Data Wrangling

Summary

The aim is to gather three datasets related to WeRatedogs Twitter account, each of them has unique data and complement one another. Then, a data wrangling process shall be implemented through its three key steps “Gather, Assess & Clean” to best transform the data into a tidy, clean, and high-quality format as far as possible to be prepared for the next step which is “Data Analysis”. We shall delve deeper into each step to demonstrate the actions which had been followed.

Data Wrangling Steps:

- **Gather**

There are three datasets were gathered from three different sources, the first one is “twitter_archive_enhanced.csv” which is downloadable and then got uploaded to the notebook to be assigned to df_archive using pd.read_csv method.

```
#Import the first data package we do have
df_archive = pd.read_csv('twitter-archive-enhanced.csv')
```

The other one is “tweet_json.txt”, Twitter API was queried for each tweet’s JSON data using Python’s Tweepy Library. Tweets were loaded and read line by line using json.loads(line) method and extracted the retweet count and favorite count depending the tweet id of archive table. It was appended to an empty list and assigned to df_api.

```
df_list = []
with open ('tweet_json.txt', 'r') as file:
    for line in file:
        response = json.loads(line)
        tweet_id = response["id"]
        retweet_count = response["retweet_count"]
        favorite_count = response["favorite_count"]
        df_list.append({"tweet_id":tweet_id, "retweet_count": retweet_count,
                        "favorite_count":favorite_count})

df_api= pd.DataFrame(df_list, columns = ["tweet_id", "retweet_count", "favorite_count"])
```

Finally, the third dataset “image_predictions.tsv” was downloaded programmatically from URL using requests.get(url) library and written to folder directory, then assigned to df_prediction.

```
#Import Image predictions Data "third data package we do have"
url ='https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
file_name = url.split('/')[-1]
response = requests.get(url)
if not os.path.isfile(file_name):
    with open (os.path.join(url.split('/')[-1]), mode='wb') as file:
        file.write(response.content)
```

- **Assess**

Data assessment was done using many ways to figure out the issues beyond each dataset like the basic ones df.shape, df.info(), df.describe(), etc. Besides more sophisticated ones to define the odds might be found in each column and the issues found were of quality and tidiness in varying degrees.

On one hand, the quality issues are like inaccurate and invalid records of rating numerator and denominator,

```
df_archive.query('rating_numerator > 20' or df_archive.query('rating_numerator < 10' or
                                                              df_archive.query('rating_denominator != 10')))
```

				Numerator	Denominator	Dog stages and Name				
NaN	NaN	https://twitter.com/dog_rates/status/716439118...		50	50	Bluebert	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/713900603...		99	90	None	None	None	None	None

unwanted entries in different forms like retweets and replies as they don't match the criteria,

source	text	retweeted_status_id	retweeted_status_user_id	retweeted_status_timestamp
<a nload/iphone" r...	RT @dog_rates: Meet Lola. Her hobbies include ...	8.352641e+17	4.196984e+09	2017-02-24 23:04:14 +0000

invalid names were extracted like "a", "this", "actually" in addition to the empty cells which were represented as string "None" not null values.

```
lowercase_names= df_archive.name.str.contains('^[a-z]', regex=True)
df_archive[lowercase_names].name.unique()
```

```
array(['such', 'a', 'quite', 'not', 'one', 'incredibly', 'mad', 'an',
       'very', 'just', 'my', 'his', 'actually', 'getting', 'this',
       'unacceptable', 'all', 'old', 'infuriating', 'the', 'by',
       'officially', 'life', 'light', 'space'], dtype=object)
```

Additionally, the most common quality issue among the datasets is the columns in different format other than they should be like tweet_id and timestamp.

```
df_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    2356 non-null   int64
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object
```

On the other hand, the main tidiness issues such dog stage columns to be represented into one column rather than four ones,

doggo	floofer	pupper	puppo
None	floofer	None	None
None	None	None	puppo

then the archive and api tables to be as one dataset as they complement each other. Moreover, dog breed prediction columns to be melted into only three columns rather than where maximum confidence prediction value per each tweet alongside its corresponding dog breed to be easily extracted and combined one dataset with the archive

p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	p3_conf	p3_dog
Pembroke	0.511319	True	Cardigan	0.451038	True	Chihuahua	0.029248	True
Irish_terrier	0.487574	True	Irish_setter	0.193054	True	Chesapeake_Bay_retriever	0.118184	True
Pomeranian	0.566142	True	Eskimo_dog	0.178406	True	Pembroke	0.076507	True
Appenzeller	0.341703	True	Border_collie	0.199287	True	ice_lolly	0.193548	False

- **Clean**

Speaking of the key step where all of the observations had been made, should be in effect. After deep investigation of the inaccurate and invalid records of numerators and denominators. Some of them got fixed manually and others were to be dropped.

```
#Manual fix for these indexes where their rating are already existing
archive_clean.at[1662, 'rating_numerator'] = 10
archive_clean.at[1202, 'rating_numerator'] = 11
archive_clean.at[1165, 'rating_numerator'] = 13
archive_clean.at[1068, 'rating_numerator'] = 14
archive_clean.at[695, 'rating_numerator'] = 9.75
archive_clean.at[763, 'rating_numerator'] = 11.27
archive_clean.at[2335, 'rating_numerator'] = 9

archive_clean.at[1662, 'rating_denominator'] = 10
archive_clean.at[1202, 'rating_denominator'] = 10
archive_clean.at[1165, 'rating_denominator'] = 10
archive_clean.at[1068, 'rating_denominator'] = 10
archive_clean.at[695, 'rating_denominator'] = 10
archive_clean.at[763, 'rating_denominator'] = 10
archive_clean.at[2335, 'rating_denominator'] = 10
```

```
#Drop the rest which have no ratings in the text besides having ratings far from the unique rating system
archive_clean =archive_clean[archive_clean['rating_denominator'] == 10]

#Drop the numerators which are greater than 20 as they don't make any sense
archive_clean =archive_clean[archive_clean['rating_numerator'] < 20]
```

As well as for the unwanted entries in different forms like retweets and replies were dropped.

```
#Filter oringinal tweets only with no retweets
archive_clean = archive_clean[~(archive_clean.in_reply_to_status_id.notnull())]
```

```
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    2097 non-null   int64
 1   in_reply_to_status_id       0 non-null      float64
 2   in_reply_to_user_id         0 non-null      float64
 3   timestamp                   2097 non-null   object
 4   source                      2097 non-null   object
 5   text                        2097 non-null   object
 6   retweeted_status_id         0 non-null      float64
 7   retweeted_status_user_id    0 non-null      float64
 8   retweeted_status_timestamp  0 non-null      object
```

And the invalid names in addition to the empty cells which were represented as string “None” not null values were also replaced.

```
#Names seem to start with capital letter, so such small letter entries aren't names
invalid_names= archive_clean.name.str.contains('^[a-z]', regex=True)
archive_clean[invalid_names].name.unique()
```

```
array(['such', 'a', 'quite', 'not', 'one', 'incredibly', 'very', 'my',
       'his', 'an', 'actually', 'just', 'getting', 'mad', 'unacceptable',
       'all', 'old', 'infuriating', 'the', 'by', 'officially', 'life',
       'light', 'space'], dtype=object)
```

```
archive_clean.loc[invalid_names, 'name'] = np.nan
archive_clean['name'] = archive_clean['name'].replace(to_replace=["None"],
                                                      value=[np.nan])
```

Besides the erroneous datatype which had been converted to proper dtype.

```
#Convert timestamp column to date time
archive_clean ['timestamp'] = pd.to_datetime(df_archive ['timestamp'])

#Convert tweet id column to date time
archive_clean ['tweet_id'] = df_archive['tweet_id'].astype(str)
```

While the tidiness issues had been handled appropriately like in dog stage columns in archives table

```
#Replace all of the None entries in the mentioned four columns with nothing:
dog_stage_list = ['doggo', 'floofer', 'pupper', 'puppo']

for c in dog_stage_list:
    archive_clean[c] = archive_clean[c].apply(lambda x: x.replace("None", ""))
```

```
#Augment the four column together for creating one columns (melt method concept)
archive_clean['dog_stage'] = archive_clean['doggo'] + archive_clean['floofer'] +archive_clean['pupper'] + archive_clean['puppo']
archive_clean.drop(['doggo', 'floofer', 'pupper', 'puppo'], axis=1, inplace= True)
```

```
#Replace and clean the entries to be easily read
archive_clean['dog_stage'] = archive_clean['dog_stage'].replace(to_replace=["", "doggopuppo", "doggofloofer","doggopupper"],
                                                value=[np.nan,"doggo-puppo", "doggo-floofer","doggo-pupper"])
```

Then, the archive and api tables was combined into one dataset using the pd.merge method on the common tweet_id column.

```
archive_clean = pd.merge(archive_clean, api_clean, on= "tweet_id", how= 'left')
```

	tweet_id	timestamp	rating	retweet_count	favorite_count
1035	704347321748819968	2016-02-29 16:47:42+00:00	10	393	1729

Furthermore, prediction table was transformed from wide to long shape, we merged the nine columns into three ones using the below code

```
p_list = ["p1", "p2", "p3" ]
p_conf_list = ["p1_conf", "p2_conf", "p3_conf"]
p_dog_list = ["p1_dog", "p2_dog", "p3_dog"]

prediction_clean = prediction_clean.melt(id_vars=["tweet_id","jpg_url", "img_num", "p1_conf", "p1_dog", "p2_conf", "p2_dog",
                                    "p3_conf", "p3_dog"],
                                value_vars= p_list, var_name = "prediction", value_name="dog_breed")

prediction_clean = prediction_clean.melt(id_vars=["tweet_id","jpg_url", "img_num", "prediction", "dog_breed", "p1_dog", "p2_dog"
                                    , "p3_dog"],
                                value_vars= p_conf_list, var_name = "Var2" ,value_name="prediction_confidence")

prediction_clean = prediction_clean.melt(id_vars=["tweet_id","jpg_url", "img_num", "prediction", "dog_breed", "Var2","prediction_
                                value_vars= p_dog_list, var_name = "Var3" ,value_name="validity")

#Removing all of unneeded rows and columns
prediction_clean = prediction_clean[(prediction_clean['prediction']== prediction_clean['Var2'].str[:2]) & (prediction_clean['Var2
             & (prediction_clean['prediction']== prediction_clean['Var3'].str[:2])]

prediction_clean.drop(['Var2', 'Var3'], axis =1, inplace=True)

#Resetting index to the normal way
prediction_clean = prediction_clean.reset_index(drop=True)
```

and the result

	tweet_id	jpg_url	img_num	prediction	dog_breed	prediction_confidence	validity
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	p1	Welsh_springer_spaniel	0.465074	True
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	p1	redbone	0.506826	True

Then, we got the maximum prediction confidence per each tweet id as well as the dog breed from the prediction table and prepared to merge.

The product before merge,

```
bestofbest= prediction_clean.groupby(['tweet_id'])['prediction_confidence'].max()

df_bestofbest = pd.DataFrame(bestofbest, columns = ["dog_breed", "prediction_confidence"])

df_bestofbest = pd.merge(df_bestofbest, prediction_clean, on = ['tweet_id', 'prediction_confidence'],
                         how = 'inner')

df_bestofbest.drop(["dog_breed_x", "jpg_url", "img_num","prediction", "validity"], axis= 1, inplace=True)

df_bestofbest = df_bestofbest.rename(columns={"dog_breed_y":"dog_breed"})
```

```
df_bestofbest.head()
```

	tweet_id	prediction_confidence	dog_breed
0	666020888022790149	0.465074	welsh springer spaniel
1	666029285002620928	0.506826	redbone
2	666033412701032449	0.596461	german shepherd
3	666044226329800704	0.408143	rhodesian ridgeback
4	666049248165822465	0.560311	miniature pinscher

Then, the merge:

```
archive_clean = pd.merge(archive_clean, df_bestofbest, on= "tweet_id", how= 'inner')
```

	tweet_id	timestamp	rating	retweet_count	favorite_count	dog_breed	prediction_confidence	name	dog_stage
0	892420643555336193	2017-08-01 16:23:56+00:00	13	8853	39467	orange	0.097049	Phineas	NaN