Project: Wrangling and Analyze Data

Data Gathering

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import requests
import os
from PIL import Image
from io import BytesIO
import json

weratedogs_archive = pd.read_csv('twitter-archive-enhanced.csv')
```

In [3]:

```
#Creating a folder if it doesn't already exist

folder_name = 'image_prediction'
if not os.path.exists(folder_name):
    os.makedirs(folder_name)
```

In [4]:

```
#Programmatic download of the TSV file

url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predicti
ons/image-predictions.tsv'
response = requests.get(url)
```

In [5]:

```
with open(os.path.join(folder_name, url.split('/')[-1]), mode = 'wb') as file:
    file.write(response.content)
```

In [5]:

```
image_prediction = pd.read_csv('/home/workspace/image_prediction/image-predictions.tsv'
,sep='\t')
image_prediction
```

1. Use the Tweepy library to query additional data via the Twitter API (tweet_json.txt)

In [7]:

```
file1 = open('tweet-json.txt', 'r')
tweet_json = file1.readlines()
```

In [8]:

```
tweets_converted = []
for tweet in tweet_json:
    tweets_converted.append(json.loads(tweet))
```

In [9]:

```
#tweet ID, retweet count, and favorite count."
df_list = []
for tweet in tweets_converted:
        tweet_id = int(tweet['id_str'])
        retweets = tweet['retweet_count']
        number_of_likes = tweet['favorite_count']
        # Append to list of dictionaries
        df_list.append({'tweet_id': tweet_id,
                        'retweets': retweets,
                        'number_of_likes': number_of_likes})
```

In [6]:

```
# Create DataFrame from list of dictionaries
tweets = pd.DataFrame(df_list, columns = ['tweet_id', 'retweets', 'number_of_likes'])
tweets
```

Assessing Data

In [11]:

```
#Programmatic Assessment
weratedogs_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [12]:

```
#Programmatic Assessment
weratedogs_archive.duplicated().sum()
```

Out[12]:

```
0
```

In [13]:

```
#Programmatic Assessment
image_prediction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

In [14]:

```
#Programmatic Assessment
image_prediction.duplicated().sum()
```

Out[14]:

```
0
```

In [15]:

```
#Programmatic Assessment
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
tweet_id           2354 non-null int64
retweets           2354 non-null int64
number_of_likes    2354 non-null int64
dtypes: int64(3)
memory usage: 55.2 KB
```

In [16]:

```
tweets.duplicated().sum()
```

Out[16]:

```
0
```

Quality issues

`weratedogs_archive` columns:

Visual Assessment

1. **name**: this column shows some unrealistic name type like single letters(a, the), there are also missing names as well as duplicated names, some name are proper case while some are all lowercase as observed via visual assessment.

---

1. **retweeted_status_id | retweeted_status_user_id | retweeted_status_timestamp**: missing entries for the following variables/columns.

---

1. **doggo | floofer | pupper | puppo**: missing entries for the following columns.

---

1. **in_reply_to_status_id | in_reply_to_user_id**: Observance of missing values from these columns.

---

1. **floofer**: Column name error. Floofer is not a dog stage according to the Dogtionary.

Programmatic Assessment

1. **expanded_urls**: during programmatic assessment, some values appear to be missing from this column.

---

1. **timestamp**: data type in this column is wrong. Should be a datetime format data type.

---

1. **tweet_id**: data type in this column is not preffered. Since I won't be using the figures here to perform any calculations, it is best practice that it is coverted to a string.

Tidiness issues

`weratedogs_archive` columns:

1. **doggo | floofer | pupper | puppo**: The following columns violate the first rule of tidiness: that each variable forms a column. They all belong under one variable: stage.

---

1. **retweeted_status_id | retweeted_status_user_id | retweeted_status_timestamp** : The following columns are not needed for the current process as majority of the data within are missing.

Cleaning Data

In [17]:

```
# Make copies of original pieces of data
weratedogs_archive_clean = weratedogs_archive.copy()
tweets_clean = tweets.copy()
image_prediction_clean = image_prediction.copy()
```

Tidiness:

Define: I will be getting rid of the unwanted columns by using the `.drop()` method.

Code

In [18]:

```
#This code drops all columns that are either unwanted from the `weratedogs_archive_clea
n` dataframe

weratedogs_archive_clean = weratedogs_archive_clean.drop(['retweeted_status_id','retwee
ted_status_user_id','retweeted_status_timestamp',
                                                          'expanded_urls','floofer','in
_reply_to_status_id','in_reply_to_user_id' ], axis=1)
```

Test

In [19]:

```
#checking if all unwanted columns have been successfully removed from the 'weratedogs_archive_clean' dataframe
weratedogs_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 10 columns):
tweet_id              2356 non-null int64
timestamp             2356 non-null object
source                2356 non-null object
text                  2356 non-null object
rating_numerator      2356 non-null int64
rating_denominator    2356 non-null int64
name                  2356 non-null object
doggo                 2356 non-null object
pupper                2356 non-null object
puppo                 2356 non-null object
dtypes: int64(3), object(7)
memory usage: 184.1+ KB
```

Re-Gathering:

Define: I will be re-gathering the dog stages, `rating_numerator` , `rating_denominator` and `name` columns using the `.extract()` method.

Code

In [20]:

```
weratedogs_archive_clean['doggo'] = weratedogs_archive_clean.text.str.extract('(doggo)',expand = True)
weratedogs_archive_clean['pupper'] = weratedogs_archive_clean.text.str.extract('(pupper)',expand = True)
weratedogs_archive_clean['puppo'] = weratedogs_archive_clean.text.str.extract('(puppo)',expand = True)
weratedogs_archive_clean['blep'] = weratedogs_archive_clean.text.str.extract('(blep)',expand = True)
weratedogs_archive_clean['floof'] = weratedogs_archive_clean.text.str.extract('(floof)',expand = True)
weratedogs_archive_clean['snoot'] = weratedogs_archive_clean.text.str.extract('(snoot)',expand = True)
weratedogs_archive_clean['rating_numerator'] = weratedogs_archive_clean.text.str.extract('(\d+\S?\d+)(/\d+)',expand = True)
weratedogs_archive_clean['rating_denominator'] = weratedogs_archive_clean.text.str.extract('(/\d+)(\s+h?)',expand = True)

weratedogs_archive_clean['name'] = weratedogs_archive_clean.text.str.extract('(is\s[A-Z][a-z]+\.)',expand = True)
```

Test

In [1]:

```
weratedogs_archive_clean
```

Quality:

Define: Cleaning the data re-generated for the `name` and `rating_denominator` columns.

Code

In [23]:

```
#cleaning the data re-gathered and testing to see the results
weratedogs_archive_clean.name = weratedogs_archive_clean.name.str[3:-1]
weratedogs_archive_clean.rating_denominator = weratedogs_archive_clean.rating_denominat
or.str[1:]
```

Test

In [2]:

```
weratedogs_archive_clean
```

Quality:

Define: Merging all dataframes into one using the `.merge()` method on the `tweet_id` column.

Code

In [25]:

```
weratedogs_archive_clean = pd.merge(weratedogs_archive_clean,tweets_clean, on = 'tweet_
id', how = 'left')
```

In [26]:

```
weratedogs_archive_clean = pd.merge(weratedogs_archive_clean,image_prediction_clean, on
= 'tweet_id', how = 'left')
```

Test

In [27]:

```
#A check to see if the merge was successful
weratedogs_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2356 entries, 0 to 2355
Data columns (total 26 columns):
tweet_id                 2356 non-null int64
timestamp                2356 non-null object
source                   2356 non-null object
text                     2356 non-null object
rating_numerator         1927 non-null object
rating_denominator       2281 non-null object
name                     1127 non-null object
doggo                    98 non-null object
pupper                   272 non-null object
puppo                    37 non-null object
blep                     1 non-null object
floof                    23 non-null object
snoot                    0 non-null object
retweets                 2354 non-null float64
number_of_likes          2354 non-null float64
jpg_url                  2075 non-null object
img_num                  2075 non-null float64
p1                       2075 non-null object
p1_conf                  2075 non-null float64
p1_dog                   2075 non-null object
p2                       2075 non-null object
p2_conf                  2075 non-null float64
p2_dog                   2075 non-null object
p3                       2075 non-null object
p3_conf                  2075 non-null float64
p3_dog                   2075 non-null object
dtypes: float64(6), int64(1), object(19)
memory usage: 497.0+ KB
```

Tidiness:

Define: `doggo` , `floof` , `pupper` , `puppo` , `blep` and `snoot` - I will be collapsing these columns into a `stage` column using the `.melt()` method.

Code

In [28]:

```
#collapsing the stage variable columns to effect proper structure
weratedogs_archive_clean = pd.melt(weratedogs_archive_clean, id_vars=['tweet_id','times
tamp','source','rating_numerator','rating_denominator',
                                                                        'name','retweets'
,'number_of_likes','text',
                                                                        'jpg_url','img_nu
m','p1','p1_conf','p1_dog','p2','p2_conf','p2_dog','p3','p3_conf','p3_dog'],
                           var_name='header', value_name = 'stage')

weratedogs_archive_clean = weratedogs_archive_clean.drop('header', axis =1)
```

Test

In [3]:

```
#checking if code implementation was a success
weratedogs_archive_clean
```

Quality:

Define: I will be tackling the duplicated rows created by the `.melt()` method used in the previous cell using the

`.drop_duplicates()` and `.drop()` methods.

Code

In [30]:

```
weratedogs_archive_clean.drop_duplicates(inplace = True)
```

In [31]:

```
#querying the duplicated tweet_id with Null values and leaving those with valid entries
mask_null_stage = weratedogs_archive_clean[weratedogs_archive_clean.tweet_id.duplicated
(keep = False)]
rows_to_drop = list(mask_null_stage[mask_null_stage.stage.isna()].index)
weratedogs_archive_clean = weratedogs_archive_clean.drop(rows_to_drop)
```

Test

In [32]:

```
#this test shows that we still have some form of duplicates hiding in our dataset
weratedogs_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2368 entries, 0 to 11021
Data columns (total 21 columns):
tweet_id              2368 non-null int64
timestamp             2368 non-null object
source                2368 non-null object
rating_numerator      1938 non-null object
rating_denominator    2293 non-null object
name                  1129 non-null object
retweets              2366 non-null float64
number_of_likes       2366 non-null float64
text                  2368 non-null object
jpg_url               2087 non-null object
img_num               2087 non-null float64
p1                    2087 non-null object
p1_conf               2087 non-null float64
p1_dog                2087 non-null object
p2                    2087 non-null object
p2_conf               2087 non-null float64
p2_dog                2087 non-null object
p3                    2087 non-null object
p3_conf               2087 non-null float64
p3_dog                2087 non-null object
stage                 431 non-null object
dtypes: float64(6), int64(1), object(14)
memory usage: 407.0+ KB
```

Quality:

Define: I will be tackling the duplicated `tweet_id` values which exposed double entry in the `stage` column using the

`.duplicated().` and `.drop()` methods.

Code

In [7]:

```
#checking if we still have duplicated tweet_ids
weratedogs_archive_clean[weratedogs_archive_clean.tweet_id.duplicated(keep= False)]
```

In [34]:

```
#querying and deleting tweet_ids that have multiple stage entries as found in the cell
 above
double_stage_entry = list(weratedogs_archive_clean[weratedogs_archive_clean.tweet_id.du
plicated(keep= False)].index)
weratedogs_archive_clean = weratedogs_archive_clean.drop(double_stage_entry)
```

Test

In [35]:

```
weratedogs_archive_clean[weratedogs_archive_clean.tweet_id.duplicated(keep= False)]
```

Out[35]:

tweet_id	timestamp	source	rating_numerator	rating_denominator	name	retweets	numbe

0 rows × 21 columns

Quality:

Define: Fiding and removing rows with missing image URLs from `weratedogs_archive_clean` dataframe.

Code

In [36]:

```
missing_image_Urls = list(weratedogs_archive_clean[weratedogs_archive_clean.jpg_url.isn
a()].index)

weratedogs_archive_clean = weratedogs_archive_clean.drop(missing_image_Urls)
```

Test

In [37]:

```
weratedogs_archive_clean[weratedogs_archive_clean.jpg_url.isna()]
```

Out[37]:

tweet_id	timestamp	source	rating_numerator	rating_denominator	name	retweets	numbe

0 rows × 21 columns

Quality:

Define : Filling in the null values present in the `retweets` and `number_of_likes` columns using the `.fillna` method.

Code

In [38]:

```
#checking the rows with null values for retweets abd number_of_likes columns

weratedogs_archive_clean[weratedogs_archive_clean.number_of_likes.isna()]
```

Out[38]:

	tweet_id	timestamp	source	rating_numerator
19	888202515573088257	2017-07-21 01:02:36 +0000	<a href="http://twitter.com/download/iphone" r...	13
3171	771004394259247104	2016-08-31 15:19:06 +0000	<a href="http://twitter.com/download/iphone" r...	12

2 rows × 21 columns

In [39]:

```
#filling the missing values with the average number of retweets and likes in the datase
t

weratedogs_archive_clean['retweets'] = weratedogs_archive_clean['retweets'].fillna((wer
atedogs_archive_clean['retweets'].mean()))
weratedogs_archive_clean['number_of_likes'] = weratedogs_archive_clean['number_of_like
s'].fillna((weratedogs_archive_clean['number_of_likes'].mean()))
```

Test

In [40]:

```
#checking if there are any null values left in the 'retwweets' and 'number_of_likes' co
lumns
weratedogs_archive_clean[weratedogs_archive_clean.number_of_likes.isna()].size, werated
ogs_archive_clean[weratedogs_archive_clean.retweets.isna()].size
```

Out[40]:

```
(0, 0)
```

Quality:

Define : Converting all inappropriate data type to preffered data types.

Code

In [41]:

```
weratedogs_archive_clean.name = weratedogs_archive_clean.name.str.title()
weratedogs_archive_clean.p1_dog = weratedogs_archive_clean.p1_dog.astype(bool)
weratedogs_archive_clean.p2_dog = weratedogs_archive_clean.p2_dog.astype(bool)
weratedogs_archive_clean.p3_dog = weratedogs_archive_clean.p3_dog.astype(bool)
weratedogs_archive_clean.retweets= weratedogs_archive_clean.retweets.astype(int)
weratedogs_archive_clean.number_of_likes= weratedogs_archive_clean.number_of_likes.astype(int)
weratedogs_archive_clean.img_num=weratedogs_archive_clean.img_num.astype(int)
weratedogs_archive_clean.tweet_id = weratedogs_archive_clean.tweet_id.astype(str)
weratedogs_archive_clean.p1 = weratedogs_archive_clean.p1.str.title()
weratedogs_archive_clean.p2 = weratedogs_archive_clean.p2.str.title()
weratedogs_archive_clean.p3 = weratedogs_archive_clean.p3.str.title()
weratedogs_archive_clean.rating_numerator = weratedogs_archive_clean.rating_numerator.astype(float)
weratedogs_archive_clean.rating_denominator = weratedogs_archive_clean.rating_denominator.astype(float)
```

Test

In [42]:

```
weratedogs_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2063 entries, 0 to 11021
Data columns (total 21 columns):
tweet_id              2063 non-null object
timestamp             2063 non-null object
source                2063 non-null object
rating_numerator      1656 non-null float64
rating_denominator    2024 non-null float64
name                  1044 non-null object
retweets              2063 non-null int64
number_of_likes       2063 non-null int64
text                  2063 non-null object
jpg_url               2063 non-null object
img_num               2063 non-null int64
p1                    2063 non-null object
p1_conf               2063 non-null float64
p1_dog                2063 non-null bool
p2                    2063 non-null object
p2_conf               2063 non-null float64
p2_dog                2063 non-null bool
p3                    2063 non-null object
p3_conf               2063 non-null float64
p3_dog                2063 non-null bool
stage                 346 non-null object
dtypes: bool(3), float64(5), int64(3), object(10)
memory usage: 312.3+ KB
```

Quality:

Define : Filling in the null values present in the `rating_numerator` and `rating_denominator` columns using the `.fillna` method.

Code

In [8]:

```
#checking the rows with null values for rating_numerator column

weratedogs_archive_clean[weratedogs_archive_clean.rating_numerator.isna()]
```

In [9]:

```
#checking the rows with null values for rating_denominator column

weratedogs_archive_clean[weratedogs_archive_clean.rating_denominator.isna()]
```

In [45]:

```
#filling the missing values with the average number of rating_numerator and rating_denominator respectively

weratedogs_archive_clean['rating_numerator'] = weratedogs_archive_clean['rating_numerator'].fillna(weratedogs_archive_clean['rating_numerator'].mean())
weratedogs_archive_clean['rating_denominator'] = weratedogs_archive_clean['rating_denominator'].fillna(weratedogs_archive_clean['rating_denominator'].mean())
```

Test

In [46]:

```
weratedogs_archive_clean[weratedogs_archive_clean.rating_numerator.isna()].size,weratedogs_archive_clean[weratedogs_archive_clean.rating_denominator.isna()].size
```

Out[46]:

```
(0, 0)
```

Tidiness:

Define: Removing all rows with 'False' dog predictions by the neural network.

Code

In [47]:

```
false_prediction_rows = list(weratedogs_archive_clean.query("p1_dog == False").index)
```

In [48]:

```
weratedogs_archive_clean = weratedogs_archive_clean.drop(false_prediction_rows, axis=0)
```

Test

In [49]:

```
weratedogs_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1521 entries, 1 to 11021
Data columns (total 21 columns):
tweet_id              1521 non-null object
timestamp             1521 non-null object
source                1521 non-null object
rating_numerator      1521 non-null float64
rating_denominator    1521 non-null float64
name                  791 non-null object
retweets              1521 non-null int64
number_of_likes       1521 non-null int64
text                  1521 non-null object
jpg_url               1521 non-null object
img_num               1521 non-null int64
p1                    1521 non-null object
p1_conf               1521 non-null float64
p1_dog                1521 non-null bool
p2                    1521 non-null object
p2_conf               1521 non-null float64
p2_dog                1521 non-null bool
p3                    1521 non-null object
p3_conf               1521 non-null float64
p3_dog                1521 non-null bool
stage                 255 non-null object
dtypes: bool(3), float64(5), int64(3), object(10)
memory usage: 230.2+ KB
```

Storing Data

Save gathered, assessed, and cleaned master dataset to a CSV file named "twitter_archive_master.csv".

In [50]:

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

Analyzing and Visualizing Data

In this section, analyze and visualize your wrangled data. You must produce at least **three (3) insights and one (1) visualization.**

In [51]:

```
df = pd.read_csv('twitter_archive_master.csv')
```

In [10]:

```
#Top viral tweets from the dataframe
df.sort_values(by=['retweets'], ascending = False).head(12)
```

In [11]:

```
# Top 10 Tweets with the highest likes

df.sort_values(by=['number_of_likes'], ascending = False).head(10)
```

In [54]:

```
#The top five most frequent dog breed predicted by the neural network

df.p1.value_counts().head()
```

Out[54]:

```
Golden_Retriever      143
Labrador_Retriever     99
Pembroke               89
Chihuahua              83
Pug                    57
Name: p1, dtype: int64
```

Insights:

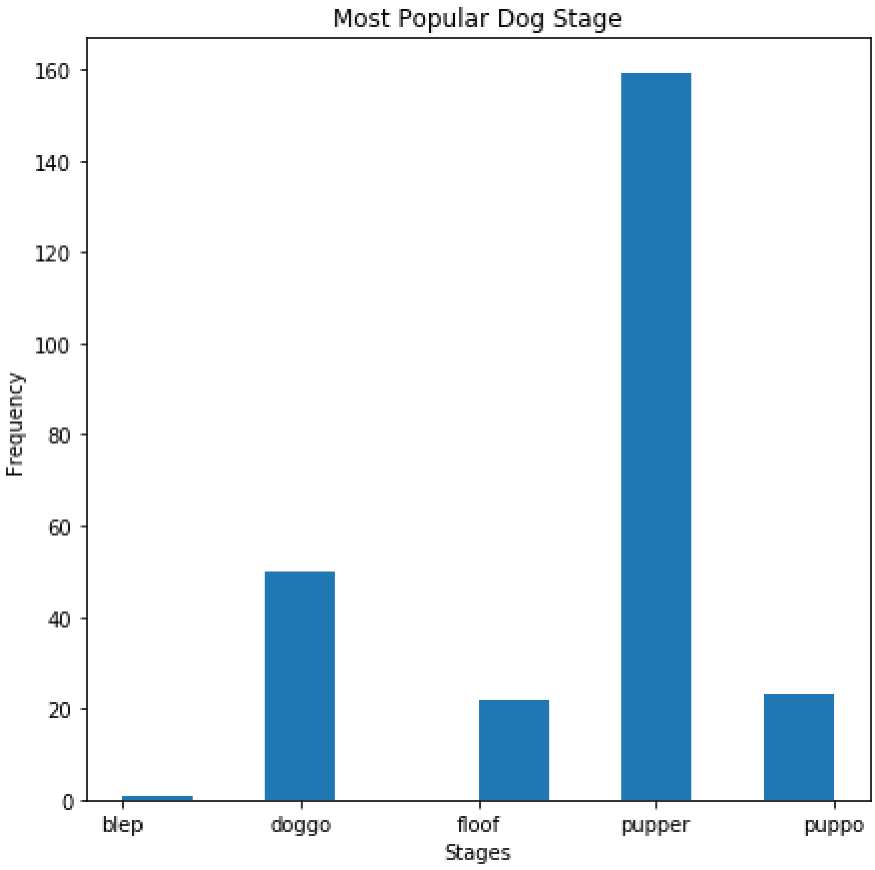
1. The top ten dog rating tweets with the highest retweets(coverage) in the dataset. On the top of the chart is a dog in its 'doggo' stage
2. The top ten most admired dog rating tweets. On the top of the chart is a dog in its 'puppo' stage
3. The top five most popular dog breeds in the neural network prediction.

Visualization

In [55]:

```
stages = df[~(df.stage.isna())]

plt.figure(figsize=(7,7))
plt.hist(stages.stage)
plt.title("Most Popular Dog Stage")
plt.ylabel('Frequency')
plt.xlabel('Stages');
```



In [ ]:

```

```