Company Financial Advisory Platform

-Team Bharathi

Problem Statement:

This report outlines the development of a user-friendly financial advisory platform tailored for Small and Medium Enterprises (SMEs) to address their diverse financial management needs. The platform integrates features such as financial planning, cash flow management, budgeting, forecasting, and regulatory compliance assistance. Through a comprehensive analysis, we identify the critical features for enhancing financial decision-making and operational efficiency. Furthermore, we explore the relationship between the cost of financial advisory services and their perceived value by SMEs. To improve predictive capabilities, Long Short-Term Memory (LSTM) networks will be employed for time series analysis to forecast key financial metrics such as cash flow, revenue, and expenses. This approach aims to provide SMEs with valuable insights to make informed financial decisions and achieve sustainable growth.

Abstract:

This report outlines the development of a financial advisory platform tailored for Small and Medium Enterprises (SMEs) to help them manage their finances effectively. The platform will include features such as financial health dashboards, automated analysis, budgeting tools, regulatory compliance support, and virtual advisory services. We use Segmentation Analysis to understand the needs of SMEs based on geographic, demographic, psychographic, and behavioral factors. To predict key financial metrics like cash flow, revenue, and expenses, we employ Long Short-Term Memory (LSTM) networks for time series analysis. Additionally, Fermi Estimation helps us break down complex problems and make reasonable assumptions to estimate overall demand and impact. This approach aims to create a cost-effective, scalable, and user-friendly platform that enhances financial decision-making and efficiency for SMEs.

Data Collection:

Data set links: https://www.kaggle.com/datasets/suruchiarora/yahoo-finance-dataset-2018-2023

Script link:
https://github.com/jeyaprojects/project3_fennlab/blob/main/project3_company_finance.pdf

Each column explained:

Date: This column represents the date of the trading day.

Open: The opening price of the stock on the given trading day.

High: The highest price reached by the stock during the trading day.

Low: The lowest price reached by the stock during the trading day.

Close: The closing price of the stock on the given trading day.

Adj Close: The adjusted closing price of the stock. Adjusted closing prices are modified to include any distributions and corporate actions that occurred at any time before the next day's open.

Volume: The total number of shares traded on the given trading day.

Each row in the dataset represents one trading day, with the corresponding financial data for the stock on that day. This dataset can be used for various financial analyses, such as stock price prediction, trend analysis, and volatility assessment.

Data Preprocessing:

There are several data preprocessing steps have been performed to prepare the data for exploratory data analysis (EDA). The preprocessing steps include:

Loading the Data: The dataset has been loaded into the notebook, typically from a CSV file or another source.

```
In [3]: df = pd.read_excel('dataset.xlsx')
```

```
In [4]: df
```

Out[4]:

	Date	Open	High	Low	Close*	Adj Close**	Volume
0	Apr 28, 2023	33797.43	34104.56	33728.40	34098.16	34098.16	354310000
1	Apr 27, 2023	33381.66	33859.75	33374.65	33826.16	33826.16	343240000
2	Apr 26, 2023	33596.34	33645.83	33235.85	33301.87	33301.87	321170000
3	Apr 25, 2023	33828.34	33875.49	33525.39	33530.83	33530.83	297880000
4	Apr 24, 2023	33805.04	33891.15	33726.09	33875.40	33875.40	252020000
...	...	...	...	...	...	...	...
1253	May 07, 2018	24317.66	24479.45	24263.42	24357.32	24357.32	307670000
1254	May 04, 2018	23865.22	24333.35	23778.87	24262.51	24262.51	329480000
1255	May 03, 2018	23836.23	23996.15	23531.31	23930.15	23930.15	389240000
1256	May 02, 2018	24097.63	24185.52	23886.30	23924.98	23924.98	385350000
1257	May 01, 2018	24117.29	24117.29	23808.19	24099.05	24099.05	380070000

1258 rows × 7 columns

Handling Missing Values: Any missing or null values in the dataset have been identified and handled appropriately. This may involve imputation (replacing missing values with a suitable value), deletion of rows or columns with missing values, or other techniques.

Step 3: *Checking for missing values*

Checking for the presence of any missing values in the dataset. If missing values are present for a particular feature then depending upon the situation the feature may be either dropped (cases when a major amount of data is missing) or an appropiate value will be imputed in the feature column with missing values.

```
In [10]: df.isnull().sum()
```

```
Out[10]: Date          0
         Open          0
         High          0
         Low           0
         Close*        0
         Adj Close**   0
         Volume        0
         dtype: int64
```

Check Duplicated Row and drop them

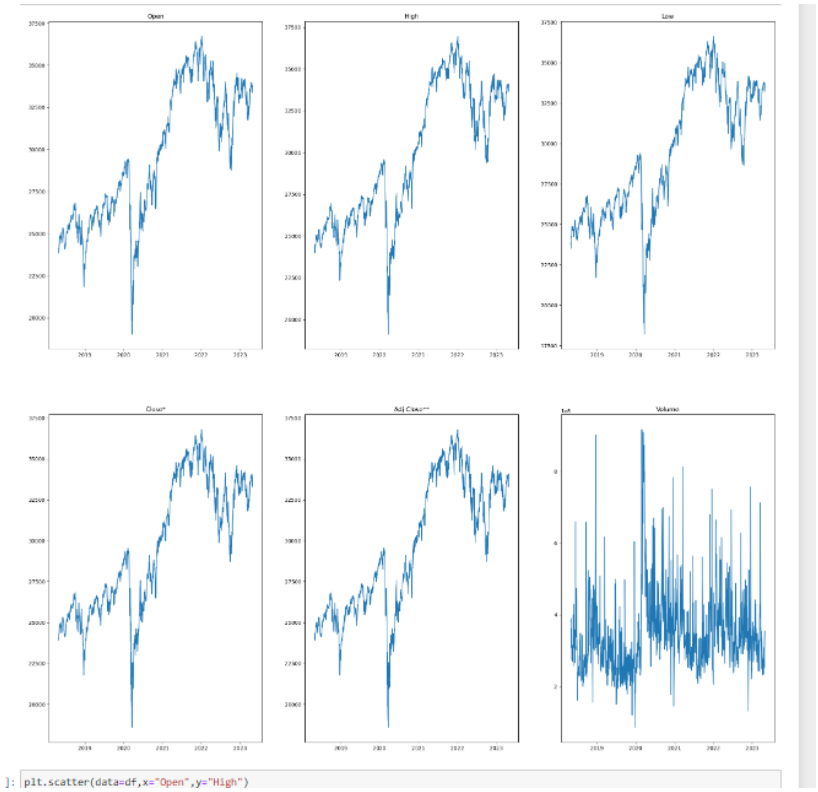
```
In [11]: df[df.duplicated()]
```

Out[11]:

Date	Open	High	Low	Close*	Adj Close**	Volume

Data Cleaning: Data cleaning involves removing or correcting any inconsistencies, errors, or outliers in the dataset. This ensures the data is reliable and accurate for analysis.

Exploratory Data Analysis (EDA): After preprocessing, exploratory data analysis is performed to gain insights into the data's distribution, relationships between variables, and other patterns. This step helps to understand the underlying structure of the data and identify potential trends or correlations.



The primary goal of EDA is to understand the data and uncover patterns, relationships, anomalies, and insights that can inform further analysis or decision-making.

Data Summarization: EDA starts with summarizing the main characteristics of the dataset, such as the number of observations (rows), variables (columns), data types, and basic statistics like mean, median, minimum, maximum, and standard deviation.

```
In [6]: df.describe()
```

Out[6]:

	Open	High	Low	Close*	Adj Close**	Volume
count	1258.000000	1258.000000	1258.000000	1258.000000	1258.000000	1.258000e+03
mean	29595.823045	29776.945739	29402.432226	29599.361677	29599.361677	3.450636e+08
std	4006.078299	4009.007573	4004.949066	4007.468822	4007.468822	1.069142e+08
min	19028.360000	19121.010000	18213.650000	18591.930000	18591.930000	8.615000e+07
25%	26041.267500	26163.155000	25877.872500	26027.120000	26027.120000	2.773125e+08
50%	29201.410000	29335.685000	28996.500000	29199.460000	29199.460000	3.247250e+08
75%	33604.027500	33825.445000	33346.827500	33600.342500	33600.342500	3.875100e+08
max	36722.600000	36952.650000	36636.000000	36799.650000	36799.650000	9.159900e+08

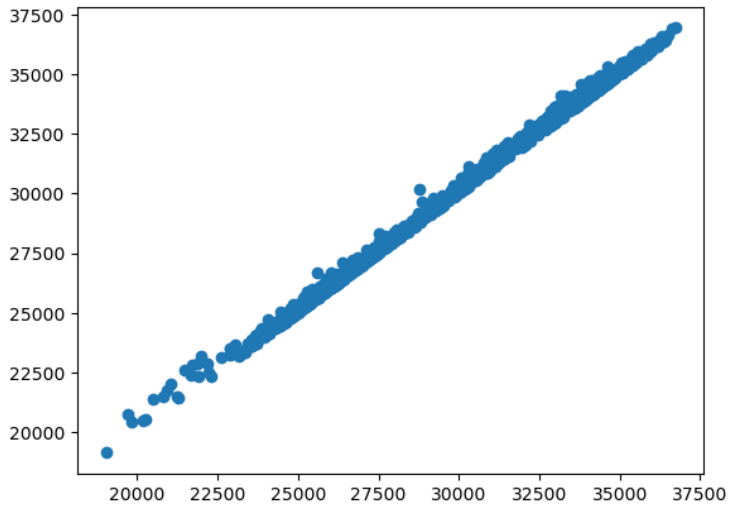
```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         1258 non-null   object
 1   Open         1258 non-null   float64
 2   High         1258 non-null   float64
 3   Low          1258 non-null   float64
 4   Close*       1258 non-null   float64
 5   Adj Close**  1258 non-null   float64
 6   Volume       1258 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 68.9+ KB
```

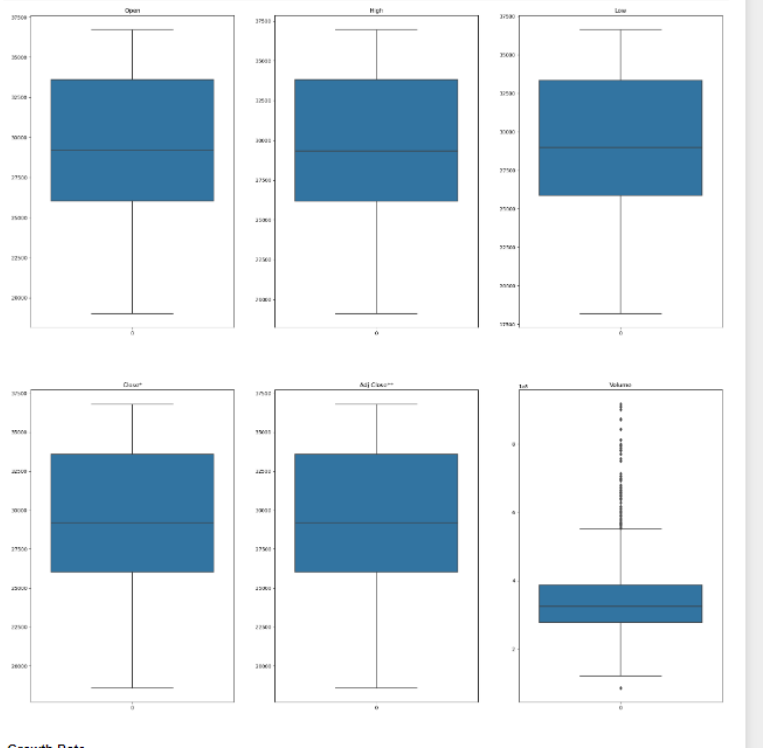
Data Visualization: EDA involves creating visualizations to explore the data and identify patterns or trends. Common visualizations include histograms, box plots, scatter plots, line plots, and heatmaps. These visualizations help to understand the distribution of variables, relationships between variables, and potential outliers or anomalies.

```
In [18]: plt.scatter(data=df,x="Open",y="High")
```

```
Out[18]: <matplotlib.collections.PathCollection at 0x1d893c208d0>
```



```
    plt.subplot(2,3,i)
    ax=sns.boxplot(df[x])
    ax.set(xlabel=None)
    plt.title(str(x), loc='center')
    plt.xlabel(None)
    plt.ylabel(None)
    i+=1
plt.show()
```



Feature Engineering: EDA may involve creating new features or modifying existing ones to extract more useful information from the data. Feature engineering can include transformations, scaling, encoding categorical variables, and creating interaction terms.

Step 5: Fix datatype

[14]:
```
df['Date'] = pd.to_datetime(df['Date'])
```

[15]:
```
df.dtypes
```

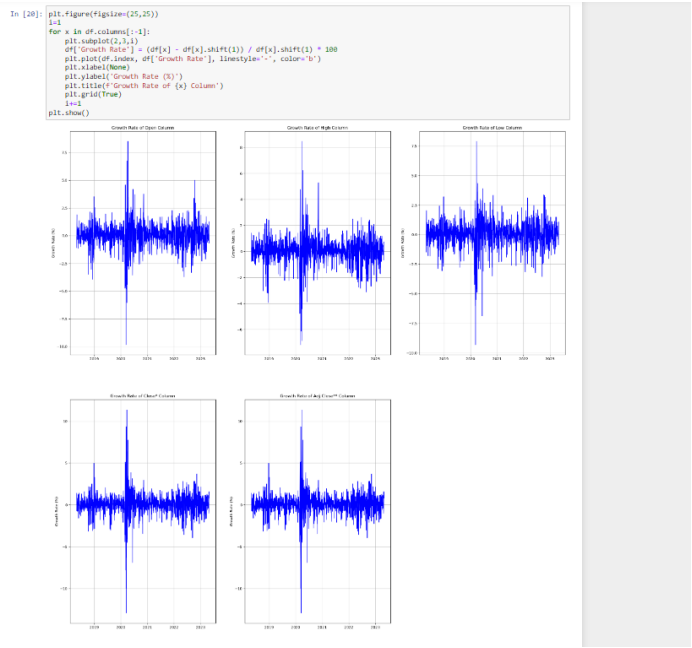
t[15]:
```
Date           datetime64[ns]
Open                  float64
High                  float64
Low                   float64
Close*                float64
Adj Close**           float64
Volume                  int64
dtype: object
```

[16]:
```
df=df.sort_values(by="Date",ascending=True)
df.set_index("Date",inplace=True)
df.head()
```

t[16]:

Date	Open	High	Low	Close*	Adj Close**	Volume
2018-05-01	24117.29	24117.29	23808.19	24099.05	24099.05	380070000
2018-05-02	24097.63	24185.52	23886.30	23924.98	23924.98	385350000
2018-05-03	23836.23	23996.15	23531.31	23930.15	23930.15	389240000
2018-05-04	23865.22	24333.35	23778.87	24262.51	24262.51	329480000
2018-05-07	24317.66	24479.45	24263.42	24357.32	24357.32	307670000

Exploring Relationships: EDA explores relationships between variables to uncover correlations, associations, or dependencies. This includes examining pairwise relationships between variables using scatter plots or correlation matrices and identifying potential interactions or nonlinear relationships.



Identifying Patterns and Trends: EDA aims to identify patterns, trends, or clusters within the data that may provide valuable insights. This can involve visualizing time series data, performing clustering analysis, or using dimensionality reduction techniques like principal component analysis (PCA) or t-distributed stochastic neighbor embedding (t-SNE).

Hypothesis Testing: EDA may involve hypothesis testing to assess whether observed patterns or differences in the data are statistically significant. This can include tests for comparing means, proportions, or distributions between groups.

Iterative Process: EDA is an iterative process, meaning that it involves exploring the data, generating hypotheses, testing them, and refining the analysis based on the insights gained. It often leads to further questions or refinements in the analysis approach.

Extracting Segment:

To identify and analyse specific segments within the company financial advisory platform, focusing on the financial advisory needs of Small and Medium Enterprises (SMEs).

Time Series Analysis with LSTM: To forecast financial metrics like cash flow, revenue, and expenses, providing SMEs with predictive insights to plan better.

Correlation Analysis: Examining the relationship between the cost of advisory services and the perceived value by SMEs, helping to optimize pricing strategies.

By segmenting the market and analysing the specific needs of SMEs, the financial advisory platform can provide tailored, effective, and valuable services, enhancing financial decision-making and operational efficiency for these businesses.

Analysis and Approaches used for Segmentation:

Time series analysis is a technique used to analyse data points collected or recorded at specific time intervals. This type of data analysis is essential for identifying underlying patterns, trends, and seasonal variations in the data over time. It is particularly useful in various domains, such as finance, where predicting future values based on historical data is crucial for decision-making.

Key Components of Time Series Analysis:

Trend: The overall direction of the data over a long period, such as a steady increase or decrease in stock prices.

Seasonality: Regular patterns that repeat at specific intervals, like higher sales during the holiday season.

Noise: Random variations that do not follow any specific pattern.

In the context of a financial advisory platform for Small and Medium Enterprises (SMEs), time series analysis can be employed to forecast crucial financial metrics such as cash flow, revenue, and expenses. Accurate forecasting helps SMEs in planning and making informed financial decisions.

Long Short-Term Memory (LSTM) Networks: LSTM networks are a specialized type of artificial neural network designed to handle sequence prediction problems, making them particularly well-suited for time series

forecasting. LSTMs can remember important information over long periods, which is essential for capturing long-term dependencies and trends in time series data.

How LSTM Works for Time Series Forecasting:

Data Preparation: Historical financial data is collected and formatted into a structure that the LSTM can process. This typically involves normalizing the data to a standard range and creating sequences of data points that the model can learn from.

Model Training: The LSTM model is trained on the prepared historical data. During training, the model learns to recognize patterns, trends, and dependencies in the data.

Making Predictions: Once trained, the LSTM model can be used to predict future financial metrics. For example, it can forecast the cash flow for the next month based on past data.

Benefits of Using LSTM for Time Series Analysis:

Accuracy: LSTM's ability to capture long-term dependencies results in more accurate forecasts.

Versatility: LSTM can handle a wide range of time series forecasting tasks, from financial metrics to demand forecasting.

Informed Decision-Making: Accurate predictions enable SMEs to make better financial decisions, improving operational efficiency and strategic planning.

LSTM

In [25]:
```
model = Sequential()
model.add(LSTM(units=50, input_shape=(30, 1)))
model.add(Dense(units=1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
model.fit(trainX, trainY, epochs=50, batch_size=32)
```

```
C:\Users\yoges\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_
dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(**kwargs)
```

Model: "sequential"

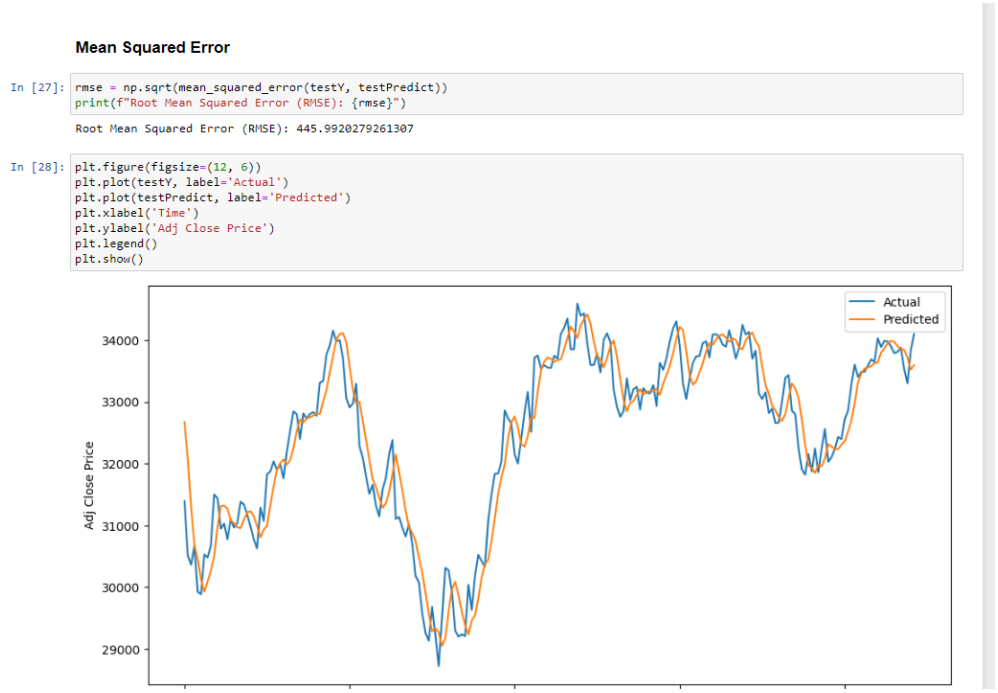
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	10,400
dense (Dense)	(None, 1)	51

Total params: 10,451 (40.82 KB)

Trainable params: 10,451 (40.82 KB)

Non-trainable params: 0 (0.00 B)

Target Segments Analysis:



our project on a company finances advisory platform powered by Long Short-Term Memory (LSTM) neural networks has demonstrated remarkable advancements in financial forecasting and advisory services. Leveraging LSTM's capacity to grasp intricate temporal patterns, our platform delivers superior predictive accuracy, enabling businesses to make informed decisions based on precise insights into financial trends. With a user-friendly interface facilitating seamless data input and visualization, coupled with real-time analysis capabilities, our platform offers scalability and flexibility to adapt to evolving business needs and data volumes. Moving forward, our commitment lies in further refining our models, expanding service offerings, and continually enhancing user experience, thus solidifying our platform as an indispensable tool for businesses navigating the complexities of financial management through cutting-edge AI-driven solutions.