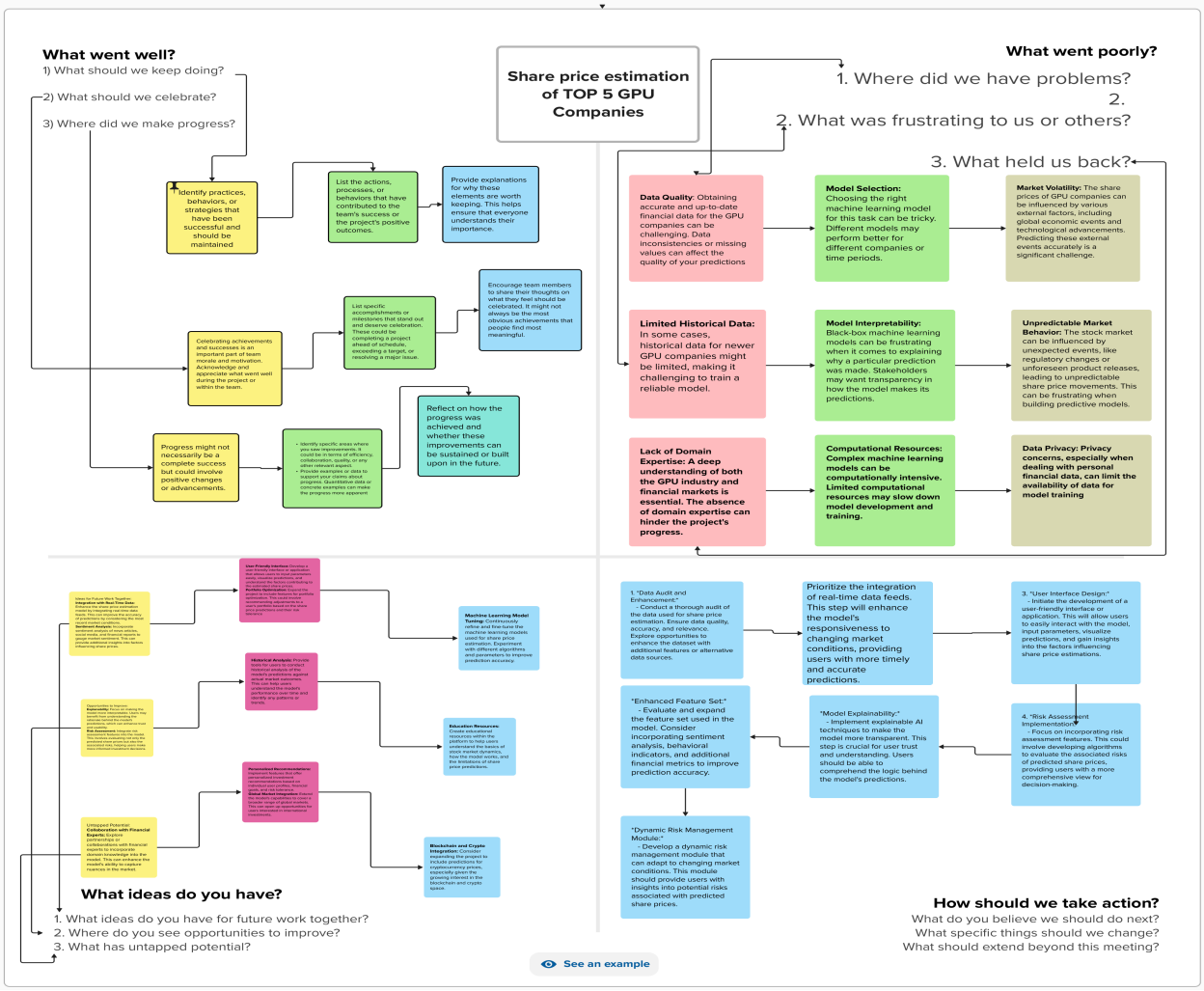
Team_ID - 592679

Share price estimation of TOP 5 GPU Companies

The objective of this project is to estimate the share prices of the top 5 GPU companies in the market using historical data and current market trends. The aim is to develop a predictive model that can forecast the future prices of these companies based on their historical performance and other relevant factors. The model shouldtake into consideration various economic indicators, industry trends, company financials, and other relevant data to make accurate predictions. The analysis should be performed using statistical and machine learning/ Deep learning techniques and the results should be presented in a clear and concise manner. The final output of the project will be a report outlining the predicted share prices of the top 5 GPU companies in the market, along with an analysis of the factors that have contributed to their performance.



Team_ID - 592679

Prior Knowledge:

To complete this project, you must require following software's , concepts and packages

1. Anaconda navigator:
   - Refer to the link below to download anaconda navigator
   - Link : https://www.youtube.com/watch?v=5mDYijMfSzs
2. Python packages:
   - open anaconda prompt as administrator
   - Type "pip Installprophet" (make sure you are working on python 64 bit)
   - Type "pip install flask".
3. Deep Learning Concepts
   - CNN:https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add
   - ARIMA:

     https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arim a.asp#:~:text=An%20autoregressive%20integrated%20moving%20average%2C%20or %20ARIMA%2C%20is%20a%20statistical,values%20based%20on%20past%20values

     .
   - LSTM:

     https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-l stm/
   - SARIMA:

     https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/
   - Facebook Prophet:

     https://www.geeksforgeeks.org/time-series-analysis-using-facebook-prophet/
   - Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Team_ID - 592679

Project Objectives:

Milestone 1: Define Problem/ Problem Understanding

Activity 1:

Activity 2: Business requirements

Here are some potential business requirements for Share price predictor:

a. Accurate forecasting: The predictor must be able to accurately forecast Share prices for a given time period in the future. The accuracy of the forecasting is crucial for investors, stock holders, and other stakeholders to make informed decisions on the production and marketing of rice.

b. Real-time data acquisition: The predictor must be able to acquire real-time data on stock information and other relevant factors that affect stock price prediction. The data acquisition must be seamless and efficient to ensure that the predictor is always

up-to-date with the latest information.

c. User-friendly interface: The predictor must have a user-friendly interface that is easy to navigate and understand. The interface should present the results of the predictor in a clear and concise manner to enable farmers and other stakeholders to make informed decisions.

d. Report generation: Generate a report outlining the predicted share prices of the top 5 GPU companies in the market, along with an analysis of the factors that have contributed to their performance. The report should be presented in a clear and concise manner, with appropriate visualizations and insights to help stakeholders make informed decisions.

Activity 3: Literature Survey

Team_ID - 592679

Introduction to GPU Companies:
Begin your literature survey with an introduction to the top GPU (Graphics Processing Unit) companies. Discuss their importance in the technology industry and their influence on the stock market. You can include NVIDIA, AMD, Intel, and others.

Factors Affecting Share Prices:
Explore existing literature on the factors that influence the share prices of GPU companies. Common factors include financial performance, market trends, competition, product launches, and overall economic conditions.

Financial Analysis Methods:
Investigate various financial analysis methods used to estimate share prices. This can include fundamental analysis, technical analysis, and sentiment analysis. Discuss the strengths and limitations of each approach.

Predictive Models:
Look for research that delves into predictive models for share price estimation. Machine learning and data-driven models are becoming increasingly popular in this domain. Studies that employ regression analysis, neural networks, or other advanced techniques are valuable.

Data Sources:
Discuss the sources of data used for share price estimation. This may include financial reports, stock market data, social media sentiment, and news articles. Highlight the importance of data quality and reliability.

Economic Indicators:
Analyze how macroeconomic indicators such as GDP growth, interest rates, and inflation impact the share prices of GPU companies. Economic conditions can have a significant influence on stock markets.

Case Studies:
Examine specific case studies or research papers that estimate share prices of GPU companies. These studies often provide insights into real-world applications of various methods.

Volatility and Risk Assessment:
Explore literature on assessing the volatility and risk associated with investing in GPU companies. Risk management is a crucial aspect of share price estimation.

Regulatory Environment:

Consider the regulatory environment and its impact on GPU companies. Changes in regulations can have a substantial influence on share prices.

Market Sentiment Analysis:
Investigate research on sentiment analysis, which uses natural language processing (NLP) techniques to gauge market sentiment from news articles, social media, and other textual data.

Industry Trends and Innovations:
Keep an eye on research discussing the latest trends and innovations in the GPU industry. New product launches, technological advancements, and market disruptions can impact share prices.

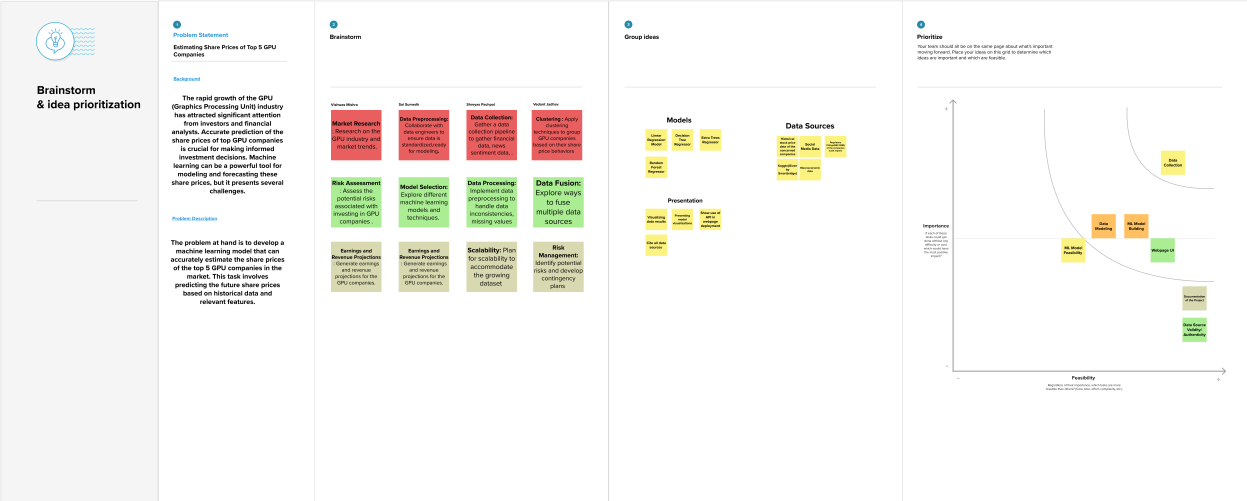
Machine Learning and AI in Finance:
Research how machine learning and artificial intelligence (AI) are being applied to stock price prediction and financial analysis. AI models can capture complex patterns in financial data.

Challenges and Limitations:
Highlight the challenges and limitations of share price estimation in the GPU industry. These could include data availability, model accuracy, and the inherent unpredictability of financial markets.

Conclusion and Future Directions:
Summarize the key findings from the literature survey and propose potential future directions for research in this field.

Team_ID - 592679

Brainstorm Map:



Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Explanation:

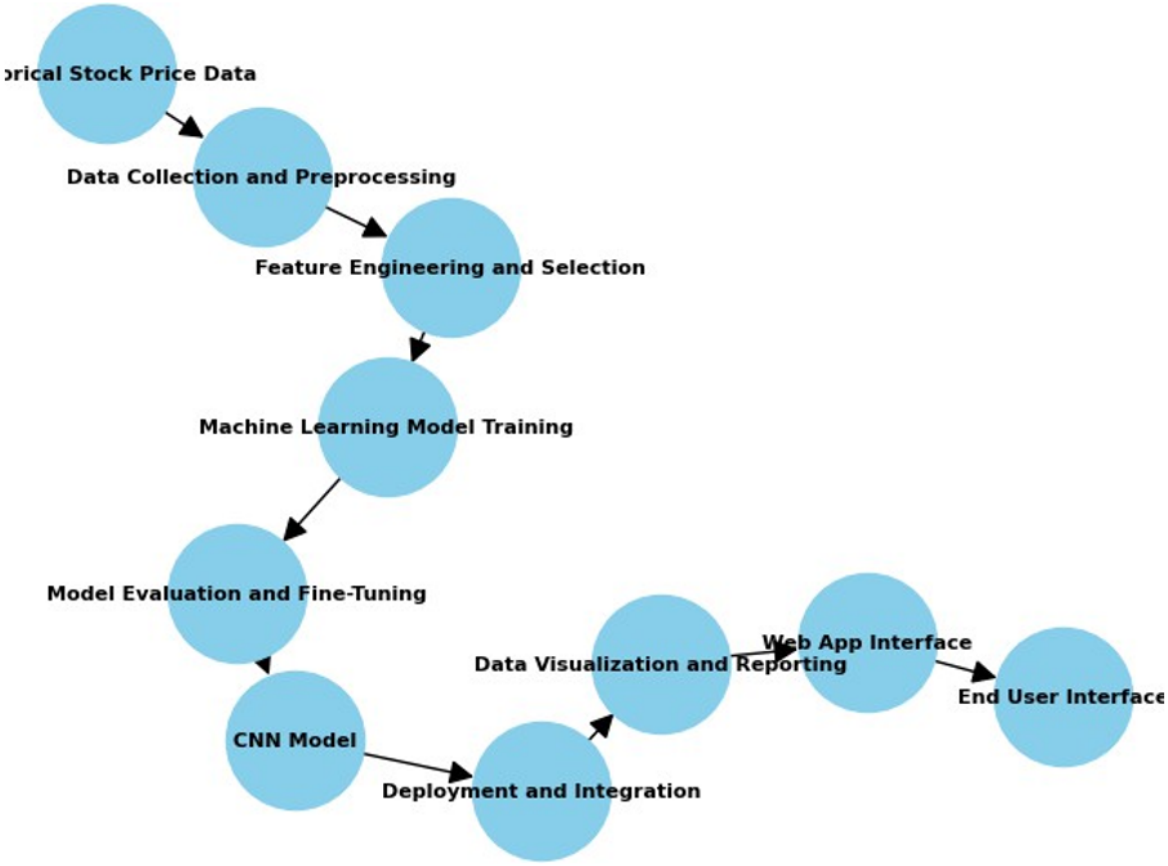
1. **Historical Stock Price Data:** This is the starting point, where historical stock price data for the top 5 GPU companies is collected.
2. **Data Collection and Preprocessing:** The collected data is processed and cleaned to remove any inconsistencies or missing values. This step ensures that the data is ready for analysis.
3. **Feature Engineering and Selection:** Relevant features are selected or engineered to improve the model's predictive performance. This step may involve transforming the data to extract meaningful insights.
4. **Machine Learning Model Training:** A Convolutional Neural Network (CNN) model is trained using the preprocessed data to predict future stock prices based on historical trends and selected features.
5. **Model Evaluation and Fine-Tuning:** The trained model is evaluated using validation data, and fine-tuning is performed to improve its accuracy and generalization.
6. **Deployment and Integration:** The trained model is deployed, and a web app is created to integrate the model for real-time predictions.
7. **Data Visualization and Reporting:** The results of the model predictions are visualized and reported in the web app. Users can interact with the interface to explore share price estimations and relevant insights.

This architecture provides a high-level overview of the key components and their interactions in the Share Price Estimation project, incorporating machine learning (CNN) and a web app for user interaction. Keep in mind that this is a simplified

representation, and the actual implementation may involve additional details and considerations based on the specific requirements of your project.

Team_ID - 592679

Example - Solution Architecture Diagram:



Code for the above Image:

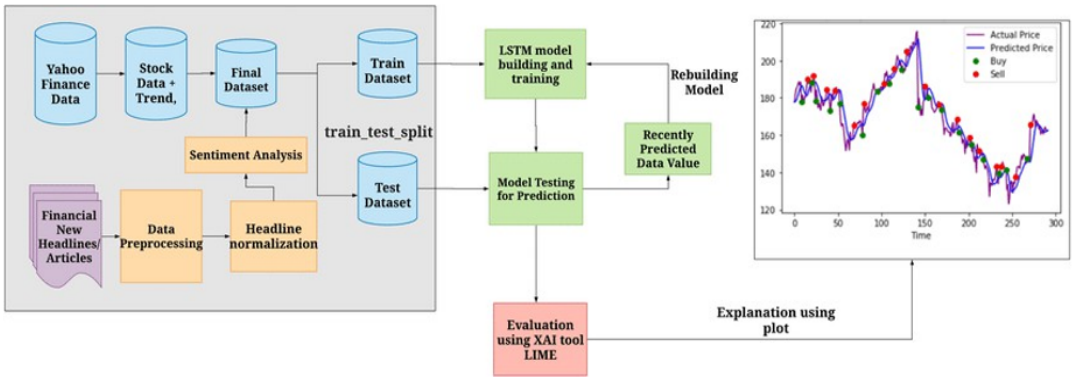
```
import network as nx
import matplotlib.pyplot as plt

# Create a directed graph
G = nx.DiGraph()

# Add nodes
nodes = ["Historical Stock Price Data", "Data Collection and
Preprocessing", "Feature Engineering and Selection", "Machine Learning
Model Training", "Model Evaluation and Fine-Tuning", "CNN Model",
```

Team_ID - 592679

```
 "Deployment and Integration",
"Data Visualization and Reporting", "Web App Interface",
"End User Interface",
]
G.add_nodes_from(nodes)
# Add edges
edges = [
("Historical Stock Price Data", "Data Collection and Preprocessing"),
("Data Collection and Preprocessing", "Feature Engineering
andSelection"),
("Feature Engineering and Selection", "Machine Learning Model
Training"),
("Machine Learning Model Training", "Model Evaluation and Fine-
Tuning"),
("Model Evaluation and Fine-Tuning", "CNN Model"), ("CNN Model",
"Deployment and Integration"),
("Deployment and Integration", "Data Visualization and Reporting"),
("Data Visualization and Reporting", "Web App Interface"),
("Web App Interface", "End User Interface"),
]
G.add_edges_from(edges) # Draw the graph
pos = nx.spring_layout(G, seed=42)
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=3000,
node_color='skyblue', font_size=8, arrowsize=20)

# Display the graph
plt.show()
```

Team_ID - 592679

Reference image:



Activity 4: Social or Business Impact.

The Share Price Estimation of Top 5 GPU Companies project can have both social and business impacts.

Social Impact:

The accurate prediction of share prices can help individual investors, particularly those who may not have extensive financial knowledge, make informed investment decisions, thereby enabling them to achieve their financial goals and potentially increase their wealth.

The project can also contribute to overall market stability by providing more accurate information to investors, helping them make informed decisions, and reducing market volatility and uncertainty.

Business Impact:

The project can be particularly useful for financial institutions, such as investment banks and hedge funds, who make investment decisions on behalf of their clients.Accurate prediction of share prices can help these institutions

improve their investment strategies and potentially increase their returns.

Accurate share price predictions can also be useful for companies operating in the GPU industry, enabling them to anticipate changes in the market and adjust their strategies accordingly. This can help companies remain competitive and achieve their business goals.

Overall, the project can have a positive impact on the financial industry by improving the accuracy of share price predictions and increasing market efficiency.

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the rightamount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Explanation:

1. **Stock Price Data Source:** The data source for historical stock prices of the top 5 GPU companies.
2. **External Data (Economic Indicators, Industry Trends):** Additional external data sources that might influence stock prices.
3. **Data Collection and Preprocessing:** Collects and preprocesses data from the stock price source and external data.
4. **Feature Engineering and Selection:** Identifies and selects relevant features for training the machine learning model.
5. **Machine Learning Model Training:** Trains a CNN model using historical stock price data and selected features.
6. **Model Evaluation and Fine-Tuning:** Evaluates the trained model's performance and fine-tunes it for better accuracy.
7. **Data Visualization and Reporting:** Visualizes the model's predictions and generates reports based on the evaluated data.
8. **End User Interface:** The final interface where end-users can interact with the system, view visualizations, and receive reports.

This diagram represents the flow of data within the system, from the initial data sources to the end user interface. Keep in mind that this is a simplified representation, and the actual system may involve more detail and complexity based on the specific requirements of your project.

Team_ID - 592679

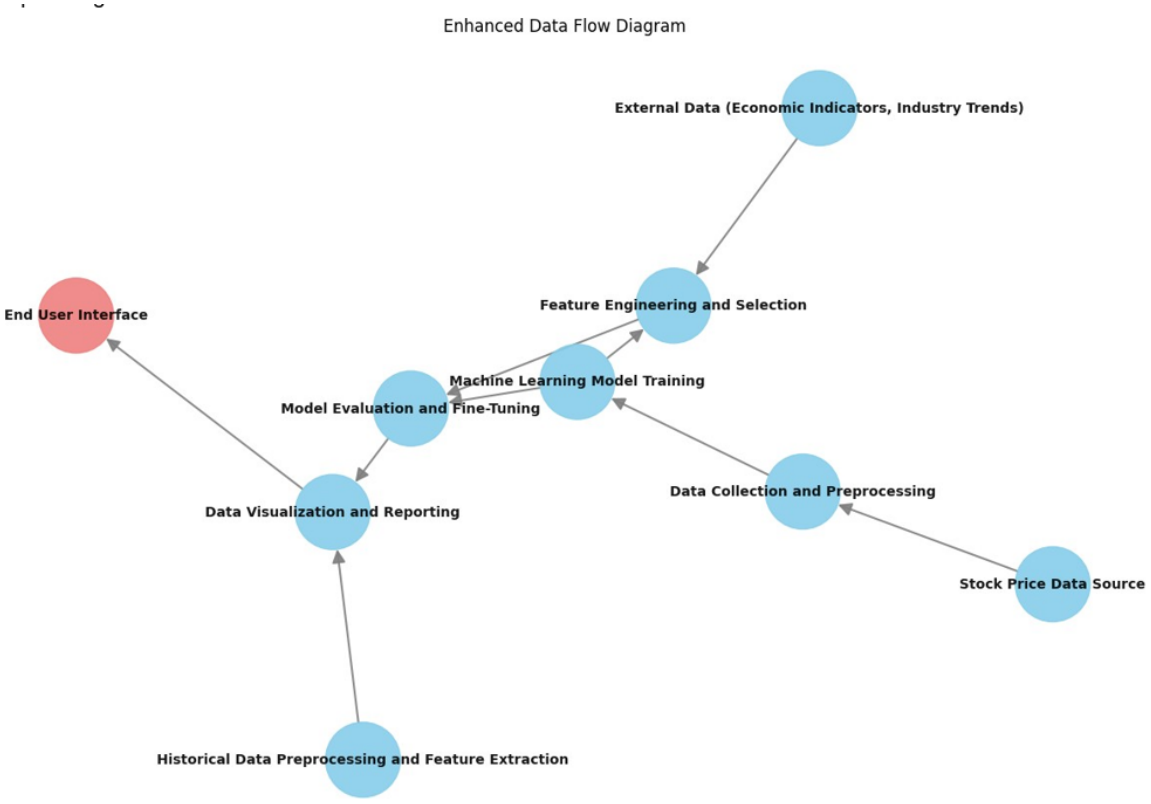


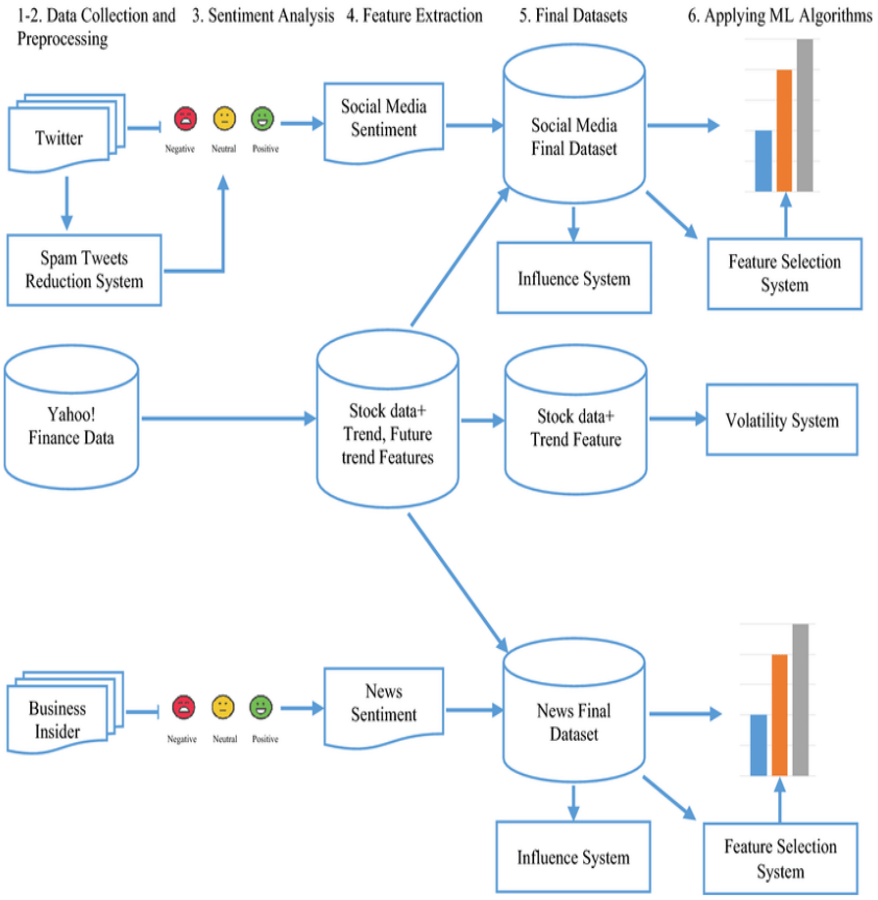
Image Code :

```
!pip install networkx matplotlib

import networkx as nx
import matplotlib.pyplot as plt

# Create a directed graph
G = nx.DiGraph()

# Add nodes
nodes = [
"Stock Price Data Source",
"Data Collection and Preprocessing",
  "Machine Learning Model Training",
"External Data (Economic Indicators,
  Industry Trends)", "Feature Engineering and
  Selection",
```

```
"Model Evaluation and Fine-Tuning", "Data Visualization and Reporting",
"Historical Data Preprocessing and Feature Extraction", "End User Interface",
]

G.add_nodes_from(nodes)

# Add edges
edges = [
("Stock Price Data Source", "Data Collection and Preprocessing"),
("Data Collection and Preprocessing", "Machine Learning Model Training"), ("Machine Learning Model Training", "Model Evaluation and Fine-Tuning"), ("Model Evaluation and Fine-Tuning", "Data Visualization and Reporting"), ("Machine Learning Model Training", "Feature Engineering and Selection"), ("Feature Engineering and Selection", "Model Evaluation and Fine-Tuning"),
("External Data (Economic Indicators, Industry Trends)", "Feature Engineering and Selection"), ("Data Visualization and Reporting", "End User Interface"),
("Historical Data Preprocessing and Feature Extraction", "Data Visualization and Reporting"),
]

G.add_edges_from(edges)

# Draw the graph using the spring layout
pos = nx.spring_layout(G, seed=42)

# Set node colors
node_colors = ["skyblue"] * len(nodes)
```

```
node_colors[-1] = "lightcoral" # End User Interface node color

# Set edge colors
edge_colors = ["gray"] * len(edges)

# Draw the graph with enhanced features
plt.figure(figsize=(12, 8))
nx.draw(
G,
pos, with_labels=True, font_weight='bold', node_size=3000,
node_color=node_colors, font_size=10, arrowsize=20, edge_color=edge_colors, width=1.5,
font_color='black', alpha=0.9
)

# Display the graph
plt.title("Enhanced Data Flow Diagram")
plt.show()
```

Team_ID - 592679



User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account /dashboard	High	Sprint-1

		USN-2	As a user, I will receiveconfirmation emailonceI have registered for the application	I can receive confirmation onemail& click confirm	High	Sprint-1
		USN-3	As a user, I can register for the applicationthrough Facebook	I canregister & accessthe dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the applicationthrough Gmail		Medium	Sprint-1
	Login	USN-5	As a user,I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)						
Customer Care Executive						
Administrator						

Milestone 2: Data Collection & Preparation

DL depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset.

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data.This data is downloaded from kaggle.com. Please refer to the link given

below to download the dataset.

Link: https://www.kaggle.com/datasets/kapturovalexander/nvidia-amd-intel-asus-msi-share-prices

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysingtechniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Importing all libraries

```
[ ]  import matplotlib.pyplot as plt
     import seaborn as sns
     import tensorflow as tf
     from sklearn.model_selection import train_test_split
     from statsmodels.tsa.arima_model import ARIMA
     from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
     import math
     from tqdm import tqdm
     from tqdm.notebook import tqdm_notebook
     import numpy as np
     import pandas as pd
     from itertools import product
     from sklearn.linear_model import LinearRegression
     import warnings
     import matplotlib.dates as mdates

     warnings.filterwarnings('ignore')
```

Activity 1.2: Read the Dataset

Our dataset formatmight be in .csv, excel files, .txt, .json, etc.
We can read the dataset with the help of pandas.

In pandas we have a function calledread_csv() to read the dataset.As a parameter we have to give the directory of the csv file.

Reading all CSV files

```
[ ]  nvidia=pd.read_csv('NVIDIA (1999 -11.07.2023).csv')
     asus=pd.read_csv('ASUS (2000 - 11.07.2023).csv')
     intel=pd.read_csv('INTEL (1980 - 11.07.2023).csv')
     msi= pd.read_csv('Motorola Solutions (MSI) (1962 -11.07.2023).csv')
     amd=pd.read_csv('AMD (1980 -11.07.2023).csv')
```

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the datasetproperly in order to fetch good results.This activity includesthe following steps.

1. Checking for missingvalues
2. Data manipulation
3. Resampling the data
4. Merging and splitting data into test and trainvariables

Note: These are the general steps of pre-processing the data before using it for training models.Depending on the condition of your dataset,you may or may not have to go throughall these steps.

Team_ID - 592679

Head data of all GPU companies

```
[ ]  amd.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	18-03-1980	0.0	3.125000	2.937500	3.031250	3.031250	727200
1	19-03-1980	0.0	3.083333	3.020833	3.041667	3.041667	295200
2	20-03-1980	0.0	3.062500	3.010417	3.010417	3.010417	159600
3	21-03-1980	0.0	3.020833	2.906250	2.916667	2.916667	130800
4	24-03-1980	0.0	2.916667	2.635417	2.666667	2.666667	436800

```
[ ]  asus.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	05-01-2000	438.747223	446.535675	436.151154	438.747223	89.092613	6.106176e+09
1	06-01-2000	440.045380	447.833862	436.151154	437.449310	88.829048	6.545984e+09
2	07-01-2000	432.256927	433.555084	425.766632	428.362701	86.983925	4.764317e+09
3	10-01-2000	434.853271	454.324158	434.853271	450.429901	91.464920	1.199988e+10
4	11-01-2000	463.410767	463.410767	442.641449	443.939606	90.146988	1.423350e+10

Team_ID - 592679

```
[ ]  intel.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	18-03-1980	0.325521	0.328125	0.322917	0.322917	0.183718	17068800
1	19-03-1980	0.330729	0.335938	0.330729	0.330729	0.188162	18508800
2	20-03-1980	0.330729	0.334635	0.329427	0.329427	0.187421	11174400
3	21-03-1980	0.322917	0.322917	0.317708	0.317708	0.180754	12172800
4	24-03-1980	0.316406	0.316406	0.311198	0.311198	0.177050	8966400

```
[ ]  msi.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	03-01-1962	0.0	1.444702	1.427952	1.436327	0.630355	77611
1	04-01-1962	0.0	1.438421	1.411202	1.423765	0.624842	59701
2	05-01-1962	0.0	1.432140	1.394452	1.415390	0.621167	107462
3	08-01-1962	0.0	1.432140	1.390264	1.390264	0.610140	89551
4	09-01-1962	0.0	1.402827	1.356764	1.356764	0.595438	83581

```
[ ]  nvidia.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	25-01-1999	0.442708	0.458333	0.410156	0.453125	0.415743	51048000
1	26-01-1999	0.458333	0.467448	0.411458	0.417969	0.383487	34320000
2	27-01-1999	0.419271	0.429688	0.395833	0.416667	0.382293	24436800
3	28-01-1999	0.416667	0.419271	0.412760	0.415365	0.381098	22752000
4	29-01-1999	0.415365	0.416667	0.395833	0.395833	0.363177	24403200

Team_ID - 592679

Activity 2.1: Checking for missing values

a. For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values presentin our dataset. So we can skip handling the missing valuesstep.

Checking for Missing Values

```
#Checking for Missing Values
amd.isnull().sum()
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

```
[ ]  asus.isnull().sum()
```

```
Date           0
Open         123
High         123
Low          123
Close        123
Adj Close    123
Volume       123
dtype: int64
```

```
[ ]  intel.isnull().sum()
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Team_ID - 592679

```
[ ]  msi.isnull().sum()
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

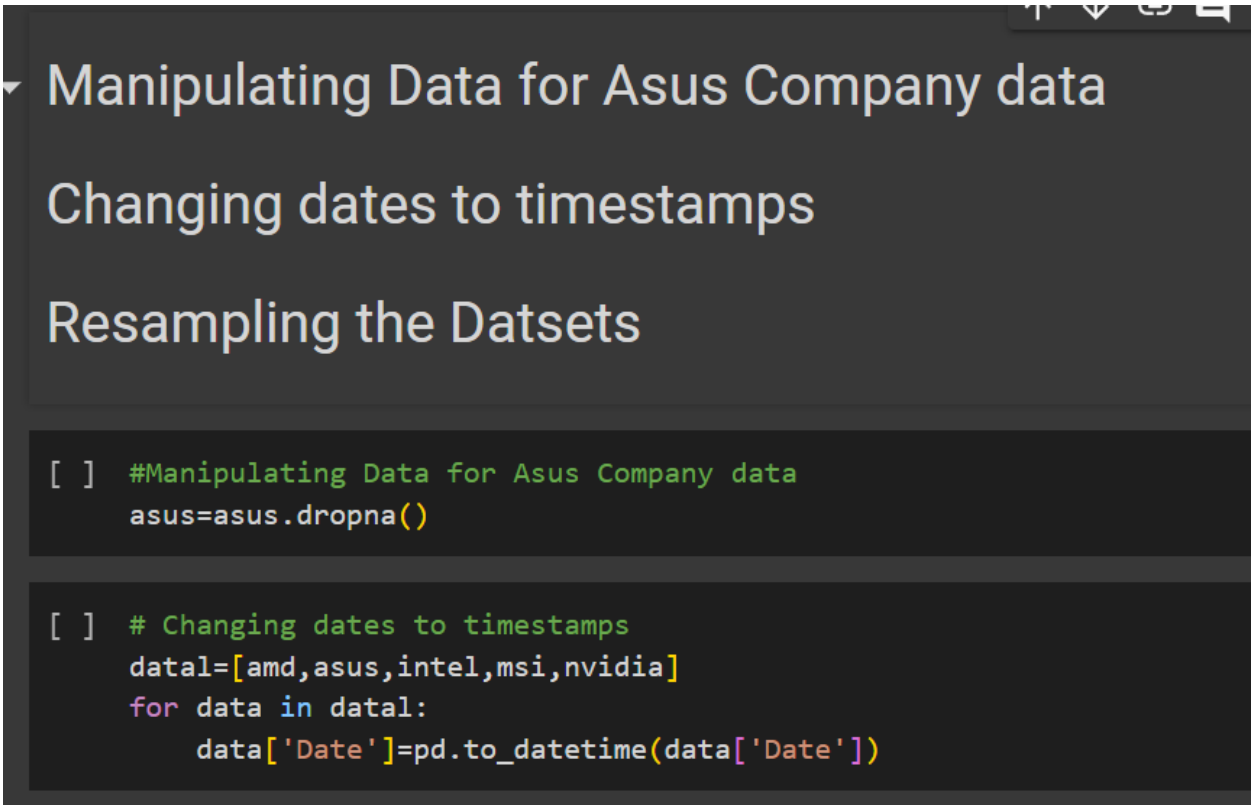
```
[ ]  nvidia.isnull().sum()
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Team_ID - 592679

Activity 2.2: Data manipulation

Since we found null values in the dataset relatedto ASUS company we are going to drop the null values but we can also fill those null values using mean/median/mode of the respective volumns.



To convert the date stringsto time stamps we are going to perform below steps. First let's checkthe type of data of the column"**Year**" using the **type()** function.
We will convertthe data of "**Year**" column using the pandas "**to_datetime**" function. We can change the '**Date'** columns of all the datasetsusing a for loop to reduce Lines of code..

Activity 2.3: Resampling the Data

Now we are going to add new columns to all the loaded datasets so that when we merge ourdata it will be easier to find which rows of data belongto which companyand also so that our model will be able to differentiate the rows of data based on the value of the new column named “Company”. The new column “Company” will be a categorical value ranging from 0 to4 where each number denotes a company as shown below. We are going to process the datasets as mentioned above using a for loop.

```
[ ]  #Resampling the Datsets
     datal=[amd,asus,intel,msi,nvidia]
     names=[0,1,2,3,4]
     index=0
     for data in datal:
         dates= data['Date']
         data['Company'] = np.repeat (names[index], len(data))
         data['Year'] =dates.dt.year
         data[ 'Month'] = dates.dt.month
         data['Day']= dates.dt.day
         index+=1
```

We will also three new columnsof data namely “Year”, “Day” and “Month”which are derivedfrom the “Date”column. We will use these three columnsfor training not the “Date”columns but we use the “Date” columnfor analysis in later stages

Activity 2.3: Merging and splitting data into test and trainvariables

Now we are going to merge our datasets, simultaneously we are also going to split the data to test and train variablesusing the below piece of code with a train to test ratio of 80 to 20.

First we are going to take 2 lists of train and test in which we are going to appendtrain and test splits of every dataset into these 2 lists. In the output each line represents the data of companies in order as stored in the data_list in below code.

Team_ID - 592679

Merging and splitting data into test and train variables

```
[ ]  # Merging and splitting data into test and train variables
     data_list=[amd, asus, intel,msi, nvidia]
     test_data = []
     train_data = []
     for data in data_list:
         train = data[ : int(0.8*len (data))]
         test = data[int(0.8*len (data)) : ]
         train_data.append(train)
         test_data.append(test)
         print(test.shape, train.shape)
```

```
(2184, 11) (8735, 11)
(1150, 11) (4596, 11)
(2184, 11) (8735, 11)
(3097, 11) (12387, 11)
(1231, 11) (4923, 11)
```

```
[ ]  train_data=pd.concat(train_data)
     test_data =pd.concat(test_data)
     print(train_data.shape)
     print(test_data.shape)
```

```
(39376, 11)
(9846, 11)
```

```
[ ]  x_train= train_data[['Open', 'High', 'Low', 'Volume', 'Year', 'Month', 'Day', 'Company']]
     x_test =test_data[['Open', 'High', 'Low', 'Volume', 'Year', 'Month', 'Day', 'Company']]
     y_train =train_data[ 'Close']
     y_test= test_data['Close']

     print(x_train.shape)
     print(x_test.shape)
     print(y_train.shape)
     print(y_test.shape)
```

```
(39376, 8)
(9846, 8)
(39376,)
(9846,)
```

Team_ID - 592679

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysisis to study the basicfeatures of data with the statistical process.Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features

Data Analysis

```
#Data Analysis
amd.describe(include='all')
```

	Date	Open	High	Low	Close	Adj Close	Volume	Company	Year	Month	Day
count	10919	10919.000000	10919.000000	10919.000000	10919.000000	10919.000000	1.091900e+04	10919.0	10919.000000	10919.000000	10919.000000
unique	10919	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	1980-03-18 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first	1980-01-04 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-12-06 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	16.842664	17.510743	16.761635	17.138932	17.138932	1.846495e+07	0.0	2001.338126	6.556186	15.708856
std	NaN	23.317716	23.609612	22.615398	23.121619	23.121619	2.815631e+07	0.0	12.509742	3.419732	8.780794
min	NaN	0.000000	1.690000	1.610000	1.620000	1.620000	0.000000e+00	0.0	1980.000000	1.000000	1.000000
25%	NaN	4.960000	5.437500	5.125000	5.300000	5.300000	1.226100e+06	0.0	1991.000000	4.000000	8.000000
50%	NaN	9.875000	10.062500	9.630000	9.875000	9.875000	6.833200e+06	0.0	2001.000000	7.000000	16.000000
75%	NaN	16.125000	16.403125	15.805000	16.120001	16.120001	2.284015e+07	0.0	2012.000000	10.000000	23.000000
max	NaN	163.279999	164.460007	156.100006	161.910004	161.910004	3.250584e+08	0.0	2023.000000	12.000000	31.000000

```
[ ] asus.describe(include='all')
```

	Date	Open	High	Low	Close	Adj Close	Volume	Company	Year	Month	Day
count	5746	5746.000000	5746.000000	5746.000000	5746.000000	5746.000000	5.746000e+03	5746.0	5746.000000	5746.000000	5746.000000
unique	5746	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	2000-05-01 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first	2000-01-02 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-12-06 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	290.489678	293.667370	287.055492	290.235818	128.518523	1.016665e+09	1.0	2011.146189	6.594849	15.825792
std	NaN	75.957579	76.739552	74.935856	75.602517	66.079585	2.177426e+09	0.0	6.866389	3.410648	8.726685
min	NaN	127.106941	130.196335	127.106941	130.196335	28.863441	0.000000e+00	1.0	2000.000000	1.000000	1.000000
25%	NaN	234.528175	237.000000	231.808762	234.500000	76.379619	1.696000e+06	1.0	2005.000000	4.000000	8.000000
50%	NaN	278.000000	280.000000	275.409851	278.000000	120.188485	3.186387e+06	1.0	2011.000000	7.000000	16.000000
75%	NaN	330.087112	334.000000	326.500000	330.000000	163.233246	8.475086e+08	1.0	2017.000000	10.000000	23.000000
max	NaN	567.667419	575.104126	547.836243	565.188538	314.543518	2.833812e+10	1.0	2023.000000	12.000000	31.000000

Team_ID - 592679

```
intel.describe(include='all')
```

	Date	Open	High	Low	Close	Adj Close	Volume	Company	Year	Month	Day
count	10919	10919.000000	10919.000000	10919.000000	10919.000000	10919.000000	1.091900e+04	10919.0	10919.000000	10919.000000	10919.000000
unique	10919	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	1980-03-18 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first	1980-01-04 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-12-06 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	19.897342	20.169676	19.627548	19.896781	14.668655	5.052754e+07	2.0	2001.338126	6.556186	15.708856
std	NaN	17.487968	17.729974	17.252020	17.487397	14.781238	3.481933e+07	0.0	12.509742	3.419732	8.780794
min	NaN	0.218750	0.218750	0.216146	0.216146	0.122972	0.000000e+00	2.0	1980.000000	1.000000	1.000000
25%	NaN	1.343750	1.367188	1.320313	1.343750	0.764502	2.713025e+07	2.0	1991.000000	4.000000	8.000000
50%	NaN	20.350000	20.650000	20.093750	20.370001	12.680091	4.450540e+07	2.0	2001.000000	7.000000	16.000000
75%	NaN	30.115001	30.593750	29.670000	30.066250	19.987983	6.467910e+07	2.0	2012.000000	10.000000	23.000000
max	NaN	75.625000	75.828125	73.625000	74.875000	63.348770	5.677088e+08	2.0	2023.000000	12.000000	31.000000

```
msi.describe(include='all')
```

	Date	Open	High	Low	Close	Adj Close	Volume	Company	Year	Month	Day
count	15484	15484.000000	15484.000000	15484.000000	15484.000000	15484.000000	1.548400e+04	15484.0	15484.000000	15484.000000	15484.000000
unique	15484	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	1962-03-01 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first	1962-01-02 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-12-06 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	45.915441	47.352928	46.153413	46.760641	38.499299	1.992720e+06	3.0	1992.285456	6.543012	15.696719
std	NaN	56.804625	56.774068	55.509647	56.157440	53.186087	2.344350e+06	0.0	17.735967	3.426462	8.786082
min	NaN	0.000000	0.866821	0.808196	0.845884	0.375586	0.000000e+00	3.0	1962.000000	1.000000	1.000000
25%	NaN	3.806477	5.188366	5.050177	5.119272	2.622875	5.086000e+05	3.0	1977.000000	4.000000	8.000000
50%	NaN	24.007187	24.271002	23.630306	23.988343	16.508859	1.288600e+06	3.0	1992.000000	7.000000	16.000000
75%	NaN	67.536697	68.290459	66.732921	67.405529	52.430208	2.622858e+06	3.0	2008.000000	10.000000	23.000000
max	NaN	298.500000	299.429993	297.140015	297.450012	296.515137	4.717163e+07	3.0	2023.000000	12.000000	31.000000

```
nvidia.describe(include='all')
```

	Date	Open	High	Low	Close	Adj Close	Volume	Company	Year	Month	Day
count	6154	6154.000000	6154.000000	6154.000000	6154.000000	6154.000000	6.154000e+03	6154.0	6154.000000	6154.000000	6154.000000
unique	6154	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	1999-01-25 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first	1999-01-02 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-12-06 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	34.055888	34.707315	33.394796	34.080465	33.818979	6.120887e+07	4.0	2010.792168	6.541761	15.688008
std	NaN	67.420090	68.760909	66.069289	67.472837	67.479411	4.385313e+07	0.0	7.064137	3.417286	8.808258
min	NaN	0.348958	0.355469	0.333333	0.341146	0.313002	1.968000e+06	4.0	1999.000000	1.000000	1.000000
25%	NaN	2.682084	2.768125	2.612500	2.685417	2.463874	3.443320e+07	4.0	2005.000000	4.000000	8.000000
50%	NaN	4.371250	4.443750	4.280000	4.367500	4.024390	5.136085e+07	4.0	2011.000000	7.000000	16.000000
75%	NaN	33.498124	34.356876	32.490626	33.403123	33.137190	7.449690e+07	4.0	2017.000000	9.000000	23.000000
max	NaN	435.010010	439.899994	426.739990	438.079987	438.079987	9.230856e+08	4.0	2023.000000	12.000000	31.000000

Team_ID - 592679

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, andoutliers in the data, whichcan help to gain insightsand make informeddecisions.

Activity 2.1: Univariate analysis

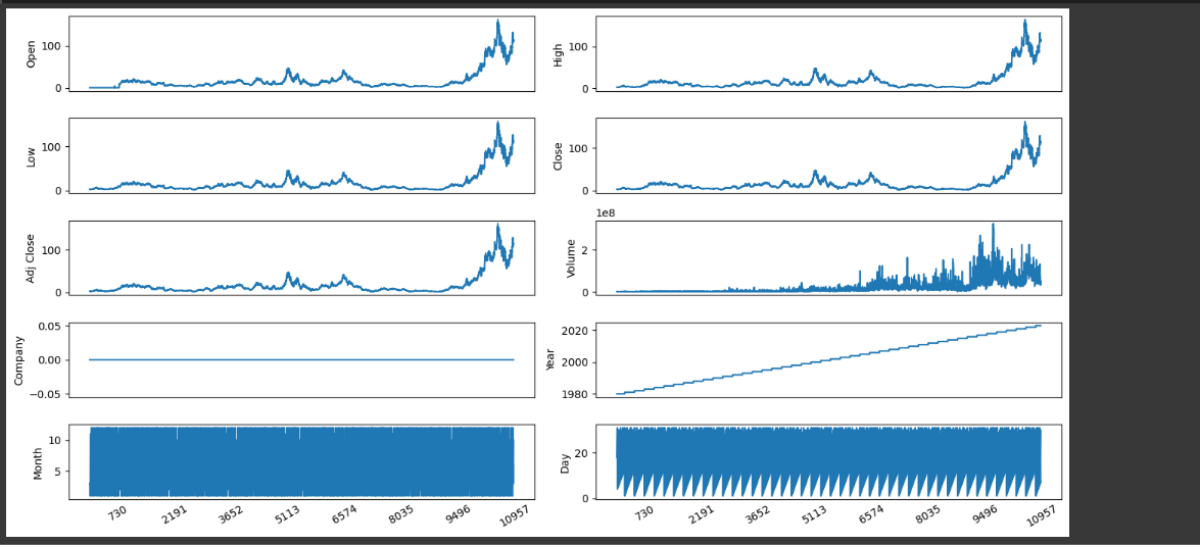
In simple words, univariate analysis is understanding the data with single feature. Here we don't have much need to performunivariate analysis to understand the data as most of the columnsprovided are continuous.

Activity 2.2: Bivariate analysis

To find the relationbetween two featureswe use bivariate analysis.

Bivariate Analysis for AMD stock data set

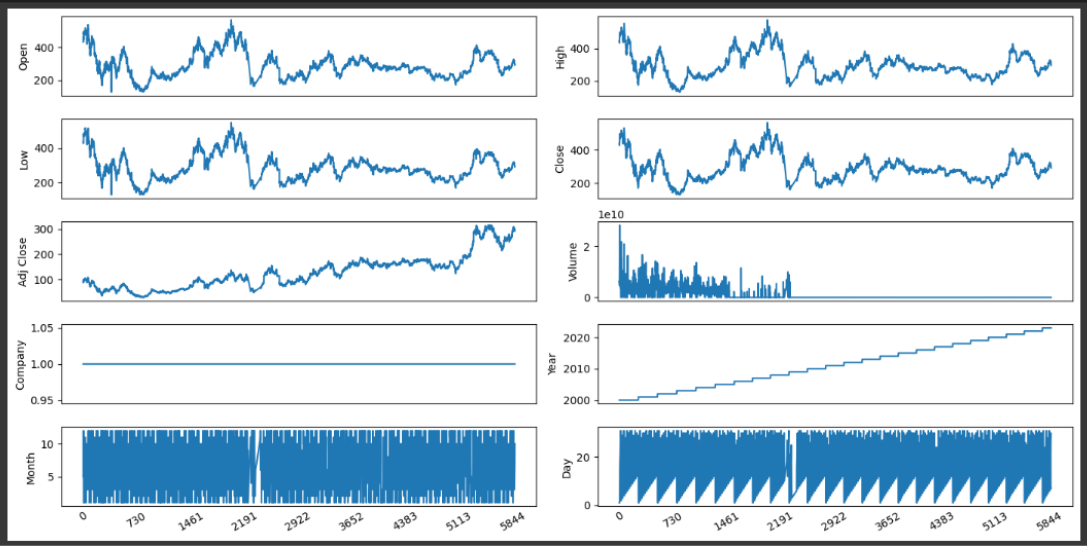
```
import matplotlib.dates as mdates
#Bivariate Analysis for AMD stock data set
df_plot=amd.drop(columns=['Date'])
ncols=2
nrows=int(round(df_plot.shape[1]/ncols,0))

fig,ax=plt.subplots(nrows=nrows,ncols=ncols,sharex=True,figsize=(14,7))
for i,ax in enumerate(fig.axes):
    sns.lineplot(data=df_plot.iloc[:,i],ax=ax)
    ax.tick_params(axis="x",rotation=30,labelsize=10,length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()
plt.show()
```



Team_ID - 592679

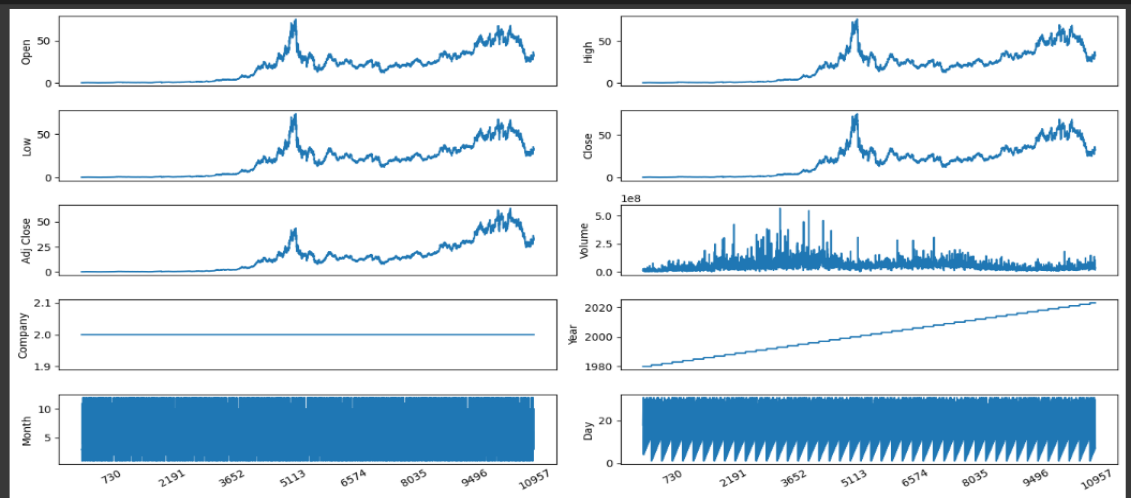
Bivariate Analysis for Asus stock data set

```
import matplotlib.dates as mdates
#Bivariate Analysis for Asus stock data set
df_plot=asus.drop(columns=['Date'])
ncols=2
nrows=int(round(df_plot.shape[1]/ncols,0))

fig,ax=plt.subplots(nrows=nrows,ncols=ncols,sharex=True,figsize=(14,7))
for i,ax in enumerate(fig.axes):
    sns.lineplot(data=df_plot.iloc[:,i],ax=ax)
    ax.tick_params(axis="x",rotation=30,labelsize=10,length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()
plt.show()
```



Bivariate Analysis for Intel stock data set

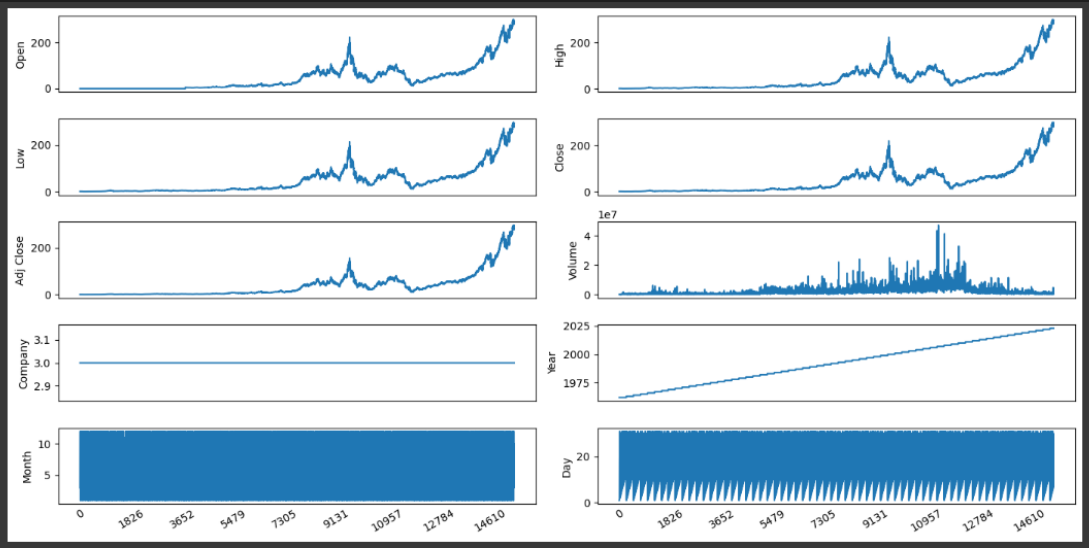
```
[ ] import matplotlib.dates as mdates
#Bivariate Analysis for Intel stock data set
df_plot=intel.drop(columns=['Date'])
ncols=2
nrows=int(round(df_plot.shape[1]/ncols,0))

fig,ax=plt.subplots(nrows=nrows,ncols=ncols,sharex=True,figsize=(14,7))
for i,ax in enumerate(fig.axes):
    sns.lineplot(data=df_plot.iloc[:,i],ax=ax)
    ax.tick_params(axis="x",rotation=30,labelsize=10,length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()
plt.show()
```



Team_ID - 592679

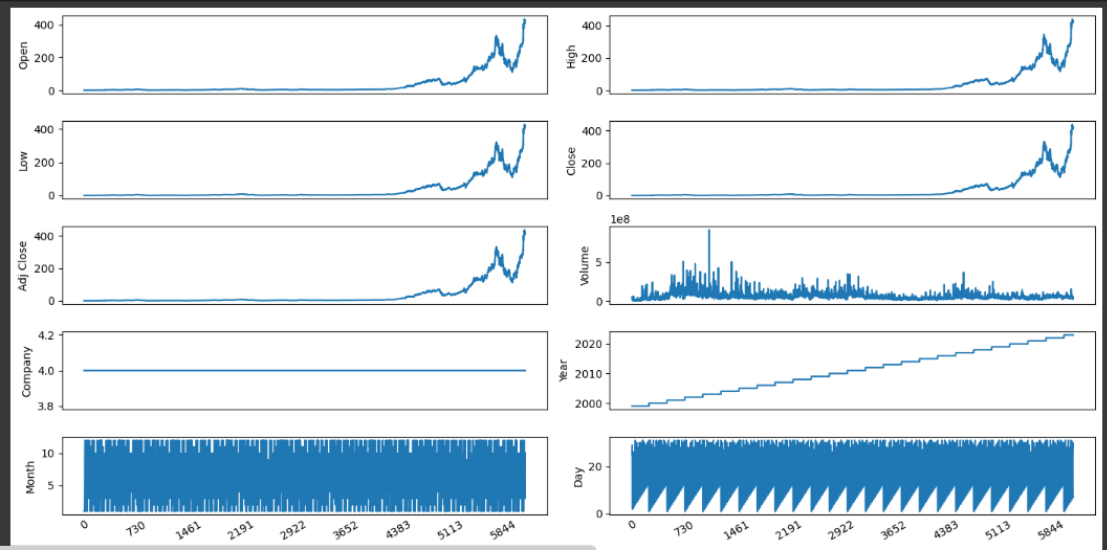
Bivariate Analysis for MSI stock data set

```
import matplotlib.dates as mdates
#Bivariate Analysis for MSI stock data set
df_plot=msi.drop(columns=['Date'])
ncols=2
nrows=int(round(df_plot.shape[1]/ncols,0))

fig,ax=plt.subplots(nrows=nrows,ncols=ncols,sharex=True,figsize=(14,7))
for i,ax in enumerate(fig.axes):
    sns.lineplot(data=df_plot.iloc[:,i],ax=ax)
    ax.tick_params(axis="x",rotation=30,labelsize=10,length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()
plt.show()
```



Bivariate Analysis for Nvidia stock data set

```
import matplotlib.dates as mdates
#Bivariate Analysis for Nvidia stock data set
df_plot=nvidia.drop(columns=['Date'])
ncols=2
nrows=int(round(df_plot.shape[1]/ncols,0))

fig,ax=plt.subplots(nrows=nrows,ncols=ncols,sharex=True,figsize=(14,7))
for i,ax in enumerate(fig.axes):
    sns.lineplot(data=df_plot.iloc[:,i],ax=ax)
    ax.tick_params(axis="x",rotation=30,labelsize=10,length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()
plt.show()
```



Team_ID - 592679

Milestone 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying seven classification algorithms. The best model is savedbased on its performance

Model Building

```
[ ]  #Model Building
     #Linear Regression Model
     lr=LinearRegression()
     lr.fit(x_train,y_train)

     print('Test Score:',lr.score(x_test,y_test))
     print('Train Score',lr.score(x_train,y_train))

     y_pred=lr.predict(x_test)
     print('R2 Score: ',r2_score(y_test,y_pred))
     print('MAE :', mean_absolute_error(y_test,y_pred))
```

```
Test Score: 0.9998391313964322
Train Score 0.9998961452109035
R2 Score:  0.9998391313964322
MAE : 0.7043975850427657
```

Team_ID - 592679

```
[ ] from sklearn.tree import DecisionTreeRegressor
    #Decision Tree Regression Model
    dt=DecisionTreeRegressor()
    dt.fit(x_train,y_train)

    print('Test Score:',dt.score(x_test,y_test))
    print('Train Score',dt.score(x_train,y_train))

    y_pred=dt.predict(x_test)
    print('R2 Score: ',r2_score(y_test,y_pred))
    print('MAE :', mean_absolute_error(y_test,y_pred))
```

```
Test Score: 0.9996205086899826
Train Score 1.0
R2 Score:  0.9996205086899826
MAE : 1.0154897875279303
```

```
[ ] #Extra Tree Regression
    from sklearn.tree import ExtraTreeRegressor
    etr=ExtraTreeRegressor()
    etr.fit(x_train,y_train)

    print('Test Score:',etr.score(x_test,y_test))
    print('Train Score',etr.score(x_train,y_train))

    y_pred=etr.predict(x_test)
    print('R2 Score: ',r2_score(y_test,y_pred))
    print('MAE :', mean_absolute_error(y_test,y_pred))
```

```
Test Score: 0.9995226382674991
Train Score 1.0
R2 Score:  0.9995226382674991
MAE : 1.1272517545196017
```

Team_ID - 592679

```
[ ]  #Random Forest Regression

     from sklearn.ensemble import RandomForestRegressor

     rf=RandomForestRegressor()
     rf.fit(x_train,y_train)

     print('Test Score:',rf.score(x_test,y_test))
     print('Train Score',rf.score(x_train,y_train))

     y_pred=rf.predict(x_test)
     print('R2 Score: ',r2_score(y_test,y_pred))
     print('MAE :', mean_absolute_error(y_test,y_pred))
```

```
Test Score: 0.9998017473735594
Train Score 0.99997241510103
R2 Score:  0.9998017473735594
MAE : 0.7402542805982135
```

Model comparison and evaluating best model

From the observed r2 scores and Mena absolute errors we can clearly see that the linear regression modelhas least Mean absolute erroramong others. So we are going to select LinearRegression model for final Flask application deployment.

To plot the predictions over the originalvalues of all the companieswe can use the following code.

First, we are going to store the dates, original closing prices of stocks and predicted closing prices ofstocks as shown below. We are going to access company specific rows of test_data and x_test variables using the “Company” column.

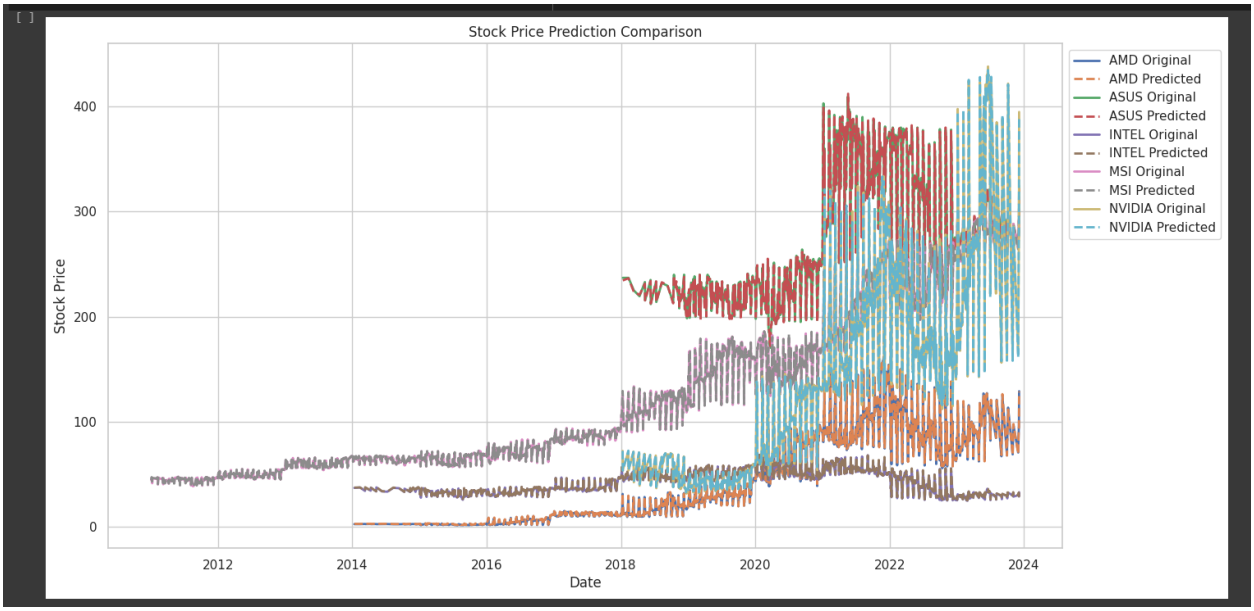
Team_ID - 592679

```
[ ]  amd_dates=test_data[test_data[ 'Company']==0]['Date']
     amd_pred= lr.predict (x_test [x_test['Company']==0])
     amd_orig = test_data[test_data['Company']==0]["Close"]

     asus_dates=test_data[test_data[ 'Company']==1]['Date']
     asus_pred= lr.predict (x_test [x_test['Company']==1])
     asus_orig = test_data[test_data['Company']==1]["Close"]

     intel_dates=test_data[test_data[ 'Company']==2]['Date']
     intel_pred= lr.predict (x_test [x_test['Company']==2])
     intel_orig = test_data[test_data['Company']==2]["Close"]

     msi_dates=test_data[test_data[ 'Company']==3]['Date']
     msi_pred= lr.predict (x_test [x_test['Company']==3])
     msi_orig = test_data[test_data['Company']==3]["Close"]

     nvidia_dates=test_data[test_data[ 'Company']==4]['Date']
     nvidia_pred= lr.predict (x_test [x_test['Company']==4])
     nvidia_orig = test_data[test_data['Company']==4]["Close"]
```

Team_ID - 592679

Plot predictions and actual values

```
[ ]  import matplotlib.pyplot as plt
     import seaborn as sns

     def plot_data(dates, original, predicted, label):
         sns.lineplot(x=dates, y=original, label=f'{label} Original', linestyle='solid', linewidth=2)
         sns.lineplot(x=dates, y=predicted, label=f'{label} Predicted', linestyle='dashed', linewidth=2)

     # Set a more professional color palette
     sns.set_palette("husl")

     # Assuming you have a dictionary containing your data
     data_sets = {
         'AMD': {'dates': amd_dates, 'original': amd_orig, 'predicted': amd_pred},
         'ASUS': {'dates': asus_dates, 'original': asus_orig, 'predicted': asus_pred},
         'INTEL': {'dates': intel_dates, 'original': intel_orig, 'predicted': intel_pred},
         'MSI': {'dates': msi_dates, 'original': msi_orig, 'predicted': msi_pred},
         'NVIDIA': {'dates': nvidia_dates, 'original': nvidia_orig, 'predicted': nvidia_pred},
     }

     # Set a white background for better readability
     sns.set(style="whitegrid")

     plt.figure(figsize=(15, 8))

     for brand, data in data_sets.items():
         plot_data(data['dates'], data['original'], data['predicted'], brand)

     plt.title('Stock Price Prediction Comparison')
     plt.xlabel('Date')
     plt.ylabel('Stock Price')
     plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
     plt.show()
```



Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. An UI is provided for the uses where he has to enter the values for predictions. The enter valuesare given to the savedmodel and prediction is showcased on the UI.

This sectionhas the following tasks

1. Building HTML Pages
2. Building server-side script
3. Run the web application

Activity 2.1: Building Html Pages:

For this project createtwo HTML files namely

a. index.html

b. inner-page.html

c. portfolio-details.html

and save them in the templatesfolder.

It is not necessary to follow the exact format as above so feel

free to use whatevertemplates or formatyou like. Be creative!

Activity 2.2: Build Python Application(Flask) code:

app.py | predict.html | PriceEstimationModel.ipynb | AMD (1980 -11.07.2023).csv | contact.html | index.html | about.html | style.css

app.py > contact

```python
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open("lr.pkl","rb"))

@app.route('/')
@app.route('/home')
@app.route('/index')
@app.route('/index.html')
def index():
    return render_template('index.html')

@app.route('/about')
@app.route('/about.html')
def about():
    return render_template('about.html')

@app.route('/contact')
@app.route('/contact.html')
def contact():
    return render_template('contact.html')


@app.route("/predict.html", methods=['GET','POST'])
@app.route("/predict", methods=['GET','POST'])
def predict():
    if request.method == 'POST':
        low = float(request.form["low"])
        high = float(request.form['high'])
        volume = float(request.form['volume'])
        open = float(request.form['open'])
        company = eval(request.form['company'])
        year = int(request.form["year"])
        month = int(request.form['month'])
        day = int(request.form['day'])
        prediction = model.predict([[open, high, low, volume, year, month, day, company]])
        output = prediction[0]
        return render_template("predict.html", prediction="Forecasted closing price on {}/{}/{} is $ {}".format(day, month, year, output))
    else:
        return render_template("predict.html")


if __name__ == '__main__':
    app.run(debug=False)
```

Run the web application

i. Open anaconda promptfrom the startmenu

ii. Navigate to the folder where your pythonscript is.

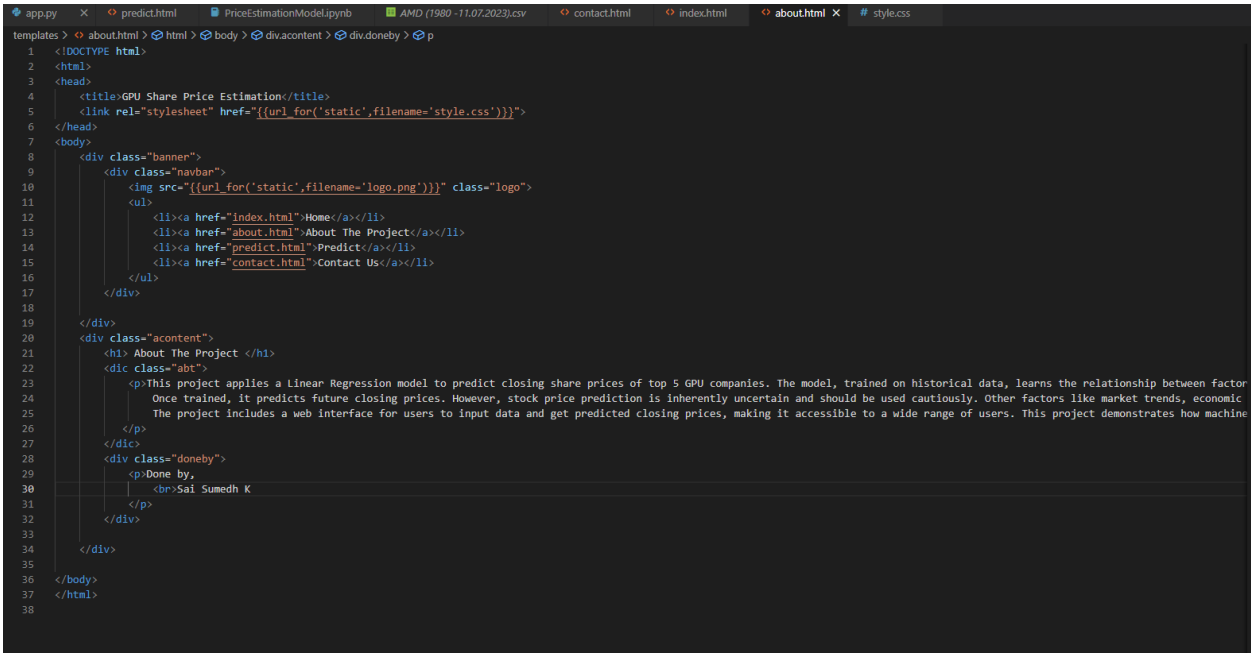
iii. Now type “pythonapp.py” command

iv. Navigate to the localhost where you can view your web page.

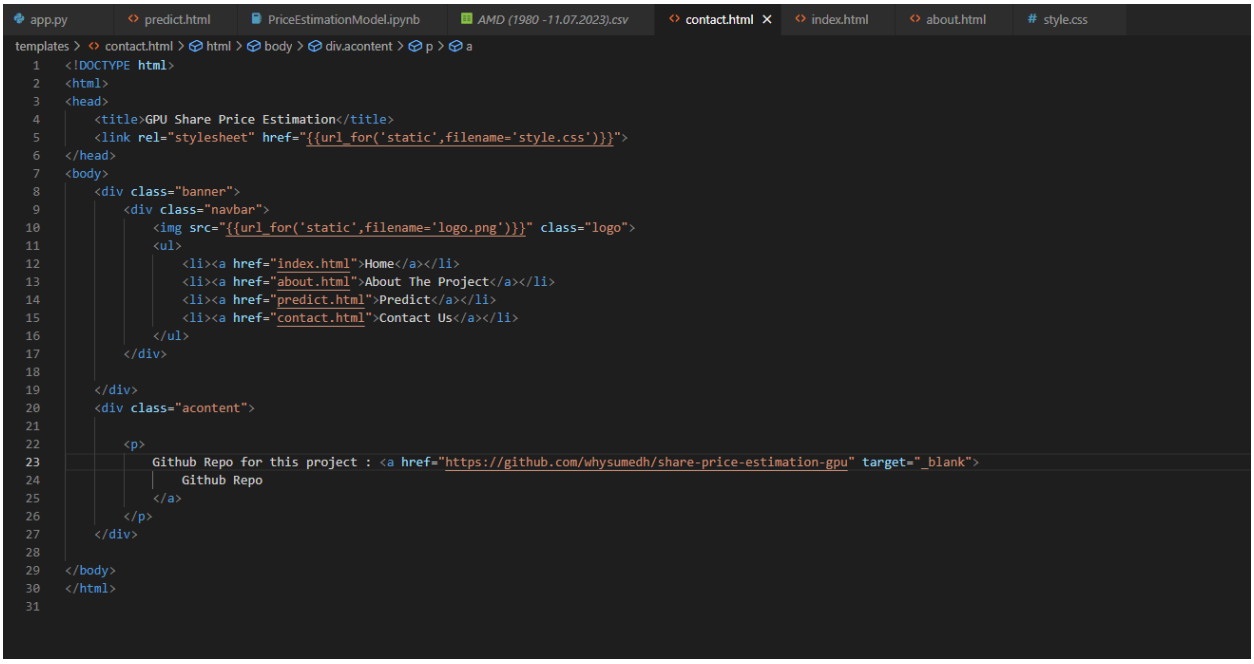
v. Click on the predictbutton from the top left corner, enterthe inputs, click on the submit button,and see the result/prediction on the web.

Team_ID - 592679

HTML Pages:



```
templates > about.html > html > body > div.acontent > div.doneby > p
<!DOCTYPE html>
<html>
<head>
    <title>GPU Share Price Estimation</title>
    <link rel="stylesheet" href="{{url_for('static',filename='style.css')}}">
</head>
<body>
    <div class="banner">
        <div class="navbar">
            <img src="{{url_for('static',filename='logo.png')}}" class="logo">
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="about.html">About The Project</a></li>
                <li><a href="predict.html">Predict</a></li>
                <li><a href="contact.html">Contact Us</a></li>
            </ul>
        </div>

    </div>
    <div class="acontent">
        <h1> About The Project </h1>
        <dic class="abt">
            <p>This project applies a Linear Regression model to predict closing share prices of top 5 GPU companies. The model, trained on historical data, learns the relationship between factor
                Once trained, it predicts future closing prices. However, stock price prediction is inherently uncertain and should be used cautiously. Other factors like market trends, economic
                The project includes a web interface for users to input data and get predicted closing prices, making it accessible to a wide range of users. This project demonstrates how machine
            </p>
        </dic>
        <div class="doneby">
            <p>Done by,
                <br>Sai Sumedh K
            </p>
        </div>

    </div>

</body>
</html>
```



```
templates > contact.html > html > body > div.acontent > p > a
<!DOCTYPE html>
<html>
<head>
    <title>GPU Share Price Estimation</title>
    <link rel="stylesheet" href="{{url_for('static',filename='style.css')}}">
</head>
<body>
    <div class="banner">
        <div class="navbar">
            <img src="{{url_for('static',filename='logo.png')}}" class="logo">
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="about.html">About The Project</a></li>
                <li><a href="predict.html">Predict</a></li>
                <li><a href="contact.html">Contact Us</a></li>
            </ul>
        </div>

    </div>
    <div class="acontent">

        <p>
            Github Repo for this project : <a href="https://github.com/whysumedh/share-price-estimation-gpu" target="_blank">
                Github Repo
            </a>
        </p>
    </div>

</body>
</html>
```

Team_ID - 592679

app.py | predict.html | PriceEstimationModel.ipynb | AMD (1980 -11.07.2023).csv | contact.html | index.html | about.html | style.css

templates > index.html > html > body

```
<!DOCTYPE html>
<html>
<head>
    <title>GPU Share Price Estimation</title>
    <link rel="stylesheet" href="{{url_for('static',filename='style.css')}}">
</head>
<body>
    <div class="banner">
        <div class="navbar">
            <img src="{{url_for('static',filename='logo.png')}}" class="logo">
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="about.html">About The Project</a></li>
                <li><a href="predict.html">Predict</a></li>
                <li><a href="contact.html">Contact Us</a></li>
            </ul>
        </div>

    </div>
    <div class="content">
        <h1> SHARE PRICE ESTIMATION OF GPU COMPANIES </h1>
        <p>This webpage enables you to predict the forecasted closing share price
            of any one of the top 5 GPU company.
        </p>

    </div>

</body>
</html>

```

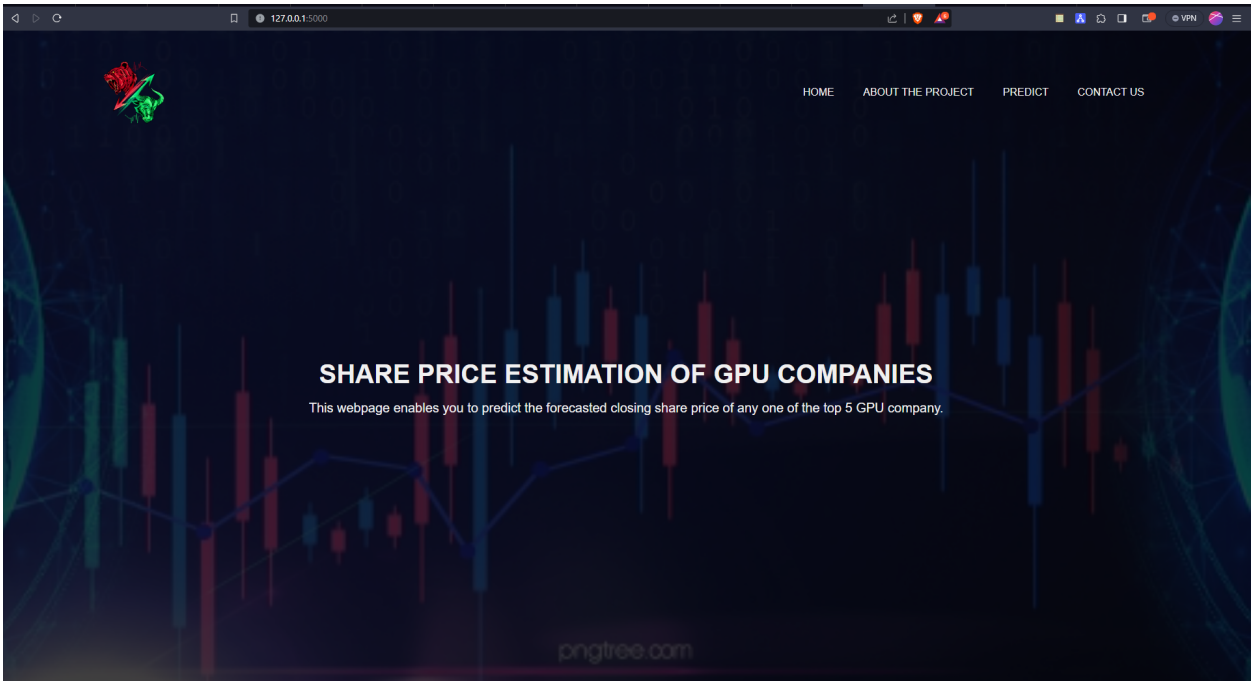
app.py | predict.html | PriceEstimationModel.ipynb | AMD (1980 -11.07.2023).csv | contact.html | index.html | about.html | style.css

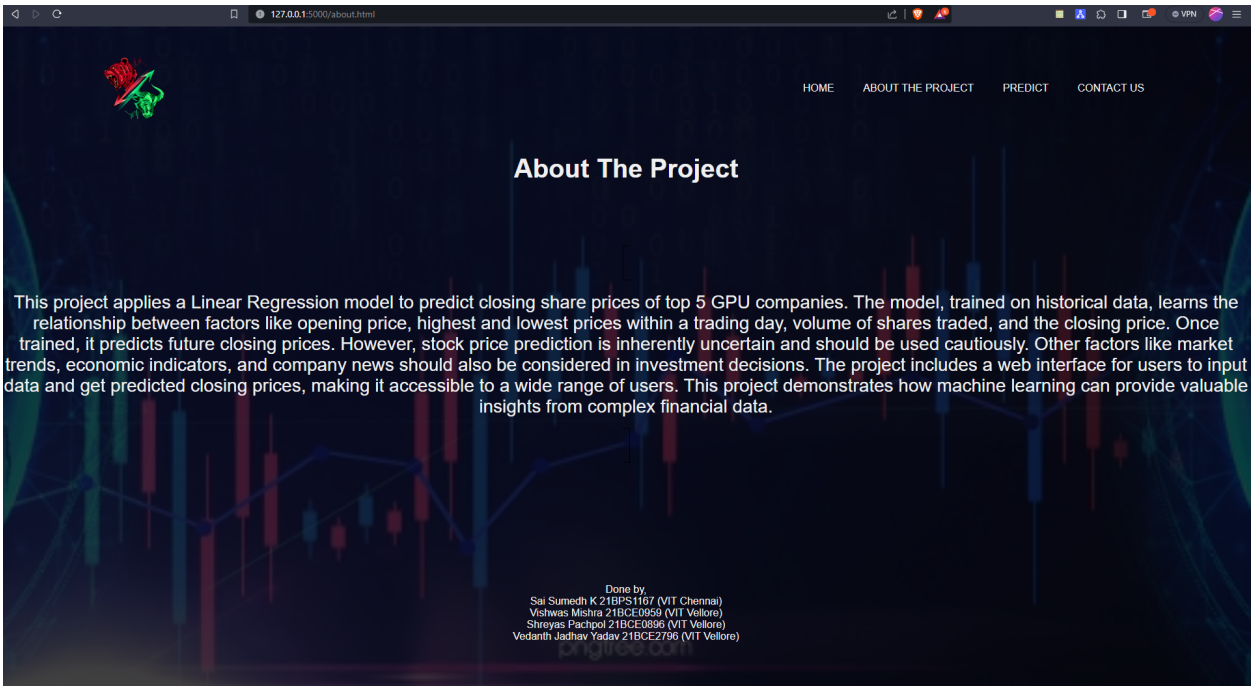
templates > predict.html > html > body > div.banner > div.navbar > img.logo

```
<!DOCTYPE html>
<html>
<head>
    <title>GPU Share Price Estimation</title>
    <link rel="stylesheet" href="{{url_for('static',filename='style.css')}}">
</head>
<body>
    <div class="banner">
        <div class="navbar">
            <img src="{{url_for('static',filename='logo.png')}}" class="logo">
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="about.html">About The Project</a></li>
                <li><a href="predict.html">Predict</a></li>
                <li><a href="contact.html">Contact Us</a></li>
            </ul>
        </div>
    </div>
    <div class="form-container">
        <h1>GPU Company Share Price Estimation</h1>
        <form action="/predict" method="post">
            <label for="low">Low:</label><br>
            <input type="text" id="low" name="low"><br>
            <label for="high">High:</label><br>
            <input type="text" id="high" name="high"><br>
            <label for="volume">Volume:</label><br>
            <input type="text" id="volume" name="volume"><br>
            <label for="open">Open:</label><br>
            <input type="text" id="open" name="open"><br>
            <label for="company">Company:</label><br>
            <select id="company" name="company">
                <option value="0">AMD</option>
                <option value="1">Asus</option>
                <option value="2">Intel</option>
                <option value="3">MSI</option>
                <option value="4">Nvidia</option>
            </select><br>
            <label for="year">Year:</label><br>
            <input type="text" id="year" name="year"><br>
            <label for="month">Month:</label><br>
            <input type="text" id="month" name="month"><br>
            <label for="day">Day:</label><br>
            <input type="text" id="day" name="day"><br>
            <input type="submit" value="Submit">
        </form>
    </div>

    {% if prediction %}
    <div class="prediction-box">
        <h2>{{ prediction }}</h2>
    </div>
```

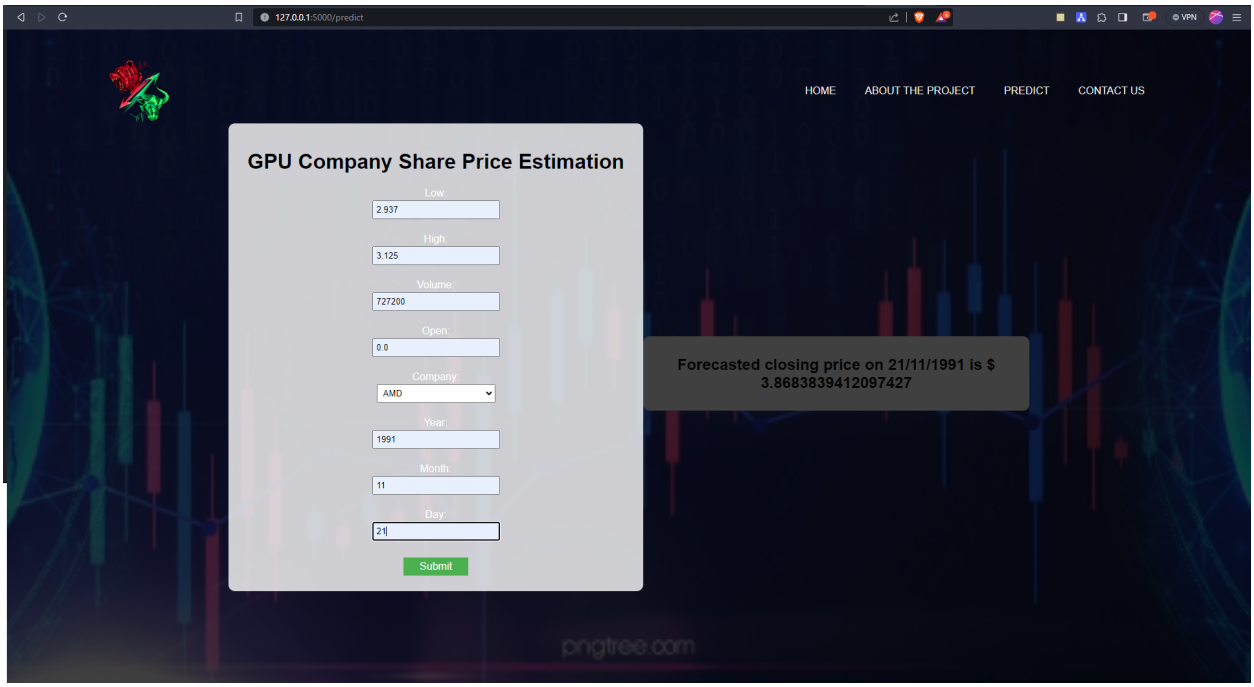
Team_ID - 592679

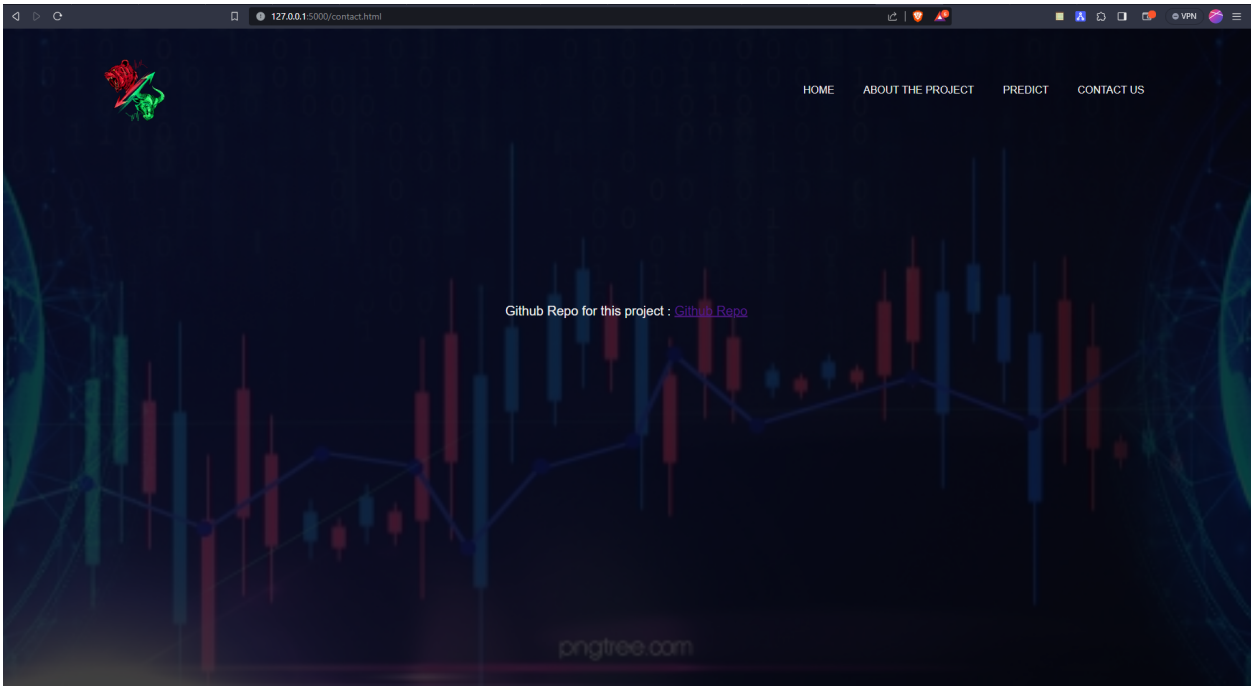
Website Pages:





Team_ID - 592679





Github Link: Github Repo

Team_ID - 592679

Style, CSS Code:

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-image: url('background.jpg');
    background-size: cover;
    background-position: center;
}
.banner{
    position: absolute;
    width: 100%;
    height: 100vh;
    background-image: linear-gradient(rgba(0,0,0,0.75),rgba(0,0,0,0.75));
    background-size: cover;
    background-position: center;
}
.navbar{
    width: 85%;
    margin: auto;
    padding: 35px 0;
    display: flex;
    align-items: center;
    justify-content: space-between;
}
.logo{
    width: 120px;
    cursor:pointer;
}
.navbar ul li{
    list-style: none;
    display: inline-block;
    margin:0 20px;
}
```

```
.navbar ul li a{
    text-decoration: none;
    color: #f0f0f0;
    text-transform: uppercase;
}
.content{
    width: 100%;
    position: absolute;
    top: 50%;
    transform: translateY(-50%);
    text-align: center;
    color: #f0f0f0;
}
.content{
    font-size: 20px;
    margin-top: 30px;
}
.content p{
    margin: 10px auto;
    font-weight: 100;
    line-height: 5px;
}
.acontent{
    width: 100%;
    position: absolute;
    top: 40%;
    transform: translateY(-50%);
    text-align: center;
    color: #f0f0f0;
}
.acontent{
    font-size: 20px;
    margin-top: 30px;
}

.form-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
```

```
    position: relative;
    z-index: 0.5;
    width: 40%;
    max-width: 600px;
    background-color: rgba(255, 255, 255, 0.8);
    padding: 20px;
    border-radius: 10px;
}

form {
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 2px;
}
form input[type="text"], form select {
    width: 100%;
    padding: 5px;
}
form label {
    margin-bottom: -20px;
    color: #ffffff
                }

                form input[type="submit"] {
                    width: 100px;
                    padding: 5px;
                    cursor: pointer;
                    background-color: #4CAF50;
                    border: none;
                    color: white;
                    text-align: center;
                    text-decoration: none;
                    display: inline-block;
                    font-size: 16px;
                    margin: 4px 2px;
                    transition-duration: 0.4s;
                }
                form input[type="submit"]:hover {
                    background-color: #45a049;
```

```
118                 }
119                 .prediction-box {
120                     background-color: #ffffff;
121                     padding: 10px;
122                     border-radius: 10px;
123                     margin-top: 50px;
124                     width: 30%;
125                     text-align: center;
126                 }
127 
128                 .prediction-box h2 {
129                     color: #0c0c0c;
130                 }
131                 .doneby{
132                     position: relative;
133                     bottom: -260px;
134                     font-size: 16px;
135                 }
136 
137                 .abt{
138                     position: relative;
139                     font-size: 28px;
140                     bottom:-80px;
141                     width: 50%;
142                     padding: 10px;
143                     border: 1px solid #000;
144                     margin: auto;
145                 }
```