wrangle_report

July 8, 2022

0.1 Reporting: wragle_report

- Create a **300-600 word written report** called "wrangle_report.pdf" or "wrangle_report.html" that briefly describes your wrangling efforts. This is to be framed as an internal document.

Wrangling comprises gathering, assessing and cleaning of dataset(s). In this project, I was required to gather 3 different dataset, through different sources. Firstly, I did all necessary imports of modules and packages like pandas, request and tweepy. The first dataset twitter-archive-enhanced.csv containing WeRate dog tweet was downloaded manually from a link provided in the project gathering guide. After which the dataset was loaded into a dataframe "tweet_archive_df" for assess and cleaning. The second dataset which contains image predictions was downloaded using the request module and written to a file "image_predictions.tsv". After downloading "image_predictions.tsv", I noticed that when I loaded the file into a dataframe using ′pd.read_csv′ function without a "sep" argument, the dataframe had a weird display with lots of ′′⁀ i.e the content of the file were separated using tabs. I quickly passed ′ sep=""⁀′, and the dataframe displayed the right column and its respective values. Lastly on gathering, was the use of twitter′s tweepy api to download "tweet_json.txt" r extract of retweet count and favorite count was made. For the last gathering to be done successfully, I requested for access to query twitter′s api from twitter, to which I was given the permission where I got the needed keys to successfully get the third dataset. In the assessing phase, both visual and programmatic assessment was carried out. Visually the data were viewed using excel package. Each dataset was programmatically assessed using ′dataframe.head(5),′ which displayed the first five record; ′dataframe.tail(5)′, which displayed the last five record; ′dataframe.sample(5)′, which displayed five random record several times whenever the cell is being run; ′dataframe.describe()′, which displayed a descriptive statistical summary of a dataframe; ′dataframe.info()′, which displayed a structural summary of a dataframe. ′dataframe.info()′ showed total number of column and rows in a dataframe. It also showed each column data type and its total values specifying missing values. Moreso, dataframe[dataframe[′tweet_id′].isnull()] checks if there is any null values of tweeet_id in a dataframe. dataframe[dataframe.tweet_id.duplicated()] was used to find cases of duplicates of tweet id in a dataframe. Also, dataframe.name.value_counts().head(7) displays the various names of dogs and the number of times that a name appears on the ′name′ column. The value counts for other columns were checked also. Checks for abnormal rating numerator and denominator was carried out respectively thus: tweet_archive_clean.query(′rating_numerator > 16′) and tweet_archive_clean.query(′rating_denominator > 10′). A lot of quality and tidiness issues were raised and they include:

Quality issues Under tweet archive dataframe we have;

```
1. missing/NaN values for in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, etc
```

2. some columns don't matter. Example: in_reply_to_status_id, in_reply_to_user_id, etc
3. incorrect datatype for timestamp
4. inconsistency in name values character case
5. Wrong/incorrect values like a, the, very, an, this, just,... in name column
6. missing data in expanded_urls column
7. Error values on dog rating

Under Image predictions dataframe we have

1. Some records with predictions are not dogs
2. inconsistency in p1, p2 and p3 character case value

Under Retweet count dataframe we have

1.rename id column to tweet_id

Tidiness issues

1. rating_numerator and rating_denominator should be merged as one column

In the cleaning phase, we made a copy of the dataframe and went forward with resolving all the quality and tidiness issues raised. We started by dropping the unimportant columns with NaN, changing incorrect datatype to right ones, converting inconsistent character case values to lower cases, dropping rows predictions that weren't dogs, correcting error dog rating and finally renaming id column in retweet count dataframe to tweet_id. In this project, a lot of iteration over the whole wrangling process were done to finally come out with a clean dataset

```
In [ ]:
```