Capstone1: Write up for Data Wrangling

The purpose of this document is to explain the steps taken to clean and process the Capstone project data set and some of the findings from the process. A number of sources were used to support this work and have been referenced in the final section of the report.

Steps

1. Set up jupyter notebook - `data_wrangling.ipynb`
2. Load in data for reviews and businesses
3. Exploratory analysis
4. Reformat column for date into data-time category
5. Extract businesses that are restaurants
6. Extract restaurants that are in the USA using a bounding box
7. Drop business columns associated with business location
8. Merge the two dataframes
9. Explore merged dataframe
10. Pre-processing the data
11. Saving the output

1. Set up jupyter notebook

The data wrangling stage can be found a Jupyter notebook called `data_wrangling.ipynb`. It will contain the key steps laid out in this document. The inputs are described in step 2, and the outputs are described in step 11.

As part of the set up, the library dependencies are loaded.

In [1]:
```
# Import libraries
import json
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Import NLP dictionaries
import nltk
import string
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [2]:
```
# get current directory
dir = os.path.dirname(os.path.abspath('__file__'))
```

In particular:
The common DataScience libraries
- json for loading in json files
- os for non-OS dependent file paths
- pandas for manipulation of DataFrames
- matplotlib.pyplot for plots and simple visualizations
- numpy to handle simple data arrays

As well as more specific NLP libraries
- nltk for stopwords, the lemmatization,
- string to get the library of punctuations
- sklearn.feature_extraction.text for the string to word vectorizer

The working directory is also generated for future reference.

2. Load in data

The dataset was obtained as part the '[Yelp dataSet Challenge - Round 10](https://www.yelp.com.sg/dataset)' (September 1, 2017 to December 31, 2017)

In particular, the `review.json` and `business.json` files from the JSON dataset.

These are currently stored in the folder labeled ` 01_raw_data`. The terms and conditions of using the dataset limits sharing of the raw data and it shall be excluded in the git repo.

The raw data in the original document comes in the format of dictionaries separated by lines:

```
{}
{}
{}
```

To load the data, first a connection is opened to the file, then the data is read using a list comprehension and finally the list is converted into a DataFrame. A similar process used to load business data.

Load review data

In [3]:
```
# Load review data
# Raw data in the format of {} {} {} seperated by lines

# get file path (generalize for different OS) for reviews
filename_review = os.path.join(dir, '01_raw_data','review.json')

# create a list of reviews
with open(filename_review, encoding="utf8", mode='r') as file:
    reviews = [json.loads(line) for line in file]

# create a pandas data frame from review data
reviews_df = pd.DataFrame(reviews)
```

Load business data

In [4]:
```
# Load business data

# get file path (generalize for different OS) for reviews
filename_business = os.path.join(dir, '01_raw_data','business.json')

# create a list of reviews
with open(filename_business, encoding="utf8", mode='r') as file:
    businesses = [json.loads(line) for line in file]

# create a pandas data frame from review data
businesses_df = pd.DataFrame(businesses)
```

3. Exploratory analysis

To understand the data set, 3 functions were used: `.shape`, `.info`, `.describe`. Then a null check was performed. And finally, a head of 50 was used to eyeball the data set.

For the review data, the following observations were made:
1. There were no null values
2. Except for the data, the data types from the column were correct
3. The full data set had ~ 4.7 million observations
4. Non-restaurant businesses were also reviewed (e.g. accommodation)
5. Reviews were not all in English

For the business data, the following observations were made:
1. There were only 2 null values (1 Lat, 1 Long)
2. The data types were not an issue but there were a number of columns related to the location of the business
3. The full data set had ~ 150 k observations
4. Non-restaurants were included
5. A number of locations were outside the USA (e.g. state: ON, postal_code: M4K 1N7 refers to Toronto)

Unique state values and the range of lat long values were further analyzed and revealed that a number of businesses were outside the USA.

4. Reformat column for date into data-time category

To clean the review data, very little clean up was necessary. The date was converted to datetime but the stars were kept as integers vs categories.

```
reviews_df['date'] = pd.to_datetime(reviews_df['date'], format='%Y-%m-%d')
```

5. Extract businesses that are restaurants

The first clean up that was done was to remove businesses that were non-restaurants. This was done first as it was expected to yield the greatest reduction in observations.

```
restaurants_df = businesses_df[ businesses_df['categories'].apply(lambda categories: any(pd.Series(categories).str.contains('Restaurants')) if len(categories)>0 else False)]
```

The new dataset of businesses were ~ 50k observations (almost 1/3 the original number of businesses).
Note: the impact on the number of reviews is less severe as we will observe later.

6. Extract restaurants that are in the USA using a bounding box

The next step of the clean up was to remove restaurants outside the USA using a bounding box of min-max latitude and longitudinal values. The bounding box for the US is (49.3457868 # north lat) (24.7433195 # south lat) (-124.7844079 # west long) (-66.9513812 # east long) .
This reduced the dataset to ~48k observations.

7. Drop business columns associated with business location

Next the columns associated with the business location and opening hours were dropped.

Namely: `dropped_columns = ['address','city','hours', 'is_open','neighborhood','postal_code', 'state', 'latitude', 'longitude']`

8. Merge the two dataframes

Restaurant business data frame was merged with the review dataframe. The data was merged using an inner join on the 'business_id'.
`joint_df = pd.merge(reviews_df, restaurants_df, on='business_id', suffixes=['_review', '_business'])`

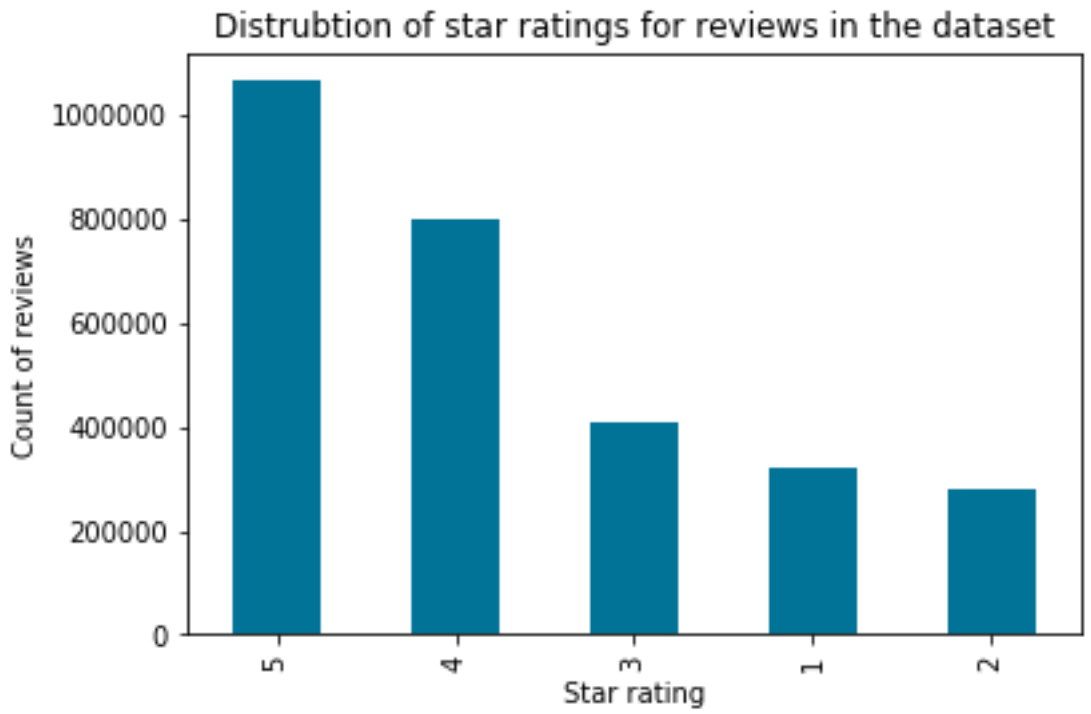
9. Explore merged dataframe

To understand the merged data set, 2 functions were used: `.shape`, `.info`.

For the merged data, the following observations were made:
1. There were no null values
2. Inner join reduced the total number of observations (i.e. reviews) from ~4.74 million from all businesses to ~ 2.92 million from restaurants alone to the ~2.88 from restaurants in the USA bounding box.

By plotting a bar chart to show the distribution of star ratings from the reviews. It may be observed that there is a bias towards positive 5 star ratings.

No outliers are apparent at this stage.



10. Pre-processing the data

The pre-processing of the reviews `joint_df['text']` involved 5 steps. These were defined in function named `pre_process_review`:

```
def pre_process_review( review_text ):

    # Remove punctuation and convert all text to lower case
    nopunc = ''.join( [character for character in review_text.lower() if character not in string.punctuation] )

    # Remove stop words and lemmatize
    non_stop_words  = [lemmatizer.lemmatize(word) for word in nopunc.split() if not word in stopWords]

    return non_stop_words
```

1. Removed punctuation
2. Convert characters to lower case
3. Removal of stop words
4. Lemmatization of words
5. Conversion for review text string to a list of words (tokenization)

Note that the additional step of expanding contractions was not used, but it would come as step 1 if it was needed.

These steps were then applied in to the `joint_df['text']` column and stored into `joint_df['processed_review']` column. Using:

```
joint_df['processed_review'] = joint_df['text'].apply(lambda review_text: pre_process_review( review_text ))
```

For more information on pre-processing, please refer to references [2] [3] [4], which helped inform the steps selected above.

To make this function more efficient, I took advice from references [5] [6] and refactored the function into the following form:

```
def pre_process_review( review_text ):
    # Convert all text to lower case, tokenize into list of strings, remove punctuation and stop words, and lemmatize
    return [lemmatizer.lemmatize(word) for word in wordpunct_tokenize(review_text.lower()) if not word in stop_words_punc]

```

Since the timing over 1 function proved too short and inconsistent, performance was measured using the cell that applied the function across all review text.

The original function took 1776 seconds, the modified function took 1389 seconds. Note for the original function, ` nopunc = ''.join( character for character in review_text.lower() if character not in string.punctuation )` was used instead of `nopunc = ''.join( [character for character in review_text.lower() if character not in string.punctuation] )`.

The processes was timed using:

```
import timeit
start_time = timeit.default_timer()
# CODE TO BE CHECKED
print(timeit.default_timer() - start_time)
```

Finally, unrequired columns from the merged dataset will be dropped. It may be noted that the text column has an observation that does not store and load well and will need to be found and removed if text column is to be included.
Namely: `dropped_columns = ['business_id', 'review_id', 'text', 'name']`

11. Saving the output

The output of this analysis is the merged data of reviews and restaurants in the USA that has been pre-processed. In total, the output CSV file will have 11 columns (excluding an index). This file ` restaurant_reviews.csv` will be stored in new files folder titled `02_processed_data`.

Description of columns in the saved data set:

1. Date — date of review (YYYY-MM-DD)
2. cool — number of people who voted this review as 'cool'
3. funny — number of people who voted this review as 'funny'
4. useful — number of people who voted this review as 'useful'
5. stars_review — number of stars given to the restaurant by this review
6. stars_business — average of star ratings given to the restaurant
7. user_id — ID of the user
8. attributes — dictionary of attributes of the restaurant (e.g. 'RestaurantsTableService': False)
9. categories — list of categories that describe the business (e.g. [Sandwiches, Restaurants])
10. review_count — number of reviews received by the restaurant
11. processed_review — pre-process text that is split into a list of words

Note that due to the restrictions of data use, these files will not be shared in a public repository.

A quick verification of the data integrity was conducted using `.shape` `.info()` on both the saved data frame, as well as the loaded data frame. No issues were identified.

References:

[1] (http://en.wikipedia.org/wiki/Extreme_points_of_the_United_States#Westernmost)
[2] (https://www.kaggle.com/c/word2vec-nlp-tutorial#part-1-for-beginners-bag-of-words)
[3] (https://radimrehurek.com/data_science_python/)
[4] (http://textminingonline.com/dive-into-nltk-part-iv-stemming-and-lemmatization)
[5] (https://stackoverflow.com/questions/46203023/how-can-i-make-my-python-nltk-pre-processing-code-more-efficient)
[6] (https://stackoverflow.com/questions/19130512/stopword-removal-with-nltk)