INTENSHIP REPORT ON
UNIFIED MENTOR PRIVATE LIMITED

A Intership Report Submitted
For the partial fulfillment of the requirements for the award of

B.TECH
IN
COMPUTER SCIENCE ENGINEERING

Submitted by

SAHIL KUMAR SINHA- (001CSL23GT003)

SESSION (2023-2026)

R.K.D.F UNIVERSTY
RANCHI
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

R.K.D.F UNIVERSITY RANCHI

CERTIFICATE

This is to certify that the project report "**Intership Report" On Unified Mentor Private Limited**" is a bonafide work of **SAHIL KUMAR SINHA (001CSL23GT003)** who carried out authentic project work under supervision and guidance of guide.

This is to further certify to the best of my knowledge that this project has not been carried out earlier in this University.

To the best of my knowledge, the matter embodied in this project has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Date:

{Signature of HOD}
Shubhangni Dey
Head of the Department
Department of computer science engineering
R.K.D.F University
Ranchi

{Signature Of Supervisor}
Abhishek kr. Singh
Supervisor
Department of Computer science Engineering
R.K.D.F University
Ranchi

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

The Intenship Training in itself is an acknowledged to the inspiration, drive, technical assistance contributed to it by many individuals. This training work would have never been completed without the guidance and assistance that I received from time to time from Internship during the whole training process.

I express my sincere gratitude and indebtedness **to Dr. AMIT KUMAR PANDEY (Registrar)** and **SHUBHAGNI DEY (Course Coordinator,Department of Computer Science Engineering), R.K.D.F University, Ranchi** for giving me an opportunity to enhance my skill in the field of Computer science Technology.

Last but not the least we also thank all my friends and other people who provided us with an atmosphere conductive to optimum learning during this project

SAHIL KUMAR SINHA-(001CSL23GT003)

CONTENTS

INTRODUCTION OF TCS Stock Data

Tata Consultancy Services (TCS) is an Indian multinational information technology (IT) services and consulting company headquartered in Mumbai, Maharashtra, India with its largest campus located in Chennai, Tamil Nadu, India. As of February 2021, TCS is the largest IT services company in the world by market capitalisation ($200 billion). It is a subsidiary of the Tata Group and operates in 149 locations across 46 countries.

TCS is the second largest Indian company by market capitalisation and is among the most valuable IT services brands worldwide.In 2015, TCS was ranked 64th overall in the Forbes World's Most Innovative Companies ranking, making it both the highest-ranked IT services company and the top Indian company. As of 2018, it is ranked eleventh on the Fortune India 500 list.In April 2018, TCS became the first Indian IT company to reach $100 billion in market capitalisation and second Indian company ever (after Reliance Industries achieved it in 2007) after its market capitalisation stood at ₹6.793 trillion (equivalent to ₹7.3 trillion or US$100 billion in 2019) on the Bombay Stock Exchange.

In 2016–2017, parent company Tata Sons owned 72.05% of TCS and more than 70% of Tata Sons' dividends were generated by TCS. In March 2018, Tata Sons decided to sell stocks of TCS worth $1.25 billion in a bulk deal.As of 15 September 2021, TCS has recorded a market capitalisation of US$200 billion, making it the first Indian IT firm to do so.

Fig:-Tcs Stock Data

OBJECTIVE

Analyze the historical data of TCS stock to gain insights into stock behavior, identify trends, and forecast future stock prices.

Dataset Columns Explanation

- Date- Date of trading data.
- Open- Opening stock price on that day.
- High- Highest stock price of the day.
- Low- Lowest stock price of the day.
- Close- Closing stock price of the day.
- Volume- Number of shares traded.
- Dividends- Dividends paid on the stock.
- Stock Splits- Number of stock splits.

Explanation Summary

This project covers EDA, visualization, feature engineering, and prediction modeling for TCS stock prices:

- EDA provides insights into the stock's historical patterns.
- Moving Averages help smooth out price trends.
- Linear Regression is used to predict closing prices.
- Evaluation metrics help validate the model’s accuracy, giving insight into its reliability

Import Required Libraries

```
# import python libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns
```

Load the Dataset

```
df = pd.read_csv('TCS_stock_info.csv', encoding = 'unicode_escape')
```

```
df.shape
```

OUTPUT

(150, 2)

```
df.head()
```

OUTPUT

	zip	400001
0	sector	Technology
1	fullTimeEmployees	509058
2	longBusinessSummary	Tata Consultancy Services Limited provides inf...
3	city	Mumbai
4	phone	91 22 6778 9999

df.head(10)

OUTPUT

	zip	400001
0	sector	Technology
1	fullTimeEmployees	509058
2	longBusinessSummary	Tata Consultancy Services Limited provides inf...
3	city	Mumbai
4	phone	91 22 6778 9999
5	country	India
6	companyOfficers	[]
7	website	http://www.tcs.com
8	maxAge	1
9	address1	TCS House

df.info()

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   zip     150 non-null    object
 1   400001  108 non-null    object
dtypes: object(2)
memory usage: 2.5+ KB
```

pd.isnull(df).sum()

OUTPUT

```
zip       0
400001    42
dtype: int64
```

 # describe() method returns description of the data in the DataFrame (i.e. count, mean, std, etc)

df.describe()

OUTPUT

	zip	400001
count	150	108
unique	150	96
top	sector	3805
freq	1	4

print(df.columns)

OUTPUT

```
Index(['zip', '400001'], dtype='object')
```

Data Preprocessing

- Check for null values and handle them.
- Convert necessary columns to numeric if needed.
- Check for any outliers in the data, especially in Volume and Close price.

```
import pandas as pd

# Load the data

# Check actual column names
print("Columns:", df.columns.tolist())

# Check for null values
print(df.isnull().sum())

# Convert numeric columns if required
df['Open'] = pd.to_numeric(df['Open'], errors='coerce')
df['High'] = pd.to_numeric(df['High'], errors='coerce')
df['Low'] = pd.to_numeric(df['Low'], errors='coerce')
df['Close'] = pd.to_numeric(df['Close'], errors='coerce')

# Fill any remaining NaN values
df.fillna(method='ffill', inplace=True)
```

OUTPUT

```
Columns: ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Dividends', 'Stock Splits']
Date            0
Open            0
High            0
Low             0
Close           0
Volume          0
Dividends       0
Stock Splits    0
dtype: int64
```

```
plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Close'], label='Close Price', color='blue')
plt.title('TCS Closing Stock Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price (INR)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

OUTPUT

From Above Graph we can see that Stock gives good return to the investors in a long period of time.

Exploratory Data Analysis (EDA)

- Price Trends: Visualize the Open, Close, High, and Low prices overtime.
- Volume Analysis: Analyze trading volumes.
- Moving Averages: Calculate moving averages for trend analysis.

```
df = pd.read_csv('TCS_stock_history.csv', encoding = 'unicode_escape')

df.shape

df.head(10)
```

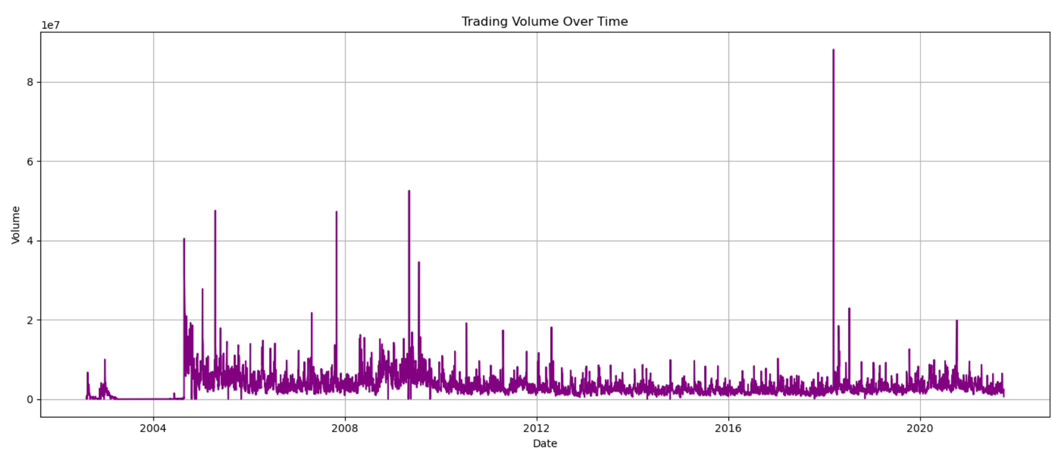
OUTPUT

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-08-12	28.794172	29.742206	28.794172	29.519140	212976	0.0	0.0
1	2002-08-13	29.556316	30.030333	28.905705	29.119476	153576	0.0	0.0
2	2002-08-14	29.184536	29.184536	26.563503	27.111877	822776	0.0	0.0
3	2002-08-15	27.111877	27.111877	27.111877	27.111877	0	0.0	0.0
4	2002-08-16	26.972458	28.255089	26.582090	27.046812	811856	0.0	0.0
5	2002-08-19	27.269876	27.269876	26.126661	26.377609	205880	0.0	0.0
6	2002-08-20	26.563503	28.794168	26.386910	27.111877	3773624	0.0	0.0
7	2002-08-21	28.608262	29.147341	27.158333	28.440964	3011064	0.0	0.0
8	2002-08-22	29.379720	30.913303	29.231009	29.667849	6732480	0.0	0.0
9	2002-08-23	29.928077	32.437575	29.565595	31.452364	4841672	0.0	0.0

```
df = pd.read_csv('TCS_stock_history.csv',parse_dates=['Date'], encoding = 'unicode_escape')

df.sort_values('Date', inplace=True)


# Show basic info

#print(df.info())

#print(df.head())
```

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


# Plot trading volume over time

plt.figure(figsize=(14, 6))

sns.lineplot(data=df, x='Date', y='Volume', color='purple')

plt.title('Trading Volume Over Time')

plt.xlabel('Date')

plt.ylabel('Volume')

plt.grid(True)

plt.tight_layout()

plt.show()
```

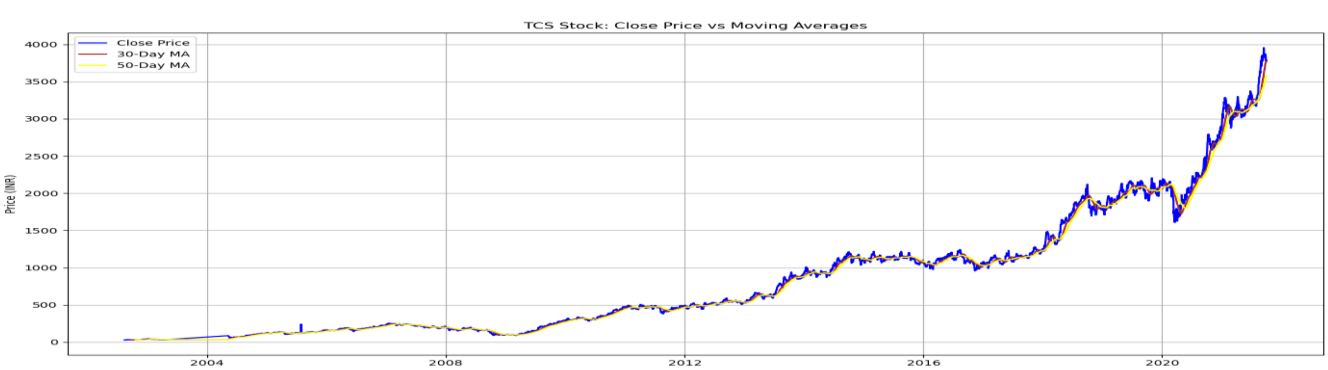
<u>OUTPUT</u>



From Above graph we can easily understand that Stock has high trading volume, which gives market sentiment and liquidity.

```python
import pandas as pd

import matplotlib.pyplot as plt

# Ensure Date is datetime and sorted

df['Date'] = pd.to_datetime(df['Date'])

df = df.sort_values('Date')

# Calculate moving averages

df['30-Day MA'] = df['Close'].rolling(window=30).mean()

df['50-Day MA'] = df['Close'].rolling(window=50).mean()

# Plotting

plt.figure(figsize=(14, 7))

plt.plot(df['Date'], df['Close'], label='Close Price', color='blue')

plt.plot(df['Date'], df['30-Day MA'], label='30-Day MA', color='brown')

plt.plot(df['Date'], df['50-Day MA'], label='50-Day MA', color='yellow')

plt.title('TCS Stock: Close Price vs Moving Averages')

plt.xlabel('Date')

plt.ylabel('Price (INR)')

plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()
```

<u>OUTPUT</u>



From Above Graph we can easily understand The consistent alignment of the moving averages with the upward movement of the stock indicates strong price momentum and supports long-term bullish sentiment in TCS stock.

Feature Engineering

● Extract features like Year, Month, Day, Day of Week from Date.
● Create lag features (e.g., previous day's close, previous day's high/low).

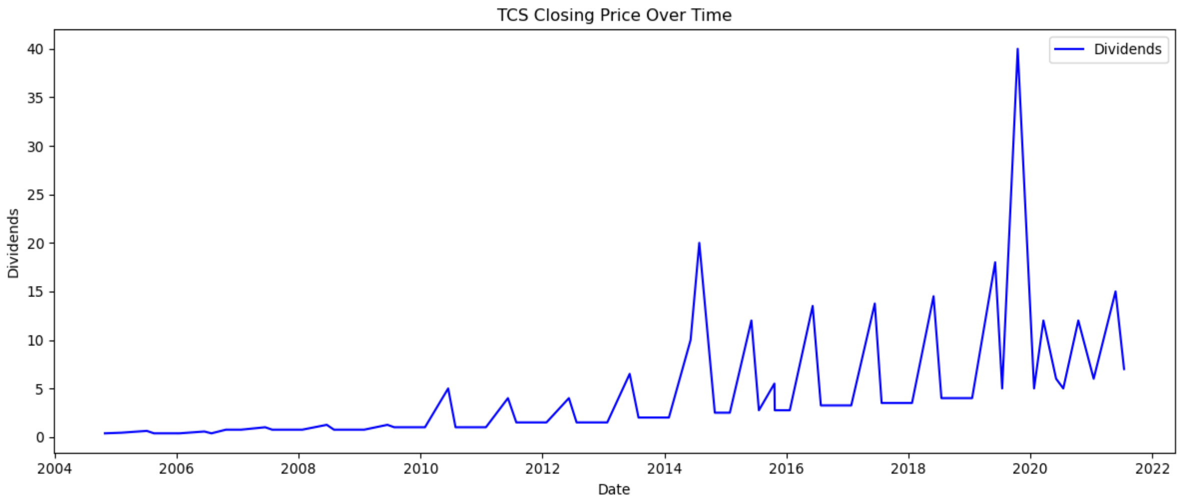
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load data
df.columns = df.columns.str.strip()  # Remove extra spaces
print("□ First 5 rows:")
print(df.head())

# Basic info
print("\n□ Data Info:")
print(df.info())

# Missing values
print("\n□ Missing Values:")
print(df.isnull().sum())

# Descriptive statistics
print("\n□ Summary Statistics:")
print(df.describe())

# Convert date column
df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values('Date')

# Plot Dividends over time
plt.figure(figsize=(12, 5))
plt.plot(df['Date'], df['Dividends'], label='Dividends', color='blue')
plt.title('TCS Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Dividends')
plt.legend()
plt.tight_layout()
plt.show()
```

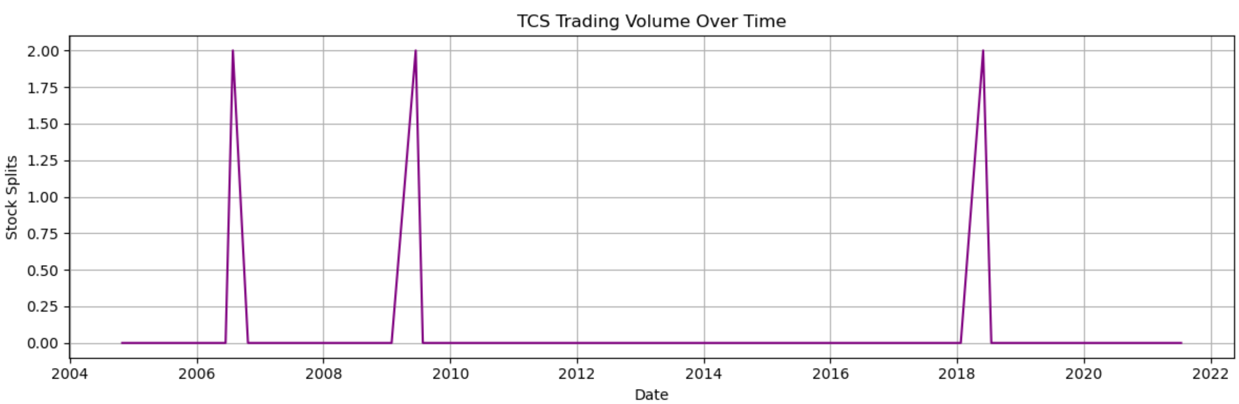
```
# Plot Stock Splits over time
plt.figure(figsize=(12, 4))
plt.plot(df['Date'], df['Stock Splits'], color='purple')
plt.title('TCS Trading Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Stock Splits')
plt.grid(True)
plt.tight_layout()
plt.show()
```

<u>OUTPUT</u>

```
🔹 First 5 rows:
         Date  Dividends  Stock Splits
0  2004-10-28     0.3750           0.0
1  2005-02-03     0.4375           0.0
2  2005-07-06     0.6250           0.0
3  2005-08-18     0.3750           0.0
4  2005-10-18     0.3750           0.0

🔹 Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Date          70 non-null     object 
 1   Dividends     70 non-null     float64
 2   Stock Splits  70 non-null     float64
dtypes: float64(2), object(1)
memory usage: 1.8+ KB
None
```

```
🔹 Missing Values:
Date            0
Dividends       0
Stock Splits    0
dtype: int64

🔹 Summary Statistics:
       Dividends  Stock Splits
count  70.000000     70.000000
mean    4.560714      0.085714
std     6.284794      0.407995
min     0.375000      0.000000
25%     1.000000      0.000000
50%     2.500000      0.000000
75%     5.000000      0.000000
max    40.000000      2.000000
```


TCS Closing Price Over Time
Dividends
Dividends
Date
0
5
10
15
20
25
30
35
40
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022


TCS Trading Volume Over Time
Stock Splits
Date
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022

Model Building and Prediction

- Use Linear Regression to predict the Close price based on features.
- Train/Test Split for model evaluation.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# (your existing data loading and feature creation code)

# Create lag features
df['Close_Lag1'] = df['Close'].shift(1)
df['Close_Lag2'] = df['Close'].shift(2)
df['Close_Lag3'] = df['Close'].shift(3)

# Drop rows with NaN values from lagging
df_ml = df.dropna()

# Features (X) and Target (y)
X = df_ml[['Close_Lag1', 'Close_Lag2', 'Close_Lag3']]
y = df_ml['Close']

from sklearn.model_selection import train_test_split

# Use 80% for training, 20% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

import numpy as np

lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predict
y_pred_lr = lr_model.predict(X_test)

# Evaluation
print("Linear Regression:")
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))
print("R² Score:", r2_score(y_test, y_pred_lr))
```

OUTPUT

```
Linear Regression:
RMSE: 36.89626262276406
R² Score: 0.9961165568926493
```

Linear Regression is performing surprisingly well with just lag-based features. The extremely high R^2 suggests the stock has a strong short-term autocorrelation

Visualize Model Performance

- Plot predicted vs. actual values.
- Scatter plot to observe prediction accuracy.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
import xgboost as xgb

# Load data
df.columns = df.columns.str.strip()
df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values('Date')

# Ensure numeric columns
for col in ['Open', 'High', 'Low', 'Close']:
    df[col] = pd.to_numeric(df[col], errors='coerce')
df.fillna(method='ffill', inplace=True)

# Feature engineering
df['Close_Lag1'] = df['Close'].shift(1)
df['Close_Lag2'] = df['Close'].shift(2)
df['Close_Lag3'] = df['Close'].shift(3)
df_ml = df.dropna()

# Prepare features and target
X = df_ml[['Close_Lag1', 'Close_Lag2', 'Close_Lag3']]
y = df_ml['Close']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)

# XGBoost
xgb_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100)
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
```

```
# Evaluation
print("Linear Regression RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))
print("XGBoost RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_xgb)))

# Plot
plt.figure(figsize=(14, 6))
plt.plot(y_test.index, y_test.values, label='Actual Close', color='blue')
plt.plot(y_test.index, y_pred_lr, label='Linear Regression', linestyle='--')
plt.plot(y_test.index, y_pred_xgb, label='XGBoost', linestyle='--', color='green')
plt.title('Actual vs Predicted Closing Prices')
plt.xlabel('Index')
plt.ylabel('Close Price')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

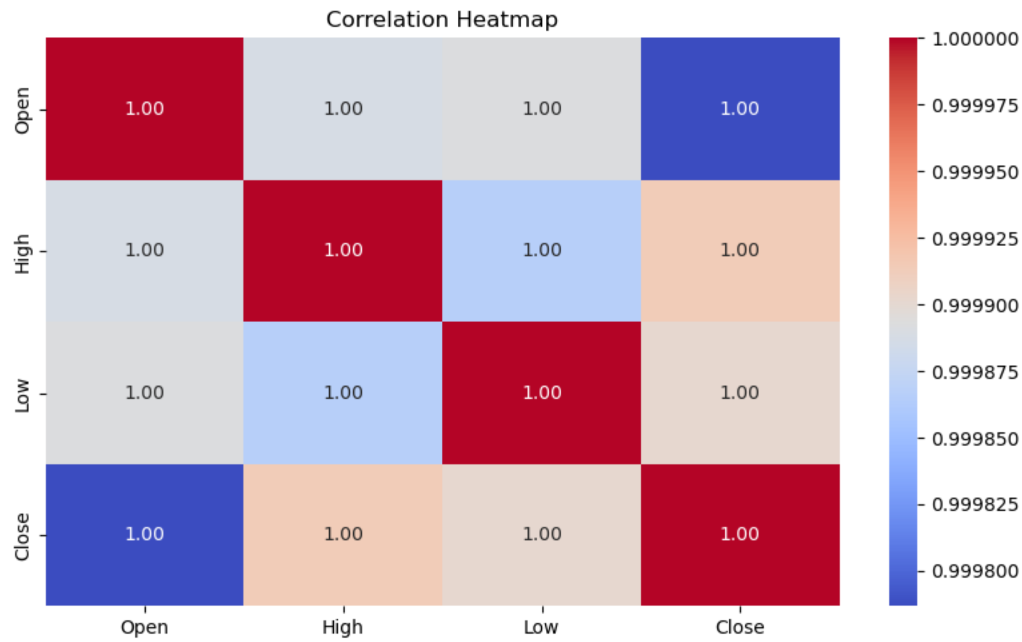
<u>OUTPUT</u>

```
Linear Regression RMSE: 36.7595558130353
XGBoost RMSE: 1021.6186858780651
```

From Above graph we can see that TCS has outperformed in the market

```
plt.figure(figsize=(8, 5))
sns.heatmap(df[['Open', 'High', 'Low', 'Close']].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()
```

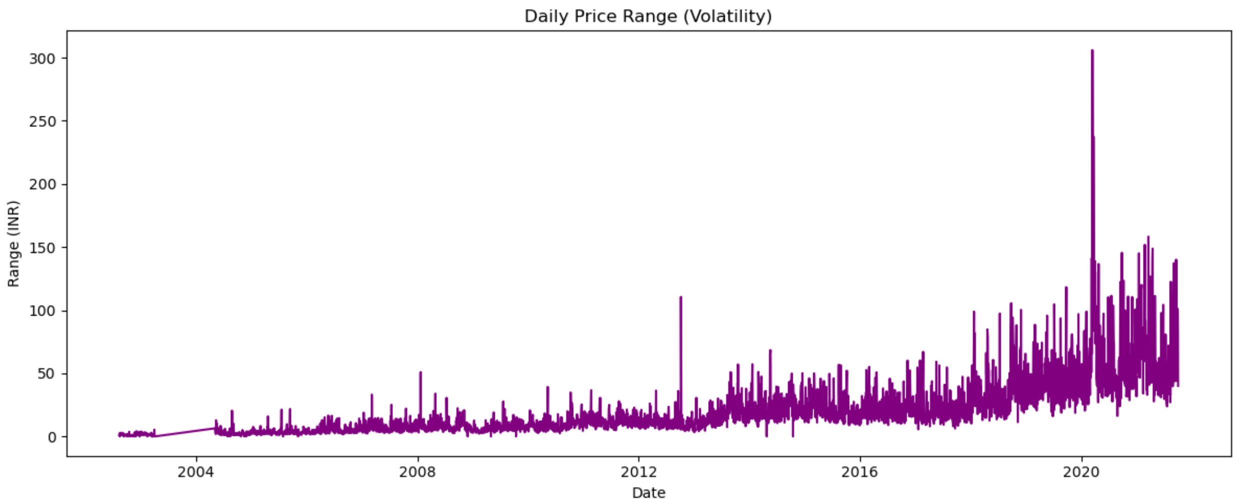
OUTPUT



The heatmap confirms that the stock's daily pricing components are tightly linked, which is typical in time-series stock data.

```
df['Range'] = df['High'] - df['Low']
plt.figure(figsize=(12, 5))
plt.plot(df['Date'], df['Range'], color='purple')
plt.title('Daily Price Range (Volatility)')
plt.xlabel('Date')
plt.ylabel('Range (INR)')
plt.tight_layout()
plt.show()
```

OUTPUT


Daily Price Range (Volatility)
Range (INR)
300
250
200
150
100
50
0
2004
2008
2012
2016
2020
Date

CONCLUSION

TCS stock has shown steady long-term growth with upward momentum, with closing prices more frequent at lower levels due to stock splits. The distribution is right-skewed, Predictive models like XGBoost perform well, confirming trend stability. Price distribution and volume patterns support consistent investor interest.