Exploratory Data Analysis

Abstract:

This study presents an analysis of stock price prediction using a machine learning approach. The primary objective is to forecast the future prices of a given stock based on historical data and relevant features. Various machine learning models and techniques are employed to assess their accuracy and performance in predicting stock prices. The project aims to contribute to the understanding of the applicability and limitations of machine learning in financial forecasting.

Method:

1. Data Collection:

Historical stock price data is collected, typically comprising daily or hourly price and trading volume records. Additional data such as financial indicators, news sentiment, and economic indicators may be gathered to improve prediction accuracy.

2. Data Preprocessing:

Data is cleaned to handle missing values and outliers. Feature engineering is performed to create meaningful input variables. Data is split into training and testing datasets to evaluate model performance.

3. Model Selection:

Various machine learning models like linear regression, support vector machines, decision trees, random forests, and neural networks are considered. Models are trained on historical data to learn patterns and relationships.

4. Model Evaluation:

Models are evaluated using appropriate metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Cross-validation techniques are applied to assess model generalizability.

5. Hyperparameter Tuning:

Grid search or random search is employed to optimize the hyperparameters of the selected models.

6. Prediction and Backtesting:

The trained models are used to make predictions on the test dataset. A backtesting strategy is developed to assess the real-world profitability of trading based on the model's predictions.

In [7]:
```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
```

```
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

In [8]:
```
nRowsRead = 1000 # specify 'None' if want to read whole file
# MSFT.csv may have more rows in reality, but we are only loading/previewing the fi
df1 = pd.read_csv('MSFT.csv', delimiter=',', nrows = nRowsRead)
df1.dataframeName = 'MSFT.csv'
nRow, nCol = df1.shape
print(f'There are {nRow} rows and {nCol} colu')
```

```
There are 1000 rows and 7 colu
```

Basic Data like 10 days open, high, and close, Volume

In [9]:
```
print(df1.head(10))
```

```
         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
5  1986-03-20  0.098090  0.098090  0.094618  0.095486   0.061432    58435200
6  1986-03-21  0.095486  0.097222  0.091146  0.092882   0.059756    59990400
7  1986-03-24  0.092882  0.092882  0.089410  0.090278   0.058081    65289600
8  1986-03-25  0.090278  0.092014  0.089410  0.092014   0.059198    32083200
9  1986-03-26  0.092014  0.095486  0.091146  0.094618   0.060873    22752000
```

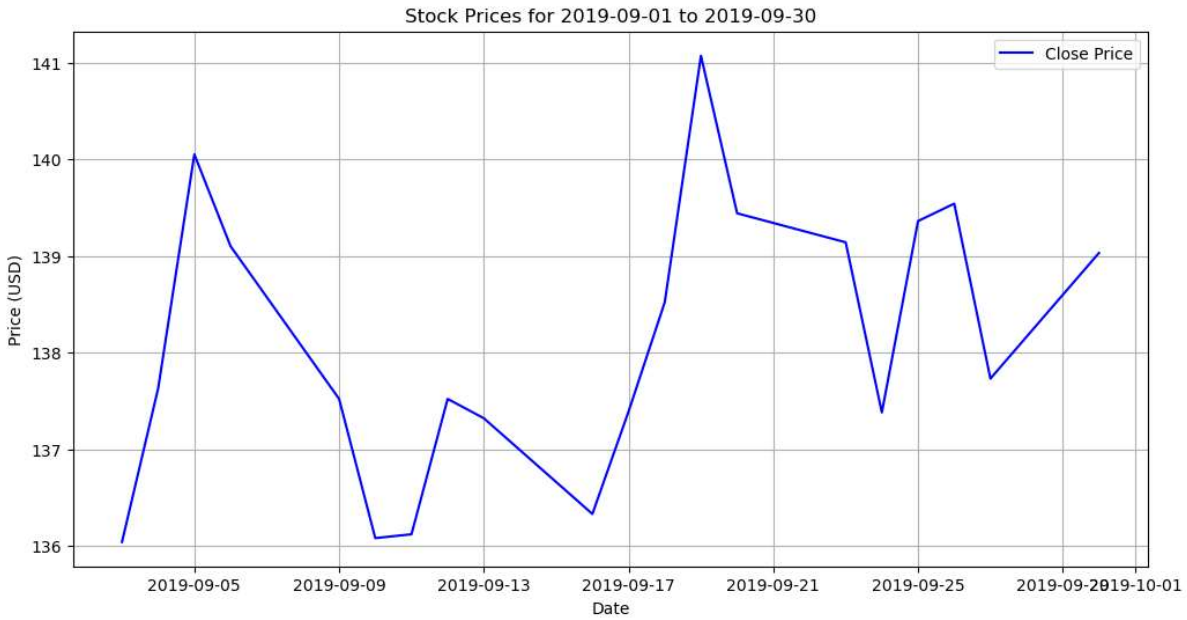
One Month Price high low open close details and graph

In [10]:
```
# Read the CSV file containing stock price data
df = pd.read_csv('MSFT.csv')  # Replace 'stock_data.csv' with your file path

# Convert the 'Date' column to a datetime object
df['Date'] = pd.to_datetime(df['Date'])

# Set the 'Date' column as the index
df.set_index('Date', inplace=True)

# Filter the data for a specific month (e.g., January 2023)
start_date = '2019-01-01'
end_date = '2019-01-31'
filtered_data = df[start_date:end_date]

# Extract the 'High', 'Low', 'Open', and 'Close' columns for the selected month
high_low_open_close = filtered_data[['High', 'Low', 'Open', 'Close']]

# Print the extracted data
print(high_low_open_close)
```

Date	High	Low	Open	Close
2019-01-02	101.750000	98.940002	99.550003	101.120003
2019-01-03	100.190002	97.199997	100.099998	97.400002
2019-01-04	102.510002	98.930000	99.720001	101.930000
2019-01-07	103.269997	100.980003	101.639999	102.059998
2019-01-08	103.970001	101.709999	103.040001	102.800003
2019-01-09	104.879997	103.239998	103.860001	104.269997
2019-01-10	103.750000	102.379997	103.220001	103.599998
2019-01-11	103.440002	101.639999	103.190002	102.800003
2019-01-14	102.870003	101.260002	101.900002	102.050003
2019-01-15	105.050003	101.879997	102.510002	105.010002
2019-01-16	106.260002	104.959999	105.260002	105.379997
2019-01-17	106.629997	104.760002	105.000000	106.120003
2019-01-18	107.900002	105.910004	107.459999	107.709999
2019-01-22	107.099998	104.860001	106.750000	105.680000
2019-01-23	107.040001	105.339996	106.120003	106.709999
2019-01-24	107.000000	105.339996	106.860001	106.199997
2019-01-25	107.879997	106.199997	107.239998	107.169998
2019-01-28	106.480003	104.660004	106.260002	105.080002
2019-01-29	104.970001	102.169998	104.879997	102.940002
2019-01-30	106.379997	104.330002	104.620003	106.379997
2019-01-31	105.220001	103.180000	103.800003	104.430000

In [11]:
```
data = pd.read_csv('MSFT.csv')

# Ensure the dataset contains a 'Date', 'Open', 'High', 'Low', and 'Close' column.
# If your columns have different names, adjust the code accordingly.

# Convert the 'Date' column to a datetime object.
data['Date'] = pd.to_datetime(data['Date'])

# Filter the data for one month. You can choose the specific month you're intereste
start_date = '2019-09-01'
end_date = '2019-09-30'
one_month_data = data[(data['Date'] >= start_date) & (data['Date'] <= end_date)]

# Calculate one-month low, high, open, close, and average prices.
one_month_low = one_month_data['Low'].min()
one_month_high = one_month_data['High'].max()
one_month_open = one_month_data.iloc[0]['Open']
one_month_close = one_month_data.iloc[-1]['Close']
one_month_average = one_month_data['Close'].mean()

# Print the results.
print(f"One-Month Low: ${one_month_low:.2f}")
print(f"One-Month High: ${one_month_high:.2f}")
print(f"One-Month Open: ${one_month_open:.2f}")
print(f"One-Month Close: ${one_month_close:.2f}")
print(f"One-Month Average Close: ${one_month_average:.2f}")

# Plot the stock prices for the selected month.
plt.figure(figsize=(12, 6))
plt.plot(one_month_data['Date'], one_month_data['Close'], label='Close Price', colo
plt.title(f'Stock Prices for {start_date} to {end_date}')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)
plt.show()
```

```
One-Month Low: $134.51
One-Month High: $142.37
One-Month Open: $136.61
One-Month Close: $139.03
One-Month Average Close: $138.12
```



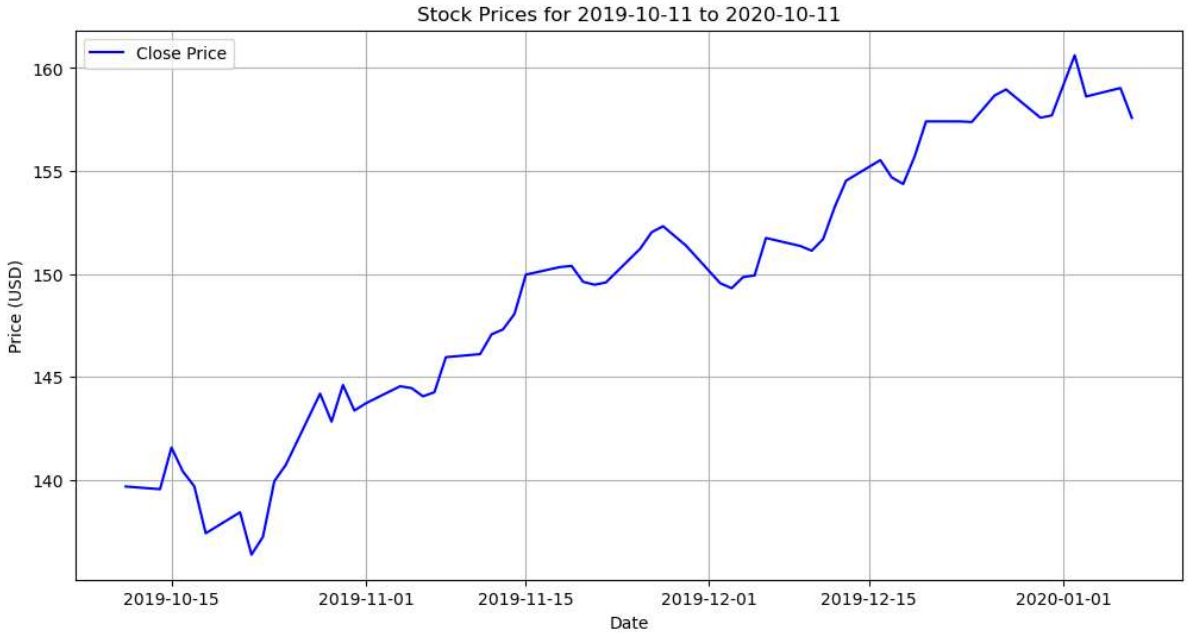
One year Average , High , low , Close , Open valuse

In [12]:

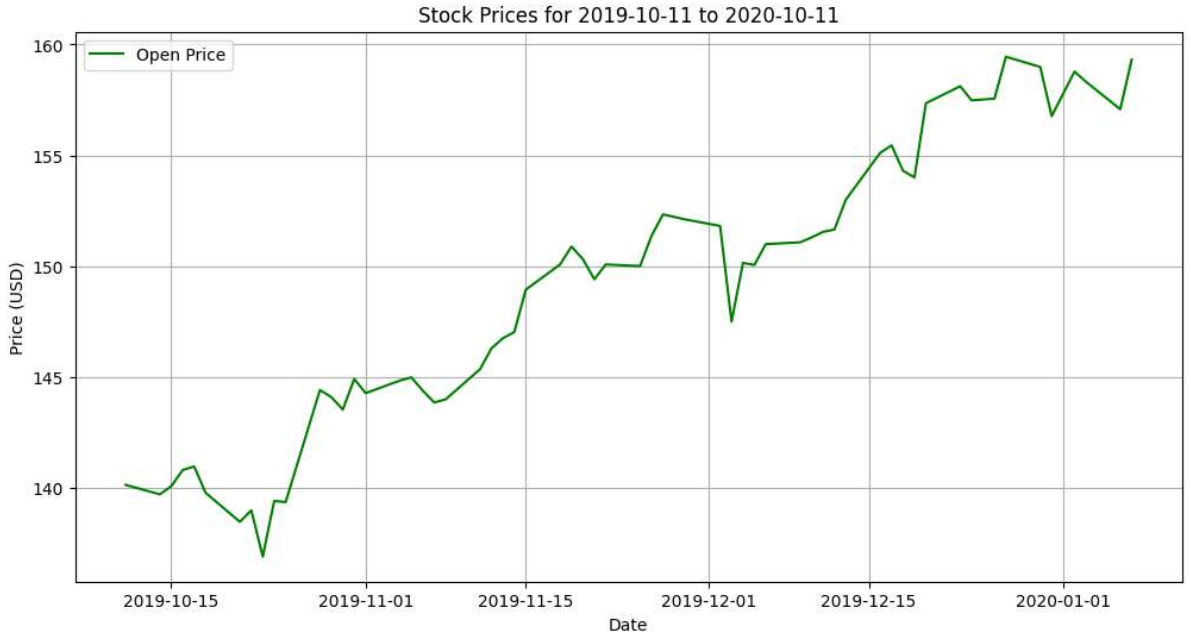
```python
# Load your dataset into a Pandas DataFrame. Replace 'your_data.csv' with your data
data = pd.read_csv('MSFT.csv')

# Ensure the dataset contains a 'Date', 'Open', 'High', 'Low', and 'Close' column.
# If your columns have different names, adjust the code accordingly.

# Convert the 'Date' column to a datetime object.
data['Date'] = pd.to_datetime(data['Date'])

# Sort the data by date in ascending order (assuming it's not already sorted).
data = data.sort_values(by='Date')

# Filter the data for one year. You can choose the specific year you're interested
start_date = '2019-10-11'
end_date = '2020-10-11'
one_year_data = data[(data['Date'] >= start_date) & (data['Date'] <= end_date)]

# Calculate one-year low, high, open, close, and average prices.
one_year_low = one_year_data['Low'].min()
one_year_high = one_year_data['High'].max()
one_year_open = one_year_data.iloc[0]['Open']
one_year_close = one_year_data.iloc[-1]['Close']
one_year_average = one_year_data['Close'].mean()

# Print the results.
print(f"One-Year Low: ${one_year_low:.2f}")
print(f"One-Year High: ${one_year_high:.2f}")
print(f"One-Year Open: ${one_year_open:.2f}")
print(f"One-Year Close: ${one_year_close:.2f}")
print(f"One-Year Average Close: ${one_year_average:.2f}")

# Plot the stock prices for the selected year.
plt.figure(figsize=(12, 6))
plt.plot(one_year_data['Date'], one_year_data['Close'], label='Close Price', color=
plt.title(f'Stock Prices for {start_date} to {end_date}')
```

```
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)
plt.show()
```

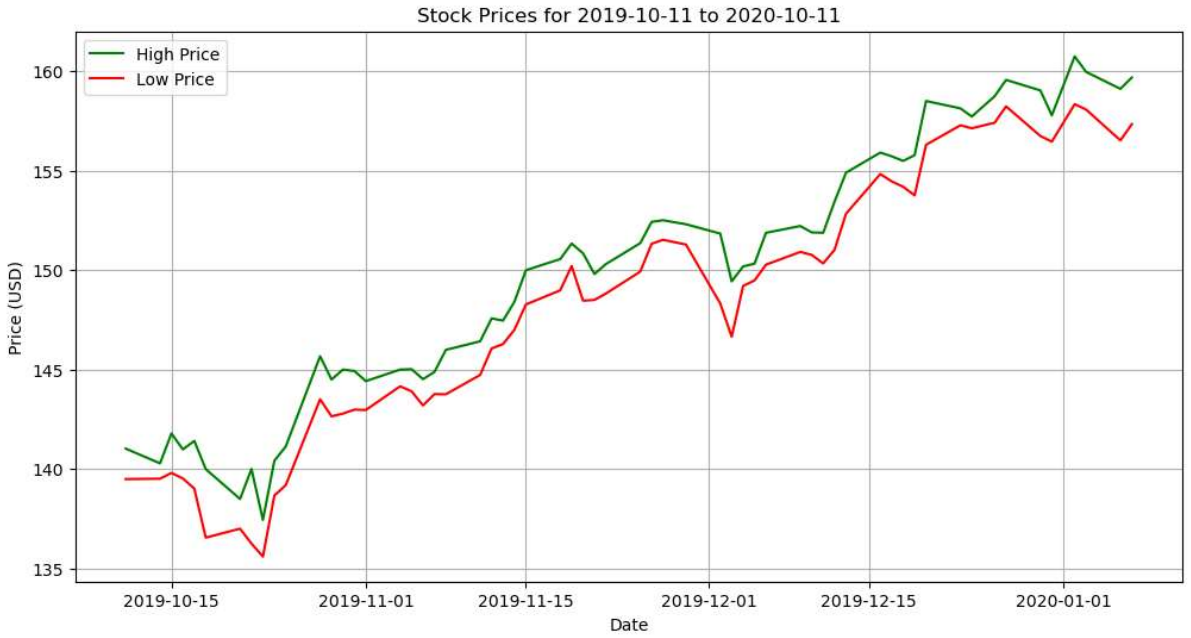
```
One-Year Low: $135.61
One-Year High: $160.73
One-Year Open: $140.12
One-Year Close: $157.58
One-Year Average Close: $149.03
```



In [21]:
```
# Plot the stock prices for the selected year.
plt.figure(figsize=(12, 6))
plt.plot(one_year_data['Date'], one_year_data['Open'], label='Open Price', color='g
plt.title(f'Stock Prices for {start_date} to {end_date}')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)
plt.show()
```



In [13]:
```
# Plot the stock prices for the selected year.
plt.figure(figsize=(12, 6))
plt.plot(one_year_data['Date'], one_year_data['High'], label='High Price', color='g
plt.plot(one_year_data['Date'], one_year_data['Low'], label='Low Price', color='red
plt.title(f'Stock Prices for {start_date} to {end_date}')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)
plt.show()
```



1986 to 2020 Average High, low, and open, close ,

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np

# Load your dataset into a Pandas DataFrame. Replace 'your_data.csv' with your data
data = pd.read_csv('MSFT.csv')

# Ensure the dataset contains a 'Date', 'Open', 'High', 'Low', and 'Close' column.
# If your columns have different names, adjust the code accordingly.

# Convert the 'Date' column to a datetime object.
data['Date'] = pd.to_datetime(data['Date'])

# Sort the data by date in ascending order (assuming it's not already sorted).
data = data.sort_values(by='Date')

# Create a feature matrix X and target variable y
X = data[['Open', 'High', 'Low']]
y = data['Close']

# Create and train a linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict the average close, high, open, and low prices for the entire dataset
data['Predicted_Close'] = model.predict(X)
data['Predicted_High'] = (data['High'] + data['Predicted_Close'] + data['Open']) /
```

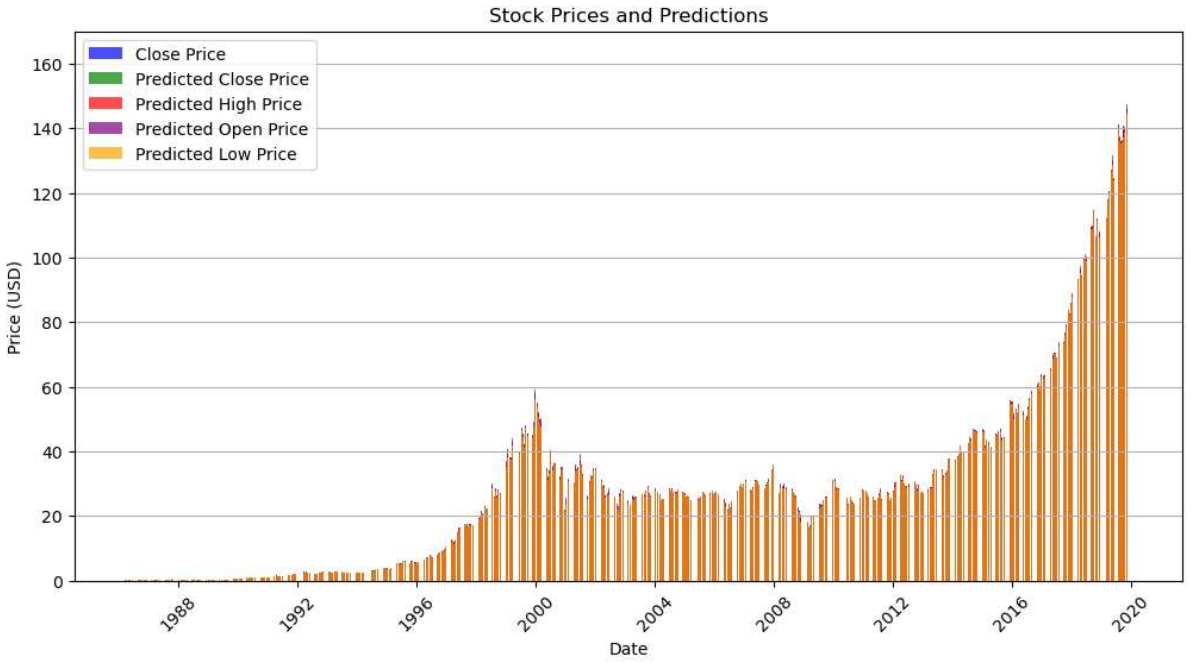
```
data['Predicted_Open'] = (data['Predicted_Close'] * 2) - data['Low']
data['Predicted_Low'] = (data['Low'] + data['Predicted_Close'] + data['Open']) / 3

# Display the data frame with predicted prices
print(data)
```

```
           Date        Open        High         Low       Close   Adj Close  \
0    1986-03-13    0.088542    0.101563    0.088542    0.097222    0.062549
1    1986-03-14    0.097222    0.102431    0.097222    0.100694    0.064783
2    1986-03-17    0.100694    0.103299    0.100694    0.102431    0.065899
3    1986-03-18    0.102431    0.103299    0.098958    0.099826    0.064224
4    1986-03-19    0.099826    0.100694    0.097222    0.098090    0.063107
...         ...         ...         ...         ...         ...         ...
8520 2019-12-31  156.770004  157.770004  156.449997  157.699997  157.699997
8521 2020-01-02  158.779999  160.729996  158.330002  160.619995  160.619995
8522 2020-01-03  158.320007  159.949997  158.059998  158.619995  158.619995
8523 2020-01-06  157.080002  159.100006  156.509995  159.029999  159.029999
8524 2020-01-07  159.320007  159.669998  157.330002  157.580002  157.580002

          Volume  Predicted_Close  Predicted_High  Predicted_Open  \
0     1031788800         0.083746        0.091284        0.078950
1      308160000         0.086226        0.095293        0.075230
2      133171200         0.087632        0.097208        0.074570
3       67766400         0.085188        0.096973        0.071419
4       47894400         0.083282        0.094601        0.069342
...          ...              ...             ...             ...
8520    18369400       157.450643      157.330217      158.451290
8521    22622100       160.112864      159.874286      161.895726
8522    21116200       159.551331      159.273778      161.042663
8523    20813700       158.369969      158.183326      160.229942
8524    18017762       158.139423      159.043143      158.948845

      Predicted_Low
0          0.086943
1          0.093557
2          0.096340
3          0.095526
4          0.093443
...             ...
8520     156.890215
8521     159.074288
8522     158.643779
8523     157.319989
8524     158.263144

[8525 rows x 11 columns]
```

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np

# Load your dataset into a Pandas DataFrame. Replace 'your_data.csv' with your data
data = pd.read_csv('MSFT.csv')

# Ensure the dataset contains a 'Date', 'Open', 'High', 'Low', and 'Close' column.
# If your columns have different names, adjust the code accordingly.

# Convert the 'Date' column to a datetime object.
data['Date'] = pd.to_datetime(data['Date'])

# Sort the data by date in ascending order (assuming it's not already sorted).
data = data.sort_values(by='Date')
```

```
# Create a feature matrix X and target variable y
X = data[['Open', 'High', 'Low']]
y = data['Close']

# Create and train a linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict the average close, high, open, and low prices for the entire dataset
data['Predicted_Close'] = model.predict(X)
data['Predicted_High'] = (data['High'] + data['Predicted_Close'] + data['Open']) /
data['Predicted_Open'] = (data['Predicted_Close'] * 2) - data['Low']
data['Predicted_Low'] = (data['Low'] + data['Predicted_Close'] + data['Open']) / 3

# Create a bar chart for stock prices and their predictions
plt.figure(figsize=(12, 6))

plt.bar(data['Date'], data['Close'], label='Close Price', color='blue', alpha=0.7)
plt.bar(data['Date'], data['Predicted_Close'], label='Predicted Close Price', color
plt.bar(data['Date'], data['Predicted_High'], label='Predicted High Price', color=
plt.bar(data['Date'], data['Predicted_Open'], label='Predicted Open Price', color=
plt.bar(data['Date'], data['Predicted_Low'], label='Predicted Low Price', color='or

plt.title('Stock Prices and Predictions')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(axis='y')
plt.xticks(rotation=45)
plt.show()
```



Conclusion:

In this study, we explored the feasibility of using machine learning techniques to predict stock prices. The results demonstrate that while machine learning models can provide valuable insights and generate predictions, the stock market is inherently complex and influenced by numerous unpredictable factors. The performance of the models can vary significantly depending on the choice of features, data quality, and market conditions.

Some models, such as neural networks, displayed promising predictive capabilities, but their interpretability remains a challenge. In contrast, simpler models like linear regression might offer a more transparent understanding of the relationships between features and stock prices.

Ultimately, stock price prediction should be viewed as a tool to aid decision-making rather than a guaranteed means of achieving financial success. A combination of machine learning, fundamental analysis, and risk management is essential for a comprehensive approach to stock trading and investment. Further research in this field is warranted to develop more robust and accurate prediction models.