Morgan Kaufmann, 2005.

[100] D. Würtz, Y. Chalabi, W. Chen, and A. Ellis. *Portfolio optimization with R/Rmetrics*. Rmetrics, 2009.

[101] Y. Zhu and G. Zhou. Technical analysis: An asset allocation perspective on the use of moving averages. *Journal of financial economics*, 92(3):519–544, 2009.

APPENDIX A

LIST OF ASSETS

Table A.1: List of Assets

S&P 500 Sectors	**Company Name**
Communication Services	The Walt Disney Company
	Verizon Communications Inc.
Consumer Discretionary	The Home Depot, Inc.
	McDonald's Corporation
	NIKE, Inc.
Consumer Staples	The Coca-Cola Company
	The Procter Gamble Company
	Sysco Corporation
	Walmart Inc.
Energy	Chevron Corporation
	Exxon Mobil Corporation
Financials	American Express Company
	JPMorgan Chase Co.
Health Care	Johnson Johnson
	Merck Co., Inc.
	Pfizer Inc.
	Walgreens Boots Alliance, Inc.
Industrials	The Boeing Company
	Caterpillar Inc.
	3M Company
Information Technology	Apple Inc.
	Cisco Systems, Inc.
	International Business Machines Corporation
	Intel Corporation
-	13-Week Treasury Bill