wrangl_report

April 7, 2019

1 Wrangling Report

By Wessam Alhallak

1.1 Table of Contents

Introduction

wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations. The Twitter archive is great, but it only contains very basic tweet information. Additional gathering, then assessing and cleaning is required for "Wow!"-worthy analyses and visualizations.

The Data ### Enhanced Twitter Archive The WeRateDogs Twitter archive contains basic tweet data for all 5000+ of their tweets, but not everything. One column the archive does contain though: each tweet's text, which I used to extract rating, dog name, and dog "stage" (i.e. doggo, floofer, pupper, and puppo) to make this Twitter archive "enhanced." Of the 5000+ tweets, I have filtered for tweets with ratings only (there are 2356).

1.1.1 Additional Data via the Twitter API

Back to the basic-ness of Twitter archives: retweet count and favorite count are two of the notable column omissions. Fortunately, this additional data can be gathered by anyone from Twitter's API. Well, "anyone" who has access to data for the 3000 most recent tweets, at least. But you, because you have the WeRateDogs Twitter archive and specifically the tweet IDs within it, can gather this data for all 5000+. And guess what? You're going to query Twitter's API to gather this valuable data.

1.1.2 Image Predictions File

One more cool thing: I 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 (the top three only) alongside each tweet ID, image URL, and the image number that corresponded to the most confident prediction (numbered 1 to 4 since tweets can have up to four images).

Gathring Data

1.1.3 Reading The CSV from local file system

Using Csv Panda Reader

1.1.4 Reading The img Data file

Using Request pacakge from python

1.1.5 Saving The img Data File

using Panda save or any other os reading writing methods

1.2 Prapring the api creditial from json file

twitter_api.py: This is the Twitter API code to gather some of the required data for the project. Read the code and comments, understand how the code works, then copy and paste it into your notebook. tweet_json.txt: This is the resulting data from twitter_api.py. You can proceed with the following part of "Gathering Data for this Project" on the Project Details page: "Then read this tweet_json.txt file line by line into a pandas DataFrame with (at minimum) tweet ID, retweet count, and favorite count."

Assessing

1.2.1 Programmatically

1.2.2 Code

- Some code I used to assess the data df_csv.head(50) df_img.head() df_img.info() df_api.head() df_csv.info() df_csv.doggo.unique() df_csv.floofer.unique() df_csv.puppo.unique() df_csv.pupper.unique() df_csv[df_csv.expanded_urls.isna()].info() df_csv[df_csv.tweet_id.duplicated()] pd.scatter_matrix(df_csv,figsize=(25,25)) pd.scatter_matrix(df_csv,figsize=(25,25)) df_img.head() df_img.p3.unique()

1.3 Assessing Results

1.3.1 Quality:

df_csv 1 - time stamp need to be changed to datatime instead of string 2 - dummy varialbs (dog type) need to be cleaned into int (boolean) 3 - in_reply_to_status_id should be changed to string type 4 - retweeted_status_user_id should be changed to string type 5 - rating_numerator ,rating_denominator changing into one float variable #### df_img: changing id's into string #### df_api

changing id's into string

1.3.2 Tideness

1 - Dog stages need to be combined into one column 2 - df_api , df_csv represent the same obeservations it would better to be merged together but we should be careful about the nullable and deference count between the raw 3 - merged , df_img represent the same obeservations it would better to be merged together but we should be careful about the nullable and deference count between the raw

Cleaning Data

This part of the data wrangling was divided in three parts: Define, code and test the code. These three steps were on each of the issues described in the assess section.

First and very helpful step was to create a copy of the three original dataframes. I wrote the codes to manipulate the copies. If there was an error, I could create a new copy from the original.

Whenever I made a mistake, I could create another copy of the dataframes and continue working on the cleaning part.

There were a couple of cleaning steps that were very challenging. One of them was in the image prediction table. I had to create a 'nested if' inside a function in order to capture the first true prediction of the type of dog. The original table had three predictions and confidence levels. I filtered this into one column for dog type and one column for confidence level.

Other interesting cleaning code was to melt the dog stages in one column instead of four columns as original presented in twitter archive.

One very challenging cleaning step was when I had to correct some numerators that were actual decimals. This issue was brought to my attention after the first Udacity review. Using Excel and visual assessment was not sufficient to verify those decimals. Therefore, I had to run a code in order to check those actual tweets (decimals numerators).

1.3.3 Quality

1.3.4 df_csv

First We will copy our dataframe to a new data frame to ensure the consisty of our work

Define chaging time into datatime type

Code

```
df_csv_clean["timestamp"]=pd.to_datetime(df_csv_clean.timestamp)
```

Test

```
df_csv_clean.info()
```

Define chaging tweet_id into string type

Code

```
df_csv_clean["tweet_id"]=df_csv_clean["tweet_id"].astype(str)
```

Test

```
df_csv_clean.head()
```

Define

Drop retweeted_status_id, retweeted_status_user_id,in_reply_to_status_id,in_reply_to_user_id

Code

```
df_csv_clean.drop(["retweeted_status_id",
                   "retweeted_status_user_id",
                   "in_reply_to_status_id",
                   "in_reply_to_user_id",
                   "retweeted_status_timestamp"],
                   axis=1,inplace=True)
```

Test

```
df_csv_clean.info()
```

Define The current pipeline captures incorrect values when rating numerators contain decimals and it will better to compine both numerator and deomerator in one column as it descripe one mesure

Code

```
df_csv_clean[df_csv_clean.rating_numerator>10].info()
ratting=df_csv_clean.text.str.extract('((?:\d+\.)?\d+)\/(\d+)', expand=True)
df_csv_clean.rating_numerator=ratting[0].astype(float)
df_csv_clean.rating_denominator=ratting[1].astype(float)
df_csv_clean=df_csv_clean[df_csv_clean["rating_denominator"]!=0]
```

Test

```
df_csv_clean.info()
df_csv_clean.describe()
```

1.3.5 df_img

Copy df_img to a new dataframe

Define 1- chaging id into string

Code

```
df_img_clean["tweet_id"]=df_img_clean["tweet_id"].astype(str)
```

Test

```
df_img_clean.info()
df_img_clean.describe()
```

1.3.6 df_api

Copy df_api into other cleaning Dataframe

```
df_api_clean=df_api.copy()
```

Define chage id into string

Code

```
df_api_clean["tweet_id"]=df_api_clean["tweet_id"].astype(str)
```

1.4 Cleaning

1.4.1 Tidiness

df_csv_clean

Define Dog stages need to be combined into one column

```
df_csv_clean.loc[(df_csv_clean[['doggo', 'floofer', 'pupper', 'puppo']] != 'None'
             ).sum(axis=1) > 1]
```

Code

```
df_csv_clean.doggo.replace("None","",inplace=True)
df_csv_clean.floofer.replace("None","",inplace=True)
df_csv_clean.pupper.replace("None","",inplace=True)
df_csv_clean.puppo.replace("None","",inplace=True)
df_csv_clean['stage'] = df_csv_clean.doggo + df_csv_clean.floofer + df_csv_clean.pupper + df_cs
df_csv_clean.loc[df_csv_clean.stage == 'doggopupper', 'stage'] = 'doggo,pupper'
df_csv_clean.loc[df_csv_clean.stage == 'doggopuppo', 'stage'] = 'doggo,puppo'
df_csv_clean.loc[df_csv_clean.stage == 'doggofloofer', 'stage'] = 'doggo,floofer'
df_csv_clean.stage.replace("","None",inplace=True)
```

Test

```
df_csv_clean.stage.value_counts()
```

Define Drop the dummy variabls

Code

```
df_csv_clean.drop(['doggo', 'floofer', 'pupper', 'puppo'],axis=1,inplace=True)
```

Test

```
df_csv_clean.info()
```

df_api , df_csv #### Define

represent the same obeservations it would better to be merged together but we should be careful about the nullable and deference count between the raw

Code We will Use merge for two Dataframe

```
merged_1=df_csv_clean.merge(df_api_clean,how="inner",on="tweet_id")
```

Test

```
merged_1.describe()
merged_1.head()
merged_1.info()
```

1.4.2 df_img , merged_1

Define df_img represent the same obeservations it would better to be merged together but we should be careful about the nullable and deference count between the raw

Code

```
merged_2=merged_1.merge(df_img_clean,how="inner",on="tweet_id")
```

Test

```
merged_2.info()
merged_2.describe()
merged_2.head()
```

Saving Data ### saving as csv in local file system

```
df=merged_2.copy()
```

1.5 Conclusion

Data wrangling is a core skill that whoever handles data should be familiar with.

I have used Python programming language and some of its packages. There are several advantages of this tool (as compared to e.g. Excel) that is used by many data scientists (including the guys at Facebook).

For gathering data there are several packages that help scraping data off the web, that help using APIs to collect data (Tweepy for Twitter) or to communicate with SQL databases. It is strong in dealing with big data (much better than Excel). It can deal with a large variety of data (unstructured data like JSON (Tweets) or also structured data from ERP/SQL databases.

It is easy to document each single step and if needed re-run each single step. Thus, one can leave a perfect audit trail (perfect for the accountant). One can re-run analysis automatically every period. Thus, we could actually re-run the dog analysis every month with much less effort now because I have set it up once. Handling, assessing, cleaning and visualizing of data is possible programmatically using code.

Refernces :

1- https://github.com/latinacode/
2- https://www.python.org/
3- https://www.kaggle.com/datasets
4- https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/
5- https://stackoverflow.com/questions/28384588/twitter-api-get-tweets-with-specific-id