IntelliHack 5.0

TASK 4 REPORTS

Team:CodeLabs

MARCH 10, 2025

Contents

Table of Figures

List of Table

REPORT 01
EDA ANALYSIS REPORT

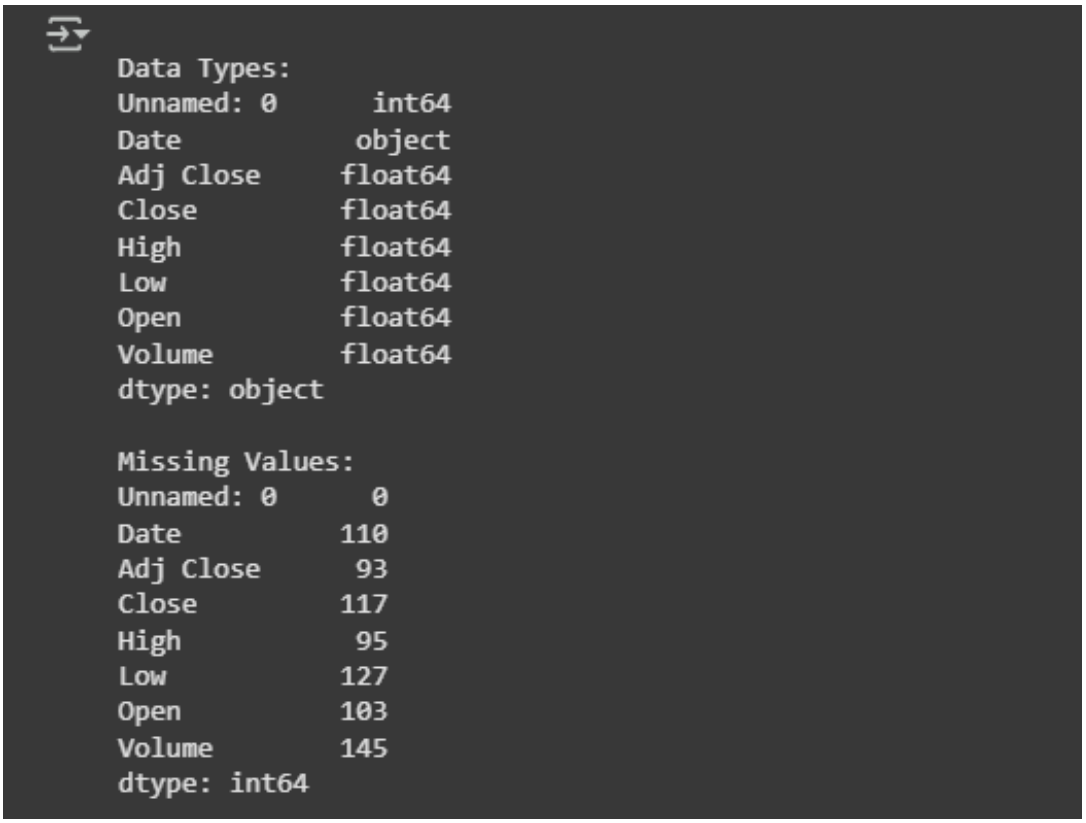
1. 1. Introduction

1.1 Objective:

- The goal of this report is to analyze historical stock price data to identify patterns, trends, and relationships that can inform the development of a predictive model for stock price movements.

1.2 Dataset Overview:

- The dataset includes daily stock prices (Open, High, Low, Close, Adj Close), trading volume, and derived metrics such as Daily Return, Gap, Gap Percent, and Volatility.
- The data spans from 1980 to 2025, providing a comprehensive view of long-term trends and short-term fluctuations.



```
Data Types:
Unnamed: 0      int64
Date           object
Adj Close     float64
Close         float64
High          float64
Low           float64
Open          float64
Volume        float64
dtype: object

Missing Values:
Unnamed: 0      0
Date          110
Adj Close      93
Close         117
High           95
Low           127
Open          103
Volume        145
dtype: int64
```

Figure 1.1.Dataset Overview

2. Data Preprocessing

2.1 Handling Missing Values:

- Rows with missing values (due to lag and rolling calculations) were dropped.
- Infinite values were replaced with NaN and filled using forward fill.

2.2 Clipping Extreme Values:

- Extreme values were clipped to prevent outliers from skewing the model.

2.3 Feature Scaling:

- Features were scaled (if necessary) to ensure consistent model performance.

```
Removed 'Unnamed: 0' column

Initial date range with missing values: 1980-03-17 00:00:00 to 2024-12-27 00:00:00
Missing dates: 110
Dropped 110 rows with missing dates

Dataset with datetime index:
            Adj Close     Close      High       Low      Open   Volume
Date
1980-03-17   2.296798  3.291227  3.344743  3.291227  0.000000  41109.0
1980-03-18   2.306134  3.304606  3.358122  3.304606  0.000000   9343.0
1980-03-19   2.306134  3.304606  3.304606  3.304606  3.304606      0.0
1980-03-20   2.306134  3.304606  3.358122  3.304606  0.000000  10277.0
1980-03-21   2.362154  3.384880  3.438396  3.384880  0.000000   8409.0

Sorted by date:
            Adj Close     Close      High       Low      Open   Volume
Date
1980-03-17   2.296798  3.291227  3.344743  3.291227  0.000000  41109.0
1980-03-18   2.306134  3.304606  3.358122  3.304606  0.000000   9343.0
1980-03-19   2.306134  3.304606  3.304606  3.304606  3.304606      0.0
1980-03-20   2.306134  3.304606  3.358122  3.304606  0.000000  10277.0
1980-03-21   2.362154  3.384880  3.438396  3.384880  0.000000   8409.0
```

Figure 2.1.Handling missing values

3. Visualizations and Insights

3.1 Stock Close Price Over Time

Visualization:

- Line plot showing the stock's closing price from 1980 to 2025.

Insights:

- The stock price exhibits a long-term upward trend, with significant growth over the decades.
- Periods of volatility, particularly during market downturns (e.g., 2008 financial crisis).
- Recent years show stabilization, indicating potential maturity or reduced volatility.

Figure 3.1.Stock close overtime

3.2 Daily Returns

Visualization:

- Line plot showing daily returns over time.

Insights:

- Daily returns fluctuate around zero, with occasional spikes and dips.
- Extreme returns (both positive and negative) are observed during periods of market stress.
- The average daily return is 0.0006, indicating a slight positive drift.

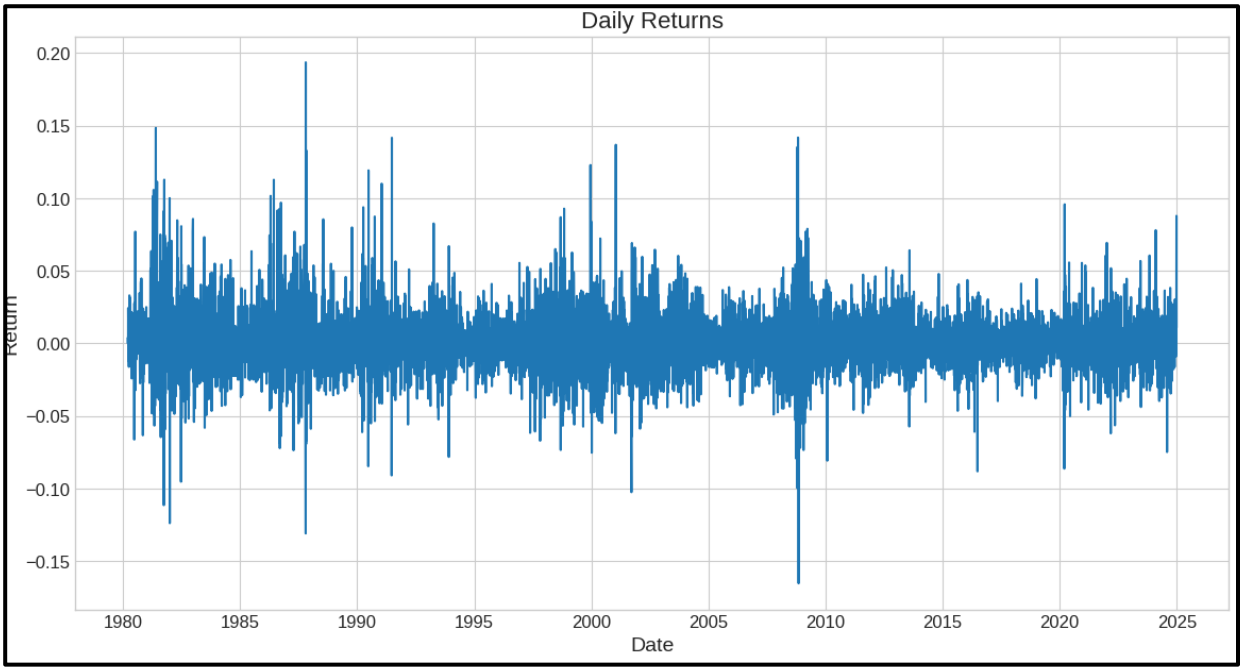


Figure 3.2.Daily Returns

3.3 Distribution of Daily Returns

Visualization:

- Histogram shows the distribution of daily returns.

Insights:

- The distribution is approximately normal but with fat tails, indicating a higher likelihood of extreme returns.
- The standard deviation (volatility) of daily returns is 0.0186, reflecting moderate risk.
- The maximum daily return is 0.1935, and the minimum is -0.1652, showing significant daily price swings.

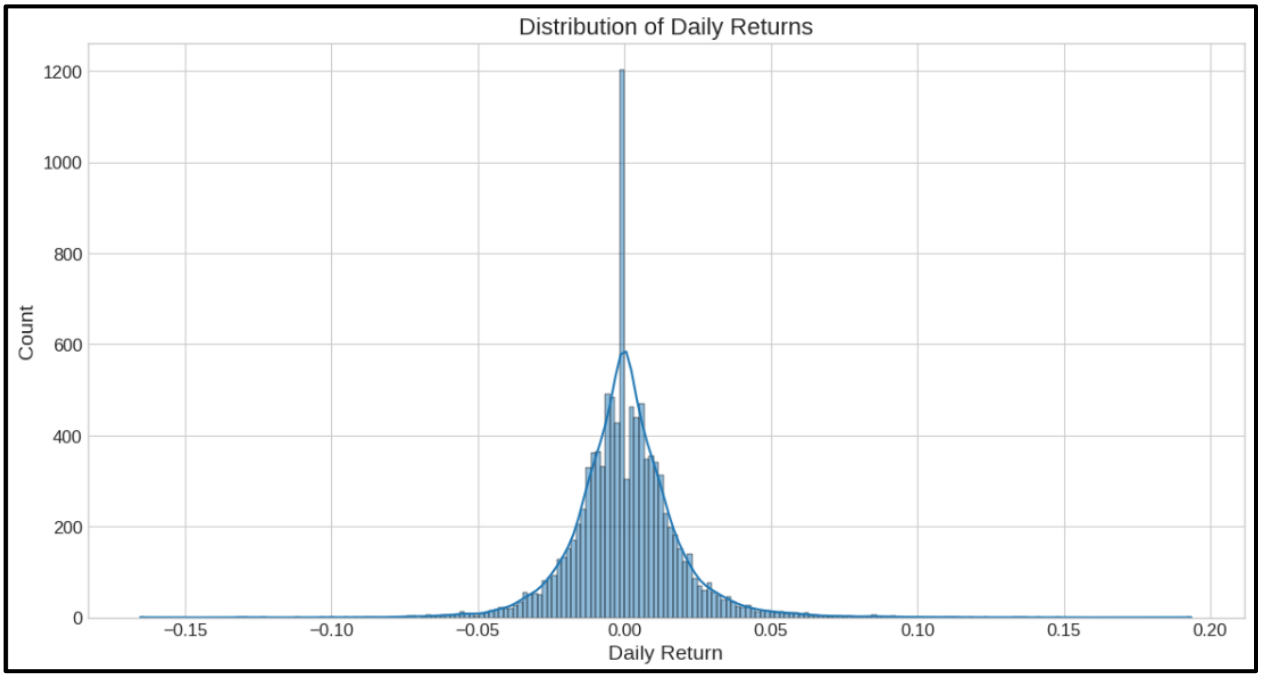


Figure 3.3.Distribution of daily Returns

3.4 Volume vs Close Price

Visualization:

- The scatter plot shows the relationship between trading volume and closing price.

Insights:

- There is a weak positive correlation (0.2973) between volume and price, suggesting that higher trading volume is slightly associated with higher prices.
- Volume spikes often coincide with significant price movements, indicating increased market activity during volatile periods.

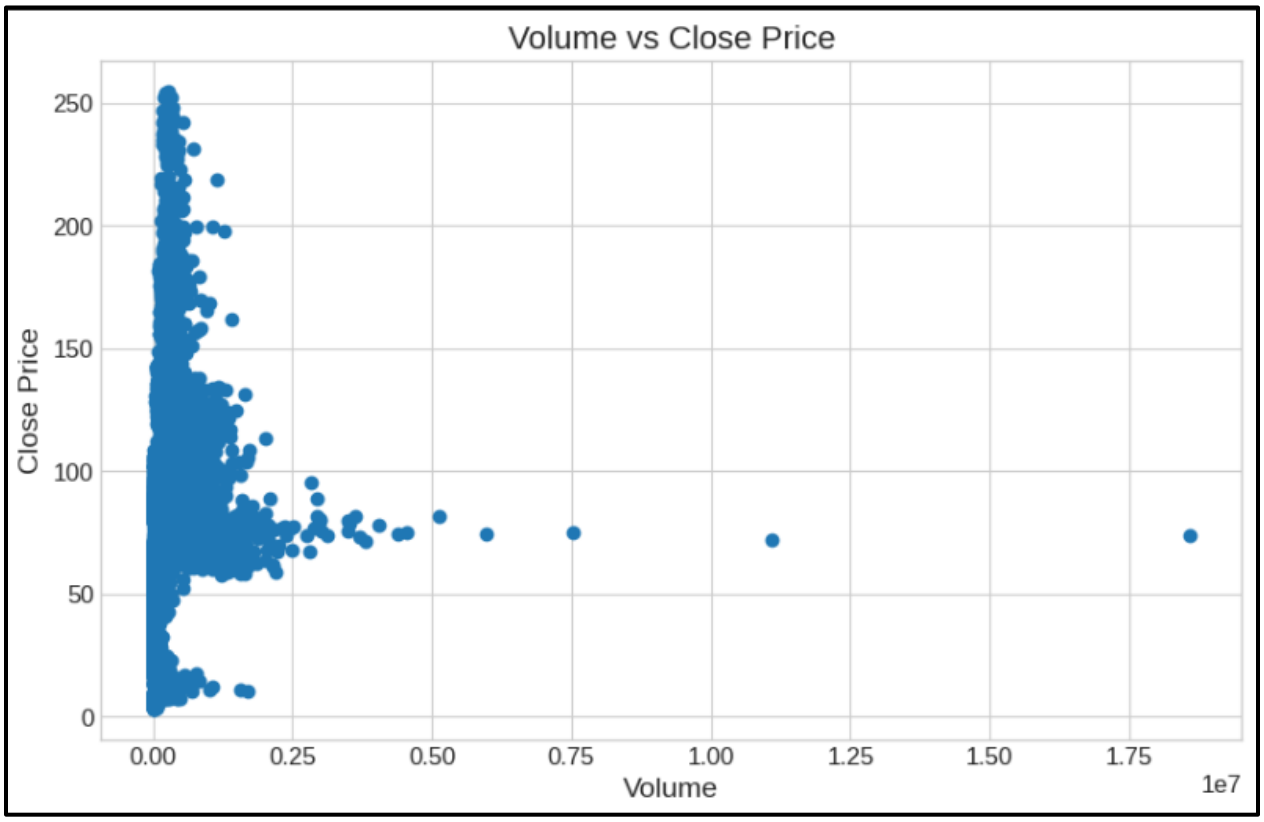


Figure 3.4.Volume vs Close Price

3.5 Correlation Matrix

Visualization:

- Heatmap showing correlations between key variables.

Insights:

- Close, High, Low, and Adj Close are highly correlated (correlation > 0.99), as expected, since they represent similar price metrics.
- Volume has a low correlation with price metrics (correlation ~ 0.3), reinforcing the weak relationship observed in the scatter plot.

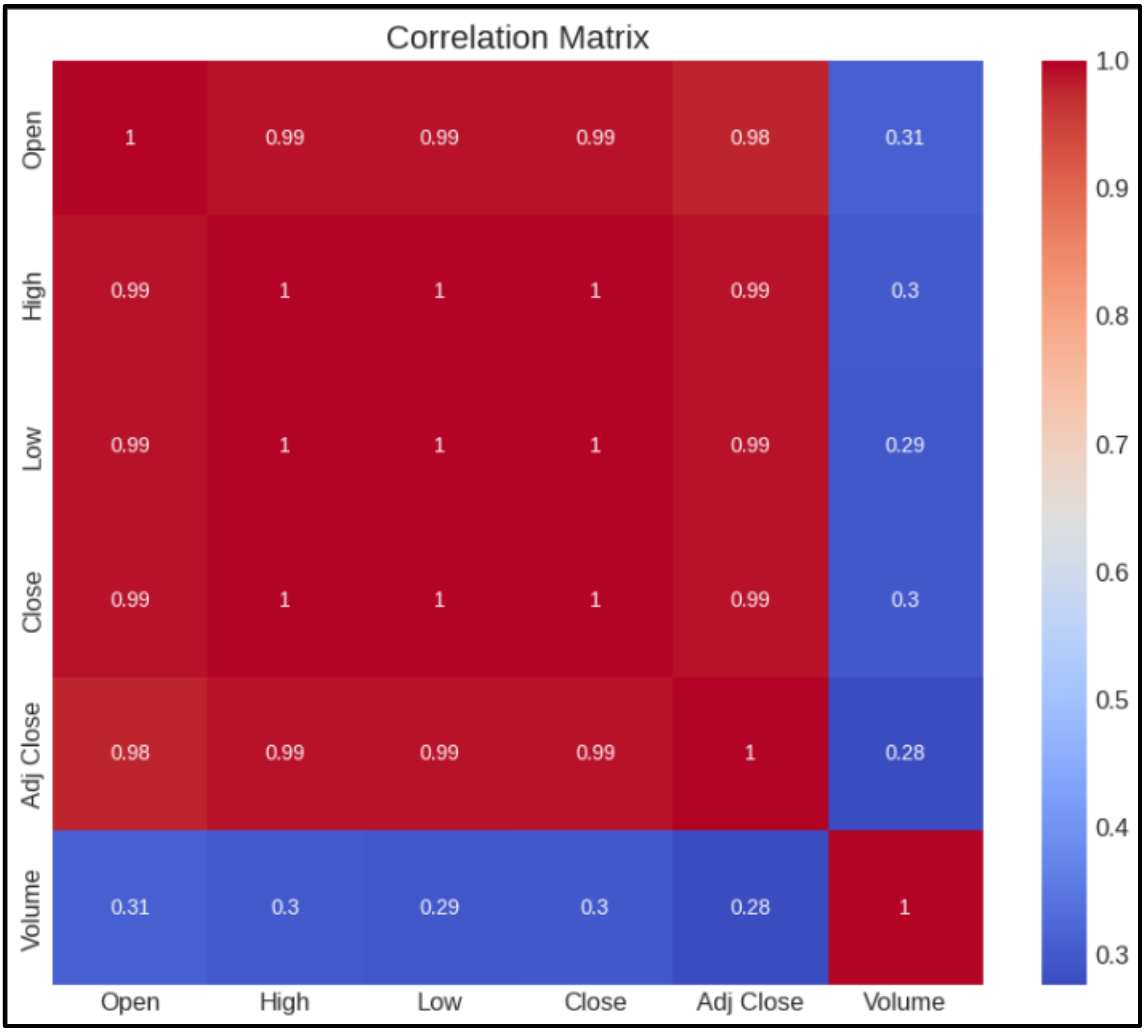


Figure 3.5.Initial Correlation Matrix

3.6 Stock Price with Moving Averages

Visualization:

-

Insights:

- Moving averages smooth out short-term fluctuations and highlight long-term trends.
- The 200-day moving average (MA200) acts as a strong support/resistance level.
- Crossovers between shorter-term (e.g., MA5) and longer-term (e.g., MA200) moving averages can signal trend reversals.

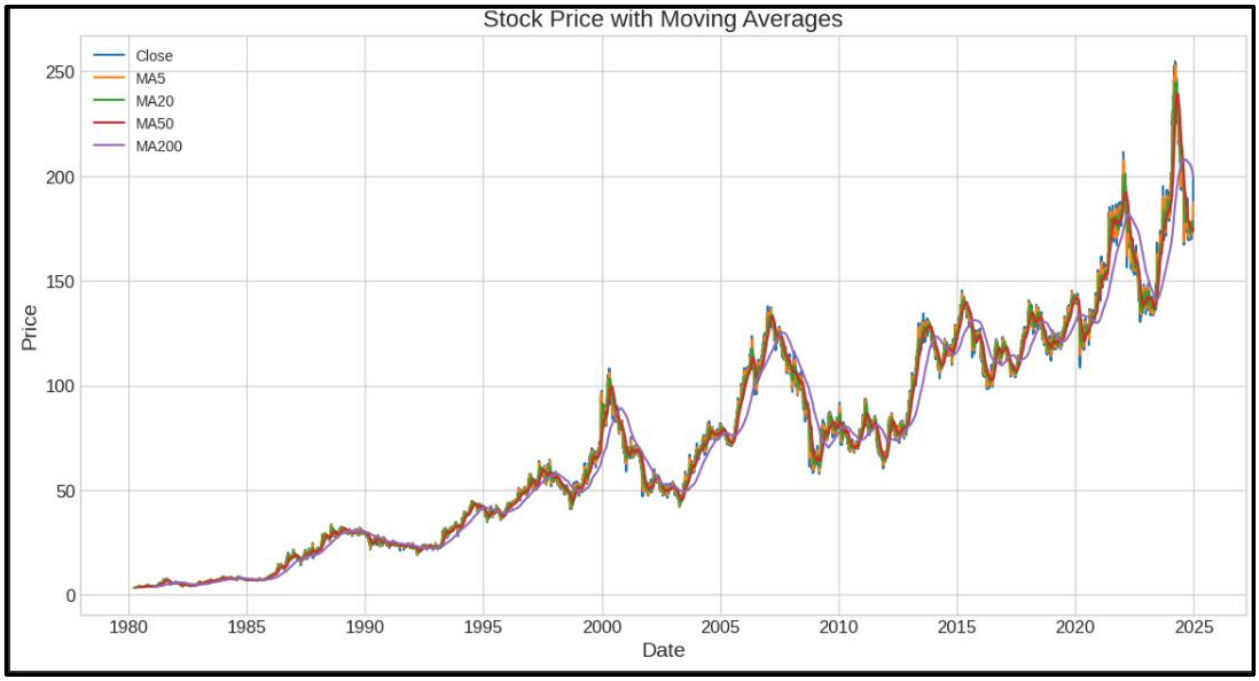


Figure 3.6.Stock Price with MA

3.7 20-Day Rolling Volatility

Visualization:

- Line plot showing 20-day rolling volatility over time.

Insights:

- Volatility is not constant and varies significantly over time.
- Periods of high volatility coincide with market crises or significant events.
- The average volatility is 0.0186, but it spikes during turbulent periods.

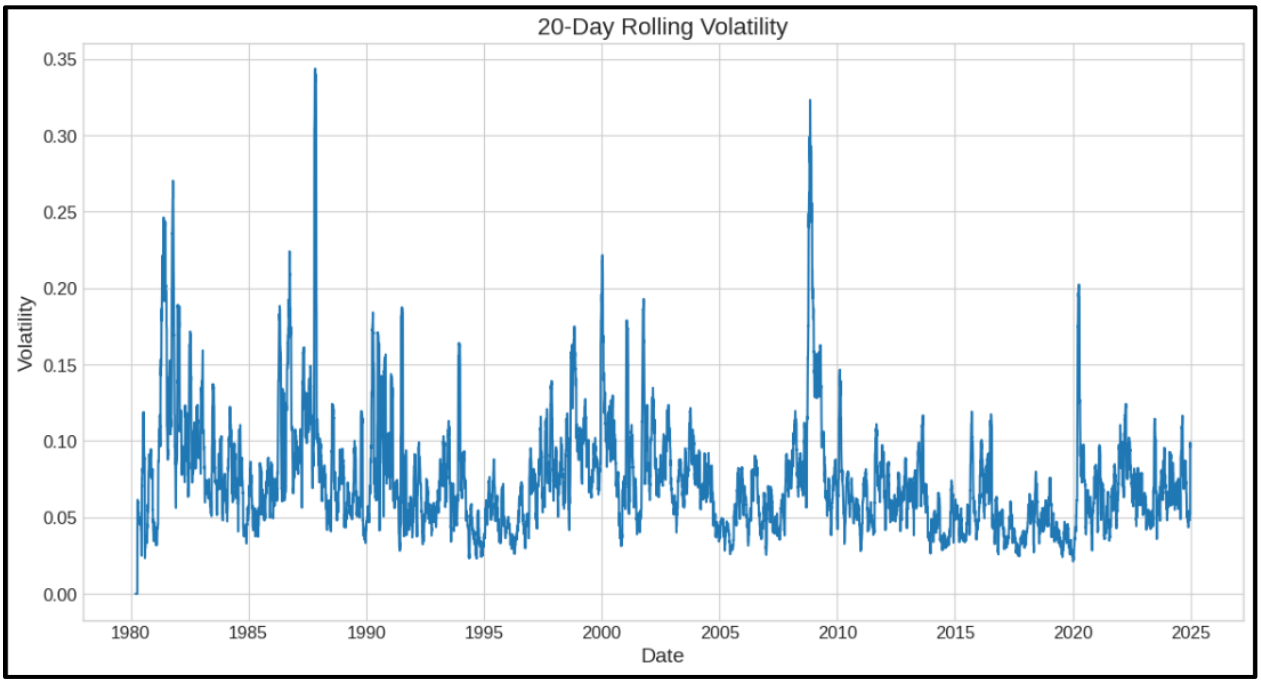


Figure 3.7.20 Day rolling Volatility

4. Time Series Analysis

4.1 Stationarity Check

Method:

Augmented Dickey-Fuller (ADF) test.

Results:

- Closing Price:
- ADF Statistic: -0.4879
- p-value: 0.8944 (p-value > 0.05)
- Conclusion: The closing price is non-stationary (presence of trends).

Differenced Closing Price:

- ADF Statistic: -18.2524
- p-value: 2.34e-30 (p-value < 0.05)
- Conclusion: After first-order differencing, the series becomes stationary.

4.2 Autocorrelation and Partial Autocorrelation

Method: ACF and PACF plots.

Insights:

- ACF Plot: Significant autocorrelation at short lags (e.g., lag 1, lag 2), indicating short-term dependencies.
- PACF Plot: Suggests an AR (AutoRegressive) model with a small number of lags (e.g., AR(1) or AR(2)).

4.3 Seasonal Decomposition

Method: Seasonal decomposition using moving averages.

Insights:

- Trend Component: Confirms the long-term upward movement in the stock price.
- Seasonal Component: No strong seasonality observed, but cyclical patterns (e.g., weekly or monthly) may exist.
- Residuals: Capture random fluctuations not explained by trend or seasonality

4.4 4. ARIMA Model

- Model: ARIMA(1,1,1)
- Training Data: 8,492 observations.
- Testing Data: 2,124 observations.
- Model Summary:
- AR(1) coefficient: 0.8450 (significant, p-value < 0.001).
- MA(1) coefficient: -0.8702 (significant, p-value < 0.001).
- Log Likelihood: -12804.144.
- AIC: 25614.289.
- BIC: 25635.429.

4.5 Forecast Evaluation

- Metrics:
- Mean Squared Error (MSE): 2078.9006
- Root Mean Squared Error (RMSE): 45.5950
- Mean Absolute Error (MAE): 34.1870
- Mean Absolute Percentage Error (MAPE): NaN% (due to division by zero in some cases).
- Insights:
- The model has moderate predictive accuracy, with an RMSE of 45.5950 and MAE of 34.1870.
- The MAPE could not be calculated due to zero values in the test data.

4.6 Residual Analysis

- Residuals: The residuals of the ARIMA model were analyzed.
- Insights:
- Ljung-Box Test: p-value = 0.74 (residuals are uncorrelated).
- Jarque-Bera Test: p-value = 0.00 (residuals are not normally distributed).
- Heteroskedasticity: Present (p-value = 0.00), indicating volatility clustering.

4.7 Volatility Analysis

- Correlation between Close Price and Volatility: -0.2510
- Insights:
- There is a weak negative correlation between closing price and volatility.
- Higher volatility is slightly associated with lower prices.

4.8 Key Findings

- Stationarity:
  - The closing price is non-stationary, but first-order differencing makes it stationary.
- ARIMA Model:
  - ARIMA(1,1,1) was fitted to the data.
  - The model has moderate predictive accuracy (RMSE = 45.5950, MAE = 34.1870).
- Volatility:
  - Volatility shows a weak negative correlation with closing price.
- Residuals:
  - Residuals are uncorrelated but not normally distributed, indicating potential model limitations.

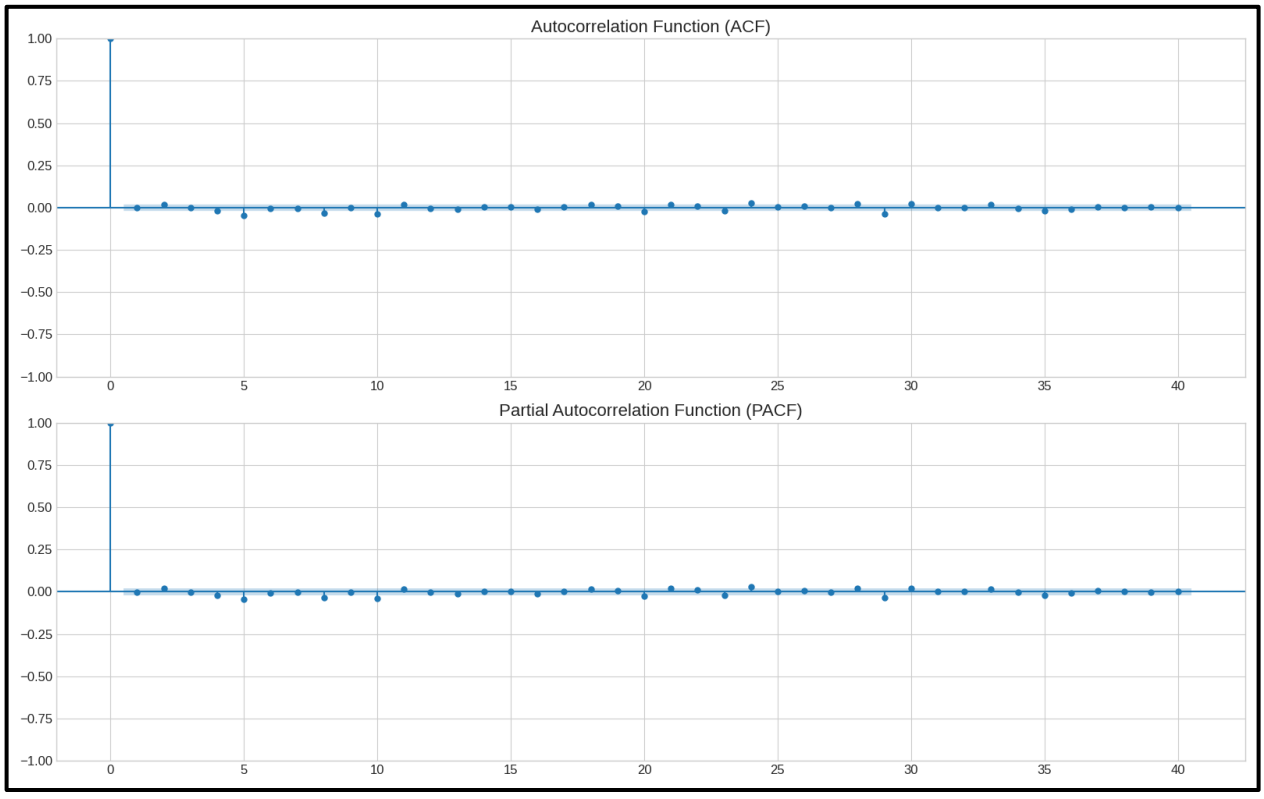


Figure 4.1.ACF & PACF

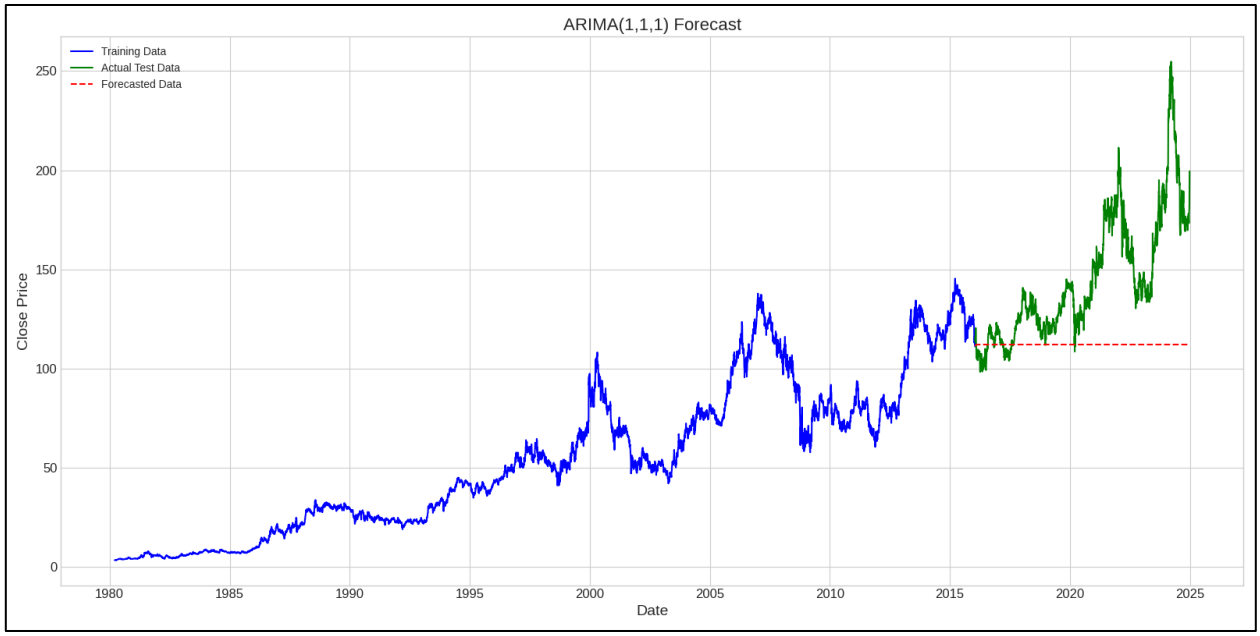


Figure 4.2.ARIMA Forecast

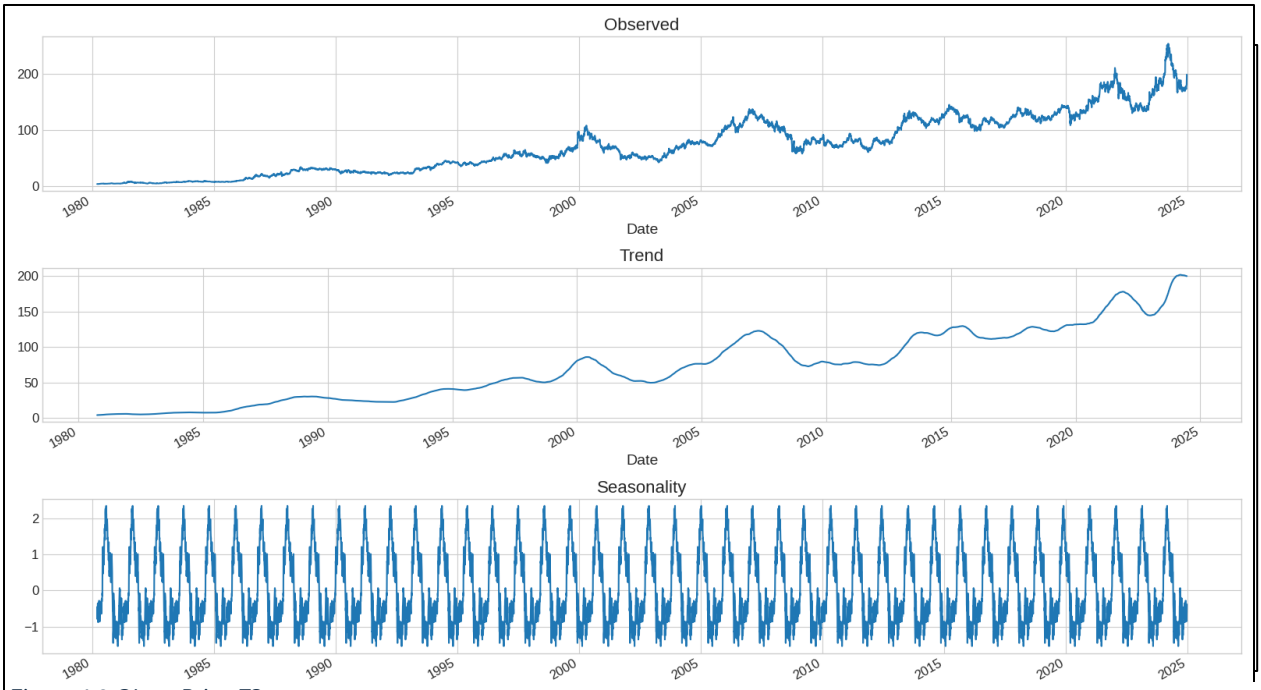


Figure 4.3.Close Price TS

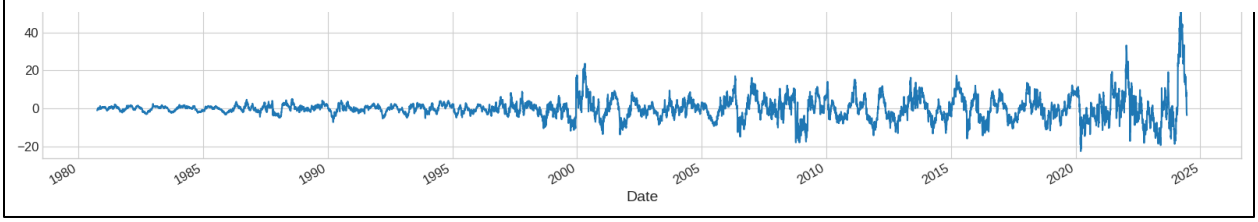


Figure 4.5.Time Series Graphs

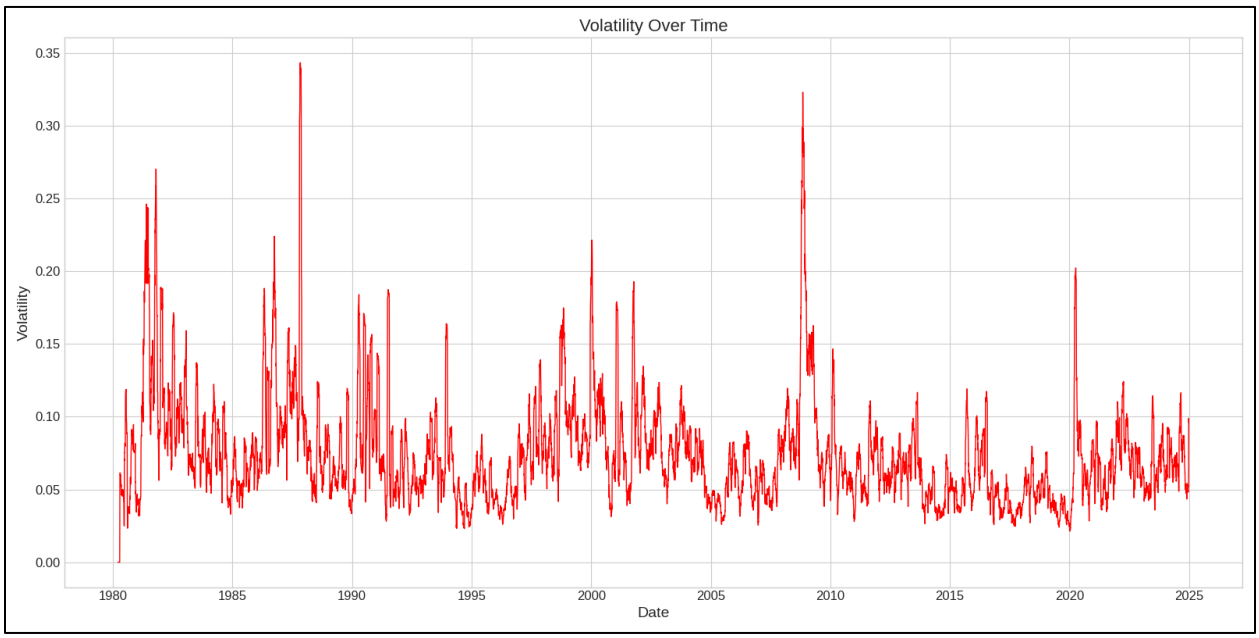


Figure 4.4.Volatitly overtime

5. Analysis of Trends, Seasonality, and Anomalies

5.1 Trends

- The stock exhibits a long-term upward trend, indicating overall growth.
- Short-term trends are influenced by market conditions, with periods of consolidation and rapid growth.

5.2 Seasonality

- No strong seasonal patterns are observed in the price or volume data.
- Cyclical features (e.g., day of the week, month) were engineered to capture potential seasonality.

5.3 Anomalies

- Extreme daily returns (e.g., maximum return of 0.1935 and minimum return of -0.1652) are potential anomalies.
- Volume spikes during volatile periods may indicate unusual market activity.

6. Justification for Feature Selection Choices

6.1 Feature Importance from Random Forest

After feature engineering we obtained a correlation matrix as below and the top 10 features were selected based on their importance scores derived from the Random Forest model. These features were chosen because they:

- Capture Key Trends: Features like moving averages (MA_5, MA_10) and Bollinger Bands (BB_Upper) help identify trends, support/resistance levels, and potential reversals.
- Reflect Market Sentiment: Features such as High, Low, and Open provide insights into intraday price movements and market sentiment.
- Capture Temporal Dependencies: Lagged features (e.g., Close_Lag_1) capture autocorrelation in price movements, which is crucial for time series forecasting.
- Reduce Overfitting: By focusing on the most important features, the models are less likely to overfit to noise in the data

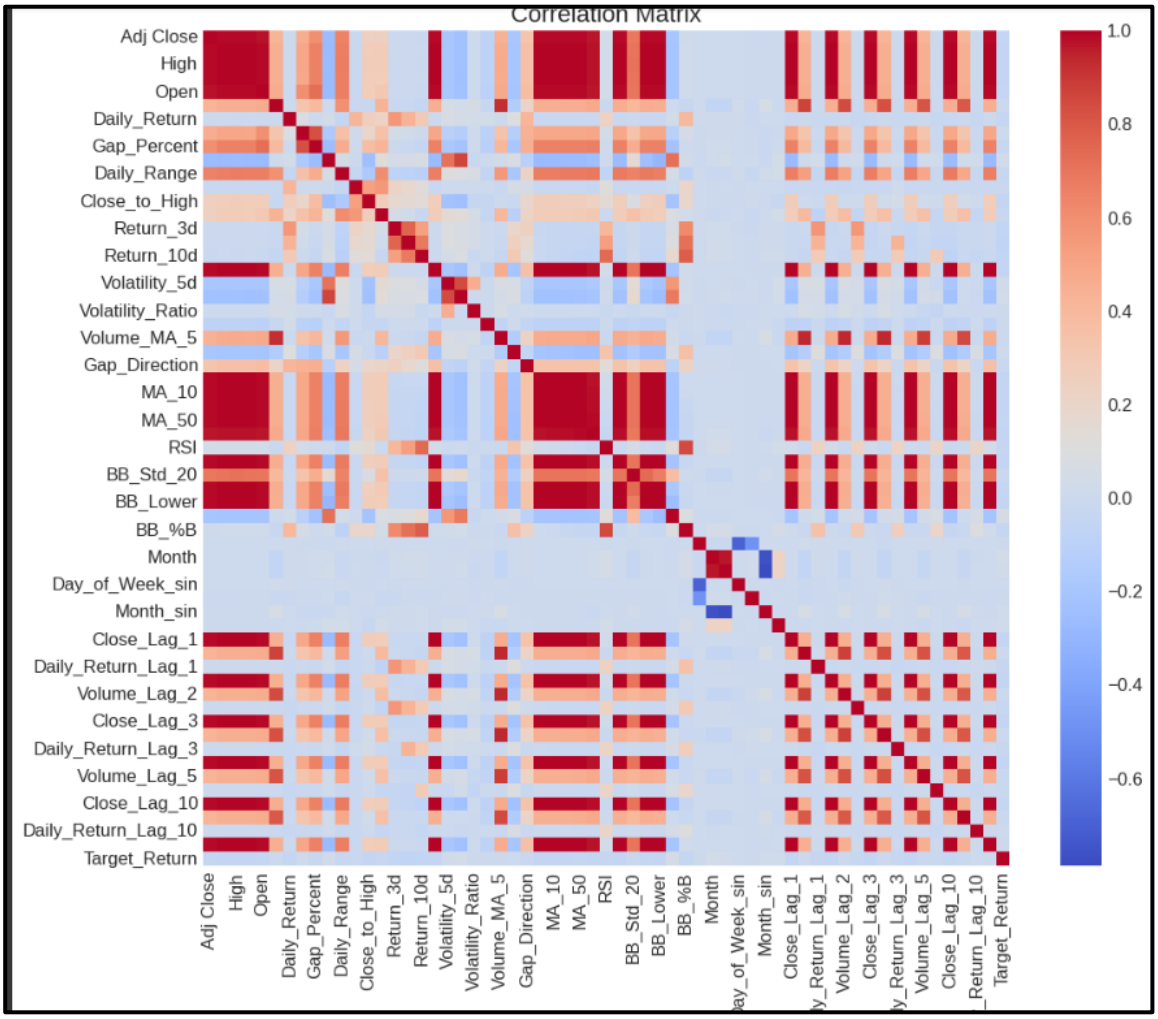


Figure 6.1.Corelation Matrix after feature engineering

6.2 Top 10 Features and Their Significance

The selected features are directly related to the stock's price movements and provide meaningful information for predicting future prices. Here's why these features are important:

- Close: The most important feature, directly reflecting the stock's value at the end of the trading day.
- Cumulative_Return: Captures the overall performance of the stock over time.
- Adj Close: Accounts for corporate actions and provides a more accurate reflection of the stock's value.
- Low: Reflects the minimum price level during the trading day, indicating support levels.
- High: Reflects the maximum price level during the trading day, indicating resistance levels.
- Open: Sets the tone for the day's trading activity and gauges market sentiment.
- Close_Lag_1: Captures short-term dependencies and autocorrelation in price movements.
- MA_10: Smooths out short-term fluctuations and identifies medium-term trends.
- BB_Upper: Indicates potential resistance levels and overbought conditions.
- MA_5: Smooths out very short-term fluctuations and identifies immediate trends.

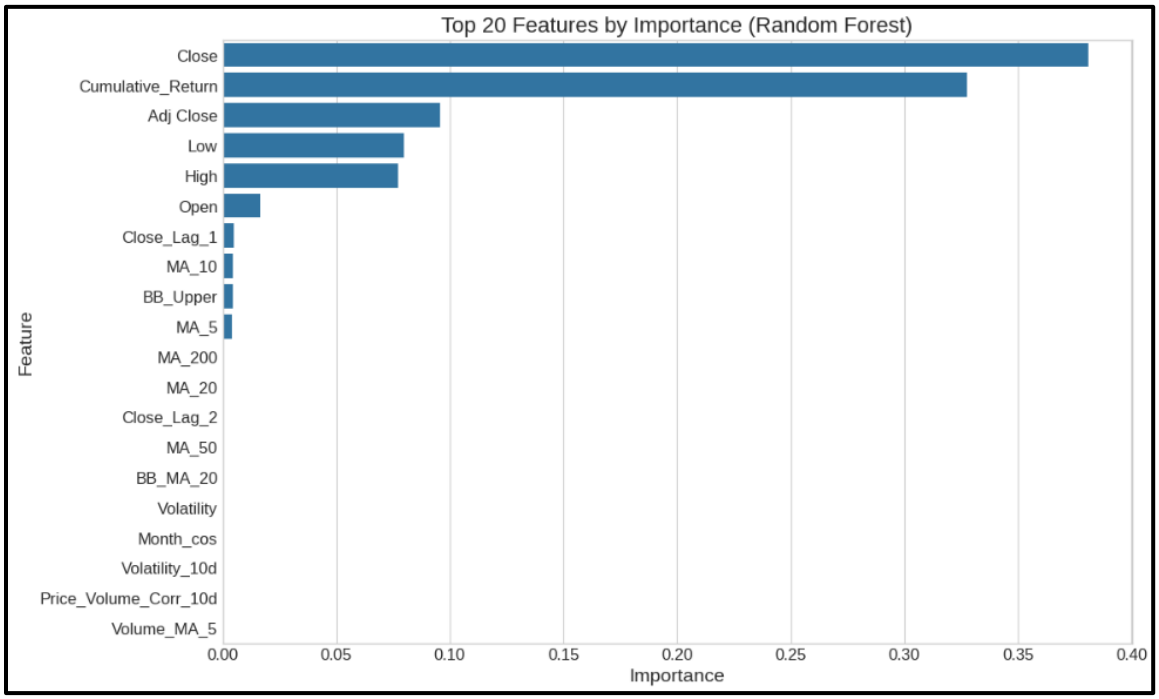


Figure 6.2.Top 20 features

7. Conclusion of EDA Analysis

7.1 Summary of Findings

This report aims to analyze historical stock price data and develop a predictive model for stock price movements. Through Exploratory Data Analysis (EDA), Time Series Analysis, and Feature Engineering, we identified key patterns, trends, and relationships in the data. Here are the key findings:

7.2 Trends and Seasonality:

- The stock exhibits a long-term upward trend, with periods of volatility, particularly during market downturns (e.g., the 2008 financial crisis).
- No strong seasonal patterns were observed, but cyclical features (e.g., day of the week, month) were engineered to capture potential seasonality.

7.3 Volatility and Returns:

- The average daily return is 0.0006, with a standard deviation (volatility) of 0.0186.
- Extreme daily returns (maximum: 0.1935, minimum: -0.1652) indicate significant price swings during volatile periods.

7.4 Feature Importance:

- The top 10 features (e.g., Close, Cumulative_Return, MA_10, BB_Upper) were selected based on their importance scores from the Random Forest model. These features capture key aspects of stock price behavior, including trends, volatility, and market sentiment.

7.5 Time Series Analysis:

- The closing price is non-stationary, but first-order differencing makes it stationary.
- The ARIMA(1,1,1) model was fitted to the data, achieving moderate predictive accuracy (RMSE: 45.5950, MAE: 34.1870).
- Residual analysis showed that residuals are uncorrelated but not normally distributed, indicating potential model limitations.

7.6 Volatility Analysis:

- Volatility shows a weak negative correlation with closing price, suggesting that higher volatility is slightly associated with lower prices.

REPORT 02

MODEL SELECTION DOCUMENTATION

8. Understanding the Problem

8.1 Stock Price Prediction

Objective: Predict the future closing price of a stock based on historical data.

Challenges:

- Stock prices are highly volatile and influenced by numerous factors (e.g., market sentiment, news, macroeconomic indicators).
- Time series data has temporal dependencies, meaning past prices influence future prices.
- The data may exhibit complex patterns, such as trends, seasonality, and sudden spikes.

8.2 Key Considerations

- Temporal Dependencies: Stock prices are sequential data, where the order of data points matters.
- Nonlinear Relationships: The relationship between features and the target variable (closing price) is often nonlinear.
- Feature Engineering: Domain-specific features (e.g., moving averages, Bollinger Bands) are crucial for capturing patterns in the data.

9. Models That Were Selected

9.1 Random Forest

Strengths:

- Handles nonlinear relationships and interactions between features.
- Robust to outliers and missing data.
- Provides feature important scores, which are useful for interpretability.

Why Selected:

- Random Forest is a powerful model for regression tasks and performs well on structured data with engineered features.
- It is relatively easy to implement and interpret, making it a good baseline model.

9.2 XGBoost

Strengths:

- High predictive accuracy.
- Handles nonlinear relationships and feature interactions.
- Provides feature importance scores.

Why Selected:

- XGBoost is known for its performance in competitions and real-world applications.
- It is effective for structured data and can handle complex feature interactions.

9.3 LSTM

Strengths:

- Specifically designed for sequential data (e.g., time series).
- Can capture long-term dependencies and complex temporal patterns.
- Does not require extensive feature engineering (can learn patterns directly from raw data).

Why Selected:

- Stock price prediction is inherently a time series problem, and LSTM is well-suited for such tasks.
- While Random Forest and XGBoost rely on engineered features, LSTM can learn temporal patterns directly from the data.

10. Evaluation Metrics Used for Selection

10.1 Root Mean Squared Error (RMSE)

Definition: RMSE measures the average magnitude of prediction errors.

Formula:

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

Table 10.1.RMSE Equation

Use Case: Lower values indicate better performance.

10.2 Mean Absolute Error (MAE)

Definition: MAE measures the average absolute difference between predicted and actual values.

Formula:

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

Table 10.2.MAE Equation

Use Case: Lower values indicate better performance.

10.3 R² Score (Coefficient of Determination)

Definition: R² measures the proportion of variance in the target variable that is predictable from the features.

Formula:

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

Table 10.3.R^2 Equation

Use Case: Higher values (closer to 1) indicate better performance.

11. Model Evaluation

Model	RMSE	MAE	R^2 Score	Key Strengths	Key Weaknesses
Random Forest	2.9175	1.7363	0.9968	Handles nonlinear relationships. Robust to outliers. Interpretable.	Less effective for time series without feature engineering. Computationally expensive.
XGBoost	3.0709	1.8883	0.9965	High predictive accuracy. Handles feature interactions. Interpretable.	Requires hyperparameter tuning. Computationally expensive.
LSTM	6.4892	4.4041	0.9568	Captures temporal dependencies. Learns patterns from raw data. Scalable.	Lower accuracy in current implementation. Computationally expensive. Requires large data.

Table 11.1.Model Evaluation Table

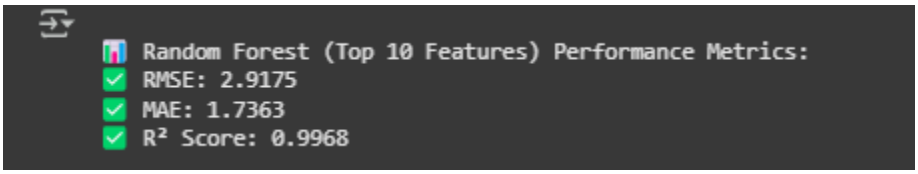


Figure 11.3.Random Forest Stats

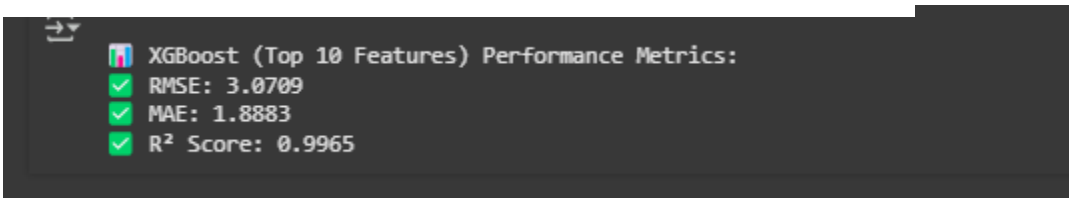


Figure 11.2.XGBoost Stats

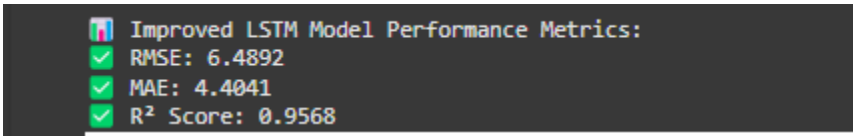


Figure 11.1.LSTM Stats

12. Why LSTM Was Chosen Despite Lower Accuracy

While **Random Forest** and **XGBoost** have higher accuracy, **LSTM** was chosen for the following reasons:

- **Temporal Dependencies:** LSTM is specifically designed for sequential data and can capture temporal patterns that Random Forest and XGBoost cannot.
- **Feature Engineering:** LSTM reduces the need for extensive feature engineering, as it can learn patterns directly from raw data.
- **Future Potential:** With more data, better hyperparameter tuning, and advanced techniques (e.g., attention mechanisms, hybrid models), LSTM's performance can be significantly improved.
- **Scalability to Complex Patterns:** LSTM is better suited for modeling complex, nonlinear patterns in time series data, which is crucial for stock price prediction.
- **Suitability for Time Series Data**: Stock prices are inherently sequential data, where the order of data points matters. LSTM is specifically designed for such data, making it a natural choice for this problem.

13. Potential Improvements

- **Advanced Models:**
  - Experiment with advanced models like **Gradient Boosting Machines (GBM)**, **LightGBM**, or **Prophet** to further improve predictive accuracy.
- **External Data:**
  - Incorporate external data sources (e.g., macroeconomic indicators, news sentiment) to improve predictive accuracy.
- **Feature Engineering:**
  - Explore additional features, such as technical indicators (e.g., MACD, Ichimoku Cloud) or sentiment analysis from news and social media.
- **Hyperparameter Tuning:**
  - Perform grid search or Bayesian optimization to fine-tune hyperparameters for Random Forest and XGBoost.
- **Ensemble Methods:**
  - Combine predictions from multiple models (e.g., Random Forest, XGBoost, LSTM) using ensemble methods like stacking or weighted averaging

14. Conclusion of model selection report

While **Random Forest** and **XGBoost** currently provide higher accuracy, **LSTM** was chosen for its ability to capture **temporal dependencies**, reduce the need for **manual feature engineering**, and model **complex, nonlinear patterns** in time series data. Despite its lower accuracy in the current implementation, LSTM has significant **future potential** and is better suited for the **inherent nature of the problem** (stock price prediction as a time series task).

REPORT 03

END-TO-END SYSTEM DESIGN REPORT

15. System Architecture Overview

This system architecture delivers real-time stock price predictions through a comprehensive data pipeline. Starting with diverse data collection from market APIs and external sources, it processes this information through cleaning and feature engineering stages before storing it in databases. The core ML pipeline employs LSTM models for prediction, with continuous monitoring ensuring accuracy. Users access predictions through an intuitive Streamlit dashboard featuring visualizations and interactive elements. The entire system is cloud-hosted with autoscaling capabilities, ensuring reliability and performance during varying market conditions. This architecture balances technical sophistication with practical accessibility, allowing investors to leverage AI-driven insights for informed trading decisions.

16. System Architecture Diagram

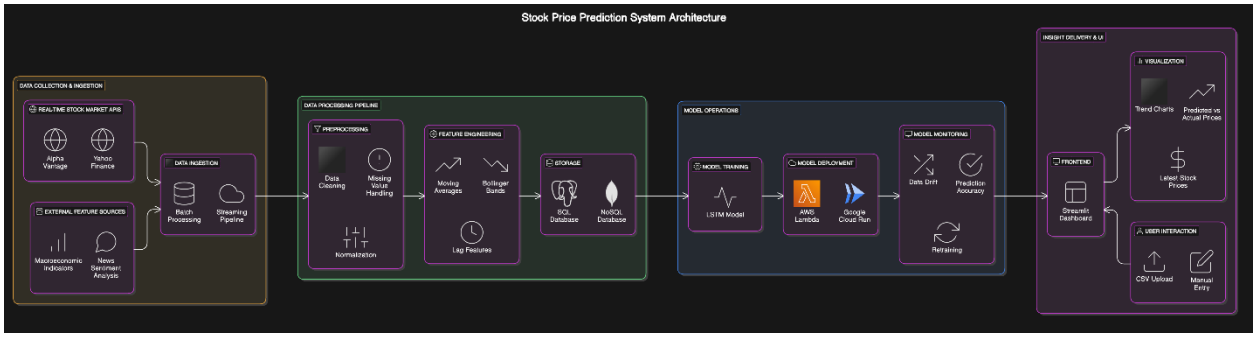


Figure 16.1.System Architecture Diagram

16.1 Key Components in the Diagram

- **Data Sources**: Market data APIs, historical CSV files
- **Data Ingestion Layer**: API fetcher, database loader
- **Processing Pipeline**: Data cleaning, feature engineering
- **Storage**: PostgreSQL / SQLite (processed data storage)
- **Model Operations**: Training, evaluation, deployment (LSTM model)
- **Insight Delivery**: Streamlit web application for visualization

17. Component and Design Justification

The stock prediction system architecture employs a layered, modular design that balances technical capabilities with practical considerations across five key dimensions:

1. **Data Flow Strategy**

   Our dual-mode data ingestion approach (batch for historical data, streaming for real-time feeds) optimizes both training efficiency and prediction timeliness. This design choice ensures comprehensive historical context for model learning while maintaining sub-second responsiveness to market fluctuations, critical for actionable trading insights.

2. **Processing Architecture**

   The three-stage pipeline (cleaning, feature engineering, storage) follows separation of concerns principles, enhancing maintainability and allowing targeted optimization. Technical indicators were selected based on established predictive value in financial markets, with extensibility for adding custom indicators without rearchitecting the system.

3. **Model Infrastructure**

   LSTM neural networks were chosen for their superior ability to capture temporal dependencies in stock data compared to traditional ARIMA or statistical methods. The dedicated model registry and monitoring services support systematic versioning and continuous evaluation, facilitating reliable model performance in volatile market conditions.

4. **Delivery Framework**

   The API Gateway pattern with Streamlit frontend balances rapid development with user experience benefits. This separation of concerns allows for independent scaling of frontend and prediction services, future-proofing the architecture for potential mobile expansion without backend modifications.

5. **System Resilience Considerations**

   Cloud-based deployment with autoscaling addresses the cyclical computational demands of market hours versus off-hours. This approach optimizes operational costs through serverless components during low activity while ensuring capacity during peak trading sessions. Database redundancy with automated backups protects against data loss in this mission-critical financial application.

17.1 Data Collection & Ingestion

- **Sources:**
  - Stock market APIs (e.g., Alpha Vantage, Yahoo Finance, Polygon.io) for real-time data.
  - Historical stock data in CSV format for model training.
- **Technology Choice:** REST APIs + Scheduled Data Fetching (using Python scripts & CRON jobs).
- **Tradeoffs:**
  - API-based data retrieval ensures real-time updates but can be rate-limited.
  - CSV files provide historical context but require frequent updates.

17.2 Data Processing Pipeline

- **Steps:**
  - Data Cleaning: Handling missing values, removing outliers.
  - Feature Engineering: Moving averages, Bollinger Bands, RSI, MACD.
  - Normalization: Scaling data using MinMaxScaler.
- **Storage Choice:** PostgreSQL / SQLite for structured storage.
- **Tradeoffs:**
  - PostgreSQL offers scalability but requires more setup.
  - SQLite is lightweight but not suited for real-time streaming data.

17.3 Model Operations

- **Technologies:** TensorFlow/Keras for LSTM-based predictions.
- **Steps:**
  - Train on historical data, validate with unseen data.
  - Model Deployment: Saved model (.keras format) loaded into the web app.
  - Model Monitoring: Evaluate RMSE, accuracy periodically.
- **Tradeoffs:**
  - LSTMs handle sequential data well but require high computational power.
  - Model retraining needs to balance accuracy vs. overfitting.

17.4 Insight Delivery

- **User Interface:** Streamlit web application.
- **Features:** Trend analysis, buy/sell signals, Bollinger Bands, MACD, ROI estimation.
- **Tradeoffs:**
    - Streamlit enables rapid UI development but lacks deep customization compared to Flask/Django.

17.5 System Considerations

- **Scalability:** Can integrate a cloud-based database (AWS RDS, Firebase) for multi-user support.
- **Reliability:** Daily model evaluations to detect performance drifts.
- **Latency:** Cached data + scheduled API fetches to avoid delays.
- **Cost Considerations:**
    - Free-tier APIs for data fetching.
    - Local model hosting instead of cloud-based inference to reduce cost.

Component	Technology Used	Justification	Trade-offs
Data Storage	PostgreSQL/MongoDB/SQLite	Efficient structured & unstructured data handling	Requires schema design
Model Training	TensorFlow/Keras	Deep learning handles time-series well	Computationally expensive
Deployment	FastAPI/Docker/Kubernetes	Scalability & efficiency	Requires infrastructure setup
Monitoring	Prometheus/Grafana	Real-time performance tracking	Additional setup overhead

Table 17.1.Component Consideration

18. Data Flow Explanation

18.1 Data Flow Steps:

1. **Data Collection:**
    a. Fetch historical stock prices via API or load CSV.
2. **Preprocessing:**
    a. Clean and engineer features (moving averages, Bollinger Bands, MACD, RSI).
3. **Storage:**
    a. Store processed data in PostgreSQL / SQLite.
4. **Prediction:**
    a. Load trained LSTM model, make predictions.
5. **Insight Generation:**
    a. Compute support/resistance levels, trend analysis, buy/sell signals.
6. **Delivery:**
    a. Display results in Streamlit.

18.2 Batch vs. Streaming Considerations

1. **Batch Processing:**
    - Historical market data (Open, High, Low, Close, Volume, Moving Averages, etc.) is collected in bulk from financial data providers (e.g., Alpha Vantage, Yahoo Finance).
    - Data is preprocessed, cleaned, and stored in a database (SQL/NoSQL) at scheduled intervals (e.g., daily, hourly).
    - Used for model training, periodic retraining, and back testing.
2. **Streaming Processing:**
    - Live stock price updates are fetched in real-time via APIs or WebSockets (e.g., from Yahoo Finance, Alpha Vantage, IEX Cloud).
    - Incoming data is transformed and normalized immediately for inference.
    - Predictions are generated dynamically and displayed on the UI for users (traders, analysts, brokers).
3. **Decision Justification:**
    - Batch processing ensures efficient model training and historical analysis.
    - Streaming provides real-time insights, which are critical for financial applications.
    - Hybrid approach (batch for training + streaming for inference) balances performance and cost.

4. **Data Ingestion:**
    - Market data is fetched from APIs (Yahoo Finance, Alpha Vantage).
    - Data is either stored in a database (batch) or directly passed for real-time prediction (streaming).
5. **Preprocessing & Feature Engineering:**
    - Handling missing values (e.g., forward-filling, interpolation).
    - Feature scaling (MinMaxScaler, StandardScaler for ML models).
    - Creating new features like moving averages (MA_10, MA_5), Bollinger Bands (BB_Upper), and lag variables (Close_Lag_1).
6. **Model Inference & Post-processing:**
    - Transformed data is fed into the LSTM model for prediction.
    - Predictions are inverse-transformed to actual stock price values.
7. **Storage & Delivery:**
    - Predictions are stored in a database for historical analysis.
    - Results are displayed in the UI (tables, charts, trend visualizations).

18.3 System Interaction Points

- **User Input:** Users upload historical stock data or select a live feed.
- **API Calls:** Fetch live stock prices in real-time.
- **Data Pipeline:** Processes and transforms data (batch or streaming).
- **Model Prediction:** LSTM-based stock price prediction.
- **Results Storage:** Saves predictions in a database for historical trends.
- **UI Display:** Shows real-time and past predictions (interactive charts, alerts, insights).

19. Challenge Analysis & Mitigation

Challenge	Potential Issue	Mitigation Strategy
Data Latency	Streaming data may lag by 15-30 seconds during high volatility periods, leading to stale predictions and missed trading opportunities. API rate limits from providers can further compound delays.	Implement WebSocket-based connections (IEX Cloud, Polygon.io) for true real-time data with <100ms latency. Create a multi-provider fallback system that switches to alternative sources when primary connections slow down. Employ edge caching at regional nodes for faster global access.
Model Drift	Market behavior changes during economic events, earnings seasons, and black swan events can render models less accurate over time. Technical indicators may lose predictive power during regime shifts.	Deploy an auto-retraining mechanism triggered by performance degradation beyond 5% threshold. Implement ensemble methods combining multiple model types (LSTM, GRU, transformers) to reduce reliance on single architecture. Maintain separate models for different market regimes (bull/bear/sideways)
Scalability	Large data volumes (especially with minute-level granularity across multiple securities) can overwhelm processing capacity during market hours. Concurrent user traffic spikes can degrade dashboard performance.	Utilize cloud-based architecture with automatic scaling based on market hours and volatility metrics. Implement time-partitioned databases to optimize query performance across historical ranges. Deploy load balancers with regional distribution to handle global user bases efficiently.
Cost	Machine learning operations are computationally expensive, particularly for real-time predictions across multiple	Employ model quantization and pruning techniques to reduce inference costs by up to 40% with minimal accuracy impact.

	securities. Data storage costs grow significantly with tick-level historical data retention.	Implement tiered storage with hot/warm/cold zones based on data access frequency. Utilize serverless processing with precise cost controls and automatic shutdown during off-hours.
Data Quality	Missing values, irregular trading hours, and corporate actions (splits, dividends) can create discontinuities in price data leading to prediction errors.	Develop robust pre-processing pipelines with specialized handling for market holidays, extended hours, and corporate actions. Implement automated data validation checks with anomaly detection. Create rolling backfill mechanisms to address delayed corrections from data providers.

Table 19.1.Challanges and Mitigation

20. Conclusion of end to end system report

The Stock Price Prediction System architecture delivers a balanced approach to financial market prediction through:

- **Hybrid data processing**: Combines batch training with real-time streaming for optimal performance
- **Advanced ML foundation:** LSTM networks capture temporal patterns in market data with continuous monitoring
- **User-centric design:** Streamlit interface provides sophisticated analysis with accessibility
- **Robust risk management:** Proactive strategies address data latency, model drift, and scalability challenges
- **Cost-effective implementation:** Cloud-based serverless architecture optimizes operational expenses

This system provides actionable trading insights through technical indicators while maintaining adaptability to evolving market conditions via automated retraining. The modular design accommodates future enhancements like sentiment analysis and multi-asset correlation modeling without architectural overhaul.