Historical Stock or Revenue Data

October 18, 2024

Extracting and Visualizing Stock Data

Description

Extracting essential data from a dataset and displaying it is a necessary part of data science; therefore individuals can make correct decisions based on the data. In this assignment, you will extract some stock data, you will then display this data in a graph.

Table of Contents

Estimated Time Needed: 30 min

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it.

[ ]:
```
#!pip install yfinance==0.2.38
#!pip install pandas==2.2.2
#!pip install nbformat
```

[ ]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

[12]:
```
import yfinance as yf
import pandas as pd
import requests
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

[13]:
```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

0.1 Define Graphing Function

In this section, we define the function `make_graph`. **You don't have to know how the function works, you should only care about the inputs. It takes a dataframe with stock data (dataframe must contain Date and Close columns), a dataframe with revenue data (dataframe must contain Date and Revenue columns), and the name of the stock.**

[1]:
```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True,␣
 ↪subplot_titles=("Historical Share Price", "Historical Revenue"),␣
 ↪vertical_spacing = .3)
    stock_data_specific = stock_data[stock_data.Date <= '2021--06-14']
    revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date),␣
 ↪y=stock_data_specific.Close.astype("float"), name="Share Price"), row=1,␣
 ↪col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date),␣
 ↪y=revenue_data_specific.Revenue.astype("float"), name="Revenue"), row=2,␣
 ↪col=1)
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue ($US Millions)", row=2, col=1)
    fig.update_layout(showlegend=False,
    height=900,
    title=stock,
    xaxis_rangeslider_visible=True)
    fig.show()
```

Use the make_graph function that we've already defined. You'll need to invoke it in questions 5 and 6 to display the graphs and create the dashboard. > **Note: You don't need to redefine the function for plotting graphs anywhere else in this notebook; just use the existing function.**

0.2 Question 1: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is `TSLA`.

[15]:
```
import yfinance as yf

# Create a ticker object for Tesla
ticker = yf.Ticker("TSLA")
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `tesla_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

[16]:
```
# Use the history function to extract stock data
tesla_data = ticker.history(period="max")

# Display the first few rows of the dataframe to verify
print(tesla_data.head())
```

```
                                 Open      High       Low     Close     Volume  \
Date
2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667  281494500
2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667  257806500
2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000  123282000
2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000   77097000
2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000  103003500

                           Dividends  Stock Splits
Date
2010-06-29 00:00:00-04:00        0.0           0.0
2010-06-30 00:00:00-04:00        0.0           0.0
2010-07-01 00:00:00-04:00        0.0           0.0
2010-07-02 00:00:00-04:00        0.0           0.0
2010-07-06 00:00:00-04:00        0.0           0.0
```

Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

[17]:
```
# Reset the index of tesla_data DataFrame
tesla_data.reset_index(inplace=True)

# Display the first five rows of tesla_data
print(tesla_data.head())
```

```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  103003500        0.0           0.0
```

0.3 Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm Save the text of the response as a variable named `html_data`.

[18]:
```
import requests

# URL of the webpage to download
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

# Send an HTTP GET request to the URL
response = requests.get(url)

# Get the HTML content of the response
html_data = response.text
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`. Make sure to use the `html_data` with the content parameter as follow `html_data.content` .

[19]:
```
from bs4 import BeautifulSoup

# Parse the HTML data using BeautifulSoup
soup = BeautifulSoup(html_data, 'html.parser')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`.

Step-by-step instructions

```
Here are the step-by-step instructions:

1. Find All Tables: Start by searching for all HTML tables on a webpage using `soup.find_all('t
2. Identify the Relevant Table: then loops through each table. If a table contains the text "Te
3. Initialize a DataFrame: Create an empty Pandas DataFrame called `tesla_revenue` with columns
4. Loop Through Rows: For each row in the relevant table, extract the data from the first and s
5. Clean Revenue Data: Remove dollar signs and commas from the revenue value.
6. Add Rows to DataFrame: Create a new row in the DataFrame with the extracted date and cleaned
7. Repeat for All Rows: Continue this process for all rows in the table.
```

Click here if you need help locating the table

```
Below is the code to isolate the table, you will now need to loop through the rows and columns

soup.find_all("tbody")[1]

If you want to use the read_html function the table is located at index 1
```

We are focusing on quarterly revenue in the lab.

[20]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup

# Step 1: Download the webpage content
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
html_data = requests.get(url).text

# Step 2: Parse the HTML content
soup = BeautifulSoup(html_data, 'html.parser')

# Step 3: Find the relevant table containing Tesla Quarterly Revenue
tables = soup.find_all('table')
tesla_revenue_table = None

for table in tables:
    if "Tesla Quarterly Revenue" in table.text:
        tesla_revenue_table = table
        break

# Step 4: Initialize an empty list to store data
data = []

# Step 5: Loop through rows in the table and extract Date and Revenue
for row in tesla_revenue_table.find_all("tr")[1:]:
    col = row.find_all("td")
    date = col[0].text.strip()
    revenue = col[1].text.strip().replace('$', '').replace(',', '')  # Clean␣
 ↪revenue data
    data.append({"Date": date, "Revenue": revenue})

# Step 6: Create the DataFrame from the list of dictionaries
tesla_revenue = pd.DataFrame(data)

# Step 7: Display the first few rows of the tesla_revenue DataFrame
print(tesla_revenue.head())
```

```
         Date Revenue
0  2022-09-30   21454
1  2022-06-30   16934
2  2022-03-31   18756
3  2021-12-31   17719
4  2021-09-30   13757
```

Execute the following line to remove the comma and dollar sign from the `Revenue` column.

[21]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

Execute the following lines to remove an null or empty strings in the Revenue column.

[22]:
```
tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

Display the last 5 row of the `tesla_revenue` dataframe using the `tail` function. Take a screenshot of the results.

[23]:
```
print(tesla_revenue.tail())
```

```
          Date Revenue
48  2010-09-30      31
49  2010-06-30      28
50  2010-03-31      21
52  2009-09-30      46
53  2009-06-30      27
```

0.4 Question 3: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is GameStop and its ticker symbol is `GME`.

[24]:
```
import yfinance as yf

# Step 1: Create a ticker object for GameStop (GME)
ticker_gme = yf.Ticker("GME")

# Step 2: Use the ticker object to extract historical stock data
gme_data = ticker_gme.history(period="max")

# Step 3: Reset the index of the DataFrame
gme_data.reset_index(inplace=True)

# Step 4: Display the first five rows of the gme_data DataFrame
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716073  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658002  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662209  1.603296  1.662209   6892800

   Dividends  Stock Splits
0        0.0           0.0
```

```
1         0.0           0.0
2         0.0           0.0
3         0.0           0.0
4         0.0           0.0
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `gme_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

[25]:
```
import yfinance as yf

# Step 1: Create a ticker object for GameStop (GME)
ticker_gme = yf.Ticker("GME")

# Step 2: Use the ticker object to extract historical stock data
gme_data = ticker_gme.history(period="max")

# Step 3: Reset the index of the DataFrame
gme_data.reset_index(inplace=True)

# Step 4: Display the first five rows of the gme_data DataFrame
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712708  1.716074  1.670626  1.683251  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658002  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666417  1.666417  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615921  1.662210  1.603296  1.662210   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

Reset the index using the `reset_index(inplace=True)` function on the gme_data DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

[26]:
```
import yfinance as yf

# Step 1: Create a ticker object for GameStop (GME)
ticker_gme = yf.Ticker("GME")

# Step 2: Use the ticker object to extract historical stock data
gme_data = ticker_gme.history(period="max")
```

```
# Step 3: Reset the index of the DataFrame
gme_data.reset_index(inplace=True)

# Step 4: Display the first five rows of the gme_data DataFrame
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658002  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666417  1.666417  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662209  1.603296  1.662209   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

0.5 Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html. Save the text of the response as a variable named `html_data_2`.

[27]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL of the webpage to scrape
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

# Send an HTTP request to the webpage
response = requests.get(url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Extract the HTML content
    html_data_2 = response.text

    # Parse the HTML content using BeautifulSoup
    soup = BeautifulSoup(html_data_2, 'html.parser')

    # Assuming the table we need is the second table in the page, extract it
    tables = soup.find_all('table')
```

```
    # Locate the correct table by checking its content or position
    gme_revenue_table = tables[1]  # Assuming the second table contains the GME␣
↪revenue data

    # Initialize an empty list to store data row by row
    data = []

    # Loop through each row in the table (skipping the header row)
    for row in gme_revenue_table.find_all("tr")[1:]:
        col = row.find_all("td")
        date = col[0].text.strip()
        revenue = col[1].text.strip().replace('$', '').replace(',', '')  #␣
↪Clean revenue data

        # Append data to the list row by row
        data.append({"Date": date, "Revenue": revenue})

    # Create a DataFrame from the list of dictionaries
    gme_revenue = pd.DataFrame(data)

    # Display the last 5 rows of the gme_revenue DataFrame
    print(gme_revenue.tail())
else:
    print("Failed to retrieve webpage:", response.status_code)
```

```
          Date Revenue
57  2006-01-31    1667
58  2005-10-31     534
59  2005-07-31     416
60  2005-04-30     475
61  2005-01-31     709
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`.

[28]:
```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_data_2, 'html.parser')
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Revenue` and store it into a dataframe named `gme_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

Note: Use the method similar to what you did in question 2.

Click here if you need help locating the table

`Below is the code to isolate the table, you will now need to loop through the rows and columns`

`soup.find_all("tbody")[1]`

If you want to use the read_html function the table is located at index 1

[30]:
```
from bs4 import BeautifulSoup
import pandas as pd
import requests

# URL of the webpage to scrape
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

# Send an HTTP request to the webpage
response = requests.get(url)

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Find all tables in the HTML
tables = soup.find_all('table')

# Identify the table containing the GameStop Revenue (based on the index 1 as␣
 ↪per instructions)
gme_revenue_table = tables[1]

# Initialize an empty list to store data
data = []

# Loop through each row in the table, skipping the header row (index 0)
for row in gme_revenue_table.find_all("tr")[1:]:
    col = row.find_all("td")
    date = col[0].text.strip()
    revenue = col[1].text.strip().replace('$', '').replace(',', '')  # Clean␣
 ↪revenue data

    # Append data to the list
    data.append({"Date": date, "Revenue": revenue})

# Convert the list of dictionaries to a DataFrame
gme_revenue = pd.DataFrame(data)

# Display the last 5 rows of the gme_revenue DataFrame
print(gme_revenue.tail())
```

```
          Date Revenue
57  2006-01-31    1667
58  2005-10-31     534
59  2005-07-31     416
```

```
60  2005-04-30      475
61  2005-01-31      709
```

Display the last five rows of the `gme_revenue` dataframe using the `tail` function. Take a screenshot of the results.

[31]:
```
print(gme_revenue.tail())
```

```
          Date Revenue
57  2006-01-31    1667
58  2005-10-31     534
59  2005-07-31     416
60  2005-04-30     475
61  2005-01-31     709
```

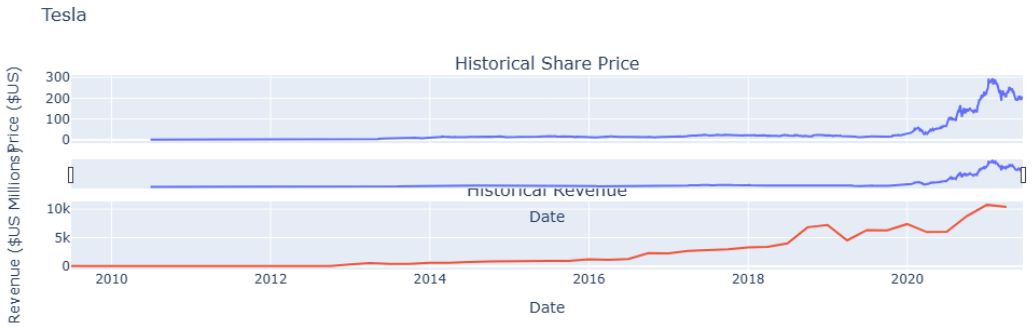
0.6 Question 5: Plot Tesla Stock Graph

Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

Hint

```
You just need to invoke the make_graph function with the required parameter to print the graphs
```

[32]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```



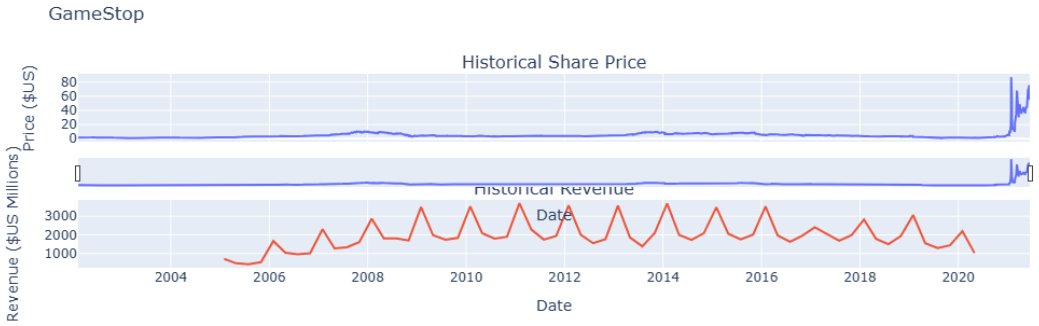
0.7 Question 6: Plot GameStop Stock Graph

Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`. Note the graph will only show data upto June 2021.

Hint

```
You just need to invoke the make_graph function with the required parameter to print the graphs
```

[33]:
```
make_graph(gme_data, gme_revenue, 'GameStop')
```



About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Azim Hirjani

0.8 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-02-28	1.2	Lakshmi Holla	Changed the URL of GameStop
2020-11-10	1.1	Malika Singla	Deleted the Optional part
2020-08-27	1.0	Malika Singla	Added lab to GitLab

##