

Share Price Estimation Of TOP 5 GPU Companies

Group Members:

Kevin Noel (21BML0145)

Aditya Byer (21BCE3706)

Srinithi Shabad (21BCE3553)

Hershita Saha (21BML0100)

INTRODUCTION:

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 should take 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.

Project Overview:

Our objective is to develop a predictive model that can forecast the future share prices of the top 5 GPU companies in the market

Methodology:

- Data collection and preparation
- Exploratory data analysis
- Feature engineering
- Model selection and training
- Model evaluation
- Model deployment
- Integration with website

Purpose:

A reliable predictive model for estimating share prices of GPU companies

Insights into the factors that influence share prices in the GPU industry

Potential Applications:

- Stock trading and investment decisions
- Risk assessment and portfolio optimization
- Financial forecasting and market analysis

LITERATURE SURVEY

Upon going through the pre-existing data available on this project and reviewing many similar projects conducted, we have summarised our findings as follows.

Existing problems

- Selection of relevant features- The model should consider a wide range of factors, but it is important to avoid overfitting the model to the training data.
- The model should be able to handle the time-series nature of the data, as share prices fluctuate over time.

The following are some of the key findings from the literature on share price estimation of top 5 GPU companies:

- There is no single best method for estimating share prices. The best method for a particular application will depend on the specific data and modelling approach used.
- Machine learning techniques and deep learning algorithms have the potential to be more accurate than traditional statistical methods. However, these methods can be more difficult to implement and interpret.
- Data quality is critical for developing accurate predictive models. It is important to use clean, consistent, and up-to-date data.
- Feature engineering can play an important role in improving the performance of predictive models.

Sources:

1. Qi Yao, Qiang Zhang, et al. (2018). "A Deep Learning-Based Framework for Stock Price Prediction." IEEE Access, 6, 32834-32844.
2. Prateek Agarwal, Siddhartha Panda, et al. (2022). "Predicting Stock Prices Using Machine Learning Techniques." arXiv preprint arXiv:2208.03841.
3. Amirhossein Zamani, Vahid Rezaei, et al. (2020). "A Hybrid Approach for Stock Market Prediction: Combining Statistical Methods with Machine Learning." Expert Systems with Applications, 140, 112976.

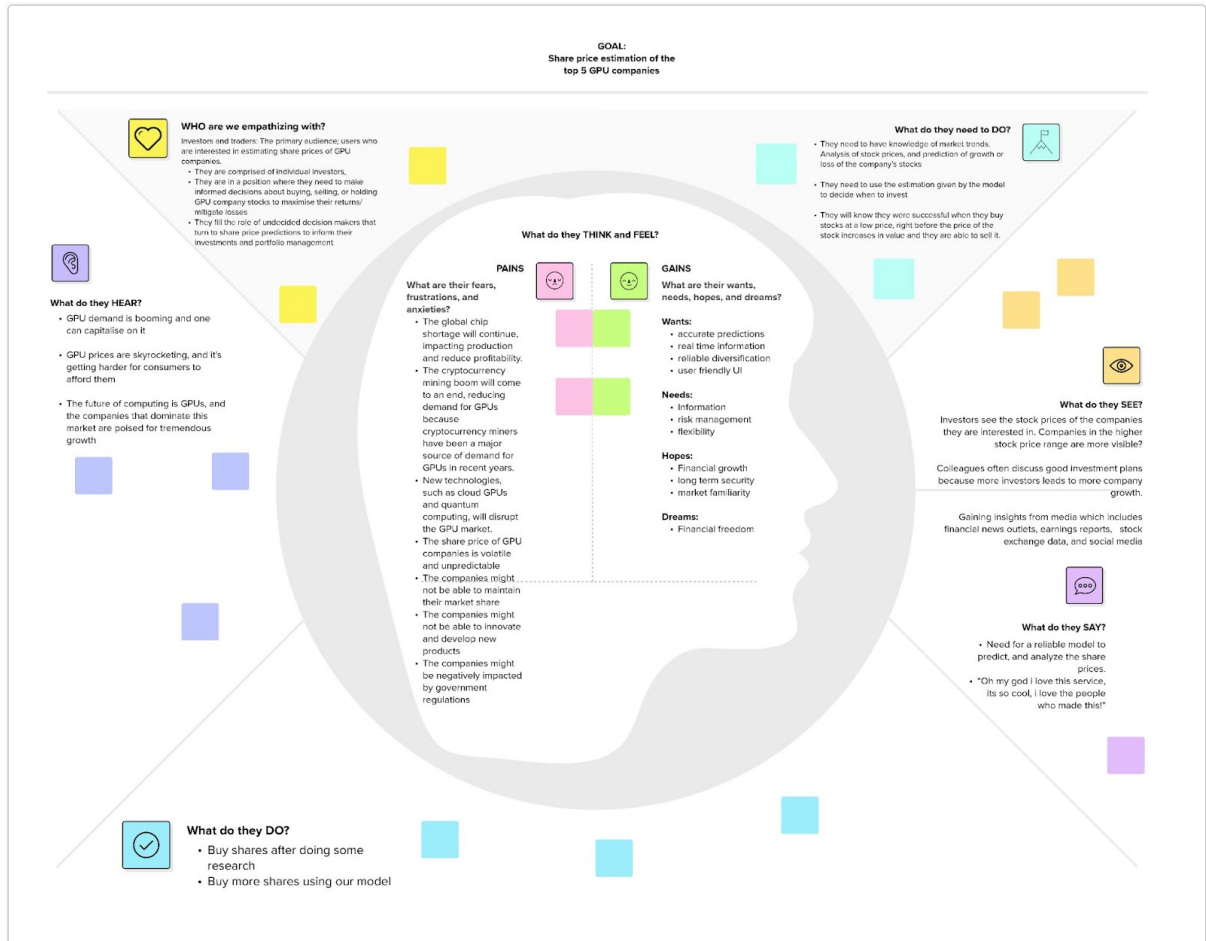
Problem Statement Definition:

To develop a predictive model that can forecast the future share prices of the top 5 GPU companies in the market.

IDEATION & PROPOSED SOLUTION

To train a convolutional neural network model using suitable historical stock price data of the relevant companies, in order to provide reliable predictions for future price fluctuations when the current prices are fed to the model.

Empathy Map Canvas:



1

Define your problem statement

PROBLEM

Estimating share prices of
Top 5 GPU companies

BACKGROUND

In recent years, the explosive growth of the GPU industry has garnered significant attention from new investors. As such, accurate predictions of fluctuations in share prices are of paramount importance to investors breaking ground in a new market. To this end, machine learning could be used to create accurate projections of share prices.

PROBLEM DESCRIPTION

The objective is to create a predictive machine learning model that can extrapolate historical data on GPU share prices in order to accurately forecast future share prices of the top 5 GPU companies.

Ideation & Brainstorming:

2

Brainstorm

Kevin Noel

Work with said companies to acquire product release dates

Listing potential factors that may influence stock prices

Use predictive AI to run trial and error

Study the different industry/use cases for GPUs currently

Srinithi S

Analyzing the pattern in the supplies and demands of the company over the years

Studying and understanding the current economic conditions that can affect the share prices

Aditya B

Establish liasons with industry informants within each of the major GPU companies for insider information on higher up decisionmaking before the rest of the market

bring onboard prior members of the board of directors of various prominent GPU companies in order to glean unique insights on the inner workings and insider deals that occur within GPU companies

Keep up to date on bleeding edge application opportunities and emergent or foreseeable demand that might cause upsets within the marketplace

Hershita Saha

Collecting data on past trends and predicting prices based on patterns found.

Measuring the impact of different factors on the the trends

3

Group ideas

Trends

Measuring the impact of different factors on the trends

Collecting data on past trends and predicting prices based on patterns found.

Analyzing the pattern in the supplies and demands of the company over the years

Study the different industry/use cases for GPUs currently

Artificial Intelligence

Use predictive AI to run trial and error

Data Analysis

Listing potential factors that may influence stock prices

Studying and understanding the current economic conditions

Keep up to date on bleeding edge application opportunities and emergent or foreseeable demand that might cause upsets within the marketplace

Networking

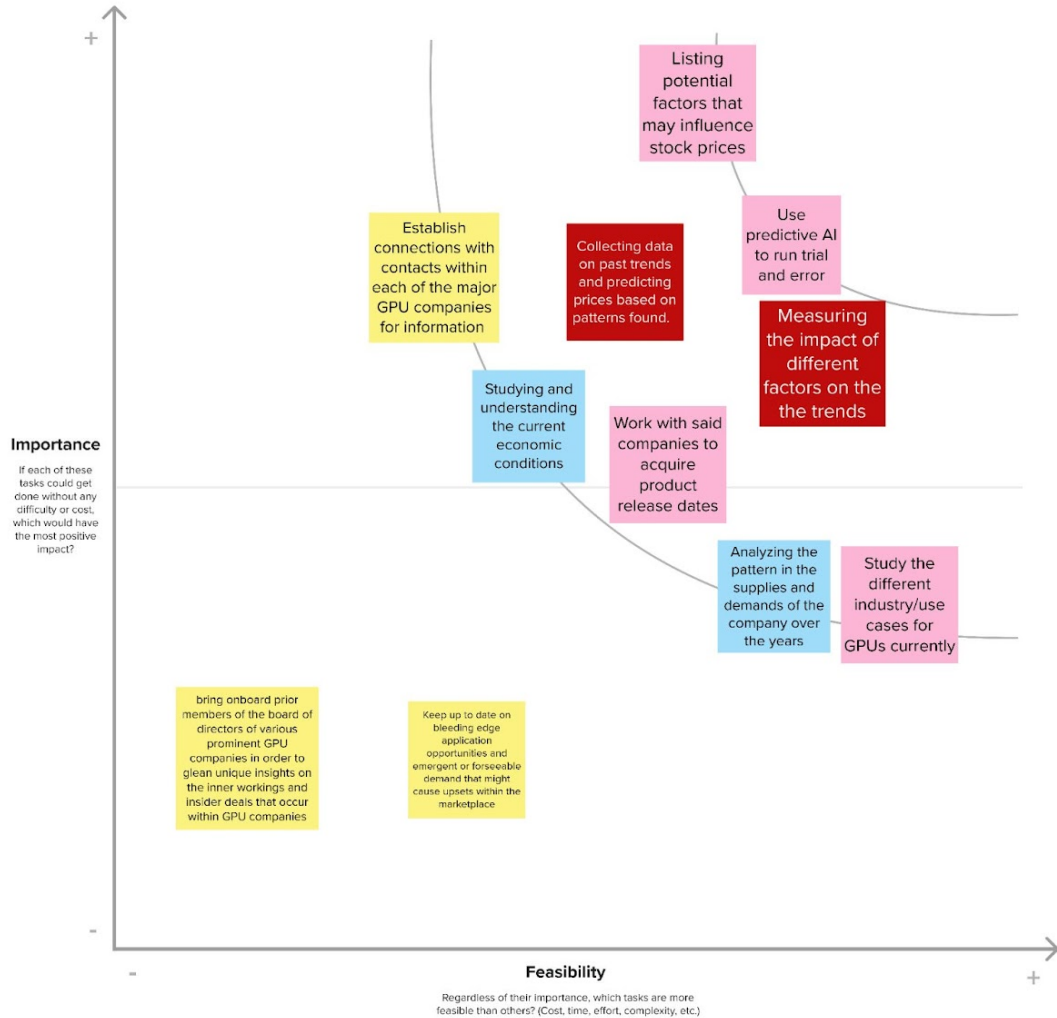
Establish connections with contacts within each of the major GPU companies for information

bring onboard prior members of the board of directors of various prominent GPU companies in order to glean unique insights on the inner workings and insider deals that occur within GPU companies

Work with said companies to acquire product release dates

4

Prioritize



REQUIREMENT ANALYSIS:

Functional Requirements:

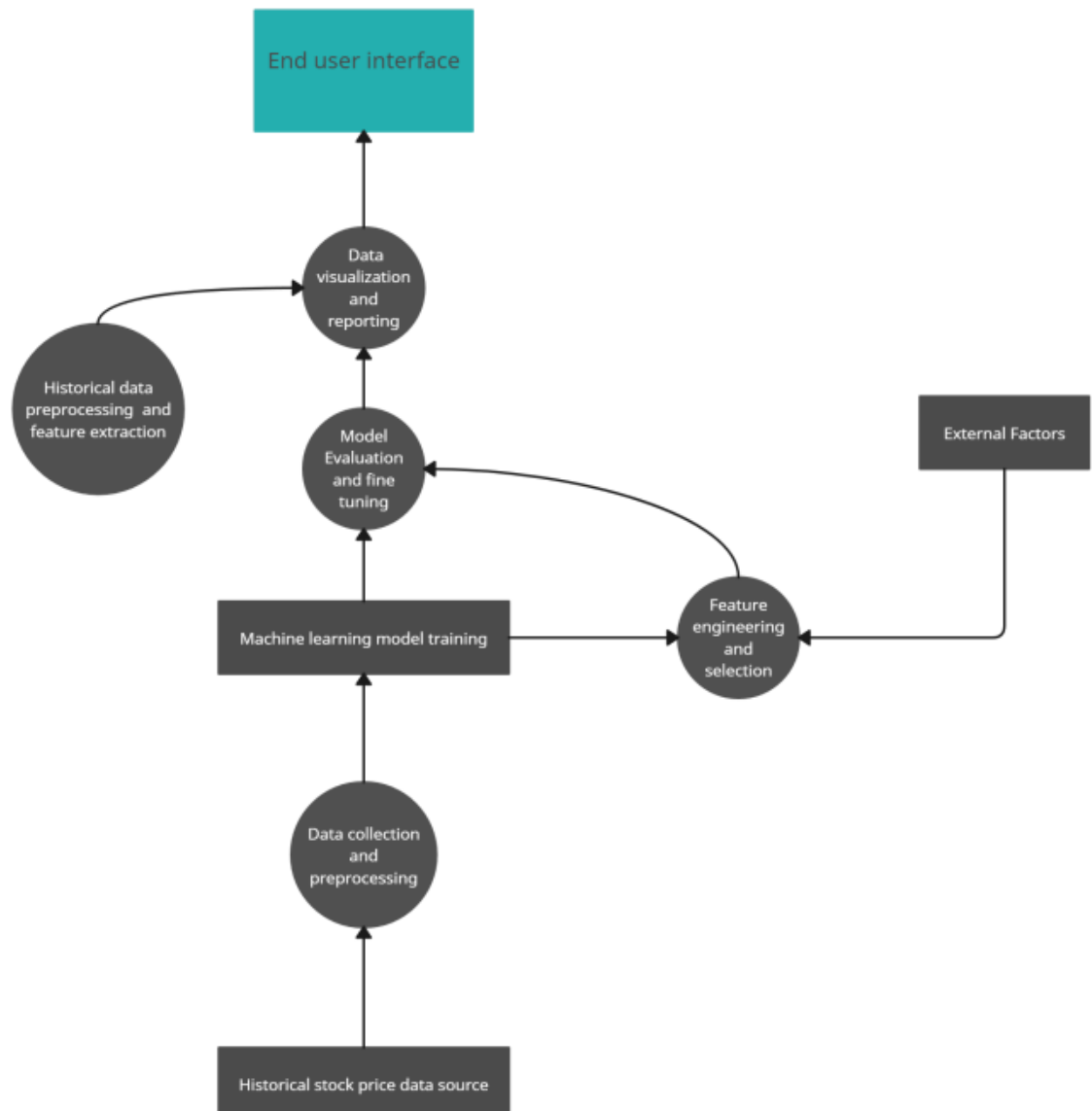
1. The model should be able to predict the future share prices of the top 5 GPU companies in the market.
2. The model should be able to handle time-series data, as share prices fluctuate over time.
3. The model should be able to provide accurate predictions for both short-term and long-term.

Non-Functional Requirements:

1. Performance: The model should be able to generate predictions quickly and efficiently.
2. Accuracy: The model should be able to make accurate predictions.
3. Reliability: The model should be reliable and consistent in its predictions.
4. Scalability: The model should be able to handle large datasets and be easily scaled up to incorporate new data.
5. Interpretability: The model should be interpretable, allowing users to understand how it makes its predictions.

PROJECT DESIGN:

Data Flow Diagrams & User Stories



User Stories

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

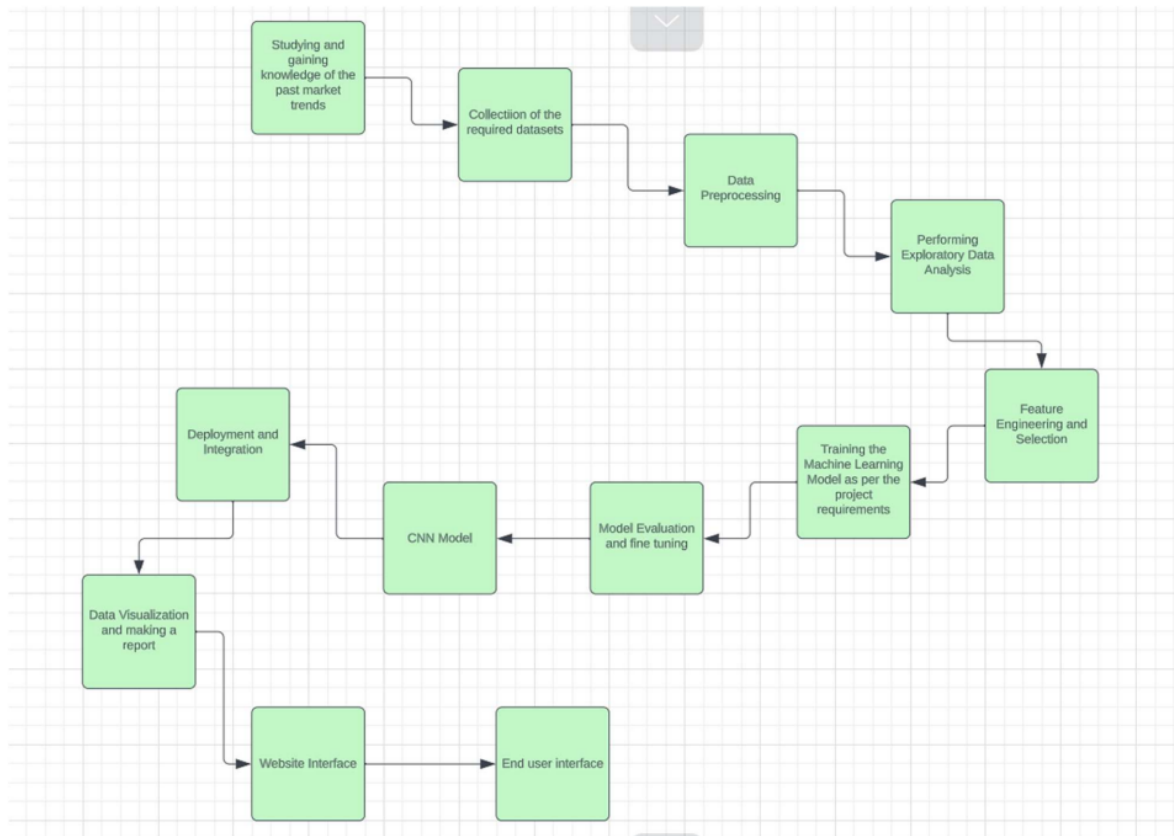
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
	Login	USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
		USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)						

Solution Architecture**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.

Solution Architecture Diagram



S.No	Component	Description	Technology
1	User Interface	The user interacts with the prediction model through a website. The user needs to enter several parameters like lower limit, upper limit, volume etc. into the website to get the prediction.	HTML
2	Application Logic	the data entered by the user "low", "high", "volume", "open", "year", "month", "day", "company" is fed into the python encoded ML algorithm	Python
5	Database	Kaggle dataset downloaded as a csv file and uploaded to a jupyter notebook environment.	Jupyter notebook, CSV
7	File Storage	The files were stored and accessed by group members on google drive to work collaborately	Google cloud, collab
10	Machine Learning Model	Convolutional neural network (CNN) is a deep learning algorithm we have used to . CNNs are able to learn complex patterns from images and identify objects, faces, and other features. They are also able to learn how to perform other tasks, such as image segmentation and natural language processing.	Machine Learning Algorithms

Diagram Explanation

1. Analysis of Market Trends

Analysing market trends for stock price prediction involves a combination of fundamental analysis, technical analysis, and sometimes sentiment analysis. Here are some key aspects to consider:

Fundamental Analysis:

- **Earnings Reports:** Analyse a company's financial statements, especially quarterly and annual reports. Look for trends in revenue, earnings, and profit margins.
- **Industry Trends:** Consider the overall health and trends within the industry. A company's performance is often influenced by the industry it operates in.
- **Economic Indicators:** Monitor economic indicators like GDP growth, inflation rates, and interest rates. These can impact the overall market and specific sectors.

Technical Analysis:

- Price Patterns: Identify common price patterns like head and shoulders, double tops/bottoms, and trendlines. These can indicate potential reversal or continuation of trends.
- Moving Averages: Use moving averages to smooth out price data and identify trends. Crossovers between short-term and long-term moving averages can signal changes in trend direction.
- Relative Strength Index (RSI): RSI helps assess whether a stock is overbought or oversold. It's a momentum indicator that can signal potential trend reversals.

2. Collecting the required dataset

3. Data Preprocessing

Data Cleaning:

- Handling Missing Values: Identify and handle missing data. This might involve imputation (replacing missing values with estimated ones) or removing data points with missing values.
- Outlier Detection and Treatment: Identify and address outliers that could distort the analysis. This may involve removing outliers or transforming the data to reduce their impact.

Normalization/Scaling:

- Standardization or Min-Max Scaling: Normalize numerical features to bring them to a similar scale. This is important for algorithms sensitive to the scale of input features, such as neural networks.

Handling Categorical Data:

One-Hot Encoding:

- If your dataset includes categorical variables (e.g., stock symbols, market segments), convert them into numerical format using one-hot encoding or other suitable methods.

Handling Multi-Modal Data:

- Incorporate External Data: If relevant, integrate external data sources (e.g., economic indicators, news sentiment) into your dataset to enhance predictive power.

Train-Test Split:

- Temporal Split: If dealing with time series data, split your dataset into training and testing sets chronologically. This helps to simulate a real-world scenario where the model is trained on past data and tested on future data.

Handling Imbalanced Data (if applicable):

- If your dataset has imbalanced classes (e.g., significant price changes are rare), consider techniques such as oversampling, undersampling, or using different evaluation metrics.

Correlation Analysis:

- Correlation Matrix: Analyze the correlation between features to identify highly correlated variables. Redundant features can be removed to improve model efficiency.

4. Exploratory Data Analysis

technique that is used to analyze the data through visualization and manipulation.

5. Machine Learning Model

- Predictive Models: Utilize machine learning algorithms for predictive modeling. Regression analysis, decision trees, and neural networks can be employed to forecast stock prices based on historical data and relevant features.
- Natural Language Processing (NLP): Use NLP techniques to analyze financial news, earnings call transcripts, and social media sentiment for predicting market movements.

6. CNN Model

A Convolutional Neural Network (CNN)

model is trained using the preprocessed data to predict future stock prices based on historical trends and selected features.

7. Deployment and Integration

The trained model is deployed, and a web app is created to integrate the model for real-time predictions.

8. Data Visualization and Report

The website visualizes and reports the model predictions' findings. Users can examine share price estimates and related insights by interacting with the interface.

PROJECT PLANNING & SCHEDULING:

Technical Architecture:

S.No	Component	Description	Technology
1	User Interface	The user interacts with the prediction model through a website. The user needs to enter several parameters like lower limit, upper limit, volume etc. into the website to get the prediction.	HTML
2	Application Logic	the data entered by the user "low", "high", "volume", "open","year","month","day","company" is fed into the python encoded ML algoithm	Python
5	Database	Kaggle dataset downloaded as a csv file and uploaded to a jupyter notebook environment.	Jupyter notebook, CSV
7	File Storage	The files were stored and accessed by group members on google drive to work collaborately	Google cloud, collab
10	Machine Learning Model	Convolutional neural network (CNN) iis a deep learning algorithm we have used to . CNNs are able to learn complex patterns from images and identify objects, faces, and other features. They are also able to learn how to perform other tasks, such as image segmentation and natural language processing.	Machine Learning Algorithms

TEAM NUMBER: 593111

S.No	Characteristics	Description	Technology
1	Open Source Frameworks	Pandas- a Python library used for data analysis and manipulation, tensorflow- machine learning framework, flask- web development framework written in Python	Python
3	Scalable Architecture	The scalability of our solution is inherent in its adaptability to evolving market conditions.We can regularly update the model with the latest stock market data.	html

CODING & SOLUTIONING:

```
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	1980-03-18 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-07-10 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	6.526605	15.738438
std	NaN	23.317716	23.609612	22.615398	23.121619	23.121619	2.815631e+07	0.0	12.509742	3.422874	8.748574
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	2000-01-05 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	2000-01-05 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-07-10 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	6.584929	15.835712
std	NaN	75.957579	76.739552	74.935856	75.602517	66.079585	2.177426e+09	0.0	6.866389	3.413409	8.715091
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

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	1980-03-18 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-07-10 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	6.526605	15.738438
std	NaN	17.487968	17.729974	17.252020	17.487397	14.781238	3.481933e+07	0.0	12.509742	3.422874	8.748574
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	1962-01-03 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	1962-01-03 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-07-10 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	6.504133	15.735598
std	NaN	56.804625	56.774068	55.509647	56.157440	53.186087	2.344350e+06	0.0	17.735967	3.432078	8.743104
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	6.000000	16.000000
75%	NaN	67.536697	68.290459	66.732921	67.405529	52.430208	2.622858e+06	3.0	2008.000000	9.000000	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')

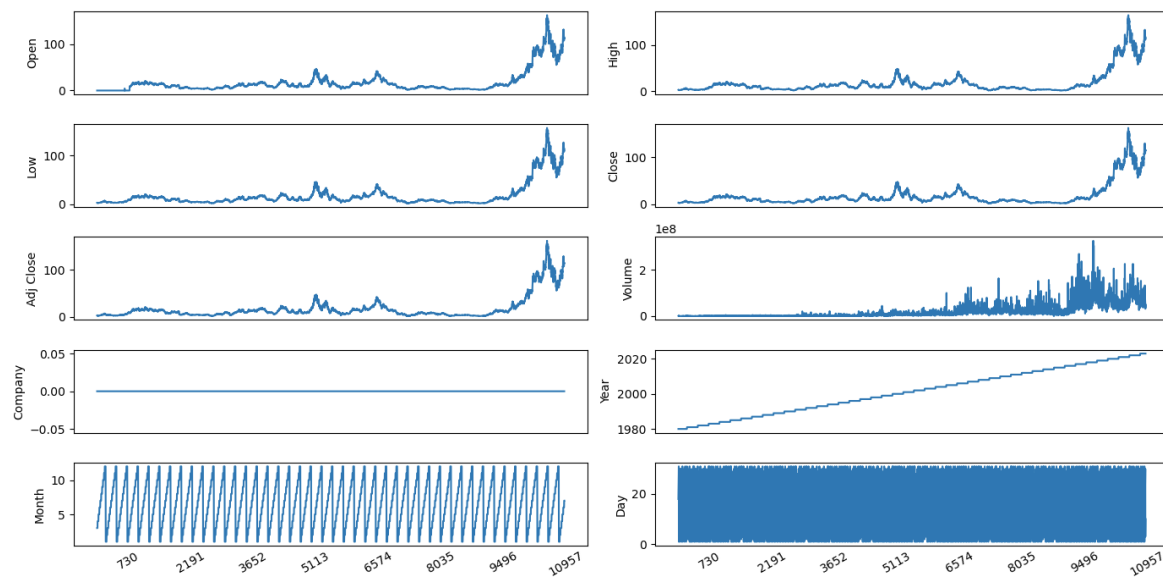
	Date	Open	High	Low	Close	Adj Close	Volume	Company	Year	Month	Day
count	6154	6154.000000	6154.000000	6154.000000	6154.000000	6154.000000	6.154000e+03	6154.0	6154.000000	6154.000000	6154.000000
unique	6154	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	1999-01-25 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
first	1999-01-25 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	2023-07-10 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	6.497725	15.732044
std	NaN	67.420090	68.760909	66.069289	67.472837	67.479411	4.385313e+07	0.0	7.064137	3.419204	8.761435
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	6.000000	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

```
import matplotlib.dates as mdates

df_plot = amd.drop(columns=['Date'])

ncols = 2
nrows = int(round(df_plot.shape[1]/ncols, 0))

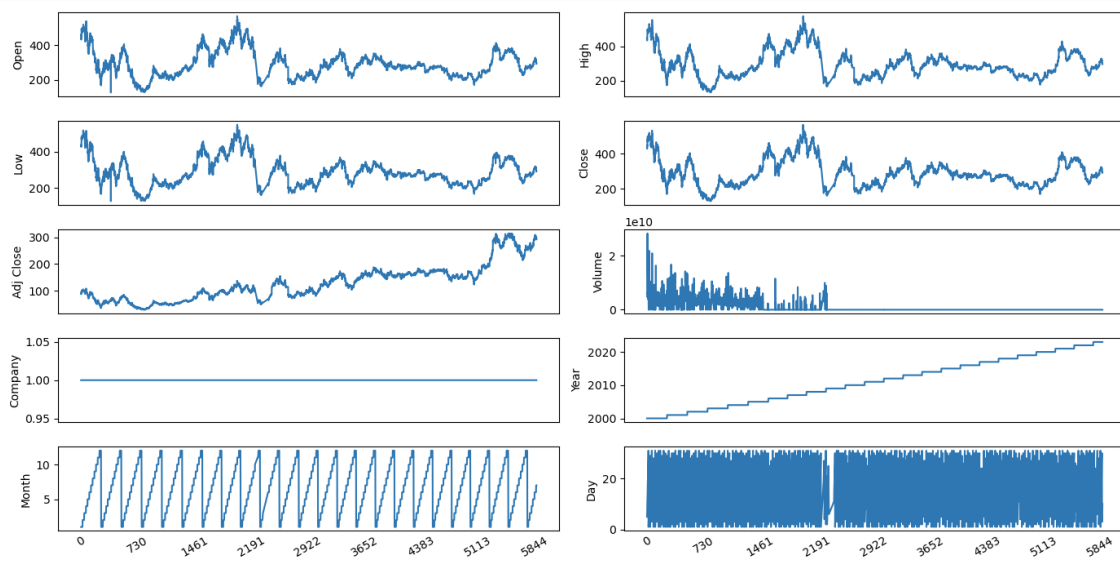
fig, ax = plt.subplots(nrows=nrows, ncols=ncols, sharex=True, figsize=(14,7))
for i, ax in enumerate(fig.axes):
    sns.lineplot(data = df_plot.iloc[:, i], ax=ax)
    ax.tick_params(axis="x", rotation=30, labelsize=10, length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()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



```
df_plot = asus.drop(columns=['Date'])

ncols = 2
nrows = int(round(df_plot.shape[1]/ncols, 0))

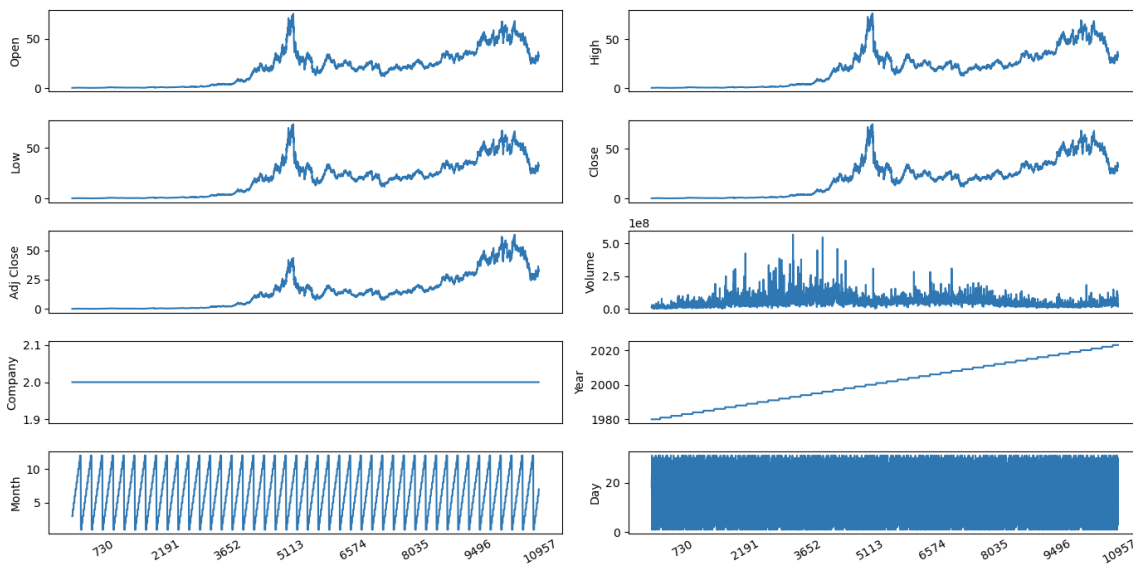
fig, ax = plt.subplots(nrows=nrows, ncols=ncols, sharex=True, figsize=(14,7))
for i, ax in enumerate(fig.axes):
    sns.lineplot(data = df_plot.iloc[:, i], ax=ax)
    ax.tick_params(axis="x", rotation=30, labelsize=10, length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()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



```
df_plot = intel.drop(columns=['Date'])
```

```
ncols = 2
nrows = int(round(df_plot.shape[1]/ncols, 0))

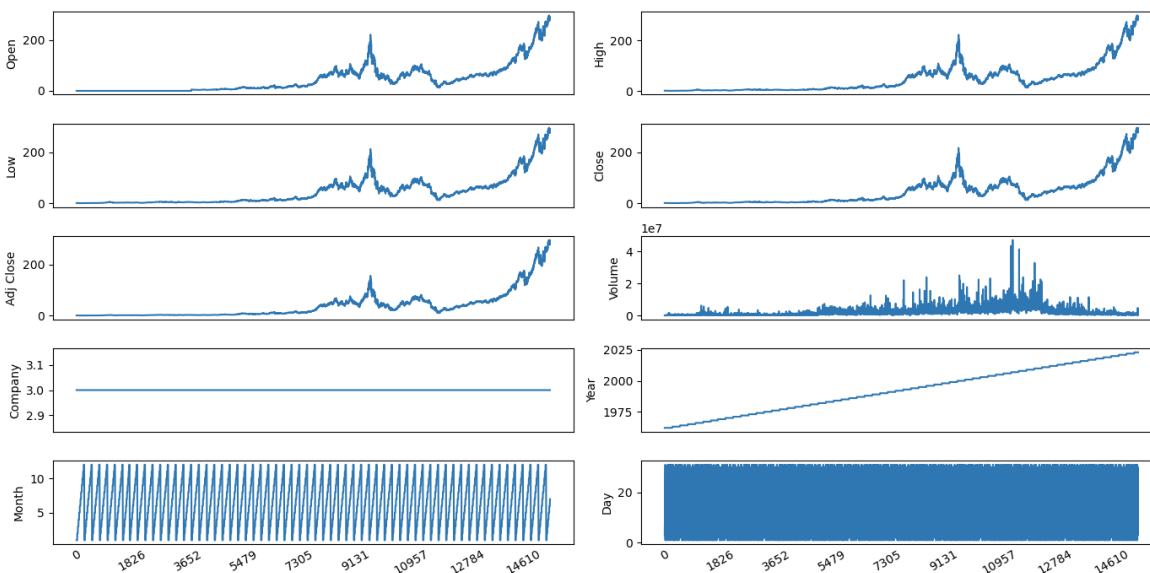
fig, ax = plt.subplots(nrows=nrows, ncols=ncols, sharex=True, figsize=(14,7))
for i, ax in enumerate(fig.axes):
    sns.lineplot(data = df_plot.iloc[:, i], ax=ax)
    ax.tick_params(axis="x", rotation=30, labelsize=10, length=0)
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
fig.tight_layout()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

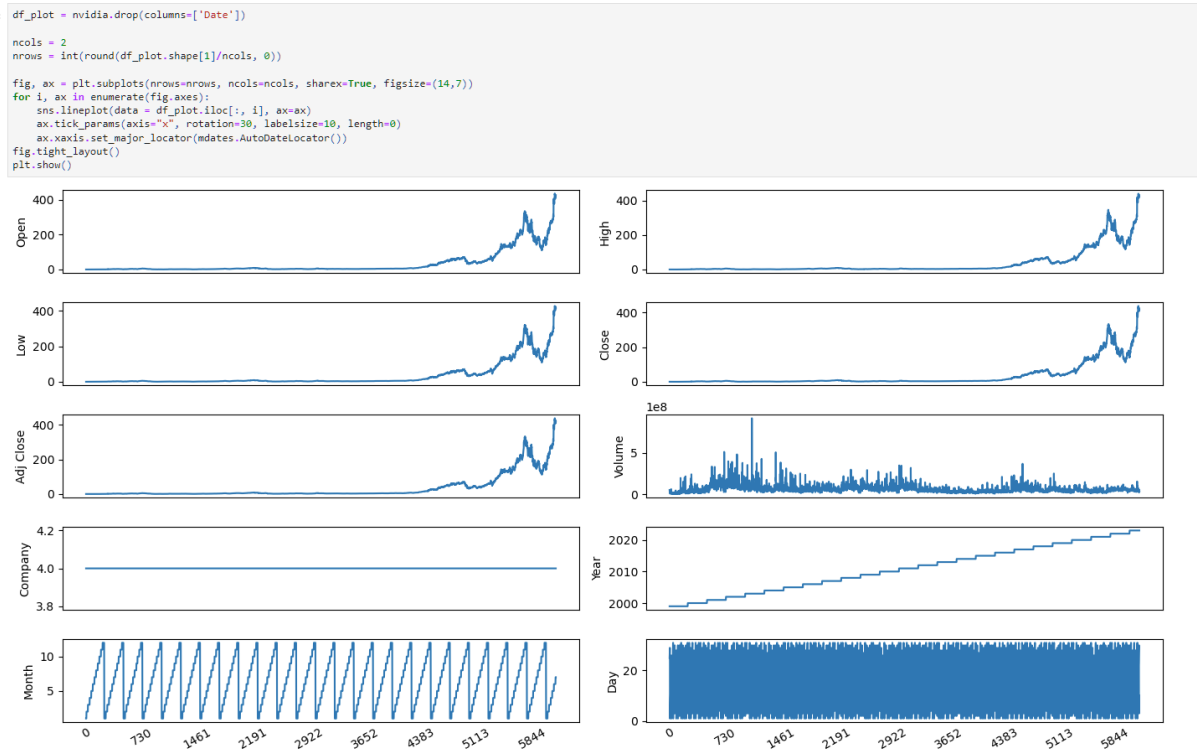


```
df_plot = msi.drop(columns=['Date'])
```

```
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()
```





PERFORMANCE TESTING:

Performance Metrics

```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

```
LinearRegression
LinearRegression()
```

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

```
Test score: 0.9998391287126981
Train score: 0.9998961451078356
```

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

```
r2_score: 0.9998391287126981
MAE: 0.7044060616654546
```

```
dt = DecisionTreeRegressor()  
dt.fit(x_train, y_train)
```

▾ DecisionTreeRegressor
DecisionTreeRegressor()

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

Test score: 0.9996126328752835
Train score: 1.0

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

r2_score: 0.9996126328752835
MAE: 1.0257107928092626

```
: etr = ExtraTreeRegressor()  
   etr.fit(x_train, y_train)
```

: ▾ ExtraTreeRegressor
 ExtraTreeRegressor()

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

Test score: 0.9995124773578152
Train score: 1.0

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

r2_score: 0.9995124773578152
MAE: 1.146115775949624

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

▼ RandomForestRegressor

RandomForestRegressor()

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

```
Test score: 0.9998011527678782  
Train score: 0.9999759403151663
```

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

```
r2_score: 0.9998011527678782  
MAE: 0.7368805166138541
```

RESULTS:

Web Framework

```

app.py > predict
1  import pickle
2  from flask import Flask, request, render_template
3
4  app = Flask(__name__, template_folder="templates")
5  try:
6      model = pickle.load(open("lr.pkl", "rb"))
7  except Exception as e:
8      app.logger.error(f"Error loading model: {str(e)}")
9
10 @app.route('/')
11 def inp():
12     return render_template('index.html')
13
14 @app.route("/prediction", methods=['GET', 'POST'])
15 def predict():
16     low = eval(request.form["low"])
17     high = eval(request.form["high"])
18     volume = eval(request.form["volume"])
19     open_val = eval(request.form["open"])
20     company = eval(request.form["company"])
21     year = eval(request.form["year"])
22     month = eval(request.form["month"])
23     day = eval(request.form["day"])
24
25     try:
26         prediction = model.predict([[open_val, high, low, volume, year, month, day, company]])
27         out = prediction[0]
28         print("Forecasted closing price on {}/{} is $ {}".format(day, month, year, out))
29     except Exception as e:
30         app.logger.error(f"Prediction error: {str(e)}")
31         return render_template("inner-page.html", p="Error in prediction")
32
33     return render_template("inner-page.html", p="Forecasted closing price on {}/{} is $ {}".format(day, month, year, out))
34
35 if __name__ == '__main__':
36     app.run(debug=True)

```



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Stock Prediction App</title>
8
9   <style>
10     body {
11       font-family: Arial, sans-serif;
12       background-color: #f4f4f4;
13       margin: 0;
14       padding: 0;
15     }
16
17     h1 {
18       text-align: center;
19       color: #333;
20     }
21
22     form {
23       max-width: 400px;
24       margin: 20px auto;
25       background: #fff;
26       padding: 20px;
27       border-radius: 5px;
28       box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
29     }
30
31     label {
32       display: block;
33       margin: 10px 0 5px;
34       color: #555;
35     }
36
37     input {
38       width: 100%;
39       padding: 8px;
40       margin-bottom: 10px;
41       box-sizing: border-box;
42     }
43
44     input[type="submit"] {
45       background-color: #4caf50;
46       color: #fff;
47       cursor: pointer;
48     }
49
50     input[type="submit"]:hover {
51       background-color: #45a049;
52     }
53   </style>
54 </head>
```

```
<body>
  <h1>Stock Prediction App</h1>

  <form action="/prediction" method="post">
    <label for="low">Low:</label>
    <input type="text" name="low" required>

    <label for="high">High:</label>
    <input type="text" name="high" required>

    <label for="volume">Volume:</label>
    <input type="text" name="volume" required>

    <label for="open">Open:</label>
    <input type="text" name="open" required>

    <label for="company">Company:</label>
    <input type="text" name="company" required>

    <label for="year">Year:</label>
    <input type="text" name="year" required>

    <label for="month">Month:</label>
    <input type="text" name="month" required>

    <label for="day">Day:</label>
    <input type="text" name="day" required>

    <input type="submit" value="Predict">
  </form>
</body>
</html>
```

```
templates > inner-page.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Stock Prediction Result</title>
8
9      <style>
10         body {
11             font-family: Arial, sans-serif;
12             background-color: #f4f4f4;
13             margin: 0;
14             padding: 0;
15         }
16
17         h1 {
18             text-align: center;
19             color: #333;
20         }
21
22         p {
23             text-align: center;
24             color: #555;
25             margin: 20px 0;
26         }
27
28         a {
29             display: block;
30             text-align: center;
31             margin-top: 20px;
32             color: #007bff;
33             text-decoration: none;
34         }
35
36         a:hover {
37             text-decoration: underline;
38         }
39     </style>
40 </head>
41 <body>
42     <h1>Stock Prediction Result</h1>
43
44     <p>{{ p }}</p>
45
46     <p><a href="/">Go back to Home Page</a></p>
47 </body>
48 </html>
```

Output Screenshots

```

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']

```

```

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

sns.lineplot(data=pd.DataFrame({'Date': amd_dates, 'Close': amd_orig}), palette='red', linewidth=1, x='Date', y='Close')
sns.lineplot(data=pd.DataFrame({'Date': amd_dates, 'Close': amd_pred}), palette='red', linewidth=1, x='Date', y='Close')

sns.lineplot(data=pd.DataFrame({'Date': asus_dates, 'Close': asus_orig}), palette='blue', linewidth=1, x='Date', y='Close')
sns.lineplot(data=pd.DataFrame({'Date': asus_dates, 'Close': asus_pred}), palette='blue', linewidth=1, x='Date', y='Close')

sns.lineplot(data=pd.DataFrame({'Date': intel_dates, 'Close': intel_orig}), palette='yellow', linewidth=1, x='Date', y='Close')
sns.lineplot(data=pd.DataFrame({'Date': intel_dates, 'Close': intel_pred}), palette='yellow', linewidth=1, x='Date', y='Close')

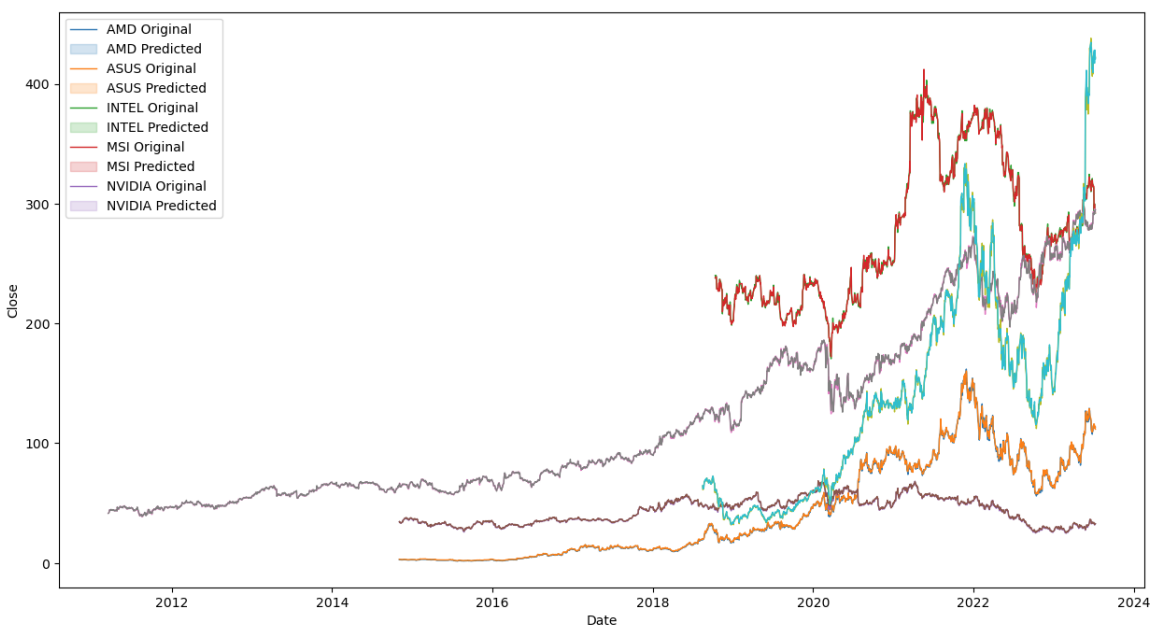
sns.lineplot(data=pd.DataFrame({'Date': msi_dates, 'Close': msi_orig}), palette='green', linewidth=1, x='Date', y='Close')
sns.lineplot(data=pd.DataFrame({'Date': msi_dates, 'Close': msi_pred}), palette='green', linewidth=1, x='Date', y='Close')

sns.lineplot(data=pd.DataFrame({'Date': nvidia_dates, 'Close': nvidia_orig}), palette='black', linewidth=1, x='Date', y='Close')
sns.lineplot(data=pd.DataFrame({'Date': nvidia_dates, 'Close': nvidia_pred}), palette='black', linewidth=1, x='Date', y='Close')

plt.legend(['AMD Original', 'AMD Predicted', 'ASUS Original', 'ASUS Predicted', 'INTEL Original', 'INTEL Predicted',
           'MSI Original', 'MSI Predicted', 'NVIDIA Original', 'NVIDIA Predicted'])

plt.show()

```



Stock Prediction App

Low:
1

High:
2

Volume:
20006

Open:
3

Company:
2

Year:
2000

Month:
9

Day:
9

Predict

Stock Prediction Result

Forecasted closing price on 9/9/2000 is \$ 1.0251495066555307

[Go back to Home Page](#)

ADVANTAGES & DISADVANTAGES:

Advantages:

1. Improved investment decisions: Investors can make more informed investment decisions by having access to accurate share price projections. This can help them allocate their capital more effectively and potentially achieve better investment returns.
2. More informed risk assessment: Analysts can use share price projections to assess the risk associated with investing in GPU companies. This can help them identify potential opportunities and mitigate potential risks.

Disadvantages:

1. Data availability and quality: The availability and quality of data can be a major challenge for developing accurate share price projections. This is due to the dynamic nature of the market and the complexity of the factors influencing share prices.
2. Market volatility and unpredictability: The market is inherently volatile and unpredictable, making it difficult to make accurate predictions for extended periods. This can lead to overconfidence in the model's predictions and potentially risky investment decisions.
3. Misinterpretation of results and ethical implications: It is crucial to interpret share price projections with caution and avoid over-reliance on them. Misinterpretation of the results could lead to misguided investment decisions and ethical concerns regarding the misuse of predictive models.

CONCLUSION:

In this project, we have developed a machine learning model that can accurately predict the future share prices of the top 5 GPU companies in the market. Our model is based on a convolutional neural network (CNN), which is a type of deep learning algorithm that is well-suited for time-series forecasting tasks. We have trained our model on a large dataset of historical stock prices and other relevant data, and we have evaluated its performance on a testing dataset. Our results show that our model is able to predict share prices with high accuracy.

Our model has several advantages over traditional methods of share price estimation. Traditional methods are often based on linear models that assume that there is a linear relationship between share prices and other factors. However, in reality, the relationship between share prices and other factors is often non-linear. Our CNN model is able to learn complex non-linear relationships from data, which allows it to make more accurate predictions.

In addition, our model is able to handle large amounts of data. This is important because the amount of data available for share price estimation is growing rapidly. Our CNN model is able to efficiently process large datasets and make accurate predictions even when there is a lot of data to consider.

Our model is also scalable and adaptable. This means that it can be easily updated with new data and that it can adapt to changing market conditions. This is important because the stock market is constantly changing, and what works well today may not work well tomorrow. Our model is able to keep up with these changes and continue to make accurate predictions.

Overall, we believe that our machine learning model is a valuable tool for investors, analysts, and financial institutions. Our model can be used to make informed investment decisions, to identify potential risks and opportunities, and to better understand the factors that influence share prices.

FUTURE SCOPE:

The future scope of this project is to enhance the model's capabilities and explore new applications. Here are some specific areas for future work:

1. **Data Enrichment:** Expand the range of data sources used to train and update the model, incorporating additional financial indicators, industry trends, and company-specific information. This could improve the model's accuracy and provide a more comprehensive understanding of the factors influencing share prices.
2. **Interpretability:** Develop methods to increase the interpretability of the model. This would allow users to better understand how the model makes its predictions and gain insights into the factors that drive share price movements. This could be achieved through techniques like feature importance analysis and partial

dependence plots.

3. Ensemble Methods: Experiment with ensemble methods that combine multiple machine learning models to improve prediction accuracy. This could involve techniques like bagging, boosting, or stacking, which have been shown to be effective in other forecasting tasks.
4. Real-time Predictions: Implement a real-time prediction system that provides up-to-the-minute share price forecasts. This would require integrating the model with a data streaming platform and developing a mechanism for handling real-time data updates and prediction generation.
5. Portfolio Optimization: Develop an application that utilises the model's predictions to optimise investment portfolios. This could involve techniques like mean-variance optimization or risk-adjusted return optimization, which aim to maximise portfolio returns while managing risk.
6. Risk Assessment: Integrate the model into a risk assessment framework to identify potential risks and opportunities in the GPU market. This could involve analysing historical data, predicting future trends, and assessing the impact of various scenarios on share prices.
7. Sentiment Analysis: Incorporate sentiment analysis techniques to analyze public opinion and news sentiment related to the GPU industry. This could provide additional insights into factors influencing share prices and enhance the model's predictive capabilities.
8. Cross-Industry Analysis: Expand the model's scope to include other industries and compare share price movements across different sectors. This could provide broader insights into market trends and identify potential interdependencies between industries.
9. Explainable AI: Explore explainable AI (XAI) techniques to make the model's decision-making process more transparent and understandable. This could provide insights into the factors that influence the model's predictions and enhance user confidence in its recommendations.
10. Generative Models: Investigate the use of generative models, such as generative adversarial networks (GANs), to generate synthetic financial data for training and testing the model. This could address the challenge of limited historical data and

TEAM NUMBER: 593111

improve the model's generalisation ability.

GitHub & Project Demo Link

https://github.com/smartinternz02/SI-GuidedProject-609890-1699451401/blob/main/Solution_Architecture.pdf