Yelp Dataset Experiments

February 11, 2021

1 Introduction

This document includes analysis and experiments carried out on Yelp Dataset. It contains reviews and corresponding users and businesses in North America. We first do some exploratory data analysis in the Section 2. Then we do a few machine learning tasks explained in Section 3. In this report, most of the code cells and outputs are removed for clarity. If you're interested, you could check the accompanying jupyter-notebook files.

2 Data Analysis

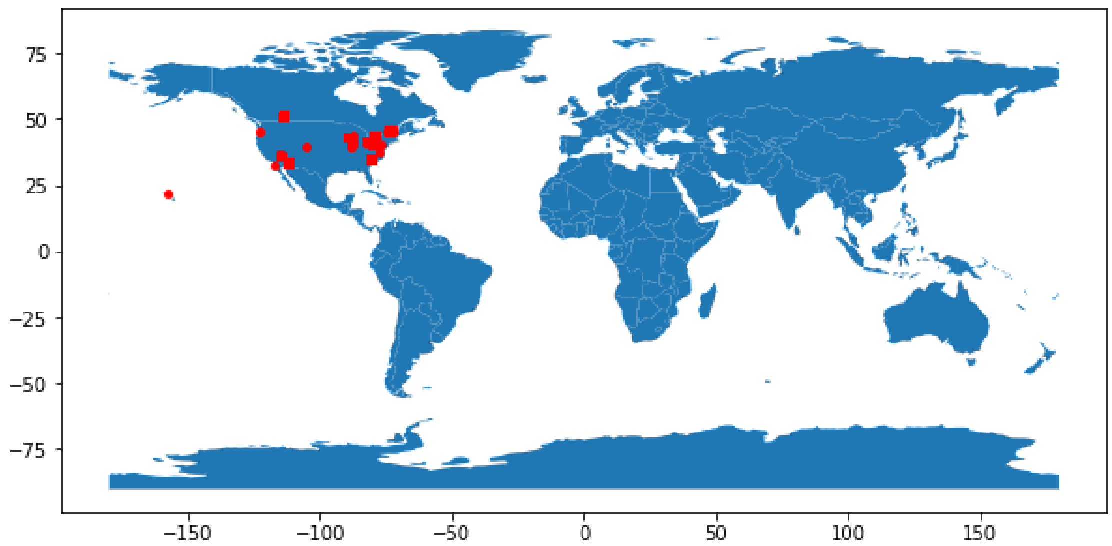
We first analyze the businesses.

Let's import necessary modules and read the business file.

[9]:
```
import pandas as pd
import matplotlib.pyplot as plt
from shapely.geometry import Point
import geopandas as gpd
from geopandas import GeoDataFrame
```

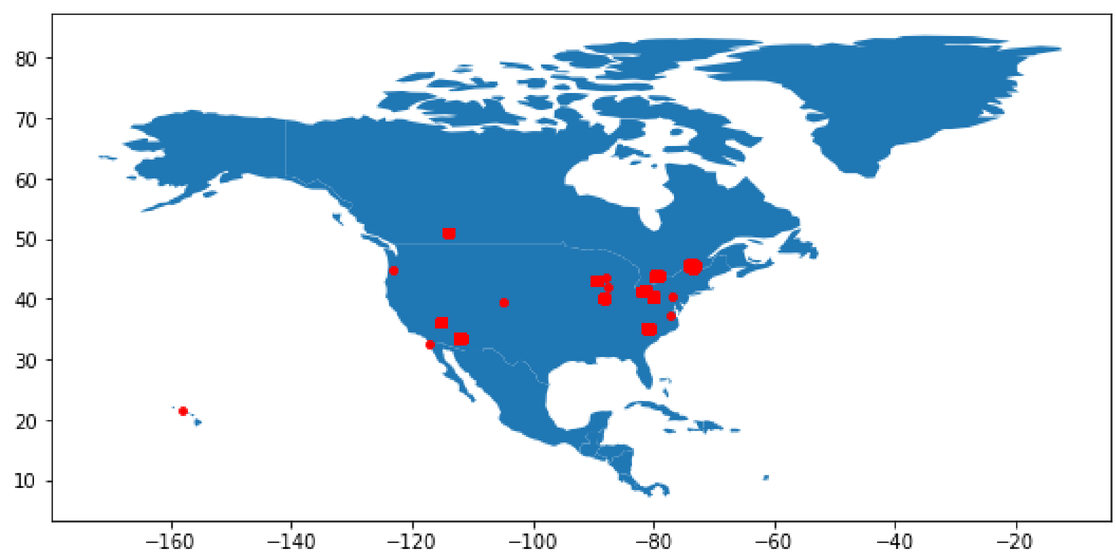
Using latitude and longitute, we plot the map and see the locations of business.

[11]:
```
geometry = [Point(xy) for xy in zip(df['longitude'], df['latitude'])]
gdf = GeoDataFrame(df, geometry=geometry)

#this is a simple map that goes with geopandas
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
gdf.plot(ax=world.plot(figsize=(10, 6)), marker='o', color='red', markersize=15);
```



[12]:
```
world = world[world.continent == 'North America']
gdf.plot(ax=world.plot(figsize=(10, 6)), marker='o', color='red', markersize=15);
```

They are mainly located in North America, so we zoom a little



[13]:
```
world = world[world.name == 'United States of America']
gdf.plot(ax=world.plot(figsize=(10, 6)), marker='o', color='red', markersize=15);
```

They are mainly located in United States, so we zoom a little more

There are 209393 rows (businesses). Atributes, categories and hours are not provided for some

businesses

[16]:
```
null_rows = df[df.isnull().any(axis=1)]
select_columns = ['business_id', 'name', 'stars', 'review_count', 'is_open',␣
 ↪'attributes', 'categories', 'hours']
null_rows[select_columns].head(10)
```

Attributes, categories and hours are strings and null values cannot directly be filled with average statistics from other rows.

[17]:
```
# check examples of other columns
list_of_columns = df.columns.tolist()
print(list_of_columns)
```

```
['business_id', 'name', 'address', 'city', 'state', 'postal_code', 'latitude',
'longitude', 'stars', 'review_count', 'is_open', 'attributes', 'categories',
'hours', 'geometry']
```

[18]:
```
df[list_of_columns[:-3]].head()
```

The majority of business are located in Las Vegas, Toronto, Phoenix, Charlotte, Scottsdale and cities like Queensville, Huntigdon, Rocky River, Rainbow Valley and Gilbert, among others have listed only 1 business each.

[20]:
```
# stats for number of reviews
df.review_count.describe()
```

[20]:
```
count    209393.000000
mean         36.937505
std         123.343597
min           3.000000
25%           4.000000
50%           9.000000
75%          27.000000
max       10129.000000
Name: review_count, dtype: float64
```

The minimum number of reviews for a business listed in the dataset is 3. 75% of the the businesses have received 27 reviews or less. That means very few businesses have received extremely high number of reviews thus skewing the mean, i.e., 36 reviews. First we check how many businesses have fewer than 50 reviews out of the 209393 business.

[21]:
```
df[df['review_count']<50].shape[0]
```

[21]:
```
177451
```

177451 out of 209393 business have fewer than 50 reviews leaving 31942 with higher number of reviews than 50.

Now, let's 10 businesses with the least number of reviews.

```
[22]: df[['name', 'city', 'state', 'review_count', 'stars']].
       ↪sort_values(by='review_count').head(10)
```

```
[22]:                              name         city state  review_count  \
209392                        Kudlow Ye      Toronto    ON             3
190418                        241 Pizza  Scarborough    ON             3
75451   MAI Montreal Arts Interculturels     Montréal    QC             3
75454                           Sachika     Montréal    QC             3
75461                      Jungle Juice      Toronto    ON             3
190414                          SAS Too      Phoenix    AZ             3
75467                   Toepel  Company         Mesa    AZ             3
75468                    Baja Ready Mix      Phoenix    AZ             3
75471                    F T Financial   Scottsdale    AZ             3
75476      Grand and Clover Cocktail Co.      Toronto    ON             3

        stars
209392    5.0
190418    3.5
75451     4.5
75454     4.5
75461     3.5
190414    2.0
75467     5.0
75468     3.5
75471     5.0
75476     5.0
```

And 10 businesses with the most number of reviews.

```
[23]: df[['name', 'city', 'state', 'review_count', 'stars']].
       ↪sort_values(by='review_count', ascending=False).head(10)
```

```
[23]:                           name       city state  review_count  stars
81545                Bacchanal Buffet  Las Vegas    NV         10129    4.0
118008                   Mon Ami Gabi  Las Vegas    NV          9264    4.0
147379                   Wicked Spoon  Las Vegas    NV          7383    3.5
83020                Hash House A Go Go  Las Vegas    NV          6751    4.0
201975             Gordon Ramsay BurGR  Las Vegas    NV          5494    4.0
95962                 Earl of Sandwich  Las Vegas    NV          5232    4.5
22754   Yardbird Southern Table & Bar  Las Vegas    NV          4828    4.5
145294                   Secret Pizza  Las Vegas    NV          4803    4.0
205740              The Buffet At Wynn  Las Vegas    NV          4803    3.5
77432   The Cosmopolitan of Las Vegas  Las Vegas    NV          4740    4.0
```

Let's aggregate the number of reviews for each city.

We first list 20 cities with the smallest total reviews for the businesses located there.

[24]:
```
grouped_review = df.groupby(['city']).agg({'review_count': ['sum','mean', 'min',␣
↪'max']})
grouped_review.columns = ['review_sum', 'review_mean', 'review_min',␣
↪'review_max']
grouped_review = grouped_review.reset_index()
grouped_review.sort_values(by='review_sum').head(20)
```

[24]:
```
                          city  review_sum  review_mean  review_min  \
553               McMasterville           3          3.0           3
1048                St-Laurent           3          3.0           3
1045        St-Jean Sur Richelieu        3          3.0           3
1044                   St-Clet           3          3.0           3
316                Fountain Hls           3          3.0           3
167                    Chateau           3          3.0           3
1029              South Heights           3          3.0           3
418                     Joliet           3          3.0           3
1025                  Somerton           3          3.0           3
1049                 St-Lazare           3          3.0           3
735             O'hara Township           3          3.0           3
176                   Citibank           3          3.0           3
1014                    Sharon           3          3.0           3
179                  Claremont           3          3.0           3
1007         Sgs Industrial Park          3          3.0           3
183                 Cleveland            3          3.0           3
1002                Scottsdsale          3          3.0           3
742                    Oakvile           3          3.0           3
408   Indian Land, South Carolina          3          3.0           3
174             Chomedey, Laval           3          3.0           3

      review_max
553            3
1048           3
1045           3
1044           3
316            3
167            3
1029           3
418            3
1025           3
1049           3
735            3
176            3
1014           3
179            3
1007           3
183            3
1002           3
```

```
742          3
408          3
174          3
```

We then list 20 cities with the highest total reviews for the businesses located there.

[25]:
```
grouped_review.sort_values(by='review_sum').tail(20)
```

[25]:

	city	review_sum	review_mean	review_min	review_max
521	Markham	59004	30.058074	3	638
705	North Las Vegas	60405	36.019678	3	815
599	Mississauga	65557	18.634736	3	1009
792	Peoria	69539	33.432212	3	713
135	Calgary	107106	12.785723	3	512
508	Madison	114475	31.065129	3	1879
341	Glendale	118096	30.882845	3	1075
182	Cleveland	129489	33.572466	3	1372
337	Gilbert	153826	41.075033	3	2369
626	Montréal	171059	24.510532	3	2667
158	Chandler	183475	40.060044	3	1346
577	Mesa	197214	29.985404	3	1362
1095	Tempe	229594	47.861997	3	2400
376	Henderson	261244	49.553111	3	2456
826	Pittsburgh	262412	34.392136	3	2001
163	Charlotte	371580	35.653425	3	2174
1000	Scottsdale	439439	47.039071	3	2369
1106	Toronto	583512	28.651282	3	2758
804	Phoenix	842321	41.759010	3	3515
461	Las Vegas	2360735	74.633587	3	10129

Las Vegas, Toronto, Phoenix, Charlotte, and Scottsdale has higher total reviews than other cities which is understandable since there are more businesses in the dataset located in these locations, as shown above. Let's show a scatter plot of number of businesses and number of reviews for each city.

We first merged the two dataframes for number of total reviews and number of businesses.

[26]:
```
grouped_business = df.value_counts('city').reset_index()
grouped_business.columns= ['city', 'num_businesses']
grouped_business
```

[26]:

	city	num_businesses
0	Las Vegas	31631
1	Toronto	20366
2	Phoenix	20171
3	Charlotte	10422
4	Scottsdale	9342
...	...	...

```
1246     Queensville                1
1247      Huntingdon                1
1248     ROCKY RIVER                1
1249  Rainbow Valley                1
1250         Gilbert               1

[1251 rows x 2 columns]
```

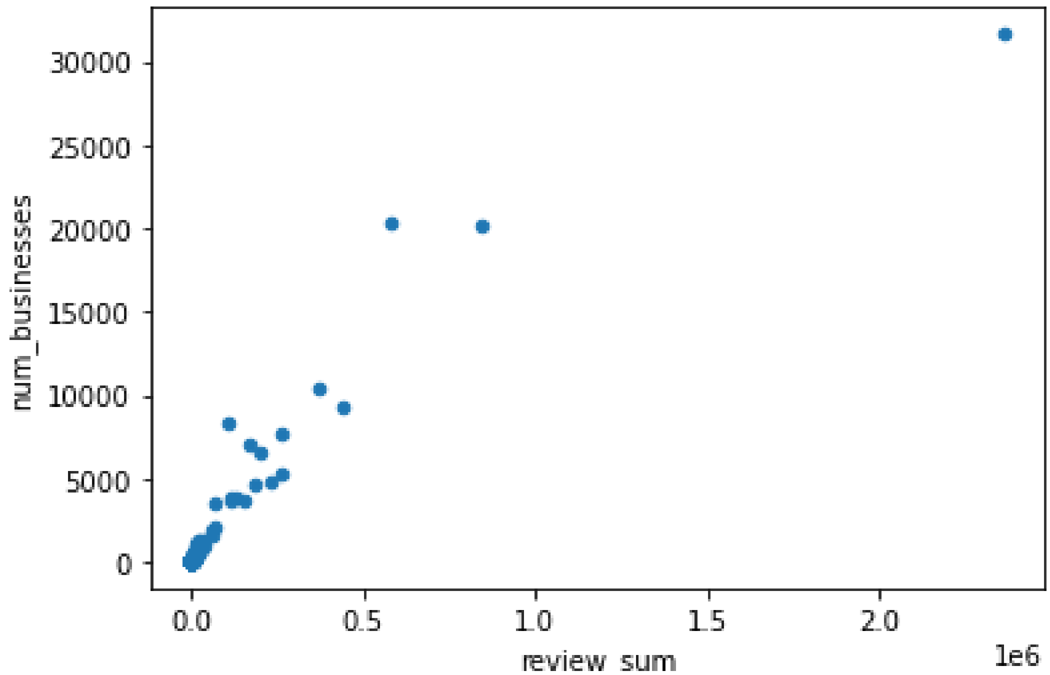
[27]:
```python
merged = pd.merge(grouped_review, grouped_business,on='city')
merged
```

[27]:
```
                      city  review_sum  review_mean  review_min  review_max  \
0                                    7          3.5           3           4
1             110 Las Vegas         43         43.0          43          43
2        4321 W Flamingo Rd        292        292.0         292         292
3                   ARSENAL          4          4.0           4           4
4                        AZ          4          4.0           4           4
...                     ...        ...          ...         ...         ...
1246                toronto         24          8.0           3          11
1247             Île-Perrot          8          8.0           8           8
1248         Île-des-Soeurs          5          5.0           5           5
1249               Avondale         5          5.0           5           5
1250                Gilbert        17         17.0          17          17

      num_businesses
0                  2
1                  1
2                  1
3                  1
4                  1
...              ...
1246               3
1247               1
1248               1
1249               1
1250               1

[1251 rows x 6 columns]
```

We can now show the scatter plot.

[28]:
```python
merged.plot(kind='scatter', x='review_sum', y='num_businesses')
```

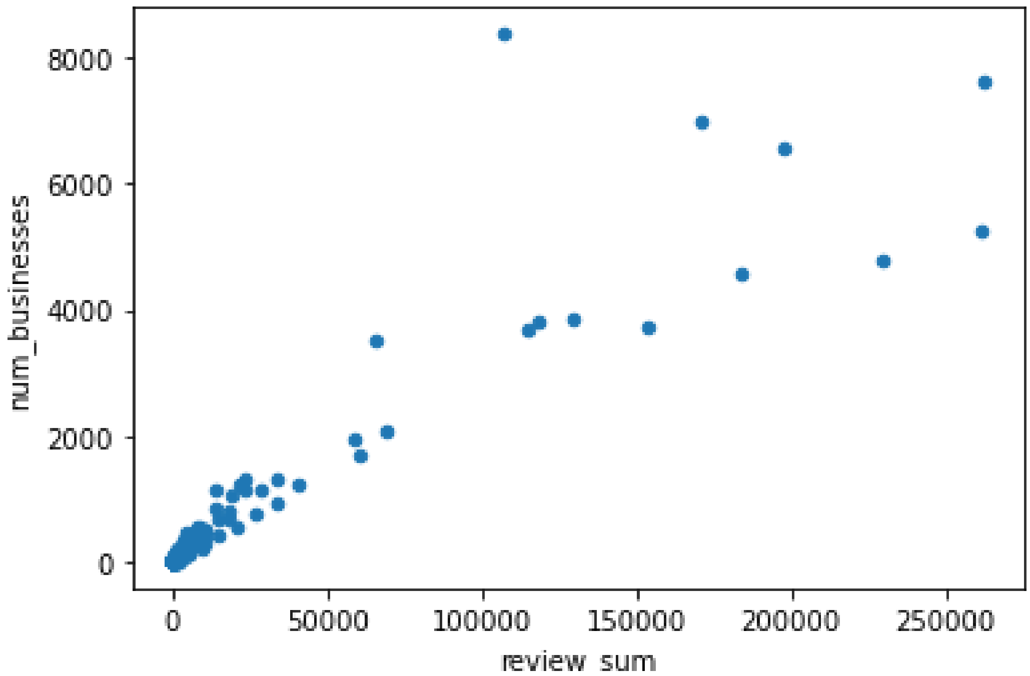
[28]: <AxesSubplot:xlabel='review_sum', ylabel='num_businesses'>



Taking a first look at the scatterplot, it seems there is a positive correlation between the number of reviews and the number of businesses. There are some extreme outlier cities with 2000-3000 businesses. Let's just plot only those with fewer than 9000 businesses in total.

[29]:
```
merged[merged['num_businesses']<9000].plot(kind='scatter', x='review_sum',␣
↪y='num_businesses')
```

[29]: <AxesSubplot:xlabel='review_sum', ylabel='num_businesses'>

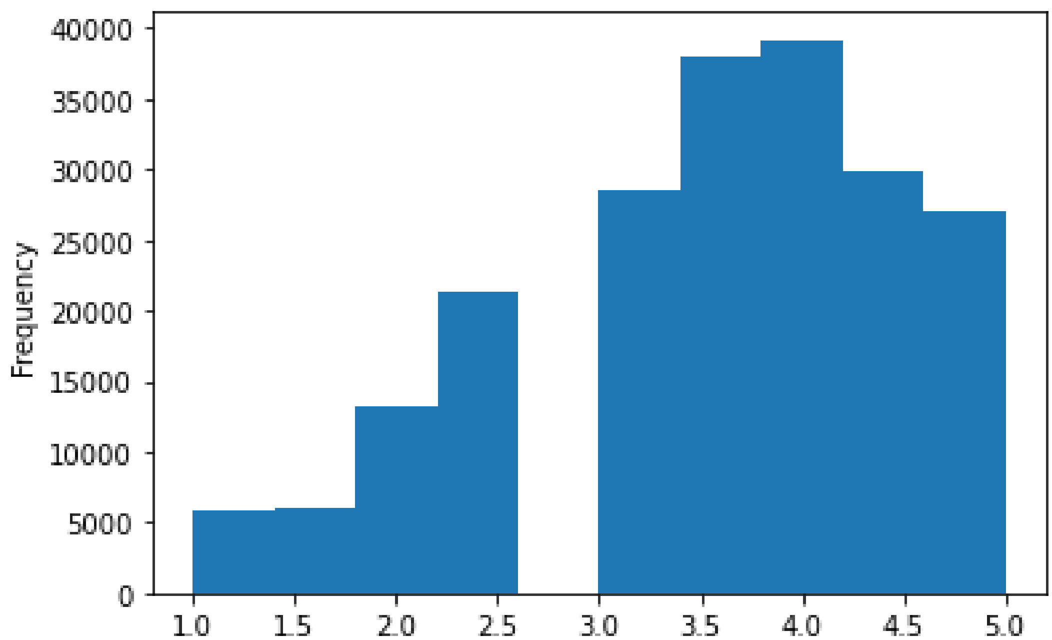


The trend is still the same, and it makes sense. The more businesses a city has, the more people it contains and thus, the higher total reviews for businesses in that city. It is not that people in some cities are less engaging.

Let's also check the star ratings given to the businesses.

[30]:
```
df['stars'].plot(kind='hist', bins=10)
```

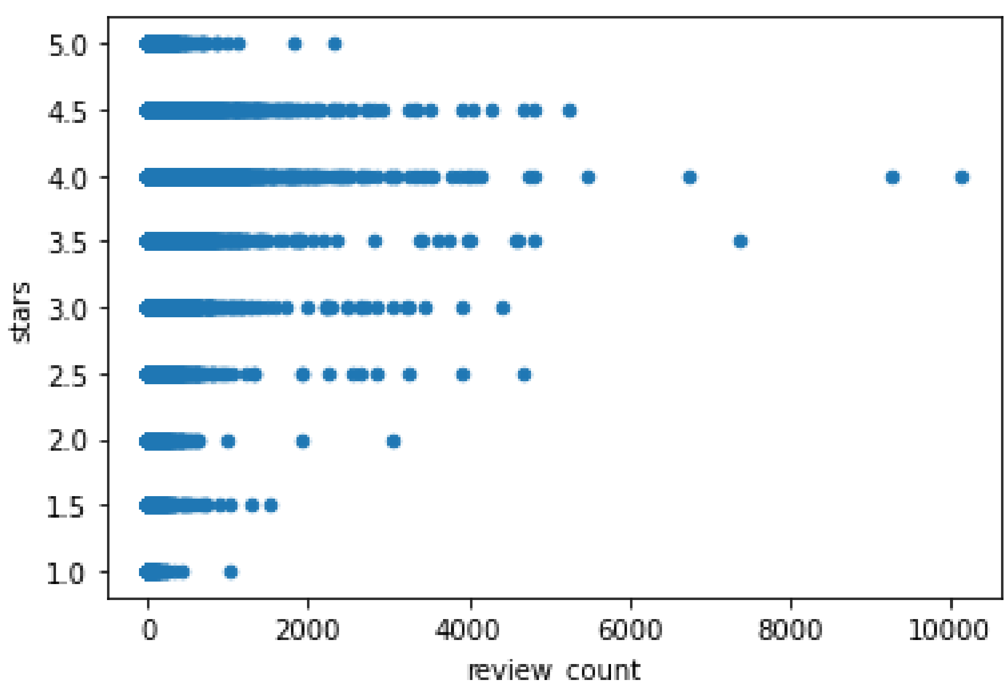
[30]: <AxesSubplot:ylabel='Frequency'>



In general, businesses have received 4 star ratings. Let's see how that correlates to the number of reviews.

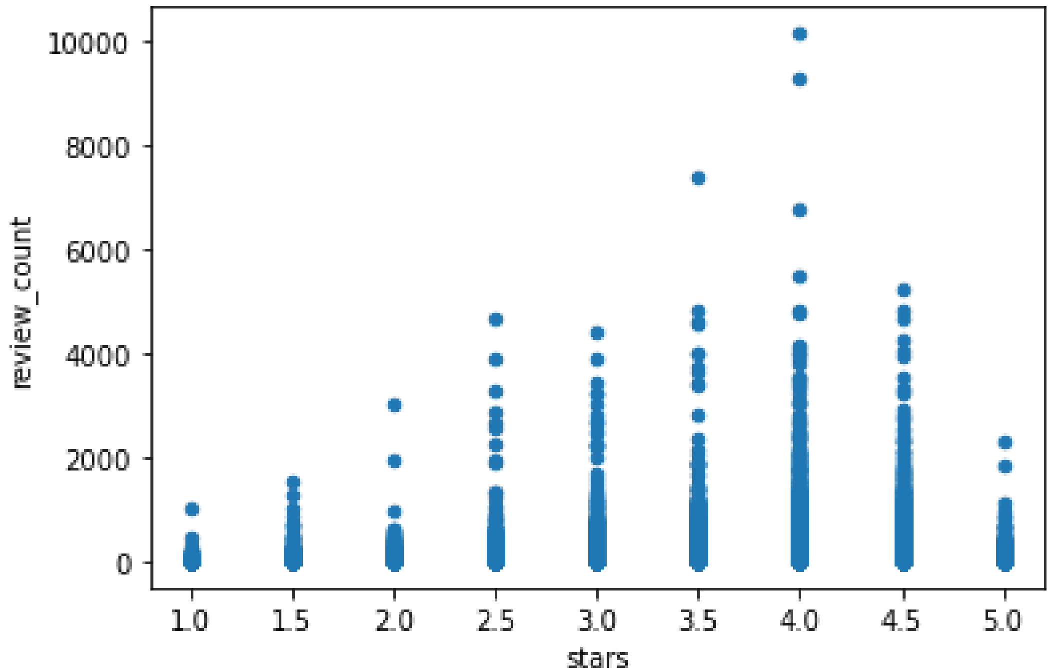
[31]:
```
df.plot(kind='scatter', x='review_count', y='stars')
```

[31]: <AxesSubplot:xlabel='review_count', ylabel='stars'>



[32]:
```
df.plot(kind='scatter', x='stars', y='review_count')
```

[32]: <AxesSubplot:xlabel='stars', ylabel='review_count'>



In general, it does not depend on how many reviews a business received. The star rating could actually reflect the general public liking of the place.

[34]:
```
df['categories']
```

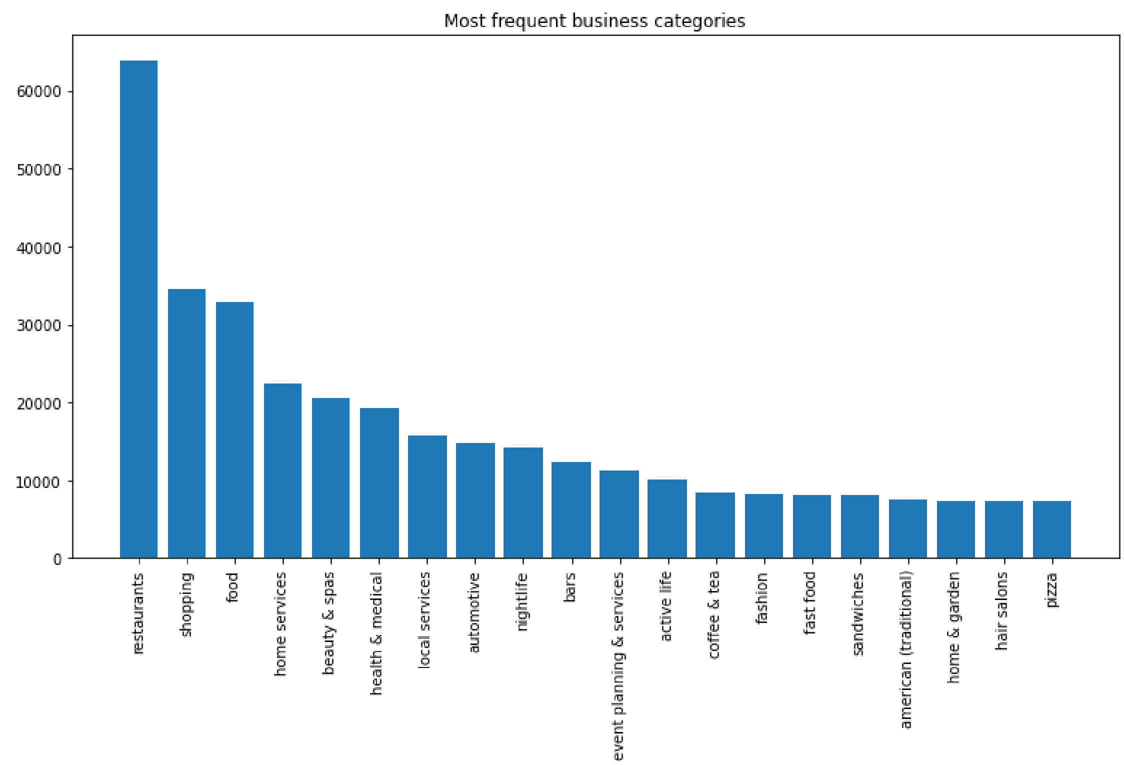
[34]:
```
0         Active Life, Gun/Rifle Ranges, Guns & Ammo, Sh...
1         Health & Medical, Fitness & Instruction, Yoga,...
2                                  Pets, Pet Services, Pet Groomers
3         Hardware Stores, Home Services, Building Suppl...
4         Home Services, Plumbing, Electricians, Handyma...
                                ...
209388                         Japanese, Sushi Bars, Restaurants
209389    Department Stores, Food, Mobile Phones, Fashio...
209390    American (New), Food, Burgers, Restaurants, Fa...
209391                          Pet Services, Pet Training, Pets
209392    Tax Services, Professional Services, Accountan...
Name: categories, Length: 209393, dtype: object
```

[35]:
```
from collections import Counter
categories = Counter()
df['categories'].str.lower().str.split(', ').apply(categories.update)
print(categories)
```

Let us plot the 20 most frequent business categories.

[36]:
```
lists = sorted(categories.items(), key=lambda item: item[1], reverse=True)
x, y = zip(*lists)
```

[37]:
```
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])
businesses = x[:20]
frequencies = y[:20]
ax.bar(businesses,frequencies)
plt.xticks(rotation='vertical')
plt.title('Most frequent business categories')
plt.show()
```

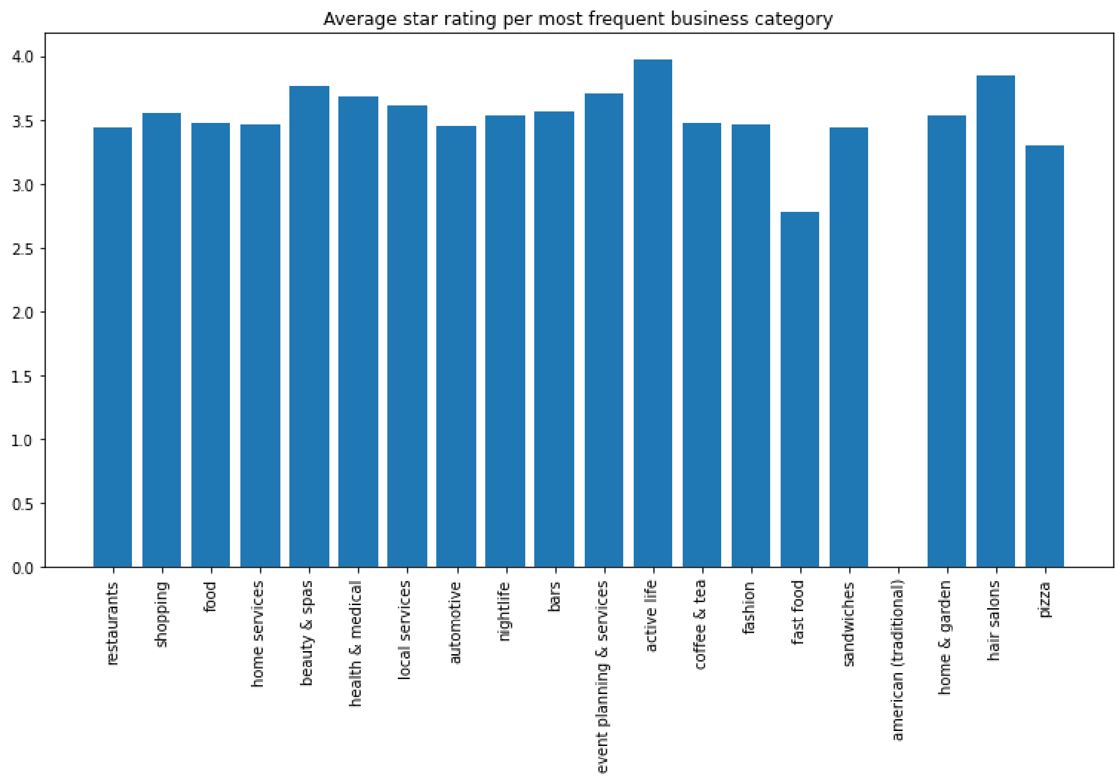


Restaurants, shopping, food and home services among others are the most frequent business categories. Let's see the average star rating these businesses receive.

[38]:
```
average_stars = []
for cat in x[:20]:
    average_stars.append(df[df['categories'].str.lower().str.contains(cat,␣
↪na=False)]['stars'].mean())
```

```
/opt/conda/lib/python3.7/site-packages/pandas/core/strings/accessor.py:101:
UserWarning: This pattern has match groups. To actually get the groups, use
str.extract.
  return func(self, *args, **kwargs)
```

[39]:
```
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])
ax.bar(businesses,average_stars)
plt.xticks(rotation='vertical')
plt.title('Average star rating per most frequent business category')
plt.show()
```

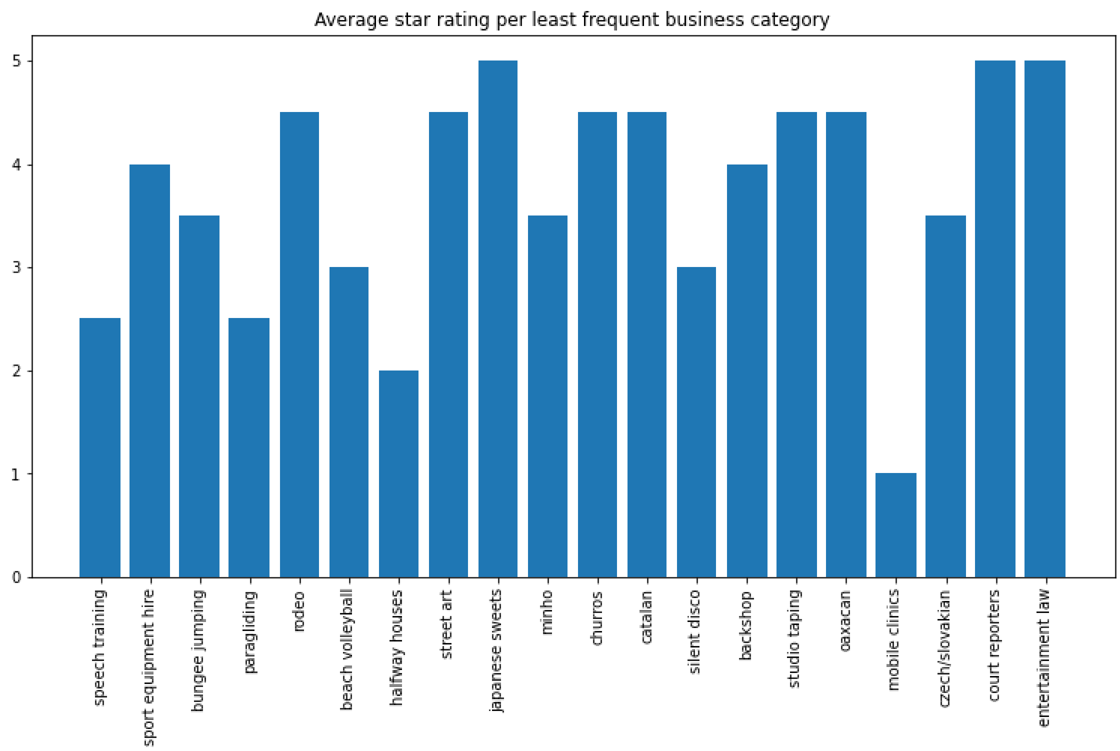


Businesses belonging to categories such as active life, hair salons and beauty and spas are ustuall rated higher in average. Let's see the average star ratings for least frequent categories.

[40]:
```
average_stars = []
for cat in x[-20:]:
```

```
    average_stars.append(df[df['categories'].str.lower().str.contains(cat,␣
↪na=False)]['stars'].mean())
```

[41]:
```
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])
businesses = x[-20:]
ax.bar(businesses,average_stars)
plt.xticks(rotation='vertical')
plt.title('Average star rating per least frequent business category')
plt.show()
```



It seems like businesses that are rare get better ratings. Japanese sweets, court reporters and entertainment law categories are highly rated.

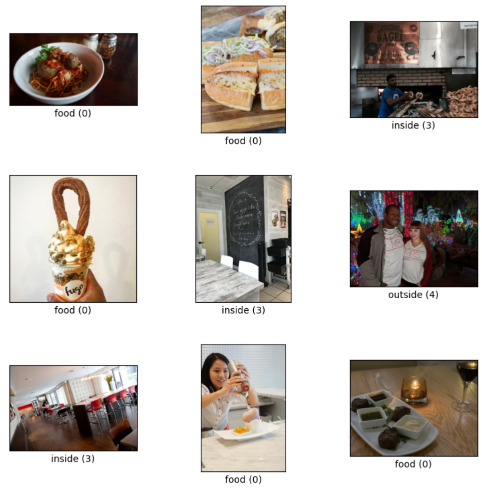
[ ]:
```

```

3 Machine Learning Experiments

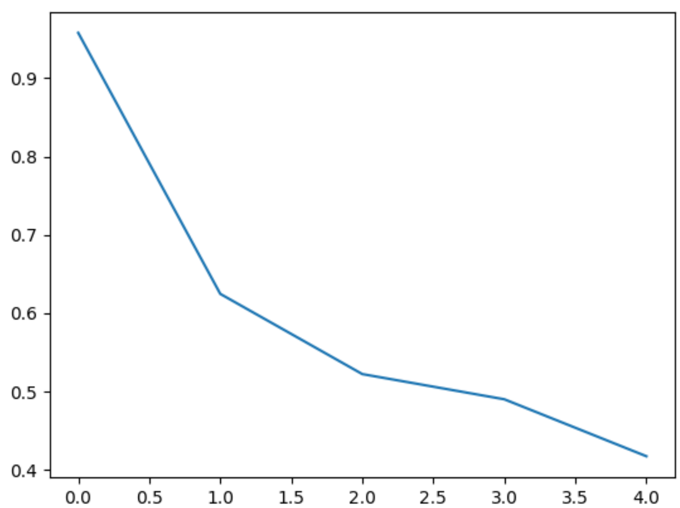
3.0.1 Image Classification

The first task carried out is image classification on yelp photos dataset. Photos are labeled into five classes including food, inside, outside, drink and menu. An example is shown in Figure 1

A pretrained MobileNetV2 model is used as feature extractor to train a multiclass classification



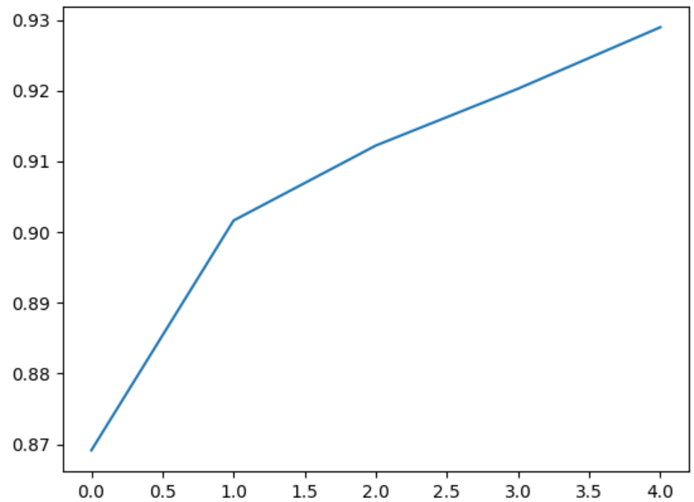
taking input images and predicting class labels. Details of training could be retrieved from the code accompanying this report. The loss and accuracy plots during training are shown in Figures below.



The trained model is tested on 200 examples from the test set, evaluation results are shown in Figure below.

3.0.2 Star Ratings Prediction

The second task carried out is based on reviews and star ratings users have given to businesses. Users write a review text about the business and give a rating ranging from 1 to 5 based on whether they like the place or not. In this section, text processing techniques in Sklearn and a bunch of



	precision	recall	f1-score	support
0.0	0.93	0.99	0.96	126
1.0	0.90	0.98	0.94	45
2.0	1.00	0.58	0.73	19
3.0	1.00	0.38	0.55	8
4.0	0.67	1.00	0.80	2
accuracy			0.93	200
macro avg	0.90	0.78	0.80	200
weighted avg	0.93	0.93	0.92	200

classical machine learning algorithms are used to predict the user's rating based on the review text.

Classifiers such as Random Forest, Nearest Neighbours, AdaBoost and decision trees are evaluated. Results of classifiers on the test set are shown in Figures below.

classifier: Nearest Neighbors

	precision	recall	f1-score	support
0	0.24	0.12	0.16	2748
1	0.21	0.00	0.01	1637
2	0.13	0.02	0.04	2236
3	0.24	0.25	0.25	4561
4	0.46	0.70	0.56	8818
accuracy			0.39	20000
macro avg	0.26	0.22	0.20	20000
weighted avg	0.32	0.39	0.33	20000

classifier: Random Forest

	precision	recall	f1-score	support
0	0.64	0.78	0.70	2748
1	0.55	0.05	0.10	1637
2	0.46	0.17	0.25	2236
3	0.44	0.32	0.37	4561
4	0.65	0.90	0.75	8818
accuracy			0.60	20000
macro avg	0.55	0.45	0.43	20000
weighted avg	0.57	0.60	0.55	20000

classifier: AdaBoost

	precision	recall	f1-score	support
0	0.60	0.67	0.63	2748
1	0.42	0.14	0.21	1637
2	0.41	0.23	0.30	2236
3	0.43	0.35	0.39	4561
4	0.65	0.84	0.73	8818
accuracy			0.58	20000
macro avg	0.50	0.45	0.45	20000
weighted avg	0.55	0.58	0.55	20000

classifier: Decision Tree

	precision	recall	f1-score	support
0	0.50	0.62	0.55	2748
1	0.17	0.11	0.14	1637
2	0.26	0.22	0.23	2236
3	0.34	0.31	0.32	4561
4	0.64	0.69	0.67	8818
accuracy			0.49	20000
macro avg	0.38	0.39	0.38	20000
weighted avg	0.47	0.49	0.48	20000

3.0.3 Recommender Systems

The third class of machine learning experiments conducted are recommendation systems. Based on users and their reviews given to businesses, Memory based recommendation algorithms (Nearest Neighbor) and Model Based Recommendation algorithms (SVD) are used to predict businesses that the user might probably like.

More details of the training and testing are given in the accompanying github repository