wrangle_report

December 27, 2017

1 Data Wrangling

1.1 Gathering

There were three different data sets that I had to wrangle. The first was an archive of tweets from weratedogs twitter handle. This was simple and straight forward to extract because all I needed to do was download the flat file that Udacity provided and read it into a dataframe using pandas like so:

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

The second dataset needed to be extracted using the requests library. This was also simple and straightforward because Udacity provided the url which had the flat file I needed and then I simply used the following command to extract the raw data:

```
image_predictions =
    re.get('https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image
```

Then I needed to take the content of the raw data (which was in type *Bytes*) and convert it into a datframe using the following code:

```
image_predictions_df = pd.read_csv(io.StringIO(rawData.decode('utf-8')), sep='\t')
```

The third dataset was the most difficult to wrangle. It was the first time I used an API wrapper. I first needed to get my creditials from https://apps.twitter.com and after I submitted my request and received the credentials I used the following code to extract the raw data I needed:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
#json parser will output tweet info in dictionary format
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True,
      parser=tweepy.parsers.JSONParser())
```

The api variable is a special object type from tweepy library and extracts the tweets in a Json format. Since Json objects are structured very similar to to a python dictionary it was easy to read the raw data using the "keys" in the key-value pairs.

The biggest hurdle I ran into was writing the tweets data into a txt file. Because this block of code takes a while to run, I first created a list of four tweet id's and did a test run before I started to go through each tweet id in the twitter archive dataframe.

```
with open('tweet_json.txt', 'a+', encoding='utf-8') as file:
    for tweet_id in twitter_archive_df['tweet_id']:
        try:
            tweet = api.get_status(id = tweet_id, tweet_mode='extended')
            file.write(json.dumps(tweet))
            file.write('\n')
        except:
            pass

file.close()
```

I then read the Json objects into a dataframe using the following code:

```
with open('tweet_json.txt') as file:
    status = []
    for line in file:
        status.append(json.loads(line))
```

1.2 Assesing

I used a combination of programmatic and visual assessments to determine a number of messy and tidy data issues that I would need to clean.

I primarily used the following commands in pandas to do my assessments:

```
df.head()
df.tail()
df.info()
df.sample()
df.describe()
df.value_counts()
```

I also used the following commands to find duplicated values and the amount of null values for certain columns:

```
df[df['column'].duplicated()]

len(df[pd.isnull(df['column'])])
```

1.3 Cleaning

In the final stage of the data wrangling process I worked on the fixing the issues I found in the assessment stage. My revisions were primarily made to the twitter archive dataframe and the tweets dataframe I extracted using the requests library. To ensure that I didn't screw up my original dataframes, I made copies of both using the following code:

```
twitter_archive_df_clean = twitter_archive_df.copy()
tweets_df_clean = tweets_df.copy()
```

At this point I used the pandas library to make adjustments to the dataframes and then combine the dataframes into one master dataframe. This is where the pandas library really shines and is extremely powerful. I love this library and plan on improving my knowledge of it as I progress my career in data analytics. After I cleaned all the issues I found in the assessment stage, I merged the dataframes into one master dataframe using the following code:

```
tweets_master_df = pd.merge(twitter_archive_df_clean, tweets_df_clean,
                            on='tweet_id', how='right')

tweets_master_df = pd.merge(tweets_master_df, image_predictions_df,
                            on='tweet_id', how='right')
```

Then I dropped any rows which still had remaining dirty data (there was only a handful) using the following code:

```
tweets_master_df.dropna(inplace=True)
```