Udacity – Data Analyst Nanodegree Program

WRANGLING WeRateDogs TWITTER DATA TO CREATE INTERESTING AND TRUSTWORTHY EXPLORATORY / PREDICTIVE ANALYSES AND VISUALIZATION USING DIFFERENT MACHINE LEARNING ALGORITHMS

Project 8

Esra Arı

İSTANBUL, 2018

1. WRANGLE REPORT

Tasks in this pdf file are given following:

- Data wrangling, which consists of:
    - Gathering data
    - Assessing data
    - Cleaning data (Missing value treatment)
    - Storing/Exporting data

1.1 Gathering Data for this Project

The three pieces of data gathered as described below in a Jupyter Notebook

1. The WeRateDogs Twitter archive includes 2356 observations and 17 features.
2. 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. This file (image_predictions.tsv) is hosted on Udacity's servers. It has 2075 entries and 12 columns without any missing values.
3. Each tweet's retweet count and favorite ("like") count at minimum, and any additional data. Using the tweet IDs in the WeRateDogs Twitter archive, query the Twitter API which is mentioned in Twitter API section detailed 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. Then read this .txt file line by line into a pandas DataFrame with (at minimum) tweet ID, retweet count, and favorite count. It includes 3715 entries and 11 columns.

1.2 Assessing Data for this Project

After gathering each of the above pieces of data, it is required to assess them visually and programmatically for quality and tidiness issues. With this aim in this section, each three pandas data frame gathered previous section will be investigated. Assessing data was done both visually (scrolling through the data in your preferred software application) and programmatically (using code to view specific portions and summaries of the data). Both quality and tidiness issue were noted end of this section. Also, sources of low quality /dirty and messy/untidy data were mentioned shortly.

1.2.1 Sources of dirty and messy data

Dirty data is also called as low quality data or content issues. There are lots of sources of dirty data. Basically, anytime humans are involved, there's going to be dirty data. There are lots of ways in which we touch data we work with.

- user entry errors
- no data coding standards, or having standards poorly applied, causing problems in the resulting data
- integrated data where different schemas have been used for the same type of item
- legacy data systems, where data wasn't coded when disc and memory constraints were much more restrictive than they are now. Over time systems evolve. Needs change, and data changes
- no unique identifiers it should
- lost in transformation from one format to another
- programmer error
- corrupted in transmission or storage by cosmic rays or other physical phenomenon

Messy data is also called as untidy data or structural issues. Messy data is usually the result of poor data planning. Or a lack of awareness of the benefits of tidy data. Fortunately, messy data is usually much more easily addressable than most of the sources of dirty data mentioned above.

1.2.2 Noted quality and tidiness issues

df1 : WeRateDogs Twitter Dataset

It includes 2356 entries and 17 columns.

- **Quality**
  - Names column should be cleaned, there is invalid records like a, the, an, the, very, unacceptable which is start with lowercase.

- timestamp, retweeted_status_timestamp column type should be date instead of object.
- text includes "'&amp;" instead of "&".
- invalid rating_denominator (different than 10). However, I checked them manually and they are true denominators, so there is no problematic extraction from text.
- Tweets_ids with no images however this problem will be solved when I joined with image prediction dataset. Because, I am not expecting the prediction which do not have any image.
- Missing values expressed as "none". (name, duppo, flopper, etc.)
- in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id data types should be integer instead of float.
- We only want original ratings (no retweets) that have images
- From the source column, via which channel users connected to twitter. Therefore, column should be cleaned.
- excluding any tweet that is a retweet.

Tidiness

- joining with tables df2 and df3.
- getting together stages in one column.
- adding new features like gender, etc.

df2 : Image Prediction Dataset

Great dataset which has 2075 entries and 12 columns without any missing values.

Quality

- Missed ID's exists in the dataset compare to d1
- Duplicated jpg_url
- p1, p2, p3 columns should be standardized as all lowercase and "-" expression should be removed.

Tidiness

- joining with tables df3 and df1.
- creating final dog prediction

df3 : Tweepy API Dataset

It includes 3715 entries and 11 columns.

Quality

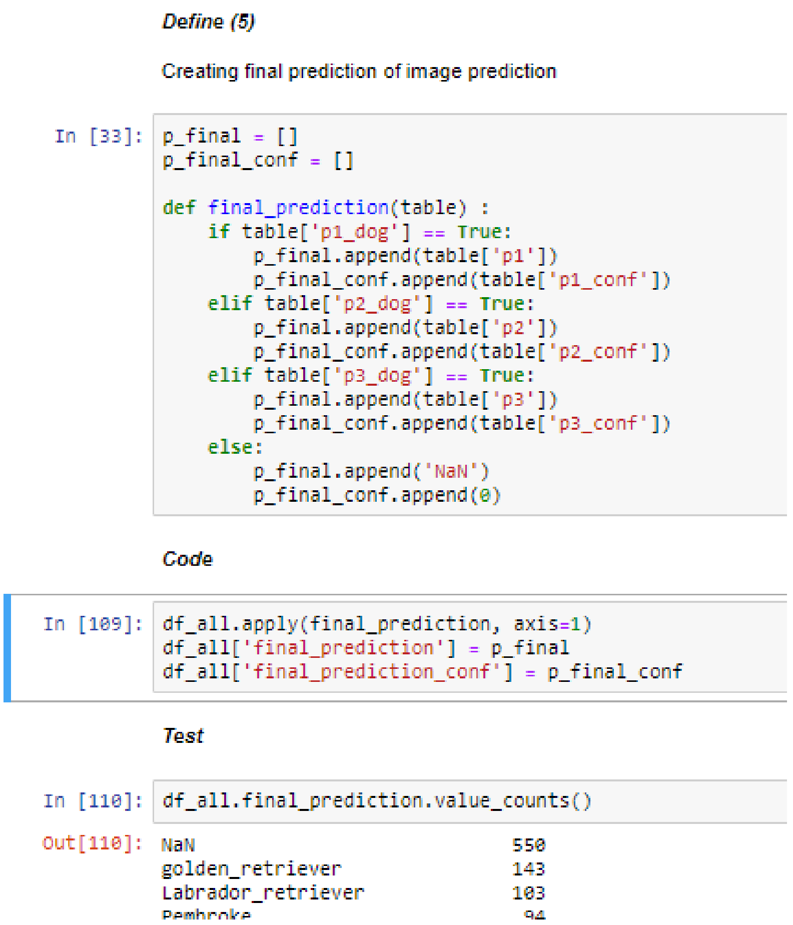
- contributors, coordinates, place and geo features should be excluded due to high missing ratio.
- 1222 numbers of id variable are duplicated
- id=666337882303524864 exits 4 times in the dataset with same results.
- lang indicates that the language of tweet. I wondered how "tl" lang is texted. Then, I realized id=668967877119254528 is problematic input.

Tidiness

- joining with tables df2 and df1.
- favorited, retweeted columns include always false inputs, therefore it should be excluded.

1.3 Cleaning Data for this Project

In this section, defined tidiness and quality issues were cleaned. As it can be seen from below picture, 19 data problems were defined, coded and tested one by one.



Define (5)

Creating final prediction of image prediction

```
In [33]: p_final = []
         p_final_conf = []

         def final_prediction(table) :
             if table['p1_dog'] == True:
                 p_final.append(table['p1'])
                 p_final_conf.append(table['p1_conf'])
             elif table['p2_dog'] == True:
                 p_final.append(table['p2'])
                 p_final_conf.append(table['p2_conf'])
             elif table['p3_dog'] == True:
                 p_final.append(table['p3'])
                 p_final_conf.append(table['p3_conf'])
             else:
                 p_final.append('NaN')
                 p_final_conf.append(0)
```

Code

```
In [109]: df_all.apply(final_prediction, axis=1)
          df_all['final_prediction'] = p_final
          df_all['final_prediction_conf'] = p_final_conf
```

Test

```
In [110]: df_all.final_prediction.value_counts()

Out[110]: NaN                                 550
          golden_retriever                    143
          Labrador_retriever                  103
          Pembroke                             94
```

Figure 1. Cleaning example[1]

1.4 Storing

Assed and cleaned data was stored in a CSV file with the main one named twitter_archive_master_v2.csv.

[1] Detailed codes can be find from part 1 jupyter notebook.