

# Data Science Portfolio Projects to Rely On

Tips to organize and document a data science portfolio project

# BACKGROUND

## ACADEMIC TRAINING

B.Sc. in Physics

PhD in Medical Physics

Post-Doctoral Research in Physics



## COMPUTING BACKGROUND

- MAPLE, FORTRAN, MATLAB
- R, PYTHON, SHELL scripts

## DATA SCIENCE and PYTHON Training

### Online Courses

- UDACITY Programming Foundations with Python
- UdeMy Python for Data Science and ML Bootcamp

WeCloudData part-time Data Science with Python

WeCloudData Full time Diploma

# Documentation

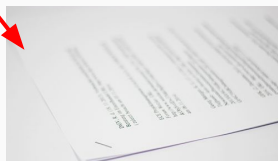
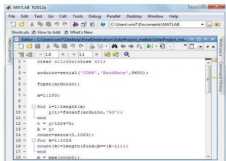
Canadian Light Source (CLS) in Saskatoon, SK

- Particle Accelerator used in Applied Science Research
- Perform Science Experiments
- Collect Data

Me → MATLAB → Analyze raw data

- Deal with headers
  - Engineering data (Skip rows)
- Regular expressions (regex)
- Perform Data Analysis
- 11 steps + Analysis based on your use-case

1 yr later Colleague → Code + docs (comments, difficulties)



## PROBLEMS:

Database changes impacted headers, Step 8. and column structure failed

Colleague: Debug, MATLAB regexp, update 1 step in docs

**Key Points:** Documentation was present, Documentation was usable



# Documentation

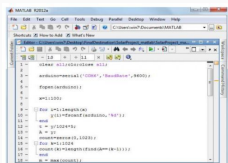
## Canadian Light Source (CLS)

- Applied Science Research Facility
- Perform Science Experiments
- Collect Data

Me → MATLAB → Analyze raw data

- Deal with headers
  - Engineering data (Skip rows)
- Regular expressions (regexp)
- Perform Data Analysis
- 11 steps + Analysis based on your use-case

1 yr later Colleague → Code + docs (comments, difficulties)



### PROBLEM:

Database changes impacted headers, Step 8. and column structure failed

Colleague: Debug, Learning MATLAB regexp, update 1 step in docs

**Key Points:** Documentation was present, Documentation was usable

## Data Science Portfolio Projects

Background: Research a topic, Identify the customer, Identify customer's needs

### Analysis Workflow

- **Extracting** insights from real-world raw data
- **Sharing** insights with a customer
- **Building solutions** that offer direct value to a customer

Parts of our portfolio project follows the workflow used in industry applications

2016: <https://www.dataquest.io/blog/build-a-data-science-portfolio/>

Shared in Repository on Github, GitLab, BitBucket, etc.



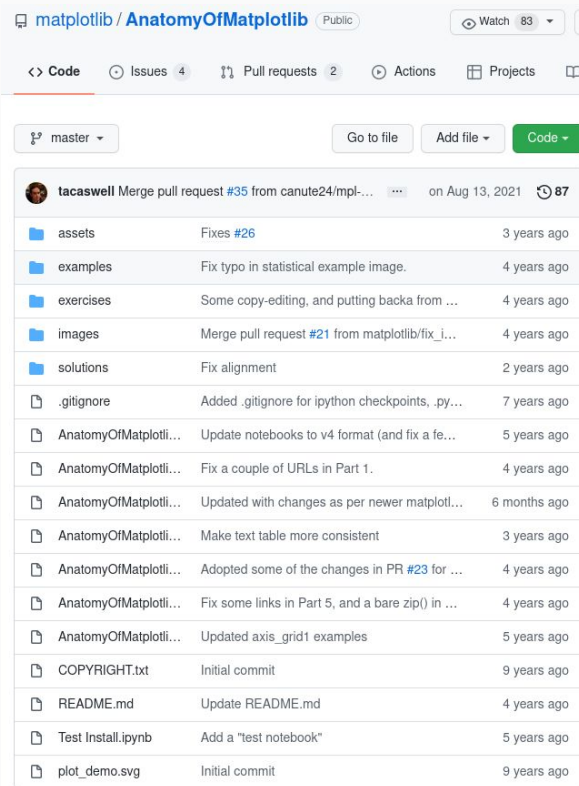
### Benefits of Sharing **Documented** Data Science Portfolio Projects

- Easy for a reader to follow
- Easy for us to follow (quick iteration)
- Clear demonstration of above workflow (sharing, extracting, building)
  - Represents our implementation of a solution to a real-world problem

# Structure of Contents in Portfolio Project

Blog Post *How to share a portfolio:*

2017: <https://www.dataquest.io/blog/how-to-share-data-science-portfolio/>



The screenshot shows the GitHub repository page for 'matplotlib / AnatomyOfMatplotlib'. At the top, there are tabs for 'Code', 'Issues' (4), 'Pull requests' (2), 'Actions', 'Projects', and a menu icon. Below the tabs, there's a 'master' branch selector and buttons for 'Go to file', 'Add file', and 'Code'. The main content area displays a list of files and their commit history:

File	Description	Time
assets	Fixes #26	3 years ago
examples	Fix typo in statistical example image.	4 years ago
exercises	Some copy-editing, and putting back from ...	4 years ago
images	Merge pull request #21 from matplotlib/fix_i...	4 years ago
solutions	Fix alignment	2 years ago
.gitignore	Added .gitignore for ipython checkpoints, .py...	7 years ago
AnatomyOfMatplotli...	Update notebooks to v4 format (and fix a fe...	5 years ago
AnatomyOfMatplotli...	Fix a couple of URLs in Part 1.	4 years ago
AnatomyOfMatplotli...	Updated with changes as per newer matplotl...	6 months ago
AnatomyOfMatplotli...	Make text table more consistent	3 years ago
AnatomyOfMatplotli...	Adopted some of the changes in PR #23 for ...	4 years ago
AnatomyOfMatplotli...	Fix some links in Part 5, and a bare zip() in ...	4 years ago
AnatomyOfMatplotli...	Updated axis_grid1 examples	5 years ago
COPYRIGHT.txt	Initial commit	9 years ago
README.md	Update README.md	4 years ago
Test Install.ipynb	Add a "test notebook"	5 years ago
plot_demo.svg	Initial commit	9 years ago

README.md

## Introduction

This tutorial is a complete re-imagining of how one should teach users the matplotlib library. Hopefully, this tutorial may serve as inspiration for future restructuring of the matplotlib documentation. Plus, I have some ideas of how to improve this tutorial.

Please fork and contribute back improvements! Feel free to use this tutorial for conferences and other opportunities for training.

The tutorial can be viewed on [nbviewer](#):

- [Part 0: Introduction To NumPy](#)
- [Part 1: Overview of Matplotlib](#)
- [Part 2: Plotting Methods](#)
- [Part 3: How To Speak MPL](#)
- [Part 4: Limits, Legends, and Layouts](#)
- [Part 5: Artists](#)
- [Part 6: mpl\\_toolkits](#)

## Installation

All you need is matplotlib (v1.5 or greater) and jupyter installed. You can use your favorite Python package installer for this:

```
conda install matplotlib jupyter
git clone https://github.com/matplotlib/AnatomyOfMatplotlib.git
cd AnatomyOfMatplotlib
jupyter notebook
```

A browser window should appear and you can verify that everything works as expected by clicking on the `Test Install.ipynb` notebook. There, you will see a "code cell" that you can execute. Run it, and you should see a very simple line plot, indicating that all is well.

Repository in the matplotlib organization: <https://github.com/matplotlib/AnatomyOfMatplotlib>

# Use a Directory Structure for a Portfolio Project

README.md

requirements.txt

01\_get\_data.ipynb

02\_clean\_data.ipynb

03\_eda.ipynb

04\_analyse\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

data/

data/raw/

data/processed/

1. README.md, requirements.txt, notebooks, extracted (custom) Python modules, data folders
2. Avoids notebooks that are too long (allows reader to stay interested in the contents)
3. Allows easier reuse of code
4. Helps keep code organized (avoids misplaced code cells)
5. Sets an appropriate context for visitor to the project (what is this project about, where to find code)

# Use a Directory Structure for a Portfolio Project

## README.md

requirements.txt

01\_get\_data.ipynb

02\_clean\_data.ipynb

03\_eda.ipynb

04\_analyse\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

data/

data/raw/

data/processed/

## README.md

1. Project title
2. Background, motivation, end-user, etc. for the the project
  - Based on reading through whitepapers, research papers, etc.
3. Add instructions for how to install Python packages and start jupyter

### Example wording:

To install all Python packages and start a Jupyter notebook, please run the following

```
$ pip install -r requirements.txt
```

```
$ jupyter notebook
```

4. How to run all the notebooks? Sequentially, based on their file names. Untitled12.ipynb



### Example wording:

Please run the notebooks in the following order

- 01\_get\_data.ipynb (to download the raw data)

- 02\_clean\_data.ipynb (to process the raw data)

- 03\_eda.ipynb (to explore the processed data)

- 04\_analyse\_data.ipynb (to perform the quantitative analysis. eg. ML)

# Use a Directory Structure for a Portfolio Project

README.md

requirements.txt

01\_get\_data.ipynb

02\_clean\_data.ipynb

03\_eda.ipynb

04\_analyse\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

data/

data/raw/

data/processed/

requirements.txt

pandas==1.35.0

plotly==5.5.0

scipy==1.7.3

To get package version numbers

- o pip: <https://pypi.org/>
- o conda: <https://anaconda.org/search>

PyPI

The screenshot shows the PyPI search results for the package 'keras'. The search bar at the top contains 'keras'. Below the search bar, it says '1,535 projects for "keras"'. A dropdown menu for 'Order by' is set to 'Relevance'. A list of results is shown, with 'keras 2.7.0' highlighted in a red box. The description for 'keras 2.7.0' is 'Deep learning for humans.' and the date is 'Nov 3, 2021'. Other results include 'rosej-keras 1.20210930' and 'keras-utils 1.0.13'.

Anaconda Search

The screenshot shows the Anaconda Search results for the package 'keras'. The search bar at the top contains 'keras'. Below the search bar, there are filters for 'Type: All', 'Access: All', and 'Platform: All'. A table of results is shown, with 'conda-forge / keras 2.7.0' highlighted in a red box. The table has columns for 'Favorites', 'Downloads', 'Package (owner / package)', and 'Platforms'. The results include 'conda-forge / keras 2.7.0', 'conda-forge / keras-preprocessing 1.1.2', and 'conda-forge / keras-applications 1.0.8'.



# Use a Directory Structure for a Portfolio Project

README.md

requirements.txt

01\_get\_data.ipynb

02\_clean\_data.ipynb

03\_eda.ipynb

04\_analyse\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

data/

data/raw/

data/processed/

**Clickable Section Headers** (HTML Hyperlinks without href attribute)

```
<a id='get-data'>
```

```
## 1. Get data
```

## Table of Contents

```
1. [Get data](#get-data)
```

```
2. ...
```

## Divider / Horizontal Rule

```
---
```

## Footers for Navigation

```
<span style="float:left;">
```

```
    <a href="./01_dont_do_this.ipynb"> >> 02_get_data.ipynb</a>
```

```
</span>
```

# Use a Directory Structure for a Portfolio Project

README.md

requirements.txt

**01\_get\_data.ipynb**

02\_clean\_data.ipynb

03\_eda.ipynb

04\_analyse\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

**data/**

**data/raw/**

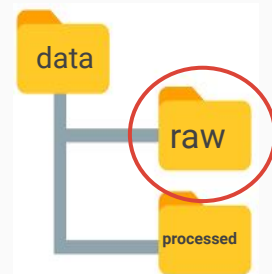
data/processed/

**01\_get\_data.ipynb**

Download the raw data

Store raw data in a sub-folder → data/raw/data\_file\_1.csv

Reduces chance of accidentally over-writing raw data with processed data



# Use a Directory Structure for a Portfolio Project

README.md

requirements.txt

01\_get\_data.ipynb

**02\_clean\_data.ipynb**

03\_eda.ipynb

04\_analyse\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

**data/**

data/raw/

**data/processed/**

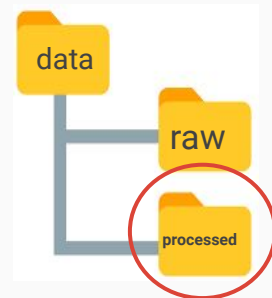
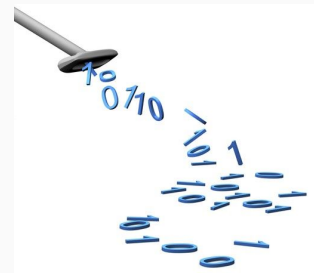
**02\_clean\_data.ipynb**

Process the raw data

- Handle missing values
- Filter the data
- Drop invalid rows
- ...

Might need to combine with supplementary data before filtering

Store processed data in a sub-folder → data/processed/cleaned\_data.csv



# Use a Directory Structure for a Portfolio Project

README.md

requirements.txt

01\_get\_data.ipynb

02\_clean\_data.ipynb

**03\_eda.ipynb**

04\_analyse\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

data/

data/raw/

data/processed/

**03\_eda.ipynb**

Explore the processed data

- Descriptive Statistics
- Visualisation



# Use a Directory Structure for a Portfolio Project

README.md

requirements.txt

01\_get\_data.ipynb

02\_clean\_data.ipynb

03\_edu.ipynb

**04\_analyse\_data.ipynb**

src/

src/\_\_init\_\_.py

src/load\_data.py

data/

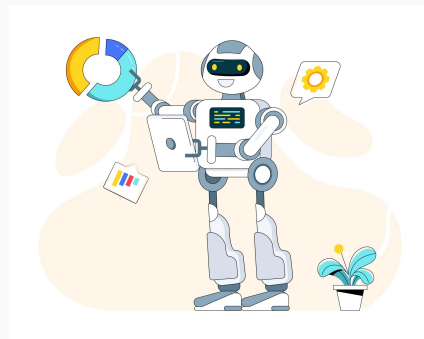
data/raw/

data/processed/

## 04\_analyse\_data.ipynb

Perform the quantitative data analysis

- Linear Regression
- Supervised Machine Learning
- Unsupervised Machine Learning
- Deep Learning
- ...



# Modularize Python Code into src/

Avoids Long / Distracting Notebooks

Put common code into functions

Extract functions from notebook into custom Python modules

`__init__.py`: required to make Python treat the directories as containing packages to be imported: `import numpy as np, from src.load_data import ...`

Consider sub-folders for a larger code base

Can use multiple modules per folder

- eg. for visualization with `matplotlib` and `plotly`, may want
  - `visualize_matplotlib.py`
  - `visualize_plotly.py`

## OPTION 1

01\_get\_data.ipynb

src/

src/`__init__.py`

src/`load_data.py`

src/`create_features.py`

src/`visualize.py`

## OPTION 2

01\_get\_data.ipynb

src/

src/`__init__.py`

src/`data/`

src/`data/``__init__.py`

src/`data/load_data.py`

src/`features/`

src/`features/``__init__.py`

src/`features/create_features.py`

src/`visualization/`

src/`visualization/``__init__.py`

src/`visualization/visualize.py`

# Modularize Python Code into src/

## BEFORE

### Contents of 01\_get\_data.ipynb

```
import pandas as pd

df = pd.read_csv(url, nrows=200000, dtype={...})

df['date'] = pd.to_datetime(...)

df = df.drop(columns=['Unnamed: 7', 'Unnamed: 18'])

df = df.drop_duplicates(...)
```

## OPTION 1

01\_get\_data.ipynb

src/

src/\_\_init\_\_.py

src/load\_data.py

src/create\_features.py

src/visualize.py

# Modularize Python Code into src/

**BEFORE**

Contents of 01\_get\_data.ipynb

```
import pandas as pd

df = pd.read_csv(url, nrows=200000, dtype={...})
df['date'] = pd.to_datetime(...)

df = df.drop(columns=['Unnamed: 7', 'Unnamed: 18'])

df = df.drop_duplicates(...)
```

**OPTION 1**

```
01_get_data.ipynb
src/
src/__init__.py
src/load_data.py
src/create_features.py
src/visualize.py
```

**Define Function**

**AFTER**

Contents of src/load\_data.py

```
import pandas as pd

def get_data(url, dtypes_dict, nrows, drop_cols):
    df = pd.read_csv(url, nrows=nrows, dtype=dtypes_dict)
    df['date'] = pd.to_datetime(...)
    df = df.drop(columns=drop_cols)
    df = df.drop_duplicates(...)

    return df
```

Contents of 01\_get\_data.ipynb

```
from src.load_data import get_data

df = get_data(url, {...}, 200000, ['Unnamed: 7', 'Unnamed: 18'])
```

**Call Function in notebook**



# **Documentation in a Data Science Notebook**

# Example Project

1. Analysis of Capital Bikeshare data
  - Washington, DC area (US)
2. Data is posted monthly
  - <https://s3.amazonaws.com/capitalbikeshare-data/index.html>
3. Steps in the workflow
  - Download Data
    - Use python to retrieve data
  - Process
    - Data Cleaning
    - Data Filtering
  - Visualise
    - EDA
  - Analyse
    - Feature Engineering
    - Machine Learning / Statistical Analysis



Name	Date Modified	Size	Type
2010-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:33:31 pm	2.41 MB	ZIP file
2011-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:45:30 pm	25.33 MB	ZIP file
2012-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:55:27 pm	43.46 MB	ZIP file
2013-capitalbikeshare-tripdata.zip	Mar 15th 2018, 07:16:26 pm	56.48 MB	ZIP file
2014-capitalbikeshare-tripdata.zip	Mar 15th 2018, 07:29:51 pm	66.50 MB	ZIP file
2015-capitalbikeshare-tripdata.zip	Mar 15th 2018, 10:04:26 pm	73.59 MB	ZIP file
2016-capitalbikeshare-tripdata.zip	Mar 15th 2018, 10:27:02 pm	78.23 MB	ZIP file
2017-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:25:30 pm	89.58 MB	ZIP file
201801-capitalbikeshare-tripdata.zip	Apr 30th 2018, 12:01:09 pm	3.87 MB	ZIP file
201802-capitalbikeshare-tripdata.zip	May 11th 2018, 04:47:35 pm	4.18 MB	ZIP file

# 01\_dont\_do\_this.ipynb

Download the ChromeDriver 97.0.4692.71: <https://chromedriver.chromium.org/downloads>. This requires Google chrome version 97.0.4692.71.

Download the bikeshare data from here: <https://s3.amazonaws.com/capitalbikeshare-data/index.html>

Install Python libraries: selenium, pandas

```
In [1]: # !pip3 install selenium pandas
```

```
In [2]: import os
```

```
In [3]: from selenium.webdriver.chrome.options import Options
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
```

```
In [4]: # Path to the Chrome webdriver on local system
user_name = os.getenv("USERNAME")
webdriver_path = f"/home/{user_name}/chromedriver_linux64/chromedriver"

# Create ChromeDriver service object
driver_service_object = Service(webdriver_path)
options = Options()
```

```
In [5]: driver = webdriver.Chrome(service=driver_service_object, options=options)
```

## Selenium Webdriver

Download all the metadata

```
In [6]: driver.get("https://s3.amazonaws.com/capitalbikeshare-data/index.html")
```

## Web Scraping

```
In [7]: from selenium.webdriver.common.by import By
```

```
In [8]: container = driver.find_element(By.XPATH, '//*[@class="container"]')
table_id = container.find_element(By.XPATH, "//*[@class='hide-while-load']")
header = container.find_element(By.XPATH, "//*[@class='hide-while-load']")
headers = [h.text for h in header.find_elements(By.CSS_SELECTOR, "th")]
```

Get table with list of files

Name	Date Modified	Size	Type
2010-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:33:31 pm	2.41 MB	ZIP file
2011-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:45:30 pm	25.33 MB	ZIP file
2012-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:55:27 pm	43.46 MB	ZIP file
2013-capitalbikeshare-tripdata.zip	Mar 15th 2018, 07:16:26 pm	56.48 MB	ZIP file
2014-capitalbikeshare-tripdata.zip	Mar 15th 2018, 07:29:51 pm	66.50 MB	ZIP file
2015-capitalbikeshare-tripdata.zip	Mar 15th 2018, 10:04:26 pm	73.59 MB	ZIP file
2016-capitalbikeshare-tripdata.zip	Mar 15th 2018, 10:27:02 pm	78.23 MB	ZIP file
2017-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:25:30 pm	89.58 MB	ZIP file
201801-capitalbikeshare-tripdata.zip	Apr 30th 2018, 12:01:09 pm	3.87 MB	ZIP file
201802-capitalbikeshare-tripdata.zip	May 11th 2018, 04:47:35 pm	4.18 MB	ZIP file

# 01\_dont\_do\_this.ipynb

```
In [9]: import pandas as pd
```

```
In [10]: # Scrape metadata
mylists = []
for row in table_id.find_elements(By.CSS_SELECTOR, "tr"):
    mylist = []
    zip_file_urls = []
    col_idx = 0
    for cell in row.find_elements(By.TAG_NAME, "td"):
        text = cell.text
        if col_idx == 0:
            data_zip_url = cell.find_element(By.CSS_SELECTOR, "a").get_at
            zip_file_urls.append(data_zip_url)
            # Append contents of single row to empty list
            mylist.append(text)
            col_idx += 1
    df_single_row = pd.DataFrame.from_records([h: r for h, r in zip(head
df_single_row["zip_file_url"] = zip_file_urls
mylists.append(df_single_row)
```

Get table values

## Data Processing

Combine the data

Create DataFrame from scraped values

```
In [11]: df = pd.concat(mylists, ignore_index=True)
```

```
In [12]: df.head(3)
```

Out[12]:

	Name	Date Modified	Size	Type	zip_file_url
0	2010-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:33:31 pm	2.41 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...
1	2011-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:45:30 pm	25.33 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...
2	2012-capitalbikeshare-tripdata.zip	Mar 15th 2018, 06:55:27 pm	43.46 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...

Get data from 2021

Filter data to only get data from 2021

```
In [13]: df = df[df["Name"].str[:4] == "2021"].copy()
```

```
df["date_modified_eod"] = pd.to_datetime(df["Date Modified"])
```

```
In [ ]:
```

```
In [14]: df.head(3)
```

Out[14]:

	Name	Date Modified	Size	Type	zip_file_url	date_modified_eod
44	202101-capitalbikeshare-tripdata.zip	Feb 4th 2021, 04:55:29 pm	3.61 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...	2021-02-04 16:55:29
45	202102-capitalbikeshare-tripdata.zip	Mar 9th 2021, 07:07:41 pm	2.78 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...	2021-03-09 19:07:41
46	202103-capitalbikeshare-tripdata.zip	Apr 8th 2021, 10:31:40 am	5.88 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...	2021-04-08 10:31:40

```
In [15]: driver.quit()
```

```
In [ ]:
```

Download the data

Download data from 2021

```
In [18]: from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile
```

```
In [19]: %%time
for _, row in df.iterrows():
    zipurl = row["zip_file_url"]
    # Extract without saving
    with urlopen(zipurl) as zipresp:
        with ZipFile(BytesIO(zipresp.read())) as zfile:
            zfile.extractall(f'data/raw')
```

CPU times: user 1.87 s, sys: 541 ms, total: 2.41 s  
Wall time: 11.3 s

# Problems with 01\_dont\_do\_this.ipynb

Download the ChromeDriver 97.0.4692.71: <https://chromedriver.chromium.org/downloads>. This requires Google chrome version 97.0.4692.71.

Download the bikeshare data from here: <https://s3.amazonaws.com/capitalbikeshare-data/index.html>

Install Python libraries: selenium, pandas

```
In [1]: # !pip3 install selenium pandas
```

```
In [2]: import os
```

```
In [3]: from selenium.webdriver.chrome.options import Options
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
```

```
In [4]: # Path to the Chrome webdriver on local system
user_name = os.getenv("USERNAME")
webdriver_path = f"/home/{user_name}/chromedriver_linux64/chromedriver"

# Create ChromeDriver service object
driver_service_object = Service(webdriver_path)
options = Options()
```

```
In [5]: driver = webdriver.Chrome(service=driver_service_object, options=options)
```

## Selenium Webdriver

Download all the metadata

```
In [6]: driver.get("https://s3.amazonaws.com/capitalbikeshare-data/index.html")
```

## Web Scraping

```
In [7]: from selenium.webdriver.common.by import By
```

```
In [8]: container = driver.find_element(By.XPATH, '//*[@class="container"]')
table_id = container.find_element(By.XPATH, "//*[@class='hide-while-loa")
header = container.find_element(By.XPATH, "//*[@class='hide-while-loa")
headers = [h.text for h in header.find_elements(By.CSS_SELECTOR, "th")]
```

1. No table of contents, No Intro section, No Title
  - a. Notebook requirements can be included in Intro section
2. Python requirements should be in requirements.txt
3. import statements
  - a. Scattered across notebook
  - b. Not sorted (<https://www.python.org/dev/peps/pep-0008/#imports>)
    - i. Python standard libraries
    - ii. Third party libraries
    - iii. Our custom modules from src/

# Problems with 01\_dont\_do\_this.ipynb

In [ ]:

In [14]: df.head(3)

Out[14]:

	Name	Date Modified	Size	Type	zip_file_url	date_modified_eod
44	202101-capitalbikeshare-tripdata.zip	Feb 4th 2021, 04:55:29 pm	3.61 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...	2021-02-04 16:55:29
45	202102-capitalbikeshare-tripdata.zip	Mar 9th 2021, 07:07:41 pm	2.78 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...	2021-03-09 19:07:41
46	202103-capitalbikeshare-tripdata.zip	Apr 8th 2021, 10:31:40 am	5.88 MB	ZIP file	https://s3.amazonaws.com/capitalbikeshare-data...	2021-04-08 10:31:40

In [15]: driver.quit()

In [ ]:

Download the data

In [18]: 

```
from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile
```

In [19]: 

```
%%time
for _, row in df.iterrows():
    zipurl = row["zip_file_url"]
    # Extract without saving
    with urlopen(zipurl) as zipresp:
        with ZipFile(BytesIO(zipresp.read())) as zfile:
            zfile.extractall(f'data/raw')
```

CPU times: user 1.87 s, sys: 541 ms, total: 2.41 s  
Wall time: 11.3 s

## 4. Blank cells are not necessary

- Distracting to reader
- Reader doesn't know if author meant to add code in these cells

## 5. Long-running cells with a loop

- Add a print statement to see output in each iteration



# Process Data - Document Future Changes to Data

## Accepted Methods of payment

- How to handle changes?
- 2 unique values becomes 1 unique value
- Keep / Drop all stations where changes have been made?
- What have others in this field done? Refer to our background research.
- **Add Documentation to Notebook...**

**Now:** ["KEY", "CREDITCARD"]

**Future(?):** ["KEY", "CREDITCARD"]

**Future(?):** ["CREDITCARD"]



# Analysis (ML) - Document Data Availability / Client Constraints

1. Use-Case: Predict number of bikeshare departures per station to rebalance a station

## 2. Client Requirements

- We can read about these in whitepapers, Journals, blog posts, etc.
- Predict 7 days of bikeshare trips departing / arriving per station
- Client will use predictions to re-balance bikes at stations
  - Need to deliver predictions ahead of time (client can plan staffing, inventory, etc.)

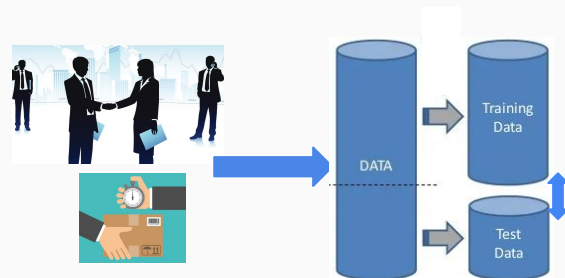


## 3. Constraints: When will data be available to use?

- 2021-12-01 - 2021-12-31 data is posted on 2022-01-06; if client needs ML predictions on 2022-01-04?
- If *latest* data is not uploaded in time, be ready to use older data (ending on 2021-11-30)
- ML Model performance might be worse

## 4. How to choose validation and test splits?

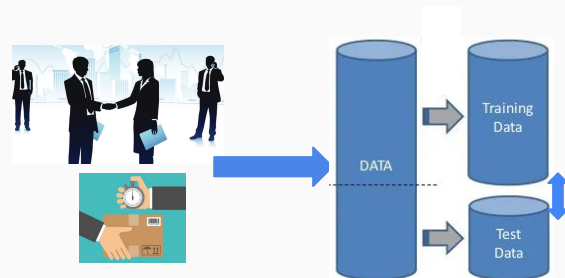
- Test split to evaluate ML model's performance, Validation split to compare models, hyperparameters
- Validation and test split must be chosen following
  - Client Requirements, Constraints





# Analysis (ML) - **Document** Data Availability / Client Constraints

1. Use-Case: Predict number of bikeshare departures per station to rebalance a station
2. **Client Requirements (README.md)**
  - We can read about these in whitepapers, Journals, blog posts, etc.
  - Predict 7 days of bikeshare trips departing from each station
  - Client will use predictions to re-balance bikes at stations
    - Need to deliver predictions ahead of time (client can plan staffing, inventory, etc.)
3. **Constraints:** When will data be available to use? **(README.md, Notebook)**
  - 2021-12-01 - 2021-12-31 data is posted on 2022-01-06; if client needs ML predictions on 2022-01-04?
  - If *latest* data is not uploaded in time, be ready to use older data (ending on 2021-11-30)
  - ML Model performance might be worse
4. How to choose validation and test splits? **(Notebook)**
  - Test split to evaluate ML model's performance, Validation split to compare models, hyperparameters
  - Validation and test split must be chosen following
    - Client Requirements, Constraints



# Additional Helpful Links

## 1. Github Repository Used here

- <https://github.com/elsdes3/portfolio-documentation-tips>

## 2. Jupyter notebook utilities

- TOC - manually:
  - i. <https://moonbooks.org/Articles/How-to-create-a-table-of-contents-in-a-jupyter-notebook/>
  - ii. [https://sebastianraschka.com/Articles/2014\\_ipython\\_internal\\_links.html](https://sebastianraschka.com/Articles/2014_ipython_internal_links.html)
- TOC - Jupyter extension:  
<https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions/toc2/README.html>

## 3. Other Files to include in a Portfolio Project

- .gitignore: <https://www.toptal.com/developers/gitignore> (enter Python in search box)
- papermill: <https://papermill.readthedocs.io/en/latest/>
- Open Source LICENSE: <https://towardsdatascience.com/a-data-scientists-guide-to-open-source-licensing-c70d5fe4207>

# THANK YOU

## Q & A, DISCUSSION



<https://www.linkedin.com/in/elstand/>



<https://github.com/elsdes3>



<https://twitter.com/elsdes3>

