STOCK MARKET PREDICTION

-By:

Ana Singh (Team Leader) ,

Vanshika Rawat (Team Member),

Harish P(Team Member) ,

Piyush Kaushik(Team Member)

Problem Statement:

Many investors and traders struggle to make informed decisions in the stock market due to the complexity and uncertainty surrounding stock price movements. They face challenges in predicting stock prices accurately, staying updated with market trends, and accessing personalized insights to optimize their portfolios. There is a need for a reliable and comprehensive stock price prediction service that offers timely predictions, advanced features, and personalized recommendations to empower users in their investment decision-making process.

Existing solutions lack accuracy and fail to provide a holistic view of the market. Investors are left to rely on traditional methods that often fall short in capturing the intricacies of stock price movements. There is a gap in the market for a subscription-based stock price prediction service that offers different plans with varying features and access levels, along with premium features that enhance the user experience and provide valuable insights.

Additionally, investors often lack access to real-time market updates and customized alerts, hindering their ability to react promptly to market events. They also struggle to find reliable expert analysis and insights to guide their investment strategies effectively. Addressing these challenges requires a robust platform that combines accurate predictions, real-time information, and expert analysis in a user-friendly manner.

Therefore, the problem at hand is to develop a stock price prediction service/app that offers a subscription-based model with different plans, a free trial period, and tiered pricing options. The service should provide accurate predictions, real-time market updates, custom alerts, and expert analysis as premium features. By solving this problem, investors and traders will gain access to reliable predictions, comprehensive market information, and personalized insights to make better-informed investment decisions and optimize their portfolios.

-Step 1: Prototype Selection

In the short- and long-term future, creating a machine learning model for stock price prediction may be a feasible and marketable good or service. An explanation of why it ought to be chosen is provided below:

Feasibility:

Data Availability: Historical information on stock prices, business finances, market trends, and geopolitical events is frequently accessible and can be gathered to train models.

Development of predictive models for stock price prediction has become more practical and successful as a result of advances in machine learning algorithms and methods.

Existing Infrastructure: Machine learning models for stock price prediction can be created more easily using pre-existing frameworks and libraries. The time and effort needed for development are decreased by this infrastructure.

Viability:

Predictive analytics, especially the ability to predict stock prices, are becoming more and more important in the financial sector. In order to make wise decisions and maximise their portfolios, traders and investors look for precise predictions.

Demand Over Time: Over time, it's anticipated that there will still be a need for stock price forecasting. In order to acquire a competitive edge in the market, investors will continue to rely on data-driven insights.

Monetization:

Direct Monetization: A stock price prediction service or app might generate income by charging consumers a monthly fee in exchange for access to precise and timely forecasts. For a price, additional premium features like tailored portfolio advice might be made available.

Market Potential: Because of the size of the financial sector, there is a sizable market for reliable stock price forecasts. Financial institutions, traders, and investors are all eager to pay for trustworthy information that has the potential to boost their returns on investment.

Scalability: The product or service can be easily scaled and deployed to serve a broad user base, maximising its potential for monetization, once it has been built.

Therefore , a machine learning-based stock price prediction service or app for stocks of companies has the potential to be a beneficial and long-lasting solution, taking into account the practicality, viability, and direct monetization options.

-Step 2: Prototype Development

Stock Market Prediction Using the Long Short-Term MemoryMethod

We will use the Long Short-Term Memory(LSTM) method to create a Machine Learning model to forecast Microsoft Corporation stock values. They are used to make minor changes to the information by multiplying and adding. Long-term memory (LSTM) is a deep learning artificial recurrent neural network (RNN) architecture.

Unlike traditional feed-forward neural networks, LSTM has feedback connections. It can handle single data points (such as pictures) as well as full data sequences

Stock Market Prediction Using the Long Short-Term Memory Method

We will use the Long Short-Term Memory(LSTM) method to create a Machine Learning model to forecast Microsoft Corporation stock values. They are used to make minor changes

to the information by multiplying and adding. Long-term memory (LSTM) is a deep learning artificial recurrent neural network (RNN) architecture.

Unlike traditional feed-forward neural networks, LSTM has feedback connections. It can handle single data points (such as pictures) as well as full data sequences

Program Implementation

We will now go to the section where we will utilize Machine Learning techniques in Python to estimate the stock value using the LSTM.

Step 1: Importing the Libraries

As we all know, the first step is to import the libraries required to preprocess Microsoft Corporation stock data and the other libraries required for constructing and visualizing the LSTM model outputs.

```
#Importing the Libraries
import pandas as PD
import NumPy as np
%matplotlib inline
import matplotlib. pyplot as plt
import matplotlib
from sklearn. Preprocessing import MinMaxScaler
from Keras. layers import LSTM, Dense, Dropout
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib. dates as mandates
from sklearn. Preprocessing import MinMaxScaler
from sklearn import linear_model
from Keras. Models import Sequential
from Keras. Layers import Dense
import Keras. Backend as K
from Keras. Callbacks import EarlyStopping
from Keras. Optimisers import Adam
from Keras. Models import load_model
from Keras. Layers import LSTM
from Keras. utils.vis_utils import plot_model
```

Step 2: Getting to Visualising the Stock Market Prediction Data

Using the Pandas Data Reader library, we will upload the stock data from the local system as a Comma Separated Value (.csv) file and save it to a pandas DataFrame. Finally, we will examine the data.

Step 3: Checking for Null Values by Printing the Data Frame Shape

```
#Get the Dataset
df=pd.read_csv("MicrosoftStockData.csv",na_values=['null'],index_col='Date',parse_dates=True,i
df.head()
```

In this step, firstly, we will print the structure of the dataset. We'll then check for null values in the data frame to ensure that there are none. The existence of null values in the dataset causes issues during training since they function as outliers, creating a wide variance in the training process.

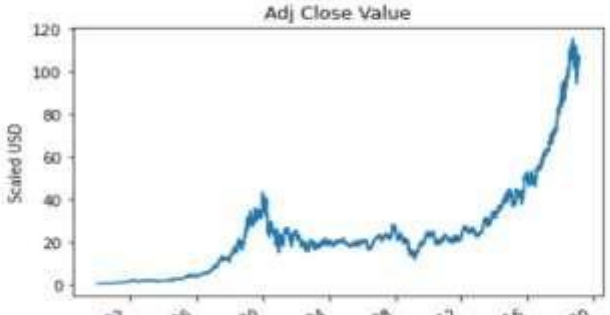
```
#Print the shape of Dataframe  and Check for Null Values
print("Dataframe Shape: ", df. shape)
print("Null Value Present: ", df.IsNull().values.any())
Output:
>> Dataframe Shape: (7334, 6)
>>Null Value Present: False
```

Date	Open	High	Low	Close	Adj Close	Volume
1990-01-02	0.605903	0.616319	0.598090	0.616319	0.447268	53033600
1990-01-03	0.621528	0.626736	0.614583	0.619792	0.449788	113772800
1990-01-04	0.619792	0.638889	0.616319	0.638021	0.463017	125740800
1990-01-05	0.635417	0.638889	0.621528	0.622396	0.451678	69564800
1990-01-08	0.621528	0.631944	0.614583	0.631944	0.458607	58982400

Step 4: Plotting the True Adjusted Close Value

The Adjusted Close Value is the final output value that will be forecasted using the Machine Learning model. This figure indicates the stock's closing price on that particular day of stock market trading.

```
#Plot the True Adj Close Value
df['Adj Close'].plot()
```



Step 5:

Setting the Target Variable and Selecting the Features

The output column is then assigned to the target variable in the following step. It is the adjusted relative value of Microsoft Stock in this situation. Furthermore, we pick the features that serve as the independent variable to the target variable (dependent variable). We choose four characteristics to account for training purposes:

- Open
- High
- Low
- Volume

```
#Set Target Variable
output_var = PD.DataFrame(df['Adj Close'])
#Selecting the Features
features = ['Open', 'High', 'Low', 'Volume']
```

Step 6: Scaling

To decrease the computational cost of the data in the table, we will scale the stock values to values between 0 and 1. As a result, all of the data in large numbers is reduced, and therefore

```
#Scaling
scaler = MinMaxScaler()
feature_transform = scaler.fit_transform(df[features])
feature_transform= pd.DataFrame(columns=features, data=feature_transform, index=df.index)
feature_transform.head()
```

Date	Open	High	Low	Volume
1990-01-02	0.000129	0.000105	0.000129	0.064837
1990-01-03	0.000265	0.000195	0.000273	0.144673
1990-01-04	0.000249	0.000300	0.000288	0.160404
1990-01-05	0.000386	0.000300	0.000334	0.086566
1990-01-08	0.000265	0.000240	0.000273	0.072656

memory consumption is decreased. Also, because the data is not spread out in huge values, we can achieve greater precision by scaling down. To perform this, we will be using the MinMaxScaler class of the sci-kit-learn library.

As shown in the above table, the values of the feature variables are scaled down to lower values when compared to the real values given above.

Step 7: Creating a Training Set and a Test Set for Stock Market Prediction

We must divide the entire dataset into training and test sets before feeding it into the training model. The Machine Learning LSTM model will be trained on the data in the training set and tested for accuracy and backpropagation on the test set.

The sci-kit-learn library's TimeSeriesSplit class will be used for this. We set the number of splits to 10, indicating that 10% of the data will be used as the test set and 90% of the data will be used to train the LSTM model. The advantage of utilizing this Time Series split is that the split time series data samples are examined at regular time intervals.

Step 8: Data Processing For LSTM

Once the training and test sets are finalized, we will input the data into the LSTM model. Before we can do that, we must transform the training and test set data into a format that the LSTM model can interpret. As the LSTM needs that the data to be provided in the 3D form, we first transform the training and test data to NumPy arrays and then restructure them to match the format (Number of Samples, 1, Number of Features). Now, 6667 are the number of samples in the training set, which is 90% of 7334, and the number of features is 4. Therefore, the training set is reshaped to reflect this (6667, 1, 4). Likewise, the test set is reshaped.

```
#Process the data for LSTM
trainX =np.array(X_train)
testX =np.array(X_test)
X_train = trainX.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test = testX.reshape(X_test.shape[0], 1, X_test.shape[1])
```

Step 9: Building the LSTM Model for Stock Market Prediction

Finally, we arrive at the point when we construct the LSTM Model. In this step, we'll build a Sequential Keras model with one LSTM layer. The LSTM layer has 32 units and is followed by one Dense Layer of one neuron.

We compile the model using Adam Optimizer and the Mean Squared Error as the loss function. For an LSTM model, this is the most preferred combination. The model is plotted and presented below.

```
#Building the LSTM Model
lstm = Sequential()
lstm.add(LSTM(32, input_shape=(1, trainX.shape[1]), activation='relu', return_sequences=False))
lstm.add(Dense(1))
lstm.compile(loss='mean_squared_error', optimizer='adam')
plot_model(lstm, show_shapes=True, show_layer_names=True)
```

Step 10: Training the Stock Market Prediction Model

Finally, we use the fit function to train the LSTM model created above on the training data for 100 epochs with a batch size of 8.

```
#Model Training
history=lstm.fit(X_train, y_train, epochs=100, batch_size=8, verbose=1, shuffle=False)
Epoch 1/100
834/834 [==============================] - 3s 2ms/step - loss: 67.1211
Epoch 2/100
834/834 [==============================] - 1s 2ms/step - loss: 70.4911
Epoch 3/100
834/834 [==============================] - 1s 2ms/step - loss: 48.8155
Epoch 4/100
834/834 [==============================] - 1s 2ms/step - loss: 21.5447
Epoch 5/100
834/834 [==============================] - 1s 2ms/step - loss: 6.1709
Epoch 6/100
834/834 [==============================] - 1s 2ms/step - loss: 1.8726
Epoch 7/100
834/834 [==============================] - 1s 2ms/step - loss: 0.9380
Epoch 8/100
834/834 [==============================] - 2s 2ms/step - loss: 0.6566
Epoch 9/100
834/834 [==============================] - 1s 2ms/step - loss: 0.5369
```

Finally, we can observe that the loss value has dropped exponentially over time over the 100-epoch training procedure, reaching a value of 0.4599.

Step 11: Making the LSTM Prediction

Now that we have our model ready, we can use it to forecast the Adjacent Close Value of the Microsoft stock by using a model trained using the LSTM network on the test set. This is

accomplished by employing the simple predict function on the LSTM model that has been created.

Step 11: Making the LSTM Prediction

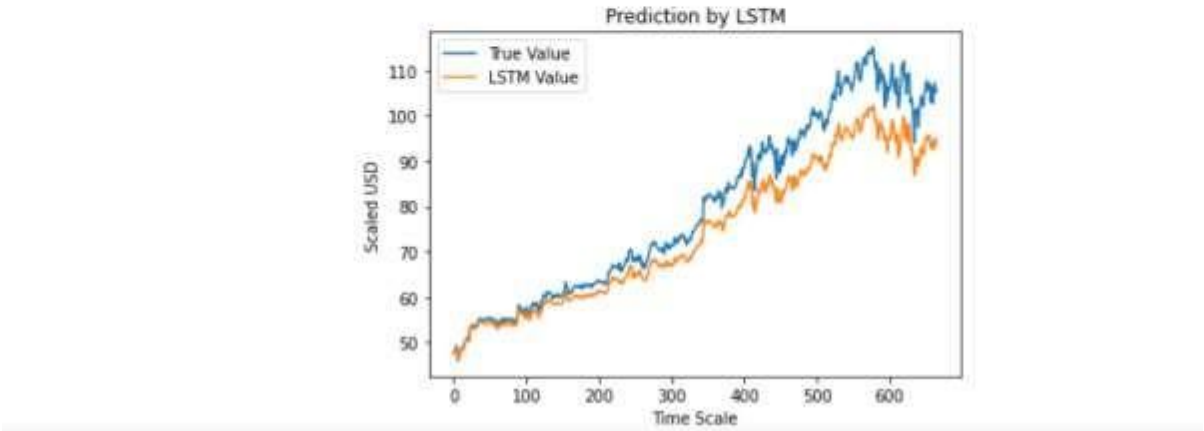
Now that we have our model ready, we can use it to forecast the Adjacent Close Value of the Microsoft stock by using a model trained using the LSTM network on the test set. This is accomplished by employing the simple predict function on the LSTM model that has been created.

```
#LSTM Prediction
y_pred= lstm.predict(X_test)
```

Step 12: Comparing Predicted vs True Adjusted Close Value – LSTM

Finally, now that we've projected the values for the test set, we can display the graph to compare both Adj Close's true values and Adj Close's predicted value using the LSTM Machine Learning model.

```
#Predicted vs True Adj Close Value – LSTM
plt.plot(y_test, label='True Value')
plt.plot(y_pred, label='LSTM Value')
plt.title("Prediction by LSTM")
plt.xlabel('Time Scale')
plt.ylabel('Scaled USD')
plt.legend()
plt.show()
```



The graph above demonstrates that the extremely basic single LSTM network model created above detects some patterns. We may get a more accurate depiction of every specific company's stock value by fine-tuning many parameters and adding more LSTM layers to the model.

-Step 3: Business Modelling

Membership-Based Model:

Offer numerous subscription plans, each with a different set of features and degrees of access. For instance, you can choose between a basic package that just gives you access to a select few stock price forecasts and a premium plan that gives you access to more stock price predictions, more market analysis tools, and advanced features.

1. Free Trial time: Offer users a free trial time during which they can utilise the service before subscribing. This enables potential clients to evaluate the precision and utility of the predictions and decide whether the service satisfies their needs.

2. Charge users a recurring subscription for continuous access to stock price predictions and other features on a monthly or annual basis. The cost can be calculated depending on elements like the number of stocks covered, the frequency of updates, and the degree of predicted accuracy.

3. Offer tier-based pricing structures to accommodate various customer preferences. For instance, you may provide various membership tiers with various degrees of frequency or accuracy of forecast. Higher-tier subscriptions may provide more regular updates and more accurate forecasts, giving customers access to more thorough and trustworthy insights.

4. Personalised Portfolio Recommendations: Provide personalised portfolio suggestions as part of higher-tier subscriptions based on users' investing

objectives, risk tolerance, and past performance.These recommendations can help users optimize their portfolios and make more informed investment decisions.

Added premium options:

5. Real-time market updates: Give current information on market developments, including trends, news, and events that could affect stock prices. Users that use this feature will be able to keep informed and make quick judgements.

6. Users can create unique alerts for particular stock movements or predetermined conditions using custom alerts. Users may be notified, for instance, when a stock crosses a specific price threshold or when the state of the market significantly changes.

7. Expert Evaluation and Suggestions: provide access to in-depth research and opinions from financial analysts or subject matter experts. This may involve in-depth analysis of the market, market commentary, or unique interviews with top business figures. Giving users access to professional views can improve the service's value proposition and draw customers who want more in-depth market knowledge.

Some of the other business models that can be considered are:

1. **Freemium Model:** Provide a basic version of the stock market predictor for free, attracting a large user base. Then offer premium features or advanced predictions through a paid upgrade. This model allows users to experience the product before deciding to pay for additional benefits.

    a. Features: Provide a basic version of the stock market predictor with limited features to attract a larger user base. Premium features may include more accurate predictions, real-time updates, advanced analytics, personalized recommendations, or access to exclusive content.
    b. Revenue Generation: Offer a free version to attract users, and then charge for access to premium features or advanced predictions. This

could be through one-time payments or a recurring subscription for premium access.

c. Advantages: The freemium model allows users to experience the product before committing to a paid version. It helps generate a larger user base, increases brand awareness, and offers the potential to upsell premium features.

2. **Licensing Model:** Instead of directly providing predictions to end-users, license the technology and data to financial institutions, hedge funds, or investment firms. They can integrate your stock market predictor into their existing platforms or trading systems, paying a licensing fee or revenue-sharing agreement.

    a. Features: Develop a robust stock market predictor and license it to financial institutions, hedge funds, or investment firms. The features can include customizable prediction models, integration options, support for various financial instruments, and access to historical and real-time data.

    b. Revenue Generation: Charge a licensing fee or enter into a revenue-sharing agreement based on the usage or success of the licensed technology.

    c. Advantages: This model allows you to leverage the expertise and resources of established financial institutions, and it can lead to significant revenue generation through licensing fees or profit-sharing arrangements.

3. **Custom Development Model:** Develop a customized version of the stock market predictor tailored to specific clients' needs. This model involves collaborating with financial institutions or professional investors to create a bespoke solution that aligns with their investment strategies and requirements.

    a. Features: Collaborate with financial institutions or professional investors to develop a customized version of the stock market predictor tailored to their specific requirements. This can include integrating the

solution with their existing platforms, incorporating proprietary algorithms, or focusing on specific market sectors.

b. Revenue Generation: Charge clients for the development, implementation, and ongoing support of the customized solution. This could involve project-based pricing or long-term service contracts.
c. Advantages: The custom development model offers personalized solutions that meet the specific needs of clients. It can result in higher-priced contracts, deeper partnerships, and long-term relationships with financial industry players.

4. **API Model:** Build an API (Application Programming Interface) that allows other developers to integrate your stock market predictor into their own applications or platforms. Charge developers based on usage or offer different pricing tiers depending on the volume of predictions they access.

    a. Features: Develop an API that provides programmatic access to your stock market predictor. The API should offer functionalities such as retrieving predictions, accessing historical data, integrating with trading systems, and enabling real-time updates.
    b. Revenue Generation: Charge developers or businesses for API usage based on factors like the number of API calls, data volume, or subscription tiers.
    c. Advantages: The API model allows other developers to integrate your stock market predictor into their own applications or platforms, expanding its reach and potential user base. It provides a scalable revenue stream based on API usage and can foster an ecosystem of third-party applications.

5. **Data Insights and Analysis Model:** Instead of directly providing predictions, focus on analyzing and providing insights on historical stock market data. Offer reports, data visualizations, and trend analysis to help investors make informed decisions. Generate revenue through subscriptions, licensing, or data sales.

a. Features: Instead of providing direct predictions, focus on analyzing and providing insights on historical stock market data. Offer reports, visualizations, trend analysis, sector performance analysis, or sentiment analysis to help investors make informed decisions.
b. Revenue Generation: Generate revenue through subscriptions, selling data reports, or partnering with financial news outlets or research firms to provide market insights.
c. Advantages: This model capitalizes on the demand for data-driven insights in the financial industry. It leverages your expertise in data analysis, visualization, and market research, offering value to investors seeking comprehensive analysis and historical trends.

External data and understanding the market :

1. https://www.investopedia.com/terms/s/stock-analysis.asp
2. https://startuptalky.com/zerodha-business-model/

Dataset we'll be using in this is :

https://www.kaggle.com/datasets/rohanrao/nifty50-stock-market-data

Source: The dataset is sourced from the National Stock Exchange of India (NSE), which is the leading stock exchange in India. The Nifty 50 index represents the performance of the top 50 companies listed on the NSE.

Features: The dataset consists of several features that provide information about each company's stock performance. Some of the common features available in the dataset include:

- Date: The date for which the stock market data is recorded.
- Open: The opening price of the stock on that 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 that day.
- Volume: The trading volume, i.e., the number of shares traded on that day.
- Turnover: The total value of shares traded on that day.

-Step 4: Financial Modelling (equation) with Machine Learning & Data Analysis

Financial Modelling

A Revenue-based financial model for a Stock Market Prediction system can be an effective way to generate revenue. Here's how it could work:

1. **Revenue Projections:** To provide revenue projections for the Stock Market Prediction system, we'll need to make some assumptions. Let's assume the following:
    a. **Average commission rate per successful referral:** $10 (this could be a percentage of the total Stock amount, but for simplicity, let's assume a fixed amount).
    b. **Average number of monthly successful referrals:** 500 (based on market research and projections).
    c. **Payment processing fees:** 2% of the revenue from successful referrals.

    we can calculate the revenue projections:
    - Monthly revenue from successful referrals: Average commission rate per successful referral * Average number of monthly successful referrals = $10 * 500 = $5,000
    - Cost of Sales: Payment processing fees = 2% of the revenue from successful referrals. Monthly cost of sales = 2% * $5,000 = $100 Based on these projections, the monthly revenue from successful referrals would be $5,000, and the monthly cost of sales would be $100.

.

2. **Cost of Sales:** The cost of sales in the Stock Market Prediction system model typically includes payment processing fees associated with the revenue generated from successful referrals. Here's a more detailed breakdown of the cost of sales:

a. **Payment Processing Fees:**
    i. When a user makes a reservation or places an order at a Stock market based on the system's recommendation, the system earns a commission. This commission is a percentage of the total amount or a fixed amount per transaction, as agreed upon with the partner.
    ii. Payment processing providers, such as credit card processors or online payment gateways, usually charge a fee for processing these transactions.
    iii. The specific percentage may vary based on the payment processor and the negotiated terms.

For example:

- - Average commission rate per successful referral: 10%
- - Monthly revenue from successful referrals: $10,000
- - Payment processing fees: 2%

  In this case, the calculation for the cost of sales would be:

  Monthly cost of sales = (Monthly revenue from successful referrals) * (Payment processing fees/100)

  = $10,000 * (2/100)

  = $200

  This means that the system incurs a cost of $200 per month for payment processing fees.

3. **Operating Expenses:**
    a. **Development and Maintenance Costs:**
        i. Website development: Including design, front-end and back-end development, hosting, and domain registration.
        ii. Platform maintenance and updates: Regular maintenance, bug fixes, and updates to ensure the platform operates smoothly.
    b. **Marketing and Advertising Expenses:**
        i. Digital marketing campaigns: Including paid advertising on platforms like Google Ads or social media platforms to increase user acquisition and engagement.

ii. Content creation: Developing engaging content such as blog posts, articles, videos, and social media posts related to stocks, charts of stocks etc.

iii. Social media management: Managing and maintaining active social media accounts to engage with the community, respond to user inquiries, and share updates about new stocks and offers.

c. Partnership Management Costs:

i. Personnel: Allocating resources to manage partnerships, negotiate commission rates, on-board stocks, and maintain on-going relationships.

ii. Relationship building: Building and nurturing relationships with broker and client through regular communication, meetings, and providing them with performance reports and feedback.

d. Customer Support Costs:

i. Personnel: Hiring customer support representatives to handle user queries, technical issues, and provide assistance with daily trades.

ii. Infrastructure: Providing necessary tools and systems to manage customer support channels such as email, live chat, or phone support.

e. Other Operating Expenses:

i. Legal and Compliance: Seeking legal advice, ensuring compliance with local regulations, and maintaining necessary licenses and permits.

ii. Accounting: Engaging accounting services for bookkeeping, financial statements, and tax filings.

4. Profit/Loss Calculation:

a. Gross Profit Calculation:

i. Gross profit is the revenue generated from successful referrals minus the cost of sales. The cost of sales in this case refers to the payment processing fees associated with the revenue-based model.

ii. Gross profit = Monthly revenue from successful referrals - Monthly cost of sales

b. Net Profit Calculation:

i. Net profit is the gross profit minus the total monthly operating expenses. It represents the overall profitability of the profile after accounting for all costs and expenses.

ii. Net profit = Gross profit - Total monthly operating expenses

iii. The total monthly operating expenses include development and maintenance costs, marketing and advertising expenses, partnership management costs, customer support costs, and other operating expenses.

c. Cash Flow Considerations:

i. Cash flow is an important aspect of financial modeling as it represents the actual inflow and outflow of cash in the business. It helps determine the sustainability and liquidity of the operation.

ii. Initial investment refers to the upfront costs required to set up the business, including equipment, website development, and initial marketing expenses.

iii. Monthly cash inflow is the revenue generated from successful referrals on a monthly basis.

iv. Monthly cash outflow is the total monthly operating expenses required to run the business.

v. Net cash flow is the difference between monthly cash inflow and monthly cash outflow, representing the net amount of cash generated or consumed each month.

vi. Cumulative cash flow is the running total of net cash flow over time, taking into account the initial investment.

d. Break-even Analysis:

i. The break-even point is the point at which the revenue generated from successful referrals equals the total monthly operating expenses. It helps determine the minimum number of successful referrals needed to cover the costs and achieve profitability.

ii. Breakeven referrals = Monthly cost of sales / Average commission rate per successful referral

e. Financial Ratios and Metrics:

i. Return on Investment (ROI): Net profit / Initial investment * 100

ii. ROI indicates the profitability of the investment relative to its cost.

iii. Payback Period: Initial investment / Monthly net cash flow

iv. The payback period represents the time it takes to recover the initial investment based on the monthly net cash flow.

5. Cash Flow Considerations:

a. Initial Investment:

i. Determine the initial investment required to set up the platform, develop the website, and cover initial marketing expenses.

ii. Consider costs such as software development, hosting, domain registration, branding, and marketing campaigns.

iii. Calculate the total amount needed for the initial investment, denoted as $F.

b. Monthly Cash Inflow:

i. The primary source of cash inflow is the revenue generated from successful referrals made through the platform.

ii. Calculate the average commission rate per successful referral (e.g., $X) and estimate the average number of monthly successful referrals (e.g., Y).

iii. Multiply the commission rate by the number of successful referrals to determine the monthly revenue from successful referrals.

c. Monthly Cash Outflow:

i. Consider the various operating expenses required to run the stock market on a monthly basis.

ii. Sum up the costs associated with development and maintenance ($A), marketing and advertising ($B), partnership management ($C), customer support ($D), and other operating expenses ($E).

iii. Calculate the total monthly operating expenses, denoted as the sum of $A, $B, $C, $D, and $E.

d. Net Cash Flow:

i. Subtract the total monthly operating expenses from the monthly revenue from successful referrals to calculate the net cash flow.

ii. Net cash flow = Monthly revenue from successful referrals - Total monthly operating expenses.

e. Cumulative Cash Flow:

i. Track the cumulative cash flow over time to understand the financial progress of the business.

ii. Start with the initial investment ($F) and add the net cash flow from each subsequent month.

iii. Cumulative cash flow = Net cash flow + Initial investment.

6. Financial Ratios and Metrics:

a. Gross Profit Margin:

i. Formula: (Monthly revenue from successful referrals - Monthly cost of sales) / Monthly revenue from successful referrals * 100

ii. This ratio measures the profitability of each successful referral after accounting for the cost of generating that revenue.

b. Net Profit Margin:

i. Formula: Net profit / Monthly revenue from successful referrals * 100

ii. This ratio indicates the overall profitability of the business after considering all operating expenses.

c. Return on Investment (ROI):

i. Formula: Net profit / Initial investment * 100

ii. ROI measures the profitability of the business relative to the initial investment made.

d. Cash Flow Break-even Point:

i. This metric determines the number of successful referrals needed to cover the monthly operating expenses. It indicates the minimum level of business activity required to reach profitability.

e. Payback Period:

i. Formula: Initial investment / Monthly net cash flow
ii. The payback period represents the time it takes for the initial investment to be recovered from the monthly net cash flows generated by the business.

f. Customer Acquisition Cost (CAC):

i. Formula: Total marketing and advertising expenses / Number of acquired customers
ii. CAC measures the cost incurred to acquire each new customer for the recommendation system. It helps assess the efficiency of your marketing efforts.

g. Customer Lifetime Value (CLV):

i. Formula: Average revenue per successful referral * Average customer retention period
ii. CLV estimates the total revenue expected to be generated from a single customer during their engagement with the recommendation system.

h. Return on Marketing Investment (ROMI):

i. Formula: (Net profit from marketing efforts - Marketing expenses) / Marketing expenses * 100
ii. ROMI measures the effectiveness of marketing campaigns by evaluating the return generated relative to the marketing expenses incurred.

i. Break-even Point in Time:

i. This metric determines the time it takes for the net cash flows to reach a breakeven point, where the total inflows equal the total outflows.

Data Pre-processing and Analysis :

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv("/content/TATASTEEL.csv")
df.head()
```

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume
0	2000-01-03	TISCO	EQ	142.35	148.00	153.2	146.10	152.50	152.45	150.92	2003185
1	2000-01-04	TISCO	EQ	152.45	150.10	153.0	143.05	151.95	150.80	151.03	1555136
2	2000-	TISCO	EQ	150.80	144.60	162.9	144.60	158.00	156.55	156.85	3840284

+ Code + Text

```
df.shape
```

```
(5306, 15)
```

```
df.dtypes
```

```
Date                  object
Symbol                object
Series                object
Prev Close           float64
Open                 float64
High                 float64
Low                  float64
Last                 float64
Close                float64
VWAP                 float64
Volume                 int64
Turnover             float64
Trades               float64
Deliverable Volume   float64
%Deliverble          float64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5306 entries, 0 to 5305
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Date                 5306 non-null   object
 1   Symbol               5306 non-null   object
 2   Series               5306 non-null   object
 3   Prev Close           5306 non-null   float64
 4   Open                 5306 non-null   float64
 5   High                 5306 non-null   float64
 6   Low                  5306 non-null   float64
 7   Last                 5306 non-null   float64
 8   Close                5306 non-null   float64
 9   VWAP                 5306 non-null   float64
 10  Volume               5306 non-null   int64
 11  Turnover             5306 non-null   float64
 12  Trades               2456 non-null   float64
 13  Deliverable Volume   4792 non-null   float64
 14  %Deliverble          4792 non-null   float64
dtypes: float64(11), int64(1), object(3)
memory usage: 621.9+ KB
```

```
df.isnull().sum()
```

```
Date                    0
Symbol                  0
Series                  0
Prev Close              0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
```

```
VWAP                   0
Volume                 0
Turnover               0
Trades              2858
Deliverable Volume   514
%Deliverble          514
dtype: int64
```

```
df.head()
```

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trac
0	2000-01-03	TISCO	EQ	142.35	148.00	153.2	146.10	152.50	152.45	150.92	2003185	3.023164e+13	N
1	2000-01-04	TISCO	EQ	152.45	150.10	153.0	143.05	151.95	150.80	151.03	1555136	2.348785e+13	N
2	2000-01-05	TISCO	EQ	150.80	144.60	162.9	144.60	158.00	156.55	156.85	3840284	6.023364e+13	N

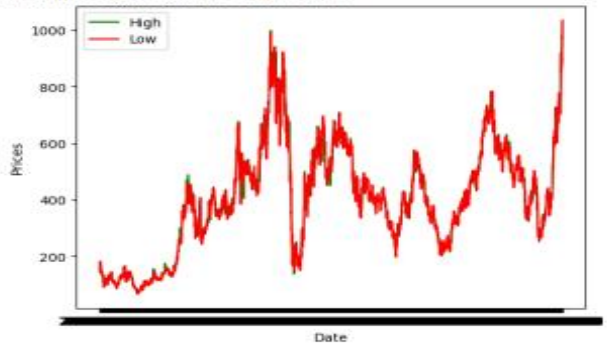
```
df.describe()
```

	Prev Close	Open	High	Low	Last	Close	VWAP	Vo
count	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5.306000
mean	403.385658	404.253581	411.210460	396.509197	403.467414	403.553703	404.062991	6.165253
std	187.146366	187.559958	190.791329	183.858461	187.265190	187.312178	187.436529	5.329084
min	67.250000	66.000000	69.700000	66.000000	67.300000	67.250000	67.970000	2.329100
25%	275.775000	275.600000	284.412500	270.000000	275.812500	275.937500	276.935000	2.801380
50%	402.850000	403.000000	409.375000	396.650000	402.700000	402.900000	403.430000	4.800300
75%	523.987500	525.000000	534.725000	516.487500	523.950000	524.075000	525.230000	7.833888
max	1031.350000	1024.000000	1052.600000	1011.100000	1035.000000	1034.000000	1031.950000	6.428460

Open vs Close Comparison

```
sns.lineplot(x=df["Date"],y=df["Open"],color="g" , label ="High")
sns.lineplot(x=df["Date"],y=df["Close"],color="r" , label ="Low")
plt.ylabel("Prices")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fd43ea43ac8>
```



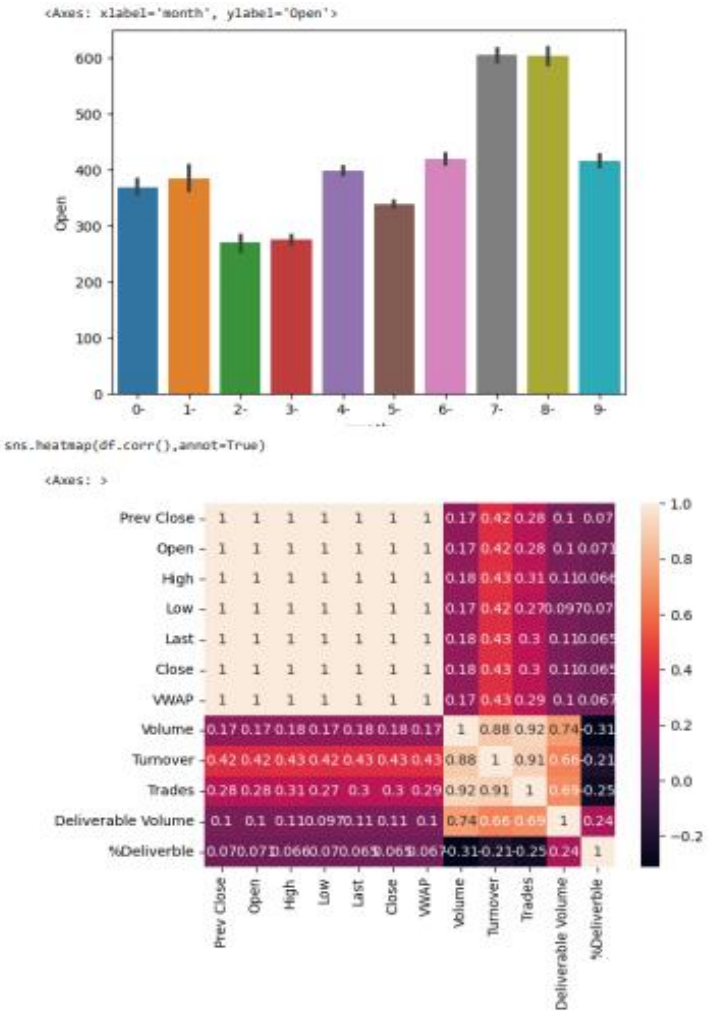

```
df["month"]=df["Date"].str[3:5]

sorted_df=df.sort_values(by="month")

sns.barplot(x=sorted_df["month"],y=sorted_df["Open"])
```


In [141…
```
plt.figure(figsize=(8,4))
sns.barplot(x=sorted_df["date"],y=sorted_df["Open"])
plt.title("Opening Prices by Dates")
```

Out[1411]: Text(0.5, 1.0, 'Opening Prices by Dates')


<Axes: xlabel='month', ylabel='Open'>
Open
0
100
200
300
400
500
600
0-
1-
2-
3-
4-
5-
6-
7-
8-
9-
sns.heatmap(df.corr(),annot=True)
<Axes: >
Prev Close
Open
High
Low
Last
Close
VWAP
Volume
Turnover
Trades
Deliverable Volume
%Deliverble
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

----------Thank you----------