PROJECT DOCUMENTATION AND SUBMISSION

**Domain : ** Applied Data Science

PROJECT: STOCK PRICE PREDICTION

Phase 05: Documentation and Submission

Applied Data Science:

Applied Data Science is the practical application of data science principles and techniques to real-world problems and decision making scenarios. It involves collecting, analyzing and interpreting data to extract valuable insights and inform practical solutions often within business, scientific or other domains.

STOCK PRICE PREDICTION:

Stock price Prediction involves using historical market data and various analytical methods to make informed estimates about the future prices of individual stocks or the broader stock market. It utilizes statistical models, maching learning algorithms and financial indicators to project potential price movements. The objective is to assist investors and traders in making decisions about buying or selling stocks, aiming to maximize returns or minimize losses.

Problem Statement:

The problem is to build a predictive model that forecasts stock prices based on historical market data. The goal is to create a tool that assists investors in making well-informed decisions and optimizing their investment strategies. This project involves data collection, data preprocessing, feature engineering, model selection, training, and evaluation.

Design Thinking:

1. Data Collection:

Data collection is the process of gathering historical stock market data, which includes essential features like date, open price, close price, volume, and other relevant indicators. This data serves as the foundation for building a stock price prediction model.

2. Data Preprocessing:

Data preprocessing refers to the steps taken to clean and prepare the collected stock market data for analysis. This includes handling missing values, removing outliers, and converting categorical features into numerical representations, ensuring that the data is in a suitable format for modeling.

3. Feature Engineering:

Feature engineering involves creating additional features or variables that have the potential to enhance the predictive power of the stock price prediction model. This can include calculating moving averages, incorporating technical indicators (e.g., Relative Strength Index), and creating lagged variables that capture historical price movements.

4. Model Selection:

Model selection is the process of choosing appropriate algorithms or methods for time series forecasting in the context of stock price prediction. Common choices include AutoRegressive Integrated Moving Average (ARIMA) models, Long Short-Term Memory (LSTM) neural networks, or other machine learning techniques tailored to time series data.

5. Model Training:

Model training involves using the preprocessed historical stock market data to teach the selected forecasting model how to make predictions. During this phase, the model learns patterns and relationships in the data that will enable it to make future stock price predictions.

6. Evaluation:

Evaluation is the assessment of the stock price prediction model's performance. This typically involves comparing the model's predictions to actual stock prices over a specified evaluation period. Common evaluation metrics in time series forecasting include Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which quantify the accuracy of the model's predictions.

7.Phases of Development:

The development phases can include:

-Data collection

-Data preprocessing

-Feature extraction

-Model selection

-Model training

-Model evaluation

-Deployment

Phase 5: Problem Definition and Design Thinking

Dataset Link:

https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset

DESIGN THINKING ON STOCK PRICE PREDICTION:-

Design thinking can be a valuable approach when working on complex problems like stock price prediction. It emphasizes empathy for users (investors, traders, or financial analysts), collaboration, creativity, and iteration. Here are some design thinking ideas and principles applied to stock price prediction:

1. User-Centered Research:

Begin by conducting interviews and surveys with various stakeholders, such as retail investors, financial analysts, and fund managers, to understand their needs, pain points, and goals related to stock price prediction.

2.Persona Development:

Create user personas based on the research findings. Develop detailed profiles of typical users, including their goals, challenges, and preferences in using stock prediction tools.

3.Empathetic Ideation:

Host brainstorming sessions with cross-functional teams to generate innovative ideas for stock price prediction solutions. Encourage participants to think from the perspective of the identified user personas.

4.Prototyping:

Build interactive prototypes or mockups of the stock prediction tool. These prototypes can be used for user testing and validation before investing heavily in development.

5.Iterative Testing:

Conduct usability testing with actual users to gather feedback on the prototype. Iterate and refine the design based on user input, making the tool more intuitive and user-friendly.

6. Visual Storytelling:

Use data visualization techniques to convey complex stock market data in a visually compelling and easy-to-understand manner. Visualization can help users make sense of historical trends and future predictions.

7.Feedback Loops:

Implement feedback mechanisms within the tool to collect user opinions, suggestions, and corrections. Continuously update the tool based on user feedback to improve its accuracy and usability.

8.Ethical Considerations:

Integrate ethical considerations into the design process. Ensure transparency in how predictions are generated, and provide clear disclaimers about the inherent risks of investing.

9.Collaboration and Cross-Disciplinary Teams

Encourage collaboration between data scientists, user experience (UX) designers, financial experts, and software developers to create a well-rounded stock prediction solution.

10.Education and Support:

Include educational components within the tool to help users understand the underlying principles of stock market dynamics and prediction methodologies.

11.A/B Testing:

Implement A/B testing to assess the effectiveness of different prediction algorithms or user interface variations. Continuously refine the tool based on the performance of these tests.

12.Scalability and Accessibility:

Ensure that the design can scale to handle large volumes of data and accommodate users with different levels of expertise and accessibility needs.

13.Data Privacy and Security:

Place a strong emphasis on data privacy and security, especially when dealing with sensitive financial data. Comply with relevant regulations and industry standards.

14.Sustainability:

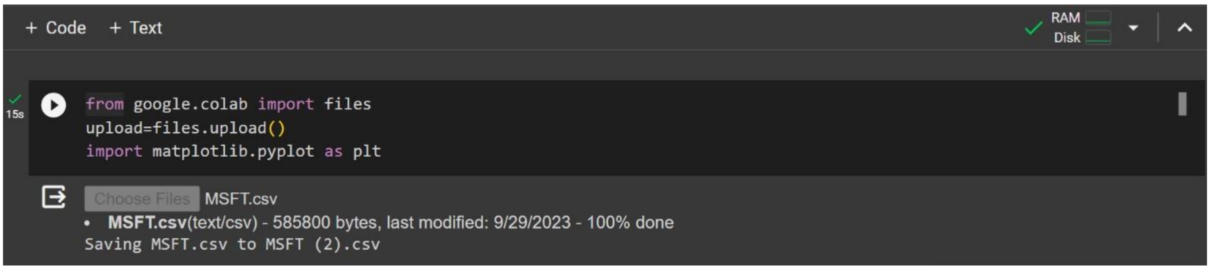
Consider the environmental impact of data processing and server infrastructure.

Aim to make the tool more energy-efficient and environmentally sustainable.

15.Real-World Validation:

Continuously track the performance of the stock prediction tool in real-world scenarios and compare its predictions with actual market movements. Use this feedback to improve the model and design.

IMPORTING FILE:



info () method:

```
msft.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       8525 non-null   object
 1   Open       8525 non-null   float64
 2   High       8525 non-null   float64
 3   Low        8525 non-null   float64
 4   Close      8525 non-null   float64
 5   Adj Close  8525 non-null   float64
 6   Volume     8525 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 466.3+ KB
```

head () method:

```
msft.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
4	1986-03-19	0.099826	0.100694	0.097222	0.098090	0.063107	47894400

describe () method:

```
[49] msft.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	8525.000000	8525.000000	8525.000000	8525.000000	8525.000000	8.525000e+03
mean	28.220247	28.514473	27.918967	28.224480	23.417934	6.045692e+07
std	28.626752	28.848988	28.370344	28.626571	28.195330	3.891225e+07
min	0.088542	0.092014	0.088542	0.090278	0.058081	2.304000e+06
25%	3.414063	3.460938	3.382813	3.414063	2.196463	3.667960e+07
50%	26.174999	26.500000	25.889999	26.160000	18.441576	5.370240e+07
75%	34.230000	34.669998	33.750000	34.230000	25.392508	7.412350e+07
max	159.449997	160.729996	158.330002	160.619995	160.619995	1.031789e+09

DATA COLLECTION:

```python
print(msft[['Open', 'Close','High','Low','Adj Close','Volume']])
```

```
             Open       Close        High         Low   Adj Close      Volume
0        0.088542    0.097222    0.101563    0.088542    0.062549  1031788800
1        0.097222    0.100694    0.102431    0.097222    0.064783   308160000
2        0.100694    0.102431    0.103299    0.100694    0.065899   133171200
3        0.102431    0.099826    0.103299    0.098958    0.064224    67766400
4        0.099826    0.098090    0.100694    0.097222    0.063107    47894400
...           ...         ...         ...         ...         ...         ...
8520   156.770004  157.699997  157.770004  156.449997  157.699997    18369400
8521   158.779999  160.619995  160.729996  158.330002  160.619995    22622100
8522   158.320007  158.619995  159.949997  158.059998  158.619995    21116200
8523   157.080002  159.029999  159.100006  156.509995  159.029999    20813700
8524   159.320007  157.580002  159.669998  157.330002  157.580002    18017762

[8525 rows x 6 columns]
```

```python
print(msft['Date'])
```

```
0       1986-03-13
1       1986-03-14
2       1986-03-17
3       1986-03-18
4       1986-03-19
           ...
8520    2019-12-31
8521    2020-01-02
8522    2020-01-03
8523    2020-01-06
8524    2020-01-07
Name: Date, Length: 8525, dtype: object
```

Finding Start Date and End Date:

```python
import pandas as pd
msft = pd.read_csv("MSFT.csv")
msft['Date'] = pd.to_datetime(msft['Date'])
start_date = msft['Date'].min()
end_date = msft['Date'].max()
print(f"Start Date: {start_date}")
print(f"End Date: {end_date}")
```

```
Start Date: 1986-03-13 00:00:00
End Date: 2020-01-07 00:00:00
```

DATE PREPROCESSING:

Finding the Missing Values

-Choose a suitable dataset that matches your problem statement.

-Perform data cleaning, handle missing values, and deal with outliers.

-Encode categorical variables if necessary.

-Normalize or scale numerical features.

-Split the dataset into training and testing sets.

0s

```
import pandas as pd
msft = pd.read_csv("MSFT.csv")
missing_values = msft.isna().sum()
print(missing_values)
```

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

FEATURE ENGINEERING:

Predictive model-Technical Indicators

```
#technical indicators
import pandas as pd
msft = pd.read_csv("MSFT.csv")
msft['Date'] = pd.to_datetime(msft['Date'])
msft.set_index('Date', inplace=True)
sma_period = 14
msft['SMA'] = msft['Close'].rolling(window=sma_period).mean()
print(msft[['Close', 'SMA']])
```

```
                 Close         SMA
Date
1986-03-13    0.097222         NaN
1986-03-14    0.100694         NaN
1986-03-17    0.102431         NaN
1986-03-18    0.099826         NaN
1986-03-19    0.098090         NaN
...                ...         ...
2019-12-31  157.699997  156.063572
2020-01-02  160.619995  156.700715
2020-01-03  158.619995  157.085000
2020-01-06  159.029999  157.406429
2020-01-07  157.580002  157.552858

[8525 rows x 2 columns]
```

MOVING AVERAGE:

Here MA_10 represents a10-day moving average and MA_50 represents a 50-day moving average

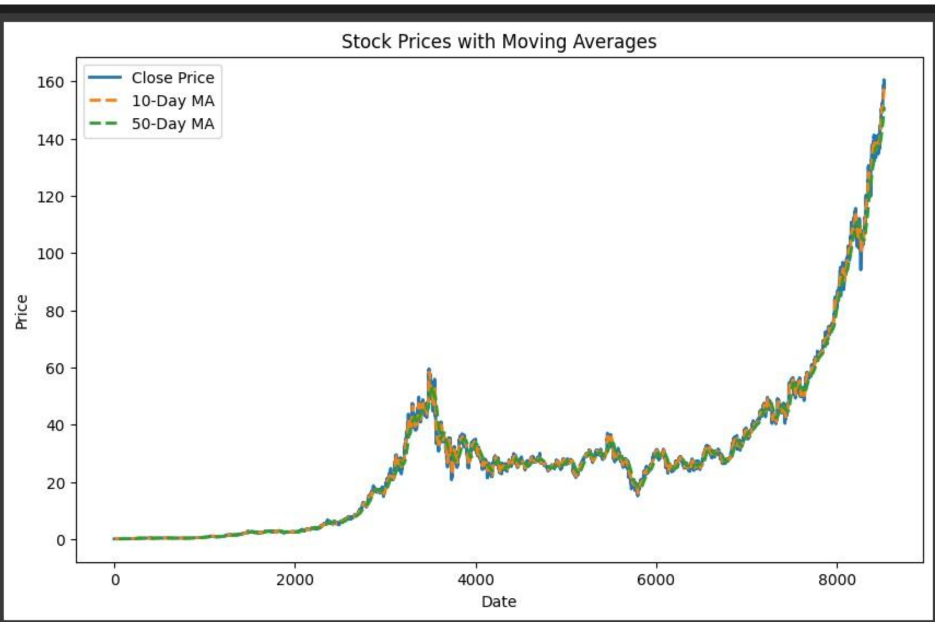
```
[7] import pandas as pd
    msft['MA_10'] = msft['Close'].rolling(window=10).mean()
    msft['MA_50'] = msft['Close'].rolling(window=50).mean()
```

```
[6] print(msft['MA_10'])
```

```
0              NaN
1              NaN
2              NaN
3              NaN
4              NaN
           ...
8520    156.989002
8521    157.582001
8522    158.007001
8523    158.339000
8524    158.356000
Name: MA_10, Length: 8525, dtype: float64
```

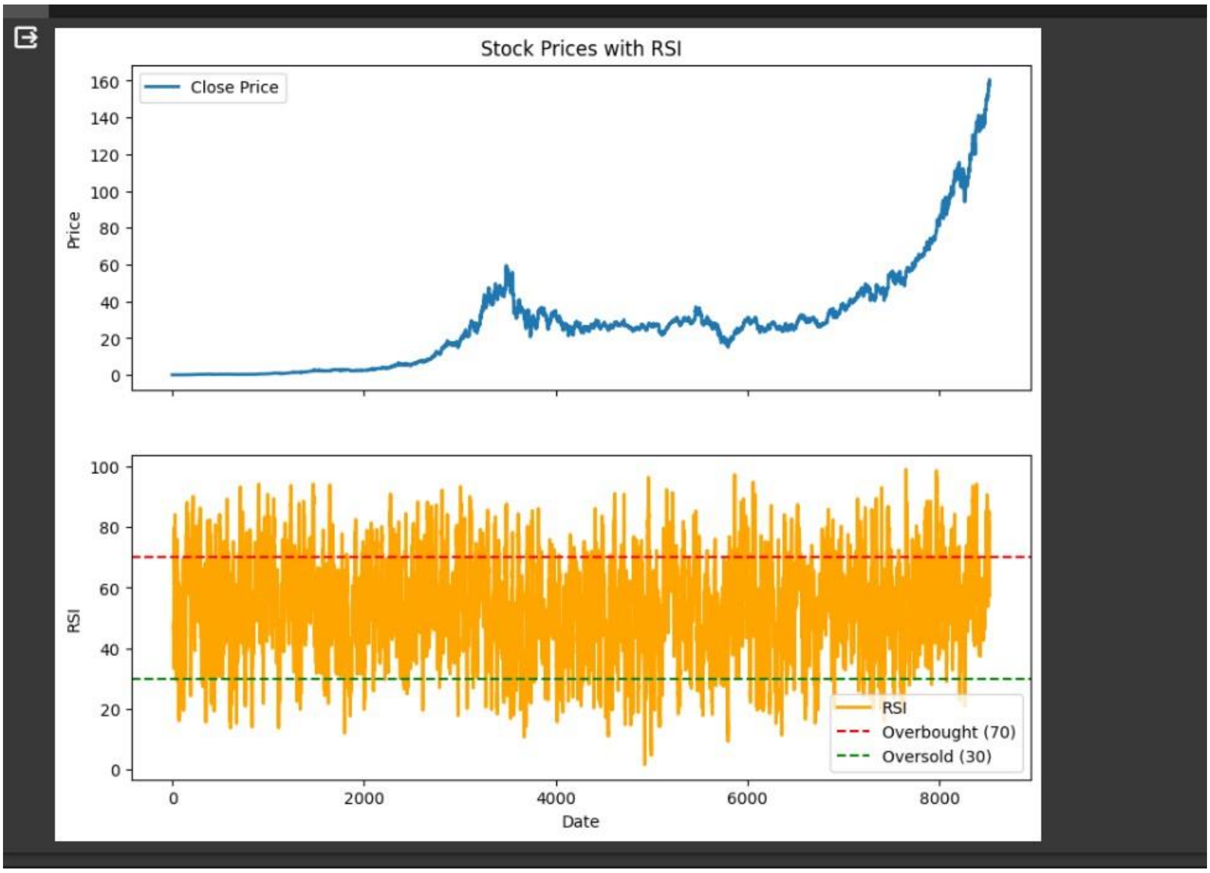
```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame 'df' with a 'Close' column
# ... (your data loading and manipulation here)

# Calculate moving averages
msft['MA_10'] = msft['Close'].rolling(window=10).mean()
msft['MA_50'] = msft['Close'].rolling(window=50).mean()

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(msft['Close'], label='Close Price', linewidth=2)
plt.plot(msft['MA_10'], label='10-Day MA', linestyle='--', linewidth=2)
plt.plot(msft['MA_50'], label='50-Day MA', linestyle='--', linewidth=2)

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Stock Prices with Moving Averages')

# Adding legend
plt.legend()

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

OUTPUT:



[24]
```python
#Technical Indicators

# Calculate daily price changes
msft['Price Change'] = msft['Close'].diff()

# Calculate average gains and losses over a specified period (e.g., 14 days)
gains = msft['Price Change'].apply(lambda x: x if x > 0 else 0).rolling(window=14).mean()
losses = -msft['Price Change'].apply(lambda x: x if x < 0 else 0).rolling(window=14).mean()

# Calculate Relative Strength (RS) and Relative Strength Index (RSI)
rs = gains / losses
msft['RSI'] = 100 - (100 / (1 + rs))
```

```python
# Plotting
fig, (ax1, ax2) = plt.subplots(2, sharex=True, figsize=(10, 8))

# Plotting the closing prices
ax1.plot(msft['Close'], label='Close Price', linewidth=2)
ax1.set_ylabel('Price')
ax1.set_title('Stock Prices with RSI')

# Plotting the RSI
ax2.plot(msft['RSI'], label='RSI', color='orange', linewidth=2)
ax2.axhline(y=70, color='r', linestyle='--', label='Overbought (70)')
ax2.axhline(y=30, color='g', linestyle='--', label='Oversold (30)')
ax2.set_xlabel('Date')
ax2.set_ylabel('RSI')

# Adding legend
ax1.legend()
ax2.legend()

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



Model Training:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Assuming 'X' is your feature matrix and 'y' is your target variable
# Replace this with your actual preprocessed feature matrix and target variable
# In this example, I assume that 'Close' is your target variable, and other columns are features.
X = msft[['Close','Open']] # Assuming 'Close' is the target variable
y = msft['Volume']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the model (replace this with your chosen model)
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error on Test Set: {mse}')

# Optionally, you can save the trained model for future use
# For example, if using a scikit-learn model:
# from joblib import dump
# dump(model, 'trained_model.joblib')
```

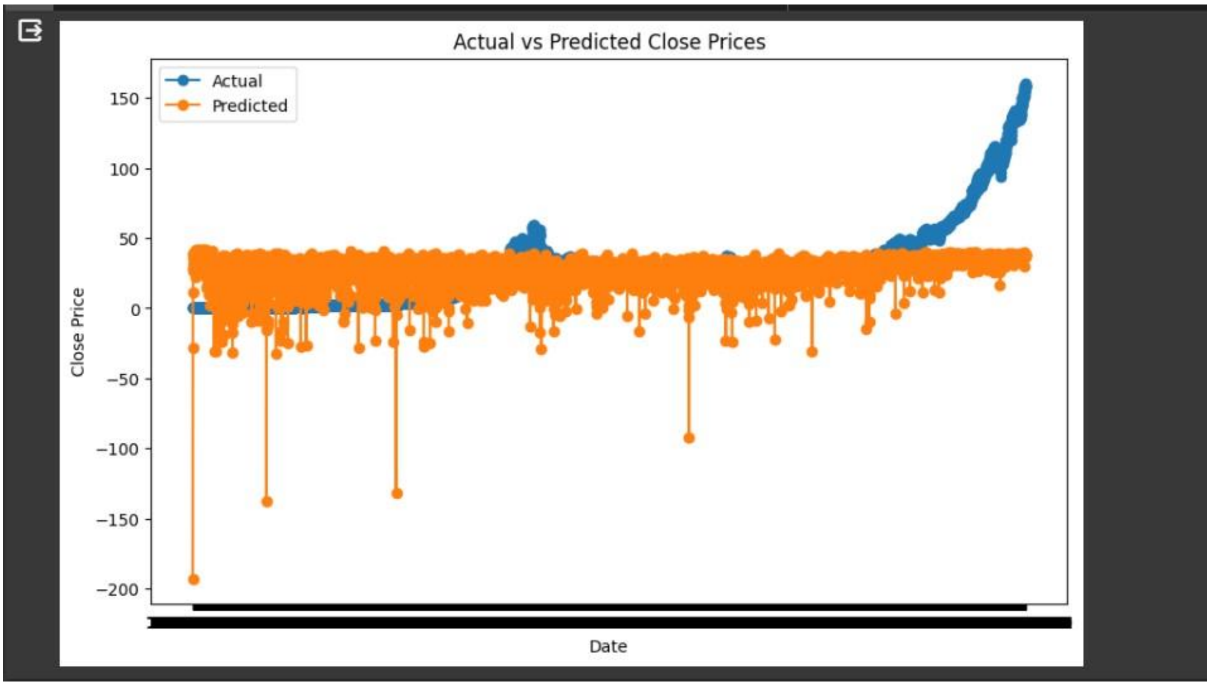
OUTPUT:

```
Mean Squared Error on Test Set: 1138679766436235.0
```

EVALUATION METRICES:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
msft=pd.read_csv(r"MSFT.csv")

X = msft['Volume'].values.reshape(-1, 1)
y = msft['Close'].values
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the model (replace this with your chosen model)
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the entire dataset
y_pred = model.predict(X)

# Align 'Date' and 'Close' arrays

dates = msft['Date']
num_dates = len(dates)

# Plotting actual vs predicted values
plt.figure(figsize=(10, 6))
plt.plot(dates[:num_dates], y[:num_dates], label='Actual', marker='o')
plt.plot(dates[:num_dates], y_pred[:num_dates], label='Predicted', marker='o')
plt.title('Actual vs Predicted Close Prices')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```

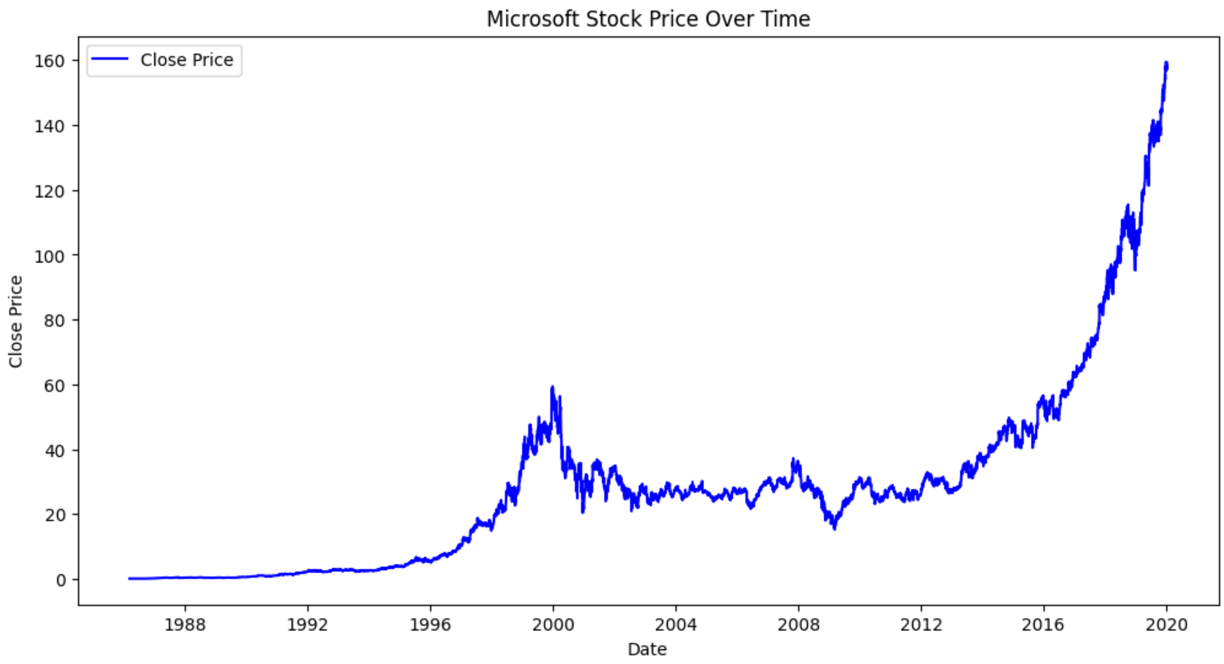
OUTPUT:



INPUT:

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

# Load your dataset (replace "your_data.csv" with the actual path)
msft = pd.read_csv("MSFT.csv")

# Ensure the 'Date' column is in datetime format
msft['Date'] = pd.to_datetime(msft['Date'])

# Plot the closing prices over time
plt.figure(figsize=(12, 6))
plt.plot(msft['Date'], msft['Open'], label='Close Price', color='blue')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Microsoft Stock Price Over Time')
plt.legend()
plt.show()
```

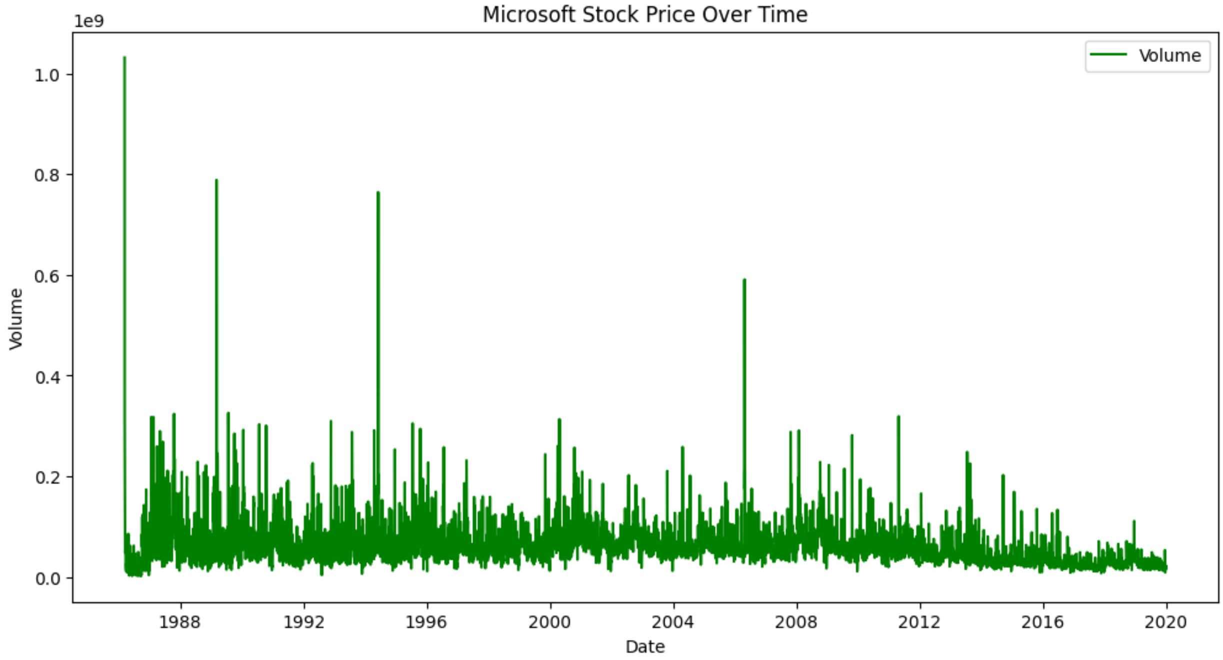
OUTPUT:



INPUT:

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

# Load your dataset (replace "your_data.csv" with the actual path)
msft = pd.read_csv("MSFT.csv")

# Ensure the 'Date' column is in datetime format
msft['Date'] = pd.to_datetime(msft['Date'])

# Plot the closing prices over time
plt.figure(figsize=(12, 6))
plt.plot(msft['Date'], msft['Volume'], label='Volume', color='Green')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.title('Microsoft Stock Price Over Time')
plt.legend()
plt.show()
```

OUPUT:



CONCLUSION:

-In conclusion, this document has outlined the key components of a typical machine learning project, starting from the problem statement and design thinking process to the phases of development. The design thinking approach emphasized empathy, defining the problem, ideation, prototyping, testing, and iteration, all aimed at creating a user-centered solution.

-The dataset selection and data preprocessing steps were discussed, demonstrating the importance of clean and relevant data for effective model training. Feature extraction techniques were highlighted as a means of capturing valuable information from the data. Python code snippets were provided as examples for data preprocessing and feature extraction.

-Here the Project Documentation and submission is obtained successfully!!...