Final Assignment

March 1, 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

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
!pip install --upgrade plotly
```

```
Requirement already satisfied: yfinance in /opt/conda/lib/python3.12/site-
packages (0.2.54)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.2.3)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.2.3)
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (0.0.11)
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)
Requirement already satisfied: peewee>=3.16.2 in /opt/conda/lib/python3.12/site-
packages (from yfinance) (3.17.9)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/opt/conda/lib/python3.12/site-packages (from yfinance) (4.13.3)
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: typing-extensions>=4.0.0 in
/opt/conda/lib/python3.12/site-packages (from beautifulsoup4>=4.11.1->yfinance)
(4.12.2)
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)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-
packages (from pandas>=1.3.0->yfinance) (2025.1)
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)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
Requirement already satisfied: bs4 in /opt/conda/lib/python3.12/site-packages
(0.0.2)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.12/site-
packages (from bs4) (4.13.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-
packages (from beautifulsoup4->bs4) (2.5)
Requirement already satisfied: typing-extensions>=4.0.0 in
/opt/conda/lib/python3.12/site-packages (from beautifulsoup4->bs4) (4.12.2)
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) (25.1.0)
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)
(0.36.2)
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)
Requirement already satisfied: typing-extensions>=4.4.0 in
/opt/conda/lib/python3.12/site-packages (from
referencing>=0.28.4->jsonschema>=2.6->nbformat) (4.12.2)
Requirement already satisfied: plotly in /opt/conda/lib/python3.12/site-packages
(5.24.1)
Collecting plotly
  Downloading plotly-6.0.0-py3-none-any.whl.metadata (5.6 kB)
Collecting narwhals>=1.15.1 (from plotly)
  Downloading narwhals-1.28.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: packaging in /opt/conda/lib/python3.12/site-
packages (from plotly) (24.2)
Downloading plotly-6.0.0-py3-none-any.whl (14.8 MB)
                         14.8/14.8 MB
140.5 MB/s eta 0:00:00
Downloading narwhals-1.28.0-py3-none-any.whl (308 kB)
Installing collected packages: narwhals, plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 5.24.1
    Uninstalling plotly-5.24.1:
      Successfully uninstalled plotly-5.24.1
Successfully installed narwhals-1.28.0 plotly-6.0.0
```

[2]:
```python
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
```

[3]:
```python
import plotly.io as pio
pio.renderers.default = "iframe"
```

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.

[4]:
```
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.**

[5]:
```
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()
    from IPython.display import display, HTML
    fig_html = fig.to_html()
    display(HTML(fig_html))
```

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`.

[6]:
```
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.

[8]:
```
tesla_date = 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.

[72]:
```
tesla_date.head()
```

[72]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.0	0.0
2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.0	0.0
2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.0	0.0
2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.0	0.0
2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	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`.

[52]:
```
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`.

[53]:
```
soup = BeautifulSoup(html_data, 'html5lib')
```

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. 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.

[54]:
```
tesla_revenue = pd.DataFrame(columns = ["Date", "Revenue"])
table = soup.find("table")
table_rows = table.find_all("tr")

df_list = []
for row in table_rows[1:]:
    cols = row.find_all("td")
    if cols:
        date = cols[0].text.strip()
        revenue = cols[1].text.strip()
        df_list.append(pd.DataFrame({"Date": [date], "Revenue": [revenue]}))

tesla_revenue = pd.concat(df_list, ignore_index=True)
```

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

[55]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

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

[56]:
```
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.

[57]:
```
tesla_revenue.tail()
```

[57]:

	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`.

[38]:
```
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.

[42]:
```
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.

[41]:
```
Gme_data.reset_index(inplace=True)
Gme_data.head()
```

[41]:

	index	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	0	2002-02-13 00:00:00-05:00	1.620128	1.693350	1.603296	1.691666	76216000	0.0	0.0
1	1	2002-02-14 00:00:00-05:00	1.712708	1.716074	1.670627	1.683251	11021600	0.0	0.0
2	2	2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	1.674834	8389600	0.0	0.0
3	3	2002-02-19 00:00:00-05:00	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	4	2002-02-20 00:00:00-05:00	1.615920	1.662209	1.603295	1.662209	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`.

[44]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
  ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/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`.

[45]:
```
soup = BeautifulSoup(html_data_2)
```

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
```

[50]:
```
gme_revenue = pd.DataFrame(columns = ["Date", "Revenue"])
table = soup.find("table")
table_rows = table.find_all("tr")

df_list = []
for row in table_rows[1:]:
    cols = row.find_all("td")
    if cols:
        date = cols[0].text.strip()
        revenue = cols[1].text.strip()
        df_list.append(pd.DataFrame({"Date": [date], "Revenue": [revenue]}))

gme_revenue = pd.concat(df_list, ignore_index=True)
```

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

[69]:
```
gme_revenue.tail()
```

[69]:

	Date	Revenue
11	2009	$8,806
12	2008	$7,094
13	2007	$5,319
14	2006	$3,092
15	2005	$1,843

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

[75]:
```
tesla_revenue["Revenue"] = tesla_revenue["Revenue"].str.replace(',|\$', "",␣
↪regex=True)
tesla_revenue["Revenue"] = tesla_revenue["Revenue"].astype(float)

tesla_date = tesla_date.reset_index()
make_graph(tesla_date, tesla_revenue, "Tesla stock")
```

```
/tmp/ipykernel_2160/109047474.py:5: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.

/tmp/ipykernel_2160/109047474.py:6: UserWarning:

The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.


<IPython.core.display.HTML object>
```

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

[77]:
```
gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(',|\$', "",␣
↪regex=True)
gme_revenue["Revenue"] = gme_revenue["Revenue"].astype(float)

df_list = df_list.reset_index()
make_graph(df_list, gme_revenue, "GME")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[77], line 1
----> 1 gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace(',|\$', "",
↪regex=True)
```

```
      2 gme_revenue["Revenue"] = gme_revenue["Revenue"].astype(float)
      4 df_list = df_list.reset_index()

File /opt/conda/lib/python3.12/site-packages/pandas/core/generic.py:6299, in␣
 ↪NDFrame.__getattr__(self, name)
   6292 if (
   6293     name not in self._internal_names_set
   6294     and name not in self._metadata
   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)

File /opt/conda/lib/python3.12/site-packages/pandas/core/accessor.py:224, in␣
 ↪CachedAccessor.__get__(self, obj, cls)
    221 if obj is None:
    222     # we're accessing the attribute of the class, i.e., Dataset.geo
    223     return self._accessor
--> 224 accessor_obj = self._accessor(obj)
    225 # Replace the property with the accessor object. Inspired by:
    226 # https://www.pydanny.com/cached-property.html
    227 # We need to use object.__setattr__ because we overwrite __setattr__ on
    228 # NDFrame
    229 object.__setattr__(obj, self._name, accessor_obj)

File /opt/conda/lib/python3.12/site-packages/pandas/core/strings/accessor.py:
 ↪191, in StringMethods.__init__(self, data)
    188 def __init__(self, data) -> None:
    189     from pandas.core.arrays.string_ import StringDtype
--> 191     self._inferred_dtype = self._validate(data)
    192     self._is_categorical = isinstance(data.dtype, CategoricalDtype)
    193     self._is_string = isinstance(data.dtype, StringDtype)

File /opt/conda/lib/python3.12/site-packages/pandas/core/strings/accessor.py:
 ↪245, in StringMethods._validate(data)
    242 inferred_dtype = lib.infer_dtype(values, skipna=True)
    244 if inferred_dtype not in allowed_types:
--> 245     raise AttributeError("Can only use .str accessor with string values
 ↪")
    246 return inferred_dtype

AttributeError: Can only use .str accessor with string values!
```

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

##