

Revenue Data and Building a Dashboard

December 24, 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.

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

Requirement already satisfied: yfinance in /opt/conda/lib/python3.12/site-packages (1.0)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.12/site-packages (from yfinance) (2.3.3)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.12/site-packages (from yfinance) (2.4.0)
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) (0.0.12)
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) (3.18.3)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/lib/python3.12/site-packages (from yfinance) (4.12.3)
Requirement already satisfied: curl_cffi<0.14,>=0.7 in /opt/conda/lib/python3.12/site-packages (from yfinance) (0.13.0)
Requirement already satisfied: protobuf>=3.19.0 in /opt/conda/lib/python3.12/site-packages (from yfinance) (6.33.2)
Requirement already satisfied: websockets>=13.0 in /opt/conda/lib/python3.12/site-packages (from yfinance) (15.0.1)
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: cffi>=1.12.0 in /opt/conda/lib/python3.12/site-packages (from curl_cffi<0.14,>=0.7->yfinance) (1.17.1)
Requirement already satisfied: certifi>=2024.2.2 in /opt/conda/lib/python3.12/site-packages (from curl_cffi<0.14,>=0.7->yfinance) (2024.12.14)
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) (2025.3)
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: pycparser in /opt/conda/lib/python3.12/site-packages (from cffi>=1.12.0->curl_cffi<0.14,>=0.7->yfinance) (2.22)
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) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.12/site-packages (from beautifulsoup4->bs4) (2.5)
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: matplotlib in /opt/conda/lib/python3.12/site-
packages (3.10.8)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (2.4.0)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (12.0.0)
Requirement already satisfied: pyparsing>=3 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (3.3.1)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

```
[3]: import yfinance as yf
import pandas as pd
import requests
from bs4 import BeautifulSoup
```

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.

```
[51]: 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.

```
[52]: # The make_graph function has been modified to use Matplotlib for static graphs.  
# Earlier, it used Plotly to generate interactive dashboards, which caused issues when uploading the notebook in the MARK assignment submission.
```

```
import matplotlib.pyplot as plt  
  
def make_graph(stock_data, revenue_data, stock):  
    stock_data_specific = stock_data[stock_data.Date <= '2021-06-14']  
    revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']  
  
    fig, axes = plt.subplots(2, 1, figsize=(12, 8), sharex=True)  
  
    # Stock price  
    axes[0].plot(pd.to_datetime(stock_data_specific.Date), stock_data_specific.  
    ↪Close.astype("float"), label="Share Price", color="blue")  
    axes[0].set_ylabel("Price ($US)")  
    axes[0].set_title(f"{stock} - Historical Share Price")  
  
    # Revenue  
    axes[1].plot(pd.to_datetime(revenue_data_specific.Date),  
    ↪revenue_data_specific.Revenue.astype("float"), label="Revenue",  
    ↪color="green")  
    axes[1].set_ylabel("Revenue ($US Millions)")  
    axes[1].set_xlabel("Date")  
    axes[1].set_title(f"{stock} - Historical Revenue")  
  
    plt.tight_layout()  
    plt.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.

```
[66]: 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.

```
[144]: 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.

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

```
[68]:          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/IBMDriverSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm> Save the text of the response as a variable named `html_data`.

```
[124]: url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/  
        ↪IBMDriverSkillsNetwork-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`.

```
[125]: soup = BeautifulSoup(html_data, 'html.parser')
soup.find_all('title')
```

```
[125]: [<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.

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.

```
[126]: tesla_revenue = pd.DataFrame(columns = ['Date', '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

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

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

```
[127]: tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.
    .replace(',', '$', regex=True)
```

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

```
[128]: 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.

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

```
[91]:      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.

```
[100]: 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.

```
[145]: gme_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.

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

```
[106]:      index          Date    Open     High     Low   Close \
0         0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667
1         1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250
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  1.615920  1.662209  1.603295  1.662209

      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/IBMDriverSkillsNetwork-PY0220EN>

SkillsNetwork/labs/project/stock.html. Save the text of the response as a variable named html_data_2

```
[129]: url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/  
↳IBMDveloperSkillsNetwork-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.

```
[134]: soup = BeautifulSoup(html_data_2, 'html.parser')  
soup.find_all('title')
```

```
[134]: [<title>GameStop Revenue 2006-2020 | GME | MacroTrends</title>]
```

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

```
[137]: gme_revenue = pd.DataFrame(columns = ['Date', 'Revenue'])  
  
for table in soup.find_all('table'):   
    if('GameStop Quarterly Revenue' in table.find('th').text):  
        rows = table.find_all('tr')  
  
        for row in rows:  
            col = row.find_all('td')  
  
            if col != []:  
                date = col[0].text  
                revenue = col[1].text.replace(',', '').replace('$', '')  
  
                new_row = pd.DataFrame({'Date': [date], 'Revenue': [revenue]})  
                gme_revenue = pd.concat([gme_revenue, new_row], ignore_index = ↳True)
```

Remove the comma and dollar sign, an null or empty strings from the Revenue column.

```
[139]: gme_revenue['Revenue'] = gme_revenue['Revenue'].str.replace(',', '$', regex = True)

gme_revenue.dropna(inplace=True)

gme_revenue = gme_revenue[gme_revenue['Revenue'] != '' ]
```

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

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

```
[141]:      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
```

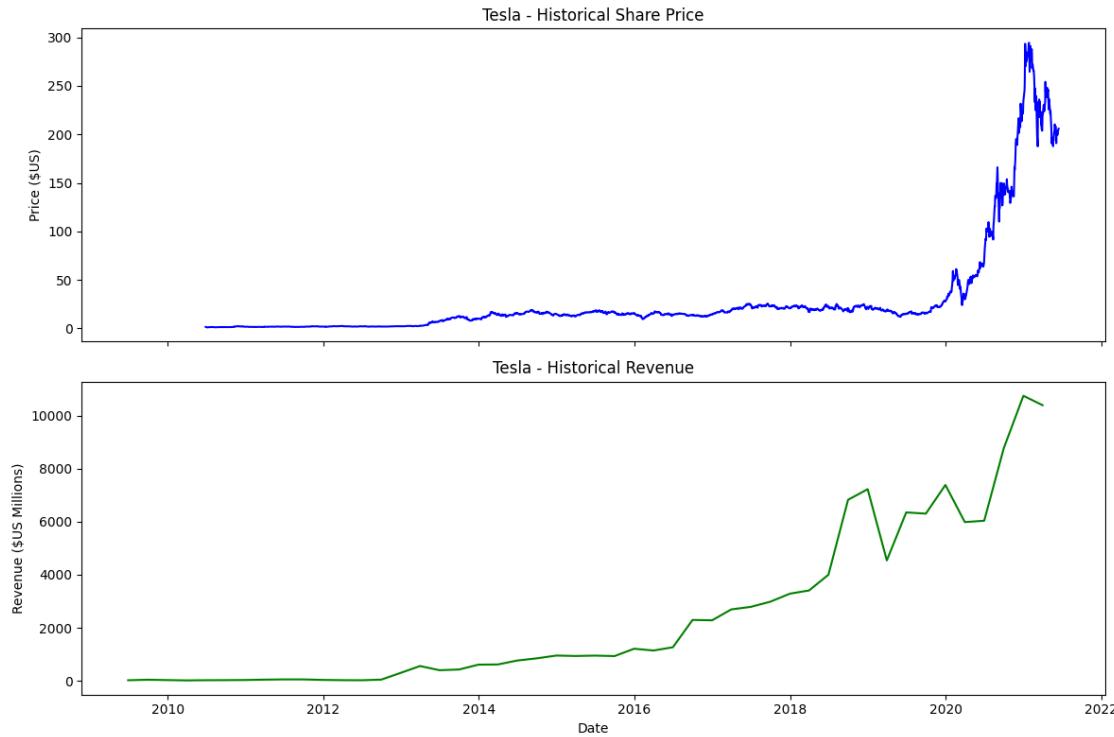
0.6 Question 5: Plot Tesla Stock Graph

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 graph.

```
[142]: make_graph(tesla_data[['Date', 'Close']], tesla_revenue, 'Tesla')
```



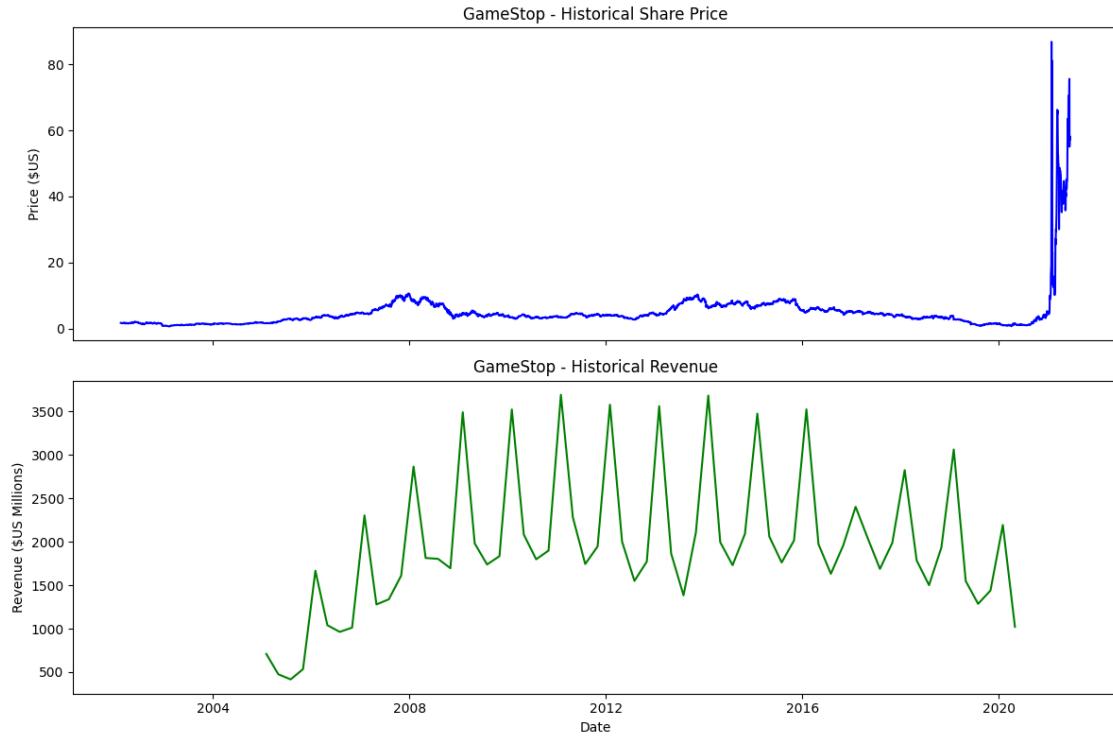
0.7 Question 6: Plot GameStop Stock Graph

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

```
[143]: make_graph(gme_data[['Date', 'Close']], gme_revenue, 'GameStop')
```



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

Copyright © 2020 IBM Corporation. All rights reserved.