Wrangle and Analyze Data

Requires Changes

7 specifications require changes

Rate this review

START

Good start here! It is great to see that you have followed the project structure correctly in your wrangle act notebook. You have also cleaned all of the important issues in the datasets and created nice reports describing your process and results. There are a couple of updates needed before we can pass this project, but they are all quite straightforward to implement:

- Tweet data need to be downloaded from the Twitter API (or at least the code for doing so is included in the notebook).
- There are incorrect tidiness issues.
- Remove incorrect issues.
- All listed issues must be cleaned.
- There are some issues in the clean-ups, check out the relevant specification below for more details.
- Update the analysis part in the notebook and the act and wrangle reports so they match the updated version of the notebook.

This is a challenging project, so please do not be discouraged by not passing it on your first few tries.

You may also ask questions in the Knowledge Hub to get some assistance from your fellow students and mentors.

Good luck with your next submission!

Code Functionality and Readability

All project code is contained in a Jupyter Notebook named wrangle_act.ipynb and runs without errors.

There are some unmet specifications in the `wrangle_act.ipynb` notebook that may require you to update some code blocks to fix. We will revisit this specification once they are corrected. Please make sure all of your code blocks do not throw any errors.

The Jupyter Notebook has an intuitive, easy-to-follow logical structure. The code uses comments effectively and is interspersed with Jupyter Notebook Markdown cells. The steps of the data wrangling process (i.e. gather, assess, and clean) are clearly identified with comments or Markdown cells, as well.

Code cells in the notebook are properly documented, good work here.

Gathering Data

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.

In this project, we need the student to demonstrate the knowledge to write a Python code to pull tweets from Twitter API. The solution may be achieved by using Tweepy plugin for the communication protocol. Note that the code does not have to work since some students do have a problem with getting Twitter developer account access, but it needs to at least be included (and ideally tested) in your notebook.

Todo: Read page "4. Twitter API" from the project descriptions. See that the first point at the bottom states:

> 1. twitter_api.py: This is the Twitter API code to gather some of the required data for the project. Read the code and comments, understand how the code works, then **copy and paste it into your notebook**.

As instructed there, please copy and paste twitter_api.py script into your notebook. You may try running it to see the errors it gathered too.

Assessing Data

Two types of assessment are used:

- **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.**

Great job doing both visual and programmatic assessments properly and documenting the process in the Jupyter notebook. You have performed various types of assessments, which is what I like about this particular section. To improve this section, I suggest adding some commentaries underneath each assessment result so readers may better follow your analysis. Check out the following lessons to remind yourself of important aspects of this section:

- Visual Assessment (lesson 3 concept 7)
- Programmatic Assessment (lesson 3 concept 14)

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.

Incorrect tidiness issues

These issues are incorrectly categorized as Tidiness Issues:

> - Not needed rating_denominator column in `twitter_archive` table, most times value is 10 with (2333 times) and we don't need the denominator in our analysis
> - Columns (in_reply_to_user_id column,retweeted_status_user_id) in `twitter_archive` table not needed in our analysis

Following the Hadley Wickham's Tidy Data description, tidy data is a standard way of mapping the meaning of a dataset to its structure. Issues that only require you to remove rows or columns or to update values, column names, or data types to fix them and do not require you to change the structure of the columns are hence should be considered Quality issues.

Please categorize them as Quality issues.

Check out these rules of tidy data. There are two interesting points to take here:

Firstly, tidy data should have the following characteristics:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

And here are the five most common problems with messy datasets (read up on the article for ways to correct them):

- 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.

Hence, there are two prominent tidiness issues in this project:

1. Information about one type of observational unit (tweets) is spread across three different datasets. Therefore, these three datasets should be merged as they are part of the same observational unit.
2. Dog stages should be a single column rather than four; one of the requirements for tidy data is that each variable forms a column.

Invalid issues

This issue is invalid:

> - Data in `image_predictions` table holds 3 separated observational units should be seperated into three tables: `image_prediction_1`, `image_prediction_2`, `image_prediction_3` for 1st, 2nd and 3rd predictions respectively

The 1st, 2nd, and 3rd predictions are **not** coming from different observations. They are all predictions for a particular tweet and therefore should be stored in a single row. It is alright to merge the prediction columns (e.g. p1, p2, p3 into prediction_order) although not critical. However, dividing into multiple tables is not appropriate.

Some listed issues are not cleaned

All listed issues must be cleaned. There is currently no clean-ups for the following issues:

> - (doggo ,floofer,pupper,puppo) in `twitter_archive` table represent one variable "stage" in four columns, and we will not invlove them in our analysis, so we can drop them
> - (p1,p2,p3) columns in `image_predictions` table represent one variable "prediction_order" in three columns
> - (p1_conf,p2_conf,p3_conf) columns in `image_predictions` table represent one variable "prediction_confidence" in three columns
> - (p1_dog,p2_dog,p3_dog) columns in `image_predictions` table represent one variable "prediction_is_dog" in three columns

Please clean these issues. Alternatively, as mentioned above, the last three issues are not needed and may be removed from your list.

(Optional) Similar issues should be merged

Some quality issues are too similar that they should be considered a single item. Here are some similar issues in the Quality Issues section, for instance:

Quality issues

***twitter_archive* table**

- Data type of tweet_id column is int64 instead of category
- Data type of in_reply_to_status_id column is float instead of category
- Data type of timestamp column is object instead of datetime
- Source column contains distracting HTML tags
- Data type of retweeted_status_id column is float instead of category
- Data type of retweeted_status_timestamp column is object instead of datetime
- Dublicated url strings in expanded_urls column
- Inaccurate username in expanded_urls column like (4bonds2carbon, kaijohnson_19,bbcworld) in urls column instead of (dog_rates)
- Embedded Urls like (https://www.gofundme.com/mingusneedsus,) and (https://www.gofundme.com/3yd6y1c,) (https://www.gofundme.com/help-my-baby-sierra-get-better,) strings in expanded_urls column
- Wrong urls of tweets in expanded_urls column
- Wrong 27 instead of 11.27 for rating of tweet with id number rating_numerator
- Data type of rating_numerator column is int64 instead of float
- 'None's in (doggo ,floofer,pupper,puppo) instead of null
- 'None's instead of null in name column
- Wrong rating_denominator column values except 10

***image_predictions* table**

- tweet_id data type is int64 instead of category
- img_num data type is int64 instead of category

***tweet_json_df* table**

- Data type of tweet_id is int64 instead of category

However, you have listed more than 8 quality issues even with these issues merged so it is okay if you prefer to keep them separated.

Cleaning Data

The define, code, and test steps of the cleaning process are clearly documented.

Define, **code**, and **test** cleaning sequences are generally performed **for each cleanup** instead of having one sequence for multiple cleanups. Here is how sequences for a single cleanup may look like:

Define

Remove 'bb' before every animal name using string slicing.

Code

```
In [ ]: df_clean['Animal'] = df_clean['Animal'].str[2:]
```

Test

```
In [ ]: df_clean.Animal.head()
```

However, you have grouped similar issues before cleaning them, and sometimes multiple issues took a fewer number of lines to clean rather than cleaning them one by one, so I think we may give this specification a passing mark.

The idea is to directly test each cleaning as you write them. This is an important technique in data wrangling as it avoids you from making hard-to-trace mistakes during the wrangling process.

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.

DataFrame objects were copied before cleaning, and a final cleaned dataset was created and filled with the cleaned data. Excellent work on this part. However, there are some issues that need to be cleaned properly before we may give this specification a passing mark (I have also included other issues that can be corrected to thoroughly clean your datasets, but they are optional so long you have cleaned at least 8 quality and 2 tidiness issues):

Retweets need to be removed - Quality issue

Retweets need to be removed as they may otherwise skew the result of your analysis (e.g. by double-counting the ratings). This may be done by removing rows that have non-empty `retweeted_status_id`, `retweeted_status_user_id`, or `retweeted_status_timestamp`. When this step is correct, there should be a fewer number of non-empty tweet ids.

Dog stages need to be combined into one column - Tidiness issue

Rather than having one column for each dog stage, combine them into a single column e.g. "stage" column where the value could be either of the four dog stages.

There are also cases where there are multiple dog stages in a row. Here is how to find them (`df` here is the Twitter archive dataset):

```
df.loc[(df[['doggo', 'floofer', 'pupper', 'puppo']] != 'None'
                ).sum(axis=1) > 1]
```

It will show you only the rows with multiple dog stages.

The pipeline needs to also handle these cases. One way to do so is by concatenating them with commas (",") as follows:

**Note: Remember to convert 'None' or np.NaN to empty string "" for all columns prior to running the following code e.g. `df.doggo.replace('None', '', inplace=True)` and `df.doggo.replace(np.NaN, '', inplace=True)`.

```
# Remember to convert None to empty string "" for all columns prior to
# running the following code.

df_1_clean['stage'] = df_1_clean.doggo + df_1_clean.floofer + df_1_clean.pupper + df_1_clean.puppo
df_1_clean.loc[df_1_clean.stage == 'doggopupper', 'stage'] = 'doggo,pupper'
df_1_clean.loc[df_1_clean.stage == 'doggopuppo', 'stage'] = 'doggo,puppo'
df_1_clean.loc[df_1_clean.stage == 'doggofloofer', 'stage'] = 'doggo,floofer'
```

To test if your update works, you may run the following code (it is better to put this in the "Test" section):

```
df.stage.value_counts()
```

(Optional) Find and update incorrect ratings - Quality issue

For rows with rating denominator != 10, there are cases where they are valid ratings and there are also invalid ones. The only way to find out (with what you have learned so far) is by manually reading the text. Fortunately, we do not have that many of those, so this is still doable. I'll give you three examples of possible scenarios:

- Tweet ID 810984652412424192. Text: "Meet Sam. She smiles 24/7 & secretly aspires to be a reindeer. \nKeep Sam smiling by clicking and sharing this link:\nhttps://t.co/98tB8y7y7t https://t.co/LouL5vdvxx". Extracted rating numerator and denumerator were 24 and 7. This is not correct. There shouldn't be any rating in this tweet.
- Tweet ID 835246439529840640. Text: "@jonnysun @Lin_Manuel ok jomny I know you're excited but 960/00 isn't a valid rating, 13/10 is tho". Extracted rating numerator and denumerator were 24 and 7. Correct ones should be 13 and 10
- Tweet ID 820690176645140481. Text: "The floofs have been released I repeat the floofs have been released. 84/70 https://t.co/NIYC820tmd". The extracted rating numerators and numerators of 84 and 70 are both correct.

(Optional) Ratings with decimal values incorrectly extracted - Quality issue

Rating numerators have not been properly cleaned. The current pipeline captures incorrect values when rating numerators contain decimals. For example, here is a value from one observation with tweet id 786709082849828864:

> "This is Logan, the Chow who lived. He solemnly swears he's up to lots of good. H*ckin magical af 9.75/10 https://t.co/yBO5wuqaPS"

Currently, the value 75 would be captured as the rating numerator. Try to capture the entire value from the text instead. Here is a code snippet as an example, where `df` here is the twitter archive dataset:

```
ratings = df.text.str.extract('((?:\d+\.)?\d+)\/(\d+)', expand=True)
```

`ratings` series object will then contain all rating numerators with decimals and rating denominators (without decimals). The next step is to extract only the numerators and denumerators from `ratings` dataframe, and then update your dataset's fields with extracted rating numerators and denominators (**NOTE: Do not forget to convert the field datatype into Float, `astype` function may be used here**):

```
df.rating_numerator = ratings
```

To improve it even further, you may also want to try adjusting the code so rating denumerators would also capture decimal values.

I find tools such as this one to be helpful in finding the correct regex.

Storing and Acting on Wrangled Data

Students will save their gathered, assessed, and cleaned master dataset(s) to a CSV file or a SQLite database.

The master dataset is analyzed using pandas or SQL in the Jupyter Notebook and at least three (3) separate insights are produced.

At least one (1) labeled visualization is produced in the Jupyter Notebook using Python's plotting libraries or in Tableau.

Students must make it clear in their wrangling work that they assessed and cleaned (if necessary) the data upon which the analyses and visualizations are based.

Well done with the visualizations and insights, they look great! Once you have properly cleaned the issues as recommended above, please redraw the visualizations and update your insights to reflect the new data.

Report

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.

Please update your wrangle report document to reflect the changes from applying my comments in the above specifications.

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.

Please update your report with the new visualizations and analyses to reflect the new data due to the above corrections.

(Optional) We suggest including pictures for aesthetic and additional context purposes on top of the required visualizations. Example: include a screenshot of a specific tweet, a specific breed of dog, etc. Anything to get the reader engaged. Picture this report like a blog post or magazine article; we want people to be engaged and have fun while reading.

Project Files

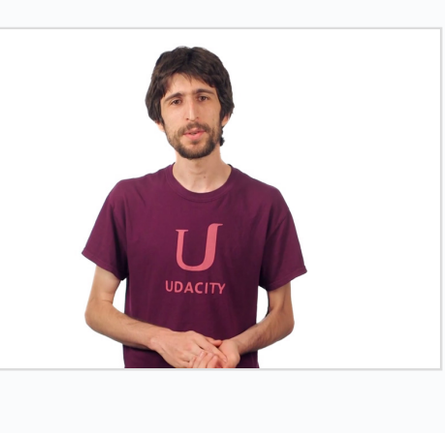
The following files (with identical filenames) are included:

- **wrangle_act.ipynb**
- **wrangle_report.pdf or wrangle_report.html**
- **act_report.pdf or act_report.html**

All dataset files are included, including the stored master dataset(s), with filenames and extensions as specified on the Project Submission page.

RESUBMIT PROJECT

DOWNLOAD PROJECT



Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH