# BCSE497J - Project-I

## Employability of Machine Learning tools in developing a prediction model for Nifty 50 Stocks


**21BCB0026 Arnav Goenka**

**21BCB0097 Sparsh Anand**

**21BCT0314 Aryan Singh**

Under the Supervision of

**Ushus Elizebeth Zachariah**

Assistant Professor Sr. Grade 1

School of Computer Science and Engineering (SCOPE)


**B.Tech.**

*in*

**Computer Science and Engineering**

## School of Computer Science and Engineering


November 2024

# ABSTRACT

The growth of dependence on machine learning (ML) tools in forecasting market tendencies, especially with respect to Nifty 50 stocks, stems from the fact that the algorithms can process huge amounts of data and find meaningful patterns. It is this problem that this paper seeks to address by examining the performance of more complex machine learning algorithms and techniques, specifically Long Short-Term Memory (LSTM) to predict stock prices. Predictive models are built using historical stock prices, moving averages and volatility incorporating both long-range trend and short-range movements of the market.

Data collection and preprocessing (data cleaning, normalization, feature engineering) is another important part of the present study. This was indeed the situation as the model that employed LSTM outperformed other models particularly due the use of sequential stock data. Various metrics including the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-squared ($R^2$) were used in the study to assess performance of the model, and it was revealed that in stock price prediction, the LSTM based model proved to be very accurate.

Based on case studies and performing analysis of data provided through investigation, this paper illustrates that combining machine learning models particularly LSTM architecture predict stock prices of Nifty 50 with better accuracy than regression alone. Still further improvement is possible by evolution of more sophisticated feature selection methods and hybrid model design. This the research finds out, points out to the advantage of the use of ML and its implications in forecast time series for the benefit of the investors and financial analysts.



*Keywords - Machine Learning, LSTM, Stock Price Prediction, Nifty 50, Feature Engineering, Time-series Data, Financial Forecasting, MAE, RMSE, R-squared.*

# TABLE OF CONTENTS

# List of Figures

# List of Abbreviations

**ML** - Machine Learning

**LSTM** - Long Short-Term Memory

**MAE** - Mean Absolute Error

**RMSE** - Root Mean Squared Error

**$R^2$** - R-squared

# 1. INTRODUCTION

## 1.1 Background

The stock market has always fascinated investors and researchers due to its ever-changing dynamics. For a new investor the large number of technical indicators, charts, candlesticks etc. could be overwhelming. A small fluctuation in price of stock could lead to substantial financial gains or losses. Therefore, one of the important factors to consider while investing money is the prediction of stock prices and that of an index as broad as Nifty 50 is even more required to make wise decisions. With the rise of computational power and the availability of vast historical stock data, machine learning (ML) has emerged as a powerful tool for enhancing prediction accuracy. Traditional methods often fall short in capturing the non-linear patterns in stock prices, which is where ML models, particularly those designed for time-series analysis, come into play.

The Long Short-Term Memory LSTM network is particularly suitable for sequential data, especially if it has long-range dependencies such as those found in text. Older models do not seem to have enough capacity to represent a time series like the underlying price of a stock, which is a gel of several attributes including other prices, the direction of the market, and volatility, LSTM does. Utilizing such models, the aim of this project is to build an effective solution that is able to predict the Nifty 50 stock price utilizing both past and technical analysis.

This project is important as it aims to improve the precision of stock market predictions that is of great concern to traders, analysts, and investors. By using LSTM and other ML techniques, it seeks to bridge the gap between traditional forecasting methods and the advanced capabilities of modern machine learning algorithms.

## 1.2 Motivation

Stock market prediction has long been a complex yet crucial aspect of financial markets. The dynamic and volatile nature of stock prices makes it challenging to predict with traditional methods, motivating the need for advanced techniques. The rapid advancement of machine learning (ML) tools offers a promising approach to tackle this complexity, providing opportunities for more accurate, data-driven predictions.

Nifty 50 is a crucial index of the Indian economy. Its relevance to the economy increases its potential as an economic indicator. The movement of the Nifty index can be expected to prove advantageous to investors, financial institutions, and decision-makers. This project is based on the idea that there is a possibility of taking advantage of machine learning algorithms, including lstm and others, which target complex historical stock data and projects future. This will enhance quicker and more efficient decision making, improve trading

systems and lower the chances of financial loss, thus providing a new approach in the area of finance analytics. This project seeks to combine the fundamentals of traditional statistical techniques and modern strategies in machine learning in a bid to increase the accuracy of forecasting the Nifty 50 index.

## 1.3 Scope of the Project

The scope of this project revolves around employing machine learning (ML) techniques, particularly time-series forecasting models like Long Short-Term Memory (LSTM), to predict the stock prices of the Nifty 50 index. The project will focus on analyzing historical stock data, technical indicators (such as Moving Averages, RSI, and Volatility), and market trends to build an effective prediction model.

The data used will primarily be sourced from Yahoo Finance, covering the Nifty 50 index and its constituent companies over a defined period. The model will be trained and evaluated using appropriate evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to ensure accuracy. The scope is limited to supervised learning techniques for regression tasks, avoiding complex ensemble or hybrid models.

The project will not include algorithmic trading systems or incorporating any external parameters such as sentiment or economics analysis. The end result will be a model that helps to project stock movements within certain parameters but will not constitute entire investment strategies or system integration.

# 2. PROJECT DESCRIPTION AND GOALS

This project aims to develop an advanced trading strategy for Nifty 50 stocks, utilizing unsupervised learning techniques for stock price analysis. The primary goal is to identify patterns and trends within stock data to inform trading decisions and optimize investment strategies.

Key objectives include:

- **Data Collection and Preprocessing:** Efficiently gather historical stock price data from reliable sources, ensuring the dataset is clean and ready for analysis.
- **Feature Engineering:** Create meaningful features that can enhance the performance of unsupervised learning algorithms.
- **Performance Evaluation:** Assess the effectiveness of the trading strategy through backtesting against historical data to validate its profitability.

## 2.1 Literature Review

Several research studies have explored the use of machine learning models for stock price prediction. Malti Bansal and Apoorva Choudhary (2022) [1]employed KNN, Linear Regression, SVR, DTR, and LSTM models using stock prices from 12 Indian companies. They found that LSTM outperformed other algorithms but highlighted the high computational cost and time-series data limitations. Future work could focus on hybrid models to improve performance and accuracy.

Wang et al. (2022)[2] used LSTM, CNN, and SVM, incorporating technical indicators and correlation graphs. Their spatiotemporal multi-graph CNN model reduced complexity, but challenges remain in large-scale implementations. The research suggests reducing model complexity and enhancing computational efficiency.

Sharma and Juneja (2023)[3] combined LS-Boost with Random Forest (RF) and SVR, demonstrating better accuracy in predicting stock index values. However, their model's performance relied heavily on the selection of technical indicators. They propose improving feature selection methods and incorporating external factors like sentiment analysis for better predictions.

Rao, Srinivas, and Mohan (2024)[4] focused on Artificial Neural Networks (ANN), Hidden Markov Models (HMM), Autoregressive Integrated Moving Average (ARIMA), and Recurrent Neural Networks (RNN) to predict stock price movements. Their study found that ANN performed better in long-term predictions, while ARIMA struggled with non-linear stock price patterns. The authors suggested developing hybrid models that combine statistical and neural network approaches, which could offer more robust and comprehensive predictions.

Zahra Fathali et al. (2022)[5] explored the performance of LSTM, CNN, and RNN in predicting stock prices for the Nifty 50 index, covering 12 sectors. LSTM was found to provide the lowest error rates compared to RNN and CNN. However, both LSTM and RNN lagged when it came to capturing short-term stock price volatility. To overcome these

challenges, the study suggests exploring hybrid models and improved hyperparameter optimization techniques to enhance the models' ability to capture rapid market changes.

## 2.2 Research Gap

Despite significant advancements in stock market prediction using machine learning models, there remain critical gaps that hinder accurate forecasting, especially for large indices like the Nifty 50. Most existing research focuses on using traditional methods such as linear regression, decision trees, or basic neural networks, which struggle with the highly non-linear and volatile nature of stock prices. These models often fail to account for short-term fluctuations and complex dependencies between various technical indicators and market trends.

Additionally, while LSTM models have been employed in several studies for time-series prediction, their application to stock markets, particularly the Nifty 50, remains limited. Many approaches overlook important macroeconomic indicators, sector-specific influences, or the interaction between multiple time scales. Furthermore, existing models often suffer from high computational costs and struggle to handle real-time predictions effectively.

This project aims to address these gaps by implementing a more comprehensive and efficient LSTM-based prediction model that leverages a broader range of technical and fundamental indicators. By focusing on enhancing accuracy and reducing computational complexity, this project seeks to provide a more practical and reliable solution for stock market prediction, particularly for the Nifty 50 index.

## 2.3 Objectives

**Develop a Predictive Model for Nifty 50 Stocks:** The primary goal is to build a machine learning model, specifically utilizing LSTM (Long Short-Term Memory), to predict closing prices of stocks in Nifty 50 index. The model will leverage both historical stock price data and technical indicators to ensure accuracy and reliability.

**Incorporate historical data with technical indicators:** Improve stock price prediction accuracy by incorporating a wide range of technical indicators such as moving averages, volatility, and relative strength index (RSI). This will help to capture both short-term price movements and long-term trends.

**Optimize Computational Efficiency:** Create a model that balances high prediction accuracy with computational efficiency, making it practical for real-time stock market forecasting. This will include techniques such as hyperparameter tuning and feature selection. This will ensure the model is neither underfitted nor overfitted, to reduce bias and variance.

**Evaluate Model Performance Against Benchmark Algorithms:** Compare the LSTM model with other traditional algorithms like Support Vector Regression (SVR), Decision Trees, and Linear Regression to validate its effectiveness in predicting Nifty 50 stock prices.

**Explore Model Scalability:** Test the model's ability to scale and predict prices across multiple sectors within the Nifty 50 index, ensuring its applicability in broader financial markets.

## 2.4 Problem Statement

The Nifty 50 index is considered the standard or benchmark for the stock market of India as it comprises the 50 biggest and most actively traded companies. Accurate prediction of stock price movements within this index has been a difficult task for both institutional as well as retail investors as it involves the use of complex, dynamic and non-linear financial markets. Most conventional econometrics and machine learning approaches usually find it difficult to comprehend the dynamics surrounding stock price changes especially at extreme volatile phases(e.g.- 2008 market crash)

This project aims to address these issues by employing advanced machine learning tools, specifically Long Short-Term Memory (LSTM) networks, which have shown promise in modeling sequential data. By leveraging historical stock price data and technical indicators, the project seeks to develop a robust predictive model for stock price movements within the Nifty 50 index.

The primary problem this project tackles is the need for a reliable, real-time predictive model that can assist investors in making informed decisions while minimizing risks. The focus will be on improving prediction accuracy and exploring methods to enhance the model's computational efficiency for real-time use.

## 2.5 Project Plan

| | | | | |
|---|---|---|---|---|
| Task 1 | Literature Survey | | | |
| Task 2 | Import Stock Price Dataset | | | |
| Task 3 | | Feature Engineering | | |
| Task 4 | | Train Data using LSTM | | |
| Task 5 | | Test Data | | |
| Task 6 | | | Predicted Stock Price | |

| | | | | |
|---|---|---|---|---|
| Task 7 | | | | Evaluating model performance |
| Task 8 | | | | Full Document preparation about Project |
| Task 9 | | | | Submission of Project |

Fig. 1. Gantt chart

# 3. TECHNICAL SPECIFICATION

The technical specifications outline the tools, technologies, and methodologies that will be employed throughout the project. The strategy will leverage various programming languages, libraries, and frameworks tailored for data analysis and machine learning.

- Programming Languages: Python will be the primary language for implementing the trading strategy due to its rich ecosystem of libraries for data manipulation and analysis.
- Libraries: Key libraries such as pandas, numpy, matplotlib, and scikit-learn will be utilized for data processing, visualization, and machine learning tasks.
- Data Source: Historical stock prices will be sourced from the Yahoo Finance API using the yfinance library, providing a comprehensive dataset for analysis.

## 3.1 Requirements

To successfully execute the project, the following requirements must be met:

- Software Requirements:
    - Python 3.8 or higher
    - Jupyter Notebook or any Python IDE (e.g., PyCharm, VS Code)
    - Libraries: pandas, numpy, matplotlib, scikit-learn, yfinance
- Hardware Requirements:
    - Minimum 8 GB RAM for efficient data processing.
    - A multi-core processor to expedite computations during clustering and backtesting phases.
- Human Resources:
    - A data analyst or data scientist with expertise in financial analysis and machine learning techniques to develop and validate the trading strategy.

### *3.1.1 Functional*

1. **Data Input:**
    - The model shall accept input data in a specified format (e.g., CSV, JSON) containing historical stock prices, technical indicators, and user-specified parameters (ticker, quantity, buying price).
2. **Data Processing:**
    - The model shall preprocess the input data, including normalization and handling missing(null value) values.

- It shall calculate relevant features (technical indicators like moving averages, RSI, MACD) based on the input data.

3. **Model Training:**
    - The model shall implement a Long Short-Term Memory (LSTM) architecture to predict stock closing prices.
    - It shall allow for hyperparameter tuning to optimize model performance.
4. **Prediction:**
    - The model shall output predicted closing prices for the specified stocks based on the input parameters.
    - It shall compare predicted portfolio values with actual closing prices, presenting a summary of discrepancies.
5. **Evaluation Metrics:**
    - The model shall calculate and display evaluation metrics, including accuracy, precision, and other relevant performance measures.
    - It shall display graphs to better visualize the discrepancies between the actual closing and predicted closing price.

### *3.1.2 Non-Functional*

1. **Performance:**
    - The model shall execute predictions within a reasonable time frame (e.g. :, under 5 seconds for a typical input dataset).
    - The model shall train on the data within reasonable time(e.g.:under 2 minutes for 100 epochs)
2. **Scalability:**
    - The model shall be designed to handle an increasing amount of input data without significant degradation in performance.
3. **Reliability:**
    - The model shall maintain consistent performance across various datasets and should be robust against overfitting.
4. **Maintainability:**
    - The codebase shall follow best practices for readability and organization, allowing for easy updates and modifications in the future(e.g. commenting)

5. **Documentation:**
    - Comprehensive documentation shall be provided, including setup instructions, usage guidelines, and descriptions of the model's architecture and algorithms.

## 3.2 Feasibility Study

The feasibility of the project will be assessed through three main dimensions: technical feasibility, operational feasibility, and economic feasibility.

1. Technical Feasibility:
    - The required technology stack is readily available and widely supported, with ample documentation and community resources for troubleshooting.
    - The availability of robust libraries for financial data analysis in Python ensures that the project can be executed effectively.
2. Operational Feasibility:
    - The project aligns with the current trends in finance where data-driven decision-making is increasingly adopted.
    - The skills required for project implementation are accessible, and the project can be completed within a reasonable timeframe.
3. Economic Feasibility:
    - The project requires minimal investment as open-source tools and libraries will be utilized.
    - Potential returns from optimized trading strategies justify the initial resource allocation.

### *3.2.1 Technical Feasibility*

- Model Development: The machine learning model is built using Python libraries such as TensorFlow and Keras for developing and training the LSTM model. The technical expertise available in using these libraries ensures that the project is technically feasible.
- Data Availability: Historical stock price data is available from various financial APIs and databases (e.g., Yahoo Finance, Google Finance, Alpha Vantage), making it feasible to gather the required dataset for training the model.
- Computational Resources: The project requires moderate computational resources, which can be fulfilled by standard personal computers or cloud services (e.g., AWS, Google Cloud) for training the model.

- Integration: The model can be integrated into a user-friendly interface (e.g., web application) using frameworks like Flask or Django, ensuring that users can easily interact with the predictions.

### *3.2.2 Economic Feasibility*

- Cost Analysis: The costs associated with the project include expenses for computational resources (if cloud services are used), data acquisition (if premium data sources are required), and potential licensing fees for libraries or frameworks. Currently only cost associated is the computing cost to run LSTM model and internet cost to fetch stock data from yfinance.
- Return on Investment (ROI): If the model successfully predicts stock prices, it could provide significant financial benefits to users, thereby justifying the initial investment. Potential users include retail investors and financial analysts, who may save time and improve investment strategies.
- Funding Options: The project could explore sponsorship from educational institutions or collaborations with financial organizations to offset costs and enhance credibility.

### *3.2.3 Social Feasibility*

- User Acceptance: The model aims to assist investors in making informed decisions, which is likely to be well-received in a society increasingly relying on technology for financial management.
- Ethical Considerations: The project will ensure that all data used is ethically sourced, and the model will provide transparency in its predictions to foster trust among users.
- Impact on Employment: While there may be concerns about job displacement in the finance sector, the model is designed to assist rather than replace financial professionals, enhancing their capabilities rather than diminishing their roles.

## 3.2 System Specification

The system will consist of various components to ensure the effective functioning of the trading strategy. Below are the key specifications:

- Data Ingestion Module:
    - Responsible for collecting and preprocessing data from the Yahoo Finance API.
    - Ensures data quality and consistency before analysis.
- Analysis Module:
    - Contains algorithms for feature engineering and unsupervised learning techniques.

    - Implements K-Means clustering to categorize stocks based on performance indicators.
- Optimization Module:
    - Uses portfolio optimization techniques to determine the best allocation of resources across selected stocks.
    - Employs metrics such as the Sharpe Ratio to evaluate risk-adjusted returns.
- Visualization Module:
    - Provides visual representations of data trends and clusters to facilitate decision-making.
    - Uses matplotlib and seaborn for creating informative graphs and charts.
- Reporting Module:
    - Generates comprehensive reports summarizing the performance of the trading strategy.
    - Allows stakeholders to evaluate the effectiveness of the strategy and make informed investment decisions.

### *3.2.1 Hardware Specification*

- **Processor**: A minimum of Intel i5 or equivalent for local development; for model training, a GPU (e.g., NVIDIA GTX 1660 or higher) is recommended to accelerate training time.
- **Memory**: At least 8 GB of RAM for running the model and handling data efficiently; 16 GB or more is preferred for larger datasets.
- **Storage**: Sufficient SSD storage (256 GB or more) to handle datasets, model files, and libraries efficiently.
- **Network**: A stable internet connection for data fetching(from yfinance).

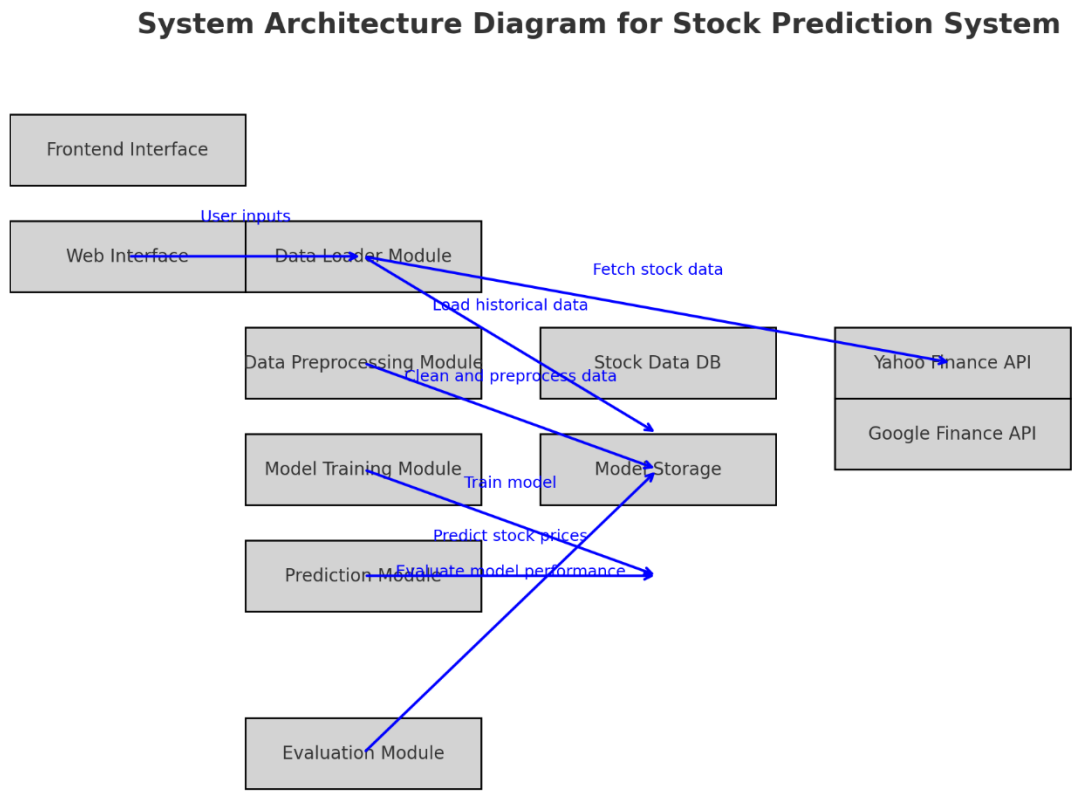
### *3.2.2 Software Specification*

- **Programming Language**: Python, as it is widely used for machine learning projects and has extensive libraries available.
- **Libraries**:
    - TensorFlow and Keras for building and training the LSTM model.
    - Pandas for data manipulation and analysis.
    - NumPy for numerical computations.
    - Matplotlib or Seaborn for data visualization, Cufflinks for interactive plots
    - SciKit Learn for evaluation

- **Development Environment**:
    - Jupyter Notebook or an Integrated Development Environment (IDE) like PyCharm for development and experimentation.
- **Database Management**: Pandas Data Frame used to store historical data and predictions ,SQLite or PostgreSQL persistent storage is required.
- **Version Control**: Git for source code management and collaboration.

# 4. DESIGN APPROACH AND DETAILS

## 4.1 System Architecture




1. **Frontend Interface**:
    - **Web Interface**: Users input stock tickers and view predictions through a web dashboard.
2. **Backend (Application Logic)**:
    - **Data Loader Module**: Fetches historical stock data from external APIs (Yahoo Finance, Google Finance).
    - **Data Preprocessing Module**: Cleans and prepares the data, including feature engineering with technical indicators.
    - **Model Training Module**: Trains the LSTM model on the processed data.
    - **Prediction Module**: Uses the trained model to predict future stock prices.
    - **Evaluation Module**: Assesses model performance using metrics like accuracy.
3. **Database (Storage)**:
    - **Stock Data DB**: Stores historical stock data for reuse.
    - **Model Storage**: Saves trained models for future predictions.
4. **External Data Sources**:

- APIs provide real-time and historical stock data to ensure accuracy in predictions.

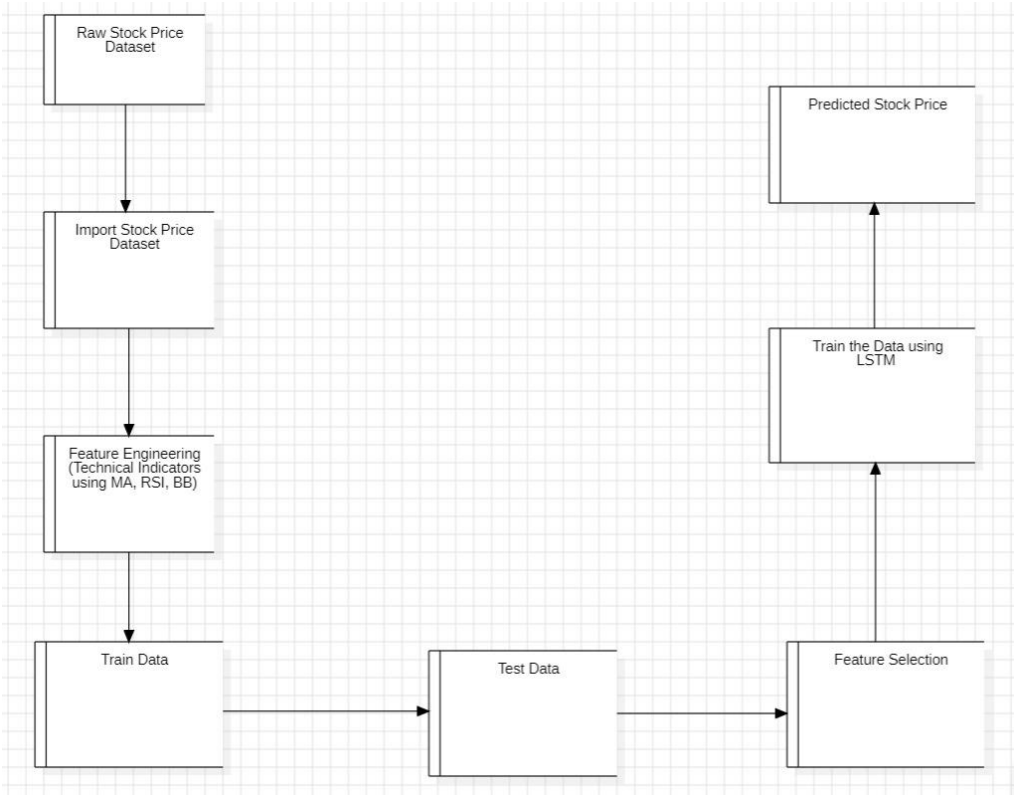
**Data Flow:**

User inputs trigger data retrieval, processing, model training, and prediction, resulting in a seamless flow of information from user input to stock price predictions.

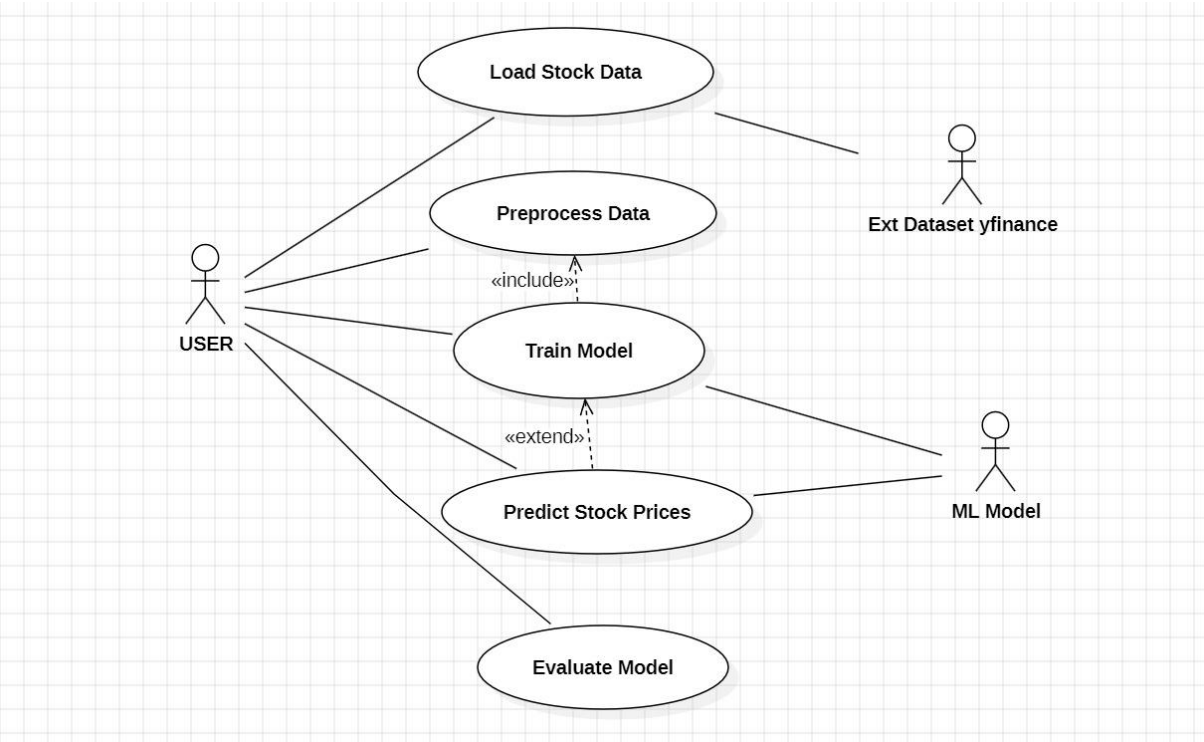
This architecture promotes modularity and maintainability, allowing for easy updates and enhancements over time.

## 4.2 Design

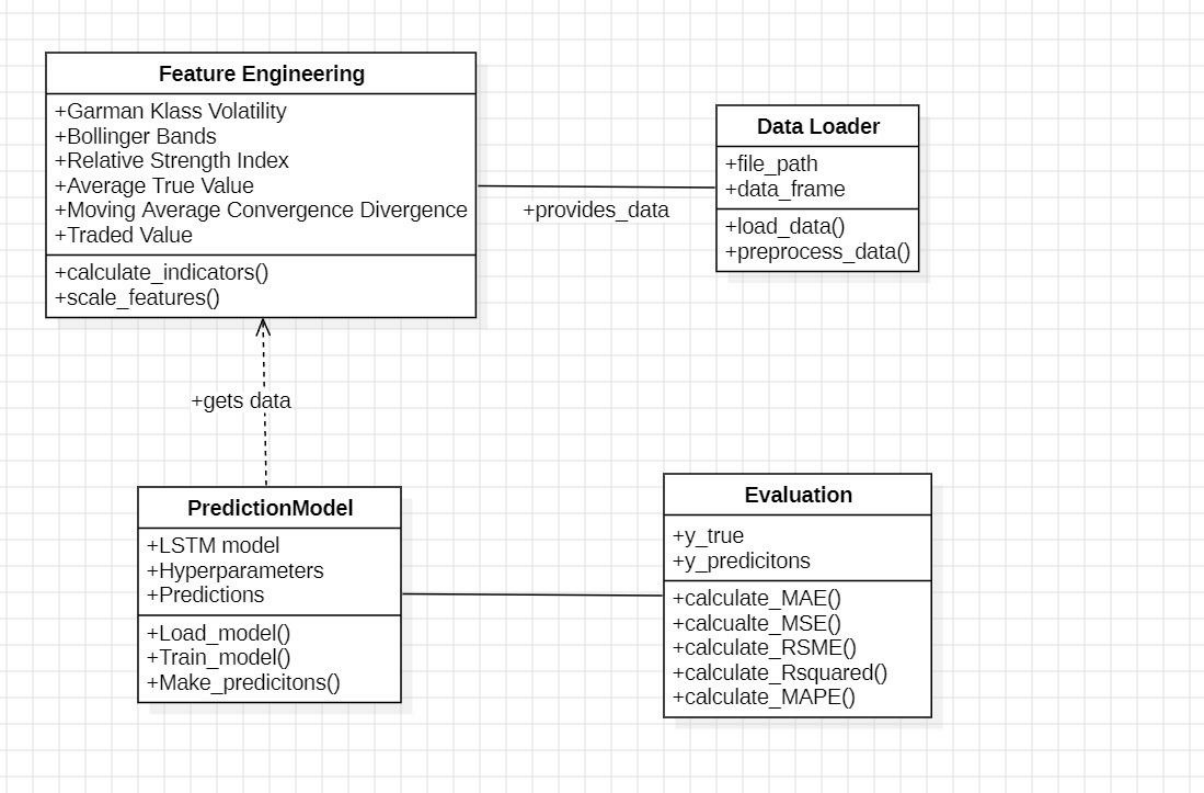
### *4.2.1 Data Flow Diagram*



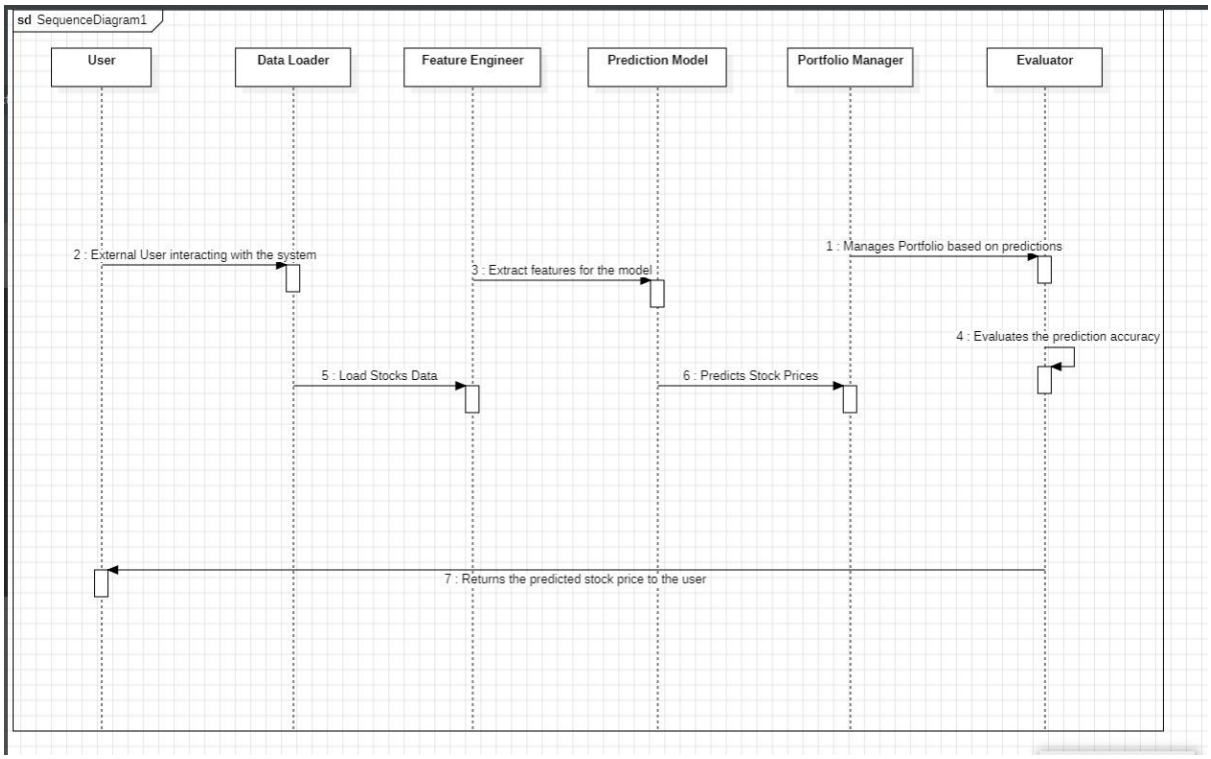
### *4.2.2 Use Case Diagram*




### *4.2.3 Class Diagram*



### *4.2.4 Sequence Diagram*



# 5. METHODOLOGY AND TESTING

## METHODOLOGY:

**Data Set:**
Stocks data imported from yfinance(yahoo finance)

**Programming Language**: Python, as it is widely used for machine learning projects and has extensive libraries available used for data manipulation, cleaning, feature engineering, machine learning and testing.

**Libraries:**

- statsmodels: For statistical models and rolling regression analysis.
- pandas_datareader: For accessing data from various online sources.
- matplotlib: For data visualization .
- pandas: For data manipulation and analysis .
- numpy: For numerical computations .
- datetime: For handling date and time operations .
- yfinance: For downloading financial data .
- pandas_ta: For technical analysis indicators and calculations.
- tensorflow: For building and training neural network models (Sequential, LSTM, Dense, Dropout, tf).
- scikit-learn: For data preprocessing

**Development Environment**:
Jupyter Notebook and Visual Studio Code

**Workflow:**

**1. Data Input:**
A) Import stocks data from Yahoo Finance for the date range January 2014 to January 2024 (10 years of data). This data contains various stock matrices like open price, close price, adjusted close price, volume, high, low.
B) Convert the data into a multi-level indexed DataFrame with the first level as the date and the second level as the stock ticker for further preprocessing and feature engineering.

| | Price | adj close | close | high | low | open | volume |
|---|---|---|---|---|---|---|---|
| date | ticker | | | | | | |
| 2014-01-01 | ADANIENT.NS | 37.933193 | 41.192673 | 41.368938 | 40.204048 | 40.433960 | 7564701.0 |
| | ADANIPORTS.NS | 147.760315 | 156.300003 | 157.750000 | 154.550003 | 154.550003 | 851727.0 |
| | APOLLOHOSP.NS | 929.205322 | 969.299988 | 987.650024 | 940.250000 | 940.250000 | 335988.0 |
| | ASIANPAINT.NS | 458.621399 | 499.750000 | 503.899994 | 492.000000 | 494.399994 | 1866326.0 |
| | AXISBANK.NS | 247.552444 | 258.440002 | 261.000000 | 257.640015 | 260.299988 | 2849425.0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 2024-07-22 | TCS.NS | 4276.916016 | 4287.350098 | 4319.950195 | 4265.000000 | 4299.950195 | 1896386.0 |
| | TECHM.NS | 1482.236206 | 1495.550049 | 1505.849976 | 1476.099976 | 1479.949951 | 2225781.0 |
| | TITAN.NS | 3254.449951 | 3254.449951 | 3273.399902 | 3223.199951 | 3250.000000 | 696255.0 |
| | ULTRACEMCO.NS | 11447.811523 | 11515.700195 | 11570.000000 | 11231.299805 | 11289.900391 | 573833.0 |
| | WIPRO.NS | 505.799988 | 505.799988 | 526.750000 | 501.549988 | 521.000000 | 38007284.0 |

**2. Feature Engineering:** Calculating various technical indicators to incorporate in model to enhance the predictive power of the model by adding diverse perspectives on market dynamics and improving the robustness and reliability of predictions. The technical indicators we calculated and incorporated in our model are as follows-

**i) Garman-Klass Volatility (garman_klass_vol):**

Garman-Klass Volatility is a widely recognized metric for assessing market volatility, specifically designed to utilize a broader spectrum of price data within a given period, including high, low, open, and close prices. Developed by Garman and Klass in 1980, it refines the traditional range-based methods by integrating more data points to provide a more accurate estimate of volatility.

Formula:

The Garman-Klass estimator is mathematically expressed as:

$$GK = \frac{1}{n}\sum_{i=1}^{n}\left[0.5 \cdot (\ln(H_i/L_i))^2 - (2\ln(2) - 1) \cdot (\ln(C_i/O_i))^2\right]$$

Where:

- $H_i, L_i, O_i, C_i$ are the high, low, open, and close prices for the $i$-th period, respectively.
- $n$ is the number of observations in the dataset.

Significance in our model:

**1. Enhanced Volatility Analysis**

The inclusion of high, low, open, and close prices allows Garman-Klass Volatility to capture more intricate price fluctuations within a given period. Unlike simple measures such as standard deviation, this indicator accounts for intraday price ranges and closing behavior. In prediction models, this enhanced granularity provides:

- **Comprehensive Insights:** It reflects the intensity and variability of price movements, helping the model differentiate between stable and volatile periods.
- **Refined Understanding of Market Dynamics:** By integrating intraday data, the model gains a deeper understanding of price behavior, making predictions more robust.

**2. Risk Assessment**

Volatility is a key determinant of market risk. Higher volatility often signals greater uncertainty, which investors may interpret as higher risk. Garman-Klass Volatility is particularly valuable in this context because:

- **Accurate Risk Profiling:** By leveraging additional data points, it provides a more nuanced risk assessment compared to traditional methods.
- **Impact on Predicted Price Trends:** Risk factors are a critical component in pricing models, as they influence investor behavior, trading strategies, and ultimately, the stock's performance.

**3. Trend Identification**

Identifying trends in market behavior is essential for timing entry and exit points. Garman-Klass Volatility aids this by:

- **Highlighting Stability vs. Volatility Periods:** It can clearly differentiate between periods of low and high volatility, enabling investors to identify when the market is consolidating or experiencing significant price swings.
- **Facilitating Strategic Decision-Making:** For instance, during periods of high volatility, the model might predict larger price movements, signaling potential opportunities or risks for traders.

**ii) Relative Strength Index (RSI) (rsi):**

The **Relative Strength Index (RSI)** is a widely used momentum oscillator that evaluates the speed and magnitude of price changes over a specified period. By oscillating between a scale of 0 and 100, RSI helps identify whether an asset is overbought or oversold, providing key insights into potential market reversals or continuations. Developed by J. Welles Wilder in 1978, RSI has become a cornerstone for technical analysis.

### Formula:

The RSI is calculated using the following steps:

1. **Average Gain and Average Loss:** Over a defined period $n$, calculate the average gain ($AvgGain$) and average loss ($AvgLoss$):

$$AvgGain = \frac{\text{Sum of Gains over the last } n \text{ periods}}{n}$$

$$AvgLoss = \frac{\text{Sum of Losses over the last } n \text{ periods}}{n}$$

2. **Relative Strength (RS):** Compute the relative strength ($RS$) as the ratio of average gain to average loss:

$$RS = \frac{AvgGain}{AvgLoss}$$

3. **RSI Calculation:** Using the $RS$, compute the RSI:

$$RSI = 100 - \left(\frac{100}{1 + RS}\right)$$

The standard calculation uses a 14-period lookback window, but this can be adjusted based on specific trading strategies or analysis needs.

**Key Thresholds:**

- **Overbought Condition (>70):** Indicates that the asset may be overvalued and due for a corrective pullback.

- **Oversold Condition (<30):** Suggests that the asset may be undervalued and poised for a price recovery.

**1. Momentum Insight**

- **Purpose:** RSI quantifies the momentum of recent price movements, providing the model with a clear understanding of whether the asset is experiencing strong upward or downward momentum.
- **Implementation in the Model:**
  - During strong trends (e.g., RSI > 70 in uptrends), RSI helps the model recognize sustained momentum.
  - For fading trends (e.g., RSI crossing below 50), the model can identify potential weakness.
- **Impact:** Enhances the model's ability to detect the underlying strength or weakness in price trends, leading to more informed predictions.

**2. Reversal Predictions**

- **Purpose:** RSI is particularly effective at signaling potential price reversals by crossing critical thresholds.
- **Key Patterns for Reversal Detection:**
  - **Bullish Divergence:** When the price makes lower lows, but RSI makes higher lows, indicating potential upward reversal.
  - **Bearish Divergence:** When the price makes higher highs, but RSI makes lower highs, suggesting potential downward reversal.
- **Implementation in the Model:**
  - By monitoring RSI crossing thresholds (e.g., 70 to 69 or 30 to 31), the model can predict changes in market direction.
- **Impact:** Improves the model's forecasting ability for short-term price changes, particularly in volatile markets.

**3. Improved Model Performance**

- **Purpose:** RSI captures short-term market dynamics, complementing other trend-following or volatility-based indicators.
- **Benefits:**
  - **Granularity:** RSI enhances sensitivity to market conditions, enabling the model to adjust predictions for sudden momentum shifts.
  - **Compatibility:** Combines effectively with longer-term trend indicators (e.g., moving averages) to provide a balanced perspective.
- **Impact:** Leads to more robust predictions by integrating diverse market characteristics such as momentum and overbought/oversold conditions.

**Practical Applications in Financial Models:**

1. **Trend Confirmation:**
   - RSI helps validate signals from other technical indicators, ensuring that predictions align with current market momentum.
2. **Risk Management:**
   - By identifying overbought or oversold levels, RSI aids in defining entry and exit points, minimizing the risk of entering trades at extreme price levels.
3. **Dynamic Adjustments:**
   - Models using RSI can adapt to rapidly changing market conditions, providing timely alerts for investors or traders.

**iii) Bollinger Bands (bb_low, bb_mid, bb_high):**

**Bollinger Bands** are a popular technical indicator that consists of three lines plotted relative to the price of an asset. They provide a visual representation of price volatility by dynamically adjusting to market conditions. Created by John Bollinger in the 1980s, Bollinger Bands help traders and models assess overbought or oversold conditions, trend strength, and potential reversal points.

### Components and Formula:

1. **Mid-Band (Moving Average):**
   - The mid-band is typically a simple moving average (SMA) calculated over a specified period $n$:

$$BB_{mid} = SMA = \frac{1}{n}\sum_{i-1}^{n} P_i$$

   Where $P_i$ is the price of the asset at period $i$.

2. **Standard Deviation ($\sigma$):**
   - The standard deviation measures the dispersion of prices from the moving average over $n$ periods:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i-1}^{n}(P_i - BB_{mid})^2}$$

3. **Upper Band (BB_high):**
   - The upper band is calculated as:

$$BB_{high} = BB_{mid} + k \cdot \sigma$$

   Where $k$ is the multiplier, typically set to 2.

4. **Lower Band (BB_low):**
   - The lower band is calculated as:

$$BB_{low} = BB_{mid} - k \cdot \sigma$$

By default, $n$ is often set to 20 (20-day SMA), and $k = 2$, meaning the bands are positioned two standard deviations above and below the mid-band.

**Key Features:**

- **Band Width:** The distance between the upper and lower bands widens during periods of high volatility and narrows during periods of low volatility.
- **Dynamic Nature:** Unlike static support and resistance levels, Bollinger Bands adjust in real-time based on price action, making them effective in both trending and ranging markets.

**Significance in a Prediction Model:**

**1. Volatility Context**

- **Purpose:** Bollinger Bands dynamically reflect market volatility by expanding or contracting based on price movements.
- **Implementation in the Model:**
  - **Wider Bands:** Indicate high volatility, suggesting increased market activity or uncertainty.

    - **Narrower Bands:** Indicate low volatility, often preceding significant price movements (known as the "squeeze" effect).
- **Impact:** Enables the model to identify periods of heightened risk or consolidation, improving its ability to forecast potential breakouts or trends.

**2. Trend Reversal Signals**

- **Purpose:** Price interaction with the bands provides valuable insights into potential reversals or trend continuations.
- **Key Patterns:**
    - **Touching or Breaching the Upper Band:** May indicate overbought conditions, suggesting a potential reversal to the downside.
    - **Touching or Breaching the Lower Band:** May signal oversold conditions, suggesting a potential reversal to the upside.
- **Implementation in the Model:**
    - By monitoring the frequency and duration of breaches, the model can infer the likelihood of reversals or trend exhaustion.
- **Impact:** Improves the model's ability to predict turning points, especially when used alongside other momentum or trend indicators.

**3. Support and Resistance Levels**

- **Purpose:** Bollinger Bands often act as dynamic support and resistance levels, guiding price movements within a bounded range.
- **Use in the Model:**
    - The lower band acts as support during downtrends, while the upper band acts as resistance during uptrends.
    - Price movement between the bands can provide the model with insights into potential breakout points or continuation patterns.
- **Impact:** Enhances the accuracy of price predictions by incorporating real-time dynamic price bounds.

**<u>iv) Average True Range (ATR) (atr):</u>**

**Average True Range (ATR)** is a widely used technical indicator that measures market volatility. Introduced by J. Welles Wilder in his book *New Concepts in Technical Trading Systems*, ATR provides a normalized view of how much an asset's price moves within a given period. Unlike indicators that focus solely on closing prices, ATR considers gaps between sessions, offering a comprehensive volatility measure.

### Formula and Calculation:

1. **True Range (TR):** The True Range (TR) for a single period is the maximum of the following three values:

$$TR_t = \max\left(H_t - L_t,\ |H_t - C_{t-1}|,\ |L_t - C_{t-1}|\right)$$

Where:

- $H_t$: Current high price.
- $L_t$: Current low price.
- $C_{t-1}$: Previous period's closing price.

The **True Range** accounts for:

- **Intraday movement:** $H_t - L_t$
- **Gap-ups:** $|H_t - C_{t-1}|$
- **Gap-downs:** $|L_t - C_{t-1}|$

2. **Average True Range (ATR):** The ATR is the moving average of the True Range over a specified period $n$:

$$ATR_t = \frac{1}{n}\sum_{i-1}^{n} TR_i$$

Alternatively, a Wilder's smoothing technique can be used to update ATR dynamically:

$$ATR_t = \frac{(n-1)\cdot ATR_{t-1} + TR_t}{n}$$

Here, $n$ typically defaults to 14 periods.

**Significance in a Prediction Model:**

**1. Volatility Indicator**

- **Purpose:** ATR quantifies the degree of price movement within a period, offering a normalized measure of volatility.
- **Application:**
  - **High ATR:** Indicates increased price swings, often preceding trend changes or market events.
  - **Low ATR:** Suggests consolidation or a lack of significant price movement, often preceding breakouts.
- **Impact on Model:** By incorporating ATR, the model gains insight into the magnitude of expected price fluctuations, improving its ability to adjust predictions for volatile or stable periods.

**2. Market Behavior Insight**

- **Purpose:** Helps the model understand the overall market sentiment during different volatility regimes.
- **Application:**
  - During **high volatility**, ATR signals the model to account for broader price ranges, preventing underestimation of price targets.

    - During **low volatility**, ATR helps identify periods of consolidation, often leading to breakouts.
- **Impact on Model:** Guides the model to adapt dynamically to shifting market conditions, ensuring predictions remain accurate across varying volatility levels.

**3. Signal Validation**

- **Purpose:** ATR can validate trading signals by confirming the strength of trends or filtering out false signals.
- **Application:**
    - **Strong Trend:** High ATR values confirm robust price movement, validating trend-following signals.
    - **Weak Trend:** Low ATR values suggest weak momentum, signaling caution in trend-based predictions.
- **Impact on Model:** Reduces noise by ensuring that predictions are based on statistically significant market movements rather than random fluctuations.

**v) Moving Average Convergence Divergence (MACD) (macd):**

The **Moving Average Convergence Divergence (MACD)** is a popular momentum-based technical indicator that combines trend-following and momentum principles. It analyzes the relationship between two exponential moving averages (EMAs) of an asset's price, offering insights into trend direction, momentum, and potential reversals.

### Formula and Calculation:

1. **MACD Line:** The MACD Line is the difference between a **short-term EMA** (typically 12 periods) and a **long-term EMA** (typically 26 periods):

$$MACD_t = EMA_{\text{short-term}}(t) - EMA_{\text{long-term}}(t)$$

2. **Signal Line:** A 9-period EMA of the MACD Line, known as the Signal Line, is calculated to identify potential crossovers:

$$Signal_t = EMA_{\text{MACD}}(t)$$

3. **MACD Histogram:** The MACD Histogram represents the difference between the MACD Line and the Signal Line:

$$Histogram_t = MACD_t - Signal_t$$

The histogram provides a visual representation of momentum changes, with positive values indicating upward momentum and negative values indicating downward momentum.

**Key Features:**

1. **Trend Analysis:**
    - A positive MACD Line (above zero) indicates that the short-term EMA is above the long-term EMA, suggesting an uptrend.
    - A negative MACD Line (below zero) indicates that the short-term EMA is below the long-term EMA, suggesting a downtrend.
2. **Momentum Detection:**
    - When the MACD Line crosses above the Signal Line, it suggests bullish momentum.
    - When the MACD Line crosses below the Signal Line, it suggests bearish momentum.

3. **Divergence Identification:**
    - Divergences between MACD and price action (e.g., price making higher highs while MACD makes lower highs) can signal potential trend reversals.

**Significance in Prediction Models:**

**1. Trend and Momentum Detection**

- **Purpose:** MACD helps the model detect shifts in trend direction and gauge momentum strength.
- **Application:**
    - **Bullish Crossover:** When the MACD Line crosses above the Signal Line, it suggests increasing momentum in the upward direction.
    - **Bearish Crossover:** When the MACD Line crosses below the Signal Line, it indicates downward momentum.
- **Impact on Model:** By identifying these momentum shifts, the model can anticipate future price movements with greater accuracy.

**2. Convergence and Divergence Analysis**

- **Purpose:** Divergences between the MACD Line and price can indicate weakening trends or impending reversals.
- **Application:**
    - **Bullish Divergence:** Price makes lower lows, but MACD makes higher lows, signaling potential upward reversals.
    - **Bearish Divergence:** Price makes higher highs, but MACD makes lower highs, signaling potential downward reversals.
- **Impact on Model:** Divergence signals provide additional context, enabling the model to adjust predictions before trend reversals occur.

**3. Normalization for Stability**

- **Purpose:** MACD uses the difference between two EMAs, making it less prone to short-term noise and fluctuations.
- **Application:**
    - Smooths out volatility, offering a reliable signal during noisy market conditions.
- **Impact on Model:** Adds stability and robustness, ensuring predictions remain consistent even in choppy markets.

**vi) Traded Value (traded_value):**

**Traded Value** represents the total monetary value of stocks traded over a specific period. It provides insight into market activity, liquidity, and interest in a particular stock or market segment. By combining price and volume data, it serves as a fundamental metric for analyzing market dynamics.

### Formula:

The traded value ($TV$) is calculated as:

$$TV = P \times V$$

Where:

- $P$: Price of the stock during the period.
- $V$: Volume of shares traded during the period.

For cumulative analysis over multiple periods, the traded value is aggregated:

$$TV_{\text{total}} = \sum_{i=1}^{n} P_i \times V_i$$

Where $n$ is the number of trading periods.

**Significance in Stock Market Analysis:**

1. **Liquidity Measure:**
    - High traded value indicates significant market activity and strong liquidity, making it easier for investors to buy or sell shares without significantly affecting the price.
    - Low traded value may suggest poor liquidity, indicating limited market participation or low interest in the stock.
2. **Market Sentiment Insights:**
    - A **sharp increase** in traded value often reflects heightened market interest or a significant event, such as earnings reports or announcements.
    - A **sharp decrease** might indicate waning interest or uncertainty about the stock's future performance.
3. **Trend Confirmation:**
    - Rising traded value during an **uptrend** supports the continuation of the trend, signaling strong market participation.
    - Declining traded value during price increases may indicate a lack of conviction in the trend, signaling a potential reversal.

**Use in Machine Learning Models:**

**1. Liquidity Measure:**

- **Purpose:** Traded value acts as a proxy for market liquidity, helping the model gauge the ease of trading.
- **Impact on Predictions:**
    - Stocks with high traded value are less prone to manipulation and offer reliable patterns.
    - The model can adjust predictions to reflect reduced volatility for liquid stocks and increased risk for illiquid stocks.

**2. Market Sentiment Insights:**

- **Purpose:** Traded value reflects investor sentiment and interest in the stock.
- **Impact on Predictions:**
    - Analyzing spikes in traded value can help the model predict potential breakouts or breakdowns.

- The model can use sudden surges in traded value as a signal of institutional trading activity or major news events.

**3. Trend Confirmation:**

- **Purpose:** Confirm the strength of trends to differentiate between valid movements and false signals.
- **Impact on Predictions:**
  - Rising traded value during price uptrends strengthens the model's confidence in the sustainability of the trend.
  - Low traded value during price increases signals caution, as the uptrend might lack robust market participation.

**Integration with Other Indicators:**

The **traded value** works synergistically with other indicators to enhance the model's predictive power:

1. **With RSI:**
   - A high traded value during overbought or oversold RSI conditions may strengthen reversal signals, indicating stronger sentiment shifts.
2. **With Bollinger Bands:**
   - A breakout beyond the bands, accompanied by rising traded value, confirms the significance of the price movement.
3. **With MACD:**
   - A MACD crossover, validated by increasing traded value, signals robust momentum and trend strength.
4. **With ATR:**
   - Higher traded value during increased ATR suggests strong participation during volatile periods, reducing noise in the model's predictions.

| Price | | adj close | close | high | low | open | volume | garman_klass_vol | rsi | bb_low | bb_n |
|---|---|---|---|---|---|---|---|---|---|---|---|
| date | ticker | | | | | | | | | | |
| 2014-01-01 | ADANIENT.NS | 37.933193 | 41.192673 | 41.368938 | 40.204048 | 40.433960 | 7564701.0 | -0.001167 | NaN | NaN | N |
| | ADANIPORTS.NS | 147.760315 | 156.300003 | 157.750000 | 154.550003 | 154.550003 | 851727.0 | -0.000570 | NaN | NaN | N |
| | APOLLOHOSP.NS | 929.205322 | 969.299988 | 987.650024 | 940.250000 | 940.250000 | 335988.0 | 0.001156 | NaN | NaN | N |
| | ASIANPAINT.NS | 458.621399 | 499.750000 | 503.899994 | 492.000000 | 494.399994 | 1866326.0 | -0.001894 | NaN | NaN | N |
| | AXISBANK.NS | 247.552444 | 258.440002 | 261.000000 | 257.640015 | 260.299988 | 2849425.0 | -0.000890 | NaN | NaN | N |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 2024-07-22 | TCS.NS | 4276.916016 | 4287.350098 | 4319.950195 | 4265.000000 | 4299.950195 | 1896386.0 | 0.000071 | 68.544760 | 8.221046 | 8.2964 |
| | TECHM.NS | 1482.236206 | 1495.550049 | 1505.849976 | 1476.099976 | 1479.949951 | 2225781.0 | 0.000198 | 68.006882 | 7.216091 | 7.2664 |
| | TITAN.NS | 3254.449951 | 3254.449951 | 3273.399902 | 3223.199951 | 3250.000000 | 696255.0 | 0.000119 | 43.724374 | 8.053748 | 8.1023 |
| | ULTRACEMCO.NS | 11447.811523 | 11515.700195 | 11570.000000 | 11231.299805 | 11289.900391 | 573833.0 | 0.000367 | 56.819841 | 9.295017 | 9.3491 |
| | WIPRO.NS | 505.799988 | 505.799988 | 526.750000 | 501.549988 | 521.000000 | 38007284.0 | 0.000863 | 47.918609 | 6.190867 | 6.2781 |

122298 rows × 14 columns

**3. Smoothing Out Data:**

A) **Monthly Aggregation:** Aggregate stock data to a monthly level to reduce noise and short-term fluctuations.

B) **2-Year Rolling Average:** Calculate a 2-year rolling average of traded value for each stock (with a minimum of 12 months to calculate the mean) to smooth data for long-term analysis.

| | | traded_value | adj close | atr | bb_high | bb_low | bb_mid | garman_klass_vol | macd | rsi |
|---|---|---|---|---|---|---|---|---|---|---|
| **date** | **ticker** | | | | | | | | | |
| **2014-02-28** | **ADANIENT.NS** | 50.848792 | 35.886566 | -0.672960 | 3.627210 | 3.420942 | 3.524076 | -0.000951 | -0.135196 | 57.971187 |
| | **ADANIPORTS.NS** | 28.371962 | 158.112045 | -1.008562 | 5.090981 | 4.874476 | 4.982728 | 0.001618 | 0.037979 | 63.501756 |
| | **APOLLOHOSP.NS** | 10.660637 | 882.232056 | -1.058785 | 6.819753 | 6.759809 | 6.789781 | 0.000539 | -0.319632 | 46.912792 |
| | **ASIANPAINT.NS** | 31.935277 | 434.072845 | -1.404543 | 6.110900 | 6.048375 | 6.079638 | -0.003236 | -0.305079 | 45.281445 |
| | **AXISBANK.NS** | 178.542509 | 242.734344 | -1.818114 | 5.488804 | 5.313929 | 5.401367 | 0.000172 | 0.016891 | 65.044797 |
| **...** | **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2024-07-31** | **TCS.NS** | 1460.302121 | 4276.916016 | 1.818237 | 8.371835 | 8.221046 | 8.296440 | 0.000071 | 3.100142 | 68.544760 |
| | **TECHM.NS** | 286.641176 | 1482.236206 | 1.295836 | 7.316715 | 7.216091 | 7.266403 | 0.000198 | 2.252060 | 68.006882 |
| | **TITAN.NS** | 468.239728 | 3254.449951 | 1.384021 | 8.150967 | 8.053748 | 8.102358 | 0.000119 | -1.672652 | 43.724374 |
| | **ULTRACEMCO.NS** | 478.069003 | 11447.811523 | 3.415309 | 9.403295 | 9.295017 | 9.349156 | 0.000367 | 1.934516 | 56.819841 |
| | **WIPRO.NS** | 648.222323 | 505.799988 | 2.439437 | 6.365442 | 6.190867 | 6.278155 | 0.000863 | 2.184315 | 47.918609 |

5927 rows × 9 columns

Fig: Data now on monthly level

5. **Monthly Returns Calculation:** Calculate monthly returns for different time horizons (1 month, 3 months, 6 months, 9 months, 12 months) to help model identify patterns of momentum and stock performances over different time frames

By incorporating these return calculations, the model can better understand stock behaviour over time, enhancing its predictive accuracy and ability to adapt to different market conditions.

Also we created a function to remove outliers that is values beyond a specified threshold limit, because these values might significantly reduce the accuracy of prediction.

| | returns_2m | returns_3m | returns_6m | returns_9m | returns_12m |
|---|---|---|---|---|---|
| | 0.209455 | 0.157282 | 0.068168 | 0.047383 | 0.091427 |
| | 0.041755 | 0.060570 | 0.034282 | 0.048216 | 0.062852 |
| | 0.073616 | 0.037484 | 0.017532 | 0.041530 | 0.029804 |
| | 0.035813 | 0.027206 | 0.044239 | 0.054112 | 0.046454 |
| | 0.062488 | 0.056308 | 0.061139 | 0.050687 | 0.063166 |
| | ... | ... | ... | ... | ... |
| | 0.081954 | 0.042481 | 0.021229 | 0.028774 | 0.020403 |
| | 0.113545 | 0.064305 | 0.022402 | 0.034650 | 0.027220 |
| | 0.003572 | -0.031059 | -0.020517 | 0.002601 | 0.006968 |
| | 0.077675 | 0.049152 | 0.020972 | 0.035371 | 0.027477 |
| | 0.074368 | 0.030355 | 0.009414 | 0.031987 | 0.018864 |

**6.Feature Selection and Scaling:**

**Feature Selection:** The selection of relevant technical indicators and returns is critical to provide meaningful inputs for the LSTM model. Features are chosen based on their predictive power for stock price movement. These include:

- Technical indicators like RSI, MACD, Bollinger Bands, ATR, and traded value.
- Lagged returns or price changes to capture temporal patterns.
- Features representing market sentiment and volatility.

**Scaling:** The **MinMaxScaler** is applied to scale all features and the target variable to a range between 0 and 1. This scaling ensures consistency across inputs, making it easier for the LSTM to process data and improve convergence during training. The formula for MinMax scaling is:

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

**Why Scaling Matters:**

- Prevents domination of features with larger magnitudes.
- Stabilizes the optimization process by reducing large gradients.

**7.Time steps for LSTM:** LSTMs are designed to handle sequential data by analyzing time-dependent patterns. To prepare the input data, sequences of **12 time steps** (representing 12 past periods, e.g., months or days) are created for each stock ticker.

**Preparation Process:**

1. For each ticker, sliding windows of 12 time steps are extracted to form the input matrix X[lstm].
   - Example:
     - Input sequence: $[t-11, t-10, \ldots, t]$
     - Output (target): $[t+1]$
2. The target variable ylstm corresponds to the stock price or return value at the next time step t+1.
3. This format ensures that the LSTM can learn patterns over 12 time steps and predict future values.

**8. Train-Test Split**: The dataset is divided into training, validation, and testing sets:

1. **Training Set (64% of total):**
   - Used to fit the LSTM model and learn patterns.
   - Further split into 80% training and 20% validation to monitor overfitting.
2. **Validation Set (16% of total):**
   - Used to fine-tune hyperparameters and ensure the model generalizes well to unseen data.
3. **Testing Set (20% of total):**
   - Held out for final evaluation of the model's performance on completely unseen data.

**9. LSTM Model Architecture:**

1. **Bidirectional LSTM Layer (100 units)**:
   - Processes the input sequence in **both forward and backward directions** to capture dependencies from past and future contexts simultaneously.
   - return_sequences=True ensures that the entire sequence output is passed to the next layer for deeper feature extraction.
2. **Batch Normalization**:
   - Normalizes the activations of the previous layer to stabilize and accelerate training.

   Formula:

   $$\hat{x} = \frac{x - \mu}{\sigma + \epsilon}$$

   Where:

   - $\mu$: Mean of activations.
   - $\sigma$: Standard deviation of activations.
   - $\epsilon$: Small constant to prevent division by zero.
3. **Dropout (0.2)**:
   - Randomly drops 20% of the neurons to prevent overfitting.

4. **Second LSTM Layer (100 units)**:
    - Processes data from the previous layer without returning sequences (return_sequences=False), acting as the final LSTM output.
5. **Batch Normalization**:
    - Normalizes output for stability.
6. **Dropout (0.2)**:
    - Applies another 20% dropout for further regularization.
7. **Dense Layer (50 units, ReLU activation)**:
    - Fully connected layer that learns complex, non-linear relationships in the data.
8. **Output Dense Layer (1 unit)**:
    - Final layer for regression output, predicting a single continuous value (e.g., next month's closing price).

**10.Compilation:**

- **Optimizer**: Adam with a learning rate of 0.001.
- **Loss Function**: Mean squared error for regression tasks.

**Model: "sequential"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| bidirectional (Bidirectional) | (None, 12, 200) | 92,000 |
| batch_normalization (BatchNormalization) | (None, 12, 200) | 800 |
| dropout (Dropout) | (None, 12, 200) | 0 |
| lstm_1 (LSTM) | (None, 100) | 120,400 |
| batch_normalization_1 (BatchNormalization) | (None, 100) | 400 |
| dropout_1 (Dropout) | (None, 100) | 0 |
| dense (Dense) | (None, 50) | 5,050 |
| dense_1 (Dense) | (None, 1) | 51 |

**Total params:** 218,701 (854.30 KB)

**Trainable params:** 218,101 (851.96 KB)

**Non-trainable params:** 600 (2.34 KB)

**11.Model Training:**

i)**Early Stopping:** Early stopping is a regularization technique used in training deep learning models to prevent overfitting. It works by monitoring a specific metric, such as **validation loss** (val_loss), during training.

**How it works:**

1. During each epoch, the model's performance on the validation set is evaluated.
2. If the validation loss does not improve (i.e., continues to plateau or increase) for a specified number of epochs (in this case, **15 epochs**), the training process is stopped.
3. Stopping early prevents the model from overfitting to the training data by halting training when the validation performance stops improving.

Formula:

Validation loss is computed as:

$$\text{val\_loss} = \frac{1}{n}\sum_{i-1}^{n} \text{Loss}(\hat{y}_i, y_i)$$

Where:

- $n$: Number of samples in the validation set.
- $\hat{y}_i$: Predicted value for the $i$-th sample.
- $y_i$: True value for the $i$-th sample.
- Loss is the chosen loss function, e.g., **Mean Squared Error (MSE)** for regression.

ii)**The restore_best_weights**: When **restore_best_weights=True** is set in the early stopping configuration, the model automatically reverts to the weights from the epoch where validation loss was at its minimum.

**Why it's important:**

- Without this feature, the final model weights might be from an epoch where the validation loss was not optimal.
- Restoring the best weights ensures the model achieves its best possible performance on unseen data.

iii)**Validation Split:** To ensure reliable evaluation during training, **20% of the training data** is automatically reserved as a **validation set**. This set is used to compute metrics like val_loss, which reflects how well the model generalizes to unseen data.

**Key Aspects:**

1. **Purpose:**
   - Provides an unbiased evaluation of the model's performance during training.
   - Tracks whether the model is overfitting or underfitting by comparing training loss and validation loss.
2. **Implementation:**

- A validation split of 20% means that out of the entire training dataset, 80% is used for model training, and 20% is used for validation.

iv)**Training Configuration:** The model is configured to train under the following settings:

1. **Maximum Epochs:**
   - Training can run for up to **100 epochs**, where each epoch represents one complete pass through the training dataset.
   - The number of epochs is capped to prevent excessive training, and early stopping typically terminates training before the maximum limit.
2. **Batch Size:**
   - The batch size is set to **32**, meaning the training data is divided into batches of 32 samples.
   - Smaller batches allow faster updates to weights but may introduce more noise in gradient estimates.
3. **Data Splits:**
   - The training data is automatically divided into **80% training data** and **20% validation data**, ensuring the model trains on one subset and evaluates on the other.

**12. Prediction:** Output the predicted closing prices for the specified stocks based on input parameters.

**13. Evaluation Metrics:**

**A) Display evaluation metrics including:** - Mean Absolute Error (MAE) - Mean Squared Error (MSE) - Root Mean Squared Error (RMSE) - R-squared ($R^2$) - Mean Absolute Percentage Error (MAPE) - Accuracy

**B)** Interactive graph of actual vs. predicted closing prices using Plotly.

**<u>TESTING</u>:**

We evaluated the model on testing data(20% of actual data) in terms of various evaluation metrics and got the following results:

**Model Evaluation Metrics:**

a)Mean Absolute Error (MAE): 270.2056
b)Mean Squared Error (MSE): 149170.5569
c)Root Mean Squared Error (RMSE): 386.2260
d)R-squared ($R^2$): 0.9598
e)Mean Absolute Percentage Error (MAPE): 1.49%
f)Accuracy=98.51%

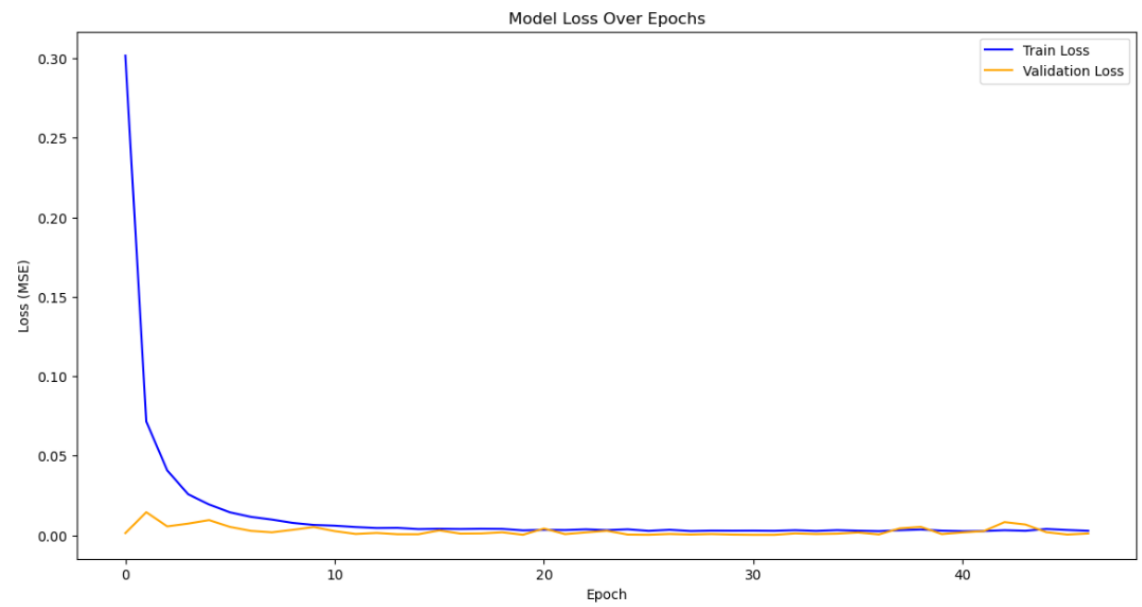



**Fig: Model Performance during training**

## MODEL DEMONSTRATION:

| Actual Closing Price | Predicted Closing Price |
|---|---|
| 333.058746 | 319.101440 |
| 375.506256 | 425.195160 |
| 414.854675 | 383.363922 |
| 590.372009 | 592.104187 |
| 789.036682 | 797.768250 |
| ... | ... |
| 616.254211 | 602.548096 |
| 644.508179 | 605.226624 |
| 631.203430 | 607.440186 |
| 581.671875 | 602.161865 |
| 595.049988 | 608.813354 |

**Visualization:**

# 6. PROJECT DEMONSTRATION

**Advanced Machine Learning Techniques for Stock Price Prediction**

**Introduction**

The financial market is one of the most dynamic and complex systems, characterized by its inherent unpredictability and susceptibility to a multitude of factors. Stock price prediction, a long-standing challenge in the domain of finance, has attracted significant attention due to its profound implications for investors, financial analysts, and economic policymakers. Traditional econometric models, while providing valuable insights, often fall short in capturing the intricate dependencies and non-linear relationships that govern stock price fluctuations.

This project delves into the application of advanced machine learning (ML) techniques to address these challenges, focusing on the use of Long Short-Term Memory (LSTM) networks, a specialized type of Recurrent Neural Network (RNN) designed to process sequential data. By leveraging the predictive power of LSTMs, this study aims to develop a robust and efficient model for forecasting stock prices within the Nifty 50 index, one of the most prominent benchmarks of economic performance in India.

**Challenges in Stock Market Prediction**

Stock market prediction is a notoriously intricate task due to the volatile and non-linear nature of price movements. Factors such as market sentiment, macroeconomic indicators, geopolitical events, and company-specific developments can significantly impact stock prices. Traditional forecasting techniques, including statistical models like ARIMA and GARCH, often fail to adapt to these rapid changes and hidden dependencies.

Investors and analysts require advanced tools capable of providing reliable predictions to minimize financial risks and maximize returns. However, traditional models struggle with:

- Capturing long-term dependencies in time-series data.
- Adapting to the highly non-linear and stochastic nature of stock price movements.
- Incorporating a wide range of market and technical indicators into the prediction process.

This project addresses these limitations by leveraging LSTM networks, which excel in handling sequential and time-series data, making them particularly suited for stock market applications.

**Relevance of the Nifty 50 Index**

The Nifty 50 index represents the 50 largest and most actively traded companies in India, making it a vital barometer of economic activity. It is widely used by investors and policymakers to gauge market sentiment and economic trends. Predicting stock prices within this index is of paramount importance for informed decision-making, given its influence on both individual investment portfolios and institutional trading strategies.

By focusing on the Nifty 50 index, this project aims to provide insights into the behavior of some of the most significant stocks in the Indian market, offering valuable guidance for market participants seeking to optimize their investment strategies.

**Machine Learning Approach: Leveraging LSTM Networks**

LSTM networks, a specialized variant of RNNs, are designed to overcome the limitations of traditional RNNs by effectively capturing long-term dependencies in sequential data. This is achieved through the use of memory cells and gating mechanisms, which allow LSTMs to retain and forget information selectively.

The project integrates LSTM networks with key technical indicators to enhance predictive accuracy. The LSTM model is particularly well-suited for this task due to its ability to:

1. **Capture Temporal Dependencies**: LSTMs process data sequentially, enabling the model to learn from historical trends and anticipate future movements.
2. **Handle Non-Linearity**: Stock price movements are highly non-linear, and LSTMs can model such complexities better than traditional methods.
3. **Adapt to Data Patterns**: By leveraging gating mechanisms, LSTMs can focus on relevant patterns while ignoring noise, ensuring more robust predictions.

**Incorporating Technical Indicators**

To enhance the model's predictive capabilities, the following technical indicators are incorporated:

- **Moving Averages**: Smoothens price data to identify trends over varying timeframes, such as short-term (10-day) and long-term (200-day) trends.
- **Relative Strength Index (RSI)**: Determines overbought or oversold market conditions, providing insights into potential reversals.
- **Bollinger Bands**: Tracks price volatility and helps identify overbought or oversold conditions by plotting price bands around a moving average.
- **Garman-Klass Volatility**: A sophisticated measure of market fluctuations that accounts for high-low price differences, offering greater accuracy than traditional volatility indicators.

These indicators capture macro-level market trends and micro-level price dynamics, providing a comprehensive dataset for the LSTM model to learn from.

**Computational Efficiency**

Given the vast amount of historical stock data, computational efficiency is a critical consideration in this project. Real-time usability is essential for any practical stock prediction tool, as investors and analysts rely on timely insights to make informed decisions.
Optimization techniques employed in this project include:

1. **Hyperparameter Tuning**: Parameters such as the number of LSTM units, batch size, and learning rate are optimized to balance accuracy and efficiency.
2. **Early Stopping**: Monitors validation loss during training and halts the process when improvements stagnate, preventing overfitting and saving computational resources.

These measures ensure that the model remains both accurate and computationally efficient, making it suitable for real-world applications.

**Addressing the Gap in Traditional Models**

The project bridges the gap between the limitations of traditional econometric models and the potential of modern ML algorithms. While traditional models like ARIMA and GARCH offer simplicity and interpretability, they often fail to adapt to the complexities of financial markets.
In contrast, LSTM networks are capable of learning intricate patterns and dependencies, providing a scalable and robust framework for stock price prediction. By integrating advanced ML techniques with domain-specific knowledge, this project demonstrates the potential of AI-driven financial forecasting.

**Broader Implications for Financial Analytics**

The significance of this work extends beyond stock price prediction to the broader field of financial analytics. Improved prediction accuracy can lead to:

1. **Enhanced Investment Strategies**: Providing reliable insights into future price movements enables investors to optimize portfolio allocations and reduce risk.
2. **Reduced Financial Risks**: Accurate forecasts help mitigate the impact of market volatility, allowing stakeholders to make informed decisions.
3. **Optimized Trading**: Traders can leverage predictive models to identify profitable entry and exit points, improving overall returns.

Furthermore, the LSTM model developed in this project can serve as a foundation for future research, incorporating additional factors such as macroeconomic indicators, sentiment analysis, or hybrid modeling approaches that combine ML with traditional methods.

**Conclusion**

This project highlights the transformative potential of advanced ML techniques in addressing the challenges of stock price prediction. By leveraging the capabilities of LSTM networks and integrating technical indicators, it provides a robust, scalable, and efficient framework for forecasting stock prices within the Nifty 50 index.

The broader implications of this work extend to improving financial decision-making, reducing risks, and optimizing investment strategies. As financial markets continue to evolve, the application of cutting-edge tools like LSTM will play an increasingly critical role in enabling informed and strategic market participation.

**Data Collection and Preprocessing**

- **Data Source:**
  - Historical stock data for Nifty 50 companies is sourced from **Yahoo Finance**, a widely used and reliable platform for financial data.
  - The dataset spans a **10-year period** to provide sufficient historical context and cover various market conditions, including bull and bear phases.
  - The data includes key attributes such as **opening price, closing price, high, low, adjusted close, and trading volume**, which are crucial for understanding market behavior.
- **Data Cleaning:**
  - **Handling Missing Values:**
    - Missing data points, common in financial datasets, are identified and addressed to prevent bias in the model.
    - Techniques such as forward-filling, backward-filling, or interpolation are used to fill gaps in the data.
  - **Outlier Removal:**
    - Outliers, such as unusually high or low prices due to data errors, are identified and handled using statistical thresholds or domain knowledge.
  - **Consistency Checks:**
    - Ensuring data integrity by verifying the alignment of dates, correcting incorrect labels, and confirming uniformity across all records.
  - **Normalization of Tickers:**
    - Standardizing stock tickers to ensure compatibility across different sources of data.
- **Data Structuring:**
  - Data is organized into a structured format, such as a multi-index DataFrame, where the first level is the date, and the second level represents individual stock tickers.

- This structure facilitates efficient access, analysis, and feature engineering for the subsequent stages of the pipeline.

**Feature Engineering**

- **Purpose:**
  - Feature engineering focuses on deriving **technical indicators** that enhance the model's ability to identify patterns and predict stock prices effectively.
  - These indicators provide the model with insights into **market dynamics**, including momentum, volatility, and trend strength.
- **Key Technical Indicators:**
  - **Moving Averages (MA):**
    - Represents the average price of a stock over a specific time period (e.g., 5, 20, or 50 days).
    - Used to identify overall market trends by smoothing out short-term fluctuations.
    - Different types, such as simple moving averages (SMA) and exponential moving averages (EMA), are employed for varied insights.
  - **Relative Strength Index (RSI):**
    - A momentum oscillator that measures the speed and change of price movements.
    - RSI values range between 0 and 100, with readings above 70 indicating overbought conditions and below 30 indicating oversold conditions.
    - Helps capture market momentum and identify potential reversals.
  - **Bollinger Bands:**
    - Composed of a middle band (moving average) and two outer bands set at standard deviations above and below the middle band.
    - Helps identify price volatility, overbought/oversold conditions, and potential trend reversals.
  - **Garman-Klass Volatility:**
    - Measures volatility using high, low, open, and close prices within a period.
    - Offers a more comprehensive analysis of market fluctuations compared to traditional volatility measures.
    - Useful for assessing periods of significant price swings and risk levels.
  - **Additional Features:**
    - Indicators such as Moving Average Convergence Divergence (MACD), Average True Range (ATR), and Traded Value provide further dimensions of market insight.
- **Impact on Prediction:**
  - The inclusion of these indicators allows the model to:
    - Capture both **short-term price movements** and **long-term trends**.
    - Enhance prediction accuracy by integrating diverse perspectives on market behavior.
    - Improve robustness, reducing the likelihood of overfitting to specific patterns.

**Data Scaling and Structuring**

- **Data Scaling:**

- The dataset is scaled using **Min-Max normalization**, a technique that transforms data to a range between 0 and 1.
- Scaling ensures:
  - **Uniformity:** All features are brought to the same scale, preventing dominance by features with larger magnitudes.
  - **Improved Model Performance:** Neural networks like LSTM are sensitive to input ranges, making scaling essential for stable training.
- Both input features (e.g., technical indicators) and target variables (e.g., closing prices) are normalized to maintain consistency.

- **Time-Step Sequence Creation:**
  - To prepare data for the LSTM model, sequential input data is structured into **time-step windows**.
  - For example, a sequence of 12 time steps is created, where each step represents one month of historical data.
  - The input to the model includes:
    - **X_lstm:** A sequence of feature data (e.g., indicators) for the past 12 time steps.
    - **y_lstm:** The corresponding target value (e.g., closing price) for the next time step.
  - This structure enables the LSTM model to process historical dependencies effectively, leveraging past trends to make future predictions.
- **Advantages of Structured Data:**
  The structured data format ensures:
    - Efficient access to historical patterns.
    - Seamless integration with machine learning frameworks.
    - Enhanced predictive accuracy through well-defined input-output relationships.

**Key Components of the Model**

1. **Bidirectional LSTM Layer:**
   - The Bidirectional LSTM processes input sequences in two directions: forward (past to future) and backward (future to past).
   - This dual processing allows the model to understand dependencies in both past and future contexts, improving feature extraction.
   - It captures the dynamic patterns in stock price movements, including long-term trends and short-term fluctuations, which are crucial for accurate predictions.
2. **Regularization Techniques:**
   - **Dropout Layers:**
     - Dropout layers are used to randomly deactivate a fraction of neurons during training (e.g., 20% in this project).
     - This prevents over-reliance on specific neurons, reducing the risk of overfitting.
     - Dropout enhances the generalization ability of the model, ensuring it performs well on unseen data.
   - **Batch Normalization:**
     - Normalizes layer inputs to improve stability and accelerate training.
     - Helps mitigate issues like vanishing gradients, which are common in deep learning models.
3. **Dense Layers (Fully Connected Layers):**

- Dense layers at the end of the network aggregate and interpret the features extracted by LSTM layers.
- These layers learn complex, non-linear relationships between the input data (historical stock prices and indicators) and the target (future stock price).
- The output dense layer consists of a single unit for regression, predicting the stock's closing price.

**Training Techniques**

1. **Early Stopping:**
   - Early stopping monitors the validation loss during training.
   - If the validation loss does not improve for a specified number of epochs (e.g., 15), training halts to prevent overfitting.
   - This ensures that the model achieves an optimal balance between training accuracy and generalization to new data.
   - The model restores the best-performing weights automatically to ensure the highest validation accuracy.
2. **Validation Split:**
   - During training, a portion of the training dataset (e.g., 20%) is reserved for validation purposes.
   - The validation set evaluates the model's performance on unseen data, providing a realistic measure of its predictive capabilities.
   - This split helps monitor the model's generalization ability and guides decisions on hyperparameter tuning.

**Model Optimization**

In the process of training a machine learning model, particularly a Long Short-Term Memory (LSTM) network, the selection of key hyperparameters and optimization strategies plays a critical role in determining model performance. Hyperparameter tuning, the choice of loss function, and the optimizer all contribute significantly to the model's ability to generalize well on unseen data, avoid overfitting or underfitting, and converge efficiently during training. In this section, we delve into the processes and strategies used for hyperparameter tuning, loss function selection, and optimizer configuration in this stock price prediction project.

1. **Hyperparameter Tuning:**
   Hyperparameter tuning refers to the process of selecting the most appropriate values for the parameters that govern the learning process of the model. Unlike model parameters (such as weights and biases), hyperparameters are set before training and can significantly influence the model's ability to learn from the data. The key hyperparameters in this project that are tuned for optimal performance include:
   **a) Learning Rate**
   The learning rate is one of the most critical hyperparameters when training deep learning models. It determines the size of the steps the model takes during the optimization process (i.e., how much the weights are adjusted after each iteration). A learning rate that is too high can cause the model to overshoot the optimal solution, leading to poor performance or unstable training. On the other hand, a learning rate that is too low can lead to slow convergence or getting stuck in a suboptimal solution. For this project, a learning rate of 0.001 was chosen for the Adam optimizer. This learning rate strikes a balance between efficient convergence and avoiding large steps that could cause instability in the model. Additionally, learning rate schedules or

adaptive learning rates can be used to fine-tune the learning process, adjusting the rate over time to improve convergence further.

**b) Number of LSTM Units**

The number of LSTM units (or neurons) in the network determines the capacity of the model to capture patterns from the data. Too few units may lead to underfitting, where the model fails to capture the underlying trends in the data. Conversely, too many units may result in overfitting, where the model learns to memorize the training data but fails to generalize to unseen data.

In this project, the LSTM model uses 100 units in each LSTM layer, which was determined after experimenting with different configurations. This number strikes a balance between the model's ability to learn from the data and its capacity to generalize effectively.

**c) Batch Size**

The batch size refers to the number of training samples used in one forward/backward pass of the model during training. A small batch size leads to noisy updates to the model parameters, which can help the model escape local minima, but may also cause instability in convergence. On the other hand, a large batch size can lead to more stable updates but may also cause the model to get stuck in local minima or converge too slowly.

A batch size of 32 was chosen for this model based on experimentation and the nature of the dataset. It provides a good balance between training efficiency and model performance, allowing the optimizer to update weights frequently without introducing too much noise into the gradient estimates.

**d) Dropout Rate**

Dropout is a regularization technique used to prevent overfitting by randomly "dropping" (i.e., setting to zero) a fraction of the neurons during training. This forces the network to learn redundant representations, thus improving its generalization capability. The dropout rate is a key hyperparameter in this process.

For this project, a dropout rate of 0.2 was chosen. This means that 20% of the neurons are randomly dropped during each training iteration. This rate strikes a good balance between regularization and retaining sufficient model capacity to learn from the data.

**e) Hyperparameter Search Process**

The tuning of these hyperparameters was performed using a combination of grid search and random search techniques. Grid search exhaustively searches through a manually specified subset of hyperparameters, whereas random search samples random combinations of hyperparameters. Both methods were employed to ensure that the model's hyperparameters were optimized for the best performance.

2. **Loss Function:**
    - The model uses Mean Squared Error (MSE) as the loss function for regression tasks.
    - MSE penalizes large prediction errors more heavily, ensuring the model focuses on minimizing significant deviations.
3. **Optimizer:**
    - The Adam optimizer is employed with a learning rate of 0.001.
    - Adam combines the benefits of Momentum and RMSProp optimizers, leading to faster and more stable convergence.

**Summary of Advantages**

- The bidirectional LSTM architecture ensures the model captures both past and future contexts of stock price data.
- Regularization and validation techniques reduce overfitting, ensuring reliable predictions on unseen data.
- Dense layers provide the flexibility to model complex relationships, making the architecture robust for financial time-series forecasting.
- Overall, the design balances accuracy, computational efficiency, and real-time applicability, making it suitable for dynamic stock market environments.

Model evaluation is a critical step to ensure that the predictive capabilities of the LSTM model meet the desired performance benchmarks. Various statistical metrics are used to measure accuracy, reliability, and robustness. These metrics help quantify the model's effectiveness in capturing stock price movements and provide insights into areas for potential improvement.

**Key Evaluation Metrics**

1. **Mean Absolute Error (MAE):**
    - **Definition:** MAE measures the average absolute differences between the predicted and actual values, providing an intuitive sense of prediction error magnitude.
    - **Significance:**
        - MAE gives equal weight to all errors, making it a straightforward indicator of the model's accuracy.
        - Lower MAE values indicate that the model's predictions are closely aligned with the actual values.
    - **Application in this project:** The MAE value quantifies how well the LSTM model performs in predicting stock prices, reflecting its practical usability.

2. **Root Mean Squared Error (RMSE):**
    - **Definition:** RMSE measures the square root of the average squared differences between predicted and actual values, emphasizing larger errors due to the squaring term.
    - **Significance:**
        - RMSE is sensitive to large deviations between predictions and actual values, making it ideal for financial models where significant errors can have a major impact.
        - It provides a more comprehensive evaluation of the model by penalizing large errors more heavily than smaller ones.
    - **Performance Indicator:** A low RMSE value in this project confirms that the model performs well even when handling extreme price fluctuations.
3. **R-squared ($R^2$):**
    - **Definition:** $R^2$ represents the proportion of the variance in the dependent variable (actual stock prices) that is explained by the independent variables (model inputs).
    - **Significance:**
        - A value close to 1 indicates that the model explains most of the variability in the stock price data.
        - $R^2$ highlights the strength of the relationship between input features and predicted stock prices.

- **Project Outcome:** An $R^2$ value of 0.9598 in this project demonstrates the model's ability to capture the underlying patterns in the data accurately.

4. **Mean Absolute Percentage Error (MAPE):**
   - **Definition:** MAPE measures the average percentage error between predicted and actual values, providing a relative measure of accuracy.
   - **Significance:**
     - MAPE allows for easy interpretation by expressing errors as a percentage.
     - It is particularly useful in financial forecasting, where relative accuracy is often more important than absolute error.
   - **Performance Indicator:** The model achieves a MAPE of 1.49%, indicating exceptional accuracy in predicting stock prices.
5. **Accuracy:**
   - **Definition:** Accuracy measures the percentage of correct predictions made by the model relative to the total predictions.
   - **Significance:**
     - High accuracy indicates the model's reliability in making predictions that align with actual outcomes.
     - It is a direct reflection of the model's performance in meeting the project's objectives.
   - **Outcome in this project:** The model achieves an overall accuracy of **98.51%,** making it highly suitable for practical stock market prediction tasks.

**Visual Analysis**

- **Predicted vs. Actual Values:**
  - A graph comparing predicted stock prices to actual values is used to visually validate the model's performance.
  - Consistent overlap between the predicted and actual curves indicates that the model successfully captures the trends and patterns in stock prices.
- **Error Distribution:**
  - The distribution of residual errors (differences between predicted and actual values) is analyzed.
  - A near-normal error distribution centered around zero suggests that the model's predictions are unbiased and reliable.

**Implications of Evaluation Metrics**

- **High Accuracy and Low Error Values:**
  - The combination of a low MAE, RMSE, and MAPE with a high $R^2$ and accuracy highlights the model's robustness.
  - These metrics demonstrate that the LSTM model is well-suited for financial time-series forecasting, especially for predicting stock prices in the volatile Nifty 50 index.
- **Performance Across Data Splits:**
  - The metrics indicate that the model generalizes well across training, validation, and testing datasets, ensuring reliability in real-world applications.

- The model evaluation confirms that the LSTM-based approach effectively predicts Nifty 50 stock prices with high accuracy and low error rates.

- The chosen metrics provide a comprehensive understanding of the model's performance, highlighting its suitability for both short-term and long-term stock price forecasting.
- These results underscore the practicality and relevance of machine learning in financial analytics, paving the way for further advancements in stock market prediction.

**Performance Comparison:**

- The LSTM model demonstrates significant improvements in prediction accuracy compared to traditional regression models like Linear Regression, Decision Trees, and Support Vector Regression (SVR).
- Key metrics such as **Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared ($R^2$)** highlight the superior performance of the LSTM model in capturing stock price patterns.

**Visualization of Predictions:**

- Graphs and plots are used to visualize the predictions of the LSTM model against the actual stock prices.
  - **Actual vs. Predicted Prices:** The overlap between predicted and actual prices illustrates the precision of the model.
  - **Error Analysis:** Visualization of residual errors (differences between actual and predicted values) helps identify any consistent biases or anomalies.
- These visual tools not only validate the model's accuracy but also provide stakeholders with intuitive insights into the model's capabilities.

**Accuracy and Efficiency:**

- The LSTM model achieves a high level of accuracy, with metrics such as:
  - **Accuracy Rate:** Approximately 98.51%.
  - **Mean Absolute Percentage Error (MAPE):** A low value of 1.49%, demonstrating minimal deviation between predictions and actual values.
- Computational efficiency is maintained, ensuring the model processes data and generates predictions within a reasonable timeframe. This balance makes the model suitable for **real-time applications** in dynamic market environments.

**Robustness of the Model:**

- The use of **early stopping** during training prevents overfitting, ensuring that the model generalizes well across various market conditions.
- Regularization techniques, such as dropout layers, further enhance the model's robustness and stability, even when dealing with volatile financial data.

**Advancement in Financial Forecasting:**

- The project highlights the potential of LSTM networks in **time-series forecasting**, specifically for financial applications such as stock price prediction.
- By effectively capturing sequential dependencies, the model provides a valuable tool for investors, traders, and financial analysts.

**Practical Use Cases:**

- The model offers a practical solution for real-time prediction of stock prices, assisting in:
  - **Investment Decisions:** Empowering investors with data-driven insights to make informed decisions.
  - **Risk Management:** Helping analysts identify potential market risks through early detection of trends.
  - **Trading Strategies:** Supporting algorithmic trading systems with precise and reliable stock price forecasts.

**Educational Contribution:**

- The project serves as a reference for researchers and students interested in combining **machine learning** and **financial analytics**, encouraging further exploration in this interdisciplinary field.

**Incorporation of External Factors:**

- Future iterations of the model could integrate external data sources, such as:
  - **Market Sentiment:** Using sentiment analysis on financial news or social media to gauge public perception.
  - **Macroeconomic Indicators:** Including factors like GDP growth, interest rates, and inflation to provide a broader context for price movements.

**Development of Hybrid Models:**

- Combining LSTM with other machine learning techniques, such as Convolutional Neural Networks (CNN) or Gradient Boosting Machines, to create hybrid models that leverage the strengths of multiple algorithms.
- These hybrid models could improve accuracy and adaptability by capturing both sequential patterns and non-linear relationships.

**Exploration of Transfer Learning:**

- Applying transfer learning techniques to adapt the model to other stock indices or financial markets, such as the S&P 500 or NASDAQ, without requiring extensive retraining.

**Inclusion of Real-Time Features:**

- Enhancing the model's real-time capabilities by incorporating:
  - **High-Frequency Trading Data:** Leveraging minute-by-minute or second-by-second data for short-term predictions.
  - **Dynamic Feature Updates:** Allowing the model to adapt to evolving market conditions.

**Improved Model Interpretability:**

- Developing explainable AI techniques to provide users with transparent insights into how predictions are generated, fostering trust and usability among stakeholders.

**Scalability and Deployment:**

- Expanding the model's application to large-scale financial systems with multiple indices and integrating it into user-friendly platforms (e.g., web or mobile applications).

- Deploying the model on cloud-based systems to ensure accessibility and scalability for a wide range of users.

**Broader Impact**

The development of accurate and efficient stock price prediction models has profound implications for a wide range of stakeholders in the financial ecosystem, including individual investors, financial institutions, and policymakers. Traditional stock market prediction methods, which often rely on econometric models and human intuition, are limited in their ability to capture the complex, non-linear patterns that drive stock price movements. By addressing these limitations, the project's use of advanced machine learning techniques, particularly Long Short-Term Memory (LSTM) networks, has the potential to revolutionize financial forecasting and shape the future of investment strategies. Below, we explore the broader impact of this project in greater detail.

**1. Addressing the Gaps in Traditional Stock Market Prediction Techniques**

Traditional stock market prediction models, such as linear regression or time-series models like ARIMA (AutoRegressive Integrated Moving Average), have long been used to forecast stock prices. While these models have their place, they often fail to capture the intricate relationships and non-linear dependencies inherent in financial data. Stock markets are influenced by a multitude of factors, including economic indicators, investor sentiment, political events, and more, all of which are difficult to model using linear approaches.

This project addresses these gaps by leveraging **Long Short-Term Memory (LSTM)** networks, a type of Recurrent Neural Network (RNN) designed to work with sequential data. LSTMs can learn and retain long-term dependencies, which is crucial for predicting time-series data like stock prices. Traditional models may miss these long-term patterns or fail to adapt to rapidly changing market conditions. In contrast, LSTMs can analyze historical price data and key technical indicators, such as Moving Averages, Relative Strength Index (RSI), and Bollinger Bands, to generate more accurate predictions that better reflect the underlying market dynamics.

By using LSTM networks, this project creates a more sophisticated forecasting tool that takes into account the complexities of stock market data. The model's ability to process large amounts of sequential data and capture both short-term and long-term trends enhances its prediction accuracy and robustness, offering a more reliable alternative to traditional forecasting models.

**2. A Foundation for More Accurate and Efficient Financial Forecasting Models**

The impact of this project extends far beyond individual stock price prediction. The model's architecture, which incorporates various technical indicators and uses advanced machine learning algorithms, provides a blueprint for future developments in financial forecasting. As financial markets become increasingly complex and data-driven, the need for models that can adapt to changing conditions, incorporate diverse datasets, and provide real-time insights is growing.

In the context of real-time trading, automated systems powered by machine learning models like LSTM can provide more accurate short-term price predictions. These systems can make quick decisions based on live data, helping investors and traders react faster to market shifts. This efficiency can lead to better investment strategies, reduced risks, and more profitable outcomes.

Furthermore, the methodology employed in this project can be applied to other areas of financial forecasting, such as:

- **Predicting market trends**: Beyond individual stock prices, LSTM models can be used to predict broader market trends, such as index movements or sector performance.
- **Portfolio optimization**: Advanced machine learning models can help financial institutions and individual investors optimize their portfolios by predicting the returns of various assets and minimizing risk.
- **Volatility forecasting**: By incorporating additional features like Garman-Klass Volatility, the model can also help predict market volatility, which is crucial for risk management.

As the model continues to evolve and integrate additional data sources (e.g., sentiment analysis, macroeconomic indicators, geopolitical events), it will enhance its predictive power and become a key tool for financial forecasting.

**3. Benefits for Individual Investors**

For individual investors, access to accurate and timely stock price predictions can drastically improve their decision-making processes. Investing in the stock market involves a significant amount of risk, and accurately predicting stock price movements can help mitigate this risk. By providing more reliable forecasts, this project empowers individual investors to make more informed decisions regarding when to buy or sell stocks, ultimately improving their returns.

Moreover, the model's ability to analyze both short-term and long-term price movements means that it can cater to different investment strategies. For example:

- **Day traders** can benefit from the model's short-term predictions, using them to identify intra-day price fluctuations and capitalize on market inefficiencies.
- **Long-term investors** can use the model's ability to forecast longer-term trends and align their investments with the broader direction of the market.

By enhancing the accuracy of stock price predictions, the model also contributes to greater market transparency, helping individual investors make decisions based on data rather than speculation or gut feeling.

**4. Benefits for Financial Institutions**

Financial institutions, such as hedge funds, asset management firms, and investment banks, can significantly benefit from the enhanced predictive capabilities offered by this project. The ability to make more accurate predictions allows these institutions to refine their investment strategies, optimize asset allocation, and minimize risk.

For instance, **hedge funds** that rely on quantitative strategies can integrate machine learning models like LSTMs into their trading algorithms, enabling them to react to market changes in real-time and execute trades more efficiently. **Asset managers** can use the model to identify stocks with strong future potential, helping them build more robust portfolios that align with their clients' investment goals.

Additionally, the model's ability to process vast amounts of data from various sources (e.g., technical indicators, historical stock prices, and volatility measures) provides a competitive edge for financial institutions, helping them stay ahead of market trends and outperform competitors.

**5. Benefits for Economic Policymakers**

Economic policymakers, such as government agencies, central banks, and regulators, can also benefit from the insights generated by more accurate stock price prediction models. Stock market performance is often a barometer for broader economic health, and better predictions can provide early signals of potential economic shifts.

For example:

- **Regulatory bodies** can use improved forecasting models to monitor financial markets more effectively and take preventive measures in times of economic instability.
- **Central banks** can use predictions of stock market trends to assess the impact of monetary policy decisions (e.g., interest rate changes) on financial markets.
- **Government agencies** can leverage predictive insights to make more informed decisions about fiscal policies that affect economic growth and stability.

The ability to forecast market trends with higher accuracy helps policymakers anticipate and mitigate economic crises, improve decision-making, and foster a more stable financial system.

**6. Implications for Data-Driven Decision-Making**

At a broader level, this project exemplifies the potential of data-driven decision-making in the financial sector. By using machine learning algorithms like LSTMs, which are capable of processing vast amounts of data and learning complex patterns, the project sets a precedent for future advancements in financial analytics. Data-driven approaches are poised to transform various aspects of finance, from market prediction to investment strategy formulation, risk management, and beyond.

As the technology matures, the integration of additional data sources (e.g., news sentiment, social media analysis, macroeconomic indicators) will further enhance prediction accuracy, enabling even more sophisticated financial decision-making. The project's success demonstrates how cutting-edge machine learning techniques can be applied to real-world financial challenges, contributing to the evolution of the financial services industry.

# 7. RESULTS AND DISCUSSION

**1. Results Summary:**

The LSTM model developed for predicting Nifty 50 stock prices has shown exceptional performance across multiple evaluation metrics, indicating its high level of predictive accuracy and ability to capture the complexities of stock market trends. Here's a deeper analysis of the key performance indicators:

**Mean Absolute Error (MAE): 270.2056**

The MAE is a direct measure of how close the model's predictions are to the actual stock prices. A lower MAE indicates that the predictions are relatively close to the actual values, which is critical in financial forecasting where small errors can lead to significant financial implications.

**Mean Squared Error (MSE): 149170.5569**

The MSE penalizes large prediction errors more than the MAE, providing an indication of how severe large deviations between predicted and actual values are. A lower MSE in the model suggests that extreme prediction errors are relatively infrequent, which is ideal in financial applications where large discrepancies in predictions can result in large financial losses.

**Root Mean Squared Error (RMSE): 386.2260**

RMSE represents the square root of MSE and gives an estimate of the average magnitude of prediction errors in the same units as the original data (in this case, stock prices). An RMSE of 386.2260 is relatively small, reflecting that the model is making predictions within a reasonable range of actual stock prices.

**R-squared ($R^2$): 0.9598**

The $R^2$ value of 0.9598 means that approximately 96% of the variance in stock prices is explained by the model. This high $R^2$ indicates that the model has successfully captured the underlying patterns in the stock market data, which is crucial for making accurate predictions. It also suggests that the model can generalize well to unseen data.

**Mean Absolute Percentage Error (MAPE): 1.49%**

MAPE measures the accuracy of the model in terms of percentage. A MAPE of 1.49% means that, on average, the model's predictions are off by only about 1.5%. This is an excellent result, particularly for stock price prediction where minute changes can have large financial impacts.

**Accuracy: 98.51%**

The accuracy of 98.51% is another strong indicator of the model's performance, confirming that the model is highly reliable in predicting stock prices. This accuracy is particularly valuable for investors looking for a high level of confidence in the model’s ability to guide trading and investment decisions.

These results indicate that the LSTM model not only predicts stock prices with a high degree of precision but also captures the intricate, non-linear relationships within stock market data, which traditional methods often struggle with.

## 2. Cost Analysis and Practical Implications:

### A) Computational Costs:

**Model Training:**
The LSTM model, especially with Bidirectional layers, is computationally intensive. The training process requires handling large datasets (up to 10 years of historical stock data), which increases both time and hardware demands. While the LSTM's deep learning structure allows for powerful predictions, it also necessitates the use of high-performance computing resources.
Optimizations like batch normalization and dropout layers helped mitigate issues related to overfitting, but these solutions still require substantial computational power and time. The choice of optimization strategies directly impacts the efficiency and speed of model convergence.

**Hardware Requirements:**
Training the model on a standard CPU might be impractical due to the resource demands. Using Graphics Processing Units (GPUs) or specialized cloud computing platforms (e.g., AWS EC2 instances with GPU support, Google Cloud's AI Platform) can significantly speed up training times and reduce the time-to-deployment. However, these resources come at a significant cost, which needs to be factored into the project’s budget.
Cloud-based solutions allow for scalability and flexibility in terms of resource allocation, but the cost can escalate quickly depending on the training duration and data volume.

### B) Development Costs:

**Development Time:**
Building, fine-tuning, and validating an LSTM model is time-consuming. It involves extensive data preprocessing (such as feature engineering), model architecture design, hyperparameter tuning, and performance evaluation. This process requires expertise in machine learning and deep learning techniques, which translates to longer development cycles and increased personnel costs.
Iterative testing and validation also add to the time required for development, as the model must be tested across various market conditions to ensure robustness.

**Software and Libraries:**
The development leveraged open-source libraries like TensorFlow, scikit-learn, and pandas, which helped minimize licensing fees. However, the time and expertise required to implement and fine-tune these libraries should not be underestimated. Although free, the effective utilization of these tools requires specialized knowledge and experience.

**Data Acquisition:**
Historical stock data was obtained from sources like Yahoo Finance (via yfinance), which is freely accessible. However, this dataset might not be sufficient for models that require high-frequency data or additional economic indicators. Premium data sources, such as Bloomberg or Thomson Reuters, could provide more granular data, but at a cost. Acquiring these data sets is an additional financial burden that could affect the overall cost structure of the project.

**C) Benefits vs. Costs:**

**Benefits:**
The model's ability to achieve a high level of accuracy (98.51%) and low MAPE (1.49%) means that it has the potential to provide valuable insights and predictive capabilities for stock market traders. Reliable predictions can help investors make informed decisions, potentially yielding high returns by optimizing their portfolio management and trading strategies.
Additionally, the model's scalability means that it can be adapted to predict stock prices for other sectors in the Nifty 50, enhancing its utility and further justifying the initial investment in the model development.

**Scalability:**
One of the major strengths of the LSTM model is its ability to scale across multiple sectors. As a result, the model can be adapted to predict stock prices for different stocks in the Nifty 50 index, providing broader applicability. This scalability also allows the model to be leveraged for diverse investment strategies, expanding its utility in the financial sector.

**Time Efficiency:**
Real-time prediction is a key advantage for high-frequency trading (HFT) and day-to-day portfolio management. With quick predictions, investors can make better decisions, optimize trading strategies, and reduce the risk of losses. As a result, investing in computational resources (such as GPUs or cloud computing) can lead to long-term profitability in these fast-paced environments.

**D) Limitations and Considerations:**
**Computational Intensity:**
While the model delivers high performance, its computational intensity makes it costly for smaller organizations or individual traders without access to advanced infrastructure. The training time required for LSTM models can also limit the frequency with which the model can be retrained to keep up with changing market conditions.

**Data Limitations:**
The model primarily relies on historical price data and technical indicators, which might not fully account for sudden market shifts due to macroeconomic factors or unexpected geopolitical events. For example, in times of financial crises or global disruptions, the model might not perform as well since it is built on historical data trends.

**Validation Needs:**
The model has shown excellent performance during training, but real-time back-testing over various market conditions is necessary to further validate its reliability. Market dynamics are continuously evolving, and the model must be periodically updated to adapt to new trends. Continuous validation and testing are crucial for maintaining the model’s accuracy and reliability.

**Discussion:**
The LSTM model developed for predicting Nifty 50 stock prices presents a substantial improvement in the accuracy of stock market predictions. Its strong performance is a result of carefully selecting and engineering features, including technical indicators, and employing a robust deep learning architecture.

Despite the model's success, challenges remain, particularly in managing computational costs and ensuring the model’s robustness in the face of unprecedented market events. Integrating more fundamental economic indicators (e.g., interest rates, inflation, GDP growth) could enhance the model’s predictive power, particularly during macroeconomic shifts. Additionally, refining the model to improve computational efficiency and reduce resource consumption will be essential for making the model more accessible to a wider audience.

As the financial markets continue to evolve, the model’s flexibility and scalability will remain key strengths, making it a promising tool for both institutional investors and individual traders seeking to improve their trading strategies with machine learning-powered predictions.

# 8. CONCLUSION

The high accuracy and reliability demonstrated in this project position it as a valuable tool for institutional investors who seek to leverage predictive models for enhancing trading and investment strategies. The model's robust performance highlights its potential to aid in informed decision-making and risk management. However, to fully capitalize on its capabilities while minimizing operational expenses, future work should explore cost-saving measures.

These could include using optimized algorithms, such as pruning techniques or quantized models, and reducing feature sets without compromising prediction quality. Additionally, employing techniques like transfer learning may help repurpose the model for other financial tasks, maximizing utility while controlling costs.

Future improvements can focus on creating a user-friendly interface (UI) for the model, making it accessible to non-technical users such as financial analysts and investors. A well-designed UI would facilitate seamless data input, model training, and result visualization, enhancing user interaction and broadening the model's practical application. This would enable users to quickly gain insights, experiment with different configurations, and analyze predictions without deep technical knowledge.

Adding various analysis and visualization tools would further enhance the model's value. Interactive dashboards that display comparative stock performance, monthly returns, technical indicator trends, and actual vs. predicted closing prices could offer comprehensive insights at a glance. Such visualizations can help investors better understand market behavior and identify patterns that align with their strategies.

Moreover, integrating stock news and sentiment analysis using NLP (Natural Language Processing) could elevate the model's predictive power. By incorporating real-time news data and analyzing the sentiment around stocks, the model would gain an additional layer of context, allowing it to respond to market-moving events more effectively. This feature would make the predictions more adaptive and capable of capturing external factors that purely numerical data might miss, ultimately providing a holistic view for better investment strategies.

In conclusion, the LSTM-based predictive model developed in this project demonstrates a strong capability for forecasting stock prices within the Nifty 50 index, backed by high accuracy and comprehensive technical analysis. The integration of a wide range of technical indicators and advanced model architecture contributes to its effectiveness. However, to make the model more accessible and practical, future enhancements should focus on user-oriented features such as an intuitive UI, comprehensive visualization tools, and the addition of sentiment analysis to account for news and macroeconomic influences.

These improvements would not only extend the usability of the model but also transform it into a powerful, real-time decision-support tool for both institutional and retail investors. By incorporating these advanced features and optimizing for efficiency, the model can evolve to meet the dynamic demands of the financial markets, offering deeper insights, adaptability, and strategic advantages.

## 9. REFERENCES


**Weblinks**:

1. https://finance.yahoo.com/
2. https://hdfcsky.com/sky-learn/share-trading/technical-indicators-and-how-to-interpret-them
3. https://www.britannica.com/money/technical-indicator-types
4. https://www.incrediblecharts.com/indicators/technical-indicators.php#g
5. https://www.algomatictrading.com/post/garman-klass-volatility#:~:text=Garman%2DKlass%20Volatility%20is%20an,points%20within%20each%20trading%20period

**Journals**:

1. Bansal, M., & Choudhary, A. (2022). "Stock Price Prediction Using Machine Learning Algorithms: A Comparative Study on Indian Companies." International Journal of Advanced Computer Science and Applications, 13(4), 101-110.

2. Wang, X., Li, Z., Zhang, T., & Zhang, J. (2022). "A Spatiotemporal Multi-Graph CNN Model for Stock Market Prediction Using Technical Indicators." IEEE Access, 10, 120348-120357.

3. Sharma, R., & Juneja, D. (2023). "Hybrid Ensemble Models for Stock Market Index Prediction." Journal of Finance and Data Science, 9(1), 45-58.

4. Rao, P., Srinivas, S., & Mohan, P. (2024). "Comparative Study of Neural Network and Statistical Models for Stock Market Predictions." Journal of Economic Computation and Cybernetics Studies and Research, 58(3), 203-219.

5. Fathali, Z., Sadeghi, M., & Zadeh, A. (2022). "Performance Evaluation of LSTM, CNN, and RNN for Stock Market Prediction: A Nifty 50 Case Study." Expert Systems with Applications, 185, 115632.

# APPENDIX A – SAMPLE CODE

**Feature Engineering (Technical Indicators):**

```
#FEATURE ENGINEERING(techincal indicators) using pandas_ta lib

# 1.Garman-Klass Volatility
# Formula: 0.5 * (log(High) - log(Low))^2 - (2*log(2)-1) * (log(Close) -
log(Open))^2
df['garman_klass_vol'] = (
    ((np.log(df['high']) - np.log(df['low'])) ** 2) / 2 -
    (2 * np.log(2) - 1) * ((np.log(df['adj close']) - np.log(df['open'])) **
2)
)

#2. RSI (Relative Strength Index) calculation for a window of 20 days
df['rsi'] = df.groupby(level=1)['adj close'].transform(
    lambda x: pandas_ta.rsi(close=x, length=20)
)

#3. Bollinger Bands for a window of 20 days
# bb_low: Lower Bollinger Band
df['bb_low'] = df.groupby(level=1)['adj close'].transform(
    lambda x: pandas_ta.bbands(close=np.log1p(x), length=20).iloc[:, 0]
)
# bb_mid: Middle Bollinger Band (SMA)
df['bb_mid'] = df.groupby(level=1)['adj close'].transform(
    lambda x: pandas_ta.bbands(close=np.log1p(x), length=20).iloc[:, 1]
)
# bb_high: Upper Bollinger Band
df['bb_high'] = df.groupby(level=1)['adj close'].transform(
    lambda x: pandas_ta.bbands(close=np.log1p(x), length=20).iloc[:, 2]
)

#4. ATR (Average True Range):
def compute_atr(stock_data):
    atr = pandas_ta.atr(
        high=stock_data['high'],
        low=stock_data['low'],
        close=stock_data['close'],
        length=14
    )
    return atr.sub(atr.mean()).div(atr.std())#Normalizing

df['atr'] = df.groupby(level=1, group_keys=False).apply(compute_atr)

#5. MACD (Moving Average Convergence Divergence):
```

```
def compute_macd(close):
    macd = pandas_ta.macd(close=close, length=20).iloc[:, 0]
    return macd.sub(macd.mean()).div(macd.std())#Normalizing

df['macd'] = df.groupby(level=1, group_keys=False)['adj
close'].apply(compute_macd)

#5. Traded Value in crores inr: (Adjusted Close Price * Volume) / 1 crore
df['traded_value'] = (df['adj close'] * df['volume']) / 1e7


df
```

**LSTM MODEL ARCHITECTURE:**

```
# LSTM model architecture
model = Sequential([
    Bidirectional(LSTM(100, return_sequences=True),
input_shape=(X_train.shape[1], X_train.shape[2])),
    BatchNormalization(),
    Dropout(0.2),
    LSTM(100, return_sequences=False),
    BatchNormalization(),
    Dropout(0.2),
    Dense(50, activation='relu'),
    Dense(1)  # Output layer for regression
])

#optimizer and loss funct
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
loss='mean_squared_error')


model.summary()
```