Comparative Analysis of Practical Time Series Forecasting Approaches: Statistical Models and Machine Learning Techniques

AN INDUSTRIAL PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF TECHNOLOGY

In

Industrial Mathematics And Scientific Computing

Submitted by:
Sohit Pathak
MA22M019

Under the supervision of
Prof. Dr. Neelesh S Upadhye

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2024

CERTIFICATE

This is to certify that the thesis titled **"Comparative Analysis of Practical Time Series Forecasting Approaches: Statistical Models and Machine Learning Techniques,"** submitted by **Sohit Pathak** (Roll No: MA22M019) to the Indian Institute of Technology Madras, Chennai, for the award of the degree of Master of Technology, is a genuine record of the work carried out by him under my supervision.

It is certified that this is a work based on the references in the bibliography.

Prof. Dr. Neelesh S Upadhye
Project Supervisor
Department of Mathematics
IIT Madras

Date: 08-05-2024

ACKOWLEDGEMENT

I would like to express my heartfelt gratitude to my project supervisor, **Prof. Dr. Neelesh Upadhye**, Professor in the Department of Mathematics, IIT Madras, for his invaluable guidance and support throughout this project. His insights, suggestions, and encouragement were instrumental in the successful completion of this thesis, titled "Comparative Analysis of Practical Time Series Forecasting Approaches: Statistical Models and Machine Learning Techniques."

I also wish to express my sincere thanks to the Department of Mathematics, IIT Madras, for their support in fulfilling the requirements of the Industrial Project coursework and providing me with the opportunity to work on this project.

Name: **Sohit Pathak**
Roll: MA22M019
M.Tech IMSC 2022-24
Department of Mathematics, IIT Madras

Contents

List of Figures

ABSTRACT

Understanding and predicting changes in the stock market has become increasingly important for financial institutions, investors, and analysts. This study focuses to forecast stock values of the Bank Nifty index by using a combination of traditional and machine learning methods. We analyze the performance of various models, including Auto Regressive Integrated Moving Average (ARIMA), rolling ARIMA, Support Vector Regression (SVR), and XGBoost. The research involves several key stages, such as collecting data, preprocessing data, features engineering, selecting models, and tuning hyperparameters. Through a comprehensive evaluation, we analyze and compare the models based on their accuracy, resilience, and computational efficiency for Bank Nifty. Our goal is to analyze and find the most effective methods for predicting Bank Nifty stock data. This project's results can provide valuable insights into investment strategies and financial planning, enhancing our understanding of forecasting techniques within the Bank Nifty stock framework. We aim to support analysts and investors in making informed decisions about stock market trends and forecast accuracy by comparing traditional statistical methods with advanced machine learning techniques.

Key terms:
Stock price forecasting - Bank Nifty - Time series analysis - ARIMA - Utilizing rolling ARIMA, Support Vector Regression (SVR), and Extreme Gradient Boosting (XGBoost) - Financial forecasting and machine learning.

Chapter 1

INTRODUCTION

The stock market is a highly unpredictable realm within any economy, with significant implications for businesses, investors, and policymakers. As stock markets evolve accurately predicting stock values has become increasingly crucial. Accurate forecasting provides valuable insights into market patterns, guiding investment decisions and risk management. There is a growing interest in various forecasting methodologies, from traditional statistical models to advanced machine learning techniques and due to the need for precise predictions. Time series forecasting involves analyzing historical data to make predictions about future outcomes. This procedure involves the use of statistical and machine learning techniques to uncover and analyze patterns and trends in the data. Statistical techniques such as ARIMA and exponential smoothing rely on assumptions about the data generation process to accurately capture its underlying structure. These strategies typically assume that future behavior can be predicted by analyzing past patterns.

On the other hand, machine learning (ML) approaches such as Support Vector Regression (SVR) and Extreme Gradient Boosting (XGBoost) do not require explicit assumptions. And these ML techniques have the ability to automatically extract patterns and relationships from large datasets, making them highly effective for analyzing complex and multi-dimensional data.

This study seeks to expand upon existing research by conducting a thorough comparison of ARIMA, rolling ARIMA, SVR, and XGBoost in the context of stock price forecasting for the Bank Nifty index. The findings will enrich the understanding of time series forecasting and provide practical insights for investors, analysts, and policymakers.

Thus, this project represents a crucial stride towards unraveling the mysteries of the Bank Nifty. By leveraging both traditional statistical methods and cutting-edge machine learning techniques, I aspire to illuminate the path towards a more informed and potentially more profitable financial future.

1.1 Literature Survey

Various statistical and machine learning models have been explored extensively to predict stock market behavior, with a particular focus on the Indian stock markets. This survey summarizes and analyzes recent research efforts in this domain. [3] propose a method to predict stock market crisis events using deep and statistical machine learning techniques. By examining cross-contagion effects among stock, bond, and currency markets, they aim to compute the probability of stock market crash events across different timeframes. Empirical results show that XGBoost outperforms other models in predicting daily returns and volatility. [4] utilize Artificial Neural Networks (ANN) with a backpropagation algorithm to predict share prices. Their decision support system employs a multilayer feed-forward network for prediction, achieving a high validation performance with a regression value of 0.996. The model's flexibility in adjusting input values and epochs enhances its predictive capabilities. [10] compare Deep Learning algorithms with XGBoost for predicting stock market returns using Yahoo's dataset. They evaluate accuracy, prediction, recall, and specificity over various prediction horizons. The study reveals the effectiveness of Deep Learning algorithms, particularly over longer prediction periods. [7] employ Ensemble Kalman Filter methods to predict stock prices, demonstrating superior estimation precision compared to traditional methods. The study highlights the importance of choosing appropriate estimation techniques for accurate stock market predictions. [6] utilize machine learning techniques, including linear regression and SVM regression, to predict future stock prices on the NSE. By incorporating external factors such as foreign exchange rates and market indicators, their model achieves high prediction accuracy. [11] investigates the use of Kalman filters to forecast intraday stock and commodity prices solely based on price history. The study explores correlations in price data and employs Kalman filters within various modeling frameworks to enhance prediction accuracy. [12] discuss technical and fundamental approaches for stock market analysis, emphasizing the efficacy of hybrid and statistical machine learning techniques. They advocate for the integration of diverse methods to improve prediction accuracy and decision-making in stock trading. [14] compare Recurrent Neural Network (RNN) LSTM models with SVM and XGBoost for predicting stock prices. By incorporating market indicators, they assess model performance and highlight the potential of deep learning techniques for accurate stock price forecasting. [15] propose a feed forward ANN with multilayer perceptron for stock price prediction. Through iterative tuning and experimentation, they optimize model configuration and achieve favorable results in predicting stock prices. [8] compare various Deep Learning models, including Multilayer Perceptron (MLP), LSTM, CNN, and RNN, with the linear ARIMA model. Their findings indicate that Neural Networks consistently outperform traditional models in predicting stock prices. [5] investigates the predictive accuracy of linear (ARIMA), nonlinear (BP), and hybrid (ARIMA-BP) models for stock market price prediction based on historical data from the Shanghai Stock Exchange. The study concludes that the ARIMA-BP hybrid model achieves the highest prediction accuracy compared to individual models. [13] compare stochastic models including ARIMA, ANN, RNN, and Holts-Winter for predicting the closing price index of the Bombay Stock Exchange.

Through MAPE evaluation, they find that ANN outperforms other models, indicating its effectiveness in stock price prediction. [9] presents the application of the ARIMA model to predict stock prices in the automobile sector using data from the NSE. Results show promising accuracy, particularly for stocks with high correlation, suggesting the viability of ARIMA in predicting stock behavior. [2] advocate for the use of the ARIMA model in predicting banking stock market data from the Amman Stock Exchange. Despite observing a gradual reduction in forecasting accuracy over time, the study highlights the model's potential for short-term forecasting. [1] explore the impact of learning dynamics on stock price prediction, particularly focusing on the UK stock market. They argue that confining learning agents to short-run dynamics and recursive learning may enhance understanding and prediction of rational expectations equilibrium.

This comprehensive literature survey underscores the breadth of methodologies employed in stock market prediction, ranging from traditional statistical models like ARIMA to advanced machine learning algorithms such as XGBoost and SVR. Each study contributes unique insights and findings, collectively enriching the field of stock market forecasting.

1.2 Aim and Objectives of our Work

The aim of my work is as follows:

- My Objective is To visualize the data and obtain simple descriptive measures (such as plotting data, looking for trends, and seasonal fluctuations) of the main properties of the series
- To Apply a specific statistical method (namely ARIMA) and a machine learning method (namely Support Vector Regressior , XGboost) to the training data and evaluate their performance on the testing data using Accuracy metrics(MAPE, MSE, RMSE), R-squared for rigorous quantitative evaluation and unveil strengths and limitations of each approach.
- To provide recommendations on which method to be used for time series forecasting based on the present results.

Chapter 2

Time Series

A Time series is a sequence of data points collected or recorded at regular intervals over a period of time.Time series data is widely used in various fields, including economics, finance and many others, to study trends, patterns, and make predictions about future values. Example:-

1. **Stock Prices:** Daily closing prices of a company's stock over several months or years. Investors use this data to analyze past performance and make predictions about future stock prices.

2. **Temperature Records:** Hourly or daily temperature measurements collected at a weather station. Meteorologists use this data to track weather patterns, identify seasonal trends, and make weather forecasts.

2.1 Types of Time Series Data

1. **Univariate Time Series:** This type of time series data consists of a single variable observed over time. For example, daily temperature readings over several years for a specific location.

2. **Multivariate Time Series:** multivariate time series data, multiple variables are observed and recorded over time. For example we have a dataset that includes both Temperature and Humidity measurements over time for the same location.

2.2 Component of time series

There are several components in a time series:-

1. **Trend:** Trend long-term movement of a time series is called Trend. It captures the direction in which the time series is moving over time. Trends can be either upward (positive trend), downward (negative trend), or horizontal (no trend).

2. **Seasonal:** Seasonality is the variation that occur at regular intervals in a time series. Seasonal patterns can be less than one year, such as daily, weekly, monthly and can be caused by factors such as weather, holidays, or business cycles.

3. **Cyclical:** Cyclical components are irregular fluctuations in a time series that can occur over varying time periods. They can be caused by economic factors such as business cycles, and can have a periodicity of several years.

4. **Irregular or Random:** Irregular components are the random fluctuations that occur in a time series. It can be caused by sudden changes in the market, natural adversity, or other unforeseen events.

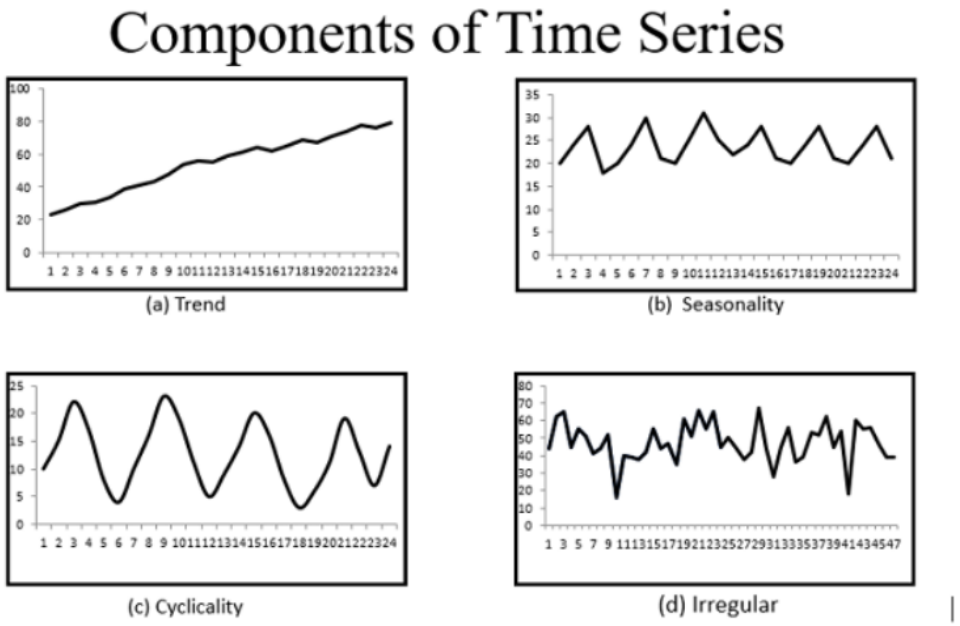

Figure 2.1: Component of time series.

2.3 Stationarity of Time Series

Stationarity is crucial for time series prediction. Stationary time series are predictable and easier to model accurately. This is because the underlying statistical properties remain the same over time, allowing us to use past observations to forecast future values. The following properties define a stationary process for time series data:

1. Expectation is constant.
2. variance is constant.
3. There is no seasonality .

2.4 Check Stationarity of Time Series

To check for stationarity in a time series, following tests can be performed:

1. **Augmented Dickey-Fuller (ADF) test:** The Augmented Dickey-Fuller Test is one of the most popular tests to check for stationarity. It tests the below hypothesis.
   (a) Null Hypothesis, H0: The time series is not stationary.
   (b) Alternative Hypothesis, H1: The time series is stationary.
   If the p-value is less than or equal to 0.05 -Reject H0 and conclude that the time series is stationary. If the p-value is greater than 0.05- Fail to reject H0 and conclude that the time series is not stationary.
2. **Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test:**

   The KPSS test, named after Kwiatkowski-Phillips-Schmidt-Shin, is a type of unit root test used to check whether a time series is stationary around a deterministic trend. Unlike the Augmented Dickey-Fuller (ADF) test, the null hypothesis of the KPSS test is that the series is stationary.

   This key difference leads to a contrasting interpretation of the p-value between the two tests. In the KPSS test, a p-value less than 0.05 suggests that the series is non-stationary. In contrast, the same p-value in the ADF test would indicate that the series is stationary.

   A significant distinction between the KPSS and ADF tests is that the KPSS test can assess stationarity even in the presence of a deterministic trend. The term "deterministic" indicates that the trend's slope is consistent and does not permanently change. Thus, if the series experiences a shock, it will eventually revert to its original trajectory, reflecting a resilient pattern. .

2.5 Time series forecasting

Time series forecasting is the analysis of time series data using statistical methods and modeling techniques to make forecasts and provide insights for strategic decision-making. The accuracy of forecasts, particularly when dealing with the frequently changing variables in time series data, is not always precise.
Forecasting has a range of application in various industries. It has practical application including say, weather forecasting, climate forecasting, finance forecasting, business forecasting, etc.

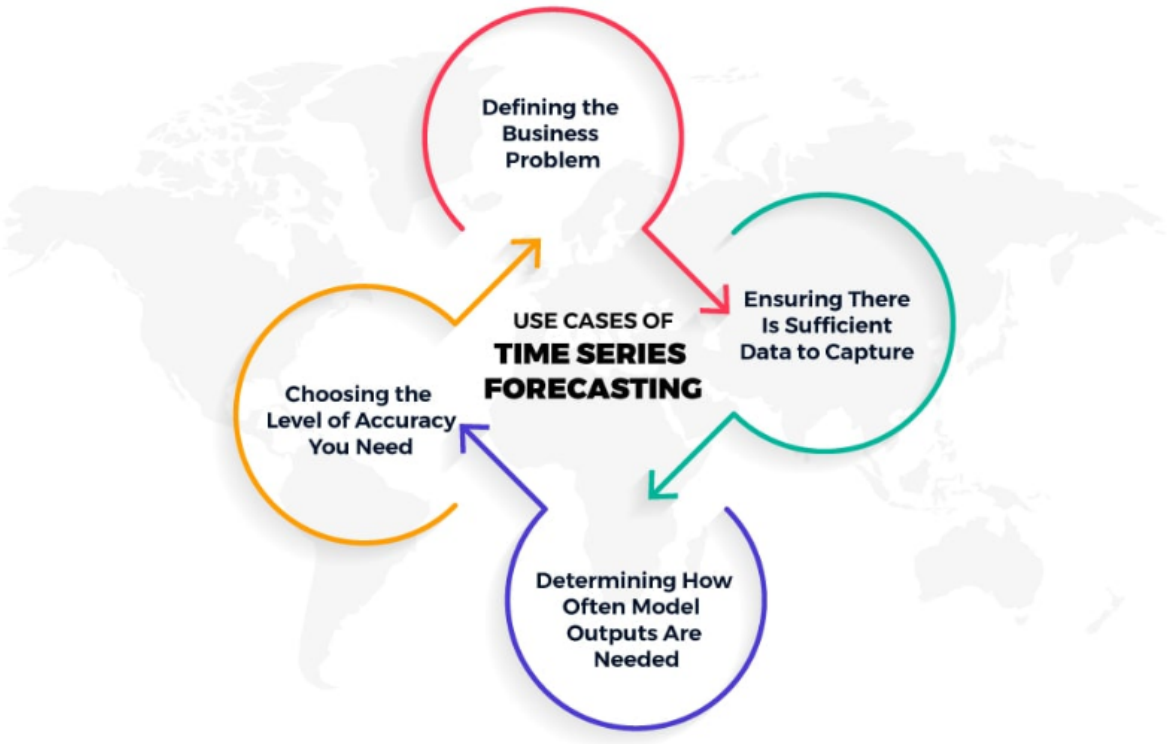

Figure 2.2: Time series forecating.

2.6 Overview of Stock Market

The stock market is a platform where shares or stocks of publicly listed companies are bought and sold. It encompasses various exchanges and over-the-counter markets where investors can trade equity securities, including common stocks, preferred stocks, exchange-traded funds etc.
The stock market plays a crucial role in the economy, allowing companies to raise capital by issuing shares, while giving investors the opportunity to buy ownership stakes in these companies and potentially earn returns through dividends or capital appreciation.

Time series analysis is widely used in the stock market because stock prices, trading volumes, and indices form natural time series, as they are recorded at regular intervals, such as daily, hourly, or even by the minute. This analysis helps identify trends, detect patterns, and forecast future prices, which are crucial for making informed investment decisions.

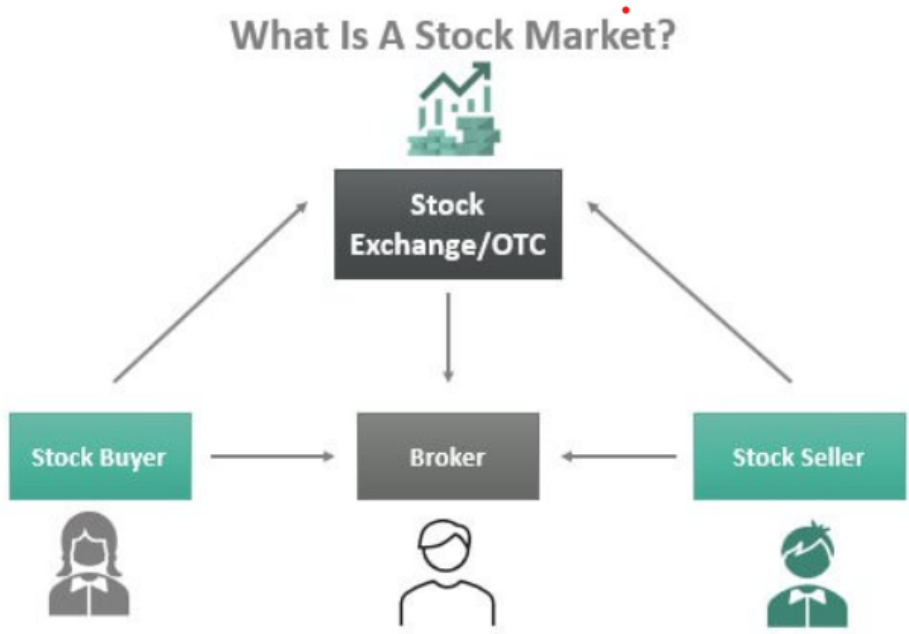

Figure 2.3: stock market structure.

Chapter 3

Models

3.1 XGBoost Model

XGBoost is a highly efficient and widely used technique for gradient boosted trees, which is implemented in an open-source manner. Gradient boosting is a type of machine learning algorithm that falls under the category of supervised learning. XGBoost aims to precisely forecast a target variable by combining the approximations of set of simpler, weaker model.

XGBoost is a technique for ensemble learning. Ensemble learning provides a methodical solution that integrates the predictive capabilities of numerous learners. The outcome is a unified model that combines the results of multiple models. The foundation learners that comprise the ensemble might be derived from either the same learning algorithm or distinct learning algorithms
Bagging and boosting are two popular ensemble learning techniques. XGBoost is mostly used for decision trees, with statistical models being the second most common application.

In gradient boosting for regression, each data point is assigned to a leaf node in a regression tree, which has a continuous score. The weak learners in this case are regression trees. XGBoost optimizes a regularized objective function that merges a convex loss function with a penalty term for model complexity. The training process occurs in an iterative manner, where new trees are added to anticipate the remaining errors of previous trees. These trees are then integrated with the previous ones to get the final prediction. Gradient boosting is named as such because it employs a gradient descent technique to minimize the loss incurred when incorporating new models.

The next page contains a schematic diagram of the XGBoost regression tree model.

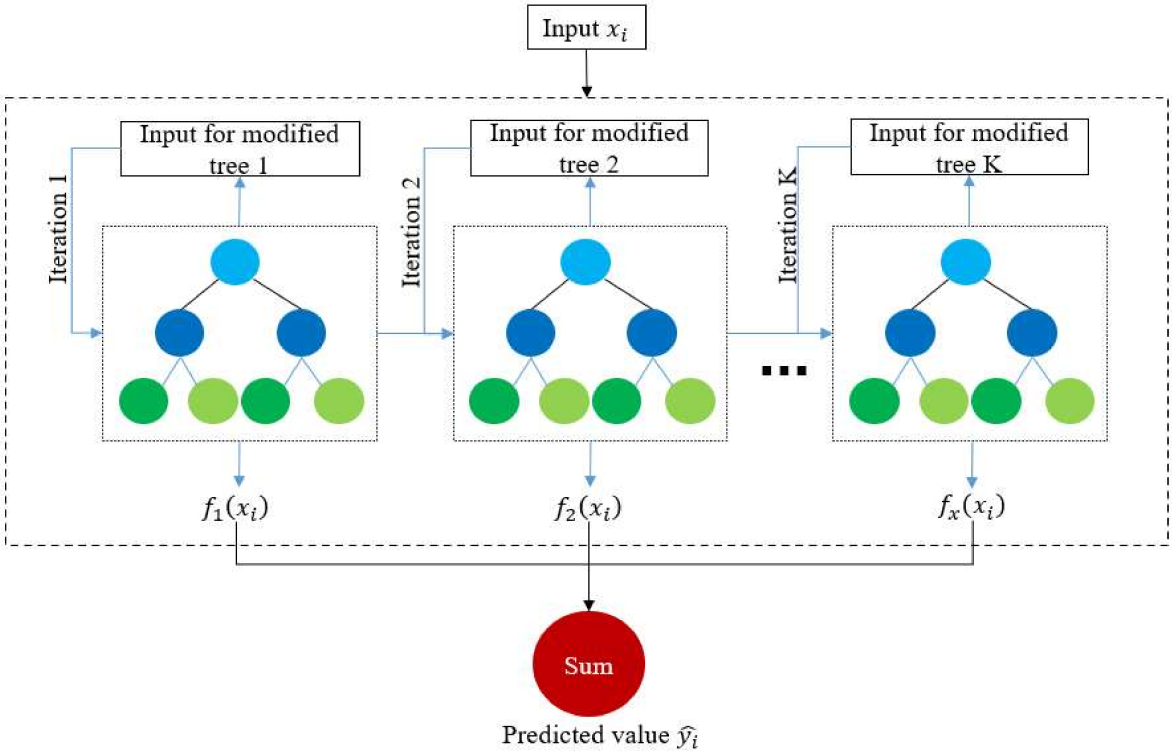

Figure 3.1: XGBoost Model.

3.2 SVR Model

Support Vector Regression (SVR) is a machine learning technique that extends the principles of Support Vector Machines (SVM) to regression tasks. It aims to find a function within a defined margin of tolerance, accounting for noise in the data.

Key Concepts of SVR

- In Support Vector Regression (SVR), the objective is to identify a hyperplane that best fit the data points within a specified tolerance or margin. The margin, typically represented as ε, represents the allowable deviation from the precise regression line.
- SVR can use several kernel functions, such as linear, polynomial, and radial basis functions, to transform data into higher-dimensional spaces. This enables the model to capture complicated correlations between variables.
- Regularization is a technique used in Support Vector Regression (SVR) that has a regularization parameter represented as λ, which trade-off between maximizing the margin and minimizing the error. A larger λ value decreases the tolerance

for error, resulting in a more precise model but with an increased likelihood of overfitting.

SVR Mathematical Equation

An SVR model has to find a function $G(x)$ that predicts a target variable y from input features x. The general form for $G(x)$ with a linear kernel is given by:

$$G(x) = W \cdot x + b, \tag{3.1}$$

where weight vector is W , input vector is x, and b is the bias term. The objective in SVR is to find W and b that minimize the error within a defined margin (ε), here the regularization parameter is λ.

The optimization problem for SVR can be formulated as:

$$\min_{W,b,\xi,\xi^*} \frac{1}{2}\|W\|^2 + \lambda \sum_{i=1}^{n} (\xi_i + \xi_i^*), \tag{3.2}$$

subject to:

$$y_i - G(x_i) \leq \varepsilon + \xi_i, \tag{3.3}$$
$$G(x_i) - y_i \leq \varepsilon + \xi_i^*, \tag{3.4}$$
$$\xi_i, \xi_i^* \geq 0. \tag{3.5}$$

3.3 ARIMA Model

1. **Autoregressive (AR) Component:** This part of the model captures the relationship between an observation and a specified number of preceding observations (lags). It involves coefficients that quantify this relationship. Here is the mathematical equation for an AR(p) model:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t \tag{3.6}$$

   where y_t is the value of the variable at time 't', c is a constant term, $\phi_1, \phi_2, \ldots, \phi_p$ are the coefficients for the past values, ϵ_t is a random error term, and 'p' is the order of the model.

2. **Integrated (I) Component:** This component represents the differencing needed to make a non-stationary time series stationary. Differencing involves subtracting consecutive data points a specified number of times until the series becomes stationary.

3. **Moving Average (MA) Component:** This section deals with the relationship between an observation and a specified number of past errors (shocks or residuals). It uses coefficients to describe this connection.

$$Y_t = \mu + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \ldots + \theta_q\epsilon_{t-q} \tag{3.7}$$

where Y_t is the observation at time 't', μ is the mean over all observations, ϵ_t is the white noise error term at time 't', and $\theta_1, \theta_2, \ldots, \theta_q$ are the coefficients of the past 'q' error terms. The order of the MA model is determined by the value of 'q'.

The ARIMA model is usually expressed as ARIMA(p, d, q), where:

1. p represents the order of the autoregressive part (the number of lags to consider).
2. d represents the degree of differencing required to achieve stationarity.
3. q indicates the order of the moving average part (the number of past errors to include).

3.4 Rolling ARIMA Model

The rolling ARIMA model is a variant of ARIMA used when there's a need for continuous or dynamic forecasting. Instead of fitting the ARIMA model to the entire time series at once, the rolling ARIMA model uses a "rolling" or "sliding" window approach. Here is the working:

1. **Rolling Window:**

   The rolling window is a subset of the time series data that is utilized to train the ARIMA model. The window advances incrementally, moving forward by one step or a predetermined interval, resulting in a succession of overlapping sub-series.

2. **Model Re-fitting:** Whenever the window is moved, the ARIMA model is adjusted again using the data contained in the current window. This allows the model to adapt to variations or emerging patterns in the data as time progresses.

3. **Utilizing Rolling ARIMA for Forecasting:**

   Once the model has been fitted to the data within the current window, a prediction is generated for a future data point (or many ones). As the window shifts, additional data is incorporated, enabling more accurate predictions and potentially enhancing the ability to respond to recent developments.

Chapter 4

Methodology

4.1 Data Description

The dataset used for this comparison research Bank Nifty data was obtained from Yahoo Finance, a well-known site known for its comprehensive financial data. Every entry in the dataset represents a trading day and includes the following attributes:

- **Date**: The particular day when trading takes place.
- **Open**: The price at which the stock is traded when the market opens on a specific trading day.
- **High**: The highest price reached by the stock during the trading day.
- **Low**: The lowest price reached by the stock during the trading day.
- **Close**: The closing price of the stock on the trading day.
- **Adj Close**: The adjusted closing price of the stock.
- **Volume**: The trading volume, representing the total number of shares traded during the trading day.

The dataset contains a total of 250 entries.

Data Table

Table 4.1: Sample of Banknifty Dataset

Date	Open	High	Low	Close	Adj Close	Volume
2022-09-05	39412.050781	39865.199219	39407.398438	39805.750000	39805.289063	214100.0
2022-09-06	39892.949219	40073.750000	39564.300781	39666.500000	39666.039063	210200.0
2022-09-07	39337.750000	39572.050781	39258.250000	39455.898438	39455.441406	138900.0
2022-09-08	39763.898438	40265.750000	39706.398438	40208.949219	40208.480469	203800.0
2022-09-09	40520.750000	40685.949219	40280.300781	40415.699219	40415.230469	207000.0

4.2 Data Preprocessing for Data Visualization

During the process of visualizing the dataset, it was noted that certain columns contained NaN (Not a Number) values. These NaN values can introduce disruptions in the plotted graphs, potentially leading to misinterpretations of the data. To maintain a smooth and uninterrupted representation of the data in the visualizations, a comprehensive data preprocessing procedure was undertaken.

4.2.1 Identification and Impact of NaN Values

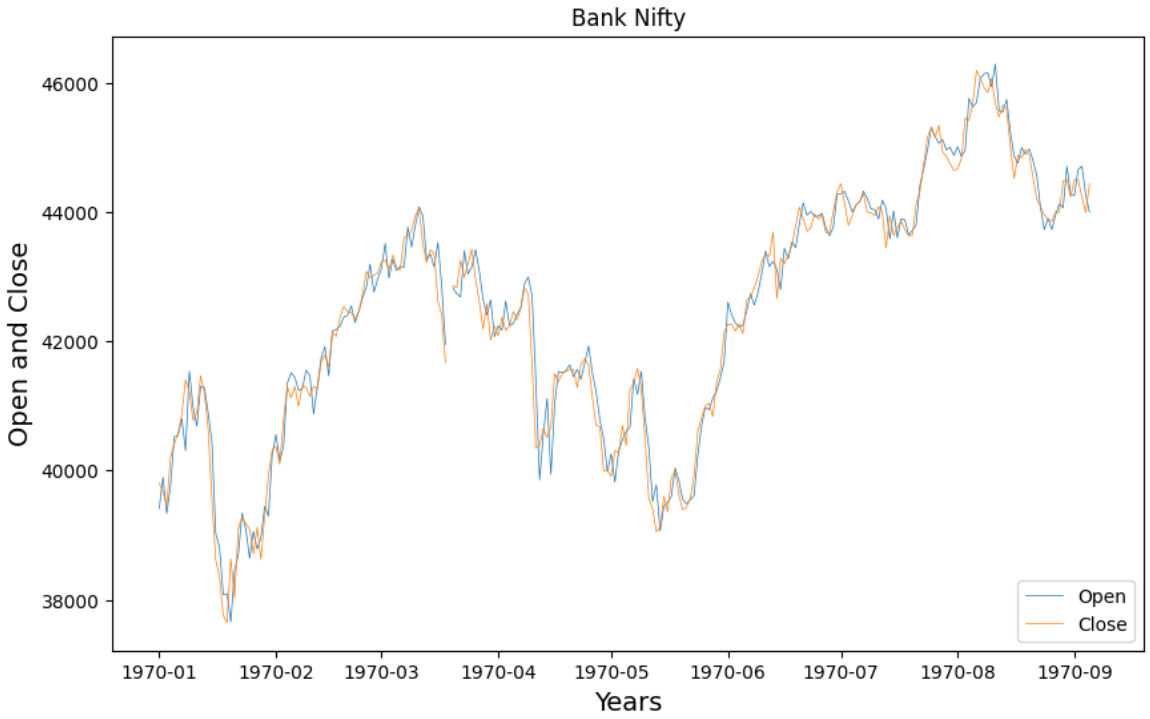

Figure 4.1: Occurrence of NaN Values in the Dataset

Figure 4.1 illustrates the occurrence of NaN values in the dataset. These missing values, denoted by blank spaces in the plotted graphs, can create discontinuities, as observed in the fluctuations of the plotted data points. Such disruptions can obscure the true underlying trends and patterns in the data, compromising the effectiveness of the visualizations.

4.2.2 Enhanced Data Visualizations before NaN Value Imputation

To address the issue of NaN values and mitigate their impact on the visualizations, a customized NaN value imputation technique was implemented. This technique involved filling the NaN values with the mean of the previous two valid values in the

respective column. By leveraging nearby historical data points, this approach aimed to preserve the overall continuity and integrity of the dataset.

4.2.3 Enhanced Data Visualization

Following the application of the customized NaN value imputation technique, significant improvements were observed in the continuity and coherence of the visualized data. Figure 4.2 showcases the enhanced visualizations, depicting a smoother representation of the underlying trends and patterns, thereby facilitating a more accurate interpretation of the data.

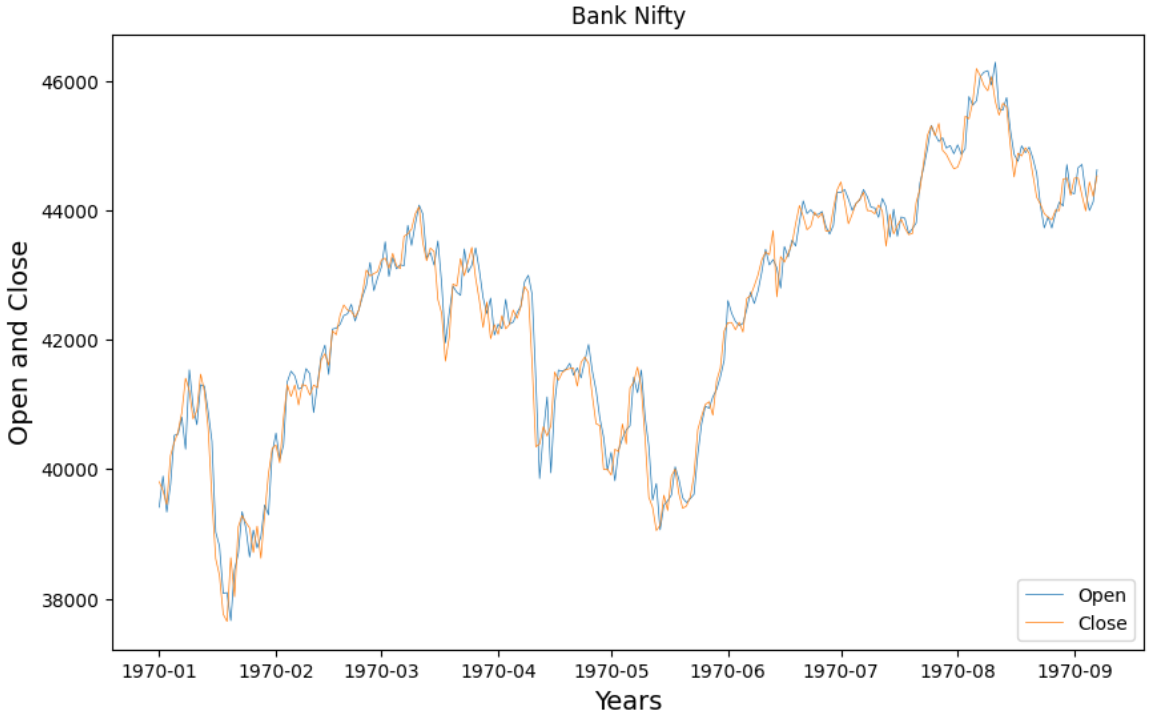

Figure 4.2: Enhanced Data Visualizations after NaN Value Imputation

4.3 OHLC Chart and Time Series Decomposition

The OHLC chart visually represents the opening, highest, lowest, and closing prices of a security or index over a specified period. Each time interval is depicted by a vertical line or bar, with horizontal lines extending from the bars to denote the opening and closing prices. This chart provides a succinct overview of price movements, facilitating the analysis of market trends and aiding in trading decisions, as illustrated in Figure 4.3.

Now, we also proceed to examine the fundamental characteristics of our data through time-series decomposition. Upon initial inspection, our data reveals an upward trend followed by a period of detrending from March 2023 to April 2024, followed

Figure 4.3: OHLC Chart

by a return to an upward trend. Additionally, the presence of seasonality is apparent, which will be further investigated using the Augmented Dickey-Fuller (ADF) test.

Upon analyzing the residue plot, it becomes evident that our residuals exhibit a random pattern, devoid of any discernible trend similar to our stock data. This observation aligns with our objective, indicating that the residuals do not contain any systematic patterns. The analysis of residues constitutes a crucial aspect of model building, signifying good model fit and adherence to statistical assumptions.

Referencing Figure 4.4, we visualize the time series decomposition, showcasing the identified trends, seasonality, and residuals.

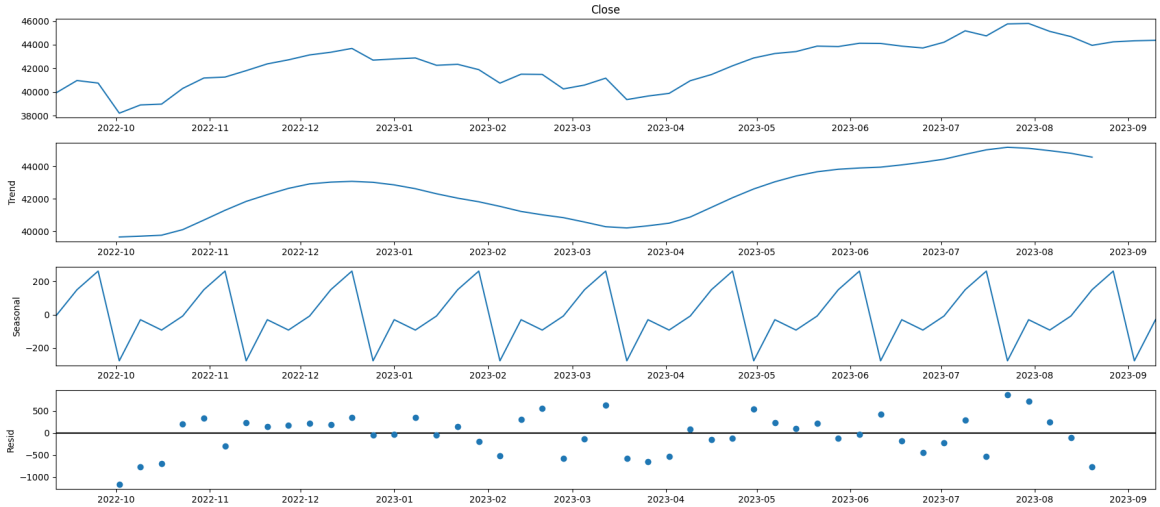

Figure 4.4: Time Series Decomposition

4.4 Feature Engineering Explanation

4.4.1 Exponential Moving Average (EMA) and Simple Moving Averages (SMA)

EMA and SMA are common techniques used in financial analysis to smooth out price data and identify trends. Here's how they work:

- **Exponential Moving Average (EMA)**: It is designed to be highly responsive to short-term price movements by giving more weight to recent prices.
- **Simple Moving Average (SMA)**: It is calculated by taking the average of closing prices over a specific time frame.

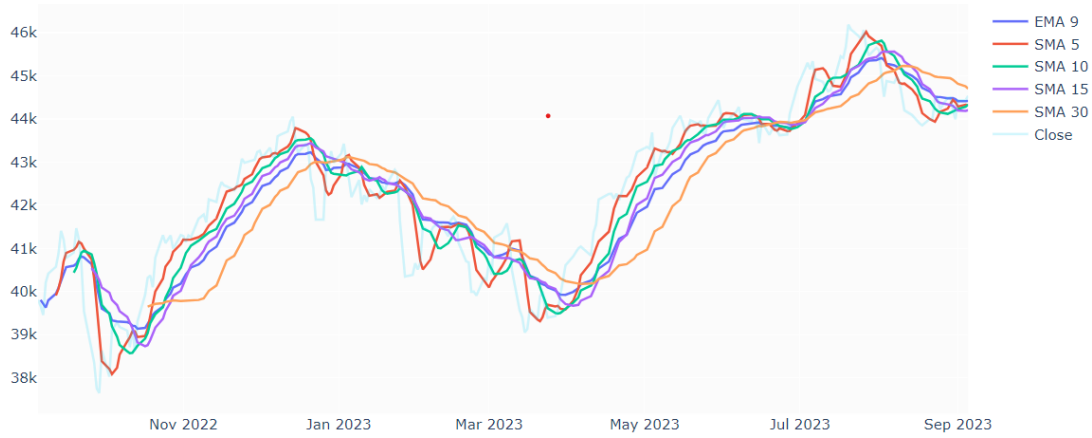

Figure 4.5: EMA and SMA

The code calculates exponential moving averages (EMA) and simple moving averages (SMA) over various periods, including 9, 5, 10, 15, and 30 days. This analysis offers valuable insights into different aspects of price trends. Changing the Aligning calculated values by one time step helps prevent data leakage by ensuring they are in sync with future data.

4.4.2 Moving Average Convergence Divergence (MACD)

MACD is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. Here's how it's calculated:

- **MACD Line**: It's the difference between two exponential moving averages (typically 12-day EMA and 26-day EMA) of the closing prices.
- **Signal Line**: It's a 9-day EMA of the MACD line.

By plotting both the MACD line and the signal line, traders can visualize crossovers and divergences, which are often used as trading signals.

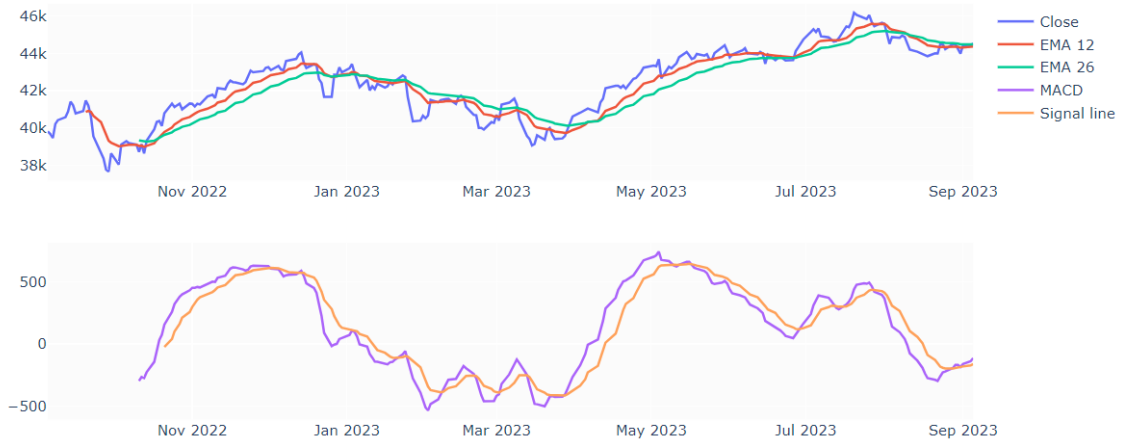

Figure 4.6: MACD

4.5 Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

Below are the plots for the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of a rolling ARIMA model with integrated (I) and moving average (MA) terms set to 0.
The ACF plot provides a visual representation of the autocorrelation of the time series at different lags. ACF values are calculated and graphed for lags 1 to 20 in this plot. A bar plot is generated to visualize the ACF values, with each bar representing the autocorrelation at a specific lag.

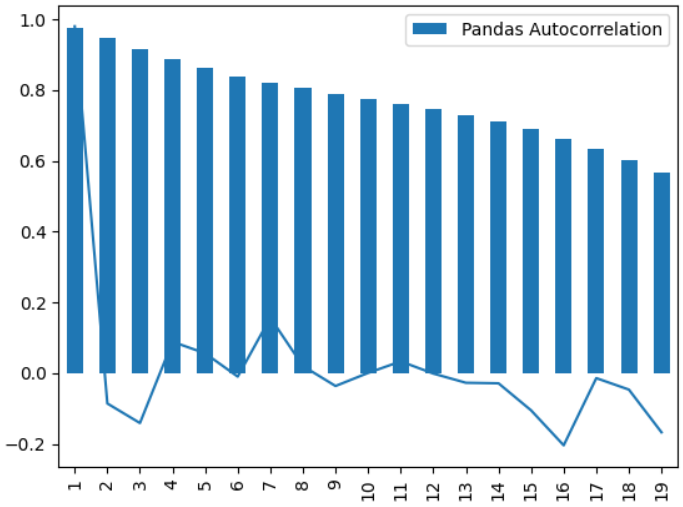

Figure 4.7: Autocorrelation Plot

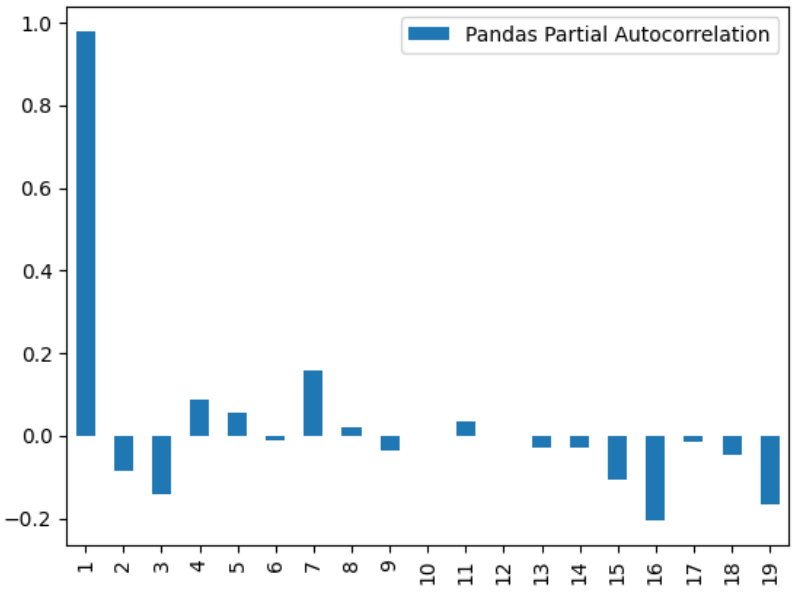

Figure 4.8: partial autocorrelation Plot

Simillarly the PACF plot displays the partial autocorrelation of the time series. It calculates the relationship between observations that are spaced apart by a specific time period, while also accounting for the impact of other time periods. The PACF values are calculated and visualized for lags 1 to 20. A bar plot is used to display the partial autocorrelation at each lag.
By examining these plots, we can pinpoint the lag values that have a significant impact on the time series. Based on the ACF and PACF plots, it appears that there is an autoregressive (AR) process with a lag of 7 to 9. This suggests that previous observations within this range have a significant influence on the current observation.

4.6 Train-Test Split for Machine Learning Models

In machine learning, the train-test split is a common technique used to evaluate the performance of a model on unseen data. Here's how it works:

```
test_size  = 0.15
valid_size = 0.15

test_split_idx  = int(df.shape[0] * (1-test_size))
valid_split_idx = int(df.shape[0] * (1-(valid_size+test_size)))

train_df  = df.loc[:valid_split_idx].copy()
valid_df  = df.loc[valid_split_idx+1:test_split_idx].copy()
test_df   = df.loc[test_split_idx+1:].copy()
```

The code above performs a train-test split on the dataset. This code splits the data into training, validation, and test sets.

Figure 4.9: Train-Test Split

The figure below illustrates the train-test split, where a portion of the data is allocated to the training set, validation set, and test set. The training set is used to train the machine learning model, while the validation set is used to tune hyperparameters and evaluate model performance during training. Finally, the test set is used to assess the model's performance on unseen data.

4.7 XGBoost Methodology

After the Feature Engineering and Train Test split I applied XGBoost algoritham as below.

4.7.1 Objective Function

XGBoost aims to minimize an objective function that combines the training loss and a regularization term. The training loss depends on the chosen loss function (e.g., squared error for regression). Here's a general form:

$$\text{Obj} = \sum L(\hat{y}_i, y_i) + \gamma \cdot G(\Omega)$$

where:

- $\sum$: Summation over all data points (i).
- $L(\hat{y}_i, y_i)$: Loss function for the predicted value ($\hat{y}_i$) and the actual target value (y_i).
- γ: Regularization parameter controlling the strength of the penalty term.
- $G(\Omega)$: Regularization term based on the model complexity (Ω).

4.7.2 Regularization Term (Example: L2 Norm)

The regularization term often uses the L2 norm of the model weights to penalize complex models. Here's an example with L2 norm:

$$G(\Omega) = \frac{\lambda}{2} \sum ||w_j||^2$$

where:

- λ: Regularization strength parameter.
- w_j: Weights of the model (e.g., weights at each leaf node in the decision trees).

4.7.3 Gradient Boosting with Decision Trees

XGBoost builds an ensemble of decision trees sequentially ($t = 1, 2, \ldots, T$). At each iteration:

- **Predict**: Predict the target value using the current ensemble:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t-1} f_k(x_i)$$

  where $f_k(x_i)$ is the prediction of the k-th tree in the ensemble for data point i.

- **Calculate Gradient**: Compute the gradient of the loss function with respect to the predictions:

$$g_i^{(t)} = \frac{\partial L(\hat{y}_i^{(t-1)}, y_i)}{\partial \hat{y}_i^{(t-1)}}$$

- **Build New Tree**: Train a new decision tree ($f_t(x_i)$) using the negative gradients (pseudo-residuals) as targets. This focuses on correcting the errors from previous trees.

- **Update Predictions**: Update the predicted value by adding the new tree's prediction weighted by a learning rate (η):

$$\hat{y}_i = \hat{y}_i^{(t-1)} + \eta \cdot f_t(x_i)$$

4.8 SVR Methodology

After the Feature Engineering and Train Test split I applied SVR algoritham as below.

4.8.1 Objective

SVR aims to predict a target variable y from input features x by finding a function $f(x)$. The model seeks to minimize prediction errors within a defined margin (ε), while considering regularization.

4.8.2 Mathematical Formulation

The general form of the SVR function $G(x)$ with a linear kernel is represented as:

$$G(x) = W \cdot x + b$$

Here, W denotes the weight vector, x is the input vector, and b is the bias term.

The optimization problem for SVR can be formulated as:

$$\min_{W,b,\xi,\xi^*} \frac{1}{2}\|W\|^2 + \lambda \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$

subject to:

$$y_i - G(x_i) \leq \varepsilon + \xi_i$$
$$G(x_i) - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

4.8.3 Code Implementation

1. Define a parameter grid (*svr_parameters*) specifying kernel types, regularization parameters λ, gamma values, and polynomial degrees.
2. Construct a pipeline incorporating SVR.
3. Utilize GridSearchCV to search for the best SVR parameters based on cross-validated mean squared error.
4. Fit the SVR model with the best parameters obtained from GridSearchCV.
5. Predict target values on the test set using the trained SVR model.
6. Evaluate the model's performance using metrics such as mean squared error (MSE) and optionally, mean absolute percentage error (MAPE).
7. Additionally, calculate the validation MSE to assess the model's performance on unseen data.

4.9 ARIMA Methodology

The ARIMA model comprises three parameters, denoted as p, d, and q. The process of constructing the model involves three sequential steps.

Step 1: Conducting tests to ascertain the stationarity of the time series data. Establishing stationarity is a fundamental aspect of modeling time series data. During this phase, efforts were made to eliminate any inherent trends in the data, thus enhancing the model's predictive capabilities. The Augmented Dickey-Fuller test was employed to examine whether the time series sample contains a unit root, thereby

testing the null hypothesis. The obtained test statistic evaluated against a significance level of 0.05 to determine stationarity. In instances where the data displayed non-stationarity, indicated by a p-value exceeding 0.05, differencing was applied to transform the data into a stationary process. This step is pivotal as it unveils novel correlations and other noteworthy statistical characteristics of the data.

Step 2: Determining the autoregressive (AR) and moving average (MA) coefficients. The Partial Auto-Correlation Function (PACF) plot was utilized for AR models to discern the appropriate order, while the Auto-Correlation Function (ACF) plot was employed for MA models.

Step 3: Estimation and prediction. Following the determination of the p, d, and q values, the precision of the ARIMA model was assessed using a designated training dataset. Subsequently, the model was leveraged to forecast future stock prices.

4.9.1 ARIMA Algorithm

1. Import the dataset that includes the following parameters: Open, High, Low, Close, Volume, and Adjusted Close.

2. Perform data preprocessing by replacing NaN values with the average of the two most recent valid values.

3. The time series will be assessed for stationarity. If the time series is nonstationary, stationarity can be attained by taking the difference (d). Utilize the Partial Autocorrelation Function (PACF) and Autocorrelation Function (ACF) to determine whether to incorporate Autoregressive (AR), Moving Average (MA), or both models.

    - In the context of ACF (Auto Correlation Function):
        (a) If there is a positive correlation between past values, utilize an Auto Regressive (AR) model.
        (b) Utilize the Moving Average (MA) model when there is a presence of negative autocorrelation in lagged values.
    - In the partial autocorrelation function (PACF), if there is a decline in the PACF plot at lagged values, it indicates the use of an autoregressive (AR) model. On the other hand, if the drop in PACF is more gradual, it suggests the use of a moving average (MA) model.

4. Here, I have implemented Auto ARIMA from Statsmodels, which automatically selects the best ARIMA model based on given parameters. After visualizing all the plots mentioned above, I have implemented Auto ARIMA.

5. Compare the predicted values to the actual values.

6. Plot the graph for actual versus predicted price and calculate the accuracy of prediction.

4.10 Rolling ARIMA Methodology

In this code snippet, a rolling ARIMA model is used to make predictions from a test set based on a training set. Here's a step-by-step breakdown with the correct line references:

- Lines 3–4: The code initializes two empty lists, `predicted1` for storing predicted values and `resid_test` for storing residuals (the difference between actual and predicted values). It also sets `history` to contain the training data, which is used to fit the ARIMA model.
- Lines 5–12: The code iterates over each item in the test set, fitting a new ARIMA model with the specified order (7,0,0). This order means the model uses 7 lags for the autoregressive component, no differencing, and no moving average.
- In each iteration, the model is fit using the current `history`. The `model_fit.forecast()` method is called to predict the next value in the series, with the result (`yhat`) stored in `predicted1`. The residual (difference between the observed test value and `yhat`) is appended to `resid_test`. The observed test value (`obs`) is added to `history` to update the series with new data for the next iteration.
- A print statement displays the predicted value (`yhat`) and the actual test value (`obs`) for each iteration (Line 11).
- Lines 13–14: At the end of the loop, the code creates a new list `test_resid`, which contains the residuals stored in `resid_test`. The mean squared error (MSE) is then calculated between the test set and `predicted1`. MSE is a common metric used to measure the accuracy of forecasts.

Chapter 5

RESULTS

5.1 Model evaluation methods

Accuracy metrics: Accuracy metrics are some of the most common methods to compare model outputs. Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) are examples of accuracy metrics.

- **Mean Squared Error (MSE)**:

  Mean Squared Error is calculated as after taking differences between the predicted values ($\hat{y}_i$) and the actual values (y_i) in the dataset, take the square then take the average, here n is the number of data points:

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

  Minimum value of this error is for better accuracy.

- **Mean Absolute Percentage Error (MAPE)**:

  The Mean Absolute Percentage Error (MAPE) is a measure of the average difference between predicted values ($\hat{y}_i$) and actual values (y_i). It is calculated by taking the absolute difference between each predicted and actual value, dividing it by the actual value, then take average and then multiplying by 100 to express it as a percentage.

  $$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100$$

  Lower MAPE values indicate better accuracy.

R-squared: R-squared measures the proportion of the variance in the dependent variable (y) that is predictable from the independent variable ($\hat{y}$) in the model. The model with the highest R-squared value is considered to be the best. R-squared is calculated as:

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

where $\bar{y}$ is the mean of the observed data. Higher R-squared values indicate better fit of the model to the data.

5.2 XGBoost Results

5.2.1 Feature Importance

I have implemented XGBoost regressor, and now I have plotted the feature importance. You can see in Figure 5.1, it is evident that EMA_9 holds the highest importance among the features, followed by SMA_5, MACD, and then SMA_15.

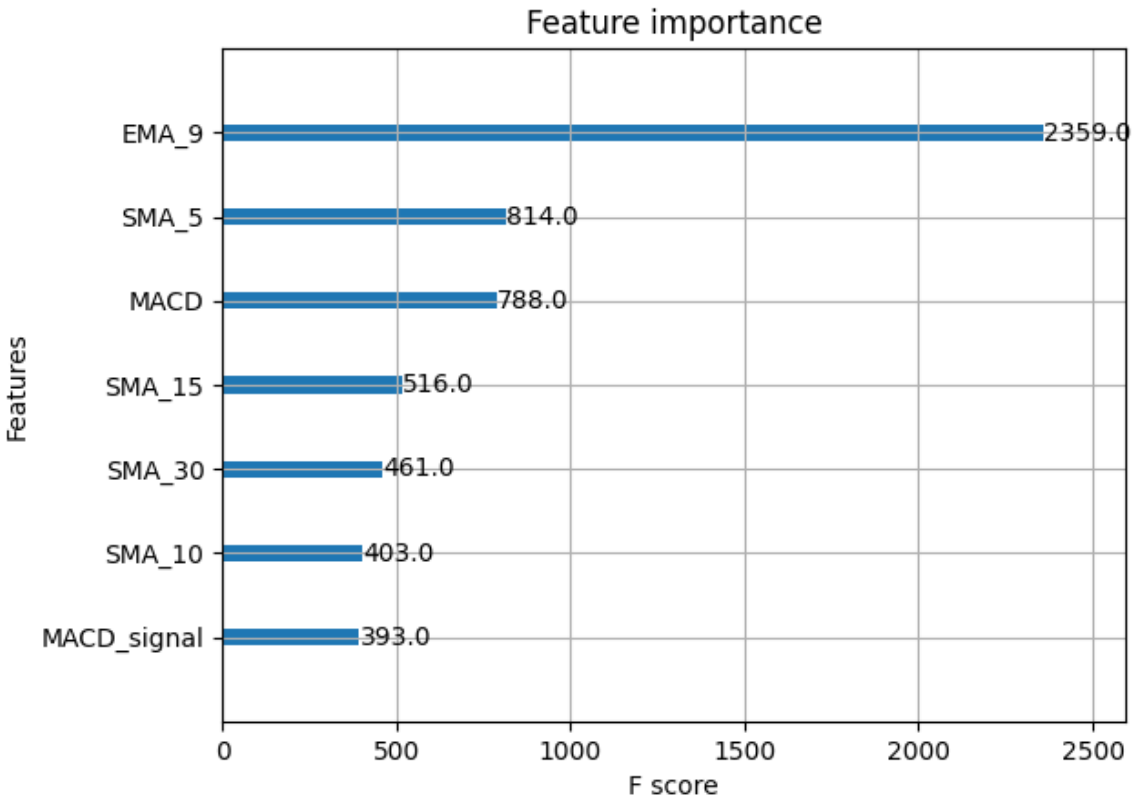

Figure 5.1: Feature Importance

Model evaluation Results for XGBoost Regressor to predict the stock prices. The results are as follows:

- For the validation data:
  - Mean Squared Error: 1699933.16
  - Mean Absolute Percentage Error (MAPE): 2.52%
- For the testing data:
  - Mean Squared Error: 2813296.31
  - Mean Absolute Percentage Error (MAPE): 3.58%

The true values (y_{true}) and predicted values (y_{pred}) are given as follows:

Table 5.1: True and Predicted Values

True Values (y_{true})	Predicted Values (y_{pred})
46075.199219	43604.05
45923.050781	43687.387
45845.0	43603.7
46062.351563	43687.387
45679.300781	43801.57
45468.101563	43741.863
45651.101563	43713.52
45592.5	43551.027
44995.699219	43551.027
44513.449219	43410.465

Additionally, the comparison between the actual and predicted stock prices for Bank Nifty can be seen in the figure 5.2.

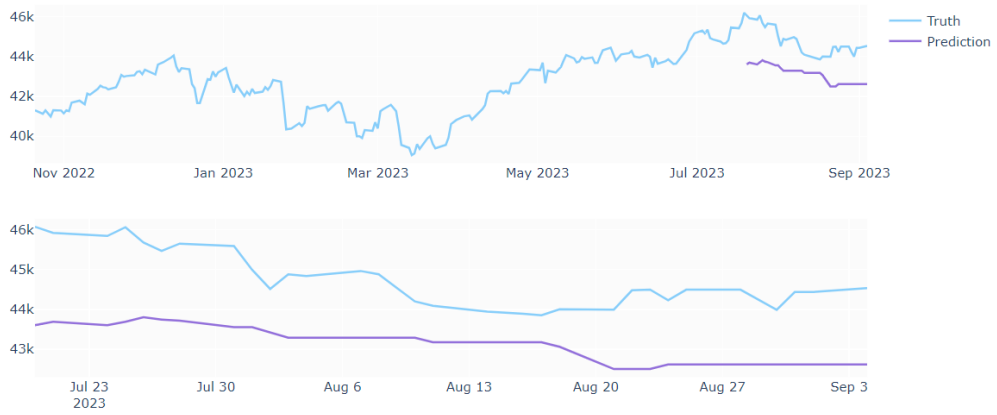

Figure 5.2: XGBoost Prediction

5.3 SVR Results

5.3.1 Feature Importance

I have implemented SVR regressor, and now I have plotted the feature importance. As shown in Figure 5.3, EMA_9 has more importance among others, followed by MACD, MACD_signal, and then SMA_15.

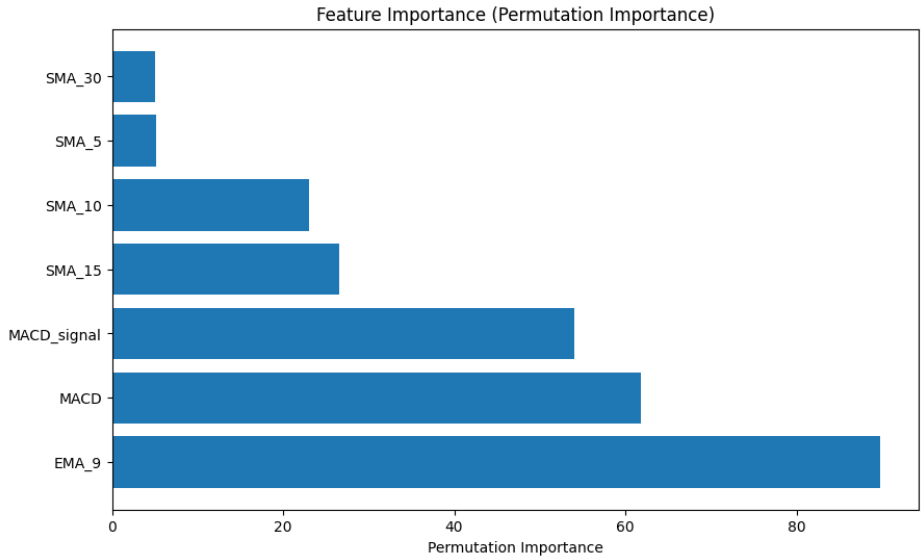

Figure 5.3: Feature Importance

Model evaluation Results for SVR to predict the stock prices. The results are as follows:

- For the validation data:
  - Mean Squared Error (MSE): 805,285.08666
  - Mean Absolute Percentage Error (MAPE): 1.7723%
- For the testing data:
  - Mean Squared Error (MSE): 2,091,730.8974
  - Mean Absolute Percentage Error (MAPE): 3.0066%

Comparison between Actual and Predicted Stock Prices (Bank Nifty)

Actual	Predicted
46075.199	47393.654
45923.051	47339.296
45845.000	47219.432
46062.352	46886.200
45679.301	46875.893
45468.102	46540.294
45651.102	46133.360
45592.500	46317.504
44995.699	46256.193
44513.449	45962.137

Additionally, the comparison between the actual and predicted stock prices for Bank Nifty can be seen in the figure 5.4

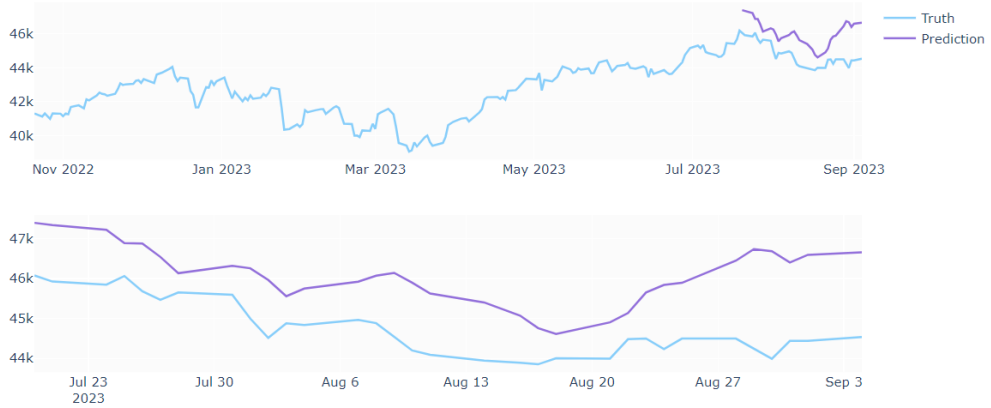

Figure 5.4: SVR Prediction

5.4 ARIMA Results

Auto ARIMA Model Results

The best model identified through stepwise search to minimize AIC is ARIMA(1,0,0)(0,0,0)[5] with intercept.

Model Statistics

- AIC: 605.255
- Total fit time: 1.027 seconds

Training Results

- Mean Squared Error (MSE) on training data: 40,900.591
- Mean Absolute Error (MAE) on training data: 168.939

Testing Results

- Mean Squared Error (MSE) on testing data: 5,424,215.362
- Mean Absolute Error (MAE) on testing data: 2,109.904

This model seems to perform reasonably well on the training data, with relatively low errors. However, the errors on the testing data are considerably higher.

Table 5.2: SARIMAX Results

Dep. Variable:	y	**No. Observations:**	175	**Model:**	ARIMA(1, 0, 0)
Date:	Mon, 06 May 2024	**AIC:**	2598.076	**BIC:**	2607.570
Time:	21:26:41	**HQIC:**	2601.927	**Sample:**	0-175
coef	std err	z	P¿—z—	[0.025	0.975]
const	4.151e+04	894.759	46.390	0.000	3.98e+04
ar.L1	0.9695	0.019	51.147	0.000	0.932
sigma2	1.559e+05	1.54e+04	10.114	0.000	1.26e+05

Ljung-Box (L1) (Q):2.23	Jarque-Bera (JB):12.36
Prob(Q):0.14	Prob(JB):0.00
Heteroskedasticity (H):0.76	Skew: − 0.53
Prob(H) (two-sided):0.29	Kurtosis:3.76

Warnings:

Figure of Train, Test, and Predicted Values for Auto-ARIMA Model is figure 5.5

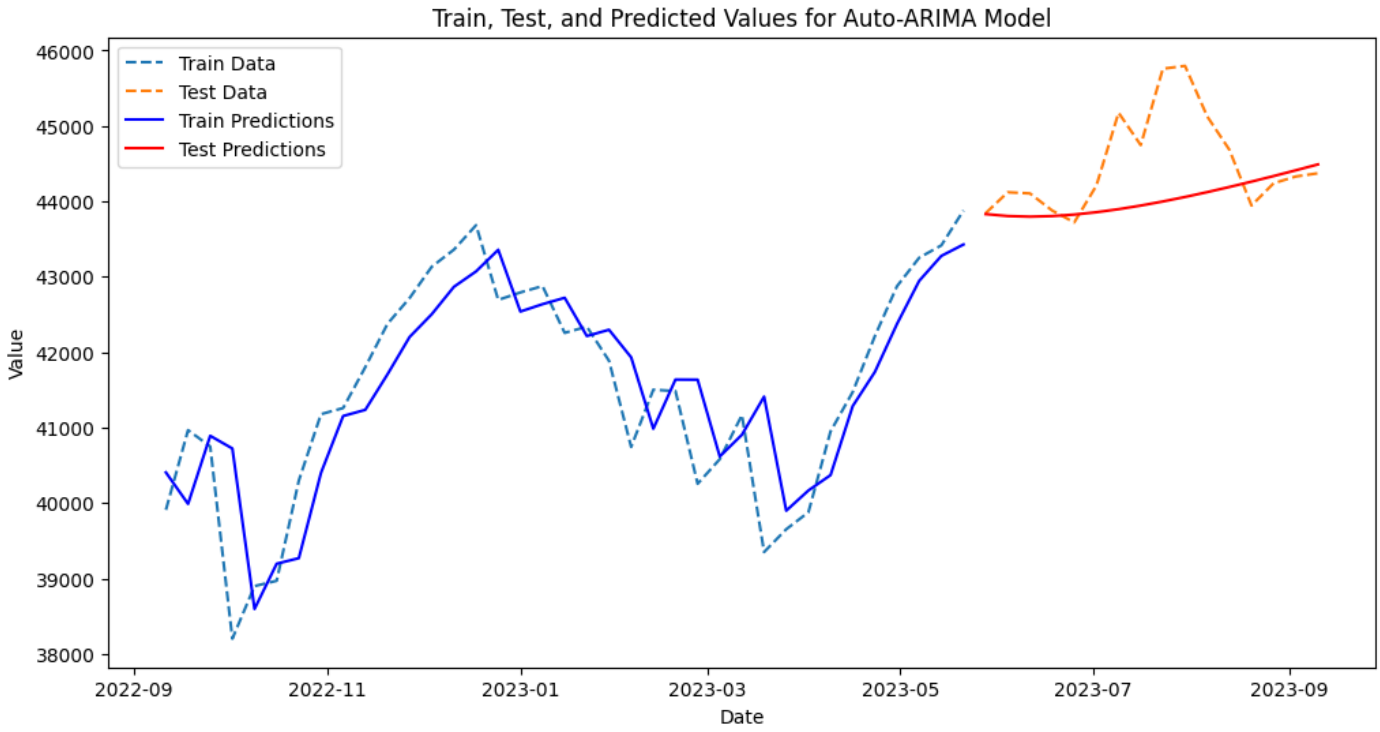

Figure 5.5: ARIMA(1,0,0) Model

5.5 Rolling ARIMA Results

Mean Absolute Percentage Error = 0.51%
Test mean squared error: 81853.846

Figure of Test, and Predicted Values for Rolling-ARIMA Model is figure 5.6

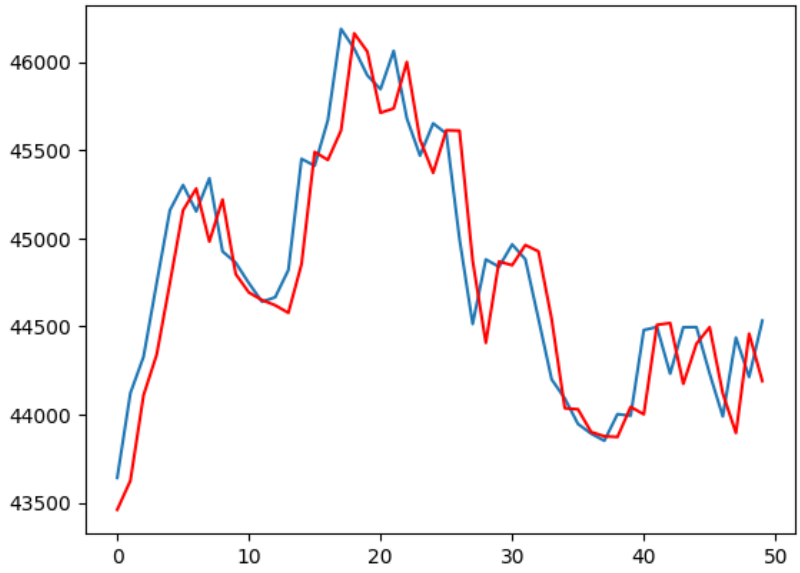

Figure 5.6: Rolling-ARIMA(7,0,0) Model

Predicted	Expected
44980.559910	45339.898438
45219.328102	44925.000000
44796.525663	44860.851563
44692.052143	44745.050781
44648.822427	44639.449219
44619.373613	44665.050781
44576.672344	44819.300781
44852.854203	45449.750000
45489.020483	45410.851563
45443.692726	45669.300781
45610.389200	46186.898438
46161.796659	46075.199219
46057.964935	45923.050781
45710.982695	45845.000000
45735.681164	46062.351563
45999.689554	45679.300781
45559.399945	45468.101563
45370.626271	45651.101563
45611.631310	45592.500000
45609.367217	44995.699219
44871.985037	44513.449219
44406.926620	44879.500000
44867.685952	44837.500000
44846.436985	44964.449219
44961.073438	44880.699219
44925.246569	44541.800781
44541.314193	44199.101563
44035.458001	44090.949219
44029.678087	43946.398438
43900.760423	43891.351563
43876.536784	43851.050781
43872.483406	44002.000000
44043.183018	43993.250000
44001.198595	44479.050781
44508.151383	44496.199219
44518.713204	44231.449219
44173.854071	44494.648438
44401.260614	44495.250000
44494.915055	44232.601563
44119.274428	43989.148438
43895.102333	44436.101563
44458.472728	44212.625000
44189.941320	44532.148438

Table 5.3: Predicted and Expected Values

Chapter 6

CONCLUSION

In this comparative analysis of practical time series forecasting approaches, we explored the effectiveness of statistical models and machine learning techniques in predicting Bank Nifty data. Our objective was to visualize the data, apply specific methods, and evaluate their performance using accuracy metrics to discern strengths and limitations.

The ARIMA model, a traditional statistical method, demonstrated suboptimal performance in both training and testing phases. Despite its simplicity and ease of implementation, ARIMA struggled to capture the complex patterns inherent in the Bank Nifty data, leading to high mean squared error (MSE) and mean absolute error (MAE) on the testing data.

On the other hand, machine learning models such as Support Vector Regression (SVR) and XGBoost shown higher prediction ability. The SVR model shows favorable outcomes with comparatively low MSE and MAPE values on both the validation and testing datasets. Similarly, XGBoost exhibited impressive performance, with competitive MSE and MAPE values so it will suggesting its effectiveness in capturing the patterns and trends in our Bank Nifty data.

Furthermore, Implementation of Rolling ARIMA produced superior outcomes in comparison to the conventional ARIMA model. However, the applicability of this system in real-world scenarios is restricted due to its need on regular updates of test or unseen data following each prediction as i have discussed in methodology.

In summary, the results indicate that ML Techniques, specifically SVR and XGBoost, are very suitable for predicting Bank Nifty data because they can effectively adapt our data patterns and accurately predict unseen data. Although traditional statistical methods such as ARIMA have their advantages, they may not adequately capture the details of datasets like Bank Nifty.

Chapter 7

FUTURE WORK

Looking ahead, there are multiple paths to explore in order to broaden the reach and effectiveness of this research. Expanding the analysis to different areas shows promise. Our stock datasets cover a wide range of market conditions and asset classes. Through careful analysis of different forecasting methods on a range of datasets, researchers can gain valuable insights into the applicability and effectiveness of these models in various financial scenarios.

For future work by conducting thorough quantitative evaluations on new datasets, researchers can gain a more comprehensive understanding of the strengths and limitations of each approach. Through a systematic analysis of statistical models and machine learning techniques on various datasets, researchers can uncover valuable insights that can inform the selection of forecasting methodologies.

This comprehensive analysis will not only enhance the academic understanding of time series forecasting in financial markets, but also offer valuable practical guidance for investors and decision-makers. This research aims to provide stakeholders with valuable insights on the performance of different models in specific datasets.

In addition, further research could delve into more sophisticated methods for optimizing models and utilizing ensemble learning, taking advantage of the latest advancements in machine learning and artificial intelligence. With the help of advanced methodologies, researchers can enhance the accuracy and reliability of time series forecasting models, making them more valuable in practical situations.

Bibliography

[1] Ayodele Adebiyi, Aderemi Adewumi, and Charles Ayo. Stock price prediction using the arima model. 03 2014.

[2] Mohammad Almasarweh and Sadam Alwadi. Arima model in predicting banking stock market data. *Modern Applied Science*, 12:309, 10 2018.

[3] Sotirios Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis, and Nikos Vlachogiannakis. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications*, 112, 06 2018.

[4] S.S. Choudhury and M. Sen. Trading in indian stock market using ann: A decision review. *Advances in Modelling and Analysis A*, 54:252–262, 01 2017.

[5] Yulin Du. Application and analysis of forecasting stock price index based on combination of arima model and bp neural network. pages 2854–2857, 06 2018.

[6] Falah Hassan and Falah Al-akashi. Stock market index prediction using artificial neural network. *Journal of Information Technology Research*, 15:1–16, 10 2022.

[7] Denis Karya, Puspandam Katias, and Teguh Herlambang. Stock price estimation using ensemble kalman filter square root method. *Journal of Physics: Conference Series*, 1008:012017, 04 2018.

[8] Hiransha M, Gopalakrishnan E.A., Vijay Krishna Menon, and Soman K.P. Nse stock market prediction using deep-learning models. *Procedia Computer Science*, 132:1351–1362, 2018. International Conference on Computational Intelligence and Data Science.

[9] Jyothi Manoj. Forecast model using arima for stock prices of automobile sector; international journal of research in finance and marketing; issn: 2231 - 5985, 11 2017.

[10] Venkata Vara Prasad, Srinivas Gumparthi, Lokeswari Y Venkataramana, S Srinethe, R M Sruthi Sree, and K Nishanthi. Prediction of Stock Prices Using Statistical and Machine Learning Models: A Comparative Analysis. *The Computer Journal*, 65(5):1338–1351, 07 2021.

[11] James M. Rankin. Kalman filtering approach to market price forecasting. 1986.

[12] Dev Shah, Haruna Isah, and Farhana Zulkernine. Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies*, 7:26, 05 2019.

[13] Aakanksha Sharaff and M. Ali Choudhary. Comparative analysis of various stock prediction techniques. *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 735–738, 2018.

[14] Yuan Song. *Stock trend prediction: Based on machine learning methods*. PhD thesis, UCLA, 2018.

[15] B. W. Wanjawa and L. Muchemi. Ann model to predict stock prices at stock exchange markets, 2014.