

# Final Assignment\_solved

November 20, 2023

## 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

- <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>

Estimated Time Needed: 30 min

**Note:-** If you are working in IBM Cloud Watson Studio, please replace the command for installing nbformat from `!pip install nbformat==4.2.0` to simply `!pip install nbformat`

```
[1]: !pip install yfinance==0.1.67
!mamba install bs4==4.10.0 -y
!pip install nbformat==4.2.0
```

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: 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: requests>=2.20 in

/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.1.67) (2.29.0)

Collecting multitasking>=0.0.7 (from yfinance==0.1.67)

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  0.1s[+] 0.2s
pkgs/main/linux-64          57.4kB
/   ???.?MB @ 373.4kB/s    0.2s
pkgs/main/noarch            69.6kB /   ???.?MB
@ 454.0kB/s    0.2s
pkgs/r/linux-64             57.4kB /   ???.?MB
@ 373.2kB/s    0.2s
pkgs/r/noarch               41.0kB
/   ???.?MB @ 266.1kB/s    0.2s[+] 0.3s
pkgs/main/linux-64          577.5kB /   ???.?MB
@   2.3MB/s    0.3s
pkgs/main/noarch            614.4kB /   ???.?MB
@   2.4MB/s    0.3s
pkgs/r/linux-64             614.4kB /   ???.?MB
@   2.4MB/s    0.3s
pkgs/r/noarch               639.0kB /   ???.?MB
@   2.4MB/s    0.3spkgs/main/noarch
853.2kB @   2.8MB/s    0.3s
[+] 0.4s
pkgs/main/linux-64          1.3MB /   ???.?MB
@   3.4MB/s    0.4s
pkgs/r/linux-64             1.2MB /   ???.?MB
@   3.3MB/s    0.4s
pkgs/r/noarch               1.3MB /   ???.?MB
@   3.3MB/s    0.4s[+] 0.5s
pkgs/main/linux-64          1.9MB /   ???.?MB
@   3.9MB/s    0.5s
pkgs/r/linux-64             1.7MB /   ???.?MB
@   3.6MB/s    0.5s
pkgs/r/noarch               1.8MB /   ???.?MB
@   3.7MB/s    0.5spkgs/r/linux-64
1.9MB @   3.6MB/s    0.5s
[+] 0.6s
pkgs/main/linux-64          2.3MB @
```

```

4.2MB/s          0.6s
pkgs/r/noarch          2.3MB @ 3.9MB/s Finalizing
0.6spkgs/r/noarch          @
3.9MB/s 0.6s
[+] 0.7s
pkgs/main/linux-64      2.9MB / ???.?MB
@ 4.4MB/s 0.7s[+] 0.8s
pkgs/main/linux-64      3.3MB / ???.?MB
@ 4.4MB/s 0.8s[+] 0.9s
pkgs/main/linux-64      3.9MB / ???.?MB
@ 4.5MB/s 0.9s[+] 1.0s
pkgs/main/linux-64      4.5MB / ???.?MB
@ 4.7MB/s 1.0s[+] 1.1s
pkgs/main/linux-64      5.1MB
/ ???.?MB @ 4.8MB/s 1.1s[+] 1.2s
pkgs/main/linux-64      5.7MB
/ ???.?MB @ 4.9MB/s 1.2s[+] 1.3s
pkgs/main/linux-64      6.3MB
/ ???.?MB @ 5.0MB/s 1.3s[+] 1.4s
pkgs/main/linux-64      6.4MB @ 5.0MB/s Finalizing
1.4s[+] 1.5s
pkgs/main/linux-64          @ 5.0MB/s
1.4s

```

Pinned packages:

- python 3.7.\*

Transaction

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

Updating specs:

- bs4==4.10.0
- ca-certificates
- certifi
- openssl

Package	Version	Build	Channel	Size
---------	---------	-------	---------	------

Install:

+ bs4	4.10.0	hd3eb1b0_0	pkgs/main/noarch	10kB
-------	--------	------------	------------------	------

Upgrade:

```
- ca-certificates    2023.5.7  hbcca054_0    conda-forge
+ ca-certificates    2023.08.22 h06a4308_0    pkgs/main/linux-64
125kB
- openssl            1.1.1t    h0b41bf4_0    conda-forge
+ openssl            1.1.1w    h7f8727e_0    pkgs/main/linux-64
4MB
```

Downgrade:

```
- beautifulsoup4     4.11.1    pyha770c72_0  conda-forge
+ beautifulsoup4     4.10.0    pyh06a4308_0  pkgs/main/noarch
87kB
```

Summary:

Install: 1 packages  
Upgrade: 2 packages  
Downgrade: 1 packages

Total download: 4MB

```
[+] 0.0s
Downloading          0.0 B
0.0s
Extracting           0
0.0s[+] 0.1s
Downloading (4)      0.0 B beautifulsoup4
0.0s
Extracting           0
0.0sca-certificates
125.5kB @ 916.0kB/s  0.1s
beautifulsoup4      86.6kB @ 554.9kB/s  0.2s
openssl              3.9MB @ 22.4MB/s  0.2s
bs4                  10.2kB @ 58.2kB/s  0.2s
[+] 0.2s
Downloading          4.1MB          0.1s
Extracting (4)       0
beautifulsoup4       0.0s[+] 0.3s
Downloading          4.1MB          0.1s
Extracting (4)       0
beautifulsoup4       0.1s[+] 0.4s
```

Downloading	4.1MB	0.1s
Extracting (4)	0	
beautifulsoup4	0.2s[+] 0.5s	
Downloading	4.1MB	0.1s
Extracting (4)	0	
beautifulsoup4	0.3s[+] 0.6s	
Downloading	4.1MB	0.1s
Extracting (4)	0 bs4	
0.4s[+] 0.7s		
Downloading	4.1MB	0.1s
Extracting (4)	0 bs4	
0.5s[+] 0.8s		
Downloading	4.1MB	0.1s
Extracting (4)	0 bs4	
0.6s[+] 0.9s		
Downloading	4.1MB	0.1s
Extracting (4)	0 bs4	
0.7s[+] 1.0s		
Downloading	4.1MB	0.1s
Extracting (4)	0 ca-	
certificates	0.8s[+] 1.1s	
Downloading	4.1MB	0.1s
Extracting (4)	0 ca-	
certificates	0.9s[+] 1.2s	
Downloading	4.1MB	0.1s
Extracting (4)	0 ca-	
certificates	1.0s[+] 1.3s	
Downloading	4.1MB	0.1s
Extracting (4)	0 ca-	
certificates	1.1s[+] 1.4s	
Downloading	4.1MB	0.1s
Extracting (4)	0 openssl	
1.2s[+] 1.5s		
Downloading	4.1MB	0.1s
Extracting (4)	0 openssl	
1.3s[+] 1.6s		
Downloading	4.1MB	0.1s
Extracting (4)	0 openssl	
1.4s[+] 1.7s		
Downloading	4.1MB	0.1s
Extracting (4)	0 openssl	
1.5s[+] 1.8s		
Downloading	4.1MB	0.1s
Extracting (4)	0	
beautifulsoup4	1.6s[+] 1.9s	
Downloading	4.1MB	0.1s
Extracting (4)	0	
beautifulsoup4	1.7s[+] 2.0s	

```

Downloading          4.1MB          0.1s
Extracting   (4)          0
beautifulsoup4      1.8s[+] 2.1s
Downloading          4.1MB          0.1s
Extracting   (4)          0
beautifulsoup4      1.9s[+] 2.2s
Downloading          4.1MB          0.1s
Extracting   (4)          0 bs4
2.0s[+] 2.3s
Downloading          4.1MB          0.1s
Extracting   (4)          0 bs4
2.1s[+] 2.4s
Downloading          4.1MB          0.1s
Extracting   (2)          2 beautifulsoup4
2.2s[+] 2.5s
Downloading          4.1MB          0.1s
Extracting          4
2.3s

```

Downloading and Extracting Packages

Preparing transaction: done

Verifying transaction: done

Executing transaction: done

Collecting nbformat==4.2.0

Downloading nbformat-4.2.0-py2.py3-none-any.whl (153 kB)

153.3/153.3 kB

25.4 MB/s eta 0:00:00

Requirement already satisfied: ipython-genutils in  
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from  
nbformat==4.2.0) (0.2.0)

Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in  
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from  
nbformat==4.2.0) (4.17.3)

Requirement already satisfied: jupyter-core in  
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from  
nbformat==4.2.0) (4.12.0)

Requirement already satisfied: traitlets>=4.1 in  
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from  
nbformat==4.2.0) (5.9.0)

Requirement already satisfied: attrs>=17.4.0 in  
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from  
jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (23.1.0)

Requirement already satisfied: importlib-metadata in  
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from  
jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (4.11.4)

Requirement already satisfied: importlib-resources>=1.4.0 in  
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from  
jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (5.12.0)

```

Requirement already satisfied: pkgutil-resolve-name>=1.3.10 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (1.3.10)
Requirement already satisfied: pyrsistent!=0.17.0,!0.17.1,!0.17.2,>=0.14.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (0.19.3)
Requirement already satisfied: typing-extensions in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (4.5.0)
Requirement already satisfied: zipp>=3.1.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from importlib-
resources>=1.4.0->jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (3.15.0)
Installing collected packages: nbformat
  Attempting uninstall: nbformat
    Found existing installation: nbformat 5.8.0
    Uninstalling nbformat-5.8.0:
      Successfully uninstalled nbformat-5.8.0
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.

jupyter-server 1.24.0 requires nbformat>=5.2.0, but you have nbformat 4.2.0
which is incompatible.

nbclient 0.7.4 requires nbformat>=5.1, but you have nbformat 4.2.0 which is
incompatible.

nbconvert 7.4.0 requires nbformat>=5.1, but you have nbformat 4.2.0 which is
incompatible.

Successfully installed nbformat-4.2.0

```

```

[3]: 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.

```

[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()
```

## 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]: ticker_objct = 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.

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

```
[10]: <bound method NDFrame.head of
Low      Close \
0        0 2010-06-29    1.266667    1.666667    1.169333    1.592667
1        1 2010-06-30    1.719333    2.028000    1.553333    1.588667
2        2 2010-07-01    1.666667    1.728000    1.351333    1.464000
3        3 2010-07-02    1.533333    1.540000    1.247333    1.280000
4        4 2010-07-06    1.333333    1.333333    1.055333    1.074000
...
3367    3367 2023-11-13    215.600006    225.399994    211.610001    223.710007
3368    3368 2023-11-14    235.029999    238.139999    230.720001    237.410004
3369    3369 2023-11-15    239.289993    246.699997    236.449997    242.839996
3370    3370 2023-11-16    239.490005    240.880005    230.960007    233.589996
3371    3371 2023-11-17    232.000000    237.389999    226.539993    234.300003

      Volume  Dividends  Stock Splits
0    281494500         0         0.0
1    257806500         0         0.0
2    123282000         0         0.0
3     77097000         0         0.0
4    103003500         0         0.0
...
3367   140447600         0         0.0
3368   149771600         0         0.0
3369   150354000         0         0.0
3370   136816800         0         0.0
3371   142532800         0         0.0

[3372 rows x 9 columns]>
```

### 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`.

```
[14]: 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`.

```
[15]: soup = BeautifulSoup(html_data)
```

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`.

[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]: tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])
table = soup.find('table')
for row in table.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
```

```
[23]:
```

	Date	Revenue
0	2021	\$53,823
1	2020	\$31,536
2	2019	\$24,578
3	2018	\$21,461
4	2017	\$11,759
5	2016	\$7,000
6	2015	\$4,046
7	2014	\$3,198
8	2013	\$2,013
9	2012	\$413
10	2011	\$204
11	2010	\$117
12	2009	\$112

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

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

```
[25]:
```

	Date	Revenue
0	2021	53823
1	2020	31536
2	2019	24578
3	2018	21461
4	2017	11759
5	2016	7000
6	2015	4046
7	2014	3198

8	2013	2013
9	2012	413
10	2011	204
11	2010	117
12	2009	112

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

```
[26]: 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.

```
[27]: tesla_revenue.tail
```

```
[27]: <bound method NDFrame.tail of      Date Revenue
0    2021    53823
1    2020    31536
2    2019    24578
3    2018    21461
4    2017    11759
5    2016     7000
6    2015     4046
7    2014     3198
8    2013     2013
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`.

```
[28]: tc_obj = 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.

```
[37]: gme_data = pd.DataFrame(tc_obj.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.

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

```
[40]:
```

	level_0	index	Date	Open	High	Low	Close	\
0	0	0	2002-02-13	1.620129	1.693350	1.603296	1.691667	
1	1	1	2002-02-14	1.712707	1.716074	1.670626	1.683250	
2	2	2	2002-02-15	1.683250	1.687458	1.658001	1.674834	
3	3	3	2002-02-19	1.666417	1.666417	1.578047	1.607504	
4	4	4	2002-02-20	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`.

```
[43]: html_data = requests.get('https://cf-courses-data.s3.us.cloud-object-storage.
      ↪appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/
      ↪project/stock.html').text
```

Parse the html data using `beautiful_soup`.

```
[44]: soup = BeautifulSoup(html_data)
```

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 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

```
[46]: gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])
      table = soup.find('table')
```

```

for row in table.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["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"")
gme_revenue

```

```

[46]:      Date Revenue
0    2020     6466
1    2019     8285
2    2018     8547
3    2017     7965
4    2016     9364
5    2015     9296
6    2014     9040
7    2013     8887
8    2012     9551
9    2011     9474
10   2010     9078
11   2009     8806
12   2008     7094
13   2007     5319
14   2006     3092
15   2005     1843

```

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

```

[47]: gme_revenue.tail()

```

```

[47]:      Date Revenue
11   2009     8806
12   2008     7094
13   2007     5319
14   2006     3092
15   2005     1843

```

## 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. 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.

```

[50]: graphics = make_graph(tesla_data, tesla_revenue, 'Tesla graphics')

```



## 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.

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



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

##

© IBM Corporation 2020. All rights reserved.