Stock tracker and trend prediction system

Vedant Hiwarde[1], Sahil Nikam[2], Prajwal Alekar[3], Prajwal Nimbalkar[4], Omkar Shinde[5], Vedant Kumbhar[6], Girija Teli[7], Jayesh Salunke[8]
[1,2,3,4,5,6]*Department of Robotics & Automation Engineering, Zeal College of Engineering & Research, Narhe, Pune-41, Savitribai Phule Pune University*

Abstract- The stock exchange of India is part of the stock exchanges investing in shares across the world. It consists of, National Stock Exchange and Bombay Stock Exchange. In the context of National Stock Exchange, the total counts of firms 2,266, while total registered firms in Bombay Stock Exchange are 5,309. At least, the Indian stock market situation is more complex and changeable, and therefore, a stock prediction system is required. The Bombay Stock Exchange (BSE) and the National Stock Exchange (NSE) are two well-known measures that prove the importance of the Indian stock market. The past experience indicates that, the Indian stock market does possess a sort of diversity which challenges investors. However, I believe that benchmark indices (BSE Sensex and Nifty 50) are not only the measure of market popularity but also economic viability. It has assumed a stronger form with liberalization in India as the stock market contributes to wealth creation, resource mobilization for businesses and overall economic progress. Thus, the necessity of presentations like stock prediction systems, in order to restore investor confidence, improve the corporate image of the capital markets and continue the growth in one of the world's fastest growing economies for shares – India. This prediction system involves data for the stock and approximately it covers the period from the year 2000 to 2024. It has been observed that heightened importance is being given to the reliable forecasting of stock prices because of the unpredictable nature and volatility of returns as well as risks associated with the stock market investments, where investment risks possess a significant impact on the institutions and at the same time both financial firms as well as the regulatory authorities have paid special concern to this problem. By its nature, stocks as another aspect of the distribution of assets have always been considered the favorite among investors due to the high return rate. unprecedented and unrelenting evolution the literature on stock price prediction as a field of study has not slowed down. The necessary data of this project concerned about the XGBoost process and extend from about 24 years of work.

INTRODUCTION

Analyzing the stock market has been an important focus in both the academic and practical arena for many years, owing to the attention it attracts due to its volatility and therefore the potential for great gains or losses. The present state of the stock market's dynamic character and the prospect of its further development continues to be a matter of keen interest among investors who care greatly about choosing an optimal trend for stocks. It is difficult to isolate specific causes for the oscillations in the stock market as multiple factors like political events, changes in economic data, culture, and even the general public mood have impact on stock prices. Therefore, the search for accurate approaches in the prediction of stock price movements has come out a constant issue of focus and research.

This endeavour has been driven by the desire to effectively reduce the amount of risk that is associated with stock market investment while at the same time pursuing the maximalization of potential gain. In the pursuit of this aim, research and practitioners have looked to draw upon numerous methods and LOGICAL methodologies, one of which is the use of historical stock price data for predictive modelling. Using historical data which includes among other things opening prices, closing prices, trading volumes and daily trends, analysts seek to identify recurring patterns and trends that may be seen to potentially provide an indication of the future direction of prices.

The key foundation for such a form of predictive modeling is within the ability of historical data to signal the future tendencies of the market. Using the matics in the historical series, analysts try to find reoccurring periodicities, relationships between variables, and contexts that could be useful for their forecasts of subsequent fluctuations in the stock price. Core to it is the application of sophisticated mathematical tools to learn information processing patterns from the data that exist by training behavior

learning models over large historical data sets and make findings that can go unnoticed by human observations.

In this context, the main goal of the model described in the present paper is to apply the principles of predictive business analytics and machine learning for evaluation of the future stock prices trend. In particular, the model aims at offering valuable recommendations to investors with regard to the opening rate, minimum rate, maximum rate, closing rate, and overall trading pattern of a particular stock during a given trading day. In order to achieve this objective, the model uses a historical stock market data as a network imported from a CSV file and applies the XGBoost as an enhanced machine learning model.

In this work, the approach taken captures several phases that are vital in helping attain the ability to forecast future stock prices well. Initially, historical stock market data from a CSV file is gained, which captures virtually all the market parameters possible such as the opening price, closing price, volumes, and daily fluctuations. Next, current data is extracted from Yahoo Finance using the yfinance library in Python so that the assumed model has a live, up-to-date outlook on the current market. This particular model is an attempt to aggregate the predictive capabilities of automative data analytics complemented by machine learning techniques to produce accurate predictions on the prospective movement of the stock price. Through the use of past data to analyze trends together with the help of highly developed machine learning methods, an investor receives essential information that may help him determine the possible future state of stock prices, thus helping investors to reduce risks effectively and make wise investment decisions based on it. In light of this, it is significant to note that the study lies in a field of development of the predictive modelling, but it is still relevant to the efforts to continue the development of the new and better ways of stock market forecasting. The libraries being used in the code of the application were the following.

- Pandas: that used for loading and processing the specified csv file, thus it was involved in both data preprocessing part of the study, and the data analysis.
- XGBoost: employed to establish the structure of the machine learning algorithm, and to train the classifier in stock forecast.
- Scikit learn: Data partition where the data set is split into training and testing sets, model optimization by grid search cross validation and converting categorical data into numerical values.
- Tkinter: GUI (Graphical User Interface)abyte
- Tcalender: It was found out that it's used in selection of date.
- Yfinance: used for fetching the financial data to fetch the financial data from yahoo finance.
- OS: For instance, the 'if' statement is utilized for confirming the existence of a file as well as handling the operations related to files.
- PIL(python imaging library) : initializing and preparing the display on the venue of the background image.

LITERATURE REVIEW

Finance as a field of study has been significantly and positively affected by the emerging technology, more especially, the use of Artificial Intelligence and Machine Learning in the study and prediction of the stock market. Due to the abundance of data sources in the financial domain that includes social media, news sentiment, economics, and other varieties of data feeding into it – advanced models have been designed to handle the incoming data streams and make sound of them.

Multi-source heterogeneous data in stock market, it could incorporate stock price data, trading volumes, financial ratios, company earnings reports, news articles, social media posts, macroeconomic data and several others. These diverse information is a treasure trove of information to be used in gauging market trends, or the psychological state of investors, or even the fundamental causes of certain changes in stock prices. Whereas EMH holds to the view that stock prices adjust for all available information, behavioural finance advocates for the notion that the behaviour of investors and their adherence to certain psychological processes significantly implicates market movements to afford certain levels of inefficiency. As a result, studies on stock market prediction endeavour to find out various patterns and signals in information that presage the probability of price variation. Of specific field of interest, predictive techniques like artificial neural networks(XGBoost) have been identified as suitable approaches to aid in the prediction of a stock market. Therefore, XGBoost, that is an Ensemble learning model that operates based on decision trees, has also been used significantly for various prediction tasks such as to predict the stock prices effectively.

This is due to the capability of XGBoost models that can capture intricate features of price history and market characteristics through historical price, technical indicators and other types of data.
Furthermore, researchers have explored the integration of XGBoost with other machine learning techniques, such as deep learning and natural language processing (NLP), to enhance prediction accuracy further. Hybrid models combining XGBoost with recurrent neural networks (RNNs) or transformer-based architectures have shown promising results in incorporating sequential information from time-series data and textual data from news articles and social media.

METHODOLOGY FOR TREND PREDICTION

The Stock Price Tracker & Trend Predictor system goes through several stages to develop, and I'll break them down for you. We start by preprocessing the data, selecting the right model, tuning the hyperparameters, designing the user interface, and evaluating the system. This report will give you all the details about each step so you can understand the development process better.
Let's start with Data Preprocessing. In this step, we load and process the dataset. The dataset is stored in a CSV file called 'TCS.csv' and it contains historical stock market data for Tata Consultancy Services (TCS). We make sure the 'Date' column is in the correct format, and we extract additional features like 'Year', 'Month', and 'Day' to capture temporal information. For the target variable, which is the 'Trend', we use the Label Encoder to assign numerical labels to bullish (1) and bearish (0) trends.

There you have it! We've covered the first step of the process, and now you know how we preprocess the data.
Table 1.1 Sample of data stored in CSV [4]

Date	Open	High	Low	Close	Volume	trend
08/12/2002	38.725	40	38.725	39.7	212976	bullish
08/13/2002	39.75	40.3875	38.875	39.1625	153576	bearish
08/14/2002	39.25	39.25	35.725	36.4625	822776	bearish
08/15/2002	36.4625	36.4625	36.4625	36.4625	0	bearish
08/16/2002	36.275	38	35.75	36.375	811856	bullish
08/19/2002	36.675	36.675	35.1375	35.475	205880	bearish
08/20/2002	35.725	38.725	35.4875	36.4625	3773624	bullish

Model Selection and Training:
We chose the XGBoost (Extreme Gradient Boosting) algorithm as our main model for predicting stock trends because it's really good at handling structured data and capturing complicated relationships. We used the XGBClassifier from the XGBoost library and trained it on our pre-processed data. To make the model perform even better, we did some hyperparameter tuning using GridSearchCV. This helped us find the best combination of hyperparameters like 'max_depth', 'learning_rate', and 'n_estimators'. We then selected the classifier that performed the best based on accuracy scores we got through cross-validation.

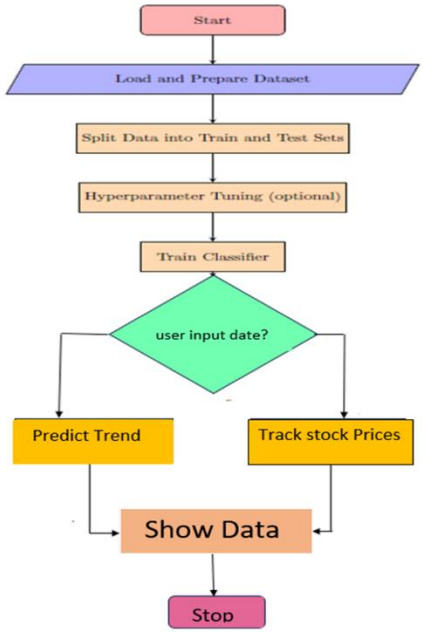

Fig.1.2 Flow chart showing working of machine learning algorithm

User Interface Design:
The Tkinter library in Python was used to build the GUI, which provides the user with an intuitive and interactive environment in which to work. The interface consists of mechanisms for picking dates, and identification stock entries are represented by widgets where the entries can be made. There are buttons to make predictions and load the stock price data; after the data is loaded, it is rendered in the interface. Furthermore, a background image was incorporated for better visual appeal of the interface.

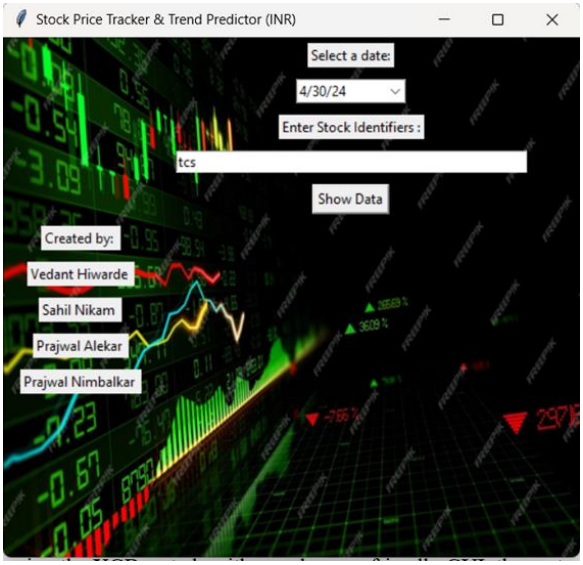

Fig.1.3 Home page of the application(GUI)[6]

System Evaluation:
The performance evaluation of the system took a two-fold approach: the accuracy of stock trend predictions and GUI functionality and usability. The former was conducted by testing the accuracy of the stock trend prediction using previous market data. The latter was done by conducting user testing and collecting user feedback on the system's GUI. Software testing on the GUI collected feedback, and all problems and bugs were identified and fixed to ensure the system was robust and reliable. In its development, the Stock Price Tracker & Trend Predictor system was designed to meet the following requirements: comprehensive data preprocessing, model selection, and hyperparameter tuning. Such a GUI, system evaluation, and the methodology in which the final results were obtained. By fine-tuning the XGBoost and creating an intuitive GUI, the project will tend to deliver accurate stock trend predictions and, therefore, the possibility of aiding users in investment decision-making. Through continuous refinements and updates, as per the feedback from users and market dynamics, make the system more effective and user-friendly.

Python Code of Implementation:[2]

```
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from tkinter import *
from tkinter import messagebox
from datetime import datetime
import yfinance as yf
from PIL import Image, ImageTk
import os
from tkcalendar import DateEntry  # Importing DateEntry widget from tkcalendar module
import tkinter.ttk as ttk

# Load your dataset (replace 'TCS.csv' with the path and filename of your CSV file)
try:
    data = pd.read_csv('TCS.csv')
except FileNotFoundError:
    print("Error: CSV file 'TCS.csv' not found.")
    exit(1)

# Parse the Date column with the correct format
data['Date'] = pd.to_datetime(data['Date'], format='%m/%d/%Y')

# Extract features (Year, Month, Day, Open, High, Low, Close) and target variable (Trend)
data['Year'] = data['Date'].dt.year
data['Month'] = data['Date'].dt.month
data['Day'] = data['Date'].dt.day

# Map trend to numerical labels (0 for bearish, 1 for bullish)
label_encoder = LabelEncoder()
data['trend'] = label_encoder.fit_transform(data['trend'])

# Prepare features (X) and target variable (y) for training
X = data[['Year', 'Month', 'Day', 'Open', 'High', 'Low', 'Close']]
y = data['trend']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize XGBoost Classifier with adjusted parameters
clf = xgb.XGBClassifier(objective='binary:logistic', random_state=42)

# Optionally perform hyperparameter tuning using GridSearchCV
param_grid = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.1, 0.01],
    'n_estimators': [100, 200]
}

grid_search = GridSearchCV(clf, param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)

best_clf = grid_search.best_estimator_

# Train the classifier on the full training data with the best parameters
best_clf.fit(X_train, y_train)

# Function to predict trend for user-provided date
def predict_trend():
    date_input = cal.get_date().strftime("%m/%d/%Y") # Convert datetime.date to string
    try:
        user_date = datetime.strptime(date_input, "%m/%d/%Y")
    except ValueError:
        messagebox.showerror("Error", "Invalid date format. Please use the format MM/DD/YYYY.")
        return

    # Prepare input data for prediction using the user-provided date
    input_data = {
        'Date': [user_date],
        'Open': [0.0],    # Example values for other features (not used for prediction)
        'High': [0.0],
        'Low': [0.0],
        'Close': [0.0]
    }
    input_df = pd.DataFrame(input_data)
    input_df['Year'] = input_df['Date'].dt.year
    input_df['Month'] = input_df['Date'].dt.month
    input_df['Day'] = input_df['Date'].dt.day

    # Make predictions on the input data
    new_predictions = best_clf.predict(input_df[['Year', 'Month', 'Day', 'Open', 'High', 'Low', 'Close']])

    # Display the predicted trend
    if new_predictions[0] == 1:
        messagebox.showinfo("Prediction Result", f"Predicted Trend for {date_input}: Upward (Bullish)")
    else:
        messagebox.showinfo("Prediction Result", f"Predicted Trend for {date_input}: Downward or Neutral (Bearish)")

# Function to get stock data from Yahoo Finance
def get_stock_data(stock_identifier):
    try:
        stock = yf.Ticker(f"{stock_identifier}.NS")
        data = stock.history(period='1d')

        if not data.empty:
            current_price = data['Close'][0]
            opening_price = data['Open'][0]
            highest_price = data['High'].max()
            lowest_price = data['Low'].min()

            # Predict trend for the current date
            date_input = datetime.today().strftime("%m/%d/%Y")
            user_date = datetime.strptime(date_input, "%m/%d/%Y")

            # Prepare input data for prediction using the current date
            input_data = {
                'Date': [user_date],
                'Open': [0.0],  # Example values for other features (not used for prediction)
                'High': [0.0],
                'Low': [0.0],
                'Close': [0.0]
            }

            input_df = pd.DataFrame(input_data)
```

```
        input_df['Year'] = input_df['Date'].dt.year
        input_df['Month'] = input_df['Date'].dt.month
        input_df['Day'] = input_df['Date'].dt.day

        # Make predictions on the input data
        prediction = best_clf.predict(input_df[['Year', 'Month', 'Day', 'Open', 'High', 'Low', 'Close']])
        predicted_price = current_price * (1 + 0.01 if prediction[0] == 1 else -0.01)  # Adjust the predicted price

        stock_data = {
            'Stock Identifier': stock_identifier,
            'Current Price (INR)': current_price,
            'Opening Price (INR)': opening_price,
            'Highest Price (INR)': highest_price,
            'Lowest Price (INR)': lowest_price,
            'Predicted Price (INR)': predicted_price  # Include the predicted price
        }

        return stock_data
      else:
        raise ValueError("No data available for the stock")

    except Exception as e:
      messagebox.showerror("Error", f"Failed to fetch data for {stock_identifier}: {e}")
      return None

# Function to track stock prices
def track_stock_prices():
    stock_identifiers = entry.get().strip().split(',')

    if not stock_identifiers:
      messagebox.showerror("Error", "Please enter at least one stock identifier")
      return

    stock_data_list = []

    for identifier in stock_identifiers:
      stock_data = get_stock_data(identifier)

      if stock_data is not None:
        stock_data_list.append(stock_data)

    if stock_data_list:
      save_to_excel(stock_data_list)
      messagebox.showinfo("Success", "Stock data tracked and saved successfully")

# Function to save stock data to Excel
def save_to_excel(stock_data_list):
    file_name = 'stock_prices_INR.xlsx'

    try:
      if os.path.exists(file_name):
        # Read existing data
        existing_df = pd.read_excel(file_name)
        df = pd.DataFrame(stock_data_list)
        # Append new data to existing DataFrame
        df_combined = pd.concat([existing_df, df], ignore_index=True)
        df_combined.to_excel(file_name, index=False)
      else:
        # Create new Excel file
        df = pd.DataFrame(stock_data_list)
        df.to_excel(file_name, index=False)

    except Exception as e:
      messagebox.showerror("Error", f"Failed to save data to Excel: {e}")

# Function to set background image
def set_background_image(window):
    try:
      image = Image.open("background.jpg")
      image = image.resize((500, 450))
      photo = ImageTk.PhotoImage(image)
      background_label = Label(window, image=photo)
      background_label.image = photo
      background_label.place(x=0, y=0, relwidth=1, relheight=1)

      label1.lift()
      entry.lift()
      cal.lift()
      # predict_button.lift()
      label2.lift()
      label3.lift()
      label4.lift()
      label5.lift()
      label6.lift()
      label7.lift()
```

```
        # track_button.lift()
        # show_data_button.lift()
        predict_track_show_button.lift()

    except Exception as e:
        messagebox.showerror("Error", f"Failed to set background image: {e}")

# Function to display fetched data in a tabular form
def show_data():
    stock_identifiers = entry.get().strip().split(',')
    if not stock_identifiers:
        messagebox.showerror("Error", "Please enter at least one stock identifier")
        return

    data_list = []
    columns = ["Stock Identifier", "Current Price (INR)", "Opening Price (INR)", "Highest Price (INR)", "Lowest Price (INR)", "Predicted Price (INR)", "Trend Prediction"]

    for identifier in stock_identifiers:
        stock_data = get_stock_data(identifier)
        if stock_data is not None:
            # Predict trend for the current date
            date_input = datetime.today().strftime("%m/%d/%Y")
            user_date = datetime.strptime(date_input, "%m/%d/%Y")

            # Prepare input data for prediction using the current date
            input_data = {
                'Date': [user_date],
                'Open': [0.0],  # Example values for other features (not used for prediction)
                'High': [0.0],
                'Low': [0.0],
                'Close': [0.0]
            }

            input_df = pd.DataFrame(input_data)
            input_df['Year'] = input_df['Date'].dt.year
            input_df['Month'] = input_df['Date'].dt.month
            input_df['Day'] = input_df['Date'].dt.day

            # Make predictions on the input data
            prediction = best_clf.predict(input_df[['Year', 'Month', 'Day', 'Open', 'High', 'Low', 'Close']])
            prediction_text = "Upward (Bullish)" if prediction[0] == 1 else "Downward or Neutral (Bearish)"

            data_list.append([
                stock_data['Stock Identifier'],
                stock_data['Current Price (INR)'],
                stock_data['Opening Price (INR)'],
                stock_data['Highest Price (INR)'],
                stock_data['Lowest Price (INR)'],
                stock_data['Predicted Price (INR)'],  # Include the predicted price
                prediction_text
            ])

    if data_list:
        # Create a new window for displaying data
        data_window = Toplevel()
        data_window.title("Fetched Stock Data")

        # Create Treeview widget for tabular display
        tree = ttk.Treeview(data_window, columns=columns, show='headings')
        for col in columns:
            tree.heading(col, text=col)

        # Insert fetched data into the Treeview
        for i, row in enumerate(data_list, start=1):
            tree.insert("", "end", values=row)

        tree.pack(expand=True, fill='both')

# Function to perform predict, track price, and show data actions
def predict_track_show():
    predict_trend()
    track_stock_prices()
    show_data()

# Create GUI window
window = Tk()
window.geometry("500x450")
window.title("Stock Price Tracker & Trend Predictor (INR)")

# GUI components
label1 = Label(window, text="Select a date:")
```

```
label1.grid(row=20, column=28, padx=15, pady=5)

# Using DateEntry widget for date selection
cal = DateEntry(window, background='darkblue', foreground='white', borderwidth=2)
cal.grid(row=22, column=28, padx=15, pady=5)

# Buttons and other components
# predict_button = Button(window, text="Predict Trend", command=predict_trend)
# predict_button.grid(row=24, column=28, padx=15, pady=5)

label2 = Label(window, text="Enter Stock Identifiers :")
label2.grid(row=30, column=28, padx=15, pady=5)

entry = Entry(window, width=50)
entry.grid(row=40, column=28, padx=15, pady=5)

# track_button = Button(window, text="Track Prices", command=track_stock_prices)
# track_button.grid(row=50, column=28, padx=15, pady=5)

# show_data_button = Button(window, text="Show Data", command=show_data)
# show_data_button.grid(row=60, column=28, padx=15, pady=5)

predict_track_show_button = Button(window, text="Show Data", command=predict_track_show)
predict_track_show_button.grid(row=70, column=28, padx=15, pady=5)

label3 = Label(window, text="Created by: ")
label3.grid(row=80, column=2, padx=15, pady=5)
label4 = Label(window, text="Vedant Hiwarde ")
label4.grid(row=82, column=2, padx=15, pady=5)
label5 = Label(window, text="Sahil Nikam ")
label5.grid(row=84, column=2, padx=15, pady=5)
label6 = Label(window, text="Prajwal Alekar ")
label6.grid(row=86, column=2, padx=15, pady=5)
label7 = Label(window, text="Prajwal Nimbalkar ")
label7.grid(row=88, column=2, padx=15, pady=5)

# Set background image
set_background_image(window)

# Start the main GUI event loop
window.mainloop()
```

Result:

After the application of each step of the methodology, we get the final result.

To make a stock analysis or prediction we can find the trend of the particular stock (TCS as per project) just by entering the date.

Whenever the date is entered in the software with help of stored data and Ml algorithm carries on its analysis and presents a trend (bullish or bearish) as output and stock prediction is carried better accuracy as compared to earlier techniques or methods.

Along with the prediction the software also implies a tending hand to know the prices (open, close, low, high) for the present day.

Due to this, there will be no overdependency on the software, and running parallel to the software analysis human analysis can be done.

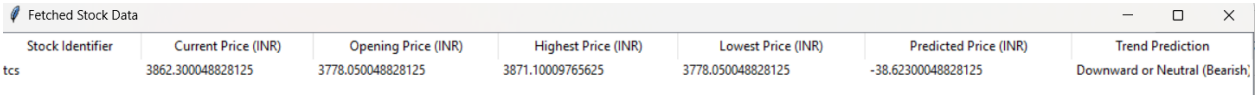

Fig.1.3Result window displaying the data

ACKNOWLEDGEMENT

I hereby take this opportunity to express my sincere thanks to deep sense of gratitude to my guide Prof. Kedar Kulkarni sir for his kind co-operation and encouragement to me during this project. This project wouldn't be possible without his motivation and ever-increasing support. His curiosity, dedication, and enthusiasm about the project made this project a success.

I would like to express sincere gratitude to Prof. Y. R. Ingole, Head of Robotics & Automation Engineering Department for believing in me and nurturing my ideas. I would also like to thank Dr. M.G Reddy & Dr. Gaurav Kumar (Project based learning Coordinator) for his support, help & cooperation provided for the Project work. I would like to thanks Prof. (Dr.) A. M.

Kate, Principal, Zeal College of Engineering & Research.
I would like to thanks the teaching as well as non-teaching staff of Robotics & Automation Engineering Department for their support. I also acknowledge with thanks, assistance provided by central library staff.
Last, but not least, I would like to thank all my colleagues, friends and most importantly my parents for their valuable co-operation and who believed in me and supported me in their own ways.

CONCLUSION

It is mentioned that stock market is major contribution to economy and numerous people especially beginners and young ones highly invest in stock market in order to achieve higher returns ,but due to lack of data, knowledge and poor analysis they suffer huge debt.
Dur to which hearing form the experiences of this beginers the next generation interested in Investment in not as keen due to their past experiences .
But this project or software consists data of last 24 years and Machine Learning is used to study the data and predict the trend due to which the analysis has become much easier.
Hence the users especially the beginners mentioned before would find it easier to analyse and invest and ensure the pillar of economy goes strong.

REFERENCE

[1]The stock market prediction system: By C.K Gomathi
[2] https://chat.openai.com/
[3] https://www.geeksforgeeks.org/xgboost/
[4] https://finance.yahoo.com/quote/TCS?.tsrc=fin-srch
[5] https://pypi.org/project/yfinance/
[6]https://www.freepik.com/search?format=search&last_filter=query&last_value=stock&query=stock