JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

"Stock Insight Hub: Multi-Source Data Collection, Mining, Pre-processing, and Its Comprehensive Analysis"

Vikas. D. Kheni[1], Mayank K. Chotaliya[2], Meet N. Ghelani[3]

***Abstract* – This paper discusses Stock Insight Hub, a platform that merges financial analysis with technology to aid stock market investors. It provides a toolkit for data collection, analysis, and interpretation, using multiple data sources and advanced tools. The platform empowers investors with curated datasets, algorithms, and real-time insights. It emphasizes multi-source data analysis, Python, web scraping, and their impact on investment strategies. Overall, Stock Insight Hub offers a powerful resource for investors to optimize decisions and achieve success.**

***Keywords* - Stock Insight Hub, Informed decision-making, Stock market investment, Financial analysis, Cutting-edge technology, Comprehensive toolkit, Data collection, Data analysis, Interpretation, Multi-source data collection, Advanced analytical tools, Investor empowerment, Market complexities, Curated datasets, Sophisticated algorithms, Real-time market insights, Trend identification, Risk assessment, Opportunity capitalization, Python, Web scraping technologies, Informed investment strategies, Paradigm shift, Investment optimization, Financial success.**

I. INTRODUCTION

A. Brief Overview of Stock Insight Hub

Stock Insight Hub is a cutting-edge platform designed to revolutionize the way investors approach stock market analysis. It serves as a centralized hub for accessing a vast array of financial data, analytical tools, and market insights. Developed by a team of seasoned financial experts and data scientists, Stock Insight Hub aims to empower investors of all levels with the information and tools needed to make informed investment decisions.

B. Importance of Multi-Source Data Collection and Analysis in Stock Market

Reliance on a single data source or a limited range of analytical tools may restrict an investor's ability to recognize opportunities and efficiently manage risks in today's dynamic and fast-paced financial markets. This is where gathering and analyzing data from multiple sources is crucial.

Investors can obtain a more comprehensive understanding of market dynamics, trends, and sentiment by utilizing data from various sources, including stock exchanges, financial news websites, regulatory filings, and social media platforms. As a result, they can find previously undiscovered information, recognize new opportunities, and make wise investment choices.

In addition, a wide range of factors impact the stock market, from company fundamentals and industry-specific trends to economic indicators and geopolitical events. By using multiple sources for data collection, investors can evaluate the influence of market drivers on both specific stocks and overall market trends.

Essentially, Stock Insight Hub acknowledges the inherent worth of gathering and analyzing data from multiple sources when navigating the intricacies of the stock market. The platform enables investors to stay ahead of the curve, adjust to shifting market conditions, and ultimately meet their investment goals by giving them access to a wide variety of data sources and sophisticated analytical tools.

II. DETAILED FINANCIAL ANALYSIS

A. Overview of curated dataset from the National Stock Exchange (NSE)

The National Stock Exchange (NSE) is the source of the carefully selected dataset that forms the basis of Stock Insight Hub's analytical capabilities. A sizable collection of historical and current data on stocks, indices, and other financial instruments traded on the NSE is included in this dataset. Stock Insight Hub gives investors a thorough understanding of the stock market environment by compiling information from a variety of sources, such as stock prices, trading volumes, company financials, and market indicators.

To guarantee accuracy and relevance, the carefully org and the dataset is updated regularly. It provides the framework for some analytical models and tools that Stock Insight Hub

[1] *Student, Department of Information Technology, Dharmsinh Desai University, Nadiad, Gujarat, India*
[2] *Student, Department of Information Technology, Dharmsinh Desai University, Nadiad, Gujarat, India*
[3] *Student, Department of Information Technology, Dharmsinh Desai University, Nadiad, Gujarat, India*

offers, allowing investors to perform comprehensive research and get useful insights.

B. Explanation of Techniques Used

(i) Bollinger Bands

Bollinger Bands is a technical analysis tool used in stock trading to identify short-term price movements and potential entry and exit points. They are a type of price envelope that defines upper and lower price range levels, plotted at a standard deviation level above and below a simple moving average of the price. The bands widen when there is a price increase, and narrow when there is a price decrease.

$$\text{Standard Deviation } \Sigma = \sqrt{\frac{\sum_{J=1}^{N}(x_J - \bar{x})}{N}} \quad \text{..Equation (i)}$$

$$\text{Mean } \bar{X} = \frac{\sum_{J=1}^{N} x_J}{N} \quad \text{.... Equation (ii)}$$

$$\text{Upper band} = \bar{x} + 2\sigma \quad \text{... Equation (iii)}$$

$$\text{Moving Average} = \bar{x} \quad \text{... Equation (iv)}$$

$$\text{Lower band} = \bar{x} - 2\sigma \quad \text{... Equation (v)}$$

Fig.1 Visual Representation of Bollinger Bands along with Upper Band, Lower Band, and Moving Average

(ii) Candlestick Analysis

Candlestick analysis is a method of analyzing price movements and market sentiment based on the shapes and patterns formed by candlestick charts. Each candlestick represents a specific period (e.g., one day) and contains information about the opening, closing, and high, and low prices during that period. By studying patterns such as doji, hammer, and engulfing, investors can gain insights into market psychology and anticipate future price movements.

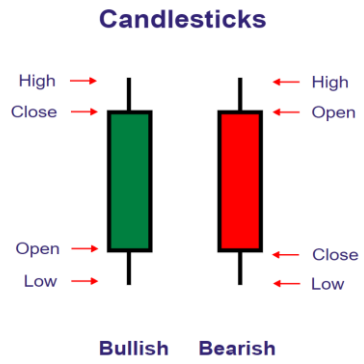


Fig. 2 Visual Representation of Candlestick

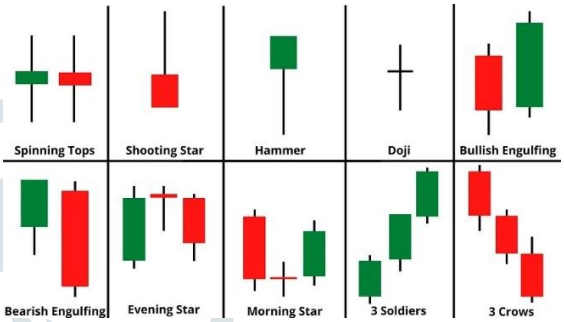


Fig. 3 Image of various patterns of Candlestick

(iii) Percentage Delivery Analysis

Percentage Delivery analysis is a fundamental analysis technique used to assess investor interest and market liquidity in a particular stock. It involves calculating the percentage of shares that are physically delivered (transferred from sellers to buyers) relative to the total volume of shares traded. High percentage delivery indicates strong investor confidence and interest in a stock, while low percentage delivery may signal weakness or manipulation.

(iv) Golden Cross Over Signals

Fig.4 Visual Representation of Golden Crossover

Golden cross-over signals are a technical analysis indicator that occurs when a short-term moving average crosses above a long-term moving average. This signal is often interpreted as a bullish sign, indicating a potential upward trend in the price of a stock or index. Investors use Golden Cross Over signals to identify entry points for long positions and confirm bullish market sentiment.

III. Data Collection And Visualization

We are scraping the data of TCS's stock with the symbol 'TCS'. Selenium has been employed for web scraping, allowing us to efficiently retrieve real-time or historical data from the web pages containing the relevant information. Selenium's capabilities facilitate the automated navigation of web pages, enabling us to access and extract the desired data points such as stock prices, trading volumes, and other market indicators associated with TCS's stock.

The data after the cleaning process is shown below. As shown in the image, we are available with 21 features (columns), which are pre-calculated.

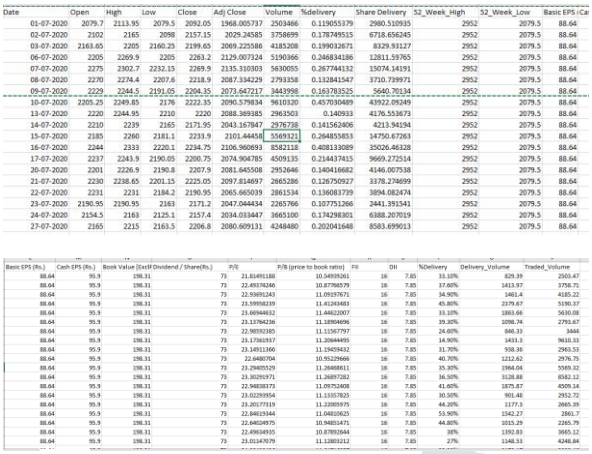


Fig. 5 Sample of prepared data for the TCS stock

Now, that data has been prepared, we can easily and accurately visualize the data using various charts.

A. Simple Visualization

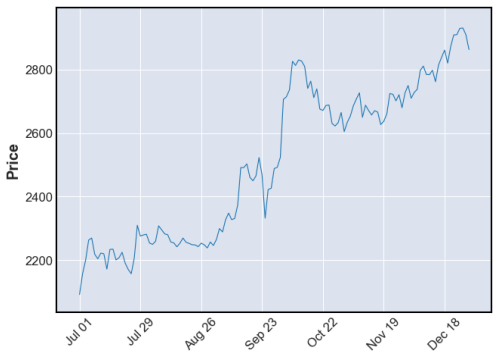


Fig.6 Line graph representation of TCS stock data

The graph in Fig. 5 has been plotted using the "plot" method if the "pdf" library. It helps users visualize how values change over different periods in 2020 by clearly and simply displaying the trends or patterns of the data over time.

B. Candlestick Representation

```
data_2023 = data['2023-05-01':'2023-07-01']
mpf.plot(data_2023, type='candle',title='Stock Open Price in May - July',
style='yahoo',figsize=(12, 6))
```

Fig. 6 Code snippet for plotting candlestick graph of TCS stock data

This code shown in Fig. 6 selects a specific portion of the data, focusing on the period from May 1, 2023, to July 1, 2023, and stores it in a new variable called "data_2023". Then, it uses the "pdf" library to create a candlestick chart from this selected data. The chart displays the open, high, low, and close prices of a stock for each day within the specified time frame. The title of the chart is set to "Stock Open Price in May - July". The chart is styled in the 'Yahoo' theme, and its size is adjusted to be 12 inches wide and 6 inches tall.

Fig. 7 Candle Stick representation of TCS stock data

C. Candlestick Representation but adding one Volume

The code shown in Fig. 8 snippet builds upon the code snippet in Fig. 6, by adding volume data to the candlestick chart, giving users insights into trading activity alongside stock price movements. The chart's title reflects this inclusion, clarifying that it visualizes both stock open prices and trading volumes. Additionally, the layout is optimized for readability with elements neatly arranged and an adjusted aspect ratio providing a broader view of the data. Overall, the second code snippet enhances the visualization's comprehensiveness and clarity, offering a more complete picture of market dynamics during the specified time frame.

```
mpf.plot(data_2023, type='candle', title='Stock Open Price & Volume Data
',style='yahoo',tight_layout=True, volume=True,figratio=(16,9))
```

Fig. 8 Code snippet of plotting candlestick plot with volume of TCS stock data

Fig. 9 Candlestick representation along with volume of TCS stock data

D. Moving Average SMA 5 & SMA 20

```
mpf.plot(data['2020-07':'2020-12'], type='candle', volume=True,mav=(20,5),
title = 'TCS - JAN - DEC(2020)',tight_layout=True, figratio=(16,9))
```

Fig. 10 Code snippet of plotting Moving Averages SMA 5 & SMA 20

The code shown in Fig. 10 generates a candlestick chart for TCS stock data from July to December 2020, showcasing the open, close, high, and low prices for each day during this period. Additionally, it includes volume bars underneath the candlesticks, depicting the amount of trading activity. The chart is further enhanced with moving average lines calculated over 20-day and 5-day periods, providing insights into longer-term trends and short-term fluctuations. Its title, "TCS - JAN - DEC(2020)", summarizes the time frame of the data. With a wide aspect ratio and optimized layout, the visualization ensures clarity and readability, offering a comprehensive overview of TCS stock performance throughout the specified months.

Fig. 11 Graph of Moving Averages SMA 5 & SMA 20 for TCS stock data

IV. Algorithms And Their Visualization

A. *Golden Crossover Signals*

```
def GoldenCrossverSignal(name, point):
  path = f'../Data/{name}.csv'
  data = pd.read_csv(path,parse_dates=['Date'],index_col='Date') #using for removing the first index


  data['20_SMA'] = data.Close.rolling(window=20, min_periods=1).mean()   #20 SMA's data
  data['50_SMA'] = data.Close.rolling(window=50, min_periods=1).mean()   #50 SMA's data

  data['Signal'] = 0
  data['Signal'] = np.where(data['20_SMA'] > data['50_SMA'],1,0)   #numpy's function which check the condition
and simply assign the value
  # 1 -> sell & 0 -> buy in data['signal']

  data['Position'] = data.Signal.diff()  #difference between upper and lower value  1 -> buy  -1 sell
  plt.figure(figsize = (20,10))

  # plot close price, short-term and long-term moving averages
  data.iloc[-point:]['Close'].plot(color = 'k', label= 'Close Price')
  data.iloc[-point:]['20_SMA'].plot(color = 'b',label = '20-day SMA')
  data.iloc[-point:]['50_SMA'].plot(color = 'g', label = '50-day SMA')

  # plotting buy signals
  plt.plot(data.iloc[-point:][data.iloc[-point:]['Position'] == 1].index,
      data.iloc[-point:]['20_SMA'][data.iloc[-point:]['Position'] == 1],
      '^', markersize = 15, color = 'g', label = 'buy')
  # plotting sell signals
  plt.plot(data.iloc[-point:][data.iloc[-point:]['Position'] == -1].index,
      data.iloc[-point:]['20_SMA'][data.iloc[-point:]['Position'] == -1],
      'v', markersize = 15, color = 'r', label = 'sell')
  plt.ylabel('Price in Rupees', fontsize = 15 )
  plt.xlabel('Date', fontsize = 15 )
  plt.title(name, fontsize = 20)
  plt.legend() #indications
  plt.grid()
  plt.show()
  #for table the data
  df_pos = data.iloc[-point:][(data.iloc[-point:]['Position'] == 1) | (data['Position'] == -1)].copy()
  df_pos['Position'] = df_pos['Position'].apply(lambda x: 'Buy' if x == 1 else 'Sell')


  print(tabulate(df_pos[['Close', 'Position']], headers = 'keys', tablefmt = 'psql'))
```

Fig. 12 Algorithm to analyse Golden Crossover

Below Figure Fig. 13 shows the tabular form of the outcome, indicating the optimal points to buy and sell the particular stock within a given time frame. This tabular representation offers a clear and concise summary of the trading strategy derived from the analysis, presenting the recommended actions for investors seeking to maximize their returns. Each entry in the table specifies the timing and conditions under which buying or selling actions are advisable, based on the underlying data and methodology employed in the analysis.

Date	Close	Position
2022-11-02 00:00:00	3241.7	Buy
2023-01-02 00:00:00	3261.45	Sell
2023-01-25 00:00:00	3429.75	Buy
2023-03-15 00:00:00	3198.9	Sell
2023-05-18 00:00:00	3199.85	Buy
2023-06-28 00:00:00	3197.35	Sell
2023-07-14 00:00:00	3514.65	Buy

Fig. 13 Tabular form of the outcome where to buy and where to sell the particular stock within a given time

Below shown in Figure Fig. 14 is the visual representation corresponding to the table in Figure Fig. 13. This visual depiction offers a more intuitive understanding of the recommended buy and sell points identified in the tabular form. Buy signals are indicated by upward-pointing green arrows, while sell signals are represented by downward-pointing red arrows.

B. *Bollinger Bands*

To calculate the Bollinger Bands effectively, we begin by computing the mean, which is essentially a 20-day moving average of the closing prices of TCS stock data. This moving average provides a smoothed representation of the stock's price trends over the specified time frame. Equation (ii) guides us in this calculation. Following this, we determine the standard deviation of the closing prices using Equation (i). The standard deviation quantifies the dispersion of the closing prices around the mean, offering insights into the volatility of the stock. Together, these metrics form the basis for establishing the upper and lower bands of the Bollinger Bands, which assist traders in identifying potential overbought or oversold conditions in the market. The code snippet presented below demonstrates how to execute these calculations programmatically.

```
data['20_MA'] = data['Close'].rolling(window=20).mean() #20 moving average
data['std'] = data['Close'].rolling(window=20).std() #standard deviation
```

Fig.14 Code snippet of calculation of SMA and Standard Deviation of Closing price of TCS stock data

After computing the Simple Moving Average (SMA) and the Standard Deviation, the next step involves calculating the upper and lower bands of the Bollinger Bands. This can be achieved by applying Equation (iii) and Equation (v). Equation (iii) helps us determine the upper band by adding two times the standard deviation to the SMA. Conversely, Equation (iv) enables the calculation of the lower band by subtracting two times the standard deviation from the SMA. These upper and lower bands serve as dynamic thresholds, indicating potential areas of resistance and support in the price movements of the stock. By comparing the current price of the stock with these bands, traders can gain valuable insights into the likelihood of price reversals or continuations. The provided code snippet illustrates how to implement these calculations effectively.

```
data['upper_band'] = data['20_MA'] + (data['std'] * 2)
data['lower_band'] = data['20_MA'] - (data['std'] * 2)
```

Fig.15 Code snippet of calculating Upper Band and Lower Band for Bollinger Bands

After completing all the necessary calculations, the next step is to visualize the Bollinger Bands using the computed

variables. This visualization provides a graphical representation of the stock's price movements relative to the upper and lower bands. By plotting the closing prices along with the upper and lower bands on a chart, traders can easily identify potential buying or selling opportunities based on the deviation of the current price from these bands.

```
plt.plot(data.index, data['Close'], label='Close Price', color='black')
plt.plot(data.index, data['20_MA'], label='20-day Moving Average',
color='blue')
plt.plot(data.index, data['upper_band'], label='Upper Bollinger Band',
color='red', linestyle='dashed')
plt.plot(data.index, data['lower_band'], label='Lower Bollinger Band',
color='yellow', linestyle='dashed')
plt.fill_between(data.index, data['upper_band'], data['lower_band'],
color='gray', alpha=0.2)

plt.title(f"{symbol} Stock Price and Bollinger Bands")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.grid()
plt.show()
```

Fig.16 Code snippet of plotting Bollinger Bands

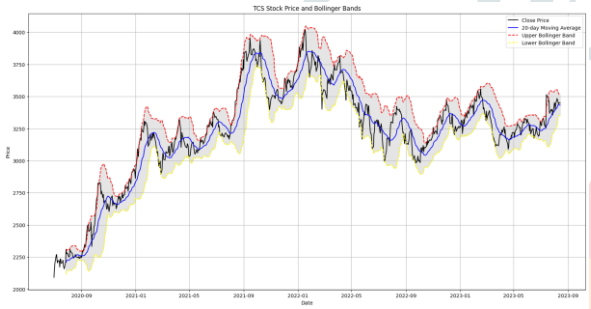


Fig.17 Graphical representation of Bollinger Bands

After plotting the graph of the Bollinger Bands, the next step is to apply the algorithm developed to identify the points of 'Overbought' and 'Oversold' on the Bollinger Band graph. This algorithm utilizes predefined criteria to determine when the stock price deviates significantly from the Bollinger Bands, indicating potential overbought or oversold conditions in the market. By analyzing the position of the closing price relative to the upper and lower bands, the algorithm generates signals for overbought and oversold levels.

```
start_date = '2023-01-01'
end_date = '2023-06-01'
selected_data = data[start_date:end_date]
selected_data['Signal'] = 'Normal'

selected_data.loc[selected_data['Close'] > selected_data['upper_band'], 'Signal'] =
'Overbought'
selected_data.loc[selected_data['Close'] < selected_data['lower_band'], 'Signal'] =
'Oversold'

signal_table = selected_data[selected_data['Signal'].isin(['Overbought',
'Oversold'])][['Close', 'Signal']].copy()

print(signal_table)
```

Fig.18 Algorithm for identifying 'Overbought' and 'Oversold' signals in Bollinger Band Graph

The table presented below illustrates the signals for overbought and oversold conditions, providing corresponding dates and closing prices where these signals are triggered. These signals serve as valuable indicators for traders, helping them identify opportune moments to enter or exit positions based on the current market sentiment.

Date	Close	Signal
2023-01-13	3374.550049	Overbought
2023-02-08	3520.100098	Overbought
2023-02-09	3540.850098	Overbought
2023-02-27	3331.850098	Oversold
2023-02-28	3312.850098	Oversold
2023-03-14	3214.949951	Oversold

Fig.19 Tabular representation of 'Overbought' and 'Oversold' Signals

A graphical representation of the 'Overbought' and 'Oversold' signals is depicted on the graph using upward arrows of green color for 'Overbought' conditions and downward arrows of yellow color for 'Oversold' conditions. These arrows are strategically placed on the graph at the corresponding points where the signals occur, indicating to traders the potential opportunities to either sell (in the case of overbought) or buy (in the case of oversold). This visual representation enhances the readability of the graph and assists traders in quickly identifying critical market conditions.

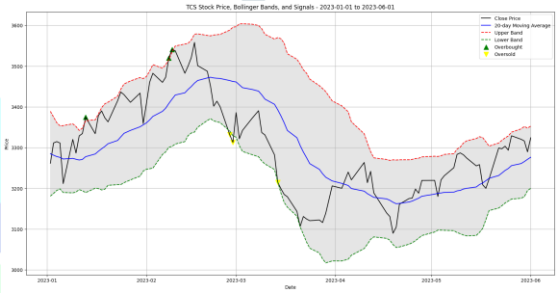


Fig.20 Graphical representation of 'Overbought' and 'Oversold' Signals

C. Percentage Delivery Analysis

We are using the data as shown in Fig.21 which contains specific timestamps and %delivery values.

2023-04-03	NaN
2023-04-05	51.0
2023-04-06	49.9
2023-04-10	26.9
2023-04-11	56.6
2023-04-12	47.4
2023-04-13	39.8
2023-04-17	62.4
2023-04-18	70.1
2023-04-19	41.3
2023-04-20	43.0
2023-04-21	59.3
2023-04-24	65.4

Fig. 21 Data of %delivery along with timestamp

Below is the graphical representation of data shown in Fig. 21.

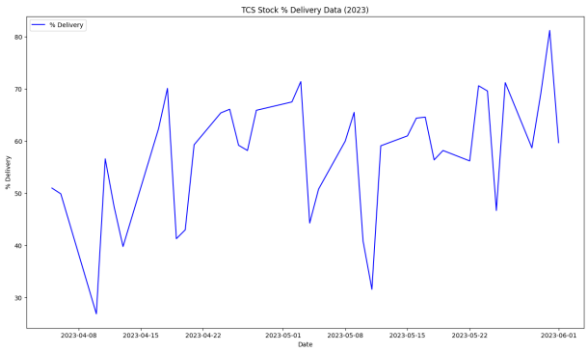
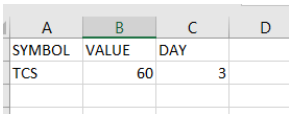


Fig.22 Graphical Representation of %delivery data

Now, for percentage delivery analysis our algorithm checks that for particular Days represented in Column-C, if the percentage delivery is continuously high for value in Column-B (i.e., here, we have taken the base percentage delivery value as 60) for the particular stock TCS denoted in Column-A.



	A	B	C	D
	SYMBOL	VALUE	DAY	
	TCS	60	3	

Fig.23 Image showing fixed parameters for our %delivery analysis algorithm

From the algorithm in Fig. 26, If any day has a percentage delivery value consistently higher than our fixed value of 60, then it indicates that it is good to sell the stock. This is represented in the image below, Fig.24, with highlighted color. From the above logic, it can be determined that stocks of 2023-05-03 and 2023-05-17 are good to sell.

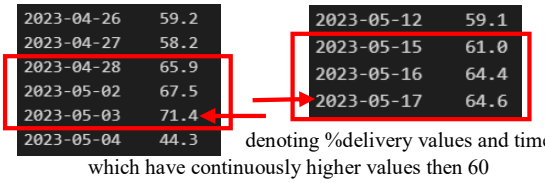
Above shown graph in Fig.25 is a graphical representation of highlighted points in Fig.24.

```
consecutive_days = 0
signal_dates = []

for date in data_2023.index:
    delivery = data_2023.loc[date, '%Delivery']
    if delivery > value:  # % > 60
        consecutive_days += 1
        if consecutive_days == day:
            signal_dates.append(date)
            consecutive_days = 0  # Reset counter
    else:
        consecutive_days = 0  # Reset counter


plt.figure(figsize=(16, 9))  # Width, Height

# Plot the sliced data with custom colors
plt.plot(data_2023['%Delivery'], color='blue', label='% Delivery')

# Set labels and title
plt.xlabel('Date')
plt.ylabel('% Delivery')
plt.title('TCS Stock % Delivery Data (2023)')

for signal_date in signal_dates:
    date_str = signal_date.strftime('%Y-%m-%d')
    print(date_str)
    num_date = date2num(signal_date) #matplotlib.dates module, converts the given Timestamp
object into a numeric val
    plt.axvline(x=num_date, color='red', linestyle='--', label='%Delivery: ' + date_str)

signal_num_dates = [date2num(date) for date in signal_dates]

# Scatter plot with triangle markers
plt.scatter(signal_num_dates, [data_2023.loc[date, '%Delivery'] for date in
signal_dates],marker='^', color='yellow', s=100, label='Signal')

plt.legend()
plt.show()
```

Fig.26 Algorithm of %delivery Analysis

D. PERCENTAGE DELIVERY ANALYSIS WITH FIXING PARAMETERS

In Topic-C, we have examined the increments that coincide with the condition established as per Fig. 23. In the above topic, we have assessed the amount of increase observed in the stock value while the condition in Fig.23 is true. In this section, in our final analysis, we calculate the average of the percentage delivery value and the increment in the stock value based on its closing value. Subsequently, utilizing these average values, we aim to identify a corresponding position to initiate the sell signal. This process allows us to refine our trading strategy by considering both the delivery percentage and the stock value increment, providing a more comprehensive approach to decision-making regarding when to sell.



2023-04-26	59.2
2023-04-27	58.2
2023-04-28	65.9
2023-05-02	67.5
2023-05-03	71.4
2023-05-04	44.3

2023-05-12	59.1
2023-05-15	61.0
2023-05-16	64.4
2023-05-17	64.6

Fig.24 Image denoting %delivery values and timestamp which have continuously higher values then 60

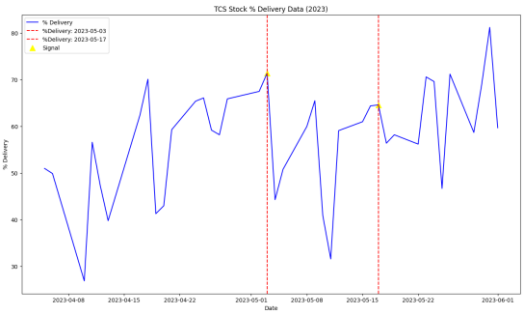


Fig.25 Graphical representation of points having timestamp '2023-05-03' and '2023-05-17'

Here is the algorithm to find the average increment in the Closing value of the stock (in %) and an average of %delivery changed by using a fixed window size of 10 and checking the conditions in Fig. 23.

```
window_size = 10
per = []
deliv = []
# Iterate through windows of size 10
for i in range(len(sample) - window_size + 1):
    window = sample.iloc[i : i + window_size]  # Get the current window

    first_close = window.iloc[0]['Close']
    last_close = window.iloc[-1]['Close']
    delivery_values =
window.iloc[:3]['%Delivery'].str.rstrip('%').astype(float)
    if all(val >= 60 for val in delivery_values):
        min_delivery = delivery_values.min()
        percentage = ((last_close - first_close) / first_close )*100
        if percentage > 0:
            date = window.index[0].date()
            #print(date, min_delivery, percentage)
            if percentage >= 2:
                per.append(percentage)
                deliv.append(min_delivery)
                print(date, min_delivery, percentage)
print("-------------------------------------")
print("Average increment Close % :-",sum(per)/len(per))
print("Average % delivery change :-",sum(deliv)/len(deliv))
```

Fig.27 Algorithm to calculate Average increment in closing value of stock and Average %delivery changed

Below Fig.28 is the output of the algorithm in Fig.27 by applying on the data present in Fig.21.

```
2022-05-19 60.5 2.8587195205558813
2022-05-20 60.5 3.9477679927118126
2022-05-23 60.5 3.564383291939493
2022-05-24 60.8 4.347626246958639
2022-05-25 63.0 6.157563336682146
2022-08-29 60.9 3.5242821430815745
2022-10-28 66.1 4.827312131510316
2022-10-31 66.1 4.4579835701055
2022-11-01 61.0 2.236406666129984
2022-11-09 60.5 3.075197851188668
2023-01-27 63.6 3.8052812809959433
2023-01-30 62.8 3.0259388687088062
2023-02-03 60.6 2.175286418002747
2023-05-15 61.0 2.268777803360902
---------------------------------------
Average increment Close % :- 3.782181129902137
Average % delivery change :- 62.21470588235293
```

Fig.28 Output of Above Algorithm

Now, we are fixing the values of Average increment close % and Average %delivery changed parameters by the output generated using the above algorithm of Fig.27.

```
window_size = 10
percent_close = 3.7821
delivery_change = 62.214

result_df = pd.DataFrame(columns=['Date', 'Min_Delivery', 'Percentage'])
# Iterate through windows of size 10
for i in range(len(sample) - window_size + 1):
    window = sample.iloc[i : i + window_size]  # Get the current window

    first_close = window.iloc[0]['Close']
    last_close = window.iloc[-1]['Close']
```

```
    delivery_values =
window.iloc[:3]['%Delivery'].str.rstrip('%').astype(float)
    if all(val >= delivery_change for val in delivery_values):
        min_delivery = delivery_values.min()
        percentage = ((last_close - first_close) / first_close )*100
        if percentage >= percent_close:
            date = window.index[0].date()
            result_df.loc[len(result_df)] = [date, min_delivery, percentage]

print(result_df)
```

Fig.29 Algorithm for calculating %delivery based on set parameters

By using above algorithm, we are available with some days where above condition values, that we have fixed based on the output of algorithm Fig.27, are satisfying.

	Date	Min_Delivery	Percentage
0	2021-07-29	63.5	4.643592
1	2021-08-17	62.4	6.569001
2	2021-11-18	64.0	4.810540
3	2021-11-22	67.3	5.263996
4	2021-12-28	63.0	4.675508
5	2022-05-25	63.0	6.157563
6	2022-10-28	66.1	4.827312
7	2022-10-31	66.1	4.457984
8	2023-01-27	63.6	3.805281

Fig.30 Output of Above algorithm

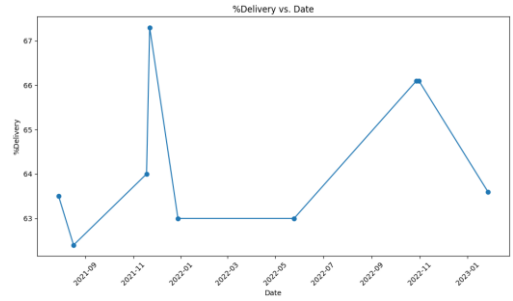


Fig.31 Graphical representation of the output of the above algorithm

In the final phase, we apply the Golden Crossover Signal algorithm to the output obtained from the previous analysis. This algorithm involves identifying instances where the short-term moving average of the stock price crosses above the long-term moving average, signaling a potential buying opportunity (Golden Cross). Conversely, when the short-term moving average crosses below the long-term moving average, it suggests a potential selling opportunity (Death Cross). By implementing this algorithm on our output data, we aim to generate precise buy and sell signals based on the identified crossovers, thereby enhancing the effectiveness of our trading strategy.

Date	Close	Position
02-03-2021	3006.35	Sell
07-04-2021	3271.4	Buy
17-05-2021	3069.75	Sell
14-06-2021	3276.35	Buy
30-07-2021	3167.45	Sell
16-08-2021	3472.95	Buy
25-10-2021	3492.95	Sell
20-12-2021	3556.9	Buy
21-02-2022	3719.4	Sell
06-04-2022	3755.35	Buy
29-04-2022	3546.7	Sell
16-08-2022	3392.7	Buy
14-09-2022	3120.4	Sell
02-11-2022	3241.7	Buy
02-01-2023	3261.45	Sell
25-01-2023	3429.75	Buy
15-03-2023	3198.9	Sell
18-05-2023	3199.85	Buy
28-06-2023	3197.35	Sell
14-07-2023	3514.65	Buy

Fig. 32 Tabular output after applying the golden crossover analysis algorithm

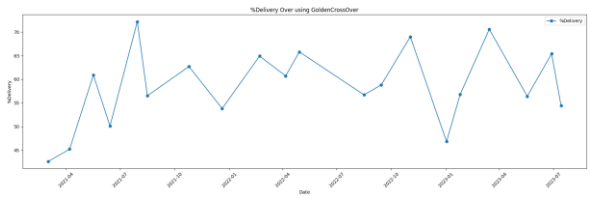


Fig.33 Graphical representation of the above tabular output

Below image Fig.34 is the graphical representation of both algorithms' outcomes: %Delivery analysis and Golden Crossover Signal Analysis, combined.

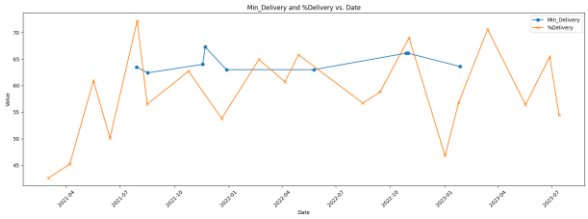


Fig.34 Combined graphical representation of the output of both algorithms: %delivery analysis and golden crossover signal analysis

V. Conclusion

In conclusion, the comprehensive stock market analysis conducted in this report offers valuable insights into the dynamics of the market. By gathering data from multiple sources and meticulously applying pre-processing techniques, we ensured the reliability and accuracy of our findings.

The utilization of various analytical methods, including Bollinger bands analysis, golden crossover assessment, and percentage delivery analysis, allowed us to delve deep into the market trends and identify potential opportunities for investors. Through these analyses, we gained a nuanced understanding of market movements, volatility, and investor sentiment.

Moreover, the combined analysis of golden crossover signals and percentage delivery provided a holistic view of market conditions, enabling us to make informed decisions and recommendations. By integrating these diverse perspectives, we enhanced the robustness of our insights and highlighted actionable strategies for traders and investors alike.

Overall, this report underscores the importance of employing a multifaceted approach to stock market analysis, leveraging both quantitative techniques and qualitative insights. By staying vigilant and adaptive to market dynamics, investors can navigate uncertainty more effectively and capitalize on emerging opportunities.

VI. Future Scope

The future scope based on the findings of this analysis holds significant potential for further exploration and application. Advanced analytical techniques, such as machine learning algorithms and sentiment analysis, could enhance market predictions, while real-time data integration and sector-specific analysis may provide timely insights into evolving market conditions and tailored investment strategies. Developing sophisticated risk assessment models and incorporating insights from behavioral finance could offer comprehensive risk management approaches and a deeper understanding of investor behavior. Additionally, integrating alternative data sources and expanding the scope of analysis to global markets could provide novel insights and opportunities for portfolio diversification. Embracing these avenues for future development could enhance the sophistication and effectiveness of stock market analysis, empowering investors to make more informed decisions and achieve their financial objectives.

VII. Acknowledgement

This work is ostensibly supported by the Dharmsinh Desai University and research opportunities under the Information Technology program. To be honest, words cannot express our gratitude to **Dr. Vipul K. Dabhi,** the head of the Information Technology department, for his invaluable patience and his feedback. We also could not have undertaken this journey without our defense committee, who generously provided knowledge and expertise and their continuous feedback and scholarly input have significantly contributed to this journey of research.

I am equally grateful to my classmates and cohort members, whose incredible support has been a source of strength and inspiration throughout this unforgettable journey. Special thanks are due to my dear mates, **Mr. Neel Kachaddiya**, and **Mr. Vasu Goti** for their invaluable editing assistance, late-night feedback sessions, and moral support.

Lastly, we would be remiss in not mentioning, **our family**, especially **our parents**, **sisters**, and **almighty god**. Their belief in me has kept my spirits and motivation high during this entire process. Last but certainly not least, we want to express our deepest gratitude to each other. Together, we have overcome challenges, celebrated milestones, and forged a bond that extends beyond the confines of academia.

References

[1] "Britannica Money," [Online]. Available: https://www.britannica.com/money/bollinger-bands-indicator. [Accessed 27 04 2024].

[2] "DAILYFX," IG, [Online]. Available: https://www.dailyfx.com/education/moving-averages/golden-cross.html. [Accessed 25 04 2024].

[3] "IncredibleCharts," [Online]. Available: https://www.incrediblecharts.com/candlestick_patterns/candlestick-patterns.php. [Accessed 23 04 2024].

[4] Pang-Ning Tan, Micheal Steinbach and Vipin Kumar, Introduction to Data Mining, Pearson India Education Service Pvt. Ltd..

[5] T. W. Khairi, R. M. Zaki and W. A. Mahmood, "Stock Price Prediction using Technical, Fundamental and News based Approach," in *Scientific Conference of Computer Sciences (SCCS)*, 2019.

[6] Y.-S. Tsai, C.-P. Chang and S.-W. Tzang, "The Impact of Golden Cross and Death Cross Frequency on Stock Returns in Pre- and Post-financial Crisis," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2018.

[7] J. Bollinger, "Using Bollinger Bands," *Stocks & Commodities,* vol. 10, no. 2, pp. 47-51.

[8] Chaimaa Lotfi, Swetha Srinivasan, Myriam Ertz, and Imen Latrous, "Web Scraping Techniques and Applications: A Literature Review," in *SCRS CONFERENCE PROCEEDINGS ON INTELLIGENT SYSTEMS*, 2021, pp. 381-394.