University of Hertfordshire UH

School of Physics,
Engineering and
Computer Science

MSc Data Science Project
7PAM2002-0509-2023

Department of Physics, Astronomy and Mathematics

Data Science Final Project Report

Project Title:

Forecast Stock Market Trends and Portfolio Optimization Using Neural Networks and Machine Learning Algorithms.

Student Name and SRN:

Venkata Naga Sai Sasidhar Pasarlaputi [21063192]

Supervisor: Mr. Ralf Napiwotzki

Date Submitted: 29th August 2024

Word Count: 7501

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6). I have not used chatGPT, or any other generative AI tool, to write the report or code (other than where declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: VENKATA NAGA SAI SASIDHAR PASARLAPUTI

Student Name signature: Venkata Naga Sai Sasidhar Pasarlaputi

Student SRN number: 21063192

ETHICAL STATEMENT

This project, titled "Forecast Stock Market Trends and Portfolio Optimization Using Neural Networks and Machine Learning Algorithms," is conducted with a commitment to ethical principles, ensuring that the research and its applications align with the highest standards of integrity and responsibility.

The analysis was conducted using historical share prices provided by Yahoo Finance and it does not involve the use of personal or sensitive information. Any data used are strictly financial in nature and do not include individual investor data and it meets the GDPR requirements and meets the UH ethical policies.

ACKNOWLEDGEMENT

As my post-graduate studies come to an end, I look back on this journey and am incredibly grateful for all the people who have supported and encouraged me along the way.

I owe a debt of gratitude to Mr. Ralf Napiwotzki, my supervisor, for his constant encouragement and guidance during this project. His patience, insightful advice, and constant encouragement have been crucial to my work's successful completion.

I want to thank each one of my University of Hertfordshire lecturers. Their commitment and knowledge have been a constant source of inspiration and have enhanced my comprehension of the subjects.

I also want to express my sincere gratitude to my parents, brother, and friends for their unwavering support and encouragement. My accomplishments have been propelled by their faith in me.

I thank everyone who has helped me along the way in my academic career, whether directly or indirectly. Without your combined support, this feat would not have been possible.

ABSTRACT

To maintain a competitive advantage in stock trading and portfolio management, it is essential to utilize sophisticated prediction models due to the fast development of financial markets. The objective of this project, named "Forecast Stock Market Trends and Portfolio Optimization Using Neural Networks and Machine Learning Algorithms," is to use advanced machine learning techniques and technical analysis to predict stock market trends and improve investment portfolios. We employ three technical indicators, namely Moving Average (MA), Relative Strength Index (RSI), and Moving Average Convergence Divergence (MACD), to catch significant market signals and augment the prediction powers of our models.

I evaluate three machine learning models: LSTM networks, CNNs, and Extreme Gradient Boosting. LSTM networks are ideal for sequential data like stock prices because they manage temporal relationships in time series data. Convolutional neural networks (CNNs) extract spatial properties from data to find intricate patterns that other approaches overlook. XGBoost was chosen for its durable gradient boosting architecture and efficient structured data management.

The findings of my experiment show that combining technical indicators with sophisticated machine learning algorithms has the capability to offer dependable recommendations for the stock market. This research enhances the area of quantitative finance by giving a systematic method for selecting models and optimizing portfolios. It offers significant information to investors who want to improve their decision-making processes.

TABLE OF CONTENTS

Table of Contents

TABLE OF FIGURES

1.INTRODUCTION

1.1. Overview

The stock market, an essential element of the worldwide economy, has consistently been a central focus for investors, financial analysts, and academics. Predicting stock market movements is a tough endeavour due to its unexpected nature and the substantial impact of several macroeconomic and microeconomic elements. Historically, financial professionals have depended on fundamental and technical research to predict stock values. Nevertheless, due to the emergence of sophisticated computational systems, the field of stock market prediction has undergone significant changes.

Neural networks and machine learning algorithms have become prominent tools for financial forecasting in recent years. These technologies have the capability to detect intricate patterns and relationships inside extensive datasets that conventional statistical approaches can fail to recognize. Machine learning models may utilize previous pricing data, market sentiment, economic indicators, and other pertinent elements to provide more accurate forecasts on future market patterns. Moreover, these models have the capability to be consistently improved as fresh data becomes accessible, hence augmenting their precision and dependability as time progresses.

Portfolio optimization is a vital component of financial management that aims to maximize returns while avoiding risks. By incorporating machine learning into portfolio management, it becomes possible to implement techniques that are more flexible and responsive. Investors may enhance the alignment of their portfolios with their risk tolerance and financial goals by utilizing algorithms such as convolution neural networks, gradient boosting machines, and LSTM. These models have the ability to analyse large quantities of data in order to determine the most effective distribution of assets and make immediate modifications in reaction to changes in the market.

The aim of this research is to employ machine learning algorithms and neural networks to forecast stock prices, particularly inside the FTSE 100 index, and enhance stock portfolios. The FTSE 100 is a stock market index that monitors the performance of the 100 largest companies listed on the London Stock Exchange. It functions as a standard for evaluating the efficiency of major sectors in the United Kingdom. Accurate forecasting in this index holds the capacity to yield significant benefits for investors, enabling them to make educated decisions and optimize their earnings.

Another essential aspect of this work is integrating technical indicators, such as Moving Averages, Relative Strength Index (RSI), and Moving Average Convergence Divergence (MACD), into these models. Traders frequently employ these indicators to analyse price patterns and market momentum. This work investigates whether the incorporation of these markers enhances the predictive capacity of machine learning models, leading to more accurate forecasts of stock returns.

1.2. Research Questions:

To guide the investigation, this project addresses the following key research questions:

1. **Which machine learning algorithms are the most efficient at forecasting stock values in the FTSE 100?**
   The project's objective is to assess and contrast the effectiveness of different machine learning algorithms, such as neural networks, in forecasting stock values inside the FTSE 100 index.

2. **Does the use of technical indicators such as Moving Averages, RSI, and MACD enhance the forecasting accuracy of machine learning models for stock returns?**
   This inquiry investigates the influence of incorporating widely used technical indicators into machine learning models and evaluates their efficacy in enhancing forecast accuracy.

1.3. Project objectives:

The main objectives of this research are:

1. **Construct a machine learning model that utilizes Yahoo Finance data to forecast stock prices from different sectors and optimize stock investments.**
   The objective is to construct resilient models capable of precisely forecasting stock values across different industries encompassed in the FTSE 100, thereby facilitating the enhancement of stock portfolios.

2. **Investigate the impact of different technical indicators on the capabilities which impact the prediction of machine learning models.**
   This objective is to examine the impact of including technical indicators on the efficacy of machine learning models in forecasting stock movements.

Essentially, this investigation seeks to utilize machine learning and neural networks to enhance the accuracy of forecasting stock market patterns, particularly in relation to the FTSE 100 index. The research aims to analyse the significance of technical indicators in improving model accuracy and optimizing portfolios.

2.LITERATURE REVIEW:

In recent years, there has been considerable interest in using machine learning (ML) and neural networks (NN) to anticipate stock market trends and optimize investment portfolios in the financial markets. Various research has investigated the efficacy of various algorithms, the incorporation of technical indicators, and the enhancement of investing techniques. This section provides an overview of the main scientific contributions in these domains.

Boosting the Accuracy of Stock Market Prediction using XGBoost and Long Short-Term Memory. [1]

In this paper, Gumelar, A.B. et al. (2020) investigates the enhancement of stock market prediction accuracy using two advanced machine learning algorithms: XGBoost and Long Short-Term Memory (LSTM). The study aims to forecast the closing values of stocks from 25 businesses listed on the Indonesia Stock Exchange (IDX). The researchers employed historical stock market data from 2000 to 2019 to conduct a comparative analysis between XGBoost, a tree-based algorithm renowned for its resilience in classification and regression tasks, and LSTM, a type of Recurrent Neural Network (RNN) well-suited for time series forecasting because of its capacity to learn long-term dependencies.

The results of the study revealed that XGBoost outperformed LSTM in terms of prediction accuracy, achieving a 99% accuracy rate. The study highlights the significance of using these algorithms for stock market predictions, noting that XGBoost's superior performance could be attributed to its gradient boosting framework, which effectively reduces prediction errors by combining weak learners into a stronger model. The paper concludes by emphasizing the potential of these techniques in guiding investment strategies and suggests further exploration into hybrid models for even greater accuracy.

The findings are crucial as they offer evidence of the effectiveness of combining traditional time series models with modern machine learning techniques to improve predictive outcomes.

Stock Movement Prediction Based On Technical Indicators Applying Hybrid Machine Learning Models.[2]

In this paper, Zouaghia et al. (2023) propose a hybrid machine learning framework to predict stock market movements using historical data from the NASDAQ index and technical indicators. The authors utilize five machine learning classifiers, namely Gaussian Naive Bayes, Random Forest, Gradient Boosting, Support Vector Machine, and K-Nearest Neighbours. Each classifier is combined with Principal Component Analysis for feature selection and Grid Search for hyperparameter optimization. The study spans from 2018 to 2023 and determines that the Random Forest model attains the best level of accuracy (61%) in forecasting stock price changes. The study emphasizes the capacity of the model to assist investors in making well-informed decisions during moments of market instability, such as those triggered by the COVID-19 epidemic and the Russia-Ukraine war. The authors propose that their models' accuracy might be enhanced by the implementation of additional optimization approaches, such as Bayesian Optimization and Convolutional Neural Networks.

Technical Analysis Indicators in Stock Market Using Machine Learning: A Comparative Analysis.[3]

In this paper Yash K. Pardeshi et al. (2021) investigates the effectiveness of various technical indicators and machine learning models in predicting stock price movements for high-volume stocks. The authors focus on the unpredictability of stock markets due to their inherent volatility, citing that most traders lose money due to a lack of systematic trading strategies. To address this issue, the study applies technical analysis indicators like the Relative Strength Indicator (RSI), Moving Average Convergence Divergence (MACD), Exponential Moving Averages (EMA), and Heiken Ashi candlesticks, among others, in conjunction with machine learning models like Decision Trees, Random Forest, Naïve Bayes, and K-Nearest Neighbour (KNN).

The research analyses data spanning five years (2015-2019) for 25 highly traded stocks in the Indian market, such as Reliance, Wipro, and HDFC Bank. Each indicator's effectiveness is measured based on its ability to predict short-term market movements and generate profitable trading signals. The study finds that the combination of technical indicators with machine learning models generally yields higher accuracy in predictions compared to using technical indicators alone. For instance, the combination of MACD and RSI together outperformed individual indicators, while machine learning models like Random Forest showed superior performance in predicting stock movements in large datasets.

The paper concludes that integrating traditional technical analysis with advanced machine learning algorithms can significantly enhance the reliability of stock market predictions, thereby potentially reducing losses and increasing profitability for traders. The findings emphasize the importance of choosing the right combination of indicators and models tailored to specific market conditions.

3.METHODOLOGY

3.1. Brief Overview:

This research employs neural networks and machine learning algorithms to predict stock market movements and optimize investment portfolios. The technique includes data extraction, exploration, pre-processing, technical parameter implementation, model training, and assessment. LSTM, CNN, and XGBoost are the main models. Each model analyses past stock market data and predicts future trends, using technical indicators like RSI, Moving Averages, and MACD to improve feature engineering and model performance.

3.2. Data extraction:

The data extraction process involves gathering historical stock market data and relevant financial metrics. Data sources include financial databases such as Yahoo Finance, Google Finance, and APIs providing real-time market data. Key data points include stock prices (open, close, high, low), trading volume, and other market indicators.

In this analysis, the data was extracted using the Yahoo Finance API, which provides a comprehensive dataset of historical stock market data. A total of 17 different stocks were selected from various sectors to ensure a diversified dataset for analysis. The selected stocks include:

S. No	Stock Name	Ticker Symbol	Sector
1	Barclays PLC	BARC	Finance
2	Lloyds Banking group PLC	LLOY	Finance
3	NatWest Group PLC	NW	Finance
4	Phoenix Group Holdings PLC	PHNX	Finance
5	Prudential PLC	PRU	Finance
6	HSBC Holdings PLC	HSBA	Finance
7	Standard chartered PLC	STAN	Finance
8	AstraZeneca PLC	AZN	Chemical
9	Croda International PLC	CRDA	Chemical
10	GlaxoSmithKline PLC	GSK	Chemical
11	Hikma Pharmaceuticals PLC	HIK	Chemical
12	Smith & Nephew PLC	SN	Chemical
13	JD Sports Fashion PLC	JD	Retail
14	Marks and Spencer Group PLC	MKS	Retail
15	Next PLC	NXT	Retail
16	Sainsbury PLC	SBRY	Retail
17	Tesco PLC	TSCO	Retail

Table1: List of Stocks analysed

The Yahoo Finance API was accessed using the 'yfinance' Python library. This library allows for the retrieval of historical data for any stock listed on major exchanges. For each selected stock in the table 1, data was downloaded covering a period from January 1, 2012, to December 31,2022, providing over a decade of historical data.

3.2.1. Data Collected:

Stock Prices: This dataset provides daily data for the Open, High, Low, Close, and Adjusted Close prices. Trading Volume refers to the total number of shares that are bought and sold on a daily basis.

3.2.2. Data Format:

The data for each stock was saved in a CSV format, with each file corresponding to one stock. The CSV files were then imported into data frames using the 'pandas' library for further analysis. Here's a snapshot of what the data for one of the stocks (e.g., Apple Inc. - AAPL) might look like:

Date	Open	High	Low	Close	Adj Close	Volume
2012-01-03	167.1978	172.3246	164.6113	172.0936	120.8791	54061195
2012-01-04	170.9851	173.9873	169.2531	173.9873	122.2093	40318622
2012-01-05	175.5115	176.6477	168.8605	169.6457	119.1598	52124639
2012-01-06	169.5071	175.5244	169.0453	172.186	120.9441	33288254
2012-01-09	171.4008	174.5877	163.2719	164.5189	115.5587	51055384

Table 2: Snapshot of data for the Barclays stock

Post extraction, the data was checked for consistency and completeness. Missing values and outliers were identified and addressed during the data pre-processing stage. From table 2, The data was organized into separate data frames for each stock, ensuring a clean and structured format ready for exploratory analysis and further processing.

3.3. Data Exploration:

After extracting the historical stock market data for the selected 17 stocks, a thorough exploration was conducted to understand the data's characteristics, identify trends, and discover any anomalies. The key analyses performed include examining the monthly trading volume per stock, visualizing the stock closing prices over time, and analysing other relevant metrics.

3.3.1 Monthly Volume Traded Per Stock

To examine trading activity, the monthly trading volume for each individual stock was computed. The statistic in question provides insight into the level of trading activity for a company across several time periods, which may serve as an indicator of investor interest. The 'pandas' function was used to aggregate the daily trading volume for each company monthly. A graph was created to illustrate the fluctuations and recurring patterns in trading activity by plotting the monthly trading volume. A line plot was generated for each stock, illustrating the monthly trading volume as it changes over time.

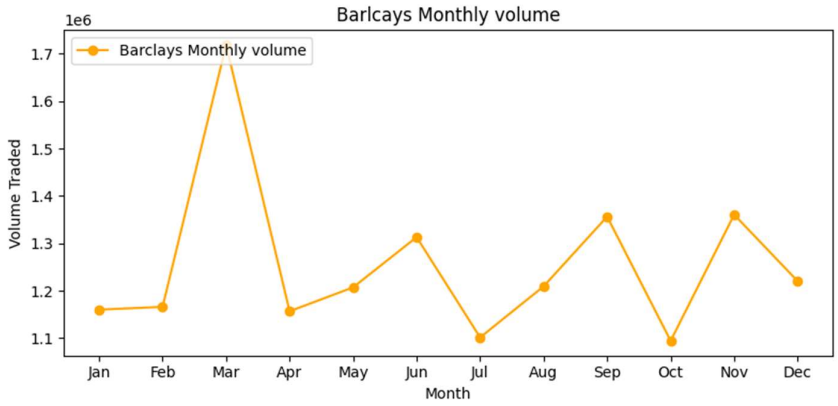

Figure 1: Monthly Volume traded for the Barclays stock over 10 years

Figure 1 illustrates the fluctuation in trade volume, exhibiting prominent spikes at moments of heightened market activity, such as earnings announcements or significant market events.

3.3.2 Visualization of Stock Closing Prices

One of the best measures of stock performance is closing price. Stock closing prices were shown over time to discover patterns. To track price changes, line charts were used for daily closing prices.

Figure 2 depicts AstraZeneca's stock price rise over time, with moving averages showing the long-term trend.

Figure 2: Close Price of the AstraZeneca stock over the 10 years

3.3.3 Descriptive Statistics

In summary, we computed descriptive statistics, including the mean, standard deviation, and range, for the closing prices and trading volumes of each stock in Table 3. Mean Closing Price is the arithmetic mean of the closing prices for the full time. Standard Deviation is a statistical metric that quantifies the degree of variability or volatility in the price of an asset. Range refers to the extent of price fluctuation in a financial market, specifically the gap between the highest and lowest closing prices.

Stock	Mean Close	Standard Deviation	Min Close	Max Close

BARC	156.15	30.3	69.3	225.60
LLOY	43.48	10.32	17.8	63.23
PRU	1013.16	260.10	374.02	1480.33

Table 3: Statistical description of the various stocks

3.3.4 Correlation Analysis

Gaining a comprehension of how several equities correlate with one another can offer valuable insights into market dynamics and techniques for diversification. A correlation matrix was constructed to analyse the interrelationships among the various aspects of a stock.
A heatmap in Figure 3 was produced to visually represent the correlation coefficients, highlighting the important features that exhibit positive or negative movement and aid in model training.

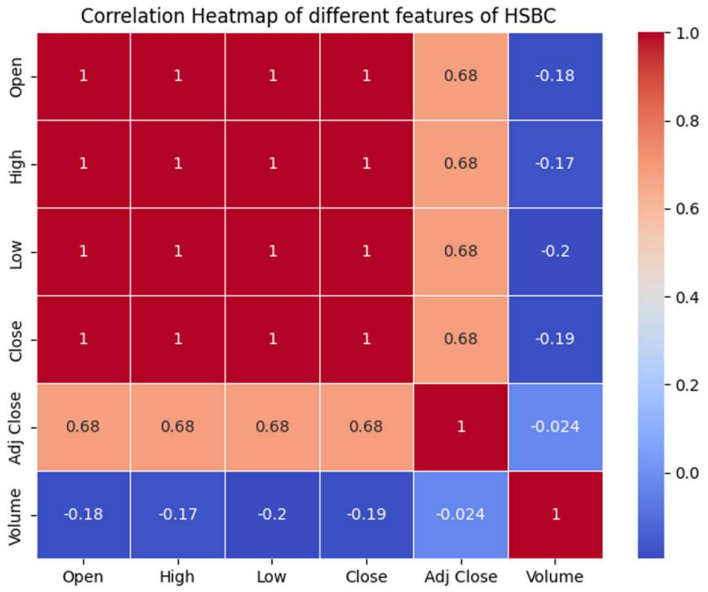

Figure 3: Correlation of the different features of HSBC stock

3.3.5. Volatility Analysis:

Financial market volatility research is essential. Understanding investing risk is helpful. Calculating the rolling standard deviation of closing prices in your project shows how much the stock price changes over time. Figure 4 shows considerable and minimum price swings by graphing rolling volatility. This approach helps detect stable and volatile stock periods. Volatility studies can inform trading strategy. We graphed Lloyds' 20-day rolling volatility and closing prices.

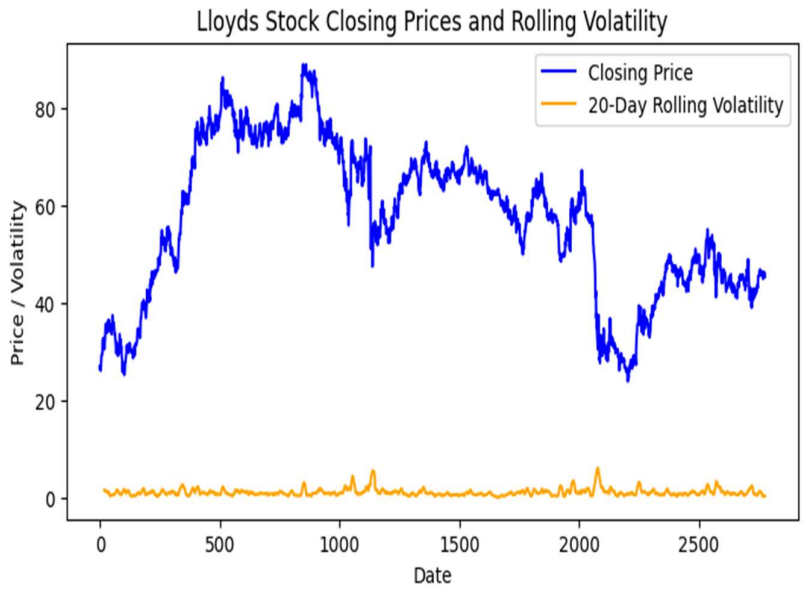

Figure 4: Rolling Volatility and close price of Lloyds Stock

3.4. Data Pre-processing:

Data pre-processing is a crucial step in preparing the dataset for training machine learning models. It ensures that the data is clean, consistent, and in a format that can be effectively used by the algorithms.

3.4.1 Data Scaling:

Machine learning models, particularly those involving neural networks, often require that input data be scaled to ensure that features contribute equally to the model's learning process. In this project, 'Min-Max Scaling' was applied to normalize the data.

Min-Max Scaling:

The goal of Min-Max Scaling is to transform the data to a fixed range, usually [0, 1], to ensure that all features have the same scale, making the training process more stable and efficient.

Formula: The Min-Max Scaling formula used was:

$$X' = \frac{X - Xmin}{Xmax - Xmin}$$

Where X' is the scaled value, X is the original value, Xmin is the minimum value in the dataset, and Xmax is the maximum value. The scaling was applied across all numerical features, including stock prices (Open, High, Low, Close), trading volumes, and technical indicators such as RSI, Moving Averages and MACD.

3.4.2. Data Preparation for Model Training

After scaling the data, model training preparation began. Data must be organized so machine learning algorithms could analyse it effectively. RSI, Moving Averages, and MACD improved the model's prediction. Time series models like LSTM developed lag characteristics by transferring stock price data numerous time steps. This helped the model learn from past data.

3.5. Technical parameters used:

This study used technical indicators to capture key stock price behaviour and enhance the model's market prediction. RSI, MACD, and SMA are the chosen indicators. These indicators are widely used in technical analysis to track market momentum, trends, and reversals.

3.5.1. Relative Strength Index (RSI):

Technical indicator RSI measures price variation velocity and magnitude. The Relative Strength Index (RSI) measures stock overbought or oversold conditions from 0 to 100.
A mathematical formula calculates RSI:

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

For RS (Relative Strength), divide the average of 'n' days up closes by the average of 'n' days down closes. Usually, 14 one-day sessions are used. The 14-day Relative Strength Index (RSI) is computed by dividing the average 14-day gain by the average loss. Based on RS, RSI is determined.

- Stocks are overbought when the RSI is over 70. These findings show the firm may be overpriced and may shortly fall.
- If the RSI falls below 30, the stock is oversold. The firm may be inexpensive and might rise in price. Figure 5 illustrates that the machine learning models contained the Relative Strength Index (RSI) for each stock in the dataset.

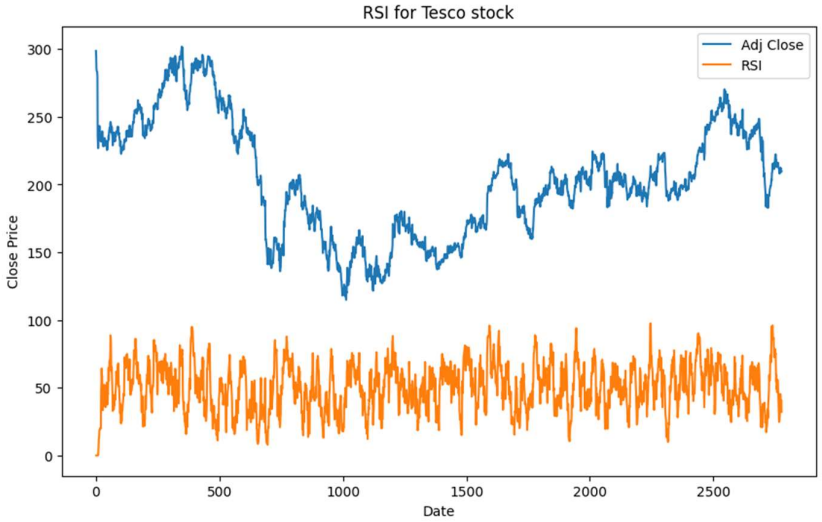

Figure 5: Relative Strength Index (RSI) for Tesco Bank

3.5.2 Moving Average Convergence Divergence (MACD)

Technical analysis uses MACD to evaluate a stock's momentum and trend-following. It compares two moving averages. This tool seeks to detect pattern magnitude, direction, velocity, and duration changes. MACD is calculated by subtracting the 26-day EMA from the 12-day EMA.

$$MACD = EMA_{12} - EMA_{26}$$

- MACD's 9-day Exponential Moving Average (EMA) is the signal line. On top of the MACD line, it triggers buy and sell signals.
- Figure 6 shows that when the Moving Average Convergence Divergence (MACD) is bigger than the Signal Line, it may be a good time to buy. MACD crossing below the signal line is a bearish indicator, recommending selling.

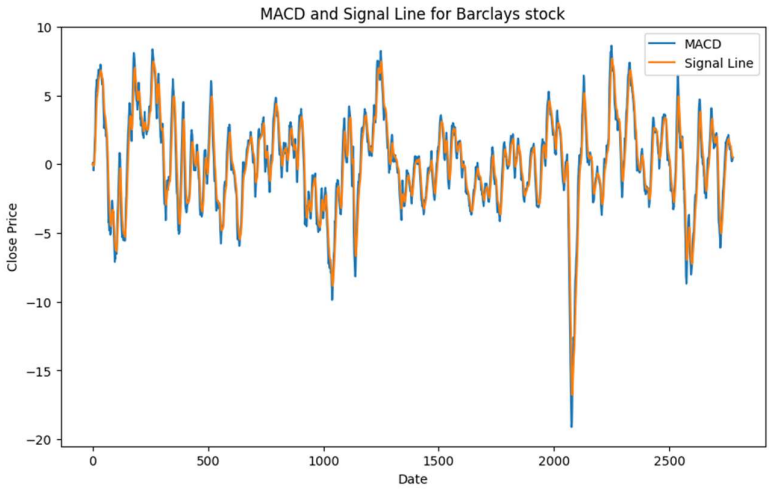

Figure 6: MACD and Signal Line for Barclays stock

3.5.3 Moving Averages (SMA):

Moving Averages are mathematical tools that employ computations to decrease the volatility of price data, so producing a trend-following indicator. They help reduce the impact of price volatility and provide a clearer view of the trend's direction.

Simple Moving Average (SMA): The Simple Moving Average (SMA) is determined by taking the average of a stock's price over a designated number of time periods.

$$SMA = \sum \frac{Price\ over\ \ n\ periods}{n}$$

In figure 7, a 20-day Simple Moving Average (SMA) computes the mean value of the closing prices over the previous 20 days.

Figure 7: Moving Averages for Tesco Stock

3.6. Machine Learning Models:

3.6.1. Long Short-Term Memory (LSTM):

Long Short-Term Memory (LSTM) is a popular artificial recurrent neural network (RNN) architecture in deep learning. Long Short-Term Memory (LSTM) networks utilize temporal correlations in input sequences via feedback connections, unlike feedforward neural networks. LSTM was created to address disappearing or bursting gradients in sequential RNN training. These traits make them ideal for sequential data tasks like NLP, speech recognition, and time series forecasting.

3.6.1.1. LSTM Architecture:

LSTM networks utilize memory cells that have the ability to store information over long periods. Each memory cell consists of three main components: an input gate, a forget gate, and an output gate. These gates are used to regulate the flow of data into and out of the memory cell.

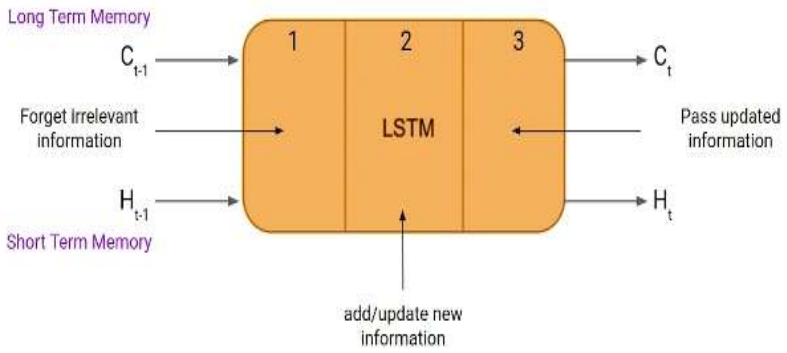

Figure 8: Simple architecture of LSTM [4]

Figure 8 shows the internal architecture of a Long Short-Term Memory (LSTM) cell, a key component of an LSTM network, a kind of Recurrent Neural Network. LSTM cells preserve and change memory cells over time to store longer associations in sequential data like time series data.

3.6.1.2. Components of the LSTM Cell:

- **Forget Gate:** This gate determines which information should be excluded from the cell state 'C_{t-1}'. The model receives the previous short-term memory 'H_{t-1}' and the current input, performs computations on both, and produces a numerical output ranging from 0 to 1 for each number in the cell state. A score of 1 indicates perfect retention, whereas a value of 0 signifies utter disregard.
- The output of the forget gate is computed as follows:

$$f_t = \sigma\left(W_f \cdot [H_{t-1}, x_t] + b_f\right)$$

- where σ is the sigmoid function, W_f is the weight matrix, H_{t-1} is the previous hidden state, x_t is the current input, and b_f is the bias.
- **Input Gate:** The input gate determines the selection of fresh information to be stored in the cell state. It is composed of two components: A sigmoid layer is utilized to determine which values should be updated, namely the i_t.A tanh layer that

generates a vector of novel candidate values, denoted as C'_t, that can be included into the state.

- The input gate operations consist of:

$$i_t = \sigma(W_i \cdot [H_{t-1}, x_t] + b_i)$$
$$C'_t = tanh(W_c \cdot [H_{t-1}, x_t] + b_c)$$

- The cell state C_t is then updated using:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot C'_t$$

- **Output Gate:** This gate decides what the next hidden state H_t should be. The hidden state contains the short-term memory and is also the output of the LSTM cell at the current time step. The output is based on the cell state, but it's a filtered version of the cell state.
- The output gate is calculated as:

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

- The final hidden state is:

$$H_t = o_t \cdot tanh(C_t)$$

- Long-Term Memory (Cell State 'C_t'): The cell state carries the long-term dependencies across time steps. The forget gate determines how much of the previous cell state C_{t-1} is retained or forgotten, and the input gate updates the cell state with new information.
- Short-Term Memory (Hidden State 'H_t'): The hidden state is updated at each time step and is used for the next step's computations. It also serves as the output of the LSTM cell at that time step.

3.6.2. Convolutional Neural Networks:

A Convolutional Neural Network (CNN), or ConvNet, analyses grid-like data like photographs. CNNs are usually used for image processing, although they can handle time series data like market prices. The main concept is to treat time series data as images: organized, grid-like data with patterns and characteristics.

3.6.2.1. Convolutional Neural Networks Architecture:

- **Input Data:** CNNs receive time series of stock prices or other financial information. Daily closing prices, trade volume, moving averages, and other technical indicators are examples. CNNs can find patterns over time since the data is organized. You might input the network 50 days of stock prices (or associated indicators) to estimate the 51st day's price.
- **Convolutional Layer:** The convolutional layer extracts feature from input data using kernel filters. After sliding across the input data, these filters convolutionally multiply the filter values by the input data values and total the results. CNN filters recognize local time series patterns like short-term stock price trends or periodic patterns. The kernel size influences the convolution window. For instance, a kernel size of 2 may analyse 2 days of stock values.

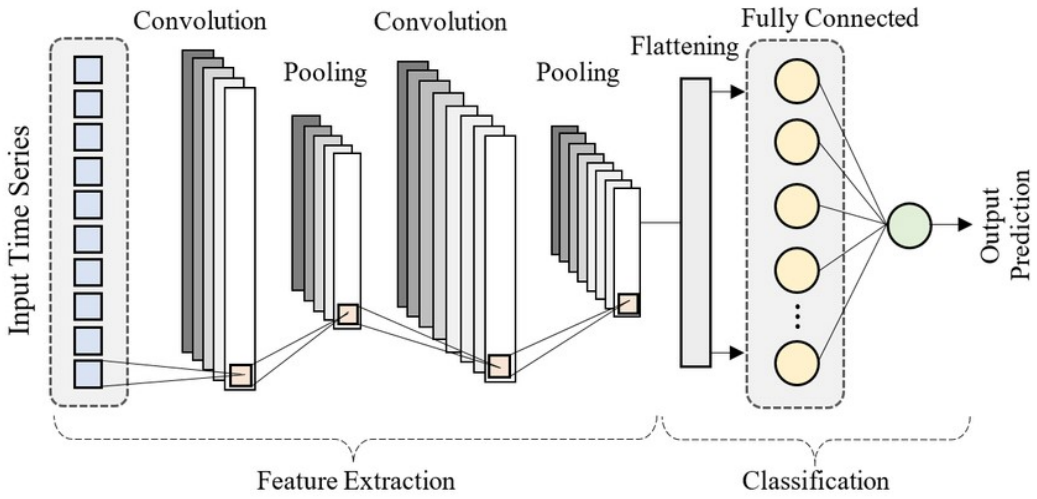

Figure 9: CNN architecture for time series Application.[6]

- **Pooling Layer:** Pooling layers minimize feature map dimensionality, maintaining key information and removing less significant features. Max pooling, which chooses the largest value in each feature map region, is widespread. This helps identify key aspects (e.g., major price movements) and simplifies computation. Setting the pooling size to 2 would down-sample the feature map by 2.
- **Flattening:** Multiple convolutional and pooling layers compress feature maps into a vector. This vector feeds the dense, completely linked layers. Flattening converts the 2D output from previous layers into a 1D vector for the dense layers that forecast the result.
- **Fully Connected (Dense) Layers:** Dense layers are typical neural network layers with every neuron coupled to the preceding layer. To create the final prediction, these layers combine convolutional layer information. Dense layers analyse collected information to anticipate the following day's stock price, price direction, or other target variable.
- **Output Layer:** Final prediction is from the output layer. This might be a single number (e.g., the following day's closing price) or a probability distribution across alternative outcomes (the price will rise or fall) for stock market prediction. The algorithm may analyse previous pricing data to identify patterns that may predict future price changes.

3.6.3. eXtreme Gradient Boosting Machines:

Powerful and extensively used machine learning technique XGBoost (Extreme Gradient Boosting) excels at structured/tabular data tasks like stock market prediction. Gradient boosting is fast and efficient using XGBoost. Machine learning approach gradient boosting trains new models to remedy prior model mistakes. XGBoost can simulate complicated associations between input characteristics (technical indicators, historical prices) and the target variable (future stock prices, price direction).

3.6.3.1. XGBoost Architecture:

While XGBoost does not possess the layered structure of CNNs or LSTMs, it can nevertheless be comprehended in terms of its components and operational principles, despite not being a neural network.

Input Features:

The XGBoost model takes as input a range of information derived from the stock data, including historical prices, technical indicators (such as RSI, MACD, Moving Averages), trading volume, and even external elements like market indices or economic indicators.

Decision Trees:

XGBoost is mostly composed on a series of decision trees. Every tree is trained with the objective of minimizing the mistakes (residuals) made by the preceding trees. Trees are incrementally included until either no substantial enhancement is detected, or a certain number of trees is attained.

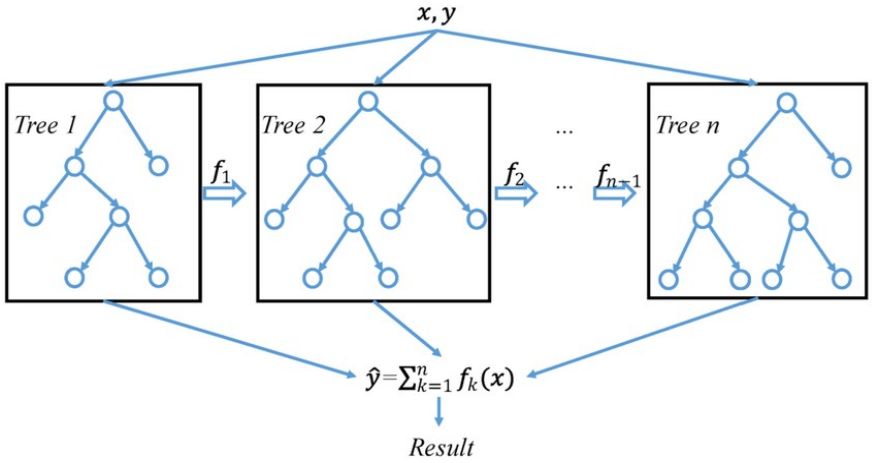

Figure 10: A general architecture of XGBoost [7]

Objective Function:

XGBoost optimizes a regularized objective function, which consists of a loss function (e.g., mean squared error for regression tasks) and regularization terms. The objective function is minimized using gradient descent, which adjusts the weights of the trees to reduce the residual errors.

Boosting Process:

The boosting method entails training each tree to rectify the faults made by the preceding ones. Following the addition of each tree, the forecasts are updated by including the contributions made by the new tree.

Output:

A weighted total of all ensemble tree forecasts is the final result. This might be a continuous value for stock market forecast or a probability score for price increase/decrease.

Predicting stock prices or trends with XGBoost is strong. It excels with structured data and financial data with various characteristics.

Each model's performance is evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared score. A comparative analysis helps determine the most effective model for forecasting stock market trends and optimizing investment portfolios.

4.IMPLEMENTATION AND RESULTS

The process is conducted systematically, beginning with the loading of data, followed by data pre-processing, model assessment, prediction, and finally, visualizing the outcomes. A data pretreatment pipeline was implemented to efficiently handle tasks such as scaling and encoding, ensuring consistency across different models. The models are trained and evaluated using many performance measures.

4.1. Loading the dataset for processing:

Historical stock prices for the 17 equities with tickers in Table 1 were downloaded into CSV files for training and testing separately. Each CSV file is turned into a panda 'DataFrame' for processing and model training. We calculated missing data, numerical values, and other pre-processing processes in chapter 3. Train CSV files contain Date, Open, High, Low, Close, Adj Close, and Volume columns for 10 years from January 2012 to December 2022. Each test CSV file has the same columns from January 1, 2023, until July 31, 2024.
The 'Date' column has been converted into Pandas 'DateTime' format for the processing of the monthly traded volume of the stocks. To make the dataset more streamlined, the initial Date Monthly traded columns were later removed.

4.2. Preparing data for training and testing:

4.2.1. Prepare data for LSTM and CNN:

Structuring data for an LSTM and CNN requires structuring it so they can efficiently ingest and process it. Recurrent neural networks like LSTMs are good for sequential data processing. Convolutional Neural Networks also handle sequence data well. The basic input data preparation steps for both models are:

- Sequences: Both models are capable of processing data in a sequential manner. It is necessary to arrange our incoming data into sequences of a predetermined length. To obtain stock market data, specifically daily data, it is necessary to generate input sequences consisting of 50 days' worth of data.
- Reformat the data into a 3D structure with dimensions (samples, time steps, features) to enable the LSTM to perceive it as sequential data. Samples: Total number of data points in your dataset. Time Steps: The quantity of time steps in each sequence. Features: Number of features present at each time step.

4.2.2. Prepare data for XGBoost:

Your data must be well-structured 2D to use XGBoost properly. The dataset should have rows for training samples and columns for features.
The target variable is the model's predicted outcome. In my XGBoost project, "closing price" is a stock's value at the conclusion of a trading day. Stock prices are often estimated using this number for the next day or other future time. It should stand out from features. The dependent variable usually has its own column. Avoid using this column as a predictor in XGBoost models while preparing data. Separating data prevents data leakage, which happens when the model accidentally acquires access to future information during training.

4.3. Training and testing Data:

4.3.1. Processing of models without technical parameters:

- The training and test datasets were divided into two parts - features (X) and the target parameter (y).
- Each dataframe in train contains columns such as Open, High, Low, Close, Adj Close, Volume for 10 years data from 01, Jan 2012 to 31, Dec 2022 and each dataframe in test have the same columns from 01, Jan 2023 to 31, Jul 2024.
- The target variable, 'Adj Close', which is a Closing price column and was separated from the rest of the columns, which were used as features.
- A preprocessing was established for numeric features, which involved scaling and standardising features.
- **Features used to Train Model:** Open, High, Low, Close, Volume.

4.3.2. Processing of models with technical parameters.

- The train dataset consists of dataframes with technical parameters such as RSI, MACD, and Moving averages. Each dataframe includes information on Open, High, Low, Close, Adj Close, Volume, MA 10 days, MA 20 days, RSI, MACD, and Signal line. The data spans a period of 10 years, from January 1, 2012, to December 31, 2022. Similarly, the test dataset also contains data frames with the same parameters, covering the period from January 1, 2023, to July 31, 2024.
- A preprocessing step was included for numeric features, which included scaling and standardizing the features.
- **Features used to Train Model:** Open, High, Low, Close, Volume, MA 10 days, MA 20 days, RSI, MACD, and Signal line

4.4. Model training:

After doing an analysis using several time series models, we have identified three models that yielded the best metrics. The detailed methodology of these models is presented in chapter 3. We have employed Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Extreme Gradient Boosting (XGBoost) algorithms.

1. Long Short-Term Memory (LSTM):

Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) are designed to analyse sequential input. This makes it ideal for stock price time series analysis. Stock prices are time-dependent because prior prices impact them. LSTM is useful for anticipating long-term stock market movements since it can preserve and use prior time steps. Traditional recurrent neural networks (RNNs) struggle with prolonged sequences due to the vanishing gradient problem. Memory cell structures allow long short-term memory (LSTM) networks to avoid this difficulty. LSTM models are great at capturing temporal dynamics and patterns in data, making them ideal for forecasting stock values using previous price movements and other time-dependent properties.

2. Convolutional Neural Networks (CNN):

Image processing often uses CNNs. They may also be used for time series data analysis to automatically extract and acquire important characteristics. CNNs can identify trends,

volatility, and cyclical behaviours in stock market data, which helps make accurate projections. CNNs can recognize spatially hierarchical data patterns. CNNs may identify time series data point associations. A CNN could analyse a week's worth of stock data to forecast price changes. Convolutional neural networks (CNNs) may discover patterns at various levels, from minute alterations to large-scale patterns, helping analyse market dynamics.

3. XGBoost:

XGBoost is a strong gradient boosting method for tabular data. The method creates a series of weak prediction models, generally decision trees, that correct each other's faults. This repetitive process enhances the model's prediction accuracy. XGBoost excels at collecting non-linear correlations, a common stock market phenomenon. It manages complex interactions that simpler models may miss. XGBoost is known for its computational efficiency and speed, making it ideal for large datasets or repetitive model training iterations like hyperparameter tuning. To reduce overfitting, XGBoost uses L1 and L2 regularization. Stock market prediction algorithms sometimes overfit to training data and perform poorly on unknown data.

Different model training parameters are:

Models	Parameters Used
Long Short-Term Memory	Units=120, return_sequences=True, input_shape= (50, 10) Units=80, return_sequences=True, Units=60, return_sequences=True
Convolution Neural Network	filters=128, kernel_size=2, activation='relu', input_shape=(50, 10) filters=64, kernel_size=2, activation='relu'
XGBoost	n_estimators=500,learning_rate=0.01,max_depth=10, min_child_weight=1,gamma=0,subsample=0.6,colsample_bytree=0.6, reg_alpha=0.001,reg_lambda=0.001,objective='reg:squarederror', eval_metric='rmse'

Table 4: Models and their Parameters

4.5. Metrics Used for evaluation:

4.5.1. Mean Absolute Error (MAE):

- Mean Absolute Error (MAE) quantifies the average size of mistakes in forecasts, regardless of their direction. The mean absolute difference between the predicted values and the actual observations is calculated by taking the average over the test sample.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- A lower Mean Absolute Error (MAE) value implies superior performance of the model.

4.5.2. Root Mean Squared Error (RMSE)

- RMSE, or Root Mean Square Error, is a mathematical measure that calculates the square root of the average of the squared discrepancies between a forecast and the actual observation. MAE is less susceptible to outliers than it.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- A lower root mean square error (RMSE) value suggests higher prediction accuracy.

4.5.3.R^2 Score

- The R^2 score, also known as the Coefficient of Determination, quantifies the degree to which the regression predictions accurately represent the actual data points. A value of 1 for R^2 signifies that the model accurately predicts the target variable, whereas a value of 0 shows that the model performs no better than a basic mean prediction.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- A higher R^2 value implies superior performance of the model.

4.6. Discussion of Results from Trained Models

I obtained separate efficiency metrics from three machine learning models: Long Short-term Memory, Convolution neural network and XGBoost. The results of each model are comprehensively evaluated, including their mean absolute error (MAE), root mean square error (RMSE), R-squared (R2). The performance measures are derived from both the training and testing datasets, offering a comprehensive understanding of the predictive capacity of each model in deciding prediction outcomes.

4.6.1. LSTM model Evaluation

The LSTM model produced a high level of accuracy and performance, with an average R2 score of 82.8%, a mean absolute error of 42.8, and a root mean square error of 53.05. These results were reached without the use of technical parameters, indicating the model's strong performance.

The inclusion of technical parameters in the LSTM model resulted in better accuracy and metrics, with an R2 score of 92.5%, a mean absolute error of 38, and a root mean square error of 46.8. The model can demonstrate the ability to make accurate future predictions.

4.6.2. CNN Model Evaluation

The algorithm demonstrated strong performance in accurately predicting the closing price of stocks. The model achieved a mean absolute error of 76.75, a root mean square error of 90.7, and an R2 score of 65.5% without considering the technical parameters. However, these results do not meet the expected performance.

Considering the technical criteria, the r2 score increased to 85.2%, which is commendable. Additionally, the mean absolute error reduced to 54.5 and the root mean square error decreased to 67.9. This demonstrates the model's proficiency in precisely depicting the final price of the stocks.

4.6.3. XGBoost Model Evaluation

The XGBoost model achieved a high level of accuracy in forecasting the closing price of stocks using technical parameters. The acquired results include a mean absolute error of 94.6, a root mean square error of 119.7, and an R2 score of 54.2%. However, these results are not as intended as they do not take into account the technical factors. The performance has significantly improved, achieving an excellent R2 score of 92%, a mean absolute error of 30.03, and a root mean square error of 37.5.

Metrics	LSTM Model	CNN Model	XGBoost Model
Mean Absolute Error	42.8	76.75	94.6
Root Mean Squared Error	53.05	90.7	119.7
R-Squared	82.8%	65.5%	54.2%

Table 5: Models with their Score without technical parameters

Metrics	LSTM Model	CNN Model	XGBoost Model
Mean Absolute Error	38	54.9	30.03
Root Mean Squared Error	46.8	67.9	37.5
R-Squared	92.5%	88%	92%

Table 6: Models with their Score with technical parameters

4.6.4 Bar Plot with R-Squared Values of different models:

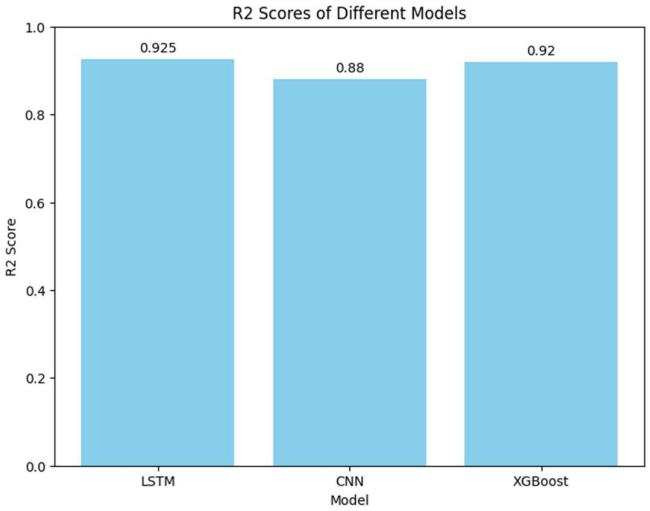

Figure 11: R2 score comparison.

The bar chart presents a comparison of the R-squared scores for three distinct time series models: LSTM, CNN, and XGBoost. The LSTM model has superior performance compared to the other models, achieving an R-squared score of 92.5%. This number indicates the highest level of accuracy in predicting the variability present in the data. The XGBoost model earns an accuracy score of 92%, which is slightly lower than that of the LSTM model. On the other hand, the CNN model obtains the lowest score of 88% among the three models. Figure 11 depicts a comparison of three models, highlighting the exceptional performance demonstrated by the LSTM in this forecast.

4.6.5. Hyperparameters tuning:

As shown previously, LSTM was found to be the most effective model based on its extraordinary scores. Hyperparameter adjustment was conducted to enhance performance and guarantee the model is correctly constructed for this purpose. After adjusting the hyperparameters, I obtained an R-squared value of roughly 89%, which is somewhat lower than the R-squared value achieved using the LSTM model with its default values.

The model is distinguished by its robustness and flexibility, and it consistently generates favourable outcomes when employing the default hyperparameters. Therefore, the model may attain its maximum efficiency for our dataset even without any modifications. The remarkable performance seen initially suggests that the default parameters were already very appropriate for our data, leaving limited room for improvement by customization.

4.7. Time Series Analysis for the stock prediction:

Time series analysis is a statistical method employed to examine data points arranged in chronological order, such as stock prices, with the aim of detecting patterns, trends, and other attributes that might aid in forecasting future values. Time series analysis in our project entails examining past stock prices, trading volumes, and other pertinent financial data over a period of time in order to predict future stock prices or patterns.

4.7.1. Actual vs Predicted

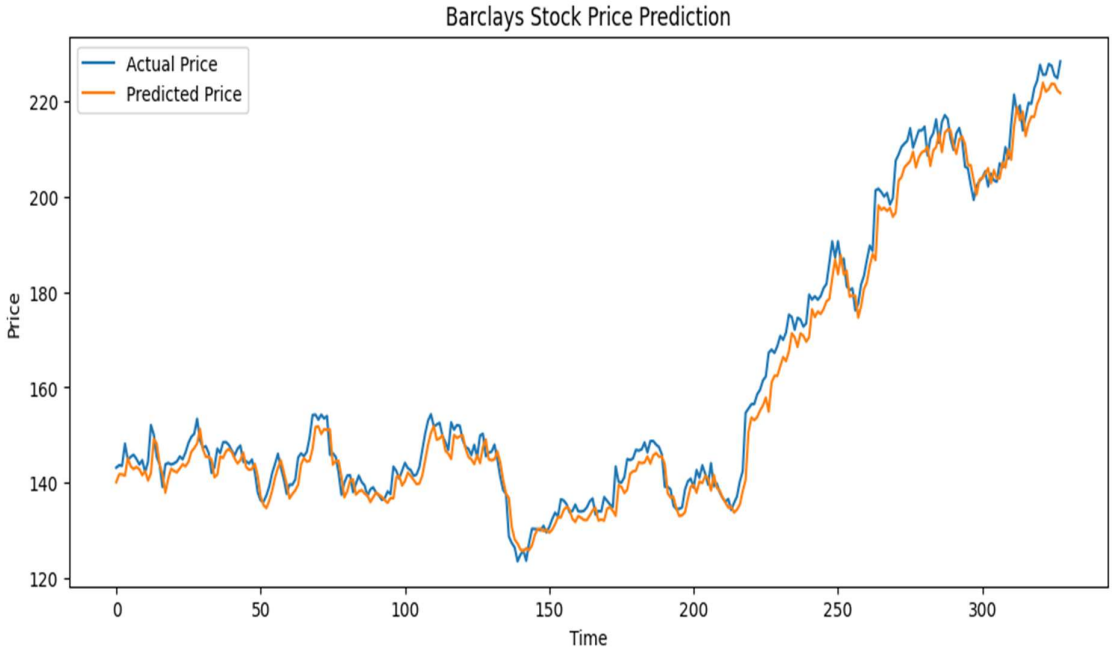

Figure 12: Time Series showing actual vs predicted values for Barclays stock closing price

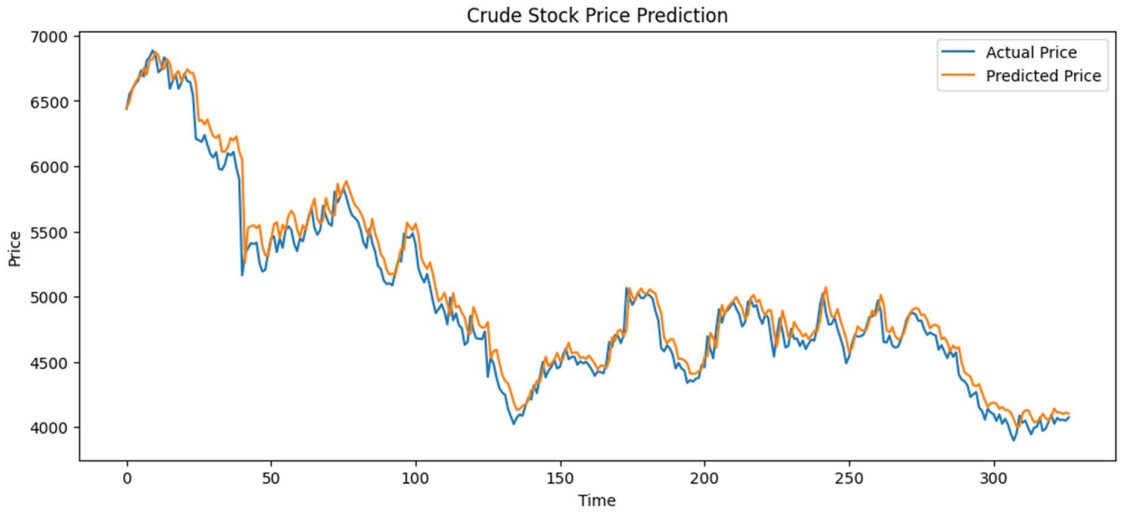

Figure 13: Time Series showing actual vs predicted values for Crude stock closing price

Figures 12 and 13 depict the time series that corresponds to the closing price of the stock. Figure 12 illustrates the Barclays stock exhibiting a discernible increasing trend, as precisely forecasted by the algorithm. Figure 13 shows that the crude stock had a negative or falling trend, which was precisely anticipated by the model.

- The blue line represents the real closing price of the stock for a span of 19 months.
- The orange line represents the projected closing price of the stock, as determined by a mathematical model.
- Figures 12 and 13 exhibit the model's capacity to faithfully reproduce the real-time stock closing price, as seen by the alignment of the orange line with the blue line.

5.CONCLUSION AND FUTURE WORK

5.1. Summary of the work:

In summary, I examined and applied a range of machine learning and deep learning models to forecast stock market patterns and enhance investment portfolios. Following an extensive investigation and examination of relevant literature, the models incorporated were Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNNs), and XGBoost. Each of these models contributed distinct advantages to the prediction procedure. By integrating technical indicators such as Relative Strength Index (RSI), Moving Averages (MA), and Moving Average Convergence Divergence (MACD), the models' forecasting capabilities were improved since they offered supplementary information regarding market conditions and trends. These indicators facilitated the capture of market mood and momentum, which are essential for precise forecasting. After doing training and testing, I determined that the LSTM model outperformed the others, reaching an R-Squared value of 92.5%. Ultimately, I used time series analysis to assess the models' capacity to forecast stock closing values as time progresses.

5.2. Conclusion:

The integration of machine learning algorithms with financial research has demonstrated substantial promise in forecasting stock market patterns and enhancing investment portfolios. Out of the three models, the LSTM model achieved a high performance of 92.5% using the default hyperparameters. Although the results show promise, the intricate characteristics of financial markets necessitate further enhancements. Future study might advance upon this project by integrating supplementary data sources, improving the comprehensibility of the model, and broadening the range of analysis. This will enable the exploration of new possibilities in financial forecasting and investment strategy.

5.3. Future Work:

Incorporation of Macro-Economic Indicators: Future study may entail the integration of macroeconomic data, such as interest rates, inflation, GDP growth, and unemployment rates, into the models. These factors frequently exert a substantial influence on market fluctuations and including them might enhance the precision of forecasts.

Sentiment Analysis: Incorporating a sentiment analysis module that evaluates news stories, social media content, and financial data might offer further insights. Through comprehending market sentiment, the models might enhance their ability to anticipate market responses to events, resulting in more precise forecasts.

Hybrid Models: Creating hybrid models that include the advantages of several machine learning methods might further improve the accuracy of predictions. An instance of a model that integrates the LSTM's proficiency in time series prediction with CNN’s capability to process structured data may provide a more resilient forecasting system.

Real-Time Predictions: Integrating real-time prediction skills would be a substantial improvement. This would need the integration of the models into a live data pipeline, where the most up-to-date market data is consistently inputted into the models, and projections are promptly updated.

6. REFERENCES

[1] A. B. Gumelar *et al*., "Boosting the Accuracy of Stock Market Prediction using XGBoost and Long Short-Term Memory," *2020 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Semarang, Indonesia, 2020, pp. 609-613, doi: 10.1109/iSemantic50169.2020.9234256.

[2] Z. Zouaghia, Z. Kodia Aouina and L. Ben Said, "Stock Movement Prediction Based On Technical Indicators Applying Hybrid Machine Learning Models," *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, Doha, Qatar, 2023, pp. 1-4, doi: 10.1109/ISNCC58260.2023.10323971.

[3] Y. K. Pardeshi and P. P. Kale, "Technical Analysis Indicators in Stock Market Using Machine Learning: A Comparative Analysis," *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 2021, pp. 1-6, doi: 10.1109/ICCCNT51525.2021.9580172.

[4]https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/[accessed 27 Aug 2024]

[5] Prospective Methodologies in Hybrid Renewable Energy Systems for Energy Prediction Using Artificial Neural Networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural network-CNN-model-for-time-series_fig9_349522240 [accessed 27 Aug 2024]

[6] A hybrid ensemble method for pulsar candidate classification - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-general-architecture-of-XGBoost_fig3_335483097 [accessed 27 Aug 2024]

[7] P. P. Kadu and G. R. Bamnote, "Comparative Study of Stock Price Prediction using Machine Learning," 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatre, India, 2021, pp. 1200-1204, doi: 10.1109/ICCES51350.2021.9489170. 3)

[8] B. Omar, B. Zineb, A. Cortés Jofré and D. González Cortés, "A Comparative Study of Machine Learning Algorithms for Financial Data Prediction," 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Rabat, Morocco, 2018, pp. 1-5, doi: 10.1109/ISAECT.2018.8618774.

[9] N. Parida, B. K. Balabantaray, R. Nayak and J. K. Rout, "A Deep Learning based Approach to Stock Market Price Prediction using Technical indicators," *2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS)*, Kochi, India, 2023, pp. 1-7, doi: 10.1109/AICAPS57044.2023.10074445.

[10] P. Oncharoen and P. Vateekul, "Deep Learning for Stock Market Prediction Using Event Embedding and Technical Indicators," *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, Krabi, Thailand, 2018, pp. 19-24, doi: 10.1109/ICAICTA.2018.8541310.

[11] A. Mathur, P. Fulwala, S. Lal and S. Relan, "A Comparative Study of Machine Learning Approaches for Stock Price Forecasting," *2023 Global Conference on Information Technologies and Communications (GCITC)*, Bangalore, India, 2023, pp. 1-6, doi: 10.1109/GCITC60406.2023.10426471.

[12] Y. Juwono, R. Sarno, R. N. E. Anggraini, A. T. Haryono and A. F. Septiyanto, "Comparative Study on Stock Price Forecasting Using Deep Learning Method Based on Combination Dataset," *2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, Bandung, Indonesia, 2024, pp. 1-6, doi: 10.1109/AIMS61812.2024.10513288.

8.APPENDIX

Code :

```
#Stock market Prediction
#!pip install yfinance

Import the libraries required

import numpy as np
import pandas as pd
import seaborn as sns
import yfinance as yf
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras import layers, models, optimizers, losses, callbacks,\
                    regularizers
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from xgboost import XGBRegressor

#Download Data for train

ftse100_tickers = [
   'BARC.L','LLOY.L', 'NWG.L', 'HSBA.L', 'STAN.L', 'PRU.L','PHNX.L',
   'AZN.L', 'CRDA.L','GSK.L','HIK.L', 'SN.L',
   'JD.L','MKS.L', 'NXT.L', 'SBRY.L', 'TSCO.L',
  ]
start_date = '2012-01-01'
end_date = '2022-12-31'
stocks_data = {}
for ticker in ftse100_tickers:
   print(f"Fetching data for {ticker}...")
   data = yf.download(ticker, start=start_date, end=end_date)
   stocks_data[ticker] = data

for ticker, data in stocks_data.items():
   data.to_csv(f'{ticker}_stock_data_train.csv')
   print(f"Data for {ticker} saved to {ticker}_stock_data_train.csv")


#Download data for Test
```

```
ftse100_tickers = [
   'BARC.L','LLOY.L', 'NWG.L', 'HSBA.L', 'STAN.L', 'PRU.L','PHNX.L',
   'AZN.L', 'CRDA.L','GSK.L','HIK.L', 'SN.L',
   'JD.L','MKS.L', 'NXT.L', 'SBRY.L', 'TSCO.L',
  ]
start_date = '2022-12-30'
end_date = '2024-07-30'
stocks_data = {}

for ticker in ftse100_tickers:
   print(f"Fetching data for {ticker}...")
   data = yf.download(ticker, start=start_date, end=end_date)
   stocks_data[ticker] = data

for ticker, data in stocks_data.items():
   data.to_csv(f'{ticker}_stock_data_test.csv')
   print(f"Data for {ticker} saved to {ticker}_stock_data_test.csv")

#Data preparation

data for financial stocks

barclays_train = pd.read_csv('BARC.L_stock_data_train.csv')
barclays_train1 = barclays_train.copy()
lloyds_train = pd.read_csv('LLOY.L_stock_data_train.csv')
lloyds_train1 = lloyds_train.copy()
natwest_train = pd.read_csv('NWG.L_stock_data_train.csv')
natwest_train1 = natwest_train.copy()
hsbc_train = pd.read_csv('HSBA.L_stock_data_train.csv')
hsbc_train1 = hsbc_train.copy()
stand_chartered_train = pd.read_csv('STAN.L_stock_data_train.csv')
stand_chartered_train1 = stand_chartered_train.copy()
prudential_train = pd.read_csv('PRU.L_stock_data_train.csv')
prudential_train1 = prudential_train.copy()
phoenix_train = pd.read_csv('PHNX.L_stock_data_train.csv')
phoenix_train1 = phoenix_train.copy()
astrazenica_train = pd.read_csv('AZN.L_stock_data_train.csv')
astrazenica_train1 = astrazenica_train.copy()
crude_train = pd.read_csv('CRDA.L_stock_data_train.csv')
crude_train1 = crude_train.copy()
glaxo_train = pd.read_csv('GSK.L_stock_data_train.csv')
glaxo_train1 = glaxo_train.copy()
hikma_train = pd.read_csv('HIK.L_stock_data_train.csv')
hikma_train1 = hikma_train.copy()
smith_train = pd.read_csv('SN.L_stock_data_train.csv')
smith_train1 = smith_train.copy()
jd_train = pd.read_csv('JD.L_stock_data_train.csv')
```

```
jd_train1 = jd_train.copy()
mks_train = pd.read_csv('MKS.L_stock_data_train.csv')
mks_train1 = mks_train.copy()
next_train = pd.read_csv('NXT.L_stock_data_train.csv')
next_train1 = next_train.copy()
sainsbury_train = pd.read_csv('SBRY.L_stock_data_train.csv')
sainsbury_train1 = sainsbury_train.copy()
tesco_train = pd.read_csv('TSCO.L_stock_data_train.csv')
tesco_train1 = tesco_train.copy()


#test data
barclays_test = pd.read_csv('BARC.L_stock_data_test.csv')
barclays_test1 = barclays_test.copy()
lloyds_test = pd.read_csv('LLOY.L_stock_data_test.csv')
lloyds_test1 = lloyds_test.copy()
natwest_test = pd.read_csv('NWG.L_stock_data_test.csv')
natwest_test1 = natwest_test.copy()
hsbc_test = pd.read_csv('HSBA.L_stock_data_test.csv')
hsbc_test1 = hsbc_test.copy()
stand_chartered_test =  pd.read_csv('STAN.L_stock_data_test.csv')
stand_chartered_test1 = stand_chartered_test.copy()
prudential_test = pd.read_csv('PRU.L_stock_data_test.csv')
prudential_test1 = prudential_test.copy()
phoenix_test = pd.read_csv('PHNX.L_stock_data_test.csv')
phoenix_test1 = phoenix_test.copy()
astrazenica_test = pd.read_csv('AZN.L_stock_data_test.csv')
astrazenica_test1 = astrazenica_test.copy()
crude_test = pd.read_csv('CRDA.L_stock_data_test.csv')
crude_test1 = crude_test.copy()
glaxo_test = pd.read_csv('GSK.L_stock_data_test.csv')
glaxo_test1 = glaxo_test.copy()
hikma_test = pd.read_csv('HIK.L_stock_data_test.csv')
hikma_test1 = hikma_test.copy()
smith_test = pd.read_csv('SN.L_stock_data_test.csv')
smith_test1 = smith_test.copy()
jd_test = pd.read_csv('JD.L_stock_data_test.csv')
jd_test1 = jd_test.copy()
mks_test = pd.read_csv('MKS.L_stock_data_test.csv')
mks_test1 = mks_test.copy()
next_test = pd.read_csv('NXT.L_stock_data_test.csv')
next_test1 = next_test.copy()
sainsbury_test = pd.read_csv('SBRY.L_stock_data_test.csv')
sainsbury_test1 = sainsbury_test.copy()
tesco_test = pd.read_csv('TSCO.L_stock_data_test.csv')
tesco_test1 = tesco_test.copy()
```

```
prudential_train.describe()

banks_train =
[barclays_train,lloyds_train,natwest_train,hsbc_train,stand_chartered_train,prudential_train
,phoenix_train]
banks_test =
[barclays_test,lloyds_test,natwest_test,hsbc_test,stand_chartered_test,prudential_test,pho
enix_test]
chemicals_train = [astrazenica_train,crude_train,glaxo_train,hikma_train,smith_train]
chemicals_test = [astrazenica_test,crude_test,glaxo_test,hikma_test,smith_test]
retail_train = [jd_train,mks_train,next_train,sainsbury_train,tesco_train]
retail_test = [jd_test,mks_test,next_test,sainsbury_test,tesco_test]
company_bank = ['Barclays', 'Lloyds', 'Natwest', 'HSBC', 'Standard Chartered', 'Prudential',
'Phoenix']
company_chemical = ['Astrazenica', 'Crude', 'Glaxo', 'Hikma', 'Smith']
company_retail = ['JD', 'MKS', 'Next', 'Sainsbury', 'Tesco']

barclays_train_target = pd.DataFrame()
lloyds_train_target =  pd.DataFrame()
natwest_train_target =   pd.DataFrame()
hsbc_train_target = pd.DataFrame()
stand_train_target = pd.DataFrame()
prudential_train_target =  pd.DataFrame()
phoenix_train_target =  pd.DataFrame()
astrazenica_train_target = pd.DataFrame()
crude_train_target = pd.DataFrame()
glaxo_train_target = pd.DataFrame()
hikma_train_target =  pd.DataFrame()
smith_train_target =  pd.DataFrame()
jd_train_target = pd.DataFrame()
mks_train_target  = pd.DataFrame()
next_train_target =  pd.DataFrame()
sainsbury_train_target = pd.DataFrame()
tesco_train_target = pd.DataFrame()

barclays_test_target =  pd.DataFrame()
lloyds_test_target = pd.DataFrame()
natwest_test_target =  pd.DataFrame()
hsbc_test_target = pd.DataFrame()
stand_test_target =  pd.DataFrame()
prudential_test_target =  pd.DataFrame()
phoenix_test_target =  pd.DataFrame()
astrazenica_test_target = pd.DataFrame()
crude_test_target = pd.DataFrame()
glaxo_test_target = pd.DataFrame()
hikma_test_target =  pd.DataFrame()
smith_test_target = pd.DataFrame()
```

```
jd_test_target =  pd.DataFrame()
mks_test_target = pd.DataFrame()
next_test_target =  pd.DataFrame()
sainsbury_test_target =  pd.DataFrame()
tesco_test_target = pd.DataFrame()

barclays_train_target1 =  pd.DataFrame()
lloyds_train_target1 = pd.DataFrame()
natwest_train_target1 =  pd.DataFrame()
hsbc_train_target1 = pd.DataFrame()
stand_train_target1 = pd.DataFrame()
prudential_train_target1 =  pd.DataFrame()
phoenix_train_target1 =  pd.DataFrame()
astrazenica_train_target1 = pd.DataFrame()
crude_train_target1 = pd.DataFrame()
glaxo_train_target1 = pd.DataFrame()
hikma_train_target1 =  pd.DataFrame()
smith_train_target1 =  pd.DataFrame()
jd_train_target1 = pd.DataFrame()
mks_train_target1  = pd.DataFrame()
next_train_target1 =  pd.DataFrame()
sainsbury_train_target1 = pd.DataFrame()
tesco_train_target1 = pd.DataFrame()

barclays_test_target1 =  pd.DataFrame()
lloyds_test_target1 = pd.DataFrame()
natwest_test_target1 =  pd.DataFrame()
hsbc_test_target1 = pd.DataFrame()
stand_test_target1 =  pd.DataFrame()
prudential_test_target1 =  pd.DataFrame()
phoenix_test_target1 =  pd.DataFrame()
astrazenica_test_target1 = pd.DataFrame()
crude_test_target1 = pd.DataFrame()
glaxo_test_target1 = pd.DataFrame()
hikma_test_target1 =  pd.DataFrame()
smith_test_target1 = pd.DataFrame()
jd_test_target1 =  pd.DataFrame()
mks_test_target1 = pd.DataFrame()
next_test_target1 =  pd.DataFrame()
sainsbury_test_target1 =  pd.DataFrame()
tesco_test_target1 = pd.DataFrame()

stocks_train =
[barclays_train,lloyds_train,natwest_train,hsbc_train,stand_chartered_train,prudential_train
,phoenix_train,
```

```
astrazenica_train,crude_train,glaxo_train,hikma_train,smith_train,jd_train,mks_train,next_tr
ain,sainsbury_train,tesco_train]
stocks_test =
[barclays_test,lloyds_test,natwest_test,hsbc_test,stand_chartered_test,prudential_test,pho
enix_test,

astrazenica_test,crude_test,glaxo_test,hikma_test,smith_test,jd_test,mks_test,next_test,sai
nsbury_test,tesco_test]

stocks_train_target =
[barclays_train_target,lloyds_train_target,natwest_train_target,hsbc_train_target,stand_trai
n_target,prudential_train_target,phoenix_train_target,

astrazenica_train_target,crude_train_target,glaxo_train_target,hikma_train_target,smith_tr
ain_target,

jd_train_target,mks_train_target,next_train_target,sainsbury_train_target,tesco_train_targe
t]

stocks_test_target =
[barclays_test_target,lloyds_test_target,natwest_test_target,hsbc_test_target,stand_test_ta
rget,prudential_test_target,phoenix_test_target,

astrazenica_test_target,crude_test_target,glaxo_test_target,hikma_test_target,smith_test_t
arget,

jd_test_target,mks_test_target,next_test_target,sainsbury_test_target,tesco_test_target]

stocks_name = ['Barclays', 'Lloyds', 'Natwest', 'HSBC', 'Standard Chartered', 'Prudential',
'Phoenix','Astrazenica',
        'Crude', 'Glaxo', 'Hikma', 'Smith', 'JD', 'MKS', 'Next', 'Sainsbury', 'Tesco']

stocks_train1 =
[barclays_train1,lloyds_train1,natwest_train1,hsbc_train1,stand_chartered_train1,prudentia
l_train1,phoenix_train1,

astrazenica_train1,crude_train1,glaxo_train1,hikma_train1,smith_train1,jd_train1,mks_train
1,next_train1,sainsbury_train1,tesco_train1]

stocks_test1 =
[barclays_test1,lloyds_test1,natwest_test1,hsbc_test1,stand_chartered_test1,prudential_tes
t1,phoenix_test1,

astrazenica_test1,crude_test1,glaxo_test1,hikma_test1,smith_test1,jd_test1,mks_test1,next
_test1,sainsbury_test1,tesco_test1]
```

```
stocks_train_target1 =
[barclays_train_target1,lloyds_train_target1,natwest_train_target1,hsbc_train_target1,stand
_train_target1,

prudential_train_target1,phoenix_train_target1,astrazenica_train_target1,crude_train_targe
t1,glaxo_train_target1,
                hikma_train_target1,smith_train_target1,jd_train_target1,mks_train_target1,
                next_train_target1,sainsbury_train_target1,tesco_train_target1]

stocks_test_target1 =
[barclays_test_target1,lloyds_test_target1,natwest_test_target1,hsbc_test_target1,stand_te
st_target1,

prudential_test_target1,phoenix_test_target1,astrazenica_test_target1,crude_test_target1,
glaxo_test_target1,
                hikma_test_target1,smith_test_target1,jd_test_target1,mks_test_target1,
                next_test_target1,sainsbury_test_target1,tesco_test_target1]


#Exploratory data analysis

years = ['2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022']
plt.figure(figsize=(8, 4))
plt.plot(astrazenica_train['Adj Close'], label='Astrazeneca Close')
plt.title('Astrazeneca Close Price')
plt.ylabel('Adj Close')
plt.xlabel('Years')
positions = range(0, len(astrazenica_train), len(astrazenica_train) // (len(years) - 1))
plt.xticks(positions, years)
plt.legend(loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()



for company in banks_train:
  company['Date'] = pd.to_datetime(company['Date'])
  company['Month'] = company['Date'].dt.month
  company['Monthly Avg'] = company['Volume'].groupby(company['Month']).mean()

plt.figure(figsize=(8,4))
plt.plot(company['Monthly Avg'], label='Barclays Monthly volume', marker='o', linestyle='-
',color='orange')
plt.title('Barlcays Monthly volume')
plt.ylabel('Volume Traded')
plt.legend(loc='upper left')
```

```
plt.xlabel('Month')
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov',
'Dec'])
plt.tight_layout()
plt.show()

barclays_train.head()

# Define the rolling window size (e.g., 20 days)
window_size = 20
for company in banks_train:
  company['Rolling Volatility'] = company['Adj Close'].rolling(window=window_size).std()

# Plot the closing prices and rolling volatility
plt.figure(figsize=(8, 4))
plt.plot(lloyds_train['Close'], label='Closing Price', color='blue')
plt.plot(lloyds_train['Rolling Volatility'], label=f'{window_size}-Day Rolling Volatility',
color='orange')
plt.title('Lloyds Stock Closing Prices and Rolling Volatility')
plt.xlabel('Date')
plt.ylabel('Price / Volatility')
plt.legend()
plt.show()


for company in banks_train:
  company.drop(columns=['Month','Monthly Avg','Rolling Volatility'],inplace=True)

#Data pre-processing

def calculate_rsi(data, period=14):
    delta = data['Adj Close'].diff()
    gain = (delta.where(delta > 0, 0)).fillna(0)
    loss = (-delta.where(delta < 0, 0)).fillna(0)

    avg_gain = gain.rolling(window=period, min_periods=1).mean()
    avg_loss = loss.rolling(window=period, min_periods=1).mean()

    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))

    return rsi

def calculate_macd(data, short_period=12, long_period=26, signal_period=9):
    short_ema = data['Adj Close'].ewm(span=short_period, adjust=False).mean()
    long_ema = data['Adj Close'].ewm(span=long_period, adjust=False).mean()
    macd = short_ema - long_ema
```

```
    signal_line = macd.ewm(span=signal_period, adjust=False).mean()
    histogram = macd - signal_line
    return macd, signal_line, histogram

# Prepare data for LSTM and CNN
def create_dataset(dataset, look_back):
    X= []
    for i in range(len(dataset) - look_back - 1):
        a = dataset[i:(i + look_back), :]
        X.append(a)
    return np.array(X)

# Prepare data for LSTM and CNN
def create_target(dataset, look_back):
    Y = []
    for i in range(len(dataset) - look_back - 1):
        Y.append(dataset[i + look_back, 0])
    return np.array(Y)

def create_plot(y_train, predictions, actual_prices,name):
  plt.figure(figsize=(12, 5))
  plt.plot(range(len(actual_prices)), actual_prices, label='Actual Price')
  plt.plot(range(len(predicted_prices)), predicted_prices, label='Predicted Price')
  plt.title(f'{name} Stock Price Prediction')
  plt.xlabel('Time')
  plt.ylabel('Price')
  plt.legend()
  plt.show()

def evaluate_model(y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    return mae, rmse, r2

for train, target in zip(stocks_train, stocks_train_target):
    train.dropna(inplace=True)
    target['Adj Close'] = (train['Adj Close'])
    train.drop(columns=['Date','Adj Close'],inplace=True)


barclays_train1.columns

for test, target in zip(stocks_test, stocks_test_target):
    test.dropna(inplace=True)
    target['Adj Close'] = (test['Adj Close'])
    test.drop(columns=['Date','Adj Close'],inplace=True)
```

```
barclays_test_target.shape

ma_day = [10, 20, 50]

for ma in ma_day:
   for company in stocks_train1:
      column_name = f"MA {ma} days"
      company[column_name] = company['Adj Close'].rolling(ma).mean()

   for company in stocks_test1:
      column_name = f"MA {ma} days"
      company[column_name] = company['Adj Close'].rolling(ma).mean()


plt.figure(figsize=(10,6 ))
plt.plot(tesco_train1[['Adj Close', 'MA 10 days', 'MA 20 days', 'MA 50 days']])
plt.title('Moving Averages for Tesco stock')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend(['Adj Close', 'MA 10 days', 'MA 20 days', 'MA 50 days']) # Added labels to legend
function
plt.show()

for company in stocks_train1:
   company['RSI'] = calculate_rsi(company)
   company['MACD'], company['Signal Line'], company['MACD Histogram'] =
calculate_macd(company)

for company in stocks_test1:
   company['RSI'] = calculate_rsi(company)
   company['MACD'], company['Signal Line'], company['MACD Histogram'] =
calculate_macd(company)


plt.figure(figsize=(10,6 ))
plt.plot(tesco_train1[['Adj Close', 'RSI']])
plt.title('RSI for Tesco stock')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend(['Adj Close', 'RSI'])
plt.show()

plt.figure(figsize=(10,6 ))
plt.plot(barclays_train1[['MACD','Signal Line']])
plt.title('MACD and Signal Line for Barclays stock')
```

```
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend(['MACD','Signal Line'])
plt.show()

for company in stocks_train1:
  company.drop(columns=['Date','MA 50 days','MACD Histogram','Volume'],inplace=True)
  company.dropna(inplace=True)

for company in stocks_test1:
  company.drop(columns=['Date','MA 50 days','MACD Histogram','Volume'],inplace=True)
  company.dropna(inplace=True)

for train, target in zip(stocks_train1, stocks_train_target1):
    target['Adj Close'] = (train['Adj Close'])
    train.drop(columns=['Adj Close'],inplace=True)

for test, target in zip(stocks_test1, stocks_test_target1):
    target['Adj Close'] = (test['Adj Close'])
    test.drop(columns=['Adj Close'],inplace=True)

lstm_model1 = Sequential()
lstm_model1.add(LSTM(120, return_sequences=True, input_shape=(50, 5)))
lstm_model1.add(LSTM(80, return_sequences=True))
lstm_model1.add(LSTM(60, return_sequences=False))
lstm_model1.add(Dense(50))
lstm_model1.add(Dense(1))
lstm_model1.summary()

lstm_model1.compile(optimizer='adam', loss='mean_squared_error')

cnn_model1 = Sequential()
cnn_model1.add(Conv1D(filters=128, kernel_size=2, activation='relu', input_shape=(50, 5)))
cnn_model1.add(Conv1D(filters=64,kernel_size=2,activation='relu',padding=
'same',kernel_regularizer= regularizers.l2(0.01)))
cnn_model1.add(MaxPooling1D(pool_size=2))
cnn_model1.add(Flatten())
cnn_model1.add(Dense(32, activation='relu'))
cnn_model1.add(Dense(1))

cnn_model1.summary()
cnn_model1.compile(optimizer='adam', loss='mean_squared_error')

from sklearn.preprocessing import MinMaxScaler
for stock, target , name in zip(stocks_train , stocks_train_target, stocks_name):
  look_back = 50
  scaler = MinMaxScaler(feature_range=(0, 1))
```

```
  stock_scaled = scaler.fit_transform(stock)
  target_scaled = scaler.fit_transform(target)
  X_train = create_dataset(stock_scaled, look_back)
  y_train = create_target(target_scaled, look_back)
  print(f'{name} iteration')

  lstm_model1.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.1)
  cnn_model1.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

results = {'name':[],'MAE': [], 'RMSE': [], 'R2': []}

for stock, target, name,train in zip(stocks_test ,
stocks_test_target,stocks_name,stocks_train_target):
  look_back = 50
  scaler = MinMaxScaler(feature_range=(0, 1))
  y_train = scaler.fit_transform(train)
  stock_scaled = scaler.fit_transform(stock)
  target_scaled = scaler.fit_transform(target)
  Y_train = create_target(y_train, look_back)
  X_test = create_dataset(stock_scaled, look_back)
  y_test = create_target(target_scaled, look_back)
  print(f'{name} new iteration')
  predictions = lstm_model1.predict(X_test)
  predicted_prices = scaler.inverse_transform(predictions.reshape(-1, 1)).flatten()
  actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
  years_to_display = 2
  create_plot(Y_train, predicted_prices,actual_prices,name)

  mae, rmse, r2 = evaluate_model(actual_prices, predicted_prices)
  results['name'].append(name)
  results['MAE'].append(mae)
  results['RMSE'].append(rmse)
  results['R2'].append(r2)

results_df = pd.DataFrame(results)
print(results_df)
avg_mae = results_df['MAE'].mean()
avg_rmse = results_df['RMSE'].mean()
avg_r2 = results_df['R2'].mean()

print(f'Average MAE: {avg_mae}')
print(f'Average RMSE: {avg_rmse}')
print(f'Average R2 Score: {avg_r2}')
  #create_plot(Y_train,predicted_prices, actual_prices,scaler,name,years_to_display)

results = {'name':[],'MAE': [], 'RMSE': [], 'R2': []}
```

```
for stock, target, name,train in zip(stocks_test ,
stocks_test_target,stocks_name,stocks_train_target):
  look_back = 50
  scaler = MinMaxScaler(feature_range=(0, 1))
  y_train = scaler.fit_transform(train)
  stock_scaled = scaler.fit_transform(stock)
  target_scaled = scaler.fit_transform(target)
  Y_train = create_target(y_train, look_back)
  X_test = create_dataset(stock_scaled, look_back)
  y_test = create_target(target_scaled, look_back)
  print(f'{name} new iteration')
  predictions = cnn_model1.predict(X_test)
  predicted_prices = scaler.inverse_transform(predictions.reshape(-1, 1)).flatten()
  actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
  years_to_display = 2
  create_plot(Y_train, predicted_prices,actual_prices,name)

  mae, rmse, r2 = evaluate_model(actual_prices, predicted_prices)
  results['name'].append(name)
  results['MAE'].append(mae)
  results['RMSE'].append(rmse)
  results['R2'].append(r2)

results_df = pd.DataFrame(results)
print(results_df)
avg_mae = results_df['MAE'].mean()
avg_rmse = results_df['RMSE'].mean()
avg_r2 = results_df['R2'].mean()

print(f'Average MAE: {avg_mae}')
print(f'Average RMSE: {avg_rmse}')
print(f'Average R2 Score: {avg_r2}')
  #create_plot(Y_train,predicted_prices, actual_prices,scaler,name,years_to_display)

# # Initialize the XGBoost regressor
from xgboost import XGBRegressor
# XGB_model = XGBRegressor(n_estimators=, learning_rate=0.01, max_depth=10,
objective='reg:squarederror')

XGB_model1 = XGBRegressor(
    n_estimators=500,
    learning_rate=0.01,
    max_depth=10,
    min_child_weight=1,
    gamma=0,
    subsample=0.6,
    colsample_bytree=0.6,
```

```
    reg_alpha=0.001,
    reg_lambda=0.001,
    objective='reg:squarederror',
    eval_metric='rmse'
)

from sklearn.preprocessing import MinMaxScaler
for stock_train, target_train,stock_test,target_test, name in zip(stocks_train ,
stocks_train_target,stocks_test,stocks_test_target, stocks_name):
  scaler= MinMaxScaler(feature_range=(0, 1))
  X_train= scaler.fit_transform(stock_train)
  y_train = scaler.fit_transform(target_train)
  X_test = scaler.fit_transform(stock_test)
  y_test = scaler.fit_transform(target_test)
  print(f'{name} iteration')
  XGB_model1.fit(X_train, y_train, eval_set=[(X_test, y_test)], verbose=True)

results = {'name':[],'MAE': [], 'RMSE': [], 'R2': []}
for test, target, name,train in zip(stocks_test ,
stocks_test_target,stocks_name,stocks_train_target):
  scaler = MinMaxScaler(feature_range=(0, 1))
  y_train = scaler.fit_transform(train)
  X_test = scaler.fit_transform(test)
  y_test = scaler.fit_transform(target)
  print(f'{name} new iteration')
  predictions = XGB_model1.predict(X_test)
  predicted_prices = scaler.inverse_transform(predictions.reshape(-1, 1)).flatten()
  actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
  years_to_display = 2
  create_plot(Y_train, predicted_prices,actual_prices,name)
  mae, rmse, r2 = evaluate_model(actual_prices, predicted_prices)
  results['name'].append(name)
  results['MAE'].append(mae)
  results['RMSE'].append(rmse)
  results['R2'].append(r2)

results_df = pd.DataFrame(results)
print(results_df)
avg_mae = results_df['MAE'].mean()
avg_rmse = results_df['RMSE'].mean()
avg_r2 = results_df['R2'].mean()

print(f'Average MAE: {avg_mae}')
print(f'Average RMSE: {avg_rmse}')
print(f'Average R2 Score: {avg_r2}')
  #create_plot(y_train,predicted_prices, actual_prices,scaler,name,years_to_display)
```

```
# LSTM model

lstm_model2 = Sequential()
lstm_model2.add(LSTM(120, return_sequences=True, input_shape=(50, 9)))
lstm_model2.add(LSTM(80, return_sequences=True))
lstm_model2.add(LSTM(60, return_sequences=False))
lstm_model2.add(Dense(50))
lstm_model2.add(Dense(1))
lstm_model2.summary()

lstm_model2.compile(optimizer='adam', loss='mean_squared_error')

cnn_model2 = Sequential()
cnn_model2.add(Conv1D(filters=128, kernel_size=2, activation='relu', input_shape=(50, 9)))
cnn_model2.add(Conv1D(filters=64,kernel_size=2,activation='relu',padding=
'same',kernel_regularizer=regularizers.l2(0.01)))
cnn_model2.add(MaxPooling1D(pool_size=2))
cnn_model2.add(Flatten())

cnn_model2.add(Dense(32, activation='relu'))
cnn_model2.add(Dense(1))

cnn_model2.summary()
cnn_model2.compile(optimizer='adam', loss='mean_squared_error')

from sklearn.preprocessing import MinMaxScaler
for stock, target , name in zip(stocks_train1 , stocks_train_target1, stocks_name):
  look_back = 50
  scaler = MinMaxScaler(feature_range=(0, 1))
  stock_scaled = scaler.fit_transform(stock)
  target_scaled = scaler.fit_transform(target)
  X_train = create_dataset(stock_scaled, look_back)
  y_train = create_target(target_scaled, look_back)
  print(f'{name} iteration')
  lstm_model2.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.1)
  cnn_model2.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

# fig, axs = plt.subplots(17,1, figsize=(8,16))
results = {'name':[],'MAE': [], 'RMSE': [], 'R2': []}

for stock, target, name,train in zip(stocks_test1 ,
stocks_test_target1,stocks_name,stocks_train_target1):
  look_back = 50
  scaler = MinMaxScaler(feature_range=(0, 1))
  y_train = scaler.fit_transform(train)
  stock_scaled = scaler.fit_transform(stock)
  target_scaled = scaler.fit_transform(target)
```

```
 Y_train = create_target(y_train, look_back)
 X_test = create_dataset(stock_scaled, look_back)
 y_test = create_target(target_scaled, look_back)
 print(f'{name} new iteration')
 predictions = lstm_model2.predict(X_test)
 predicted_prices = scaler.inverse_transform(predictions.reshape(-1, 1)).flatten()
 actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
 years_to_display = 2
 create_plot(Y_train, predicted_prices,actual_prices,name)
 #results = {'name':[],'MAE': [], 'RMSE': [], 'R2': []}
 mae, rmse, r2 = evaluate_model(actual_prices, predicted_prices)
 print(f'Model Evaluation - {name}:')
 print(f'Mean Absolute Error (MAE): {mae}')
 print(f'Root Mean Squared Error (RMSE): {rmse}')
 print(f'R-squared (R2) Score: {r2}')
 results['name'].append(name)
 results['MAE'].append(mae)
 results['RMSE'].append(rmse)
 results['R2'].append(r2)

results_df = pd.DataFrame(results)
print(results_df)
avg_mae = results_df['MAE'].mean()
avg_rmse = results_df['RMSE'].mean()
avg_r2 = results_df['R2'].mean()

print(f'Average MAE: {avg_mae}')
print(f'Average RMSE: {avg_rmse}')
print(f'Average R2 Score: {avg_r2}')
 #create_plot(Y_train,predicted_prices, actual_prices,scaler,name,years_to_display)

results = {'name':[],'MAE': [], 'RMSE': [], 'R2': []}
for stock, target, name,train in zip(stocks_test1 ,
stocks_test_target1,stocks_name,stocks_train_target1):
 look_back = 50
 scaler = MinMaxScaler(feature_range=(0, 1))
 y_train = scaler.fit_transform(train)
 stock_scaled = scaler.fit_transform(stock)
 target_scaled = scaler.fit_transform(target)
 Y_train = create_target(y_train, look_back)
 X_test = create_dataset(stock_scaled, look_back)
 y_test = create_target(target_scaled, look_back)
 print(f'{name} new iteration')
 predictions = cnn_model2.predict(X_test)
 predicted_prices = scaler.inverse_transform(predictions.reshape(-1, 1)).flatten()
 actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
 years_to_display = 2
```

```
 create_plot(Y_train, predicted_prices,actual_prices,name)

 mae, rmse, r2 = evaluate_model(actual_prices, predicted_prices)
 print(f'Model Evaluation - {name}:')
 print(f'Mean Absolute Error (MAE): {mae}')
 print(f'Root Mean Squared Error (RMSE): {rmse}')
 print(f'R-squared (R2) Score: {r2}')
 results['name'].append(name)
 results['MAE'].append(mae)
 results['RMSE'].append(rmse)
 results['R2'].append(r2)

results_df = pd.DataFrame(results)
print(results_df)
avg_mae = results_df['MAE'].mean()
avg_rmse = results_df['RMSE'].mean()
avg_r2 = results_df['R2'].mean()

print(f'Average MAE: {avg_mae}')
print(f'Average RMSE: {avg_rmse}')
print(f'Average R2 Score: {avg_r2}')

 #create_plot(Y_train,predicted_prices, actual_prices,scaler,name,years_to_display)

# XGBoost model

XGB_model2 = XGBRegressor(
   n_estimators=500,
   learning_rate=0.01,
   max_depth=10,
   min_child_weight=1,
   gamma=0,
   subsample=0.6,
   colsample_bytree=0.6,
   reg_alpha=0.001,
   reg_lambda=0.001,
   objective='reg:squarederror',
   eval_metric='rmse'
)

from sklearn.preprocessing import MinMaxScaler
for stock_train, target_train,stock_test,target_test, name in zip(stocks_train1 ,
stocks_train_target1,stocks_test1,stocks_test_target1, stocks_name):
 scaler= MinMaxScaler(feature_range=(0, 1))
 X_train= scaler.fit_transform(stock_train)
 y_train = scaler.fit_transform(target_train)
 X_test = scaler.fit_transform(stock_test)
```

```
 y_test = scaler.fit_transform(target_test)
 print(f'{name} iteration')
 XGB_model2.fit(X_train, y_train, eval_set=[(X_test, y_test)], verbose=True)

results = {'name':[],'MAE': [], 'RMSE': [], 'R2': []}
for test, target, name,train in zip(stocks_test1 ,
stocks_test_target1,stocks_name,stocks_train_target1):
 scaler = MinMaxScaler(feature_range=(0, 1))
 y_train = scaler.fit_transform(train)
 X_test = scaler.fit_transform(test)
 y_test = scaler.fit_transform(target)
 print(f'{name} new iteration')
 predictions = XGB_model2.predict(X_test)
 predicted_prices = scaler.inverse_transform(predictions.reshape(-1, 1)).flatten()
 actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
 years_to_display = 2
 create_plot(Y_train, predicted_prices,actual_prices,name)

 mae, rmse, r2 = evaluate_model(actual_prices, predicted_prices)
 results['name'].append(name)
 results['MAE'].append(mae)
 results['RMSE'].append(rmse)
 results['R2'].append(r2)

results_df = pd.DataFrame(results)
print(results_df)
avg_mae = results_df['MAE'].mean()
avg_rmse = results_df['RMSE'].mean()
avg_r2 = results_df['R2'].mean()

print(f'Average MAE: {avg_mae}')
print(f'Average RMSE: {avg_rmse}')
print(f'Average R2 Score: {avg_r2}')

models = ['LSTM', 'CNN', 'XGBoost']
r2_scores = [0.925, 0.88, 0.92]

plt.figure(figsize=(8, 6))
bars = plt.bar(models, r2_scores, color='skyblue')
plt.title('R2 Scores of Different Models')
plt.xlabel('Model')
plt.ylabel('R2 Score')
plt.ylim(0, 1)

for bar in bars:
   yval = bar.get_height()
```

```
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.01, round(yval, 3), ha='center',
va='bottom')

plt.show()
```