WERATEDOGS

A Report on Data Analysis Effort

ABSTRACT

"Valuable insights comes from hard wrangling" Throughout this report we will review the back scene circumstances, steps, actions, encountered challenges and the result during doing the "Data Analysis Process" for "WeRateDogs" Twitter account.

Maged Baheig
Udacity Student

11/13/2020

Contents

Background 2

Study Context 2

Study Objective 3

Challenges 3

Datasets 4

Input Datasets 4

Output Datasets 4

Data Analysis Process Efforts 5

Stage1: Project Preparation 6

1. Questions to be answered 6

2. Import Required Packages 6

Stage2: Main Phases 7

1. Data Gathering 7

2. Data Assessing 9

3. Data Cleaning and Storing 10

4. Data Analyzing and Visualizing 17

5. Making Reports 20

Background

Study Context

- **@dog_rates**, also known as **WeRateDogs** is a **Twitter account** that rates people's dogs with a humorous comment about the dog.

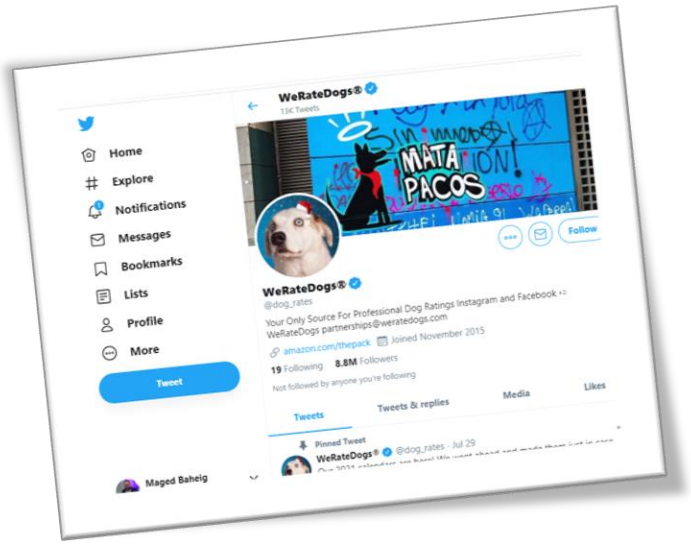


Figure 1: ***WeRateDogs Twitter Account***

- These **ratings** almost always have **a denominator of 10**. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc.

Figure 2: ***WeRateDogs Tweet Example***

- Nowadays, WeRateDogs has **over 8 million followers** and has received international **media coverage**.
- **WeRateDogs profile URL:** https://twitter.com/dog_rates?s=20

Back to Contents

Study Objective

- Wrangling and **analyzing WeRateDogs Twitter account's tweet data** (tweet ID, timestamp, text, etc.) for 5000+ of their tweets as they stood on **August 1, 2017**.
- **We will work on:**
    - **Original ratings only** (no retweets and/o no replies).
    - Tweets that **have images only.**

- Our **goal** is to **create interesting and trustworthy analyses and visualizations**. The Twitter archive is great, but it only contains very basic tweet information.

Challenges

1) **Messy data with low quality**.That requires additional **hard gathering**, then **assessing** and **cleaning** is required for "*Wow!*"-worthy analyses and visualizations.
    - **Overcoming data Challenge techniques:**
    1. Take a good time in knowing the datasets and WeRateDogs Twitter account.
    2. An organized mindset.
    3. Start small-end big way.
    4. A lot of practice with "try-error".

2) **Twitter Approval Cycle**. Creating a **Twitter Developer Account** and getting the **approval** to use Twitter API. It takes a lot of time with a provability to be declined.
    - **Overcoming Approval Cycle Challenge techniques:**
    a. Start approval cycle early.
    b. Demonstrate my request purpose clearly with integrity.
    c. Answer every piece of question from Twitter regarding my request.

3) **Data Cleaning Complex Code**. Some cleaning actions may require a complex code.
    - **Overcoming Complex Code Challenge techniques:**
    a. Search at Udacity FWD Community (Discourse).
    b. Use google search.
    c. Search at (stackoverflow.com, geeksforgeeks.org, github.com).

Back to Contents

Datasets

*WE started with using **3 datasets** as "**data input**", **processed** them by **making** required assessing and cleaning actions, finally ended with 2 datasets in hand as "**data output**".*

Input Datasets

Input_File #1**: **twitter_archive_enhanced.csv – a file on hand

- **Manually downloaded from Udacity classroom.**
- Contains basic tweet data for all 5000+ of their tweets, but not everything.
- Contains one column named ['tweet's text'], which Udacity instructor used to extract rating, dog name, and dog "stage" (i.e. doggo, floofer, pupper, and puppo).
- Of the 5000+ tweets, Udacity instructor has filtered for tweets with ratings only (there are **2356**).

Input_File #2**: **imagee_prediction.csv – Programmatically downloaded file

- **Programmatically downloaded using python request library.**
- A neural network output file, which made by Udacity instructor, as he ran every image in the WeRateDogs Twitter archive through a neural network that can classify breeds of dogs*.
- The results: a table full of image predictions alongside each tweet ID, image URL, and the image number that corresponded to the most confident prediction.

Input_File #3**: **tweet_json.txt – API file

- **Programmatically downloaded using python tweepy library.**
- Contains every missing data from **twitter_archive_enhanced.csv** file, such as: **retweet count**, **favorite count** and **followers count**.

Output Datasets

Output_File #1**: **twitter_archive_master.csv

- The main output file.
- Contains the clean datasets from **input_file#1** (**twitter_archive_enhanced.csv**) and input **file#3 (tweet_json.txt – API file)**.

Output_File #2**: **imagee_prediction.csv

- The Complementary output file.
- Contains the clean dataset from input_**file#2 (imagee_prediction.csv)**.

Back to Contents

Data Analysis Process Efforts

- Executing the **"Data Analysis Process"** ***–for "WeRateDogs" Twitter account–*** required a huge efforts.
- **Efforts went through 2 main stages as below...**

Stage1: Project Preparation

1. **Questions to be answered**
2. **Import Required Packages**

Stage2: Main Phases

1. **Data Gathering**
2. **Data Assessing**
3. **Data Cleaning and Storing**
4. **Data Analyzing and Visualizing**
5. **Making Reports**

Back to Contents

Stage1: Project Preparation

1. Questions to be answered

Below are what we tried to find out...

- **Q1** What are the main devices/apps that WeRateDogs' users use?
- **Q2** Is there a relationship between dog rates and retweet count?
- **Q3** Is there a relationship between dog rates and favorite count?
- **Q4** Is there a relationship between favorite count retweet counts?
- **Q5** What time that most of tweets are tweeted at?
- **Q6** Is high confidence prediction meet reality more than low ones?

2. Import Required Packages

*Below **snapshot** is taken from Jupyter Notebook, and demonstrate the imported **required packages** to execute **python code**.*

```
# import required packages to deal with files and folder
import os
import zipfile

# import required packages to gather required data and read/handle it
# Gathering data needed packages
import tweepy
from tweepy import OAuthHandler          ###################################
from timeit import default_timer as timer ###################################
import requests
import re
import json
import glob
from PIL import Image
from io import BytesIO
from bs4 import BeautifulSoup
from scipy import stats
import warnings                           ###################################
warnings.filterwarnings('ignore')         ###################################

# import required packages to read, discover, manipulate, organize, analyze and visualize data
# Assessing and cleaning data needed packages
import pandas as pd
import numpy as np
import datetime

# Analyze and Visualize data needed packages
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('darkgrid')

# call magic key words matplotlib
%matplotlib inline
```

*Figure 3: **Python required packages***

Back to Contents

Stage2: Main Phases

1. Data Gathering

***Input_File #1**: twitter_archive_enhanced.csv – a file on hand*

- We already downloaded this file manually. However, We preferred to re-download it programmatically**.**
- **File gathering done through 2 steps:**
  d. Download (twitter_archive_enhanced.csv) file programmatically.
  e. Read (twitter_archive_enhanced.csv) file and import it to Jupyter workspace using pandas DataFrame.

```
In [6]:
1 # Check imported df before proceeding to next step
2 archive_df.head(1)
```

Out[6]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	ret
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	<a href="http://twitter.com/download/iphone" r...	This is Phineas. He's a mystical boy. Only eve...	

Figure 4: ***twitter_archive_enhanced.csv***

***Input_File #2**: imagee_prediction.csv – Programmatically downloaded file*

- **File gathering done through 2 steps:**
  1. Download (image_predictions.tsv) file programmatically.
  2. Read (image_predictions.tsv) file and import it to Jupyter workspace using pandas DataFrame.

```
In [11]:
1 # Check imported df before proceeding to next step
2 image_df.head(1)
```

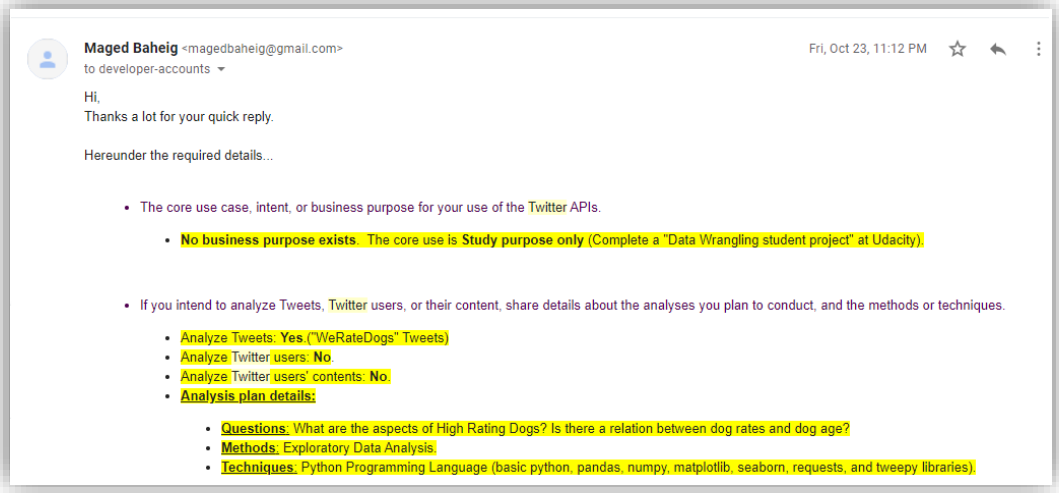
Out[11]:

	tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.465074	True	c

Figure 5: ***imagee_prediction.csv***

***Input_File #3**: tweet_json.txt – API file*

- **File gathering done through 2 steps:**
  1. Query Twitter API and Create (tweet_json.txt) file programmatically.
  2. Extract required info from (tweet_json.txt) and load it to Jupyter workspace using pandas DataFrame.



Maged Baheig <magedbaheig@gmail.com> Fri, Oct 23, 11:12 PM

to developer-accounts

Hi,
Thanks a lot for your quick reply.

Hereunder the required details...

- The core use case, intent, or business purpose for your use of the Twitter APIs.
  - **No business purpose exists**. The core use is **Study purpose only** (Complete a "Data Wrangling student project" at Udacity).
- If you intend to analyze Tweets, Twitter users, or their content, share details about the analyses you plan to conduct, and the methods or techniques.
  - Analyze Tweets: **Yes**.("WeRateDogs" Tweets)
  - Analyze Twitter users: **No**.
  - Analyze Twitter users' contents: **No**.
  - **Analysis plan details:**
    - **Questions**: What are the aspects of High Rating Dogs? Is there a relation between dog rates and dog age?
    - **Methods**: Exploratory Data Analysis.
    - **Techniques**: Python Programming Language (basic python, pandas, numpy, matplotlib, seaborn, requests, and tweepy libraries).

Figure 6: ***Twitter Dev. Account communication***

File 3_Step 1. Query Twitter API and Create (tweet_json.txt) file programmatically

```
# Query Twitter API for each tweet in the Twitter archive and save JSON in a text file
# These are hidden to comply with Twitter's API terms and conditions

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

auth = 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)
```

```
# Experimenting to extract one tweet's id information after creating the API oject
exp_tweet = api.get_status(archive_df.tweet_id[0], tweet_mode='extended')
content = exp_tweet._json
print(content)
```

```
{'created_at': 'Tue Aug 01 16:23:56 +0000 2017', 'id': 892420643555336193, 'id_str': '8924206435
s Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 13/10 https://t.co/MgU
display_text_range': [0, 85], 'entities': {'hashtags': [], 'symbols': [], 'user_mentions': [],
420639486877696, 'id_str': '892420639486877696', 'indices': [86, 109], 'media_url': 'http://pbs.
```

Figure 7: ***Code to query Twitter API***

```
In [24]: api_df.head()
```

Out[24]:

	tweet_id	favorite_count	retweet_count	followers_count	friends_count
0	666020888022790149	2360	453	8869482	19
1	666029285002620928	120	41	8869482	19

Figure 8: ***tweet_json.txt – API file***

Back to Contents

2. Data Assessing

*Assessing data process went through **2 sections**:*

1. Assessing Effort.

- After gathering each of the above pieces of data, we assessed for detecting quality and tidiness issues.
- **Assessing done in 2 ways:**
  1. **Visually**, using Jupyter Notebook and Spreadsheets.
  2. **Programmatically**, using Jupyter Notebook.

2. Assessing Result.

- **Assessing efforts ended with 30 detected data issues as below:**
  - **(22)** Data quality issues.
  - **(6)** Tidiness issues.
  - **(2)** Feature Engineering actions.
- **We will list them in the next section (along with Data Cleaning and Storing Section**

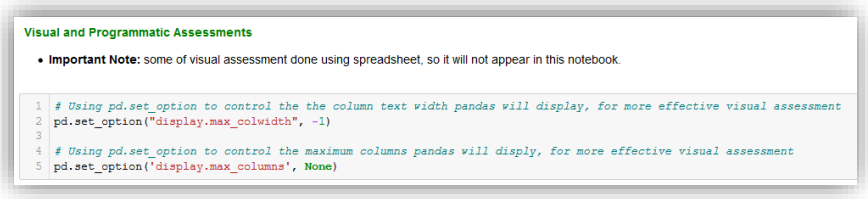


*Figure 9: **Pandas Visual Assessment Setting***

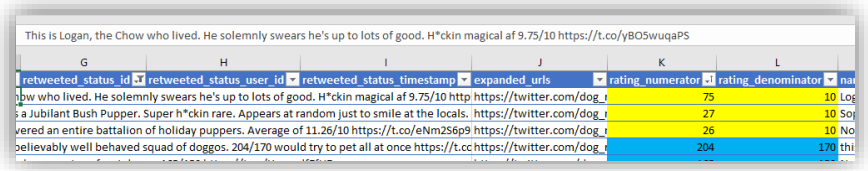


*Figure 10: **Visual Assessment using Spreadsheets***

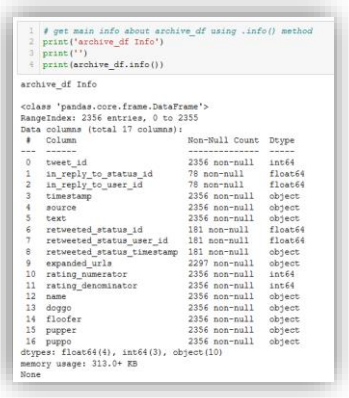


*Figure 11: **Programmatic Assessment example 1***

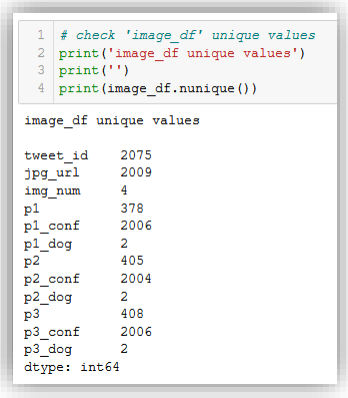


*Figure 12: **Programmatic Assessment example 2***

Back to Contents

3. Data Cleaning and Storing

Data cleaning process went through 3 sections:

1. Cleaning Preparation.

- The only and most important task in the preparation for data cleaning is to "**Making a a copy from the 3 iput data files**".

Section 1: Cleaning Preparation

Making a copy from the 3 dfs, to keep the original ones AS IS.

```
archive_df_clean = archive_df.copy()
image_df_clean = image_df.copy()
api_df_clean = api_df.copy()
```

Test the copied _clean dfs

```
# Using pd.reset_option to back to default column text width pandas will display
pd.reset_option("display.max_colwidth")

# Using pd.reset_option to back to default maximum columns pandas will disply
pd.reset_option('display.max_columns')
```

Figure 13: ***Making a copy from all tables before strat cleaning***

```
archive_df_clean.sample(1)
```

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_stat
74	878316110768087041	NaN	NaN	2017-06-23 18:17:33 +0000	<a href="http://twitter.com/download/iphone" r...	RT @dog_rates: Meet Terrance. He's being yelle...	6.69000

```
image_df_clean.sample(1)
```

	tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog
1017	709918798883774466	https://pbs.twimg.com/media/CdojYQmW8AApv4h.jpg	2	Pembroke	0.956222	True	Cardigan	0.020727	True

```
api_df_clean.sample(1)
```

	tweet_id	favorite_count	retweet_count	followers_count	friends_count
1850	813157409116065792	7768	2200	8869473	19

Figure 14: ***Testing Tables' copies***

2. Cleaning Process

- **We grouped any data issues that required the same cleaning efforts code in one group.**
- **Below table demonstrates all the details and the cleaning logic for:**
  - **(22)** Data quality issues.
  - **(6)** Tidiness issues.
  - **(2)** Feature Engineering actions.
- **Assessing and Cleaning Efforts Result Table (Page No. 11)**

Group 1: "No Action Needed" Group					A various issues that wouldn't affect our analysis	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic and Steps**
1	**1.1**	**twitter_archive_enhanced**	**Quality**	**Validity**	retweeted_status_timestamp column's values in archive_df table contains extra '**+0000**'.	All of Group 1 data issues doesn't affect our analysis scope, so it was left as it is without any CLEANING action from our side
2	**1.2**	**twitter_archive_enhanced**	**Quality**	**Accuracy**	**['name']** column's values in archive_df table contains inaccurate pet names like the letter "a" and "an".	
3	**1.3**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**in_reply_to_status_id** column's **Dtype** in archive_df table is **float64** while **no calcus** will be made, so it's better to be **str**.	
4	**1.4**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**in_reply_to_user_id** column's **Dtype** in archive_df table is **float64** while **no calcus** will be made, so it's better to be **str**.	
5	**1.5**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**retweeted_status_id** column's **Dtype** in archive_df table is **float64** while **no calcus** will be made, so it's better to be **str**.	
6	**1.6**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**retweeted_status_user_id** column's **Dtype** in archive_df table is **float64** while **no calcus** will be made, so it's better to be **str**.	
7	**1.7**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**retweeted_status_timestamp** column's **Dtype** in archive_df table is **object (str)** while it's supposed to be **datetime**.	
8	**1.8**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**['name']** column in archive_df table contains **Inconsistent values** regarding representation of **null** values, as **"None" strings.**	
Group 2: "Getting Original Tweets with Images" Group					**Cleaning all of the below data issues will help us getting original tweets with image only**	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
9	**2.1**	**twitter_archive_enhanced**	**Quality**	**Validity**	There **are Retweets and replies** column's and rows in archive_df and that doesn't **conform to analysis scope schem**a, so they needs to be **dropped**.	▪ **Use** the image_prediction table to guide the selection and removal of tweets without photos in the archive table. ▪ **Filter** archive_df to contain only (Original Tweets) that has (image). ▪ **Drop** the records in archive_df and that doesn't has expanded_urls. ▪ **Drop** Retweets and replies columns in archive_df.
10	**2.2**	**twitter_archive_enhanced**	**Quality**	**Validity**	There are some **records** in archive_df and that doesn't has **expanded_urls**, which means **no image exist**, so they needs to be **dropped**.	

Group 3: "Dogtionary Cleaning" Group					All of the below data issues relating to dog_stage name based on dogtionary items	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
11	**3.1**	**twitter_archive_enhanced**	**Tidy**	**Column headers are values, not variable names**	In archive_df table **Column headers (['doggo'] , ['floofer'] , ['pupper'] , ['puppo']) are values, not a variable names.** That resulting 1 variable ['dog_stage'] is stored 4 columns in a messy way.	▪ **Concatenate** the 4 ['doggo', 'puppo', 'pupper','floofer'] columns in one new column inarchive_df. ▪ **Replace** 'None' values with an empty str '' in the new column. ▪ **Replace** the empty str '' values with NaN values in the new column. ▪ **Separate** any value contains more than 1 stage by a dash (-) sign, using replace method. ▪ **Change** the columns Dtype from str to category using .astype() method.
12	**3.2**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**(['doggo'] , ['floofer'] , ['pupper'] , ['puppo'])** columns' in **archive_df** table contains **Inconsistent values** regarding representation of **null** values, as **"None" strings** in the (doggo, floofer, pupper, puppo) columns.	
13	**3.3**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**(['doggo'] , ['floofer'] , ['pupper'] , ['puppo'])** columns' **Dtype** in **archive_df** table are **object (str)** while it supposed to be **category** dtype.	
Group 4: "Dog Rating" Group					**All of the below data cleaning issues are relating to dog rating columns ['rating_numerator'] and ['rating_denominator']**	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
14	**4.1**	**twitter_archive_enhanced**	**Quality**	**Accuracy**	About **11 records** in **rating_numerator** and **rating_denominator** columns' **values** in **archive_df** table are **aggregated and featuring many dogs** based on **dog counts** in the picture, while it's supposed to be for **1 dog**. **That clearly appears in [['rating_numerator'] > 40]**	▪ Using regex, **extract** the right dog rating (rating_numerator and rating_denominator) as could as possible from text column and put the result in new column ['temp_rating']. ▪ **Test** the result randomly and also specifically using above known tweet_ids. **The 3 and 5 invalid records** ▪ **Replace** any still-wrong value programmatically or manually, according to the number of occurrence.
15	**4.2**	**twitter_archive_enhanced**	**Quality**	**Accuracy**	About **5 records** in **rating_numerator** and **rating_denominator** columns' **values** in **archive_df** table are **inaccurate** -mostly during text extraction, 1st digits occurrence took instead of 2nd one- and needed to be **corrected**, **That clearly appears in [['tweet_id'] == 740373189193256964, 722974582966214656, 716439118184652801,**	

					682962037429899265, 666287406224695296]]	**1 invalid record** ▪ **Drop** the record which contains wrong value 24/7 and has (No rating at text column). **11 aggregated dog rating records** ▪ **Split** ['temp_rating'] column into 2 new columns using "/" delimiter, to create new_numerator and new_denominator columns. ▪ **Convert** Dtypes for the new 2 columns to float. ▪ **Getting** dogs_count by devide archive_df_clean.new_numerator >= 40]/10. ▪ **Assign** the value 10 for new_denominator column. ▪ **Create** the FINAL new rating column with a name 'dog_rating' using calcus (new_numerator/new_denominator)*100, rounding the result to nearest decimal 1 point.
16	**4.3**	**twitter_archive_enhanced**	**Quality**	**Accuracy**	about **3 records** in **rating_numerator** column's **values** in **archive_df** table is **inaccurate** -mostly during text extraction, after decimal point value taken instead of the whole value- and needed to be **corrected**, **That clearly appears in [['tweet_id'] == 786709082849828864, 778027034220126208, 680494726643068929]]**	
17	**4.4**	**twitter_archive_enhanced**	**Quality**	**Accuracy**	**1 records** in **rating_numerator** and **rating_denominator** columns' **values** in **archive_df** table are **inaccurate** -mostly during text extraction, 1st digits occurrence while it is not a rating- and needed to be **totally removed**, **That clearly appears in [['tweet_id'] == 810985000000000000]]**	
18	**4.5**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**rating_numerator** column's **Dtypes** in **archive_df** tables is **int64** while -originally- it has **decimal values** at text column, so it should be **float**.	
Group 5: "DateTime Issue" Group					**All of the below codes relating to DateTime issue**	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic and Steps**
19	**5.1**	**twitter_archive_enhanced**	**Tidy**	**Multiple variables are stored in one column**	**timestamp** column's **values** in **archive_df** table should be splitted into **Date and Time** 2 column's.	▪ **Convert** timestamp Dtype from str to datetime using pd.to_datetime(). ▪ **Create** a new column ['date'] contains date only. ▪ **Create** a new column ['time'] contains time only.
20	**5.2**	**twitter_archive_enhanced**	**Quality**	**Validity**	**timestamp** column's **values** in **archive_df** table contains extra **'+0000'** needs to be **removed.**	
21	**5.3**	**twitter_archive_enhanced**	**Quality**	**Consistency**	**timestamp** column's **Dtype** in **archive_df** table is **object (str)** while it's supposed to be **datetime**.	

Group 6: "Feature Engineering" Group					All of the below codes relating to finding new valuable info in the dataset, for augmenting EDA.	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
22	**6.1**	**twitter_archive_enhanced**	**Feature Engineering**	**Quality-Validity**	**Getting source App/Device** from source column in **archive_df**.	▪ **Extract** device/app from source column using regex could be useful for EDA. ▪ **Create** a new column ['hour'] contains the hour only.
23	**6.2**	**twitter_archive_enhanced**	**Feature Engineering**	**Quality-completeness**	**Create ['hour'] column** in **archive_df**.	
Group 7: "Dropping Extraneous columns" Group					**Just dropping all not-needed columns in twitter_archive_enhanced.csv**	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
24	**7.1**	**twitter_archive_enhanced**	**Quality**	**Validity**	**Many columns in archive_df doesn't conform analysis scope schema.**	▪ Drop all extraneous (Not-Needed) columns in archive_df_clean using this code **archive_df_clean = archive_df_clean.drop(['rating_numerator', 'rating_denominator', 'doggo', 'floofer', 'puppe.**
Group 8: "Data Type Issues" Group					**All of the below codes relating to datatype issues**	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
25	**8.1**	**twitter_archive_enhanced, imagee_prediction.csv & tweet_json.tx (API file)**	**Quality**	**Consistency**	**tweet_id** columns' **Dtypes** in **archive_df, image_df**, and **api_df** tables are **int64** while **no calcus** will be made, so it's better to be **str**.	▪ **Convert** tweet_id columns' Dtypes in archive_df, image_df, and api_df tables str, using .astype() method.

Group 9: "image.csv Tidy & Quality Issues" Group					All of the below codes relating to image_df Tidy & Quality issues	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
26	**9.1**	**imagee_prediction.csv**	**Tidy**	**Column headers are values, not variable names**	In **image_df** table **Column headers (['p1'] , ['p2'] , ['p3']) are values, not a variable names. That resulting 2 variables ['prediction_number' and 'prediction_result'] are stored 3 columns in a messy way.**	1. **Create 3 TEMP copies of image_df_clean.** ▪ Copy#1 **image_df_clean_1:** will be used to **melt p1, p2, and p3 columns** ▪ Copy#2 **image_df_clean_2:** will be used to **melt p1_conf, p2_conf, and p3_conf columns** ▪ Copy#3 **image_df_clean_3:** will be used to **melt p1_dog, p2_dog, and p3_dog columns** 2. **Melt** each copy based on above mentioned columns. 3. **Drop** any Extraneous columns may affect our target shape. 4. **Merge** the 3 copies into **2 new dfs**, as below: ▪ df#1 **image_clean_temp1:** pd.merge(**image_df_clean_2, image_df_clean_3**, on='tweet_id'). ▪ df#1 **image_clean_temp2:** pd.merge(**image_df_clean_1, image_df_clean_3**, on='tweet_id') 5. **Drop Duplicates** on both 2 dfs **image_clean_temp1** and **image_clean_temp2** 6. **Replace** the value **'p*_conf'** with **'p*'** in prediction_number_confidence column at **image_clean_temp1** 7. **Create new ['key']** columns by **concatenating** 'tweet_id' and 'prediction_number_confidence' in at **image_clean_temp1**, and concatenate 'tweet_id' and 'prediction_number' in at **image_clean_temp2** 8. **Merge** the 2 dfs **image_clean_temp1** and **image_clean_temp2** on **['key']** column, into new df called **image_df_clean_melted** 9. **Drop** any Extraneous columns at **image_df_clean_melted**. 10. **Rename image_df_clean_melted** columns based on mentioned name at **Target Shape**. 11. **Create** a new column **prediction_match_breed_bin** with values (0 and 1) for more **EDA insights**.
27	**9.2**	**imagee_prediction.csv**	**Tidy**	**Column headers are values, not variable names**	In **image_df** table **Column headers (['p1_conf'] , ['p2_conf'] , ['p3_conf']) are values, not a variable names. That resulting 2 variables ['prediction_number' and 'prediction_confident'] are stored 3 columns in a messy way**	
28	**9.3**	**imagee_prediction.csv**	**Tidy**	**Column headers are values, not variable names**	In **image_df** table **Column headers (['p1_dog'] , ['p2_dog'] , ['p3_dog']) are values, not a variable names. That resulting 2 variables ['prediction_number' and 'prediction_validity'] are stored 3 columns in a messy way**	
29	**9.4**	**imagee_prediction.csv**	**Quality**	**Validity**	The **p** column's **(['p1], ['p1_conf'], ['p1_dog'] , etc.])** in **image_df** table has **'Non-descriptive columns' names** needs to be **adjusted.**	

Group 10: "archive.csv and api.csv Tidy Issues" Group					All of the below codes relating to archive_df and api_df Tidy Issues	
S#	**Index**	**Table**	**Issue Category**	**Issue Dimension**	**Issue Description**	**Cleaning Logic** and Steps
30	**10.1**	**twitter_archive_enhanced & tweet_json.tx (API file)**	**Tidy**	**A single observational unit is stored in multiple tables**	**api_df** table **should be a part of archive_df** table.	▪ **Merge** archive_df and api_df into 1 df **'twitter_archive_master'**

Back to Contents

3. Cleaning Result Storing

- Cleaning efforts results **2** output tables:

  Output_File #1: twitter_archive_master.csv
  Output_File #2: imagee_prediction.csv

- **Both were stored locally as csv files.**

twitter_archive_master.csv

```
1 twitter_archive_master.sample()
```

	tweet_id	timestamp	source	text	expanded_urls	name	dog_stage	dog_rating	date	time	hour	favorite_count	retweet_c
824	733828123016450049	2016-05-21 01:13:53+00:00	Twitter for iPhone	This is Terry. The harder you hug him the fart...	https://twitter.com/dog_rates/status/733828123...	Terry	NaN	100.0	2016-05-21	01:13:53	1	3489	

```
1 twitter_archive_master.to_csv('twitter_archive_master.csv', index=False)
```

*Figure 15: **twitter_archive_master.csv***

image_prediction.csv

```
1 image_prediction = image_df_clean_melted.copy()
```

```
1 image_prediction.sample()
```

	tweet_id	jpg_url	prediction_number	prediction_result	prediction_confidence	prediction_match_breed	prediction_match_breed
1785	679511351870550016	https://pbs.twimg.com/media/CW4b-GUWYAAa8QO.jpg	p1	Chihuahua	0.761972	True	

```
1 image_prediction.to_csv('image_prediction.csv', index=False)
```

*Figure 16: **imagee_prediction.csv***

4. Data Analyzing and Visualizing

Data Analyzing and Visualizing process will go through 2 sections:

1. Exploratory Data Analysis (EDA).

- Using python libraires (**Pandas, Numpy, Matplotlip, and Seaborn)**, we performed a **descriptive statistics,** trying to discover pattern in data and finding any valuable insights may augmenting our research questions.

Section 1: Exploratry Data Analysis (EDA)

In [131]:
```
1 twitter_archive_master.describe()
```

Out[131]:

	dog_rating	hour	favorite_count	retweet_count	followers_count	friends_count
count	1963.000000	1963.000000	1963.000000	1963.000000	1.963000e+03	1963.0
mean	116.499134	9.725420	8167.989302	2412.443199	8.869482e+06	19.0
std	409.574803	8.624072	12011.852740	4309.922747	6.027120e+01	0.0
min	0.000000	0.000000	70.000000	11.000000	8.869464e+06	19.0
25%	100.000000	1.000000	1750.000000	538.000000	8.869474e+06	19.0
50%	110.000000	4.000000	3684.000000	1167.000000	8.869479e+06	19.0
75%	120.000000	18.000000	10198.500000	2758.000000	8.869480e+06	19.0
max	17760.000000	23.000000	153082.000000	75802.000000	8.870127e+06	19.0

In [132]:
```
1 image_prediction.describe()
```

Out[132]:

	prediction_confidence	prediction_match_breed_bin
count	6.225000e+03	6225.000000
mean	2.631537e-01	0.738313
std	2.908324e-01	0.439588
min	1.740170e-10	0.000000
25%	5.123350e-02	0.000000
50%	1.351790e-01	1.000000
75%	3.796240e-01	1.000000
max	1.000000e+00	1.000000

Figure 17: ***EDA using Descriptive Statistics***

```
1 # Subplot grid for plotting pairwise relationships at twitter_archive_master dataset
2 g = sns.pairplot(twitter_archive_master)
```

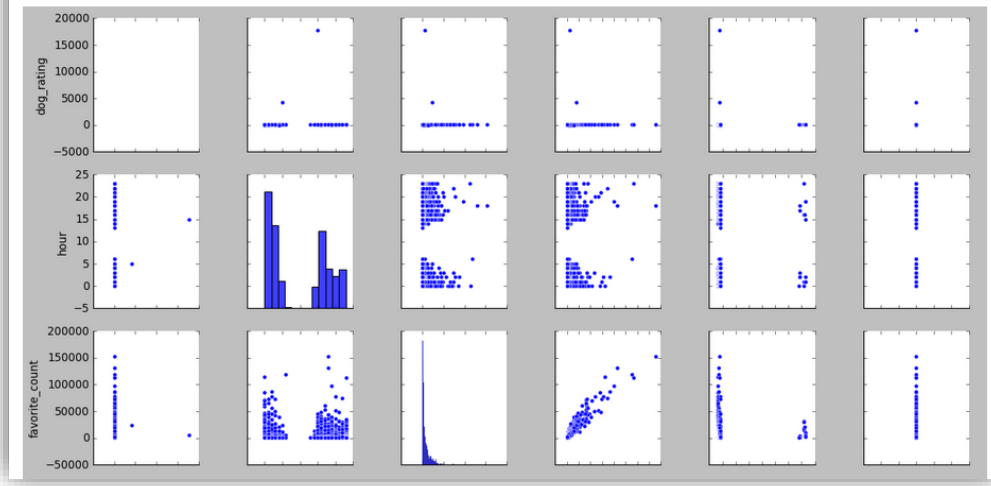


Figure 18: ***EDA using Visuals***

Back to Contents

2. Research Questions & Conclusion.

- **Finally**, we could **answer our research questuion** and made a **conclusion**.

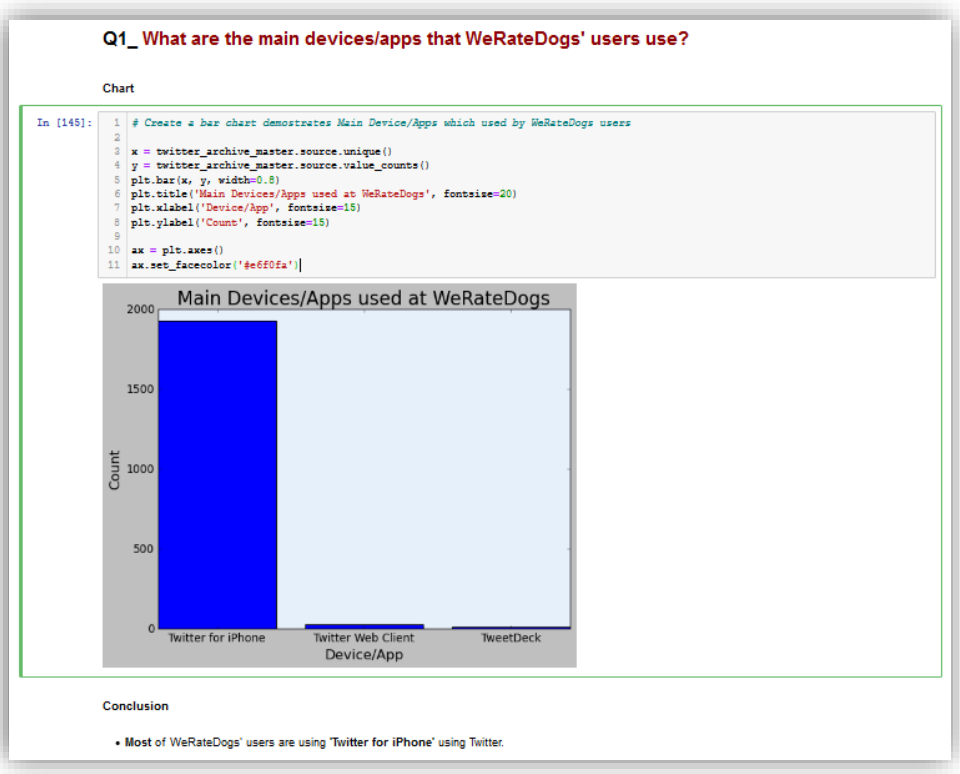


Figure 19: ***Example 1 of Research Question answering***

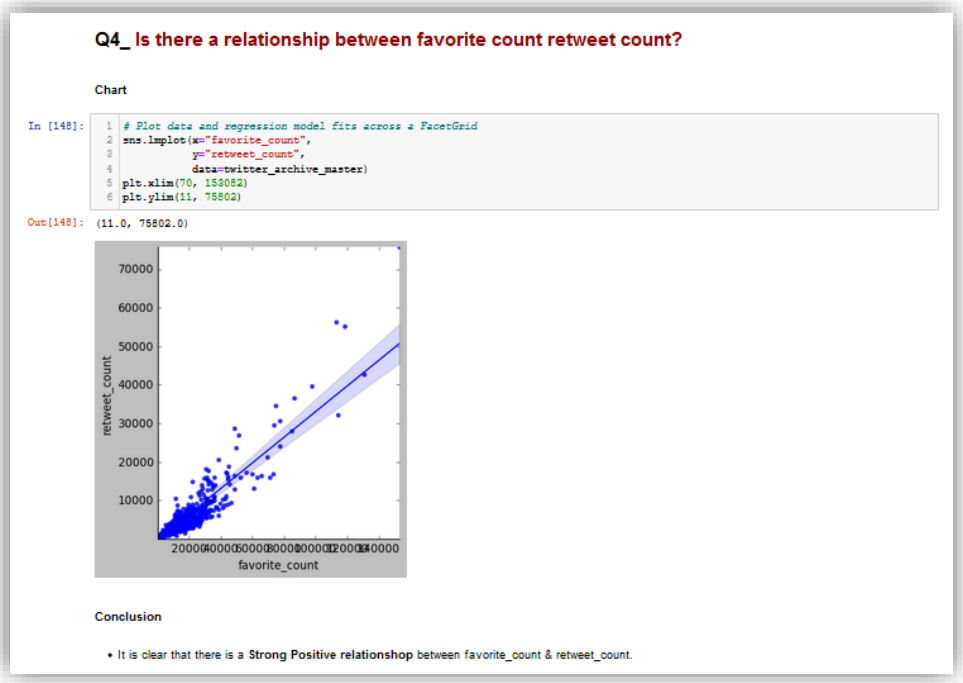


Figure 20: ***Example 2 of Research Question answering***

Back to Contents

5. Making Reports

Our study for "WeRateDogs" Twitter account ends with 2 reports:

1. **Effort Report:** ***"Wrangle_act Report"*** **– The one you're reading now.**

- **'wrangle_report.pdf'** demostrates all **"Data Analysis Efforts"** for "WERATEDOGS" Twitter account.

2. **Insights Report:** ***"act_report"***

- It demostrate the Analysis Conclusion. ***For the insights Full report, please check 'act_report.pdf'***.

--- End of Report ---