WRANGLE REPORT
WE RATE DOGS

Odiba Anthony
2019

Introduction

For this project, we wrangled the WeRateDogs dataset. WeRateDogs is a twitter account that rates peoples dogs with a humorous comment about the dog.

My tasks in this project were as follows:

- Data wrangling, which consists of **Gathering data, Assessing data** and **Cleaning data**
- Storing, analyzing, and visualizing your wrangled data
- Reporting

This report is meant to serve as an internal document with a focus on my wrangling efforts.

Gathering Data

Three pieces of data were gathered from three sources:

1. The WeRateDogs Twitter archive. Download manually.
2. The tweet image predictions, i.e., what breed of dog (or other objects, animal, etc.) is present in each tweet according to a neural network. This file was downloaded programmatically from Udacity's servers using the Requests library.
3. Additional data you find interesting gotten by querying Twitter API for each tweet's JSON data using Python's Tweepy library.

Key Points

To Note while data wrangling: **Only original ratings (no retweets) that have images, no replies.**

Assessing Data

I used the below-listed methods and functions at various stages while assessing the data. Sometimes in a combination.

Method & Functions	Application
.head()	Get a sense of the full, sliced or indexed dataframe
.describe()	Get a sense of the full, sliced or indexed dataframe
.sample()	Get a sense of the full, sliced or indexed dataframe
indexing	
.value_counts()	Get a sense of the full, sliced or indexed dataframe
`percent_missing(eval(i), i)`	A custom function that takes two arguments and returns the number percentage of entries missing from a dataset.
`msno.matrix(archive_df)` `msno.bar(archive_df)`	Missingno imported as msno. Used to visualize missing data
`Pandas_profiling.ProfileReport(df)`	
`With pd.option_context('display.max_colwidth',200):` `  display(#df slice)`	To fully display the sliced dataframe
`Assert (image_predictions_df.jpg_url.value_counts()>1).any()`	An alternative way to check if any uses the same picture

Findings

From our custom function, I saw that the dataframe with missing values is the Twitter Archive dataframe with **~28% missing**. I dove deeper using pandas profiling and I got a better view of the archive dataset like where the values are missing, the frequency of the values in each variable of the datasets etc.

All in all, I was able to highlight issues that I needed to address while cleaning the three dataframes.

Below, I highlighted the issues and how they were taken care of.

Cleaning the Data

Firstly I used .copy() method on the three dataframes in case of mistakes

S/N	ISSUES	Approach and *Functions used*
1	**Missing Data** Missing counts for doggo, floofer, pupper and puppo columns in **archive_df** In several columns null objects are non-null (None to NaN) Inaccurate or Missing Names	***First approach*** • create two copies of the archive_df_clean dataframe • In one copy, Use forloop and `.str.contains()` to re-identify if text contains each column header. Include text if it is found. If not, return `NaN`. • create a separate column for the dog stages ***Second Approach*** • In another copy, go ahead and create an extra column with dog stages without re-identifying if the text contains each column header. • Compare relevant entries in both copies ***Finally:*** Determine the better approach then, Drop **dog_types** column
2	**Inaccurate or Missing Names**	• The first approach is to check the name column for names with the first letter in lowercase. • Next call our **actual_names()** function on the whole df using `.apply()` method

3	**Data Contains Retweets, Replies and Duplicates**	Check for rows where **retweeted_status_id** and **retweeted_status_user_id** have a number instead of NaN • Use `sum(archive_df_clean.str.match(' ^RT @'))` to find the retweets • Remove all those rows where there is a retweet_status present i.e `notnull == False`. 181 rows will be removed in archive_df_clean. Remove all rows where expanded URLs are duplicated in **archive_df_clean** `.drop_duplicates(subset=['jpg_url],keep='last',inplace=True)` Remove replies
4	**Numerator and Denominators have Incorrect or Inaccurate ratings** used regex to look for rating that are decimals or floats instead of integers and I found ratings like 13.5/10, 9.5/10 which were incorrectly extracted as 5/10. There are instances of ½, 50/50, 4/20 and 24/7 have been confused as ratings.	• Check the tweets with decimals • use `actual_decimal()` function which uses regex groups to replace the incorrect value with the correct value
5	**p1, p2, and p3 in predictions_df_clean dataframe is inconsistent** p1, p2, and p3 are inconsistently capitalized, also contain dash and spaces instead of underscores instead in entries.	• convert strings in p1, p2, and p3 lowercase using .lower() method
6	**the jpg_url column contains Duplicates** Some tweet_ids have the same jpg_url.	• Use • `.drop_duplicates(subset=['jpg_url],keep='last',inplace=True)`
7	**p1, p2, p3 should be one column** There are multiple columns containing the same type of data, e.g. p1, p2, p3 all contain dog breed predictions	• I created 2 prediction dfs one I converted wide to long using pd.wide_to_long() and renamed prediction_df_alt • The other will be merged along with the other two dfs to form a single dataframe • For **both** of them, I created a new column called **prediction** that had three categories; **dog, not_dog,** and **not_certain** based on the **p1_dog, p2_dog** and **p3_dog** columns • For the prediction df to be merged later I created two columns **dog_type(representing dog breeds)** and **confidence_level** • To do this use `.apply()` and `conf_and_type2()` function **predictions_df_clean.apply(conf_and_type2, axis=1)**

8	**Incorrect Datatypes**	• Tweet_id is an integer, it should be a **string** • **in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id** should be integers or strings instead of float. • **timestamp** and **retweeted_status_timestamp** are object types, change to **DateTime** **Do this for all 3 dataframes where appropriate** **Approach** Use `.astype()` method and the pd.to_datetime() Use `assert` to test
	ADDRESS ALL TIDINESS ISSUES	
1	Drop appropriate columns and Merge dfs	• Rename created_at column in archive_api_df to 'timestamp' • Merge **api_twitter_df_clean** with **archive_df_clean** • In the merged df, remove entries without images • remove entries in **predictions_df_clean** that are not in **archive_api_df** • Merge **archieve_api_df** with **predictions_df_clean** to form a final single Dataframe called **twitter_archive_master**
	Here are some things I did not have the luxury of time to clean before the analysis	• some entries in the Archive_df have one single entry representing two separate dogs at different dog stage and with different names. Ideally to have a more robust dataset, in cases where that occurs I would have separated it into two unique entries, but I was constrained by time. • The source column has the html link tag "<a>..." still attached to it • Text column includes both text and short version of link

Final Dataframe= twitter_archive_master

```
twitter_archive_master.columns
```

```
Index(['dog_stages', 'expanded_urls', 'name', 'rating_denominator',
       'rating_numerator', 'text', 'timestamp', 'tweet_id', 'User_followers',
       'favorite_count', 'retweet_count', 'jpg_url', 'img_num', 'p1',
       'p1_conf', 'p2', 'p2_conf', 'p3', 'p3_conf', 'Prediction', 'dog_type',
       'confidence_level'],
      dtype='object')
```

```
twitter_archive_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1904 entries, 0 to 1903
Data columns (total 22 columns):
dog_stages            292 non-null object
expanded_urls         1904 non-null object
name                  1323 non-null object
rating_denominator    1904 non-null float64
rating_numerator      1904 non-null float64
text                  1904 non-null object
timestamp             1904 non-null object
tweet_id              1904 non-null int64
User_followers        1900 non-null float64
favorite_count        1900 non-null float64
retweet_count         1900 non-null float64
jpg_url               1904 non-null object
img_num               1904 non-null int64
p1                    1904 non-null object
p1_conf               1904 non-null float64
p2                    1904 non-null object
p2_conf               1904 non-null float64
p3                    1904 non-null object
p3_conf               1904 non-null float64
Prediction            1904 non-null object
dog_type              1904 non-null object
confidence_level      1605 non-null float64
dtypes: float64(9), int64(2), object(11)
memory usage: 327.3+ KB
```