

Twitter Stock Market Analysis

Introduction

The introduction should provide an overview of the project. Twitter is a social media platform that has been publicly traded. Analyzing its stock market data can provide insights into its financial performance and market trends, including daily stock prices (Open, High, Low, Close, Adj Close) and trading volume.

Methodology

The methodology involves analyzing the given stock market data using Python. This includes:


- Importing necessary libraries such as pandas for data manipulation and matplotlib/seaborn for data visualization.
- Loading the data into a pandas DataFrame.
- Performing exploratory data analysis (EDA) to understand the distribution of the data.
- Calculating daily returns and other relevant metrics.
- Visualizing the data to identify trends and patterns.



✓ Requirement Analysis

To analyze the twitter stock market data, we need to:

- Load the data into a suitable data structure (eg. pandas DataFrame).
- Clean the data if necessary (eg, handling missing values).
- perform key metrics such as daily returns, moving averages, and volatility.

```
import pandas as pd
path = ('/content/TWTR.xlsx')
ev_data = pd.read_excel(path)
ev_data.head()
```




	Date	Open	High	Low	Close	Adj Close	Volume	
0	2018-01-01	45.099998	50.090000	44.000000	44.900002	44.900002	117701670.0	
1	2018-01-02	45.930000	46.939999	40.685001	41.650002	41.650002	27925307.0	
2	2018-01-03	40.500000	43.000000	39.400002	42.900002	42.900002	16113941.0	
3	2018-01-04	43.660000	43.779999	41.830002	41.900002	41.900002	6316755.0	
4	2018-01-05	41.029999	42.869999	40.759998	42.599998	42.599998	8688325.0	



Next steps: [Generate code with ev_data](#) [View recommended plots](#) [New interactive sheet](#)

✓ Task 1: Data Analysis

1. Read the data and display the first 100 rows from the data:

```
ev_data.head(100)
```



	Date	Open	High	Low	Close	Adj Close	Volume	
0	2018-01-01	45.099998	50.090000	44.000000	44.900002	44.900002	117701670.0	
1	2018-01-02	45.930000	46.939999	40.685001	41.650002	41.650002	27925307.0	
2	2018-01-03	40.500000	43.000000	39.400002	42.900002	42.900002	16113941.0	
3	2018-01-04	43.660000	43.779999	41.830002	41.900002	41.900002	6316755.0	
4	2018-01-05	41.029999	42.869999	40.759998	42.599998	42.599998	8688325.0	
...	
95	2018-04-06	45.090000	46.400002	43.310001	46.320000	46.320000	15507597.0	
96	2018-04-07	46.650002	47.340000	45.700001	47.299999	47.299999	9610491.0	
97	2018-04-08	47.549999	47.750000	46.430000	46.669998	46.669998	5794497.0	
98	2018-04-09	46.709999	47.590000	46.180000	46.980000	46.980000	6916147.0	
99	2018-04-10	47.400002	47.439999	45.509998	45.730000	45.730000	7911260.0	

100 rows × 7 columns

Next steps: [Generate code with ev_data](#) [View recommended plots](#) [New interactive sheet](#)

2. Give the column insights:

```
ev_data.info()
```

```
>>> <class 'pandas.core.frame.DataFrame'>
RangeIndex: 2264 entries, 0 to 2263
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   Date        2264 non-null   datetime64[ns]
 1   Open        2259 non-null   float64
 2   High        2259 non-null   float64
 3   Low         2259 non-null   float64
 4   Close       2259 non-null   float64
 5   Adj Close   2259 non-null   float64
 6   Volume      2259 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 123.9 KB
```

3. Check whether this dataset contains any null values or not; if there are any, then remc

```
ev_data.isnull().sum()
```

```
>>>
      0
Date    0
Open    5
High    5
Low     5
Close   5
Adj Close 5
Volume  5

dtype: int64
```

4. Find the statistical description of the data:

```
ev_data.describe()
```



	Date	Open	High	Low	Close	Adj Close	Vol
count	2264	2259.000000	2259.000000	2259.000000	2259.000000	2259.000000	2.259000e
mean	2021-02-05 12:00:00	36.020286	36.699881	35.339465	36.003625	36.003625	2.175186e
min	2018-01-01 00:00:00	13.950000	14.220000	13.725000	14.010000	14.010000	0.000000e
25%	2019-07-20 18:00:00	25.550000	26.215001	24.912501	25.410000	25.410000	1.233530e
	2021-						

5. Find the missing values in the data:

```
ev_data.isnull().sum()
```



	0
Date	0
Open	5
High	5
Low	5
Close	5
Adj Close	5
Volume	5

dtype: int64

✓ Task : 2 Statistical Tests and Analysis

1. Give me the Z-test or T-test over High, Low, and Close columns and see if the null hypc

```
from scipy import stats
high_mean = ev_data['High'].mean()
low_mean = ev_data['Low'].mean()
close_mean = ev_data['Close'].mean()
print(high_mean)
print(low_mean)
print(close_mean)
```

```

⇒ 36.69988069278442
   35.33946480035414
   36.00362549048251

```

2. Check if the data is dependent or independent by using the chi-square method:

```

from scipy.stats import chi2_contingency
contingency_table = pd.crosstab(ev_data['High'], ev_data['Low'])
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square statistic:", chi2)
print("P-value:", p)
print("Degrees of freedom:", dof)
print("Expected frequencies:\n", expected)

```

```

⇒ Chi-square statistic: 3349789.5249999994
   P-value: 2.261960543587645e-80
   Degrees of freedom: 3300864
   Expected frequencies:
   [[0.00044267 0.00044267 0.00044267 ... 0.00044267 0.00044267 0.00044267]
    [0.00044267 0.00044267 0.00044267 ... 0.00044267 0.00044267 0.00044267]
    [0.00044267 0.00044267 0.00044267 ... 0.00044267 0.00044267 0.00044267]
    ...
    [0.00044267 0.00044267 0.00044267 ... 0.00044267 0.00044267 0.00044267]
    [0.00044267 0.00044267 0.00044267 ... 0.00044267 0.00044267 0.00044267]
    [0.00044267 0.00044267 0.00044267 ... 0.00044267 0.00044267 0.00044267]]

```

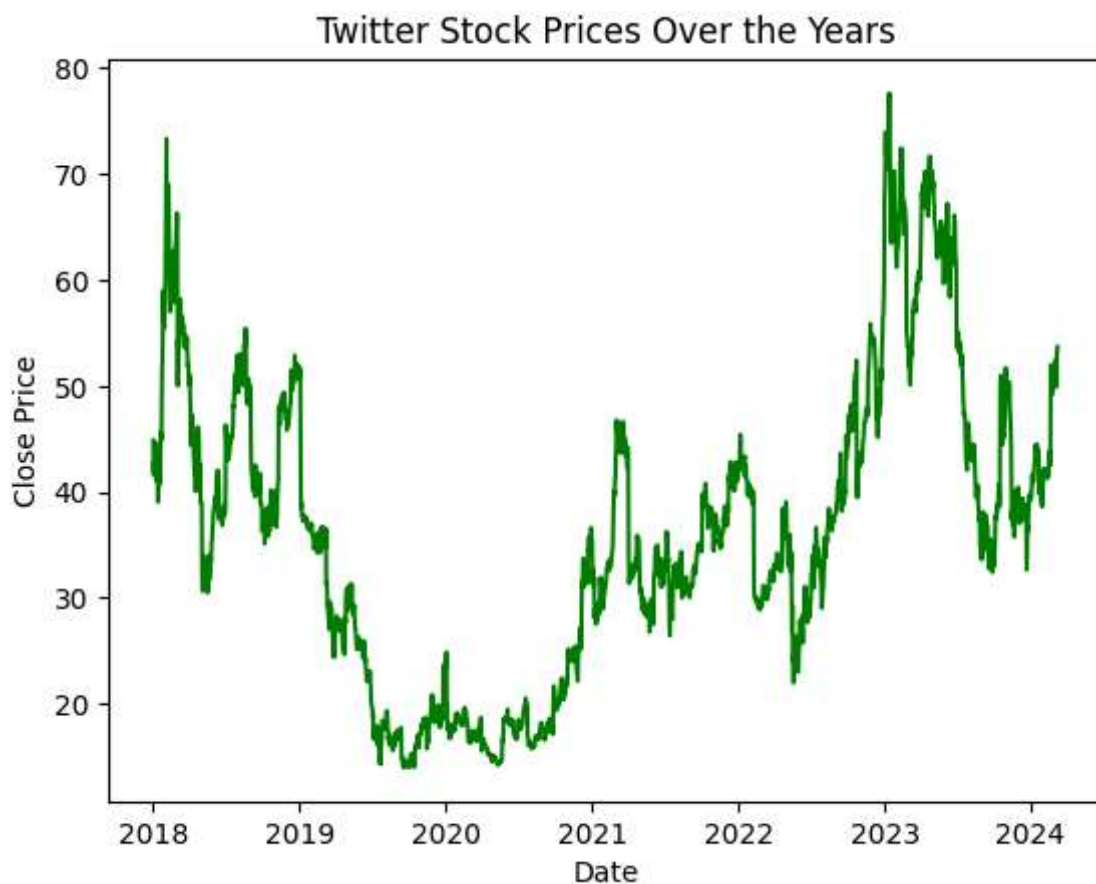
✓ Task 3: Visualization and Analysis

1. Show the Twitter stock prices over the years and give a conclusion:

```

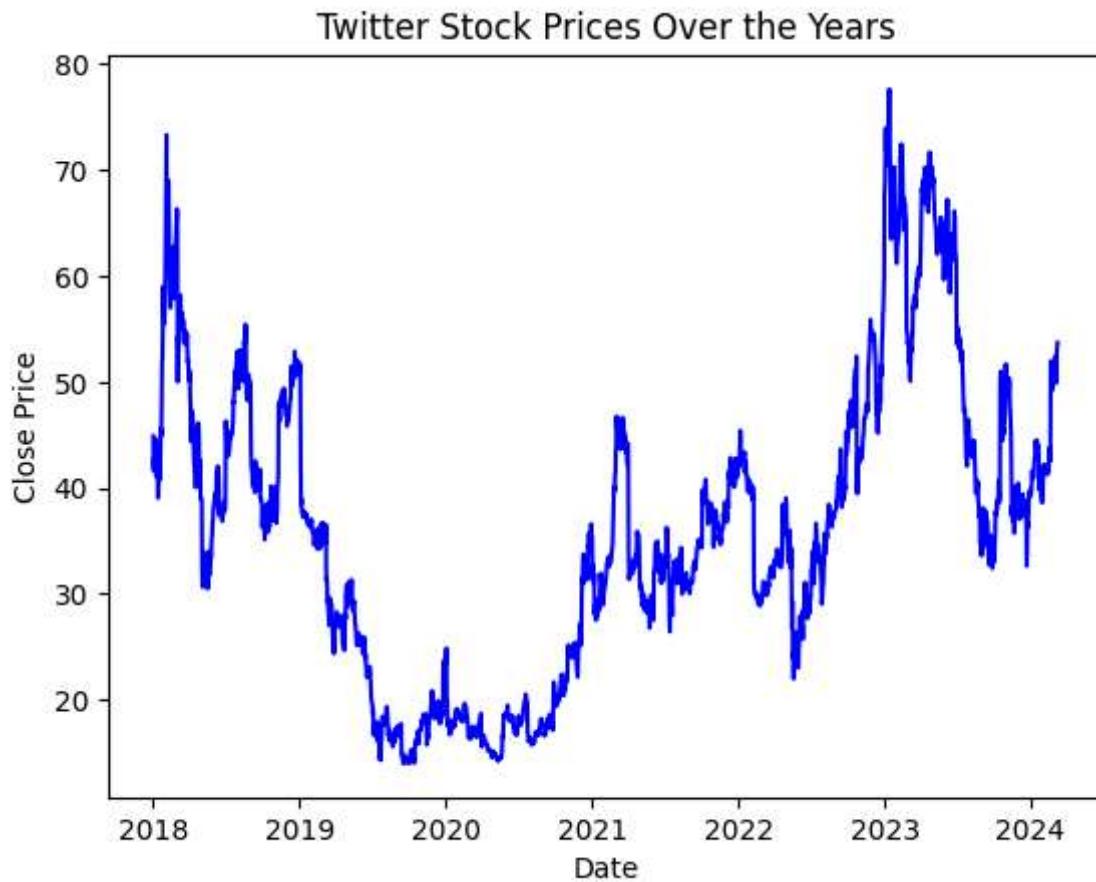
import matplotlib.pyplot as plt
plt.plot(ev_data['Date'], ev_data['Close'], color = 'green')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Twitter Stock Prices Over the Years')
plt.show()

```



2. Compare the Close vs Date column for Twitter prices over the years:

```
import matplotlib.pyplot as plt
plt.plot(ev_data['Date'], ev_data['Close'], color = 'blue')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Twitter Stock Prices Over the Years')
plt.show()
```



3. Assign buttons to control time periods. Add the buttons to analyze the stock prices of

```
import ipywidgets as widgets
from IPython.display import display
start_date = widgets.DatePicker(description='Start Date')
end_date = widgets.DatePicker(description='End Date')
display(start_date, end_date)
```

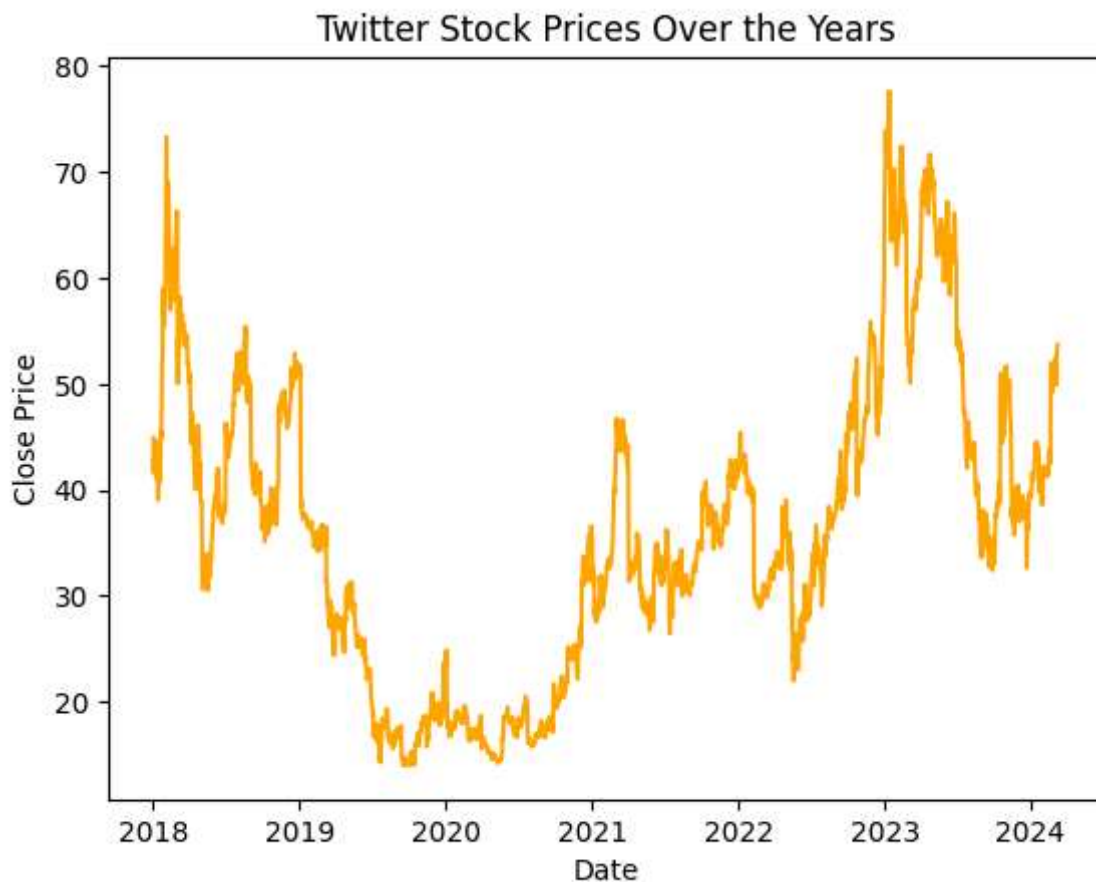


Start Date

End Date

4. Give the complete timeline of Twitter in the stock market (Line Graph):

```
import matplotlib.pyplot as plt
plt.plot(ev_data['Date'], ev_data['Close'], color = 'orange')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Twitter Stock Prices Over the Years')
plt.show()
```



5. Give the insights for the above analysis and make a word cloud for that analysis:

```
from wordcloud import WordCloud
text = ' '.join(ev_data)
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```




Date⁰Open High

6. Create a Dashboard from the above statements by using Power BI, Tableau, or Python and

```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import ipywidgets as widgets
from IPython.display import display
import plotly.express as px
import plotly.graph_objects as go
path = ('/content/TWTR.xlsx')
ev_data = pd.read_excel(path)
ev_data.head()
```