itida هيئة تنمية صناعة تكنولوجيا المعلومات | egyptfwd initiative

UDACITY

fwd initiative

Project-Data Wrangling and analysis
Walk-through

Agenda

Data wrangling

Storing & Reporting

1. **Gather** (Different Sources)
2. **Assess** (Quality & Tidiness)
3. **Clean** (Define, Code & Test)
4. **Store**
5. **Analyze & Visualize & Report**

Using Python Libraries

itida
هيئة تنمية صناعة تكنولوجيا المعلومات

Key Points

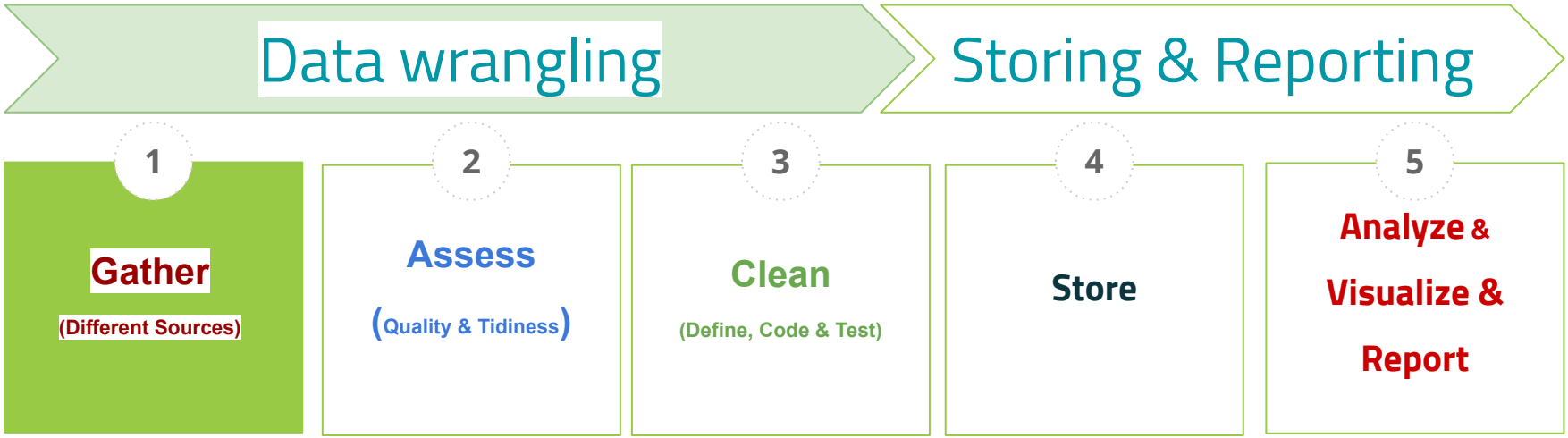
Gathering:

- You only want **original ratings (no retweets)** that **have images**. Though there are 5000+ tweets in the dataset, *not all are dog ratings* and some are *retweets*.
- You **do *not* need** to gather the tweets **beyond August 1st, 2017.** You can, but note that you **won't be able to gather the image predictions** for these tweets since you don't have access to the algorithm used.

Assessing & Cleaning

- Assessing and cleaning the entire dataset completely would require a lot of time, and is not necessary to practice and demonstrate your skills in data wrangling. Therefore, the requirements of this project are only to assess and clean ***at least 8 quality issues*** and ***at least 2 tidiness issues*** in this dataset.
- Cleaning includes ***merging individual pieces of data*** according to the rules of ***tidy data***.
- The fact that the rating numerators are greater than the denominators does not need to be cleaned. This **unique rating system** is a big part of the popularity of WeRateDogs.

Agenda



itida

Data Gathering

Inputs(Data sources):

1. Enhanced Twitter Archive (.csv)
2. Additional Data via the Twitter API
3. Image Predictions File

`wrangle_act.ipynb`

Outputs (DataFrames):

1. archive_df
2. Tweet_json.txt / api_df
3. Image_predictions_df & image-predictions.tsv

(Rubric excerpt) Data is successfully gathered:

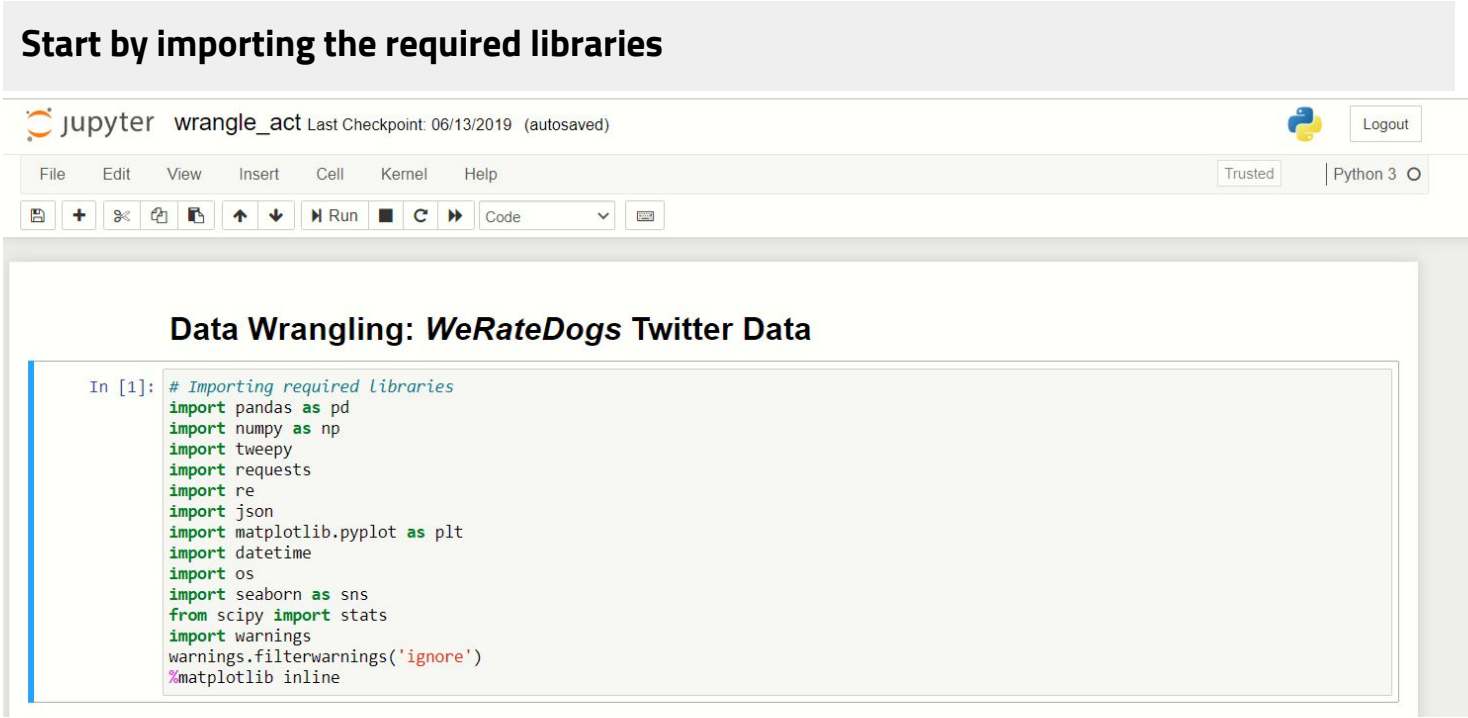
- From at least the three *(3) different sources* on the Project Details page.
- In at least the three *(3) different file formats* on the Project Details page.

Each piece of data is imported into a separate pandas DataFrame at first.

itida

Data Gathering

Start by importing the required libraries



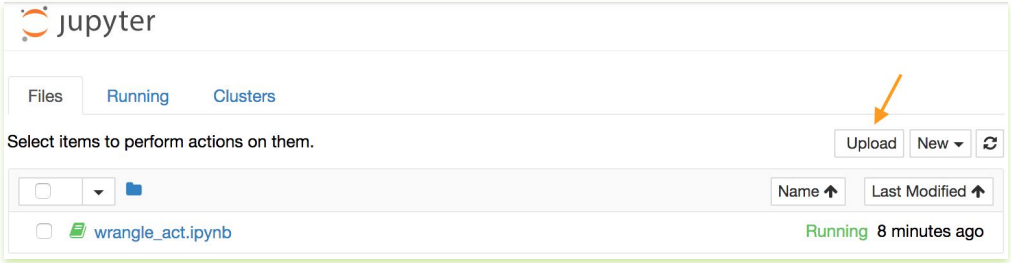
Data Gathering (1)

Inputs(Data sources):
Enhanced Twitter Archive
(.csv)

`wrangle_act.ipynb`

Outputs (DataFrames):
archive_df
(Using pandas function)

1. Download the file **manually** by clicking the following link: **twitter_archive_enhanced.csv** then read into a dataframe with an appropriate name using the pandas library.
2. Consider downloading on the same directory if working outside the project workspace
3. Working on the classroom workspace, you can see the files by clicking on the jupyter sign on the upper left corner of the workspace and upload any file after download on your machine.



Data Gathering (2)

Inputs(Data sources):

Image Predictions File

`wrangle_act.ipynb`

Outputs (DataFrames):

Image-predictions.tsv &
Image_predictions_df

Outlined Steps:

1. The tweet image predictions, i.e., what breed of dog (or other object, animal, etc.) is present in each tweet according to a neural network.
2. This file (**image_predictions.tsv**) is hosted on Udacity's servers and should be **downloaded programmatically** using the **Requests** library and the following URL:
   https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv

itida
هيئة تنمية صناعة تكنولوجيا المعلومات

Data Gathering (2)

Inputs(Data sources):

Image Predictions File

`wrangle_act.ipynb`

Outputs (DataFrames):

Image-predictions.tsv &
Image_predictions_df

Tips & Tricks on Downloading:

```
# Downloading and saving the image prediction data using Requests
url =
'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions
.tsv'
```

`file_name =` Extract the **file name from** the url string (**url**)

`response =` Get your **response** object by applying the requests package

```
if not os.path.isfile(file_name):
    with open(file_name, 'wb') as f:
        f.write(use the appropriate response attribute)
```

itida

Data Gathering (3)

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):
Tweet_json.txt / api_df

Outlined Steps:

1. Each tweet's **retweet count** and **favorite** ("like") **count** at minimum, and any **additional** data you find interesting.
2. Using the **tweet IDs** in the WeRateDogs Twitter **archive**, query the **Twitter API for each tweet's JSON data** using Python's Tweepy library and store each tweet's entire set of JSON data in a file called **tweet_json.txt** file. Each ***tweet's JSON data should be written to its own line***.
3. Then read this **.txt** file line by line into a **pandas DataFrame** with (at minimum) **tweet ID, retweet count**, and **favorite count**.
4. ***Note: do not include your Twitter API keys, secrets, and tokens in your project submission.***
5. Refer to concept 4 under the project in the classroom to set up a developer account

Data Gathering (3)

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

Tweet_json.txt / api_df

Tips & Tricks on API Querying:

```
consumer_key = '**************************'
consumer_secret = '**************************'
access_token = '**************************'
access_secret = '**************************'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

itida

Data Gathering (3)

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

Tweet_json.txt

Tips & Tricks on API Querying:

1. Experimenting to extract one tweet's id information after creating an API object.

```
exp_tweet = api.get_status(archive.tweet_id[1000], tweet_mode = 'extended')
content = exp_tweet._json
print(Content)
```

2. Checking the `keys` of the test tweet through `content.keys()`
3. Then Getting the `retweet_count` and `favorite_count` for the test tweet. There are two ways to do that:
   a. `exp_tweet.retweet_count`, - `exp_tweet.id`, - `exp_tweet.favorite_count`
   b. `content['retweet_count']` , `content['id']`, `content['favorite_count']`
   c. `Content['user']['followers_count']`

https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview/tweet-object

itida

Data Gathering (3)

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

Tweet_json.txt

Tips & Tricks on API Querying:

1. Creating the `'tweet_json.txt'` that contains Each tweet's JSON data in its own line. In this step the following code will be time saving when running the whole notebook at the end. Moreover well preserve your project results to the time when you queried the twitter API.

```
if not os.path.isfile('tweet_json.txt'):
```

2. After that we can create the file and write on it with the efficient `with` context manager as we did before.

```
with open ('tweet_json.txt', 'w') as file:
```

3. Using the **tweet_ids** from the `archive_df` to get the data for each tweet and write it in its own line in the file created in the previous step. .....**PAUSE AND THINK**....**there may be some deleted tweets for which no status would be found.**

itida

Data Gathering (3)

Tips & Tricks on API Querying:

1. To do this, we can loop over the `archive_df['tweet_id']` and in each round of the loop we want to:
   a. `try` to get the tweet JSON data and assign it to a variable name of your choice (`'tweet'` , `'status'`, etc…)
   b. `dump` the content of the let's say `tweet._json` in the created text file.
   c. `write` a new line character (`'\n'`) in the file after dumping each `tweet._json` file
2. But there may be some deleted tweets for which no status would be found.
   a. Here comes the role of an `except` statement to capture the tweet_ids for which there is no status.
   b. You can actually append such not found `tweet_ids` to an empty list to know later on how many tweets in your archive data set was not found and make sure that the number is acceptable.

Data Gathering (3)

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

Tweet_json.txt

Tips & Tricks on API Querying:

```
errors = []
if not os.path.isfile('tweet_json.txt'):
    # create the file and write on it
    with open ('tweet_json.txt', 'wb') as file:
        for tweet_id in archive['tweet_id']:
            try:
                status = api.get_status(tweet_id, wait_on_rate_limit=True,  wait_on_rate_limit_notify=True, tweet_mode =
'extended')
                json.dump(status._json, file)
                file.write('\n')
            except Exception as e:
                print("Error on tweet id {}".format(tweet_id) + ";" + str(e))
                errors.append(tweet_id)
```

itida

Data Gathering (3)

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

`api_df`

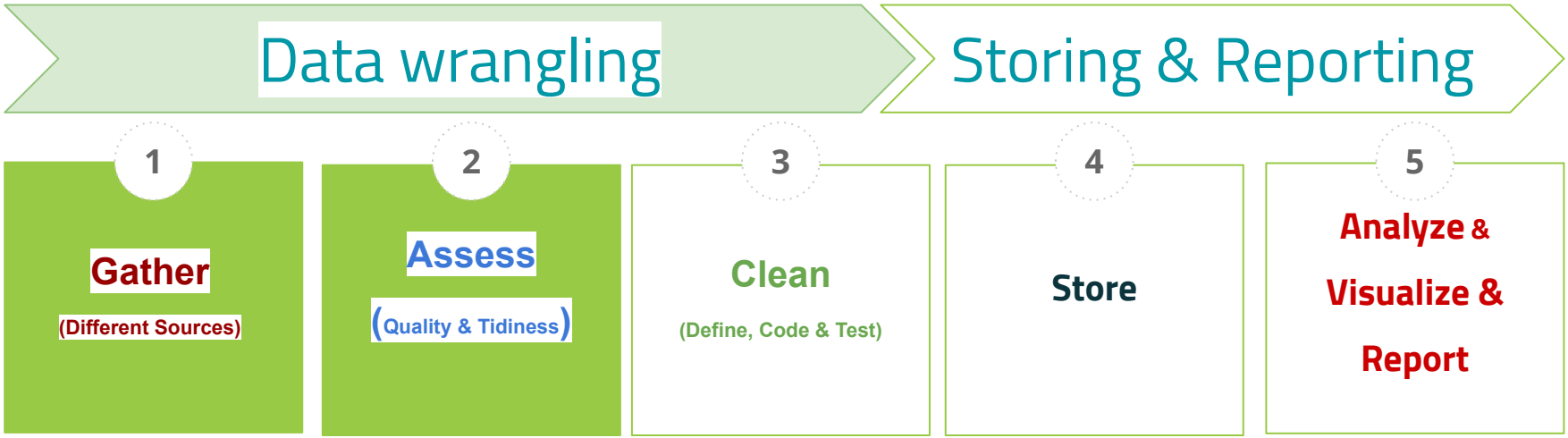
Tips & Tricks on API Querying:

Then read this **.txt** file line by line to create a **pandas DataFrame** with (at minimum) **tweet ID, retweet count**, and **favorite count**.

1. Keep in mind that we can read the file **line by line**.
2. Each line carries a tweet's data that needs to be converted to **python dict object**.
   https://www.geeksforgeeks.org/python-convert-string-dictionary-to-dictionary/
3. Each tweet's data (dict) can then be used to build a list of dicts.
4. From each tweet's data we can access the values of of the dict keys by using square brackets like what we have done when exploring the test tweet.
5. **`pd.DataFrame()`** function can take a list of dictionaries to build a dataframe which can then be whittled down.
6. Build your **`api_df`** that contains at least **`retweet_count`** and **`favorite_count`**

itida

Agenda



itida

Data Assessing "That's where the inspection of our collected data sets from both the ***Quality*** and ***Tidiness*** perspectives will be conducted."

Inputs (DataFrames):

1. archive_df
2. api_df
3. Image_predictions_df

`wrangle_act.ipynb`

Outputs (Assessment Summary):

1. At least eight (8) data quality issues
2. At least two (2) tidiness issues

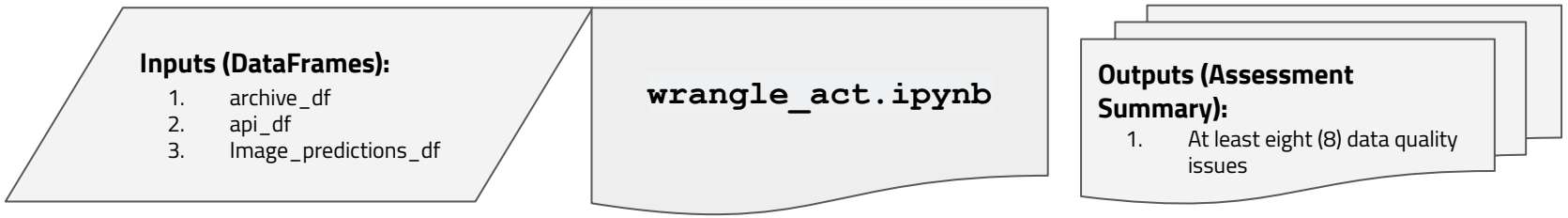
Two types of assessment are used:(Rubric excerpt)

- Visual assessment: each piece of gathered data is displayed in the Jupyter Notebook for visual assessment purposes. Once displayed, data can additionally be assessed in an external application (e.g. Excel, text editor).
- Programmatic assessment: pandas' functions and/or methods are used to assess the data.

At least eight (8) data quality issues and two (2) tidiness issues are detected, and include the issues to clean to satisfy the Project Motivation. Each issue is documented in one to a few sentences each.

itida

Data Assessing

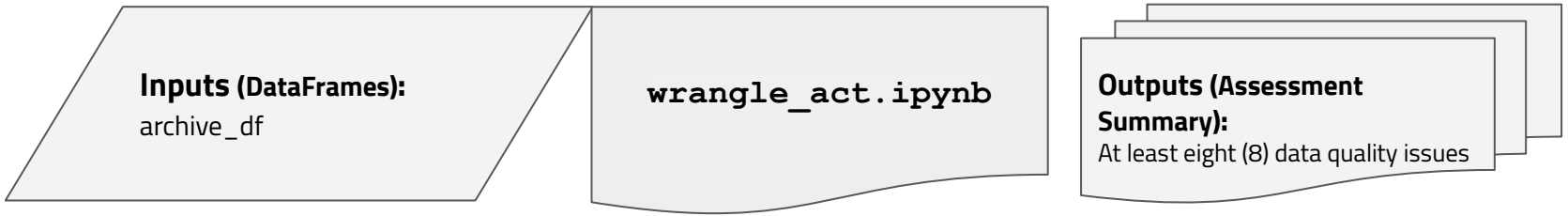


Outlined Steps:

Data ***quality dimensions*** help guide the thought process while assessing as well as cleaning efforts:

1. **Completeness:** do we have all of the records that we should? Do we have missing records or not? Are there specific rows, columns, or cells missing?
2. **Validity:** we have the records, but they're not valid, i.e., *they don't conform to a defined schema*. A schema is a defined set of rules for data. These rules can be real-world constraints (e.g. negative height is impossible) and table-specific constraints (e.g. unique key constraints in tables).
3. **Accuracy:** inaccurate data is wrong data that is valid. It adheres to the defined schema, but it is still incorrect. Example: a patient's weight that is 5 lbs too heavy because the scale was faulty.
4. **Consistency:** inconsistent data is both valid and accurate, but there are multiple correct ways of referring to the same thing. Consistency, i.e., a standard format, in columns that represent the same data across tables and/or within tables is desired.

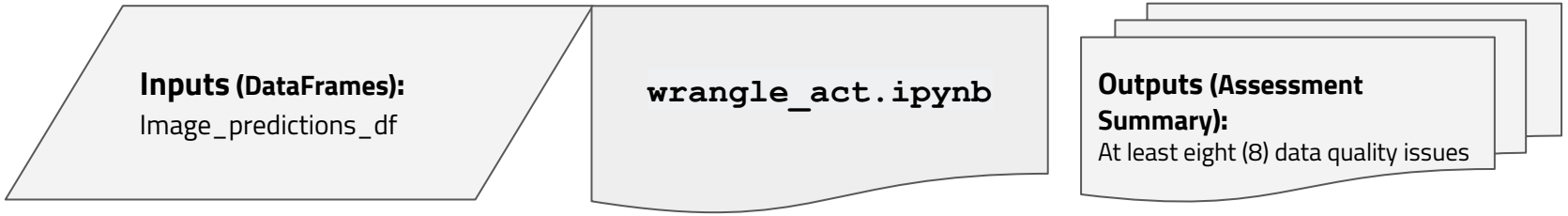
Data Assessing (1)



The Twitter `archive_df`:

1. Investigate visually and programmatically using what we have learned (`.info()`, `.columns`, `.shape`, etc..)
2. Delve deeper into the different variable, for example:
    a. Looking into the pets' **classification** of: **doggo**, **floofer**, **pupper**, **puppo**; Are all or most of dogs classified?, Is the classification correct and mutually exclusive?
    b. Checking the **name column** value counts; do you believe that the name was correctly extracted from the tweet's text? What about the weird values, the none values? Do theses values have a name in the text or they don't? How the names was captured from your observation?
    c. Checking the **ratings values**, their count, mean, five number summary.
3. As per the project requirements; only original ratings (**no retweets**) that **have images** should be included

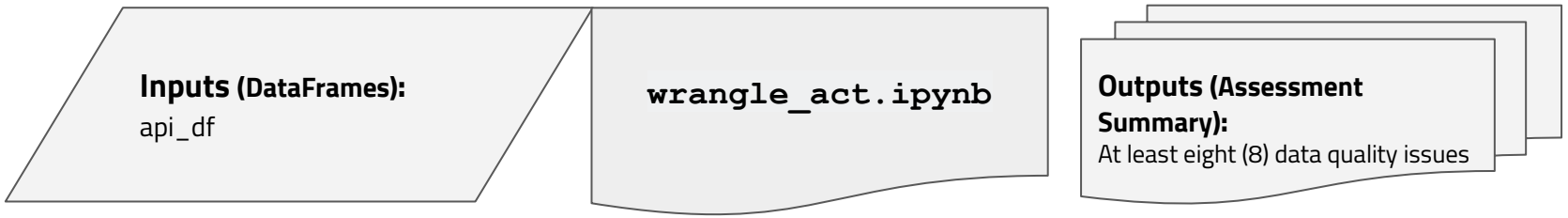
Data Assessing (2)



The Twitter `Image_predictions_df`:

1. `p1_dog` is whether or not the #1 prediction is a breed of dog (boolean), and so is `p2_dog` and `p3_dog`. Consequently we can check for false values and what are the predictions for those entries.
2. What is the most predicted breed for each prediction level.
3. Can we examine the confidence of each prediction (`'p1_conf'`, `'p2_conf'`, `'p3_conf'`) statistically.

Data Assessing (3)



The Twitter `api_df`:

1. Can we examine the `'retweet_count'`, `'favorite_count'` and any other chosen data like `'followers_count'` statistically.

Data Assessing

Outputs (Assessment Summary):

Quality aspects:

a. `archive_df`
   i. **Data types(consistency issues)**: Look at date and time, ids, representations of null values as string “none”, and look at the columns that will not be of use in your analysis.
   ii. **completeness issues**:
      1. ask yourself if any of the missing date can be found but it’s missing due to poor manipulation, for example; the name variable.
      2. Also the tweets with no images as you can notice a discrepancy in the number of tweets between the `archive_df` dataset and the `image_prediction_df.`
      3. Some tweets are actually retweets and replies not original tweets that have to be deleted as per the data wrangling scope mandated by the project specification. (Note: those tweets should be removed from the three tables in hand)
   iii. **Accuracy issues:**
      1. Erroneous
      2. Incorrect and weird values

Data Assessing

Outputs (Assessment Summary):

Tidiness aspects:

a. `archive_df`
   i. values are column names
b. `image_predictions`
c. `Api_df`

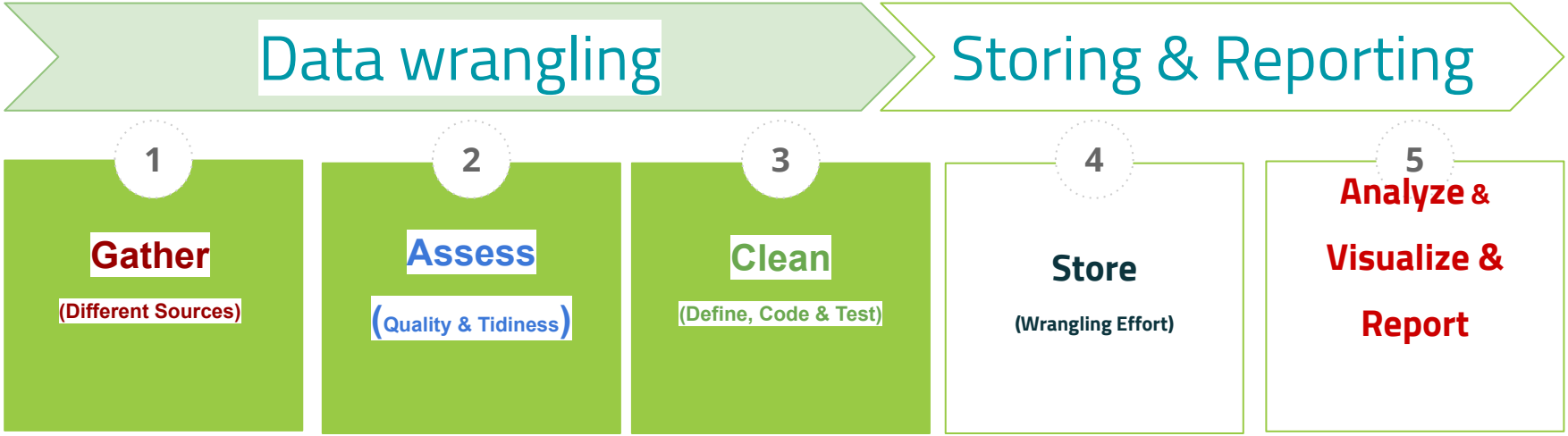
Reminder:

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Variables are stored in both rows and columns.
- Multiple types of observational units are stored in the same table.
- A single observational unit is stored in multiple tables.

https://www.jeannicholashould.com/tidy-data-in-python.html

itida

Agenda



itida

Data Cleaning

Inputs (DataFrames):

1. archive_df
2. api_df
3. Image_predictions_df

Assessment Summary

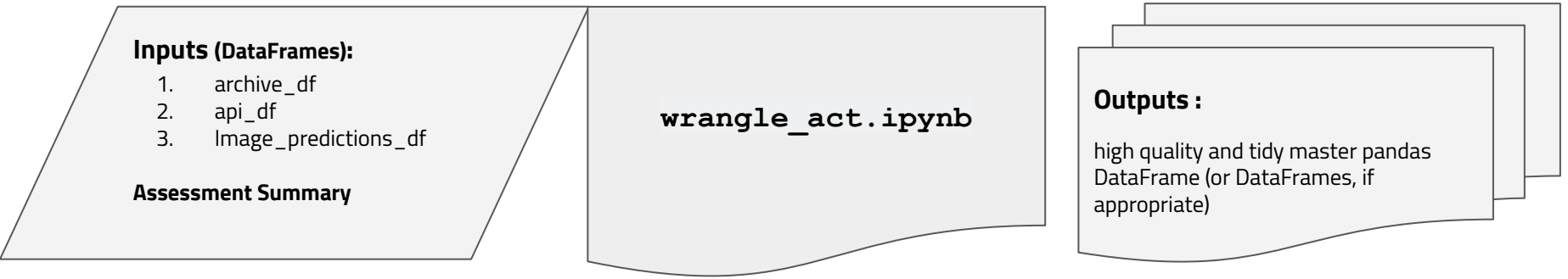
`wrangle_act.ipynb`

Outputs :

high quality and tidy master pandas DataFrame (or DataFrames, if appropriate)

- The **define**, **code**, and **test** steps of the cleaning process are clearly documented.
- Copies of the original pieces of data are made prior to cleaning.
- All issues identified in the assess phase are successfully cleaned (if possible) using Python and pandas, and include the cleaning tasks required to satisfy the Project Motivation.
- A tidy master dataset (or datasets, if appropriate) with all pieces of gathered data is created.

Data Cleaning

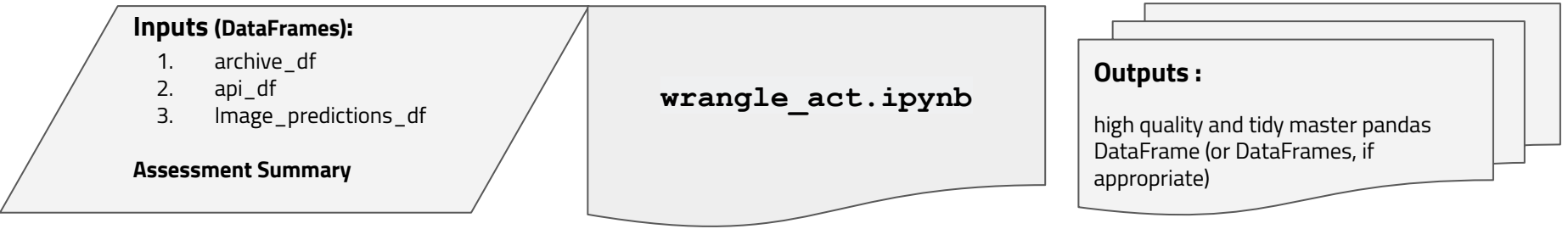


Tips & Tricks:

1. Issues that are of one off occurrence can be fixed manually.
2. Cleaning efforts that present in high rate, should be fixed programmatically.
3. EXtracting strings from text columns requires good understanding of REGEX
4. In the `expanded_url` column of the archive_df, the missing values are for tweets without photos so those entries can be dropped safely.

itida

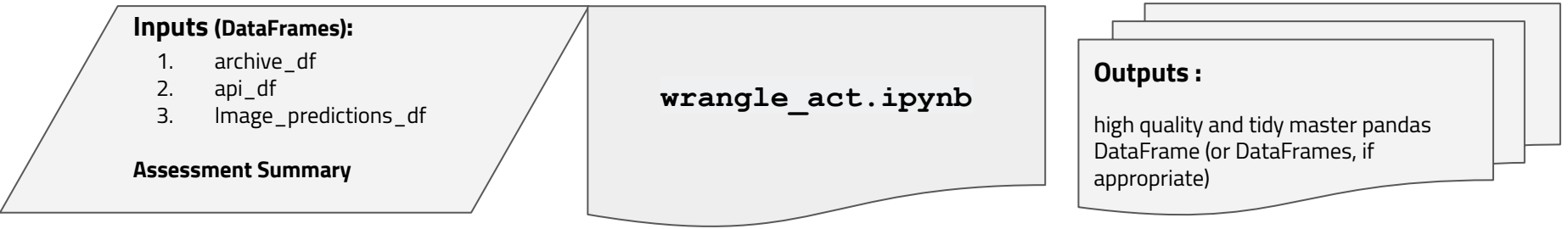
Data Cleaning (2)



Tips & Tricks:

1. For the tidiness issue of the dog_stage, there are actually both quality and tidiness issues and this issue is addressed in Discourse under a topic named (Pd.melt Inquiry)
2. There are many tidiness issues in the `image_predictions` of the type **Column headers are values, not variable names**.
3. A Function that can be of benefit `pd.wide_to_long()`.
4. Use the `image_predictions` dataframe to guide the selection and removal of *tweets without photos* in the `archive` dataframe

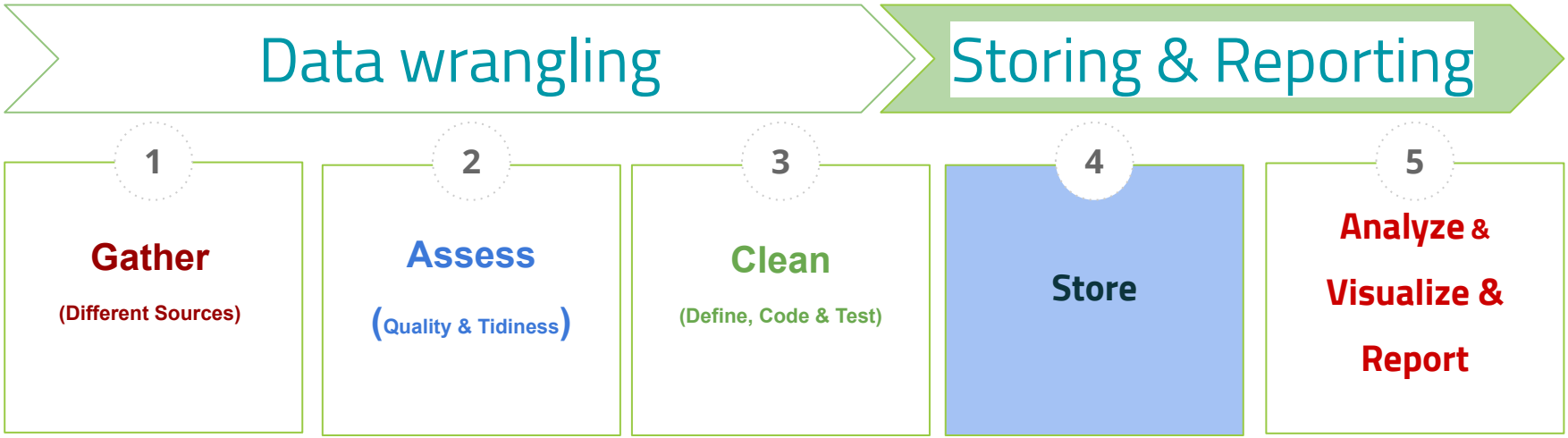
Data Cleaning (3)



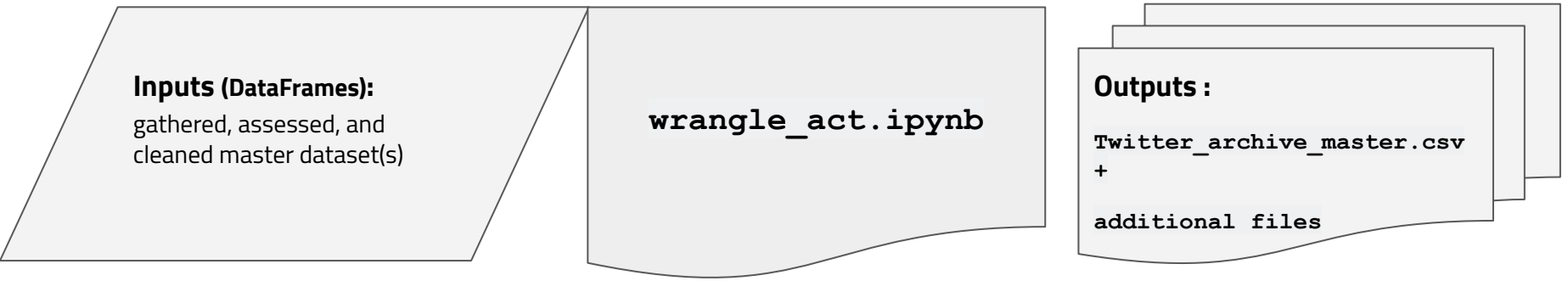
Tips & Tricks:

1. The following columns (`'in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_timestamp'`) will be utilized to shed the retweet and replies from our datasets and then will be dropped.
2. After dropping the replies and retweets, check the `image_predictions` table for extra tweet ids not in the archive table.

Agenda

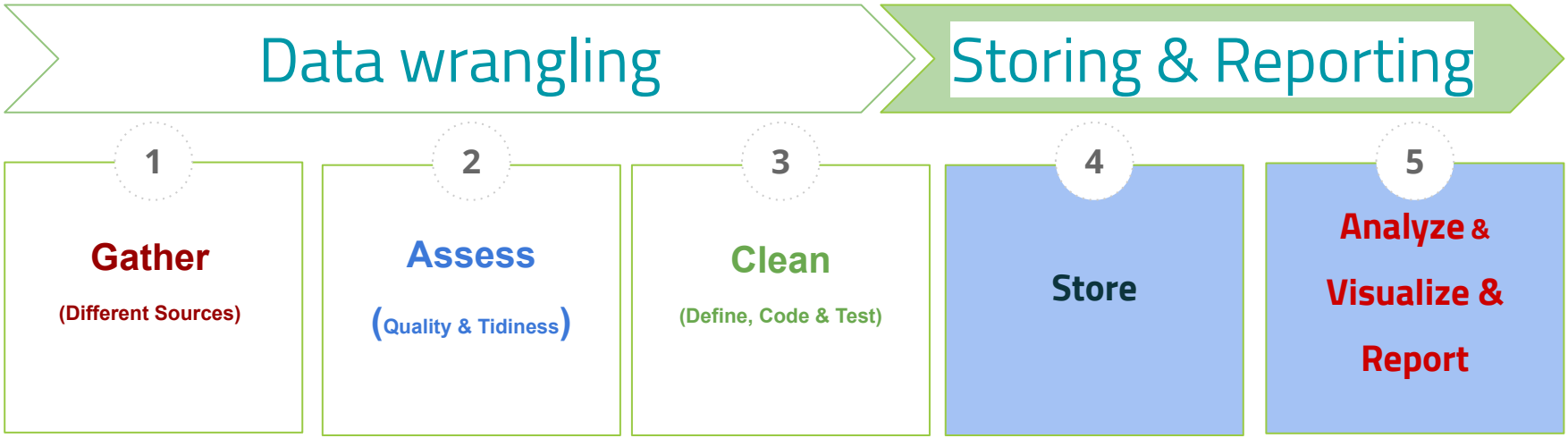

Data wrangling
Storing & Reporting
1
Gather
(Different Sources)
2
Assess
(Quality & Tidiness)
3
Clean
(Define, Code & Test)
4
Store
5
Analyze &
Visualize &
Report

Data Storing



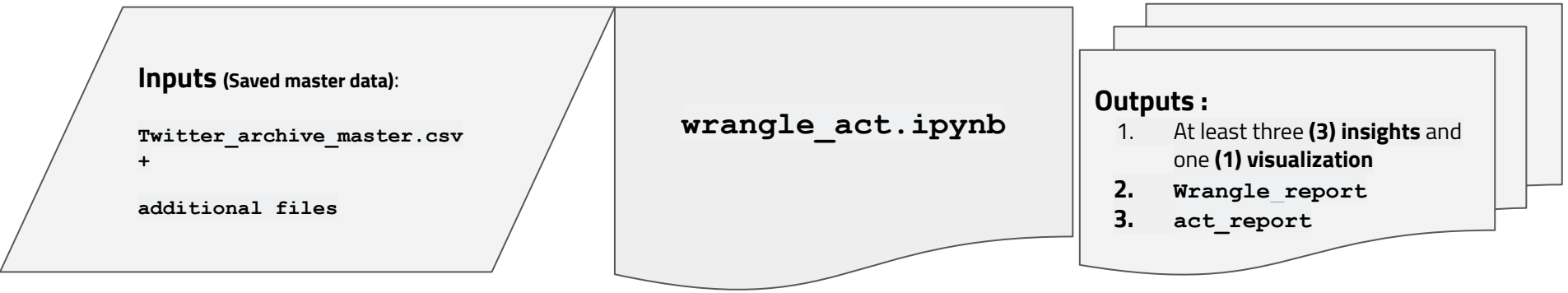
Store the clean DataFrame(s) in a **CSV file** with the main one named `twitter_archive_master.csv`. If additional files exist because multiple tables are required for tidiness, name these files appropriately. Additionally, you may store the cleaned data in a **SQLite** database (which is to be submitted as well if you do)..

Agenda



itida

Data Analysis, Viz & Reporting



- Analyze and visualize your wrangled data in your `wrangle_act.ipynb` Jupyter Notebook. At least three **(3) insights** and one **(1) visualization** must be produced.
- The student's **wrangling efforts** are briefly described. This document (`wrangle_report.pdf` or `wrangle_report.html`) is concise and approximately **300-600 words** in length.
- The three (3) or more insights the student found are communicated. At least one (1) visualization is included. This document (`act_report.pdf` or `act_report.html`) is at least **250 words** in length.

itida