act_report

January 28, 2023

1 Project: Wrangling and Analyze Data

1.1 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.**

1.1.1 Proportions of tickets per prediction model

Function to plot pie graph

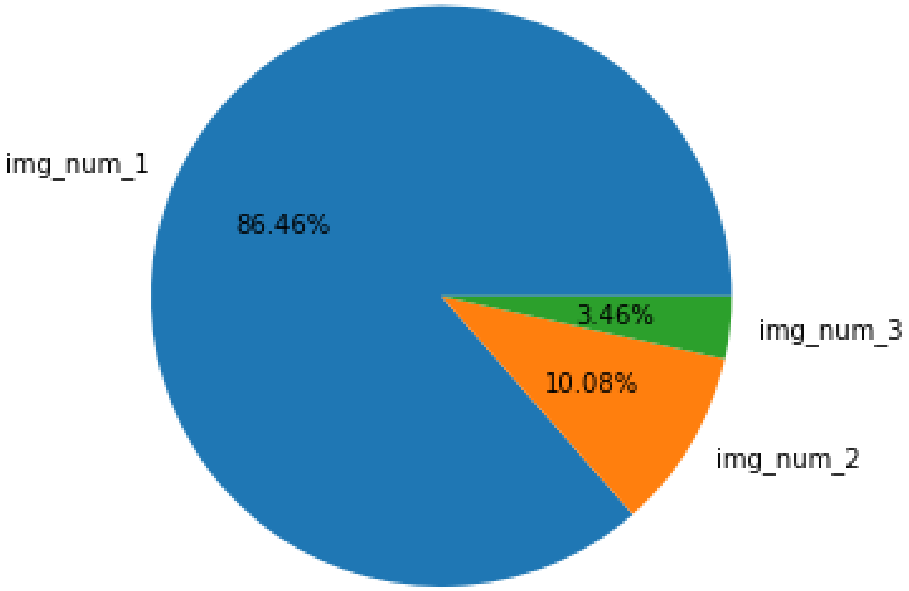
[585]:
```python
from matplotlib import pyplot as plt
def pie_plot(data, labels):
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    ax.axis('equal')
    ax.pie(data, labels = labels,autopct='%1.2f%%')
    plt.show()
```

Count number of rows from df_tweet_clean table for each img_num

[578]:
```python
im1 = len(df_tweet_clean[df_tweet_clean['img_num']==1].index)
im2 = len(df_tweet_clean[df_tweet_clean['img_num']==2].index)
im3 = len(df_tweet_clean[df_tweet_clean['img_num']==3].index)
data = [im1, im2, im3]
labels = ['img_num_1','img_num_2','img_num_3']
```

Plot graph of proportions of tweet by image number

[579]:
```python
pie_plot(data, labels);
```



P1 is more reliable for image prediction

1.1.2 Proportion of tickets per dog stage

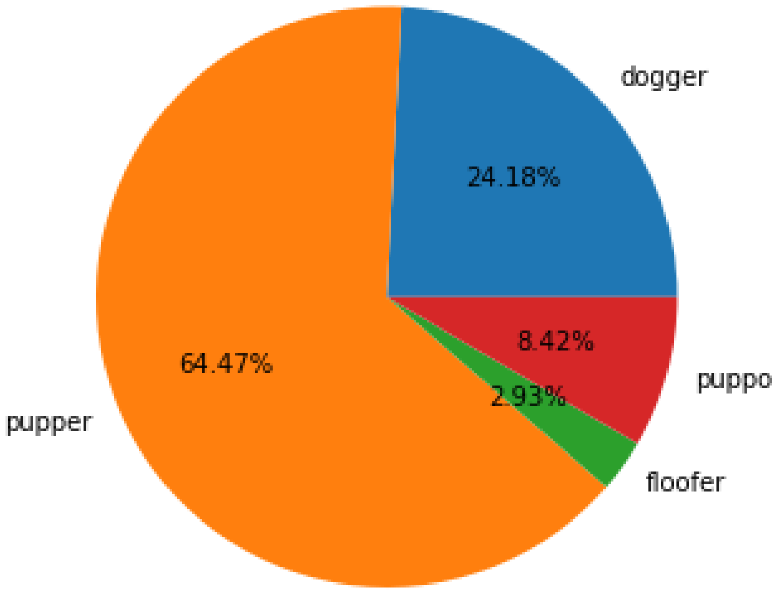
Count rows from twitter_archive_clean table for each dog stage

```
[596]: nb_dogger = len(twitter_archive_clean.
        ↪loc[(twitter_archive_clean['yes_Or_no']=='1')&
                                                   ␣
        ↪(twitter_archive_clean['dog_stage']=='doggo')].index)
       nb_pupper = len(twitter_archive_clean[(twitter_archive_clean['yes_Or_no']=='1')&
                                               ␣
        ↪(twitter_archive_clean['dog_stage']=='pupper')].index)
       nb_floofer =␣
        ↪len(twitter_archive_clean[(twitter_archive_clean['yes_Or_no']=='1')&
                                                ␣
        ↪(twitter_archive_clean['dog_stage']=='floofer')].index)
       nb_puppo = len(twitter_archive_clean[(twitter_archive_clean['yes_Or_no']=='1')&
                                              ␣
        ↪(twitter_archive_clean['dog_stage']=='puppo')].index)
       data = [nb_dogger, nb_pupper, nb_floofer, nb_puppo]
       labels = ['dogger', 'pupper', 'floofer', 'puppo']
```

Plot graph which show each proportion of tweets by dog stage

[597]:

```
pie_plot(data, labels);
```



Pupper dog stage is more predominant

1.1.3 Description of retweet_count and favorite_count per dog stage

Extract rows from twitter_archive clean table for each dog stage

[605]:

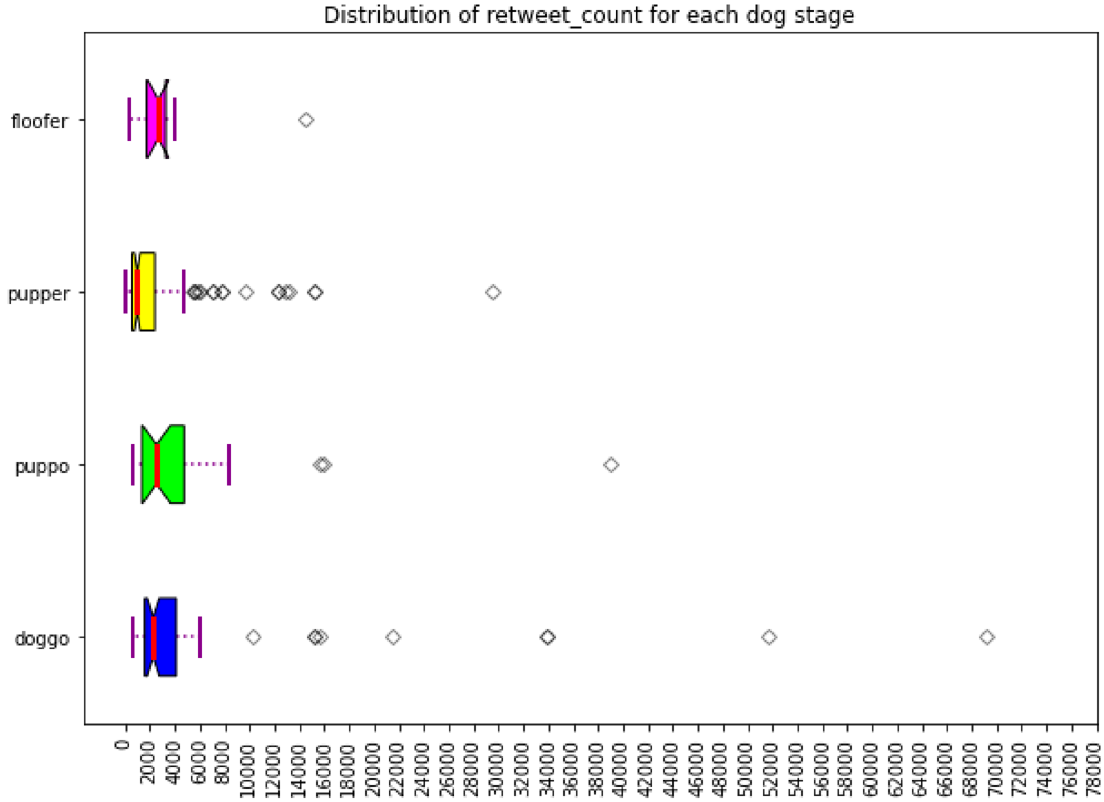
```
doggo_tweet_id =␣
 ↪twitter_archive_clean[(twitter_archive_clean['dog_stage']=='doggo')&(twitter_archive_clean[
 ↪to_numpy()
puppo_tweet_id =␣
 ↪twitter_archive_clean[(twitter_archive_clean['dog_stage']=='puppo')&(twitter_archive_clean[
 ↪to_numpy()
pupper_tweet_id =␣
 ↪twitter_archive_clean[(twitter_archive_clean['dog_stage']=='pupper')&(twitter_archive_clean
 ↪to_numpy()
floofer_tweet_id =␣
 ↪twitter_archive_clean[(twitter_archive_clean['dog_stage']=='floofer')&(twitter_archive_clea
 ↪to_numpy()
```

Function to plot box plot for distributions of retweet_count and favourite_count

```
[653]: def SumBoxPlots(param, stop, step):
           doggo_RT_count = df_tweet_clean[df_tweet_clean['tweet_id'].
        ↪isin(doggo_tweet_id)]['{}'.format(param)].to_numpy()
           puppo_RT_count = df_tweet_clean[df_tweet_clean['tweet_id'].
        ↪isin(puppo_tweet_id)]['{}'.format(param)].to_numpy()
           pupper_RT_count = df_tweet_clean[df_tweet_clean['tweet_id'].
        ↪isin(pupper_tweet_id)]['{}'.format(param)].to_numpy()
           floofer_RT_count = df_tweet_clean[df_tweet_clean['tweet_id'].
        ↪isin(floofer_tweet_id)]['{}'.format(param)].to_numpy()
           data = [doggo_RT_count, puppo_RT_count, pupper_RT_count, floofer_RT_count]
           def box_plot(data):
               fig = plt.figure(figsize =(10, 7))
               ax = fig.add_subplot(111)

               # Creating axes instance
               bp = ax.boxplot(data, patch_artist = True,
                       notch ='True', vert = 0)

               colors = ['#0000FF', '#00FF00',
                     '#FFFF00', '#FF00FF']

               for patch, color in zip(bp['boxes'], colors):
                   patch.set_facecolor(color)

               # changing color and linewidth of
               # whiskers
               for whisker in bp['whiskers']:
                   whisker.set(color ='#8B008B',
                       linewidth = 1.5,
                       linestyle =":")

               # changing color and linewidth of
               # caps
               for cap in bp['caps']:
                   cap.set(color ='#8B008B',
                       linewidth = 2)

               # changing color and linewidth of
               # medians
               for median in bp['medians']:
                   median.set(color ='red',
                      linewidth = 3)

               # changing style of fliers
               for flier in bp['fliers']:
                   flier.set(marker ='D',
                         color ='#e7298a',
                         alpha = 0.5)

               # x-axis labels
               ax.set_yticklabels(['doggo', 'puppo',
                           'pupper', 'floofer'])

               # Adding title
               plt.title("Distribution of {} for each dog stage".format(param))

               # Removing top axes and right axes
               # ticks
               ax.get_xaxis().tick_bottom()
               ax.get_yaxis().tick_left()

               # show plot
               xticks = np.arange(0, stop, step)
               plt.xticks(xticks, xticks)
               plt.xticks(rotation = 90)
               plt.show()
           box_plot(data)
```

[654]:

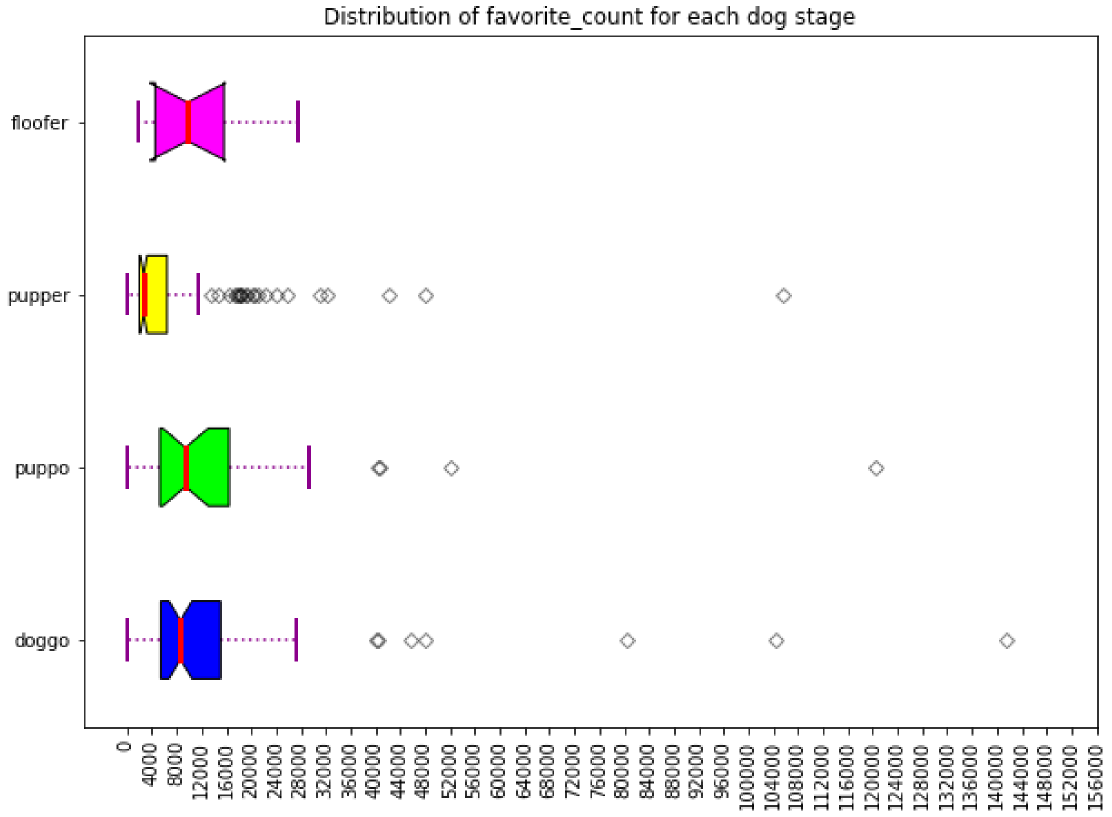
```
SumBoxPlots('retweet_count', 80000, 2000)
```



- **floofer** dog stage has a larger average of retweet_count(about 4000)
- for floofer dog stage, retweet_count max is 4000 (retweet_count average is 4000)
- for pupper dog stage, retweet_count max is 5000 (retweet_count average is 1000)
- for puppo dog stage, retweet_count max is 9000 (retweet_count average is 3000)
- for doggo dog stage, retweet_count max is 6000 (retweet_count average is 3000)

[655]:

```
SumBoxPlots('favorite_count', 160000, 4000)
```



- **floofer** and **puppo** dog stages has a larger average of favorite_count(about 10000)
- for floofer dog stage, favorite_count max is 28000 (favorite_count average is 10000)
- for pupper dog stage, favorite_count max is 12000 (favorite_count average is 2000)
- for puppo dog stage, favorite_count max is 28000 (favorite_count average is 10000)
- for doggo dog stage, favorite_count max is 28000 (favorite_count average is 8000)

1.1.4 Description of rating_numerator and rating_denominator per dog stage

Function to plot box plots for distributions of rating numerator and rating Denominator

[648]:

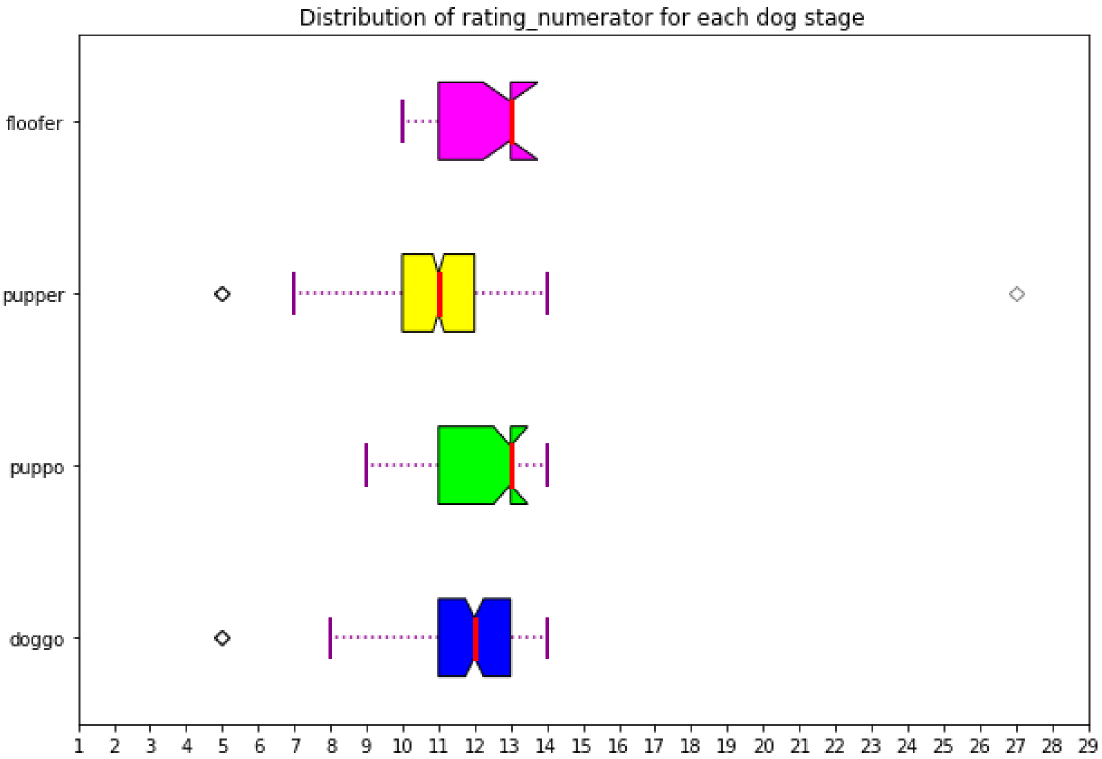
```
def SumBoxPlots2(param):
    doggo_RT_count = twitter_archive_clean[twitter_archive_clean['tweet_id'].
 ↪isin(doggo_tweet_id)]['{}'.format(param)].to_numpy()
```

```
puppo_RT_count = twitter_archive_clean[twitter_archive_clean['tweet_id'].
↪isin(puppo_tweet_id)]['{}'.format(param)].to_numpy()
pupper_RT_count = twitter_archive_clean[twitter_archive_clean['tweet_id'].
↪isin(pupper_tweet_id)]['{}'.format(param)].to_numpy()
floofer_RT_count = twitter_archive_clean[twitter_archive_clean['tweet_id'].
↪isin(floofer_tweet_id)]['{}'.format(param)].to_numpy()
data = [doggo_RT_count, puppo_RT_count, pupper_RT_count, floofer_RT_count]
def box_plot(data):
    fig = plt.figure(figsize =(10, 7))
    ax = fig.add_subplot(111)

    # Creating axes instance
    bp = ax.boxplot(data, patch_artist = True,
            notch ='True', vert = 0)

    colors = ['#0000FF', '#00FF00',
          '#FFFF00', '#FF00FF']

    for patch, color in zip(bp['boxes'], colors):
        patch.set_facecolor(color)

    # changing color and linewidth of
    # whiskers
    for whisker in bp['whiskers']:
        whisker.set(color ='#8B008B',
            linewidth = 1.5,
            linestyle =":")

    # changing color and linewidth of
    # caps
    for cap in bp['caps']:
        cap.set(color ='#8B008B',
            linewidth = 2)

    # changing color and linewidth of
    # medians
    for median in bp['medians']:
        median.set(color ='red',
           linewidth = 3)

    # changing style of fliers
    for flier in bp['fliers']:
        flier.set(marker ='D',
              color ='#e7298a',
              alpha = 0.5)

    # x-axis labels
```

```
        ax.set_yticklabels(['doggo', 'puppo',
                    'pupper', 'floofer'])

        # Adding title
        plt.title("Distribution of {} for each dog stage".format(param))

        # Removing top axes and right axes
        # ticks
        ax.get_xaxis().tick_bottom()
        ax.get_yaxis().tick_left()

        # show plot
        xticks = np.arange(1, 30)
        plt.xticks(xticks, xticks)
        plt.show()
    box_plot(data)
```

[649]:

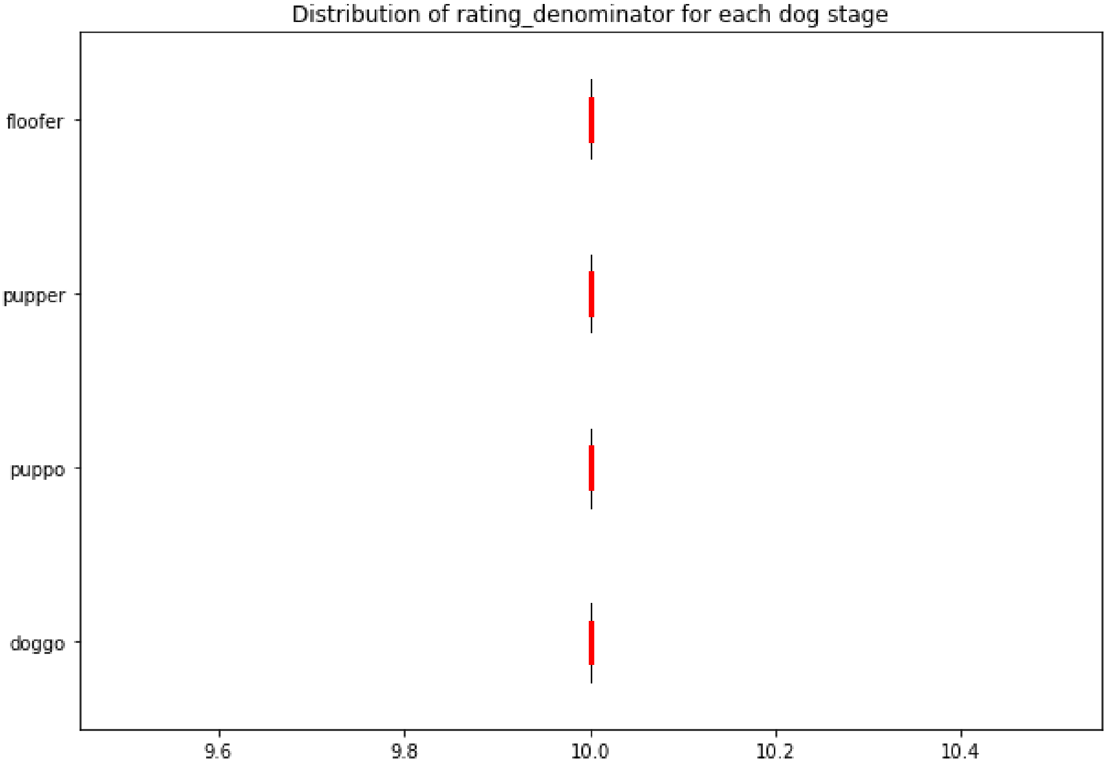
```
SumBoxPlots2('rating_numerator')
```



- for floofer dog stage, rating numerator min is 10 and rating numerator max is 13(equal to rating numerator average)
- for pupper dog stage, rating numerator min is 7 and rating numerator max is 14(rating numerator average is 11)

- for doggo dog stage, rating numerator min is 8 and rating numerator max is 14(rating numerator average is 12)
- for puppo dog stage, rating numerator min is 9 and rating numerator max is 14(rating numerator average is 13)

[637]:
```
SumBoxPlots2('rating_denominator')
```



rating_denominator is always equal to 10.0

1.1.5 Insights:

1. Predictive model **p1** is more reliable
2. **pupper** dog stage is more predominant
3. **puppo** dog stage has a greater spread of the whole tweet, and a greater spread in the middle 50 % of tweet in retweet_count and favorite_count distributions
4. **floofer** dog stage has a larger average of retweet_count(about 4000) and favorite_count(about 10000)
5. **puppo** and **floofer** dog stage have a larger average of ratings(13.0/10.0)