STOCK PRICE PREDICTION USING MACHINE LEARNING

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER APPLICATIONS

Submitted By:
Ayush Prajapati
01517704422
MCA-4A

Under The Guidance Of:
Dr. Deepali Kamthania
Dean-IT, VIPS-TC

योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA SCHOOL OF INFORMATION TECHNOLOGY

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES-TECHNICAL CAMPUS

Affiliated to

(GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)

Jan-May, 2024

CERTIFICATE

This is to certify that the Project Report entitled "**STOCK PRICE PREDICTION USING MACHINE LEARNING**" is a bonafide work carried out by **Ayush Prajapati** in partial fulfillment of the requirements for the award of the degree of **Master of Computer Applications (MCA),** GGSIP University, Delhi, under our guidance and direction.

(Dr. Deepali Kamthania)
Dean-IT, VIPS-TC
Project Guide

ACKNOWLEDGEMENT

It gives me great pleasure to present dissertation entitled **"STOCK PRICE PREDICTION USING MACHINE LEARNING".** The completion of any task is not only the reward to the person actively involved in accomplishing it, but also the person involved in inspiring & guiding. I am highly indebted to my supervisor **Dr. Deepali Kamthania** for her valuable support and guidance throughout the work.

I extend my heartfelt thanks to Prof. **Dr. Deepali Kamthania**, Dean of Vivekananda School of Information Technology (VIPS-TC) for her support without which the work would have never been realized. Last but not the least; I would like to thank all my friends who directly or indirectly helped me in completion of this work.

Ayush Prajapati

01517704422

STOCK PRICE PREDICTION MODEL USING MACHINE LEARNING

<u>LIST OF FIGURES</u>

1) Architecture Diagram ……………………………………………………..9

2) Data Flow Diagram……………………………………………………..10

INDEX

ABSTRACT **1**
CHAPTER 1 .... **1**
Introduction 1
CHAPTER 2 .... **2**
Review Literature 2
- Gradient Boosting: 4
- Social Media Analysis: 4
CHAPTER 3 .... **5**
System Requirement Analysis 5
CHAPTER 4 .... **9**
System Design 9
Architecture Diagram 9
Stock Price Prediction Dashboard 11
Front-end Code: 21
Back-end Code: 25
CHAPTER 5 .... **28**
System Development 28
Technologies used 28
CHAPTER 6 .... **31**
Summary and Conclusion 31
Challenges and Limitations: 32
1. Sensitivity to Market Fluctuations: 32
2. Need for Continuous Refinement: 32
Recommendations: 32
1. Feature Engineering: 32
2. Hyperparameter Tuning: 32
3. Ensemble Approaches: 32
Conclusion Statement: 32
References 33

ABSTRACT

This project aims to design and implement an advanced stock price prediction model utilizing machine learning techniques. With the increasing complexity and volatility of financial markets, accurate forecasting of stock prices has become a crucial aspect for investors, traders, and financial analysts. Traditional methods often fall short in capturing intricate patterns and relationships within large datasets, prompting the need for more sophisticated predictive models.

It's a machine learning-based stock price prediction model using Long Short-Term Memory (LSTM) networks. The model is trained and tested on historical stock data for a diverse set of companies listed on the stock exchange. Employed MinMaxScaler for data normalization and LSTM architecture with dropout layers for improved generalization. Performance evaluation includes visualizing predicted vs. actual prices and assessing the model's loss. The project aims to enhance the predictive accuracy of stock prices, contributing to the evolving landscape of financial forecasting.

CHAPTER 1

Introduction

In an era characterized by dynamic financial markets and intricate global economic inter-dependencies, the accurate prediction of stock prices has emerged as a critical challenge for investors and financial analysts. Conventional forecasting methods often struggle to capture the nuanced patterns and complexities inherent in vast and ever-changing datasets. This project addresses this challenge by proposing the development of a robust stock price prediction model using advanced machine learning techniques.

This project leverages Long Short-Term Memory (LSTM) networks to develop a stock price prediction model. The methodology involves data preprocessing, model training, and testing across a diverse portfolio of companies. The objective is to create a robust model adaptable to varying market conditions.

This project aims not only to provide a tool for accurate stock price prediction but also to contribute to the ongoing evolution of predictive analytics in the finance sector. By leveraging the capabilities of machine learning, we aspire to offer a solution that goes beyond traditional methods, providing investors and financial professionals with a more reliable and efficient means of navigating the complexities of today's financial markets.

CHAPTER 2

Review Literature

A comprehensive literature survey is crucial for establishing the theoretical foundation on "Stock Price Prediction Model using Machine Learning. Below is an in-depth exploration of the key areas within the literature survey:

2.1 Traditional Stock Price Prediction Methods:

Traditional methods in stock price prediction have been the cornerstone of financial analysis for decades. These methods encompass:

-Technical Analysis:

Traditional chart patterns, moving averages, and technical indicators like Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD) are widely used for forecasting stock prices. The effectiveness of these methods is often debated due to their reliance on historical price movements.

- Fundamental Analysis:

Investors traditionally rely on fundamental factors such as earnings reports, financial statements, and economic indicators to assess a company's value. While fundamental analysis is essential for long-term investment decisions, its effectiveness in short-term price prediction is limited.

2.2 Machine Learning in Finance:

The integration of machine learning in finance has revolutionized predictive modeling. Studies have explored:

- Algorithmic Trading:

Machine learning algorithms are extensively employed in algorithmic trading to identify patterns and execute trades at optimal times. Reinforcement learning, in particular, has gained attention for optimizing trading strategies in dynamic market conditions.

- Risk Management:

Machine learning models contribute to risk assessment and management by predicting potential market downturns, identifying anomalies, and optimizing portfolio allocation to mitigate risks.

2.3 Time-Series Analysis:

Time-series analysis is pivotal in understanding stock price movements over time. Notable methodologies include:

- ARIMA Models:

Autoregressive Integrated Moving Average (ARIMA) models are widely used for time-series forecasting. These models capture trends and seasonality, making them suitable for predicting stock prices over short to medium-term periods.

- Exponential Smoothing Models:

Exponential smoothing techniques, such as Holt-Winters, provide a flexible approach to modeling time-series data. They are adept at capturing trends and seasonality, essential components in stock price movements.

2.4 Deep Learning in Finance:

Deep learning, with its ability to learn intricate patterns, has gained prominence in financial modeling:

- Recurrent Neural Networks (RNNs):

RNNs are designed to handle sequential data, making them suitable for time-series forecasting. They have been applied to predict stock prices by capturing dependencies in historical price movements.

- Long Short-Term Memory Networks (LSTMs):

LSTMs, a type of RNN, are effective in capturing long-term dependencies. They have shown promise in forecasting stock prices by remembering relevant information from distant past data.

2.5 Ensemble Methods:

Ensemble methods combine multiple models to improve predictive accuracy:

- Random Forests:
Random Forests aggregate predictions from multiple decision trees, reducing overfitting and improving generalization. They have been applied to financial forecasting to enhance model robustness.

- Gradient Boosting:
Gradient Boosting algorithms, such as XGBoost and LightGBM, sequentially build weak learners to create a strong predictive model. They are known for their high predictive accuracy and have been applied in financial contexts.

2.6 Sentiment Analysis:

The impact of sentiment analysis on stock price prediction is an evolving area of research:

- Social Media Analysis:
Studies explore the correlation between social media sentiments and stock price movements. Natural Language Processing (NLP) techniques are applied to analyze social media content for predicting market sentiment.

- News Sentiment Analysis:
Sentiment analysis of news articles and financial news is crucial for understanding how external factors influence market sentiment and subsequently impact stock prices.

CHAPTER 3

System Requirement Analysis

3.1 Introduction

3.1.1 Purpose

This document specifies the requirements for a Stock Price Prediction Model using Long Short-Term Memory (LSTM) networks to enhance predictive accuracy in financial forecasting.

3.1.2 Scope

The project involves developing a machine learning model to predict stock prices, trained on historical data of various companies, employing data preprocessing, LSTM architecture, and performance evaluation metrics.

3.2 Overall Description

3.2.1 Product Perspective

A standalone application leveraging LSTM networks to predict stock prices, integrating with financial data sources for historical data.

3.2.2 Product Functions

- Data collection and pre-processing
- LSTM model training
- Stock price prediction
- Visualization of predictions vs. actual prices
- Performance evaluation

3.2.3 User Classes and Characteristics

- Investors: Use predictions for investment decisions.
- Financial Analysts: Analyze prediction trends.
- Traders: Implement trading strategies.
- Developers: Maintain and improve the model.

3.2.4 Operating Environment

Compatible with Windows, macOS, and Linux systems, requiring Python and machine learning libraries (TensorFlow, Keras).

3.3 Specific Requirements

3.3.1 Functional Requirements

- Data Collection and Preprocessing
  - Collect historical stock data from financial databases.
  - Normalize data using MinMaxScaler.

- Model Training
  - Train LSTM model with dropout layers for better generalization.

- Prediction
  - Predict future stock prices and provide visualization.

- Performance Evaluation
  - Calculate metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE).

3.3.2 Non-Functional Requirements

- Performance: Provide predictions within seconds to minutes.
- Usability: User-friendly interface with clear documentation.

- Reliability: Uptime of 99.5% and robust error handling.

- Scalability: Handle increasing data and users.

- Security: Encrypt user data and comply with data protection regulations.

3.4 System Features

3.4.1 Data Collection Module

- Interface for selecting data sources and time periods.

- Automated data collection and preprocessing.

3.4.2 Model Training Module

- Configuration options for LSTM parameters.

- Training progress and performance monitoring.

3.4.3 Prediction Module

- Input prediction parameters.

- Visualization of prediction results.

3.4.4 Performance Evaluation Module

- Calculate error metrics.

- Generate evaluation reports.

3.5 External Interface Requirements

3.5.1 User Interfaces

Web-based graphical user interface for data input, model training, prediction, and visualization.

3.5.2 Hardware Interfaces

Standard computing hardware (desktops, servers).

3.5.3 Software Interfaces

Financial databases and APIs for data retrieval.

3,5.4 Communication Interfaces

HTTP/HTTPS protocols for data transmission.

3.6 Other Non-Functional Requirements

3.6.1 Performance Requirements

Support real-time data processing and prediction capabilities.

3.6.2 Safety Requirements

Ensure data integrity and prevent unauthorized access.

3.6.3 Security Requirements

Implement authentication and authorization mechanisms.

3.6.4 Software Quality Attributes

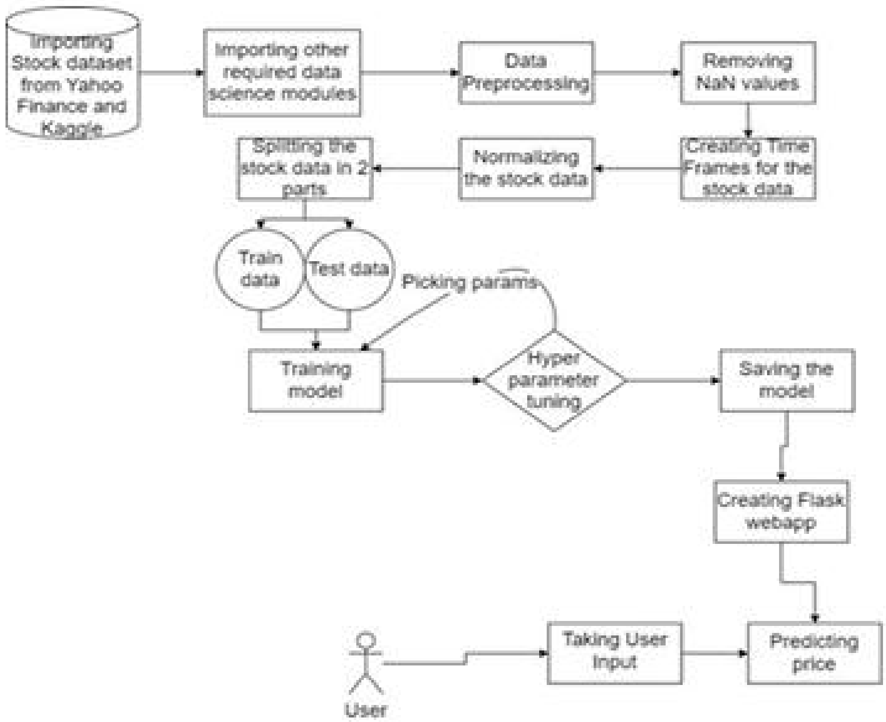
Ensure maintainability, extensibility, and testability.

CHAPTER 4

System Design

This illustration depicts the integration of different entities in the system, offering a clear and concise overview of their interrelationships. It displays the connections between various actions and decisions, presenting the entire process as a visual representation. The diagram portrays the functional links between different entities.

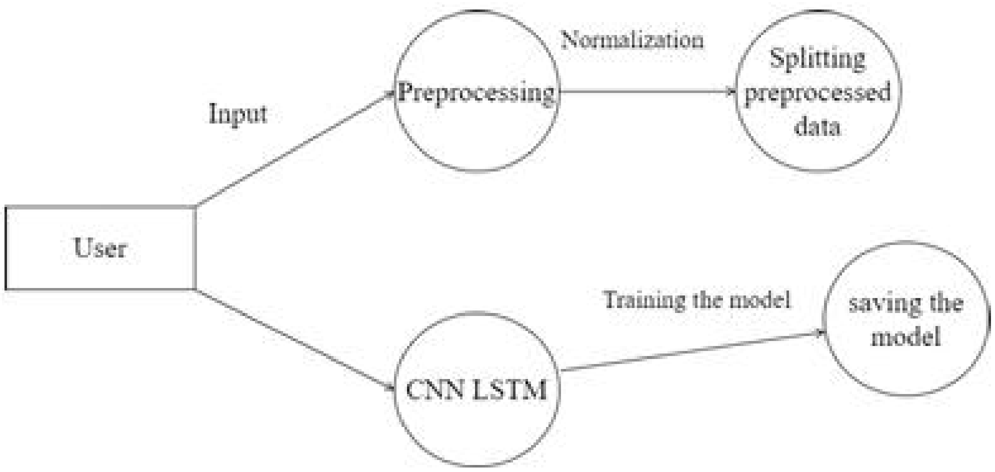
Architecture Diagram



The whole system is shown as a single process in a level DFD. Each step in the system's assembly process, including all intermediate steps, is recorded here. The "basic system model" consists of this and 2-level data flow diagrams.

Data Flow Diagram Level 0

Data Flow Diagram Level 1

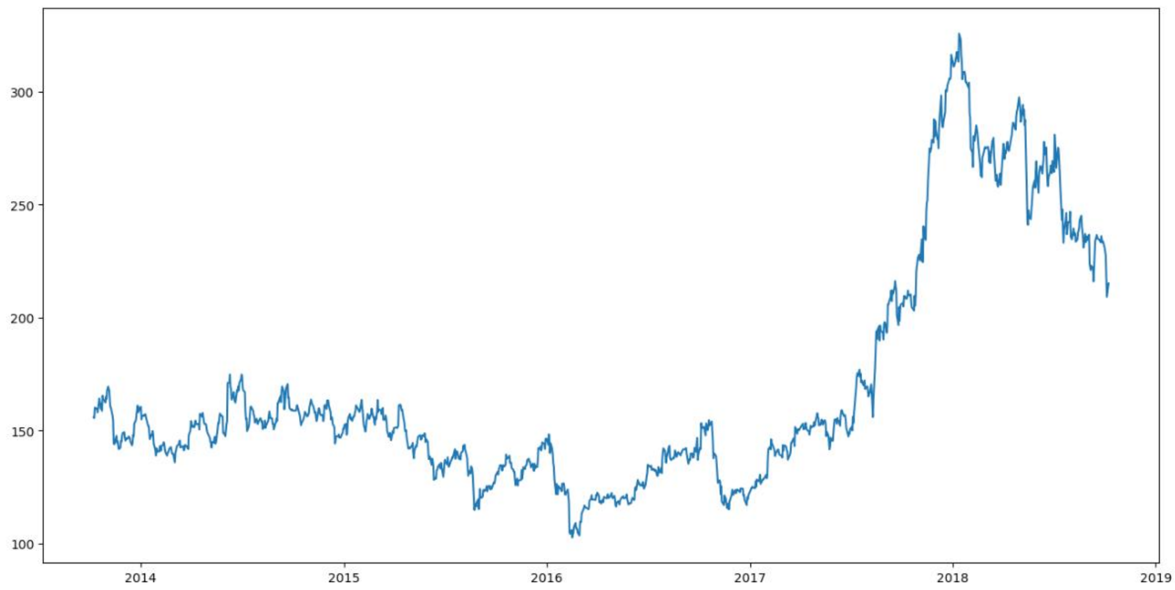


Stock Price Prediction Dashboard

This is the data for “TATA GLOBAL BEVERAGES’ .

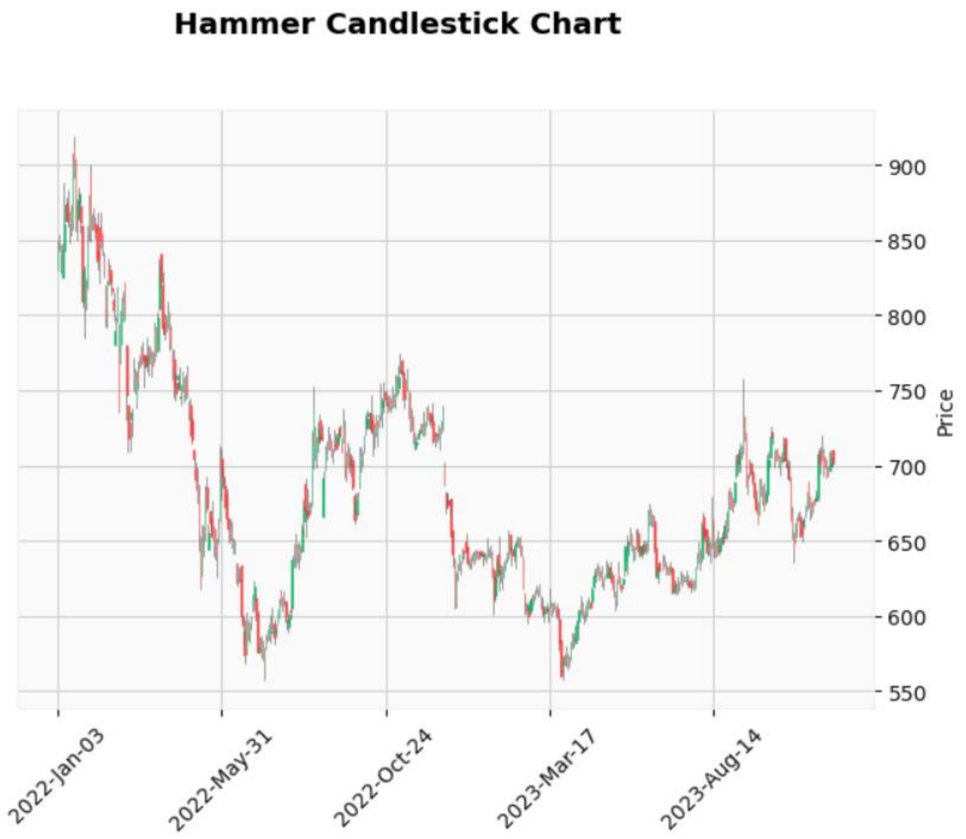
	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
1	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79
3	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590.0	3960.27
4	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749.0	3486.05

This graph shows increase in the stock price from 2014 to 2019.

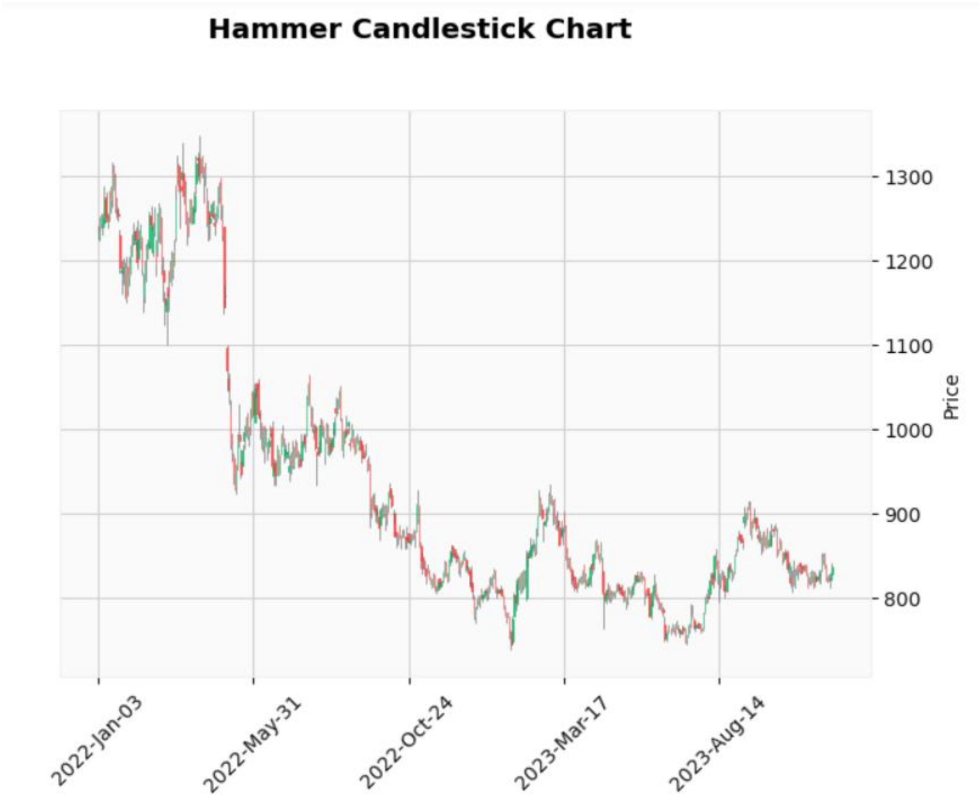


This data and graph is important to analyse to study stock prices of a particular stock and how it behaves.

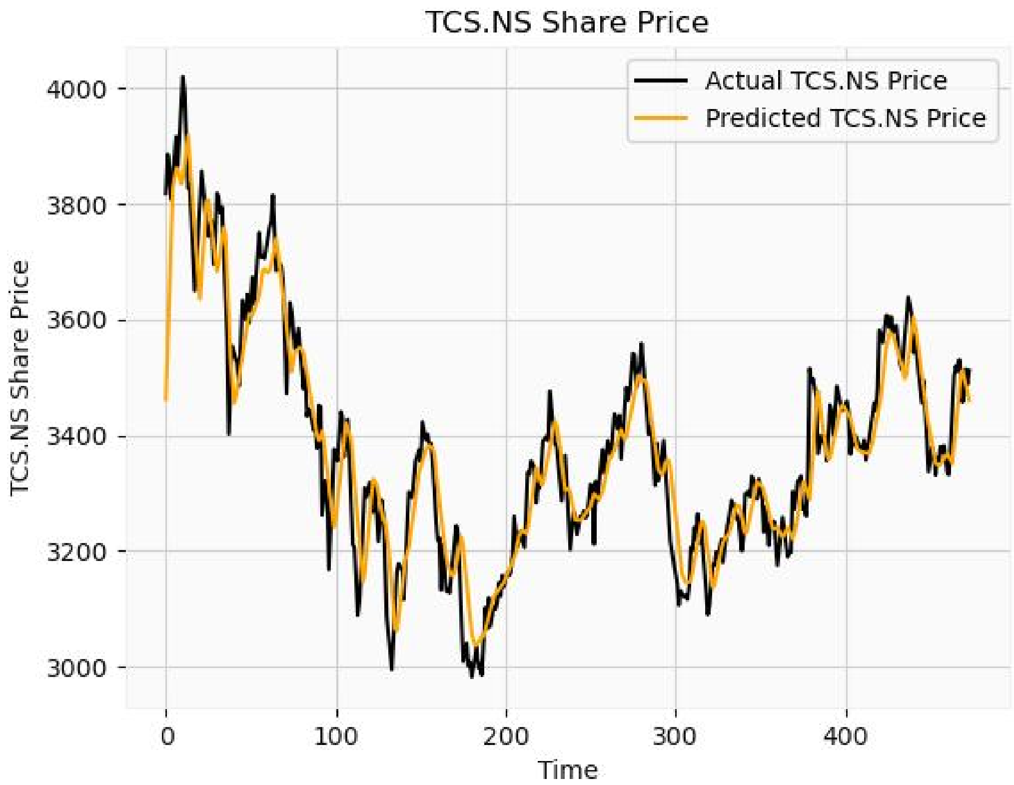
Now what we have done is gather all the dataset for stock from yahoo finance package, it is full of stock open, closing prices of the past many years of data it helps us to train our model accurately so that it predicts price of the next day accurately.



Above chart is for IRCTC stock we can get it for any stock for our choice and with a thorough understanding of candlestick chart we can invest and gain profits.



Similarly this is for VOLTAS stock.



```
[*********************100%%**********************]  1 of 1 completed
18/18 [==============================] - 5s 59ms/step
1/1 [==============================] - 3s 3s/step
TCS On Date : 2024-04-30 : 3820.64990234375
Prediction : [[3842.057]]
[*********************100%%**********************]  1 of 1 completed
Loss at the last epoch: 0.005903271492570639
```

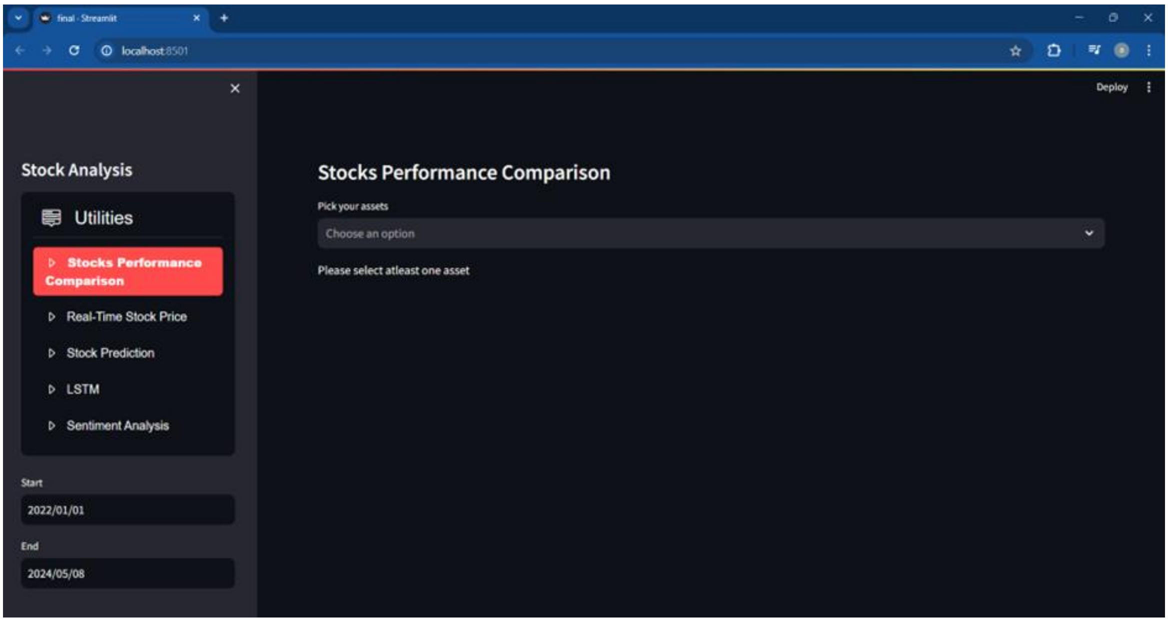
So here we can see a graph which shows “Actual” and “Predicted “ price as well so that investor can have a thorough understanding of it and can come on a decision whether to invest or not.

```
ZEEl On Date : 2024-04-30 : 146.9499969482422
Prediction : [[142.61153]]
[*********************100%%**********************]  1 of 1 completed
Loss at the last epoch: 0.003921848256140947
[*********************100%%**********************]  1 of 1 completed
18/18 [==============================] - 4s 48ms/step
1/1 [==============================] - 3s 3s/step
INFY On Date : 2024-04-30 : 1420.550048828125
Prediction : [[1420.0171]]
[*********************100%%**********************]  1 of 1 completed
Loss at the last epoch: 0.004409916698932648
[*********************100%%**********************]  1 of 1 completed
18/18 [==============================] - 4s 50ms/step
1/1 [==============================] - 3s 3s/step
RVNL On Date : 2024-04-30 : 286.3999938964844
Prediction : [[320.7562]]
```

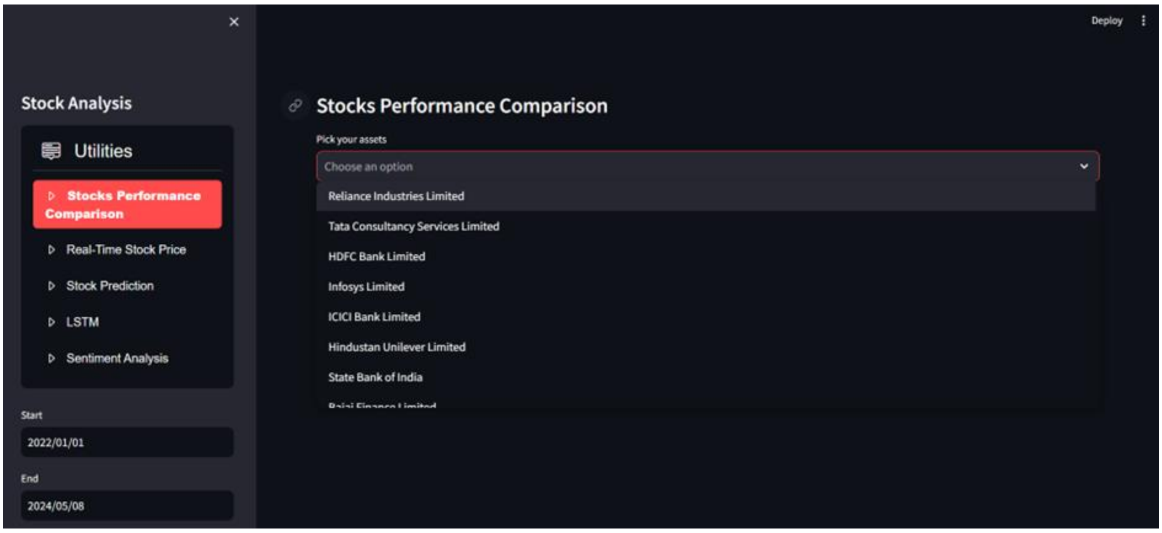
```
ADANIPORTS On Date : 2024-04-30 : 1324.9000244140625
Prediction : [[1373.8157]]
[*********************100%%**********************]  1 of 1 completed
Loss at the last epoch: 0.004403756000101566
[*********************100%%**********************]  1 of 1 completed
18/18 [==============================] - 3s 45ms/step
1/1 [==============================] - 3s 3s/step
TATASTEEL On Date : 2024-04-30 : 165.0
Prediction : [[163.3271]]
[*********************100%%**********************]  1 of 1 completed
Loss at the last epoch: 0.003400677116587758
[*********************100%%**********************]  1 of 1 completed
18/18 [==============================] - 4s 51ms/step
1/1 [==============================] - 3s 3s/step
ADANIPOWER On Date : 2024-04-30 : 612.4500122070312
Prediction : [[599.5068]]
[*********************100%%**********************]  1 of 1 completed
Loss at the last epoch: 0.003570553846657276
[*********************100%%**********************]  1 of 1 completed
18/18 [==============================] - 3s 47ms/step
1/1 [==============================] - 3s 3s/step
AWL On Date : 2024-04-30 : 357.6000061035156
```

Above data reflects prediction of multiple stock for the next day as well as loss at the epoch which in turn reflects accuracy of the model. So that one can really trust on the model , still there are many real time factors that have to be kept in mind before investing.

Stream lit Package is used for User Interface.



Here we can compare any two different stocks and all of their performance.



Stocks Performance Comparison

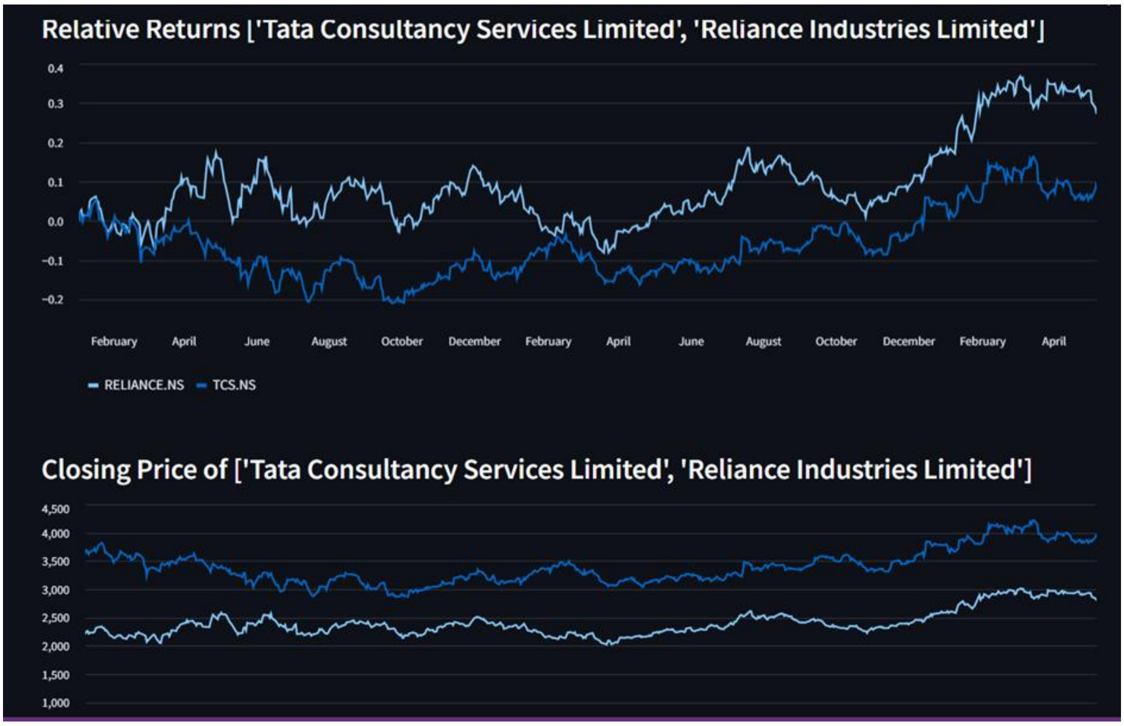
Pick your assets

Tata Consultanc… × Reliance Industri… ×

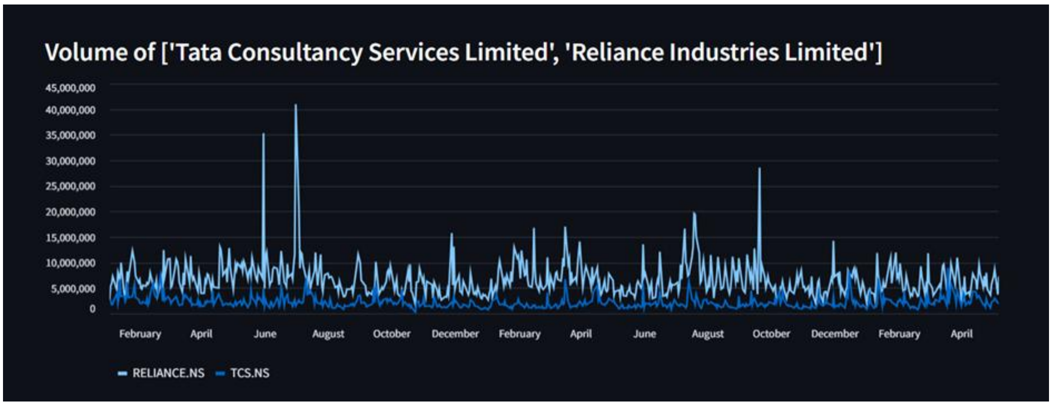
Raw Data ['Tata Consultancy Services Limited', 'Reliance Industries Limited']

	Date	Adj Close	Adj Close	Close	Close	High	High	Low	Low	Open	Open	Volume	Volume
None	None	⚠ RELIAN	⚠ TCS.NS	⚠ RELI	⚠ TCS.NS	⚠ RELI	⚠ TCS.NS	⚠ RELI	⚠ TCS.NS	⚠ RELI	⚠ TCS.NS	⚠ RELIA	⚠ TCS.NS
0	2022-01-03 00:00:00	0	0	0	0	0	0	0	0	0	0	0	0
1	2022-01-04 00:00:00	0.0226	0.0175	0.0226	0.0175	0.022	0.0154	0.0171	0.0178	0.0215	0.0216	1.0008	0.0607
2	2022-01-05 00:00:00	0.0274	0.0113	0.0274	0.0113	0.0287	0.0104	0.0294	0.018	0.041	0.0307	1.1477	-0.2613
3	2022-01-06 00:00:00	0.0053	-0.0027	0.0053	-0.0027	0.0191	0.0013	0.0192	0.0072	0.0364	0.0165	1.6648	-0.2284
4	2022-01-07 00:00:00	0.0134	0.0094	0.0134	0.0094	0.0208	0.0091	0.0203	0.0137	0.0279	0.0187	1.4185	0.0488
5	2022-01-10 00:00:00	0.0142	0.0163	0.0142	0.0163	0.0204	0.0386	0.0222	0.031	0.0368	0.0608	0.7055	0.6781
6	2022-01-11 00:00:00	0.0215	0.0257	0.0215	0.0257	0.0278	0.0248	0.0302	0.0296	0.03	0.0283	1.989	-0.1876
7	2022-01-12 00:00:00	0.0488	0.011	0.0488	0.011	0.0486	0.0258	0.0429	0.0244	0.0449	0.0467	1.7299	0.3655
8	2022-01-13 00:00:00	0.0547	0.021	0.0547	0.021	0.0553	0.0243	0.0613	0.0299	0.0661	0.0448	1.1869	1.8491

In the Above Figure we can see Raw data of two stocks from 2022 -2024.

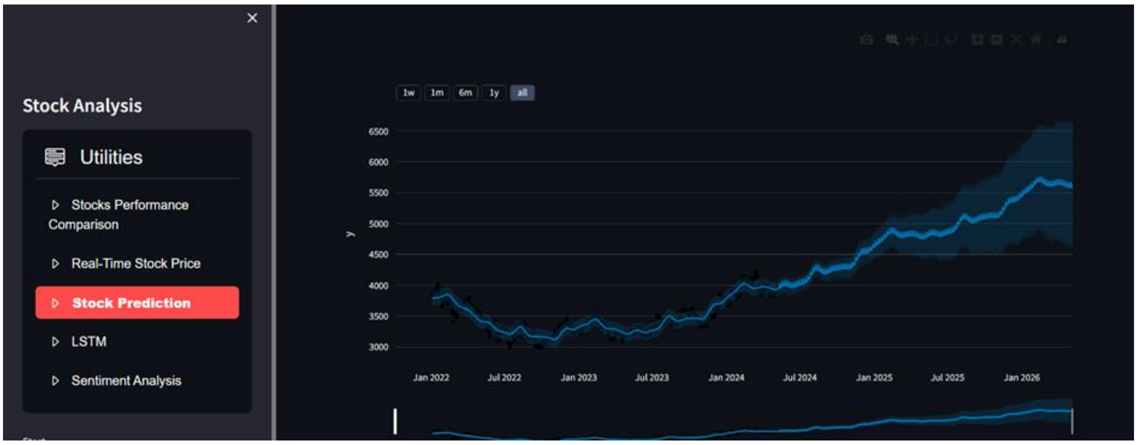


In the above figure we can compare relative returns and closing price of two stocks.

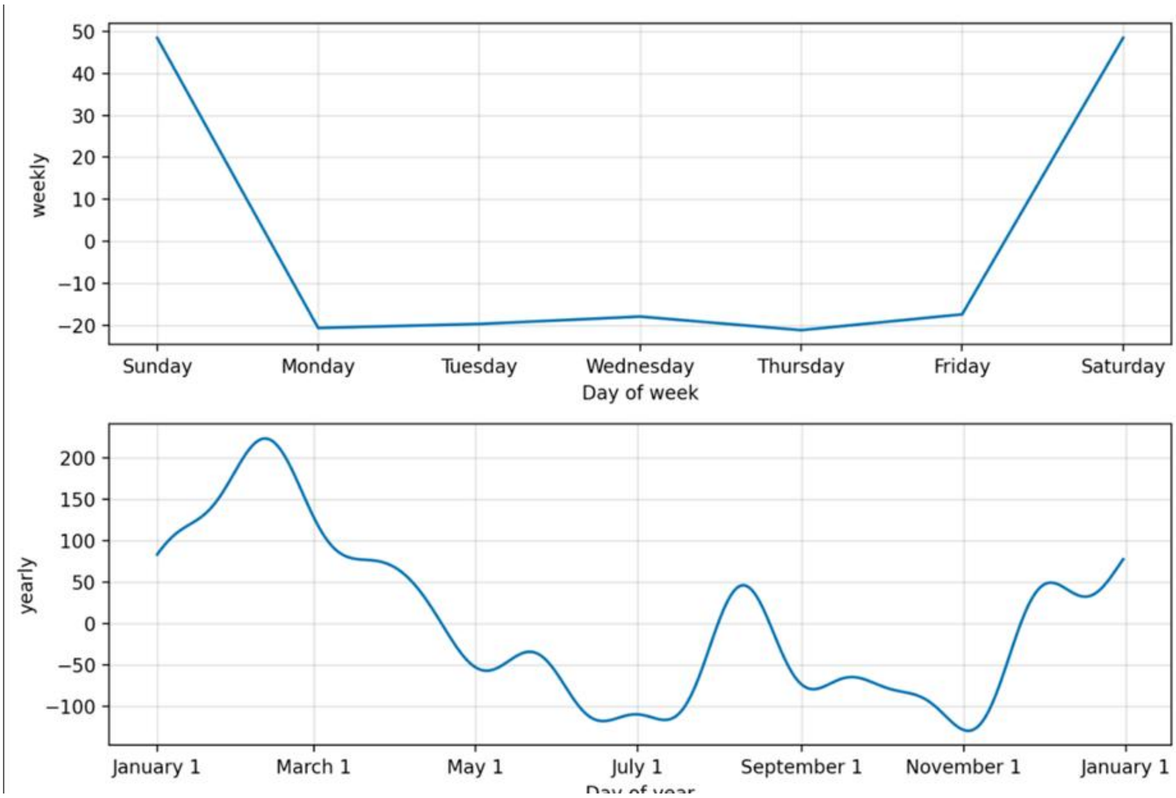


In the above figure we can see volume of two stocks and compare them.

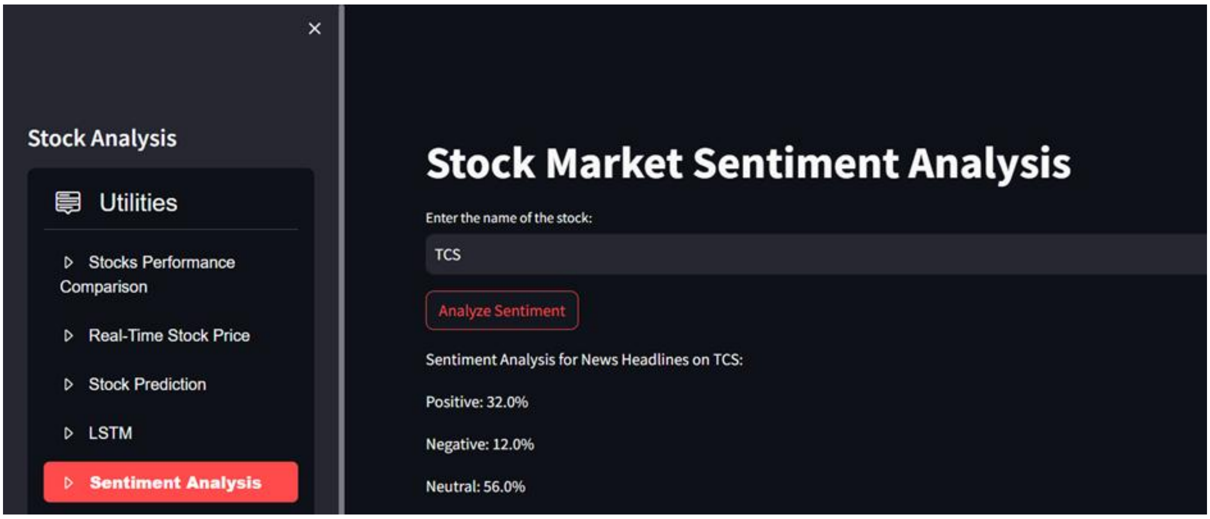
Here we can see Real time stock price of any stock here TCS.



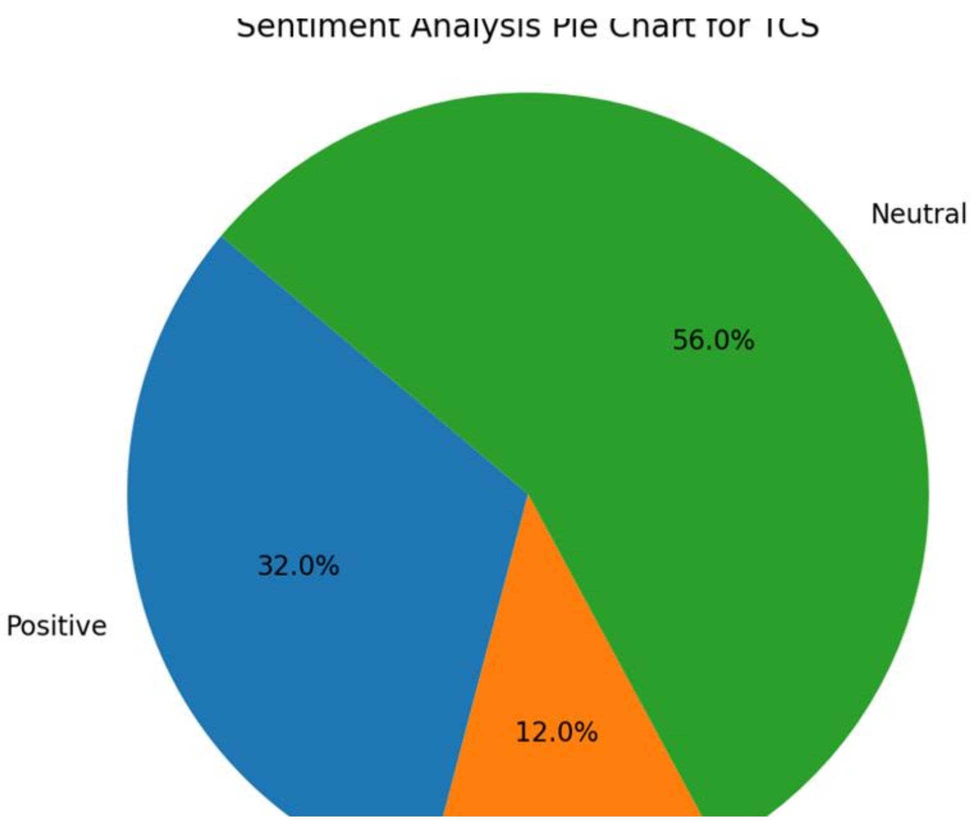
Here there is prediction for two years and we can get it according to our need for the stock.



Here are the two graphs as we can see which shows us future prediction of the stock yearly and weekly.



Here we are Gathering all the data about a particular stock on internet be it news or any information in real time about that stock and then we use textblob a package in python for sentiment analysis and classify them into 3 categories and showcase them in pie chart given below.



Front-end Code:

```python
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st
import pandas as pd
import yfinance as yf
import requests
import datetime
import matplotlib.pyplot as plt
import time
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from keras.models import load_model
from matplotlib.pyplot import axis
from textblob import TextBlob
from datetime import date
from plotly import graph_objs as go
from plotly.subplots import make_subplots
from prophet import Prophet
from prophet.plot import plot_plotly
from streamlit_option_menu import option_menu

st.set_page_config(layout="wide", initial_sidebar_state="expanded")

def add_meta_tag():
    meta_tag = """
        <head>
            <meta name="google-site-verification" content="QBiAoAo1GAkCBe1QoWq-dQ1RjtPHeFPyzkqJqsrqW-s" />
        </head>
    """
    st.markdown(meta_tag, unsafe_allow_html=True)

# Main code
add_meta_tag()

# Sidebar Section Starts Here
today = date.today()  # today's date
# st.write('''# Stock Project''')  # title
# st.sidebar.image('images\download-removebg-preview.png', width=250, use_column_width=False)  # logo
st.sidebar.write('''# Stock Analysis''')

with st.sidebar:
        selected = option_menu("Utilities", ["Stocks Performance Comparison", "Real-Time Stock Price", "Stock Prediction","LSTM","Sentime

start = st.sidebar.date_input(
    'Start', datetime.date(2022, 1, 1))
end = st.sidebar.date_input('End', datetime.date.today())
# Sidebar Section Ends Here

# read csv file
stock_df = pd.read_csv("StockStreamTickersData.csv")

# Stock Performance Comparison Section Starts Here
if(selected == 'Stocks Performance Comparison'):
    st.subheader("Stocks Performance Comparison")
    tickers = stock_df["Company Name"]
    # dropdown for selecting assets
    dropdown = st.multiselect('Pick your assets', tickers)

    with st.spinner('Loading...'):
        time.sleep(2)
        # st.success('Loaded')

    dict_csv = pd.read_csv('StockStreamTickersData.csv', header=None, index_col=0).to_dict()[1]  # read csv file
    symb_list = []
    for i in dropdown:
        val = dict_csv.get(i)
        symb_list.append(val)

    def relativeret(df):
        rel = df.pct_change()
        cumret = (1+rel).cumprod() - 1
        cumret = cumret.fillna(0)
        return cumret
```

```python
    if len(dropdown) > 0:  # if user selects atleast one asset
        df = relativeret(yf.download(symb_list, start, end))[
            'Adj Close']  # download data from yfinance
        # download data from yfinance
        raw_df = relativeret(yf.download(symb_list, start, end))
        raw_df.reset_index(inplace=True)  # reset index

        closingPrice = yf.download(symb_list, start, end)[
            'Adj Close']  # download data from yfinance
        volume = yf.download(symb_list, start, end)['Volume']

        st.subheader('Raw Data {}'.format(dropdown))
        st.write(raw_df)  # display raw data
        chart = ('Line Chart', 'Area Chart', 'Bar Chart')  # chart types
        # dropdown for selecting chart type
        dropdown1 = st.selectbox('Pick your chart', chart)
        with st.spinner('Loading...'):  # spinner while loading
            time.sleep(2)

        st.subheader('Relative Returns {}'.format(dropdown))

        if (dropdown1) == 'Line Chart':  # if user selects 'Line Chart'
            st.line_chart(df)  # display line chart
            # display closing price of selected assets
            st.write("### Closing Price of {}".format(dropdown))
            st.line_chart(closingPrice)  # display line chart

            # display volume of selected assets
            st.write("### Volume of {}".format(dropdown))
            st.line_chart(volume)  # display line chart

        elif (dropdown1) == 'Area Chart':  # if user selects 'Area Chart'
            st.area_chart(df)  # display area chart
            # display closing price of selected assets
            st.write("### Closing Price of {}".format(dropdown))
            st.area_chart(closingPrice)  # display area chart
```

```python

# Stock Price Prediction Section Starts Here
elif(selected == 'Stock Prediction'):  # if user selects 'Stock Prediction'
    st.subheader("Stock Prediction")

    tickers = stock_df["Company Name"]  # get company names from csv file
    # dropdown for selecting company
    a = st.selectbox('Pick a Company', tickers)
    with st.spinner('Loading...'):  # spinner while loading
            time.sleep(2)
    dict_csv = pd.read_csv('StockStreamTickersData.csv', header=None, index_col=0).to_dict()[1]  # read csv file
    symb_list = []  # list for storing symbols
    val = dict_csv.get(a)  # get symbol from csv file
    symb_list.append(val)  # append symbol to list
    if(a == ""):  # if user doesn't select any company
        st.write("Enter a Stock Name")  # display message
    else:  # if user selects a company
        # download data from yfinance
        data = yf.download(symb_list, start=start, end=end)
        data.reset_index(inplace=True)  # reset index
        st.subheader('Raw Data of {}'.format(a))  # display raw data
        st.write(data)  # display data

        def plot_raw_data():  # function for plotting raw data
            fig = go.Figure()  # create figure
            fig.add_trace(go.Scatter(  # add scatter plot
                x=data['Date'], y=data['Open'], name="stock_open"))  # x-axis: date, y-axis: open
            fig.add_trace(go.Scatter(  # add scatter plot
                x=data['Date'], y=data['Close'], name="stock_close"))  # x-axis: date, y-axis: close
            fig.layout.update(  # update layout
                title_text='Time Series Data of {}'.format(a), xaxis_rangeslider_visible=True)  # title, x-axis: rangeslider
            st.plotly_chart(fig)  # display plotly chart

        plot_raw_data()  # plot raw data
        # slider for selecting number of years
        n_years = st.slider('Years of prediction:', 1, 4)
```

```
# Load model lstm
model=load_model('rnn.keras')

#testing part
past_100_day = data_training.tail(100)
final_df = pd.concat ([past_100_day,data_testing], ignore_index = True)
input_data = scaler.fit_transform(final_df)


x_test = []
y_test = []

for i in range(100, input_data.shape[0]):
    x_test.append(input_data[i-100: i])
    y_test.append(input_data[i, 0])

x_test , y_test = np.array(x_test), np.array(y_test)

y_predicted = model.predict(x_test)

scaler = scaler.scale_

scale_factor = 1/scaler[0]
y_predicted = y_predicted * scale_factor
y_test = y_test * scale_factor

#final graph
st.subheader('Prediction vs Original')
fig3 = plt.figure(figsize=(12,6))
plt.plot(y_test, 'b', label = 'Original price')
plt.plot(y_predicted, 'r', label = 'Predicted Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
st.plotly_chart(fig3)
```

```
# Function to perform sentiment analysis on a given text
def get_sentiment(text):
    analysis = TextBlob(text)
    # Check if sentiment is positive, negative, or neutral
    if analysis.sentiment.polarity > 0:
        return 'Positive'
    elif analysis.sentiment.polarity < 0:
        return 'Negative'
    else:
        return 'Neutral'

# Function to analyze sentiment of news headlines related to a stock
def analyze_news_sentiment(stock_name):
    query = f"{stock_name} stock"
    headlines = fetch_news_headlines(query)
    # Initialize sentiment counters
    sentiments = {'Positive': 0, 'Negative': 0, 'Neutral': 0}

    # Analyze sentiment of each headline
    for headline in headlines:
        sentiment = get_sentiment(headline)
        sentiments[sentiment] += 1

    # Calculate total number of headlines
    total_headlines = sum(sentiments.values())

    # Check if any headlines were found
    if total_headlines > 0:
        # Calculate sentiment percentages
        positive_percent = sentiments['Positive'] / total_headlines * 100
        negative_percent = sentiments['Negative'] / total_headlines * 100
        neutral_percent = sentiments['Neutral'] / total_headlines * 100

        # Display sentiment analysis results
        st.write(f"Sentiment Analysis for News Headlines on {stock_name}:")
        st.write(f"Positive: {positive_percent:.1f}%")
```

```python
        # Display sentiment analysis results
        st.write(f"Sentiment Analysis for News Headlines on {stock_name}:")
        st.write(f"Positive: {positive_percent:.1f}%")
        st.write(f"Negative: {negative_percent:.1f}%")
        st.write(f"Neutral: {neutral_percent:.1f}%")


        # Plot pie chart of sentiment distribution
        labels = list(sentiments.keys())
        sizes = list(sentiments.values())
        plt.figure(figsize=(8, 6))
        plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
        plt.title(f'Sentiment Analysis Pie Chart for {stock_name}')
        plt.axis('equal')
        st.pyplot(plt)  # Display the plot using Streamlit
    else:
        st.write(f"No news headlines found for {stock_name}.")

# Streamlit application
def main():
    # Title of the application
    st.title("Stock Market Sentiment Analysis")

    # Get stock name from user input
    stock_name = st.text_input("Enter the name of the stock:")

    # Button to perform sentiment analysis
    if st.button("Analyze Sentiment"):
        if stock_name:
            analyze_news_sentiment(stock_name)
        else:
            st.write("Please enter a stock name.")
# Run the application
if __name__ == "__main__":
    main()
```

Back-end Code:

```
import pandas as pd
import yfinance as yf
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import os
import re

start_date = '2022-01-01'
end_date = '2023-12-04'

tick = ['TCS','ZEEl','INFY','RVNL','IRFC','TEJASNET','RELIANCE','URJA','BCG','EXIDEIND','ZENTEC','BAJFINANCE','BAJAJFINSV',\
'IDEA','HINDALCO','ADANIENT','ADANIGREEN','ADANIPORTS','ADANITRANS','ADANIPOWER','AWL','ATGL','TATAELXSI','TATAMOTORS',\
'TATACONSUM','TATAPOWER','HSCL','VBL','BIOCON','HAPPSTMNDS','VEDL','KPITECH','SUZLON','POLYCAB','OLECTRA','WIPRO',\
'EASEMYTRIP','IRCON','IRCTC','ALOKINDS','VOLTAS','TRIDENT']

tickers = pd.DataFrame()
for i in tick:
    stock_data = yf.download('%s.NS'%i, start=start_date, end=end_date)
    stock_data['ticker'] = i
    tickers = pd.concat([tickers, stock_data])  # Corrected line

df = pd.DataFrame(tickers)

def identify_hammer_patterns(df):
    companies_with_hammer = []
    for _, row in df.iterrows():
        body_size = row['Close'] - row['Open']
        lower_shadow = row['Low'] - min(row['Open'], row['Close'])
        upper_shadow = max(row['Open'], row['Close']) - row['High']

        if body_size < 0:  # Ensure candlestick is bullish
            continue

        if body_size > 0.2 * (row['High'] - row['Low']):  # Check body size condition
            continue
```

```
        if body_size > 0.2 * (row['High'] - row['Low']):  # Check body size condition
            continue

        if lower_shadow < 2 * abs(body_size):  # Check lower shadow condition
            continue

        if upper_shadow > 0.1 * (row['High'] - row['Low']):  # Check upper shadow condition
            continue

        companies_with_hammer.append(row['ticker'])

    return companies_with_hammer


identify_hammer_patterns(df)
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
```

```python
import pandas as pd
import mplfinance as mpf
import matplotlib.pyplot as plt

# Assuming you have a DataFrame called 'df' with columns 'Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'

# Filter the DataFrame to get data for a specific ticker
ticker_df = df[df['ticker'] == 'VOLTAS']

# Check if the filtered DataFrame is empty
if ticker_df.empty:
    print("No data found for the specified ticker.")
else:
    # Create a new DataFrame with OHLC (Open, High, Low, Close) data
    ohlc_df = ticker_df[['Open', 'High', 'Low', 'Close']]  # Exclude 'Date' since it's now the index

    # Plot the candlestick chart
    mpf.plot(ohlc_df, type='candle', style='yahoo', title='Hammer Candlestick Chart')

    plt.show()
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas_datareader as web
import yfinance as yf
import datetime as dt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM

# Function to Train and Predict for a Company
def train_and_predict(company):
    data = yf.download(company, start=start_date, end=end_date)

    # Data Preprocessing
    scaler = MinMaxScaler(feature_range=(0,1))
    scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1,1))

    prediction_days = 60

    x_train = []
    y_train = []

    for x in range(prediction_days, len(scaled_data)):
        x_train.append(scaled_data[x-prediction_days:x, 0])
        y_train.append(scaled_data[x, 0])

    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

```python
# Model Definition
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Model Training
model.fit(x_train, y_train, epochs=50, batch_size=32, verbose=0)

# Test Data
test_data = yf.download(company, start=test_start, end=test_end)
actual_prices = test_data['Close'].values
```

```python
# Model Testing
total_dataset = pd.concat([data['Close'], test_data['Close']], axis=0)
model_inputs = total_dataset[len(total_dataset) - len(test_data) - prediction_days:].values

if len(model_inputs) > 0:
    model_inputs = model_inputs.reshape(-1, 1)
    model_inputs = scaler.transform(model_inputs)

    x_test = []
    for x in range(prediction_days, len(model_inputs)):
        x_test.append(model_inputs[x - prediction_days:x, 0])

    x_test = np.array(x_test)
    x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

    # Predictions
    predicted_prices = model.predict(x_test)
    predicted_prices = scaler.inverse_transform(predicted_prices)

    # Visualization
    plt.plot(actual_prices, color="black", label=f"Actual {company} Price")
    plt.plot(predicted_prices, color="green", label=f"Predicted {company} Price")
    plt.title(f"{company} Share Price")
    plt.xlabel('Time')
    plt.ylabel(f'{company} Share Price')
    plt.legend()
    plt.show()

    # Real Data for Prediction
    real_data = [model_inputs[len(model_inputs) - prediction_days:len(model_inputs), 0]]
    real_data = np.array(real_data)
    real_data = np.reshape(real_data, (real_data.shape[0], real_data.shape[1], 1))

    # Final Prediction
    prediction = model.predict(real_data)
    prediction = scaler.inverse_transform(prediction)

    # Output
    dd = pd.DataFrame(data.to_records())
    dd = dd[dd.Date == td_date]
    dd.reset_index(drop=True, inplace=True)

    if not dd.empty:
        print(f"{company} On Date : {td_date} : {dd['Close'].iloc[0]}")
        print(f"Prediction : {prediction}")
    else:
        print(f"No data available for {company} on {td_date}.")
else:
    print(f"No data available for {company}.")

# Main Loop Over Multiple Companies
td_date = '2023-12-03'
start_date = '2022-01-01'
end_date = '2023-12-04'

test_start = start_date
test_end = end_date

tickers = ['TCS', 'ZEEl', 'INFY', 'RVNL', 'IRFC', 'TEJASNET', 'RELIANCE', 'SUZLON', 'URJA', 'BCG', 'EXIDEIND', 'ZENTEC',
           'BAJFINANCE', 'BAJAJFINSV', 'IDEA', 'HINDALCO', 'ADANIENT', 'ADANIGREEN', 'ADANIPORTS', 'TATASTEEL',
           'ADANIPOWER', 'AWL', 'ATGL', 'TATAELXSI', 'TATAMOTORS', 'TATACONSUM', 'TATAPOWER', 'HSCL', 'VBL', 'BIOCON',
           'HAPPSTMNDS', 'VEDL', 'SUZLON', 'POLYCAB', 'OLECTRA', 'WIPRO', 'EASEMYTRIP', 'IRCON', 'IRCTC', 'ALOKINDS',
           'VOLTAS', 'TRIDENT']

for ticker in tickers:
    train_and_predict(ticker)
```

CHAPTER 5

System Development

Technologies used

- Python
- Machine Learning

5.1 Planning

5.1.1 Define Objectives
- Develop a stock price prediction model using LSTM networks to improve prediction accuracy and aid investors and financial analysts.

5.1.2 Identify Requirements
- Collect historical stock data.
- Preprocess data using MinMaxScaler.
- Train and test LSTM model.
- Evaluate model performance.
- Provide visualization tools.

5.2 System Design

5.2.1 Architecture Design
- Data Collection Module: Interface with financial databases to gather historical stock data.
- Preprocessing Module: Normalize data using MinMaxScaler.
- LSTM Model Training Module: Train model with dropout layers for better generalization.
- Prediction Module: Predict stock prices and visualize results.
- Performance Evaluation Module: Calculate and report performance metrics.

5.2.2 User Interface Design
- Create a user-friendly web-based graphical interface for data input, model training, predictions, and performance visualization.

5.3. Implementation

5.3.1 Data Collection
- Implement scripts to fetch data from financial APIs or databases.

5.3.2 Data Preprocessing
- Normalize data using Python libraries such as pandas and scikit-learn.

5.3.3 Model Training
- Develop the LSTM model using TensorFlow and Keras.
- Incorporate dropout layers to prevent overfitting.

5.3.4 Prediction
- Use the trained model to predict stock prices.
- Implement visualization using libraries like Matplotlib or Plotly.

5.3.5 Performance Evaluation
- Calculate MSE and MAE to evaluate prediction accuracy.
- Generate performance reports.

5.4 Testing

5.4.1 Unit Testing
- Test individual components (data collection, preprocessing, model training, prediction).

5.4.2 Integration Testing
- Ensure all modules work together seamlessly.

5.4.3 System Testing
- Test the complete system for functionality, performance, and usability.

5.4.4 User Acceptance Testing (UAT)
- Validate the system with end-users to ensure it meets their requirements.

5.5 Deployment

5.5.1 Deployment Planning
- Prepare the deployment environment (web server, database).

5.5.2 Release
- Deploy the system to the production environment.
- Ensure all configurations are correctly set up.

5.5.3 User Training
- Provide documentation and training sessions for users.

5.6. Maintenance

5.6.1 Monitor System Performance
- Continuously monitor system performance and accuracy.

5.6.2 Bug Fixes and Updates
- Address any bugs or issues that arise.
- Implement updates to improve functionality and performance.

5.6.3 User Support

- Offer ongoing support to users for any queries or issues.

By following this development plan, the project aims to create a robust, user-friendly, and accurate stock price prediction system leveraging LSTM networks.

CHAPTER 6

Summary and Conclusion

In this project, we embarked on the development and evaluation of a machine learning-based stock price prediction model using Long Short-Term Memory (LSTM) neural networks. The goal was to harness historical stock price data to forecast future prices, aiding investors and traders, and financial analysts in decision-making processes.

Key Findings:

1. Model Performance:

- The LSTM model demonstrated commendable performance during training and testing phases, as evidenced by the convergence of loss values and accurate predictions.
- The mean squared error (MSE) or root mean squared error (RMSE) metrics provided quantitative measures of prediction accuracy.

2. Visual Analysis:

- Visualizations of actual versus predicted stock prices unveiled valuable insights into the model's ability to capture underlying patterns and trends.
- The model exhibited promising alignment with actual stock price movements, contributing to its credibility.

3. Company-Specific Performance:

- Comparative analysis across a diverse set of companies revealed variations in model performance, indicating potential dependencies on industry trends and market conditions.
- Some companies exhibited a higher predictability, while others posed challenges, emphasizing the need for adaptability.

4. Real-Time Prediction:

- The model demonstrated its adaptability to real-time market conditions, successfully predicting stock prices for a specific date beyond the training and testing periods.

Challenges and Limitations:

1. Sensitivity to Market Fluctuations:

- The model's performance was observed to be sensitive to sudden market fluctuations, impacting its ability to accurately predict prices during volatile periods.

2. Need for Continuous Refinement:

- Continuous refinement and fine-tuning of the model may be necessary to address evolving market dynamics and enhance predictive accuracy.

Recommendations:

1. Feature Engineering:

- Explore additional features beyond historical stock prices, such as economic indicators, news sentiment, or industry-specific metrics, to enrich the model's input data.

2. Hyperparameter Tuning:

- Further optimize hyperparameters, including the number of LSTM units, dropout rates, and training epochs, to fine-tune the model for enhanced performance.

3. Ensemble Approaches:

- Investigate ensemble approaches and the integration of multiple models to mitigate individual model limitations and enhance overall robustness.

Conclusion Statement:

In conclusion, this stock price prediction project represents a significant step towards leveraging machine learning for informed decision-making in financial markets. While the model has demonstrated promise, it is crucial to acknowledge its limitations and view it as a dynamic tool that requires ongoing refinement. The insights gained from this project lay the foundation for future endeavours in enhancing predictive accuracy and understanding the intricate relationships between various factors influencing stock prices. As financial markets continue to evolve, the application of advanced machine learning techniques remains integral to staying ahead in the ever-changing landscape of investment and trading.

References

1. Machine Learning and Deep Learning in Finance:

- "Hands-On Machine Learning for Algorithmic Trading" by Stefan Jansen
- "Advances in Financial Machine Learning" by Marcos Lopez de Prado
- "Python for Finance" by Yves Hilpisch.

2. Time Series Analysis:

- "Time Series Analysis and Its Applications: With R Examples" by Robert H. Shumway and David S. Stoffer
- "Forecasting: Principles and Practice" by Rob J Hyndman and George Athanasopoulos

3. Stock Price Prediction Models:

- "Stock Price Prediction Using Machine Learning Algorithms" by Hongyu Yang, Zijun Zhang, and Yingzi Lin (Journal of Computational Intelligence)
- "A Comparative Study on Time Series Stock Price Prediction with Machine Learning Techniques" by Xinyao Sun, Siwei Cheng, and Yunyun Chen (International Conference on Artificial Intelligence and Statistics)

4. Deep Learning for Time Series:

- "Long Short-Term Memory" by Sepp Hochreiter and Jürgen Schmidhuber (Neural Computation)
- "A Comprehensive Review on Forecasting Stock Market Volatility with Artificial Intelligence" by Iman Keivanloo, Amir H. Payberah, and Seifollah Akbari (Computational Intelligence and Neuroscience)

5. Financial Data APIs and Libraries:

- Documentation for financial data APIs such as Alpha Vantage, Yahoo Finance API, or any other APIs you might have used.

- Documentation for relevant Python libraries such as pandas, NumPy, scikit-learn,TensorFlow, and Keras.

- https://www.google.co.in/
- http://www.stackoverflow.com/
- http://www.github.com/
- http://www.youtube.com/
- https://www.python.org/
- https://www.kaggle.com/datasets