Analyzing Historical Stock-Revenue Data and Building a Dashboard

September 29, 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

```
<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.

```
[ ]: !python -m pip install pip setuptools wheel --upgrade
```

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

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

```
[4]: 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.

```
[5]: import warnings # Tüm uyarıları yoksay
     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.**

```
[10]: 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.

```
[8]: 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.

```
[9]: 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.

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

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`.

```
[20]: url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
      ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
      html_data  = requests.get(url)
```

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` .

```
[21]: soup = BeautifulSoup(html_data.content, '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.
> Note: Instead of using the deprecated pd.append() method, consider using pd.concat([df, pd.Da

```python
[46]: len(soup.find_all('table'))
      tables = soup.find_all('table')
      text = "Tesla Quarterly Revenue"
      table_num = 0
      for num in range(len(tables)):
          index = str(tables[num]).find(text)
          if index != -1:
              table_num = num
              break
```

```python
[55]: tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])
```

```python
[56]: for row in tables[1].find_all("tr"):
          if len(row.find_all("td"))==0:
              continue
          col = row.find_all("td")
          date = col[0].text
          revenue = col[1].text.replace("$", "").replace(",", "")

          tesla_revenue = pd.concat([tesla_revenue,pd.DataFrame({"Date":[date],␣
      ↪"Revenue":[revenue]})], ignore_index=True)
```

```python
[57]: tesla_revenue.head()
```

```
[57]:          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.

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

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

```python
[59]: 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.

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

```
[60]:           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`.

```
[61]: GameStop = yf.Ticker(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.

```
[63]: gme_data = GameStop.history(period = "max")
      gme_data.reset_index(inplace = True)
```

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.

```
[65]: gme_data.head()
```

```
[65]:                        Date      Open      High       Low     Close    Volume  \
      0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691667  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  1.658001  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  1.662210  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
```

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`.

```
[67]: 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)
```

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

```
[69]: soup = BeautifulSoup(html_data_2.content, '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.

```
[72]: len(soup.find_all('table'))
      tables = soup.find_all('table')
      text = "GameStop Quarterly Revenue"
      table_num = 0
      for num in range(len(tables)):
          index = str(tables[num]).find(text)
          if index != -1:
              table_num = num
              break
      gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])
      for row in tables[1].find_all("tr"):
          if len(row.find_all("td"))==0:
              continue
          col = row.find_all("td")
          date = col[0].text
          revenue = col[1].text.replace("$", "").replace(",", "")

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

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

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

```
[73]:           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
```
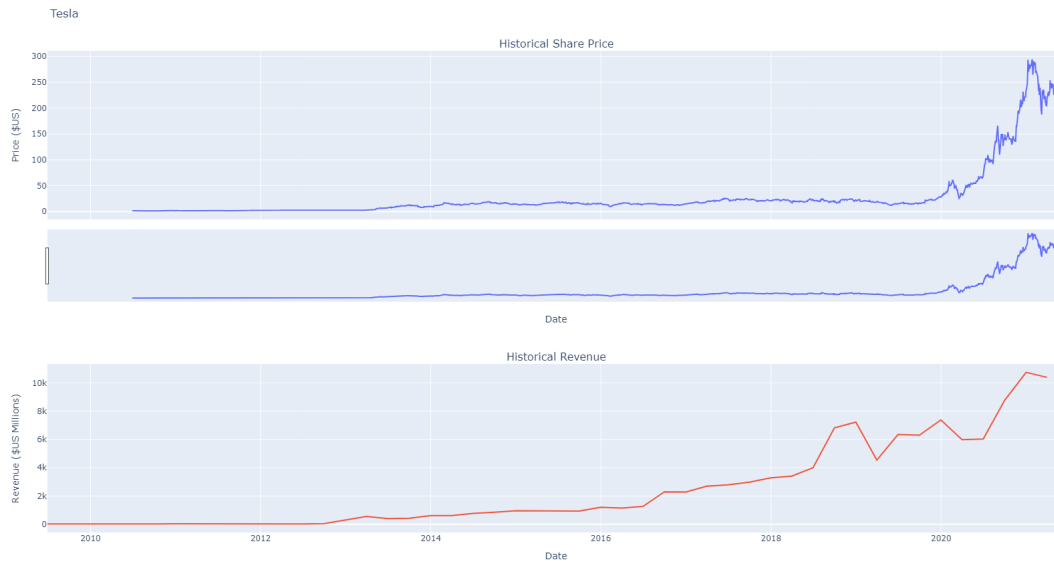
0.6   Question 5: Plot Tesla Stock Graph

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

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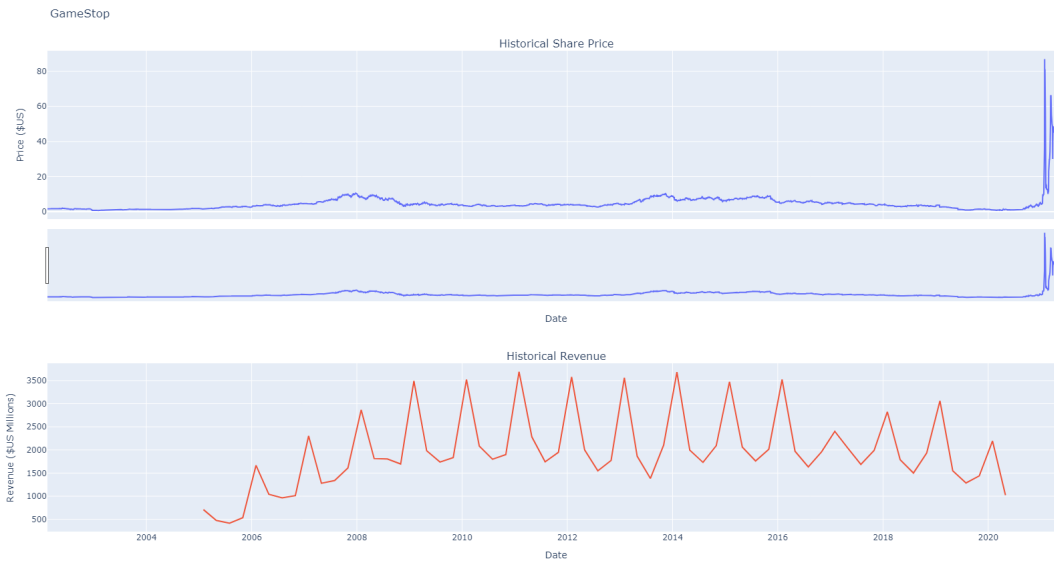

0.7   Question 6: Plot GameStop Stock Graph

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

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

GameStop

Historical Share Price

Price ($US)

80
60
40
20
0

Date

Historical Revenue

Revenue ($US Millions)

3500
3000
2500
2000
1500
1000
500

2004   2006   2008   2010   2012   2014   2016   2018   2020

Date

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.

##

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