Final Assignment_StephaneDEDIEU

April 28, 2022

Extracting and Visualizing Stock Data

Stephane DEDIEU, April 27th, 2022

Course: Python Project for Data Science

0.1 Preliminary remark.

My IBM Cloud trial period is over, and I was unable to reactivate it with the code provided in this course. Therefore I am unable to add this notebook to Watson Studio. For that reason, I shared it on my Github account. I apologize for the inconvenience.
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

[5]:
```
!pip install yfinance==0.1.67
#!pip install pandas==1.3.3
#!pip install requests==2.26.0
!mamba install bs4==4.10.0 -y
#!pip install plotly==5.3.1
```

```
Collecting yfinance==0.1.67
  Downloading yfinance-0.1.67-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: pandas>=0.24 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
yfinance==0.1.67) (1.3.5)
```

```
Requirement already satisfied: requests>=2.20 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
yfinance==0.1.67) (2.27.1)
Requirement already satisfied: lxml>=4.5.1 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
yfinance==0.1.67) (4.6.4)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.10.tar.gz (8.2 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: numpy>=1.15 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
yfinance==0.1.67) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7.3 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
pandas>=0.24->yfinance==0.1.67) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
pandas>=0.24->yfinance==0.1.67) (2022.1)
Requirement already satisfied: certifi>=2017.4.17 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
requests>=2.20->yfinance==0.1.67) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
requests>=2.20->yfinance==0.1.67) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
requests>=2.20->yfinance==0.1.67) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
requests>=2.20->yfinance==0.1.67) (2.0.12)
Requirement already satisfied: six>=1.5 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from python-
dateutil>=2.7.3->pandas>=0.24->yfinance==0.1.67) (1.16.0)
Building wheels for collected packages: multitasking
  Building wheel for multitasking (setup.py) … done
  Created wheel for multitasking: filename=multitasking-0.0.10-py3-none-
any.whl size=8500
sha256=512fbe4c2dc55523f867013edf98ba3b69c10ee861dd243dc4731217ef17a071
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/34/ba/79/c0260c6f1a03f
420ec7673eff9981778f293b9107974679e36
Successfully built multitasking
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.10 yfinance-0.1.67


                  --    --    --    --
                 / \  / \  / \  / \
                /   \/   \/   \/   \
        /  / /  / /  / /  /
```

```
              /  / \   / \   / \   / \  \____
             /  /   \_/   \_/   \_/   \    o \__,
            / _/                       \_____/  `
            |/




        mamba (0.22.1) supported by @QuantStack

        GitHub:  https://github.com/mamba-org/mamba
        Twitter: https://twitter.com/QuantStack




Looking for: ['bs4==4.10.0']

[+] 0.0s
[+] 0.1s
pkgs/main/linux-64                 0.0 B /  ??.?MB
@  ??.?MB/s  0.1s
pkgs/main/noarch                   0.0 B /  ??.?MB
@  ??.?MB/s  0.1s
pkgs/r/linux-64                    0.0 B /  ??.?MB
@  ??.?MB/s  0.1s
pkgs/r/noarch                      0.0 B
/  ??.?MB @  ??.?MB/s
0.1spkgs/main/noarch
No change
pkgs/r/linux-64                                             No change
pkgs/r/noarch                                               No change
pkgs/main/linux-64                                          No change

Pinned packages:
  - python 3.7.*


Transaction

  Prefix: /home/jupyterlab/conda/envs/python

  All requested packages already installed
```

[6]:
```
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
```

0.2 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.

[37]:
```
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=True,
    height=900,
    title=stock,
    xaxis_rangeslider_visible=True)
    fig.show()
```

0.3 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]:
```
tsla = 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]: tsla_share_price_data = tsla.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.

```
[10]: tsla_share_price_data.reset_index(inplace=True)
      tsla_share_price_data.head()
```

```
[10]:         Date   Open   High    Low  Close    Volume  Dividends  Stock Splits
      0 2010-06-29  3.800  5.000  3.508  4.778  93831500          0           0.0
      1 2010-06-30  5.158  6.084  4.660  4.766  85935500          0           0.0
      2 2010-07-01  5.000  5.184  4.054  4.392  41094000          0           0.0
      3 2010-07-02  4.600  4.620  3.742  3.840  25699000          0           0.0
      4 2010-07-06  4.000  4.000  3.166  3.222  34334500          0           0.0
```

0.4 Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage https://www.macrotrends.net/stocks/charts/TSLA/tesla/reve
Save the text of the response as a variable named `html_data`.

```
[11]: url="https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue?
        ↪utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_
      html_data  = requests.get(url).text
```

Parse the html data using `beautiful_soup`.

```
[12]: #soup = BeautifulSoup(html_data, 'html5lib')
      soup = BeautifulSoup(html_data, "html.parser")
```

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

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 use a technique learned in the Web scrapping lab. (1) We determine the number of tables on the webpage, and (2) the index of the table of interest: "Tesla Quarter Revenue".

[13]:
```
tables = soup.find_all('table')
print("Number of Tables:", len(tables))

for index,table in enumerate(tables):
    if ("Tesla Quarterly Revenue" in str(table)):
        table_index = index
print("Quaterly Revenue Table Index=", table_index)
```

```
Number of Tables: 6
Quaterly Revenue Table Index= 1
```

*** We find the index of the table of interest to be 1. (3) We build the dataframe after table with table_index=1.***

[14]:
```
#print(tables[table_index].prettify())
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

for row in tables[table_index].tbody.find_all("tr"):
    col = row.find_all("td")
    if (col != []):
        Date = col[0].text
        Revenue = col[1].text

        tesla_revenue = tesla_revenue.append({"Date":Date, "Revenue":Revenue},␣
 ↪ignore_index=True)

tesla_revenue.head(2)
```

[14]:
```
         Date  Revenue
0  2022-03-31  $18,756
1  2021-12-31  $15,339
```

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

[15]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will
change from True to False in a future version.
  """Entry point for launching an IPython kernel.
```

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

[16]:
```
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.

Note that the unit in the Revenue column is: millions of $.

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

```
[17]:          Date Revenue
      46  2010-09-30      31
      47  2010-06-30      28
      48  2010-03-31      21
      50  2009-09-30      46
      51  2009-06-30      27
```

0.5 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`.

```
[18]: gme = 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.

```
[19]: gme_share_price_data = gme.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.

```
[20]: gme_share_price_data.reset_index(inplace=True)
      gme_share_price_data.head()
```

```
[20]:         Date      Open      High       Low     Close    Volume  Dividends  \
      0 2002-02-13  6.480514  6.773400  6.413183  6.766666  19054000        0.0
      1 2002-02-14  6.850830  6.864296  6.682505  6.733002   2755400        0.0
      2 2002-02-15  6.733001  6.749833  6.632006  6.699336   2097400        0.0
      3 2002-02-19  6.665672  6.665672  6.312189  6.430017   1852600        0.0
      4 2002-02-20  6.463682  6.648839  6.413184  6.648839   1723200        0.0

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

0.6 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.

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

      gme_html_data  = requests.get(url).text
```

Parse the html data using beautiful_soup.

```
[22]: gme_soup = BeautifulSoup(gme_html_data, 'html.parser')
```

Using BeautifulSoup or the read_html function extract the table with GameStop Quarterly 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 using a 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
```

```
[23]: tables = gme_soup.find_all('table')
      print("Number of Tables:", len(tables))

      for index,table in enumerate(tables):
          if ("GameStop Quarterly Revenue" in str(table)):
              table_index = index
      print("Quaterly Revenue Table Index=", table_index)
```

```
Number of Tables: 6
Quaterly Revenue Table Index= 1
```

```
[24]: #print(tables[table_index].prettify())
      gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])

      for row in tables[table_index].tbody.find_all("tr"):
          col = row.find_all("td")
          if (col != []):
              Date = col[0].text
              Revenue = col[1].text

              gme_revenue = gme_revenue.append({"Date":Date, "Revenue":Revenue},␣
       ↪ignore_index=True)
```

```
gme_revenue.head(2)
```

[24]:
```
         Date Revenue
0  2020-04-30  $1,021
1  2020-01-31  $2,194
```

[25]:
```
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"")
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will
change from True to False in a future version.
  """Entry point for launching an IPython kernel.
```

[26]:
```
gme_revenue.dropna(inplace=True)
gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]
gme_revenue.head(2)
```

[26]:
```
         Date Revenue
0  2020-04-30    1021
1  2020-01-31    2194
```

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

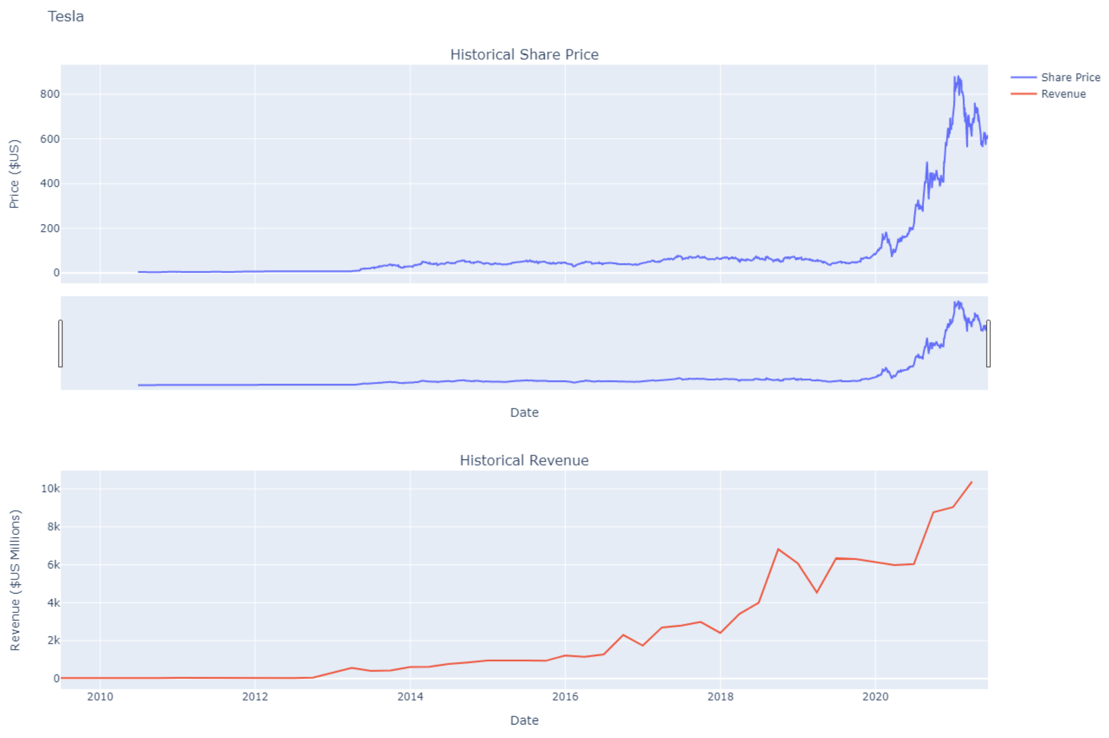
[27]:
```
gme_revenue.tail()
```

[27]:
```
          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
```

0.7 Question 5: Plot Tesla Stock Graph

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

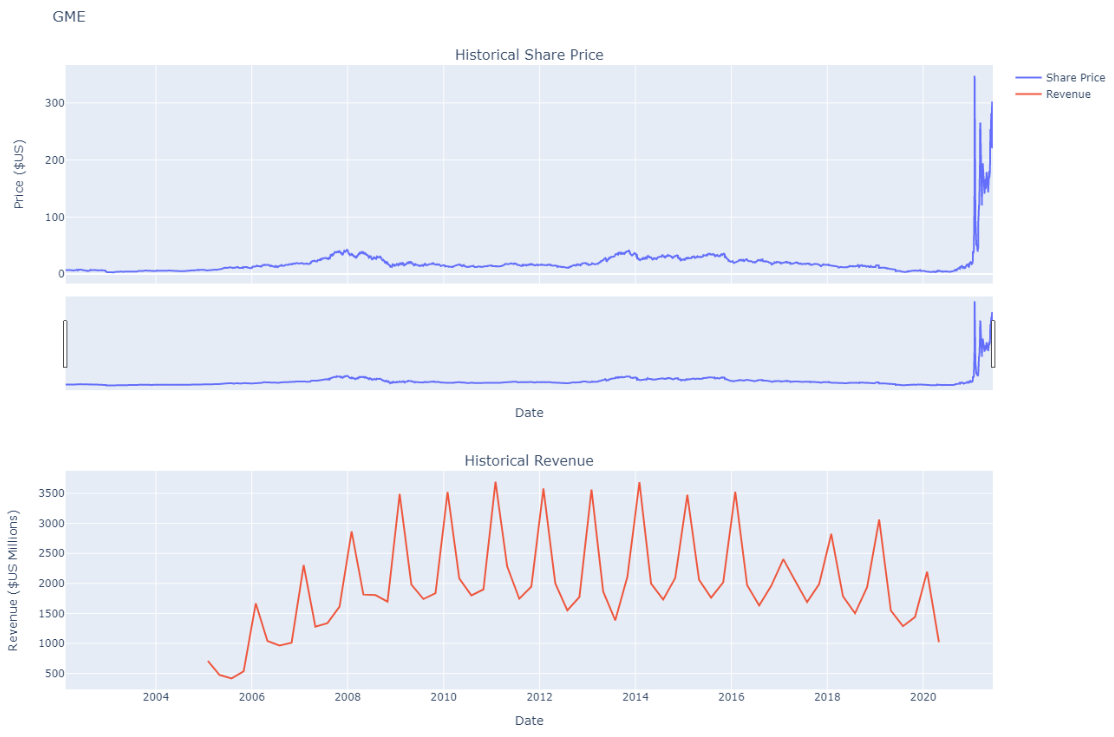
[38]:
```
make_graph(tsla_share_price_data, tesla_revenue, 'Tesla')
```



0.8 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.

[39]:
```
make_graph(gme_share_price_data, gme_revenue, 'GME')
```



0.8.1 Conclusions

1. Tesla revenues grew exponentially from 2010 to 2021. And the stock price increased accordingly over the same period of time. Although "revenue" is not the only criteria for the evolution of a stock price, Tesla stock price is correlated with the revenues and its evolution makes sense.

2. Gamestop revenues peaked in 2011 and then decreased and eventually collapsed in 2020. Meanwhile the stock price was skyrocketing in 2021, with erratic variations. This points to a highly speculative, risky asset, which price is totally disconnected with Gamestop financial performance.

```
[ ]:
```

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

0.9 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

##