Final Assignment

January 25, 2025

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

```
<ul>
    <li>Define a Function that Makes a Graph</li>
    <li>Question 1: Use yfinance to Extract Stock Data</li>
    <li>Question 2: Use Webscraping to Extract Tesla Revenue Data</li>
    <li>Question 3: Use yfinance to Extract Stock Data</li>
    <li>Question 4: Use Webscraping to Extract GME Revenue Data</li>
    <li>Question 5: Plot Tesla Stock Graph</li>
    <li>Question 6: Plot GameStop Stock Graph</li>
</ul>
```

Estimated Time Needed: 30 min

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it. Use the version as per your python version.

[1]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

```
Collecting yfinance
  Downloading yfinance-0.2.52-py2.py3-none-any.whl.metadata (5.8 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading
pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(89 kB)
Collecting numpy>=1.16.5 (from yfinance)
  Downloading
numpy-2.2.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(62 kB)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.32.3)
Collecting multitasking>=0.0.7 (from yfinance)
```

```
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Collecting lxml>=4.9.1 (from yfinance)
  Downloading lxml-5.3.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (3.8 kB)
Requirement already satisfied: platformdirs>=2.0.0 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (2.4.6)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.8.tar.gz (948 kB)
                          948.2/948.2 kB
52.6 MB/s eta 0:00:00
  Installing build dependencies … one
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Collecting html5lib>=1.1 (from yfinance)
  Downloading html5lib-1.1-py2.py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.12/site-
packages (from html5lib>=1.1->yfinance) (1.17.0)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.12/site-
packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance)
(2.9.0.post0)
Collecting tzdata>=2022.7 (from pandas>=1.3.0->yfinance)
  Downloading tzdata-2025.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.12/site-
packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.12/site-packages (from requests>=2.31->yfinance)
(2024.12.14)
Downloading yfinance-0.2.52-py2.py3-none-any.whl (108 kB)
Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
Downloading lxml-5.3.0-cp312-cp312-manylinux_2_28_x86_64.whl (4.9 MB)
                         4.9/4.9 MB
143.7 MB/s eta 0:00:00
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading
numpy-2.2.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.1 MB)
```

```
                          16.1/16.1 MB
165.7 MB/s eta 0:00:00
Downloading
pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.7
MB)
                          12.7/12.7 MB
179.4 MB/s eta 0:00:00
Downloading tzdata-2025.1-py2.py3-none-any.whl (346 kB)
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml) … one
  Created wheel for peewee:
filename=peewee-3.17.8-cp312-cp312-linux_x86_64.whl size=303769
sha256=dfe0b06056ffe8026dbec4ce844470ac7fa8d20d2020ca935f686c71c9798772
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/8f/65/34/456800445efea
fb05164fe95285c70e81ba1d96bae30f43917
Successfully built peewee
Installing collected packages: peewee, multitasking, tzdata, numpy, lxml,
html5lib, pandas, yfinance
Successfully installed html5lib-1.1 lxml-5.3.0 multitasking-0.0.11 numpy-2.2.2
pandas-2.2.3 peewee-3.17.8 tzdata-2025.1 yfinance-0.2.52
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.12/site-
packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4->bs4) (2.5)
Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
Installing collected packages: bs4
Successfully installed bs4-0.0.2
Requirement already satisfied: nbformat in /opt/conda/lib/python3.12/site-
packages (5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (4.23.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/opt/conda/lib/python3.12/site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/lib/python3.12/site-
packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.12/site-
packages (from jsonschema>=2.6->nbformat) (24.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in
/opt/conda/lib/python3.12/site-packages (from jsonschema>=2.6->nbformat)
(0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.12/site-
```

```
packages (from jsonschema>=2.6->nbformat) (0.22.3)
Requirement already satisfied: platformdirs>=2.5 in
/opt/conda/lib/python3.12/site-packages (from jupyter-
core!=5.0.*,>=4.12->nbformat) (4.3.6)
```

[2]:
```
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.

[3]:
```
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.**

[4]:
```
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,␣
 ↪infer_datetime_format=True), 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,␣
 ↪infer_datetime_format=True), 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`.

```
[5]:  Tesla = 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.

```
[6]: tesla_data = Tesla.history(period='max')
```

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.

```
[7]: tesla_data.reset_index(inplace=True)
     tesla_data.head()
```

```
[7]:                        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`.

```
[8]: 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
```

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

[9]:
```
Soup = BeautifulSoup(html_data,'html5lib')
Soup.find_all('title')
```

[9]:
```
[<title>Tesla Revenue 2010-2022 | TSLA | MacroTrends</title>]
```

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`.

[12]:
```
tesla_revenue = pd.DataFrame(columns=['Data','Revenue'])

for row in Soup.find_all('tbody')[1].find_all('tr'):
    col = row.find_all('td')
    date = col[0].text
    revenue = col[1].text

    tesla_revenue = tesla_revenue.append({'Date':date,'Revenue':revenue},␣
 ↪ignore_index = True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_132/1201930337.py in ?()
      4     col = row.find_all('td')
      5     date = col[0].text
      6     revenue = col[1].text
      7 
----> 8     tesla_revenue = tesla_revenue.append({'Date':date,'Revenue':
 ↪revenue}, ignore_index = True)

/opt/conda/lib/python3.12/site-packages/pandas/core/generic.py in ?(self, name)
   6295             and name not in self._accessors
   6296             and self._info_axis.
 ↪_can_hold_identifiers_and_holds_name(name)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'append'
```

Step-by-step instructions

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

1. Create an Empty DataFrame
2. Find the Relevant Table
3. Check for the Tesla Quarterly Revenue Table
4. Iterate Through Rows in the Table Body
5. Extract Data from Columns
```

```
6. Append Data to the DataFrame
```

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.
```

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

[12]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

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

[13]:
```
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.

[14]:
```
tesla_revenue.tail()
```

[14]:
```
    Date Revenue
8   2013  $2,013
9   2012    $413
10  2011    $204
11  2010    $117
12  2009    $112
```

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`.

[33]:
```
GameStop = yf.Ticker('GME')
```

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.

[34]:
```
gme_data = GameStop.history(period='max')
```

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.

```
[36]: gme_data.reset_index(inplace = True)
      gme_data.head()
```

```
[36]:    index                      Date      Open      High       Low     Close  \
      0      0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666
      1      1 2002-02-14 00:00:00-05:00  1.712708  1.716074  1.670626  1.683251
      2      2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834
      3      3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504
      4      4 2002-02-20 00:00:00-05:00  1.615921  1.662210  1.603296  1.662210

           Volume  Dividends  Stock Splits
      0  76216000        0.0           0.0
      1  11021600        0.0           0.0
      2   8389600        0.0           0.0
      3   7410400        0.0           0.0
      4   6892800        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`.

```
[15]: url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
        ↪IBMDeveloperSkillsNetwork/labs/project/stock.html'
      html_data_2= requests.get(url).text
```

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

```
[ ]: soup = BeautifulSoup(html_data_2,'html.parser')
     table = soup.find('table')
     rows = table.find_all('tr')
```

```
/opt/conda/lib/python3.12/html/parser.py:171: XMLParsedAsHTMLWarning: It looks
like you're parsing an XML document using an HTML parser. If this really is an
HTML document (maybe it's XHTML?), you can ignore or filter this warning. If
it's XML, you should know that using an XML parser will be more reliable. To
parse this document as XML, make sure you have the lxml package installed, and
pass the keyword argument `features="xml"` into the BeautifulSoup constructor.
  k = self.parse_starttag(i)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[31], line 3
      1 soup = BeautifulSoup(html_data_2,'html.parser')
      2 table = soup.find('table')
----> 3 rows = table.find_all('tr')
```

```
AttributeError: 'NoneType' object has no attribute 'find_all'
```

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]:
```
gme_revenue = pd.DataFrame(columns=['Date','Revenue'])

for row in Soup.find_all('tbody').find_all('tr'):
    col = row.find_all('td')
    date = col[0].text
    revenue = col[1].text

    gme_revenue =pd.concat([gme_revenue,df.DataFrame({'Date':[date],'Revenue':
 ↪[revenue]})],ignore_index = True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[30], line 3
      1 gme_revenue = pd.DataFrame(columns=['Date','Revenue'])
----> 3 for row in Soup.find_all('tbody').find_all('tr'):
      4     col = row.find_all('td')
      5     date = col[0].text

File /opt/conda/lib/python3.12/site-packages/bs4/element.py:2433, in ResultSet.
 ↪__getattr__(self, key)
   2431 def __getattr__(self, key):
   2432     """Raise a helpful exception to explain a common code fix."""
-> 2433     raise AttributeError(
   2434         "ResultSet object has no attribute '%s'. You're probably␣
 ↪treating a list of elements like a single element. Did you call find_all()␣
 ↪when you meant to call find()?" % key
   2435     )
```

```
AttributeError: ResultSet object has no attribute 'find_all'. You're probably␣
↪treating a list of elements like a single element. Did you call find_all()␣
↪when you meant to call find()?
```

[30]:
```
gme_revenue = pd.DataFrame(columns=['Date','Revenue'])

for row in Soup.find_all('tbody').find_all('tr'):
    col = row.find_all('td')
    date = col[0].text
    revenue = col[1].text

    gme_revenue =pd.concat([gme_revenue,df.DataFrame({'Date':[date],'Revenue':
↪[revenue]})],ignore_index = True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[30], line 3
      1 gme_revenue = pd.DataFrame(columns=['Date','Revenue'])
----> 3 for row in Soup.find_all('tbody').find_all('tr'):
      4     col = row.find_all('td')
      5     date = col[0].text

File /opt/conda/lib/python3.12/site-packages/bs4/element.py:2433, in ResultSet.
 ↪__getattr__(self, key)
   2431 def __getattr__(self, key):
   2432     """Raise a helpful exception to explain a common code fix."""
-> 2433     raise AttributeError(
   2434         "ResultSet object has no attribute '%s'. You're probably␣
 ↪treating a list of elements like a single element. Did you call find_all()␣
 ↪when you meant to call find()?" % key
   2435     )

AttributeError: ResultSet object has no attribute 'find_all'. You're probably␣
 ↪treating a list of elements like a single element. Did you call find_all()␣
 ↪when you meant to call find()?
```

[ ]:

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

[ ]:

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
```

```
[ ]:
```

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
```

```
[ ]:
```

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

##