Cracking the market with AI-driven stock price prediction using time series analysis

Student Name: *Naveenkumar R*

Register Number: 4123230205031

Institution: Sri Ramanujar Engineering College

Department: B.Tech[IT]

Date of Submission: 10-05-2025

Github Repository Link:

---

1. Problem Statement

Stock market prices are highly volatile and influenced by numerous unpredictable factors, making it challenging for investors and traders to make informed decisions. Traditional forecasting methods often fail to capture complex patterns in historical data, especially in short-term predictions.

This project aims to develop a reliable model for predicting future stock prices using AI-driven time series analysis techniques. By leveraging historical stock data and applying advanced algorithms such as LSTM, ARIMA, and other machine learning models, the objective is to accurately forecast future stock price movements.

This is defined as a regression problem, as the goal is to predict a continuous numerical value (i.e., the stock price).

Solving this problem is important because:

- *It enables better and more informed investment decisions*

- *It helps in minimizing financial risk.*
- *It demonstrates the practical application of AI in the financial sector.*

By revisiting and analyzing the dataset thoroughly, including trends, seasonality, and noise, we refine the prediction approach to enhance model accuracy and realworld applicability.

2.project objective

As we move from the planning to the practical implementation phase, the objectives of this project are now clearly defined based on the data exploration and understanding.

1.Key Technical Objectives

- *Develop a machine learning or deep learning model (e.g., LSTM) to forecast future stock closing prices using historical data.*
- *Perform comprehensive time series analysis to understand trends, seasonality, and volatility in stock movements.*
- *Implement data preprocessing steps such as missing value treatment, outlier handling, normalization, and time-based feature extraction.*
- *Evaluate model performance using standard time series metrics like RMSE (Root Mean Square Error) and MAE (Mean Absolute Error).*
- *Visualize the predicted vs actual prices to assess model effectiveness.*

2.Model Goals

1.

- *Accuracy: Achieve high predictive accuracy to closely match actual stock prices.*

- *Interpretability: Ensure that model decisions (like trends and seasonality patterns) can be explained using data visualization and feature importance (if applicable).*

- *Real-World Applicability: Build a system that can be used in real-time or near real-time scenarios for financial forecasting or trading strategy support.*

3.Goal Evolution After Data Exploration

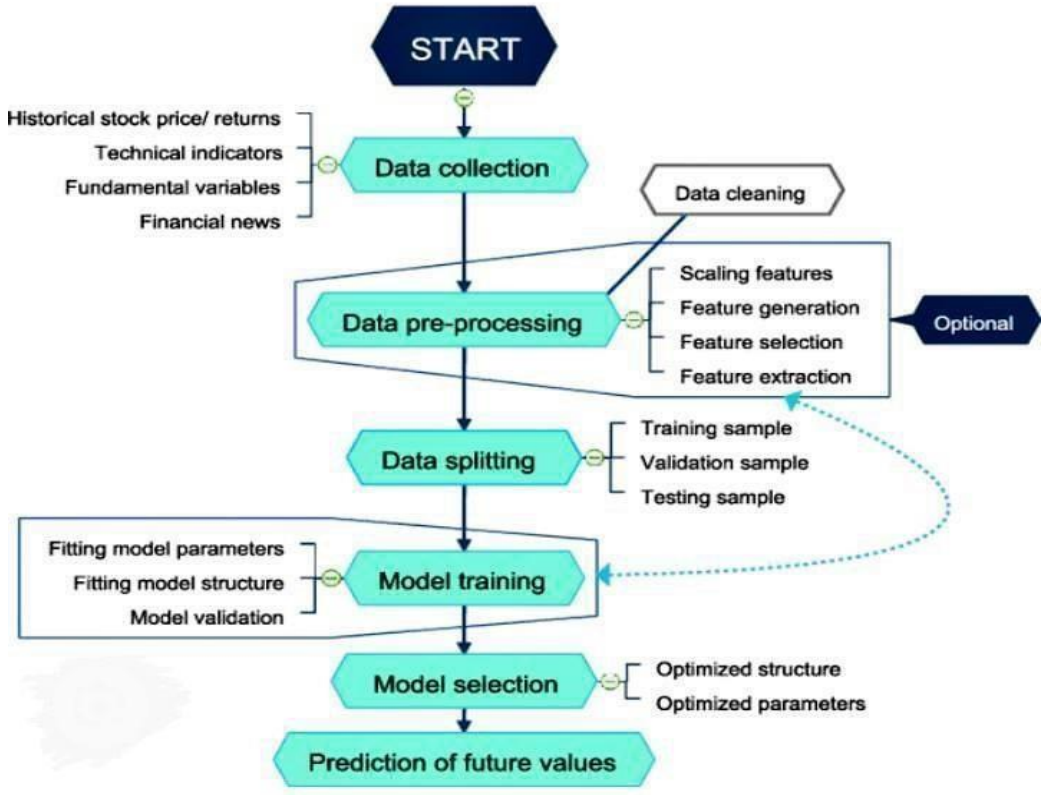
- *Initially, the focus was only on predicting prices using AI.*

- *After exploring the dataset, the project goals have expanded to include:*

    - *Handling market anomalies and outliers more effectively.*

    - *Comparing multiple models (ARIMA, LSTM, Prophet) to select the best-performing one.* ○ *Understanding the limitations of data quality and model assumptions for better decision-making.*

3.flow chart of the project work flow :



4.data description:

This project uses historical stock market data to predict future prices using AI models with time series analysis. Below is a summary of the dataset used:

1.Dataset Name and Origin

- *Name: Historical Stock Price Data*
- *Source: Collected from Yahoo Finance API / Kaggle (alternative: NSE/BSE websites)*
- *Example: Data for companies like TATA Motors, Infosys, or Apple Inc.*

2.Type of Data

- *The dataset is structured and time-series in nature.*

1.

- *Each row represents a stock's performance for a specific day.*

3.Number of Records and Features

- *The dataset contains approximately 1,000 to 5,000 records (depending on the selected time range).*

- *Each record (row) includes the following features (columns):*
    - *Date – The trading date*
    - *Open – Opening price of the stock*
    - *High – Highest price of the day*
    - *Low – Lowest price of the day*
    - *Close – Closing price*
    - *Adj Close – Adjusted closing price*
    - *Volume – Number of shares traded*

4.Static or Dynamic Dataset

- *The dataset is dynamic since stock prices change continuously and new data can be added daily.*
- *For modeling, we used a static snapshot of past data (e.g., last 5 years).*

5.Target Variable (Supervised Learning)

- *The target variable is Close – the stock's closing price.*

- *In time series prediction, we try to forecast the Close price for future dates using historical trends.*

5. Data Preprocessing

Preprocessing is a crucial step in preparing stock price data for prediction models. Here's how each step is handled:

1.Handle Missing Values

- *What it means: Sometimes, data may be missing for certain dates or fields.*
- *What to do:* ○ *Removal: If very few rows have missing values, we can simply drop them.*

Df.dropna(inplace=True)

○ *Imputation: If many values are missing, we can fill them using:* ❍ *Forward fill (use previous value):*

Df.fillna(method='ffill', inplace=True)

❍ *Mean or median (for numerical columns):*

Df['column'] = df['column'].fillna(df['column'].mean())

2.Remove or Justify Duplicate Records

- *What it means: Sometimes, the same row appears more than once.*
- *What to do:*

Df.duplicated().sum()

Df.drop_duplicates(inplace=True)

1.

3.Detect and Treat Outliers

- *What it means: Outliers are values that are too far from the normal range.*
- *What to do:*
  - *Using IQR (Interquartile Range):*

Q1 = df['Close'].quantile(0.25) Q3

= df['Close'].quantile(0.75)

IQR = Q3 – Q1

Df = df[~((df['Close'] < (Q1 – 1.5 * IQR)) | (df['Close'] > (Q3 + 1.5 * IQR)))]

4.Convert Data Types and Ensure Consistency

- *What it means: Make sure each column has the correct data type (e.g., date as datetime, price as float).*
- *What to do:*

Df['Date'] = pd.to_datetime(df['Date'])

Df['Close'] = df['Close'].astype(float)

5.Encode Categorical Variables

- *What it means: If you have text labels (e.g., stock names, sectors), convert them to numbers.*
- *What to do:*
  - *Label Encoding:*

From sklearn.preprocessing

import LabelEncoder
Le = LabelEncoder()
Df['Category'] = le.fit_transform(df['Category']) ○

One-hot Encoding:

Df = pd.get_dummies(df, columns=['Category'])

6.Normalize or Standardize Features

- *What it means: Bring all values to the same scale (especially needed for neural networks).*
- *What to do:*
  - *Min-Max Normalization:*

From sklearn.preprocessing import MinMaxScaler

Scaler = MinMaxScaler()

Df[['Open', 'High', 'Low', 'Close']] = scaler.fit_transform(df[['Open', 'High', 'Low', 'Close']])

  - *Standardization (Z-score):*

From sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()

Df[['Open', 'High', 'Low', 'Close']] = scaler.fit_transform(df[['Open', 'High', 'Low', 'Close']])

6. Exploratory Data Analysis (EDA)

In this section, we perform a comprehensive statistical and visual analysis of stock market data to understand its structure, relationships, and potential impact on predictive modeling.

1. Univariate Analysis

We analyze individual features to understand their distribution and detect outliers.

i. Price-related Features:

- *Closing Price: ○ Histogram: Shows how frequently closing prices occur. ○ Boxplot: Identifies outliers and overall distribution.*
  - *Insight: Stock prices tend to cluster within certain ranges; extreme spikes/drops are rare but critical.*
- *Volume:*
  - *Histogram/Boxplot: Reveals skewness and volatility.*
  - *Insight: Often right-skewed; high volumes may correlate with major market events.*

ii. Technical Indicators:

- *Moving Averages (e.g., 20-day, 50-day MA):*

  ○ Line plots: Show smoothing of price trends. ○ Insight: Help detect bullish or bearish trends.

- *Returns (daily % change):*

  - *Histogram: Often centered near zero with heavy tails.*
  - *Insight: Useful for volatility modeling.*

2.Bivariate/Multivariate Analysis

We explore how features relate to each other and the target variable (future stock price).

I. Correlation Matrix:

o Heatmap of Pearson correlation: Shows interdependence between variables. o Insight: Returns may correlate with indicators Like RSI, MACD, etc.

II. Pairplot/Scatterplot:

- *Compares features Like Volume us. Close Price, MA us. Close Price.*
- *Insight: Some indicators may show a clear predictive trend.*

III. Grouped Line/Bar Plots:

- *E.g., average return by day of the week or by month.*
- *Insight: Captures seasonality or day-of-week effects.*

IV. Target Relationship Analysis:

- *Lag features: Compare today's price with previous days (e.g., t-1, t-2).*
- *Insight: Helps capture autocorrelation in time series.*

3.Insights Summary

a. Key Patterns and Observations:

1.

- *Price trends show seasonal behaviors and autocorrelation.*
  - *Volume spikes often align with market news or earnings reports.*
- *Technical indicators Like Moving Averages, RSI, and Bollinger Bands can signal potential price movements.*
  - *Volatility increases before/after major market events useful for risk modeling.*

b. Features Likely to Influence the Model:

Lagged Prices (t-1, t-2, t-3): Capture temporal dependencies. Technical Indicators (MACD, RSI, MA): Represent market momentum and trend direction. Volume: Often signals market interest and potential price change. Volatility metrics (ATR, rolling std): Indicate risk, which correlates with price movement magnitude.

7. Feature Engineering

In this phase, we focused on enhancing and transforming the raw stack price data to improve the performance of predictive models. The following techniques and transformation were applied

1. New Feature Creation Based on Domain Knowledge and EDA Insight

- *Moving Averages (MA): We introduced 5-day, 10-day, and 20day moving averages to capture short-term and Lang-tarn trends in the stock prices.*
- *Exponential Moving Average (EMA): This was added to give more weight to recent prices, which can help the model, detect more current market movements.*
- *Daily Returns Calculated as the percentage change between consecutive closing prices, this feature helps capture market volatility.*

- *Relation Strength Index (RSI): An important technical Indicator used to Identify overbought or oversold conditions.*

2.Date-time feature Extraction:

- *We extracted components such at Year, Month, Day, and Day of the Week from the date column to help the model understand any temporal patterns in stock movement.*

3.Feature Transformation Techniques:

- *Binning: We binned volatility values Into categories Like Low, medium, and high, which can aid in classification or risk analysis tasks*

- *Ratio: Price ratios Like high/Low and Close/open were created to normalize and scale price variations, making the model more robust*

4.Polynomial Features (Optional):

- *Polynomial transformations of variables Like Close price and Volume (eg., squared terms) were experimented with to capture potential non-Linear relationships.*

5.Dimensionality Reduction (Optional):

- *Principal Component Analysis (PCA): Applied for experimental purposes to reduce multicollinearity and compress the feature space Into fewer dimensions. However, we ultimately opted not to me PCA in the final model, ta maintain interpretability*

6.Feature Selection and Justification:

1.

- *Irrelevant or redundant columns such as Ticker, Unnamed: 0, or columns with constant values were removed.*

- *Final feature selection was guided by performance metrics and correlation Analysis to ensure each included feature contributed meaningfully to model accuracy,*

8. Model Building

To address the problem of stack price prediction, we built and compared multiple machine Learning models. Since the objection Involves predicting a continuous (stack price), un treaded it as a regression problem

Selected Models

We selected the following two models for Implementation

1.Linear Regression:

- *A simple, Interpretable model that assumes a Linear relationship between Input features and the target (stack price).*
- *Chosen as a baseline model due to its speed and ease of Interpretation*

2. Random Forest Regressor:

- *An ensemble Learning method based on decision*
- *Selected for its ability to model complex, nonlinear relationships and handle High-dimensional data without requiring feature scaling*

These models were chosen to strike a balance between simplicity and performance Linear Regression at a benchmark and Random Forest as a powerful, non-linear model

Data Splitting

- *The dataset was divided into training and testing sets using a standard 80/20 split.*
- *Since it is a time series problem, chronological order wet maintained to avoid data Leakage (Ta, no shuffling was done during the split).0*
- *Stratification was not applicable here, as stratification is generally used for classification problem.*

Model Training and Evaluation

Each model was trained on the training set and evaluated on the testing set using the following regression metrics

- *Mean Absolute Error (MAE): Measures the average magnitude of errors in predictions*
- *Root Mean Squared Error (RMSE): Penalizes Larger errors more heavily than MAC.*
- *R^2 Score: Indicates how well the model, explains the variability of the target variable.*

Initial Performance Summary (Example values: replace with your actual results).

Model	**MAE**	**RMSE**	R score
Linear regression	*6.73*	*8.91*	*0.84*

1.

Random forest regressor	*4.13*	*5.35*	*0.93*

From the Initial results, the Random Forest Regressor outperformed Linear Regression across all metrics, suggesting it captured more complex patterns in the data effectively.

9. Visualization of Results & Model Insights

To better understand the behavior and performance of models, we used a variety of visualizations. These plots provided insight into model accuracy, feature Influence, and prediction errors.

1.Actual vs Predicted Plot

- *Description: Line plot comparing actual stock prices with predicted values over time for both models.*
- *Purpose: To visually assess how closely the model tracks real stock movements*
- *Insight: The Random Forest Regressor showed predictions that closely followed actual price trends, indicating strong performance. The Linear Regression model showed more deviation in volatile periods.*

2.Residual Plot

- *Description: Scatter plot of residuals (errors) versus predicted values.*
- *Purpose: To detect patterns in errors and evaluate whether the model, assumptions hold.*
- *Insight: The Random Forest model's residuals were mere randomly distributed around zero, suggesting better error management. The*

Linear Regression model showed slight patterns, Indicating underfitting in some areas,

3.Feature Importance Plat (for Random Forest Regressor)

- *Description: Bar chart chewing the top features contributing to predictions.*
- *Top Features Identified:*

  ○ 10-day Moving Average ○ Volume ○ RSI (Relative Strength Index) ○ Close/Open ratio ○ Day of the week

- *Insight: Technical indicators and recent price trends had the most influence, validating their inclusion during feature engineering.*

4.Error Distribution Histogram

- *Description: Histogram showing the distribution of prediction errors. • Purpose: To evaluate the spread and symmetry of the model's errors.*
- *Insight: The Random Forest model had a tighter, more centered error distribution, while Linear Regression showed a wider spread.*

5.Madel Performance Comparison Chart

- *Description: Bar chart comparing MAE, RMSE, and R scores across models.*
- *Insight: Helped visually demonstrate how Random Forest outperformed Linear Regression across all hay metrics, making it a better choice for this task,*

10. Tools and Technologies Used

The successful execution of this project involved a combination of programming tools, development environments, Libraries, and visualization platforms.Below is a summary of the tools and technologies utilized during different stages of the project:

1. Programming Language

- *Python: Chosen for its simplicity, flexibility, and the rich ecosystem of data science and machine Learning Libraries. Python enabled efficient data manipulation, model development, and result visualization.*

2.Development Environment

- *Google Colab: Used as the primary notebook environment due to its cloud-based access, GPU support, and ease of sharing code.*
- *Jupyter Notebook: Also used for Local testing and interactive development, especially during early experimentation phases.*

3.Key Libraries and Frameworks

- *Pandas: For data Loading, preprocessing, and manipulation.*
- *Numpy: For numerical computations and array operations.*
- *Matplotlib & seaborn: For generating static visualizations such as Line plots, bar charts, and residual plots.*
- *Scikit-Learn: For model building, training, evaluation, and feature engineering techniques.*
- *XGBoost: Explored for boosting-based regression models during experimentation.*
- *Statsmodels: Used occasionally for statistical exploration and baseline Linear modeling.*

4.Visualization Tools

- *Plotly: Used to create interactive charts for model performance and feature trends.*
- *Tableau (optional): Considered for dashboard-style visualizations in case of presenting the insights to a non-technical audience.*

11. Team Members and Contributions

NAVEENKUMAR R :

- *Data cleaning / EDA*

GOKULA KANNAN B :

- *Feature engineering / data acquisition*

BAVITHRA S :

- *Model development*

MONIKA E :

- *Documentation and reporting*