Algorithmic Trading

Mukesh Patel School of Technology Management and Engineering

A069 – Harsh Siddhpura

A074- Tanmay Thakar

A077 – Udit Upadhyay

Date – March 29, 2024

Course Number:Data Analysis with Python

Abstract(Brief summary of the project, including objectives, methods, results, and conclusions. Usually, it should be around 100-200 words.)

This project aims to develop a comprehensive stock market analysis application to empower both novice and experienced investors with the tools and insights needed to make informed investment decisions. By leveraging advanced technology and data analytics, the application integrates real-time market data, technical analysis tools, and portfolio management features into a user-friendly interface. Through a review of existing literature and research, we identified the need to democratize access to sophisticated market analysis tools and promote financial literacy and inclusion in the investment community. Our project builds upon previous work by offering a holistic solution that bridges the gap between research and practice, ultimately empowering users to navigate the stock market with confidence and achieve their financial goals.

Table of Contents:

Introduction:

In today's dynamic financial landscape, access to timely and accurate market analysis is crucial for traders and investors to navigate the complexities of the stock market. Recognizing this need, our project aims to develop a comprehensive stock market analysis application that empowers users with robust tools and insights to make informed investment decisions.

Introduction:

In today's dynamic financial landscape, access to timely and accurate market analysis is crucial for traders and investors to navigate the complexities of the stock market. Recognizing this need, our project aims to develop a comprehensive stock market analysis application that empowers users with robust tools and insights to make informed investment decisions.

Background Information about the Project:

The stock market is a complex ecosystem influenced by various factors, including economic indicators, company performance, geopolitical events, and investor sentiment. Traditionally, market analysis involved manual tracking of stock prices, fundamental data, and technical indicators, which could be time-consuming and prone to errors. However, with advancements in technology and data analytics, there's an opportunity to leverage automation and data-driven insights to enhance market analysis processes.

Objectives or Aims of the Project:

* Develop a User-Friendly Interface: Create an intuitive and user-friendly interface that allows users to access a wide range of financial data and analysis tools seamlessly.
* Enable Portfolio Management: Develop portfolio management features that allow users to track their investments, analyze performance metrics, and receive personalized insights for portfolio optimization.
* Input Custom Datasets as per the user need to get Customized data output.

Scope and significance of the project.

This project is important because it has the potential to make advanced market analysis tools accessible to everyone. Whether you're new to investing or have years of experience, this application aims to give you the tools you need to make smart investment decisions and effectively manage your portfolio.

Literature Review:

In recent years, there has been a wealth of research and development in the field of financial technology (FinTech), particularly in the area of stock market analysis applications. A review of relevant literature reveals several key themes and findings that inform our project.

Our project builds upon existing research by addressing the limitations and gaps in current stock market analysis applications. While previous studies have focused on specific aspects of market analysis or targeted institutional investors, our project takes a holistic approach to democratizing access to sophisticated tools for both novice and experienced investors.

Methodology:

1. **Data Collection:**

* Utilize the Yahoo Finance API to fetch historical stock price data for the desired stocks.
* Retrieve necessary features such as Open, High, Low, Close prices, and Volume for the specified time period.
* Calculate Moving Averages (MA) for different periods (e.g., 50, 100, 200 days) using the fetched historical stock price data.
* Ensure data accuracy and reliability for robust predictions.

1. **Data Preprocessing:**

* Handle missing values, outliers, or inconsistencies in the fetched data.
* Normalize the data using MinMaxScaler to scale features to a range between 0 and 1.
* Optionally, adjust for any stock splits or dividend adjustments in the historical data to maintain consistency.

1. **Feature Engineering:**

* Extract additional features such as technical indicators (e.g., RSI, MACD), fundamental data, or market sentiment indicators if available.
* Perform feature selection or transformation to enhance model performance.

1. **Model Training:**

* Split the preprocessed data into training and testing sets.
* Choose an appropriate machine learning or deep learning model for stock price prediction.
* Train the model using the training dataset, considering the target variable as the future stock price or price movement.

1. **Model Evaluation:**

* Evaluate the trained model using appropriate evaluation metrics such as MAE, MSE, RMSE, or accuracy.
* Validate the model's performance on the testing dataset to ensure generalization capability.
* Perform cross-validation if necessary to assess model stability and robustness.

1. **Deployment:**

* Develop a user-friendly interface for the stocks prediction app, allowing users to input desired stocks and view predictions.
* Integrate the trained model with the app's backend, enabling real-time or near-real-time prediction capabilities.
* Implement visualization features to display historical stock price data, moving averages, and predicted future prices.
* Deploy the app on a suitable platform (web, mobile, or desktop) for user accessibility.

1. **Monitoring and Maintenance:**

* Regularly update the model with new data from the Yahoo Finance API to adapt to market changes.
* Monitor the app's performance and user feedback, making necessary improvements or updates.
* Stay informed about changes or limitations in the Yahoo Finance API to ensure continuous data access and reliability.

Code

import numpy as np

import pandas as pd

import yfinance as yf

from keras.models import load\_model

import streamlit as st

import matplotlib.pyplot as plt

import seaborn as sns

model = load\_model('Stock Predictions Model.keras')

st.set\_page\_config(page\_title='Algorithmic Trading And Data Visualization', initial\_sidebar\_state = 'auto')

st.header('Algorithmic Trading And Data Visualization')

stock =st.text\_input('Enter Stock Symnbol :',"GOOG")

start =  st.date\_input("Enter the Staring Date : ","2014-03-28")

end = st.date\_input("Enter the Ending Date : ")

def work():

    data = yf.download(stock, start ,end) # downloading data

    st.subheader('Stock Data')

    st.write(data)

    # saperating data 80% training and testing data 20

    data\_train = pd.DataFrame(data.Close[0: int(len(data)\*0.80)])

    data\_test = pd.DataFrame(data.Close[int(len(data)\*0.80): len(data)])

    #  MinMax scale in sklearn library

    from sklearn.preprocessing import MinMaxScaler

    scaler = MinMaxScaler(feature\_range=(0,1))

    pas\_100\_days = data\_train.tail(100)

    data\_test = pd.concat([pas\_100\_days, data\_test], ignore\_index=True)

    data\_test\_scale = scaler.fit\_transform(data\_test)

    st.title("Data Visualization")

    st.subheader("Volumn Chart")

    # Plot the volume chart

    fig0 = plt.figure(figsize=(10, 6))

    plt.bar(data.index, data['Volume'], color='blue', alpha=0.6)

    plt.xlabel('Date')

    plt.ylabel('Volume')

    plt.title('Volume Chart for ' + stock)

    plt.grid(True)

    plt.tight\_layout()

    plt.show()

    st.pyplot(fig0)

    # ploting the MA50 and Actual Close price

    st.subheader('Price vs MA50')

    ma\_50\_days = data.Close.rolling(50).mean()

    fig1 = plt.figure(figsize=(8,6))

    plt.plot(ma\_50\_days, 'r')

    plt.plot(data.Close, 'g')

    plt.legend(["MA50","Price"],loc="lower right")

    plt.show()

    st.pyplot(fig1)

    # ploting the MA50 VS MA100 and Actual Close price

    st.subheader('Price vs MA50 vs MA100')

    ma\_100\_days = data.Close.rolling(100).mean()

    fig2 = plt.figure(figsize=(8,6))

    plt.plot(ma\_50\_days, 'r')

    plt.plot(ma\_100\_days, 'b')

    plt.plot(data.Close, 'g')

    plt.legend(["MA50","MA100","Price"],loc="lower right")

    plt.show()

    st.pyplot(fig2)

    # ploting the MA100 VS MA200 and Actual Close price

    st.subheader('Price vs MA100 vs MA200')

    ma\_200\_days = data.Close.rolling(200).mean()

    fig3 = plt.figure(figsize=(8,6))

    plt.plot(ma\_100\_days, 'r')

    plt.plot(ma\_200\_days, 'b')

    plt.plot(data.Close, 'g')

    plt.legend(["MA100","MA200","Price"],loc="lower right")

    plt.show()

    st.pyplot(fig3)

    # ploting the MA50 VS MA100 VS MA200 and Actual Close price

    st.subheader('Price vs MA50 vs MA100 vs MA200')

    ma\_200\_days = data.Close.rolling(200).mean()

    fig3 = plt.figure(figsize=(8,6))

    plt.plot(ma\_50\_days,'g')

    plt.plot(ma\_100\_days, 'r')

    plt.plot(ma\_200\_days, 'b')

    plt.plot(data.Close, 'gray')

    plt.legend(["MA50","MA100","MA200","Price"],loc="lower right")

    plt.show()

    st.pyplot(fig3)

    # preadicting price on the basis of test model

    x = []

    y = []

    for i in range(100, data\_test\_scale.shape[0]):

        x.append(data\_test\_scale[i-100:i])

        y.append(data\_test\_scale[i,0])

    x,y = np.array(x), np.array(y)

    predict = model.predict(x)

    scale = 1/scaler.scale\_

    predict = predict \* scale

    y = y \* scale

    # Plot predicted and origianl price

    st.subheader('Original Price vs Predicted Price')

    fig4 = plt.figure(figsize=(8,6))

    plt.plot(predict, 'r', label='Original Price')

    plt.plot(y, 'g', label = 'Predicted Price')

    plt.xlabel('Time')

    plt.ylabel('Price')

    plt.legend(["Original Price","Predicted Price"],loc="lower right")

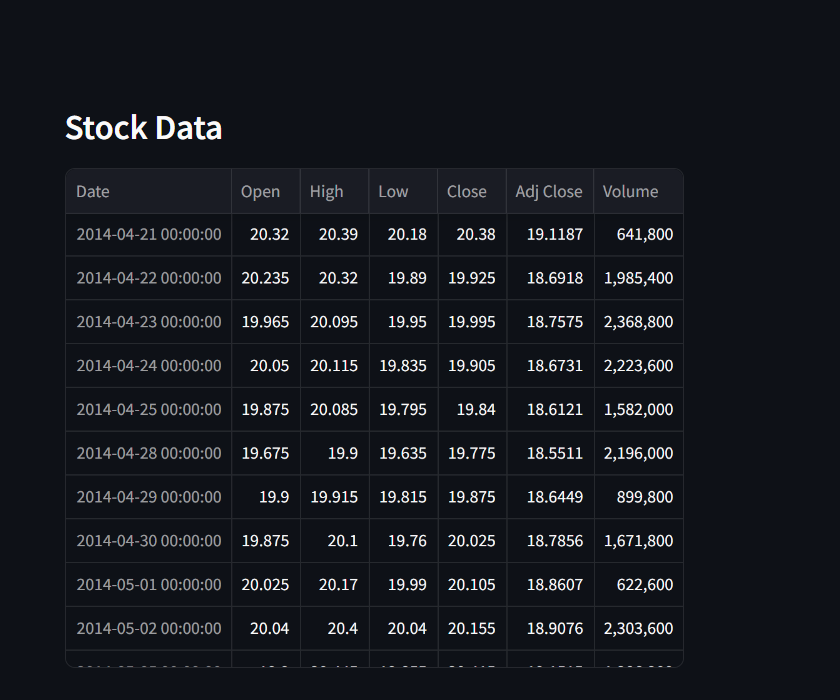
    plt.show()

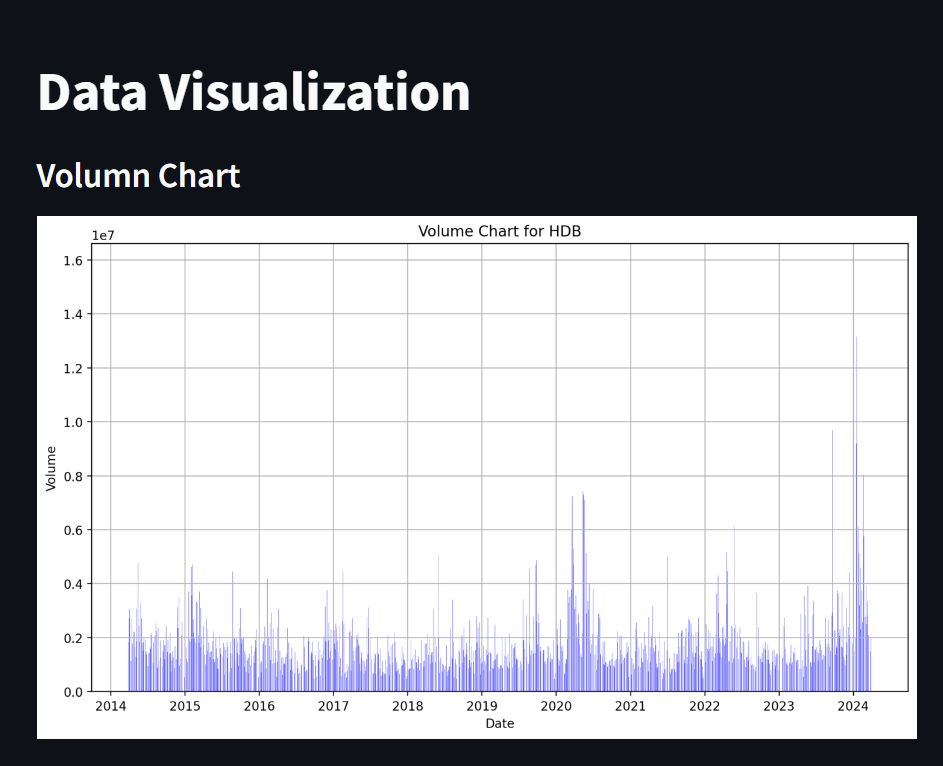
    st.pyplot(fig4)

st.button(type="primary", label="Submit !", on\_click=work)

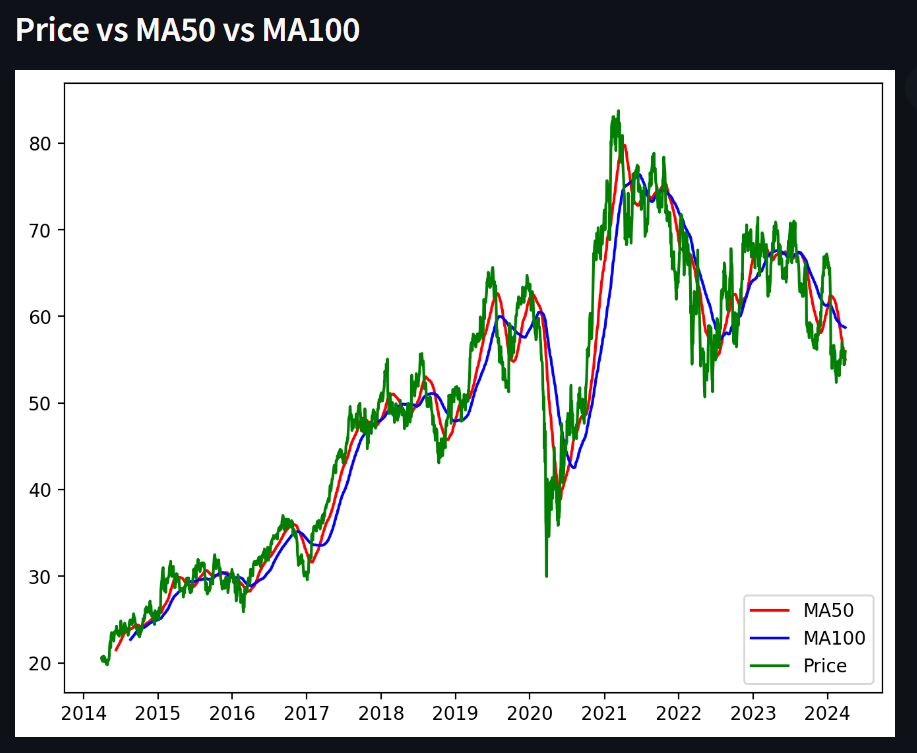
Screen Shorts

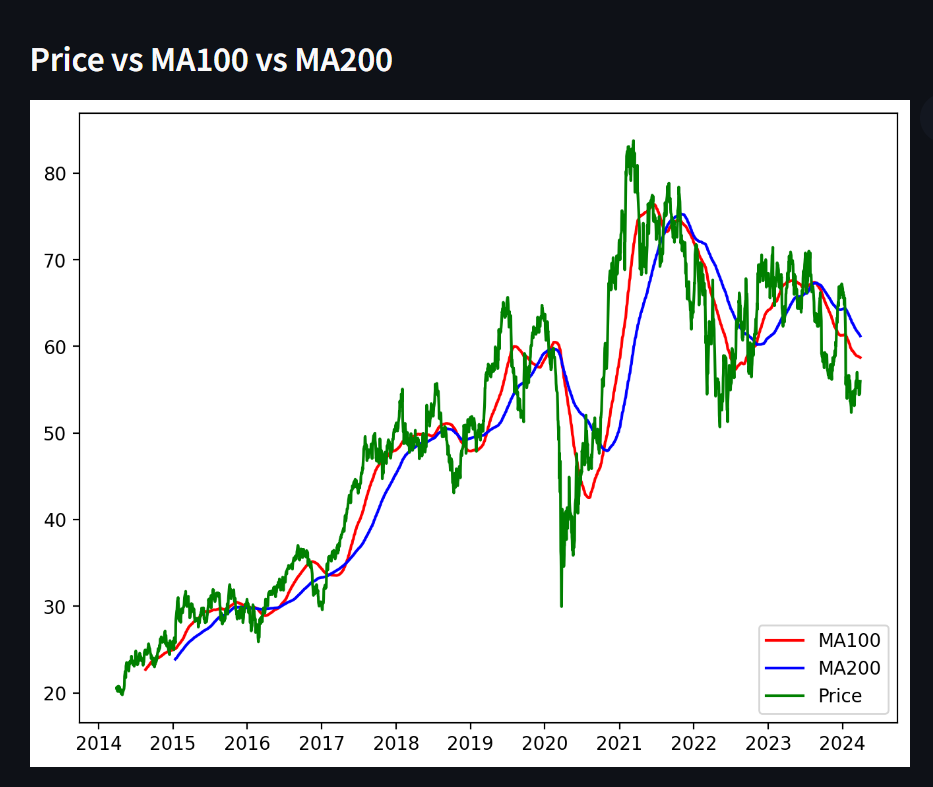


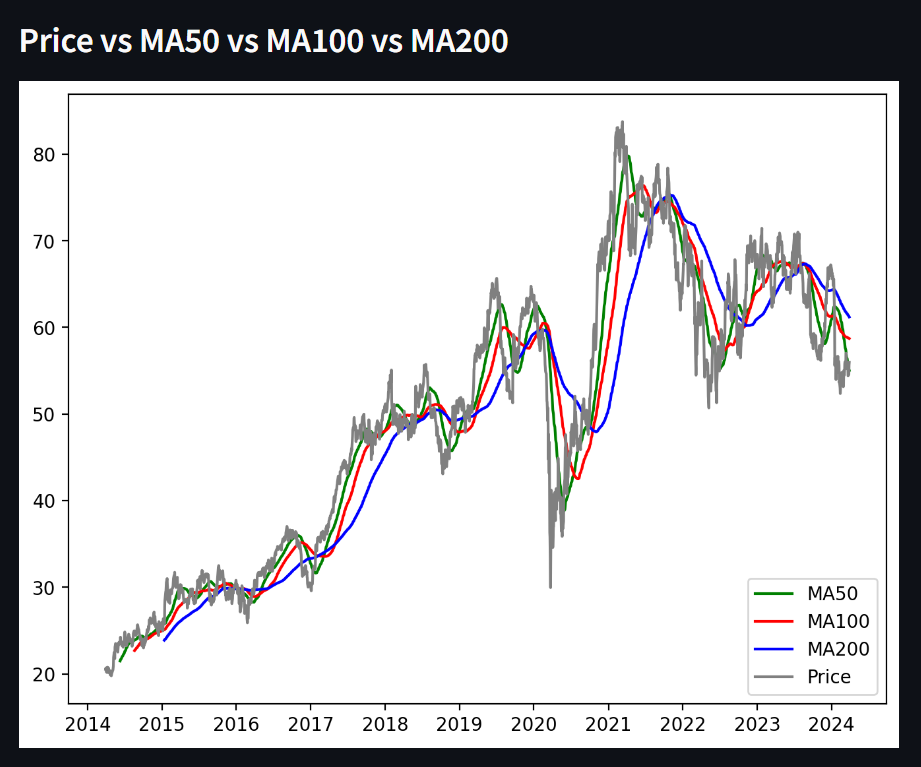


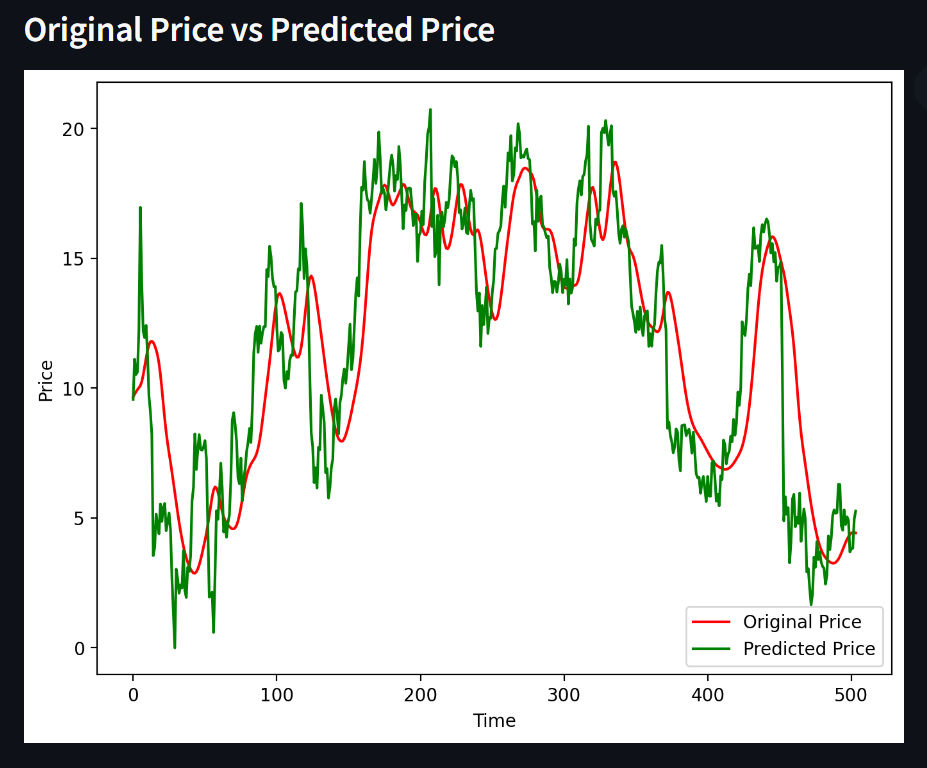












Explanation

1. **Experimental Setup:**

* The goal of the experiment is to develop a stocks prediction app using machine learning techniques, focusing on Moving Averages (MA) of different lengths (MA50, MA100, and MA200) and MinMaxScaler for normalization.
* The experiment involves collecting historical stock price data, preprocessing it, training machine learning models, evaluating their performance, and deploying the app for real-time predictions.
* The setup aims to create an accurate and user-friendly app for predicting future stock prices based on historical data.

1. **Data Collection Process:**

* Historical stock price data is collected using the Yahoo Finance API.
* The API provides access to a vast repository of historical data for various stocks and indices.
* Data is fetched for selected stocks, including features such as Open, High, Low, Close prices, and Volume.
* Moving Averages (MA) for different periods (e.g., 50, 100, 200 days) are calculated from the fetched data.

1. **Tools and Technologies:**

* Programming Language: Python is used for its rich ecosystem of libraries and tools for data analysis, machine learning, and web development.
* Data Collection: The Yahoo Finance API is utilized for fetching historical stock price data.
* Data Preprocessing: Pandas library is used for data manipulation and preprocessing tasks such as handling missing values, calculating Moving Averages, and normalization using MinMaxScaler.
* Machine Learning Models: Scikit-learn library is employed for training machine learning models such as Support Vector Machines (SVM), Random Forest, or Gradient Boosting for stock price prediction.
* Model Evaluation: Scikit-learn provides various metrics for evaluating model performance, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and accuracy.
* Deployment: Flask or Django frameworks are used for developing the web-based interface of the stocks prediction app. Visualization libraries like Matplotlib or Plotly can be employed for displaying historical data and predicted prices.

1. **Experimental Process:**

* Data is collected and preprocessed, including handling missing values, calculating Moving Averages, and normalizing the data using MinMaxScaler.
* Features are engineered, including the addition of technical indicators or sentiment analysis if deemed necessary.
* Machine learning models are trained on the preprocessed data, with hyperparameters tuned if required.
* The trained models are evaluated using cross-validation techniques to assess their performance.
* Once satisfactory performance is achieved, the app is developed with a user-friendly interface and deployed for real-time predictions.
* The deployed app is monitored for performance metrics and user feedback, with periodic updates and improvements made as needed.

Conclusion

Key Findings –

* Stock Prediction using ML and Custom Dataset
* Stock Prediction For Growth or Fall

In conclusion, the stock market analysis application is a powerful tool for traders and investors to make informed decisions in financial markets. By integrating various analytical tools and data sources, such an application can provide valuable insights into market trends, volatility, and potential investment opportunities. Here's a summary of the key aspects and benefits of a stock market analysis application:

* 1. Comprehensive Data Analysis:
     + The application offers access to a wide range of financial data, including stock prices, trading volumes, market indices, and company fundamentals.
     + Users can perform in-depth technical analysis, fundamental analysis, and sentiment analysis to evaluate the performance and prospects of individual stocks or the overall market.
  2. Technical Indicators and Charting Tools:
     + It provides a variety of technical indicators and charting tools to identify patterns, trends, and potential entry or exit points.
     + Users can customize charts, apply technical indicators, and analyze historical price movements to make informed trading decisions.

Suggestions for future work or further research. –

For the Future of this project, Live Api can be integrated for Realtime Data Analysis.