Coursework2_CST4050_

April 22, 2020

1 1. Introduction:

Yelp is a public company which publishes crowd-sourced reviews about local businesses, as well as the online reservation service Yelp Reservations. It also assists small businesses in how to respond to reviews. The company was founded in the year 2004 by former PayPal employees. By the year 2010 it had $30 million in revenues and the website had published more than 4.5 million crowd-sourced reviews.

2 2. Dataset Description

The data used here is collected from Yelp website (https://www.yelp.com/dataset). The data was uploaded there for learning/academic purposes. The data consists of many JSON files. There are 6 JSON files available on the website. For my project I have used 2 of them. One file consists of the business data. It contains many vital business data. The other file used in the analysis consists of the reviews received from the users. The details of the attributes of these two files are listed below:

Business Data:

”business_id”: It is a unique string string business id.

”name”: It is a string character. It is name of the business.

”address”: It is a string character, It is the full address of the business.

”city”: It is a string character, It consists of the city where the business is located.

”state”: It is a string character, It consists of the character state code, if applicable.

”postal code”: It is a string character, It consists of the postal code of the location of the business.

”latitude”: It is a float character. It is the latitude of the location.

”longitude”: It is a float character. It is the longitude of the location.

”stars”: It is a float character. It consists of the star rating, rounded to half-stars.

”review_count”: It is an integer character, it consists of a number of reviews.

”is_open”: It is an integer, 0 or 1 for closed or open, respectively.

”attributes”: It is an object, business attributes to values. note: some attribute values might be objects.

”categories”: It is an array of strings of business categories.

”hours”: It is an object of key day to value hours, hours are using a 24hr clock.

Review Data:

”review_id”: It uses a string character. It is an unique review id.

”user_id”: It is a string character. It is an unique user id that maps to the user in user.json.

”business_id”: It is a string character. It is a business id that maps to business in business.json.

”stars”: It is an integer that consists of the star rating.

”date”: It is a string character which is in date format YYYY-MM-DD.

”text”: It is a string character which consists of the reviews itself.

”useful”: It is an integer which consists of the number of useful votes received.

”funny”: It is an integer which consists of the number of funny votes received.

”cool”: It is an integer which consists of number of cool votes received.

[231]:
```
# Loading the required libraries

# numpy is the library imported for doing the linear algebra
import numpy as np

# pandas are used for data processing, JSON file I/O (e.g. pd.read_JSON)
import pandas as pd

import collections # this is used to store collections of data
import re, string # this is used for string searching and manipulation
import sys # the sys module provides information about constants, functions and
 ↪methods of python interpreter.
import time
```

[232]:
```
# The below function loads the JSON file and converts it into a dataframe in
 ↪pandas.
import json

def init_ds(json):
    ds= {}
    keys = json.keys()
    for k in keys:
        ds[k]= []
    return ds, keys

def read_json(file):
    dataset = {}
    keys = []
    with open(file,encoding="utf8") as file_lines:
```

```
        for count, line in enumerate(file_lines):
            data = json.loads(line.strip())
            if count ==0:
                dataset, keys = init_ds(data)
            for k in keys:
                dataset[k].append(data[k])

        return pd.DataFrame(dataset)
```

[3]:
```
yelp_business= read_json('yelp_academic_dataset_business.json')# The business␣
↪data file
yelp_review= read_json('yelp_academic_dataset_review.json') # The review data␣
↪file
```

First glance at the review file

[5]:
```
yelp_review.head()
```

[5]:
```
                review_id                 user_id             business_id  \
0  xQY8N_XvtGbearJ5X4QryQ  OwjRMXRC0KyPrIlcjaXeFQ  -MhfebM0QIsKt87iDN-FNw
1  UmFMZ8PyXZTY2QcwzsfQYA  nIJD_7ZXHq-FX8byPMOkMQ  lbrU8StCq3yDfr-QMnGrmQ
2  LG2ZaYiOgpr2DK_90pYjNw  V34qejxNsCbcgD8C0HVk-Q  HQl28KMwrEKHqhFrrDqVNQ
3  i6g_oA9Yf9Y31qt0wibXpw  ofKDkJKXSKZXu5xJNGiiBQ  5JxlZaqCnk1MnbgRirs40Q
4  6TdNDKywdbjoTkizeMce8A  UgMW8bLE0QMJDCkQ1Ax5Mg  IS4cv902ykd8wj1TR0N3-A

   stars  useful  funny  cool  \
0    2.0       5      0     0
1    1.0       1      1     0
2    5.0       1      0     0
3    1.0       0      0     0
4    4.0       0      0     0

                                                text                 date
0  As someone who has worked with many museums, I…  2015-04-15 05:21:16
1  I am actually horrified this place is still in…  2013-12-07 03:16:52
2  I love Deagan's. I do. I really do. The atmosp…  2015-12-05 03:18:11
3  Dismal, lukewarm, defrosted-tasting "TexMex" g…  2011-05-27 05:30:52
4  Oh happy day, finally have a Canes near my cas…  2017-01-14 21:56:57
```

First glance at the business file

[6]:
```
yelp_business.head()
```

[6]:
```
              business_id                      name  \
0  f9NumwFMBDn751xgFiRbNA  The Range At Lake Norman
1  Yzvjg0SayhoZgCljUJRF9Q         Carlos Santo, NMD
2  XNoUzKckATkOD1hP6vghZg                   Felinus
3  6OAZjbxqM5ol29BuHsil3w      Nevada House of Hose
```

```
4  51M2Kk903DFYI6gnB5I6SQ   USE MY GUY SERVICES LLC

                    address            city state postal_code   latitude  \
0           10913 Bailey Rd       Cornelius    NC       28031  35.462724
1  8880 E Via Linda, Ste 107      Scottsdale    AZ       85258  33.569404
2      3554 Rue Notre-Dame O        Montreal    QC     H4C 1P4  45.479984
3            1015 Sharp Cir  North Las Vegas    NV       89030  36.219728
4         4827 E Downing Cir            Mesa    AZ       85205  33.428065

    longitude  stars  review_count  is_open  \
0  -80.852612    3.5            36        1
1 -111.890264    5.0             4        1
2  -73.580070    5.0             5        1
3 -115.127725    2.5             3        0
4 -111.726648    4.5            26        1

                                          attributes  \
0  {'BusinessAcceptsCreditCards': 'True', 'BikePa…
1  {'GoodForKids': 'True', 'ByAppointmentOnly': '…
2                                               None
3  {'BusinessAcceptsCreditCards': 'True', 'ByAppo…
4  {'BusinessAcceptsCreditCards': 'True', 'ByAppo…

                                          categories  \
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…

                                               hours
0  {'Monday': '10:0-18:0', 'Tuesday': '11:0-20:0'…
1                                               None
2                                               None
3  {'Monday': '7:0-16:0', 'Tuesday': '7:0-16:0', …
4  {'Monday': '0:0-0:0', 'Tuesday': '9:0-16:0', '…
```

3 3. Machine learning challenge

The challenge here is to build a model to classify the Yelp Reviews into 1 star, 3 star or 5 star(Negative,average,good) categories based on the text content.

We can also use the model for Sentiment Analysis and Prediction of Review Ratings.

This can be achieved by performing machine learning for ”textual data analysis” . This allows us to extract and classify the reviews to make better predictions and create insights.

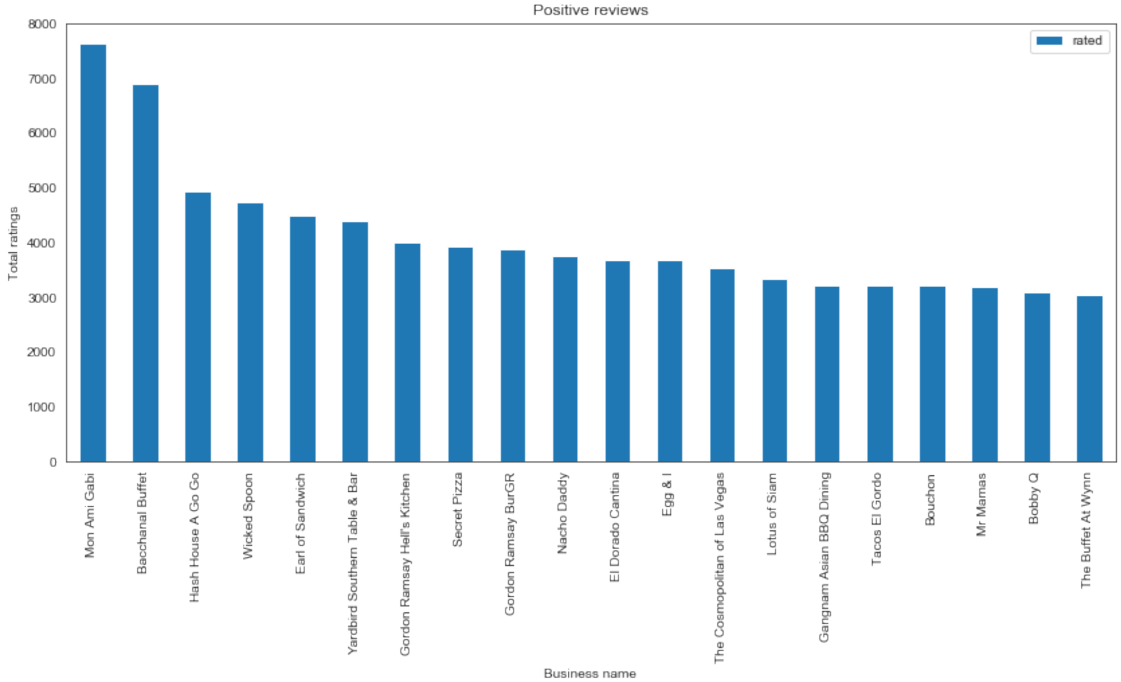
4 4. Better Understanding of the data

Before preparing the required dataset, I want to analyse what will be the best parameter to filter the data.

5 -- Top reviwed busines(by name)

[233]:
```
top_reviewed = yelp_review[yelp_review["stars"]>3]
top_reviews_dict ={}

for business_id in top_reviewed["business_id"].values:
    try :
        top_reviews_dict[business_id] =top_reviews_dict[business_id]+1
    except:
        top_reviews_dict[business_id]=1

topbusiness = pd.DataFrame.from_dict(data= top_reviews_dict,orient="index")

topbusiness.reset_index(inplace=True)
topbusiness.columns = ['business_id', 'rated']
del(top_reviews_dict)
del(top_reviewed)
```

[234]:
```
top_count= 20
right=pd.DataFrame(yelp_business[['business_id',"name","categories"]].values,
                    columns=['business_id',"Business name","categories"])

top_business_data = pd.merge(topbusiness,right=right,␣
 ↪how="inner",on='business_id')
top_business_data.sort_values("rated")[::-1][:top_count].plot(x="Business␣
 ↪name",y="rated",
                                                              kind="bar",figsize=(14,6),
                                                              title='Positive reviews').
 ↪set_ylabel("Total ratings")

del(topbusiness)
del(right)
```



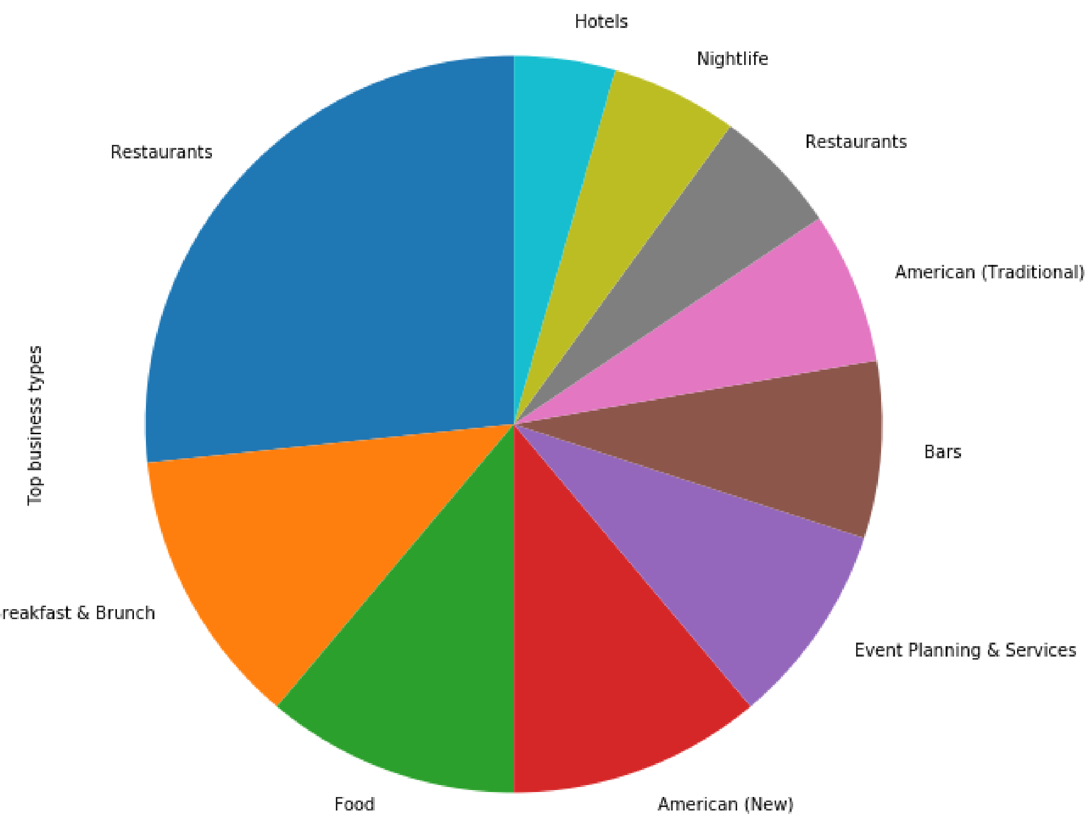
The top rewiwed business merchants are Mon Ami Gabi, Bacchanal Buffet, Hash House A Go Go. All these are restuarants.

6 -- Categories of top reviewed businesses

[9]:
```
num_cat =10 # to show top 10 catrgories
top_business = 30 # choose categories of top 30 businesses
cat_data = top_business_data.sort_values("rated")[::-1][:top_business]
#cat_data.categories
Categories={}
for cat in cat_data.categories.values:
    all_categories= cat.split(",")
    for x in all_categories:
        try :
            Categories[x] =Categories[x]+1
        except:
            Categories[x]=1
top_categories = pd.DataFrame.from_dict(data= Categories,orient="index")
top_categories.reset_index(inplace=True)
top_categories.columns = ['category', 'occurance']

x_val=top_categories.sort_values("occurance")[::-1][:num_cat].occurance.values
labels=top_categories.sort_values("occurance")[::-1][:num_cat].category.values
```

```
series = pd.Series(x_val, index=labels, name='Top business types')
series.plot.pie(figsize=(10, 10),startangle=90)
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x16ad62ff0c8>



Restuarants are the top reviwed business categories.

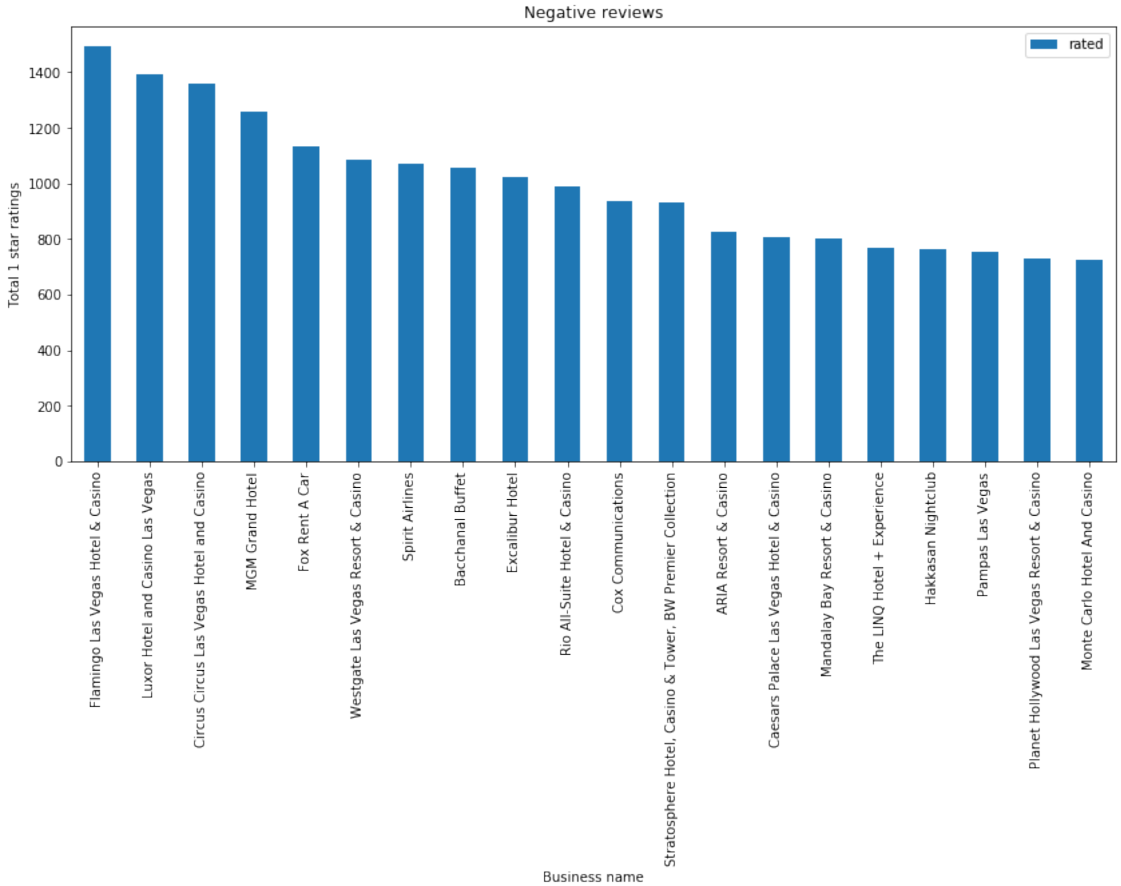
7 -- Negatively reviewed businesses

[10]:
```
bottom_reviewed = yelp_review[yelp_review["stars"]<2]
bottom_reviews_dict ={}

for business_id in bottom_reviewed["business_id"].values:
    try :
        bottom_reviews_dict[business_id] =bottom_reviews_dict[business_id]+1
    except:
```

```
        bottom_reviews_dict[business_id]=1

bottombusiness = pd.DataFrame.from_dict(data=␣
 ↪bottom_reviews_dict,orient="index")

bottombusiness.reset_index(inplace=True)
#bottombusiness.head()
bottombusiness.columns = ['business_id', 'rated']
```

[11]:

```
top_count= 20
right=pd.DataFrame(yelp_business[['business_id',"name","categories"]].values,
                    columns=['business_id',"Business name","categories"])

bottom_business_data = pd.merge(bottombusiness,right=right,␣
 ↪how="inner",on='business_id')
bottom_business_data.sort_values("rated")[::-1][:top_count].plot(x="Business␣
 ↪name",y="rated",
                                                   kind="bar",figsize=(14,6),
                                                   title='Negative reviews').
 ↪set_ylabel("Total 1 star ratings")

del(bottom_reviewed)
del(bottom_reviews_dict)
del(bottombusiness)
del(right)
```



Casinos are the negatively reviwed business.

8 5. Methodology

9 5.1 Data Collection

From the above analysis, it is clear that the restaurants are the top reviewed business.Therefore, for sentimental analysis I want to collect the reviews for ”Indian” restaurants only.

I will merge the two datafiles now. I want to develop the rating predictive model for the ”Indian” restaurants only based on their reviews.

Since I want to collect the reviews only for the ”Indian restaurants”, I have to extract the category column as individual elements. So that I can filter my data accordingly. Further I want to collect the data for the Indian restaurants which are still open.

[12]:
```
# The explode() function is used to transform each element of a list-like to a␣
↪row.
```

```
# To use this function, pandas version needs to be above 0.25.
# Checking the version of pandas.
pd.__version__
```

[12]: '0.25.1'

The explode() function is used to transform each element of a list-like to a row

[13]:
```
# Applying the explode function to the column "categories"
df_explode = yelp_business.assign(categories = yelp_business.categories
                          .str.split(', ')).explode('categories')
```

[14]:
```
#We can then list out all the individual category
df_explode.categories.value_counts()
```

[14]:
```
Restaurants       63944
Shopping          34644
Food              32991
Home Services     22487
Beauty & Spas     20520
                  …
Halfway Houses        1
Street Art            1
Osteopaths            1
Tonkatsu              1
Bungee Jumping        1
Name: categories, Length: 1336, dtype: int64
```

I want to get the reviews for the Indian Restaurants

[15]:
```
#Find the categories containing Indian Restaurants
df_explode[df_explode.categories.str.contains('Indian',
                      case=True,na=False)].categories.value_counts()
```

[15]:
```
Indian    1652
Name: categories, dtype: int64
```

[16]:
```
print('Total number of categories',len(df_explode.categories.value_counts()))
```

```
Total number of categories 1336
```

[17]:
```
# Finding the categories that contains Restaurants
df_explode[df_explode['categories'].str.contains('Indian',case=True,na=False)].
 ↪categories.value_counts()
```

[17]:
```
Indian    1652
Name: categories, dtype: int64
```

[18]:
```python
# Extracting the data only for the Indian Restaurants from the buiness file.
business_Restaurants = yelp_business[yelp_business['categories'].str.contains(
                'Indian',case=False, na=False)]
```

[19]:
```python
business_Restaurants.head()
```

[19]:
```
                business_id                          name  \
82   RrapAhd8ZxCj-iue7fu9FA             Ganga Restaurant
218  Yy_SB9r7VQTlIVv_MeOmaA                    Cari Mela
366  LZWXP-D4YPlzsFjVx6b9XA                  Spice South
511  CvOxwAhmwwy3sQRKYVAjGg  Bombay Buffet Indian Cuisine
525  IjYGSm4_DjP6VULpP27o_g                My Roti Place

               address         city state postal_code   latitude  \
82     515 4th Avenue SW      Calgary    AB     T2P 0J8  51.049407
218      2556 Rue Centre     Montréal    QC     H3K 1J8  45.478639
366  8145 Ardrey Kell Rd    Charlotte    NC       28277  35.039010
511     795 Markham Road  Scarborough    ON     M1H 2Y1  43.767071
525   948 Queen Street E      Toronto    ON     M4M 1J7  43.660961

      longitude  stars  review_count  is_open  \
82  -114.072656    1.5             3        1
218  -73.568839    3.5             9        1
366  -80.793656    3.5            20        0
511  -79.228062    2.5            22        0
525  -79.341156    4.0             8        1

                                            attributes  \
82   {'RestaurantsReservations': 'True', 'WiFi': 'u…
218  {'BusinessParking': '{'garage': False, 'street…
366  {'OutdoorSeating': 'True', 'RestaurantsGoodFor…
511  {'Caters': 'False', 'RestaurantsPriceRange2': …
525  {'OutdoorSeating': 'True', 'GoodForMeal': '{'d…

                                            categories  \
82                 Restaurants, Buffets, Indian, Halal
218                                Indian, Restaurants
366                                Indian, Restaurants
511  Restaurants, Indian, Event Planning & Services…
525                  Indian, Restaurants, Fast Food

                                                 hours
82   {'Monday': '17:0-22:0', 'Tuesday': '17:0-22:0'…
218  {'Monday': '11:30-22:0', 'Tuesday': '11:30-22:…
366  {'Tuesday': '11:30-21:30', 'Wednesday': '11:30…
511                                               None
525  {'Monday': '11:0-21:0', 'Tuesday': '11:0-21:0'…
```

[20]:
```
# Collecting the data only for the Indian restaurants which are still open.
# 1 = open, 0 = closed
business_Restaurants = business_Restaurants[business_Restaurants['is_open']==1]
```

[21]:
```
# Remove the columns that are not required in business_Restaurants
drop_columns = ['city','state','postal_code','latitude','longitude','hours']
business_Restaurants= business_Restaurants.drop(drop_columns, axis=1)
business_Restaurants.head()
```

[21]:
```
                business_id                    name  \
82   RrapAhd8ZxCj-iue7fu9FA       Ganga Restaurant
218  Yy_SB9r7VQTlIVv_MeOmaA               Cari Mela
525  IjYGSm4_DjP6VULpP27o_g           My Roti Place
650  ZDx7kt4aOPTlmYTqXDrTGA             Canbe Foods
969  RE4qn28MEiDrM1PbdYVgxA  Lena's Roti & Doubles

                                      address  stars  review_count  is_open  \
82                          515 4th Avenue SW    1.5             3        1
218                          2556 Rue Centre    3.5             9        1
525                         948 Queen Street E    4.0             8        1
650                        336 Rossland Road E    4.0            15        1
969  100 Maritime Ontario Boulevard, Unit 69    4.0            20        1

                                           attributes  \
82   {'RestaurantsReservations': 'True', 'WiFi': 'u…
218  {'BusinessParking': '{'garage': False, 'street…
525  {'OutdoorSeating': 'True', 'GoodForMeal': '{'d…
650  {'Caters': 'True', 'WheelchairAccessible': 'Tr…
969  {'RestaurantsGoodForGroups': 'True', 'GoodForK…

                                 categories
82   Restaurants, Buffets, Indian, Halal
218                  Indian, Restaurants
525       Indian, Restaurants, Fast Food
650      Indian, Restaurants, Sri Lankan
969       Caribbean, Restaurants, Indian
```

Review data file is a huge file. If we try to load all the data at once, it is likely to crash the memory of the computer. Therefore we will load large data by segmenting the file into smaller chunks. Also to reduce the memory usage I am identifying the datatype of each column.

[22]:
```
# To reduce the memory usage identifying the datatype of each column
size = 100000
review = pd.read_json('yelp_academic_dataset_review.json', lines=True,
                      dtype={'review_id':str,'user_id':str,
                             'business_id':str,'stars':int,
                             'date':str,'text':str,'useful':int,
```

```
                                        'funny':int,'cool':int},
                             chunksize=size)
```

[23]:
```
# There are multiple chunks to be read
chunk_list = []
for chunk_review in review:
    # Drop columns that aren't needed
    chunk_review = chunk_review.drop(['review_id','date'], axis=1)
    # Renaming column name to avoid conflict with business overall star rating
    chunk_review = chunk_review.rename(columns={'stars': 'review_stars'})
    # Inner merge with edited business file so only reviews related to the␣
 ↪business remain
    chunk_merged = pd.merge(business_Restaurants, chunk_review,␣
 ↪on='business_id', how='inner')
    # Show feedback on progress
    print(f"{chunk_merged.shape[0]} out of {size:,} related reviews")
    chunk_list.append(chunk_merged)
# After trimming down the review file, concatenate all relevant data back to one␣
 ↪dataframe
df = pd.concat(chunk_list, ignore_index=True, join='outer', axis=0)
```

```
787 out of 100,000 related reviews
780 out of 100,000 related reviews
800 out of 100,000 related reviews
793 out of 100,000 related reviews
810 out of 100,000 related reviews
862 out of 100,000 related reviews
867 out of 100,000 related reviews
905 out of 100,000 related reviews
892 out of 100,000 related reviews
957 out of 100,000 related reviews
934 out of 100,000 related reviews
838 out of 100,000 related reviews
814 out of 100,000 related reviews
778 out of 100,000 related reviews
801 out of 100,000 related reviews
958 out of 100,000 related reviews
1464 out of 100,000 related reviews
1430 out of 100,000 related reviews
1419 out of 100,000 related reviews
1460 out of 100,000 related reviews
1298 out of 100,000 related reviews
1105 out of 100,000 related reviews
985 out of 100,000 related reviews
964 out of 100,000 related reviews
1014 out of 100,000 related reviews
1082 out of 100,000 related reviews
```

```
1049 out of 100,000 related reviews
1047 out of 100,000 related reviews
1009 out of 100,000 related reviews
1083 out of 100,000 related reviews
1071 out of 100,000 related reviews
1061 out of 100,000 related reviews
670 out of 100,000 related reviews
580 out of 100,000 related reviews
642 out of 100,000 related reviews
591 out of 100,000 related reviews
586 out of 100,000 related reviews
681 out of 100,000 related reviews
654 out of 100,000 related reviews
593 out of 100,000 related reviews
910 out of 100,000 related reviews
1109 out of 100,000 related reviews
1083 out of 100,000 related reviews
1036 out of 100,000 related reviews
1035 out of 100,000 related reviews
969 out of 100,000 related reviews
856 out of 100,000 related reviews
783 out of 100,000 related reviews
738 out of 100,000 related reviews
692 out of 100,000 related reviews
734 out of 100,000 related reviews
722 out of 100,000 related reviews
723 out of 100,000 related reviews
773 out of 100,000 related reviews
881 out of 100,000 related reviews
860 out of 100,000 related reviews
1081 out of 100,000 related reviews
1055 out of 100,000 related reviews
1082 out of 100,000 related reviews
1134 out of 100,000 related reviews
1039 out of 100,000 related reviews
1033 out of 100,000 related reviews
935 out of 100,000 related reviews
895 out of 100,000 related reviews
922 out of 100,000 related reviews
901 out of 100,000 related reviews
897 out of 100,000 related reviews
905 out of 100,000 related reviews
933 out of 100,000 related reviews
943 out of 100,000 related reviews
834 out of 100,000 related reviews
840 out of 100,000 related reviews
1168 out of 100,000 related reviews
1284 out of 100,000 related reviews
```

```
1236 out of 100,000 related reviews
1246 out of 100,000 related reviews
1139 out of 100,000 related reviews
1133 out of 100,000 related reviews
823 out of 100,000 related reviews
863 out of 100,000 related reviews
178 out of 100,000 related reviews
```

We have finally collected our required data for the application of our machine learning models (Sentimental analysis)..

[24]:
```
# To view the collected data
df.head()
```

[24]:
```
              business_id                                name  \
0  RrapAhd8ZxCj-iue7fu9FA                    Ganga Restaurant
1  Yy_SB9r7VQTlIVv_MeOmaA                           Cari Mela
2  ZDx7kt4aOPTlmYTqXDrTGA                         Canbe Foods
3  RE4qn28MEiDrM1PbdYVgxA                Lena's Roti & Doubles
4  _PKXarw3GjlbwbXhjdpUMA  Tangra Villa Hakka Chinese Cuisine

                                  address  stars  review_count  is_open  \
0                       515 4th Avenue SW    1.5             3        1
1                         2556 Rue Centre    3.5             9        1
2                     336 Rossland Road E    4.0            15        1
3  100 Maritime Ontario Boulevard, Unit 69    4.0            20        1
4               411 Manhattan Drive, Suite 3C    3.5            29        1

                                          attributes  \
0  {'RestaurantsReservations': 'True', 'WiFi': 'u…
1  {'BusinessParking': '{'garage': False, 'street…
2  {'Caters': 'True', 'WheelchairAccessible': 'Tr…
3  {'RestaurantsGoodForGroups': 'True', 'GoodForK…
4  {'RestaurantsAttire': 'u'casual'', 'Caters': '…

                          categories                 user_id  review_stars  \
0  Restaurants, Buffets, Indian, Halal  pBUsRjJLTN-TVsoIv5ue8w             3
1                  Indian, Restaurants  h1UcaSPIPpQqMiWf12Csqw             4
2      Indian, Restaurants, Sri Lankan  zFGpxwJewI6OjC2u9EnZ-g             5
3       Caribbean, Restaurants, Indian  F-nLfSJ4qtdZRwNW993lSw             4
4         Chinese, Indian, Restaurants  R_TJQ6Hy1BtOYK_hCl25bQ             5

   useful  funny  cool                                               text
0       0      0     0  Ordered biryani for tonight's supper. I wasn't…
1       2      1     1  We started with a mix platter which had all ki…
2       7      1     2  You know this place is good when most of the p…
3       0      0     0  Visited this place with my Office colleagues, …
4       1      1     0  So spicy and flavorful! Shrimp Pakoras really …
```

10 5.2 Data Analysis Process

11 5.2.1 Data Preprocessing

The preprocessing of the data includes removing the unwanted columns from the dataset. It also includes eliminating stop words and punctuations. This is achieved by using nltk in python. It stands for natural language Toolkit. It is a suite of libraries and programs for symbolic and statistical natural language processing. I have also created a new column in the name as "text length". It returns the number of words in the column "text". Since, text data requires special preparation before we start it for predictive modeling, text must be parsed to remove words, called tokenization. For the input to a machine learning algorithm, the words need to be encoded as integers or floating point values. This is called feature extraction (or vectorization).

[25]:
```
# To get the information about the object types in the new data.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75517 entries, 0 to 75516
Data columns (total 14 columns):
business_id     75517 non-null object
name            75517 non-null object
address         75517 non-null object
stars           75517 non-null float64
review_count    75517 non-null int64
is_open         75517 non-null int64
attributes      75375 non-null object
categories      75517 non-null object
user_id         75517 non-null object
review_stars    75517 non-null int32
useful          75517 non-null int32
funny           75517 non-null int32
cool            75517 non-null int32
text            75517 non-null object
dtypes: float64(1), int32(4), int64(2), object(7)
memory usage: 6.9+ MB
```

[26]:
```
# To get a statics information on the numerical column of the data
df.describe()
```

[26]:

	stars	review_count	is_open	review_stars	useful \
count	75517.000000	75517.000000	75517.0	75517.000000	75517.000000
mean	3.790239	379.438789	1.0	3.794391	1.150800
std	0.544166	588.797230	0.0	1.379577	3.849485
min	1.000000	3.000000	1.0	1.000000	0.000000
25%	3.500000	75.000000	1.0	3.000000	0.000000
50%	4.000000	171.000000	1.0	4.000000	0.000000
75%	4.000000	386.000000	1.0	5.000000	1.000000

```
max        5.000000   2855.000000      1.0      5.000000    758.000000

              funny          cool
count  75517.000000  75517.000000
mean       0.402294      0.506826
std        3.224632      2.318385
min        0.000000      0.000000
25%        0.000000      0.000000
50%        0.000000      0.000000
75%        0.000000      0.000000
max      786.000000    321.000000
```

[27]:
```
# Removing the un-wanted columns from the dataset
drop_columns =␣
↪['name','address','stars','review_count','is_open','attributes','categories']
df= df.drop(drop_columns, axis=1)
```

[28]:
```
# Creating a new column which gives the number of words in the text column
df['text length'] = df['text'].apply(len)
df.head()
```

[28]:
```
              business_id                 user_id  review_stars  useful  \
0  RrapAhd8ZxCj-iue7fu9FA  pBUsRjJLTN-TVsoIv5ue8w             3       0
1  Yy_SB9r7VQTlIVv_MeOmaA  h1UcaSPIPpQqMiWf12Csqw             4       2
2  ZDx7kt4aOPTlmYTqXDrTGA  zFGpxwJewI6OjC2u9EnZ-g             5       7
3  RE4qn28MEiDrM1PbdYVgxA  F-nLfSJ4qtdZRwNW993lSw             4       0
4  _PKXarw3GjlbwbXhjdpUMA  R_TJQ6Hy1BtOYK_hCl25bQ             5       1

   funny  cool                                               text  text length
0      0     0  Ordered biryani for tonight's supper. I wasn't…          475
1      1     1  We started with a mix platter which had all ki…          323
2      1     2  You know this place is good when most of the p…         1929
3      0     0  Visited this place with my Office colleagues, …          419
4      1     0  So spicy and flavorful! Shrimp Pakoras really …          320
```

[29]:
```
# To check if there is any missing data in the new dataFrame
df.isnull()
```

[29]:
```
       business_id  user_id  review_stars  useful  funny   cool   text  \
0            False    False         False   False  False  False  False
1            False    False         False   False  False  False  False
2            False    False         False   False  False  False  False
3            False    False         False   False  False  False  False
4            False    False         False   False  False  False  False
…                …        …             …       …      …      …      …
75512        False    False         False   False  False  False  False
75513        False    False         False   False  False  False  False
```

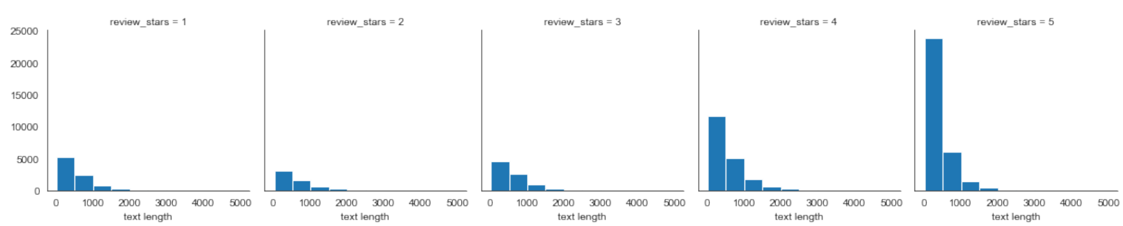
```
75514       False   False       False  False  False  False  False
75515       False   False       False  False  False  False  False
75516       False   False       False  False  False  False  False

       text length
0            False
1            False
2            False
3            False
4            False
…            …
75512        False
75513        False
75514        False
75515        False
75516        False

[75517 rows x 8 columns]
```

12 5.2.2 Data Visualisation

[30]:
```
# Importing the libraries to visualise the data
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
%matplotlib inline
```

[31]:
```
# Using FacetGrid from the seaborn library created a grid of 5 histograms of␣
↪text length based on the star ratings
g = sns.FacetGrid(df,col='review_stars')
g.map(plt.hist,'text length')
```

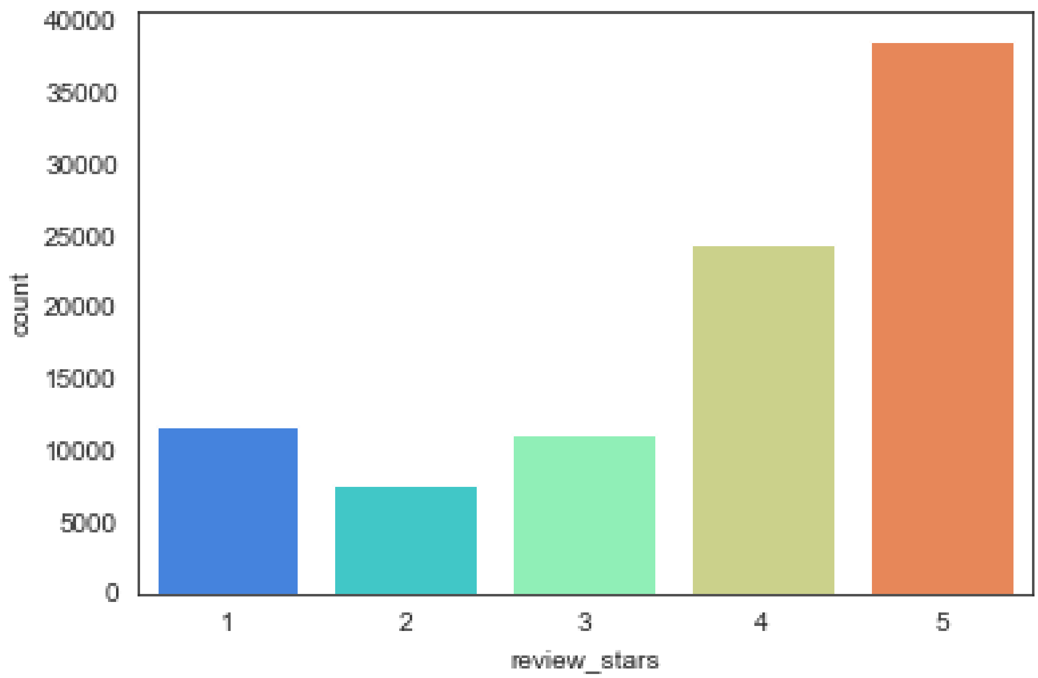
[31]: <seaborn.axisgrid.FacetGrid at 0x16b112a54c8>



The above graph shows that the more is the review__stars the lenght of the text lies between 0-1000

[47]:
```
# Counting the number of occurrences for each type of star rating
sns.countplot(x='review_stars',data=df,palette='rainbow')
```

[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1dff2cca748>



The above graph shows that the Indian restaurants have recieved more positive reviews

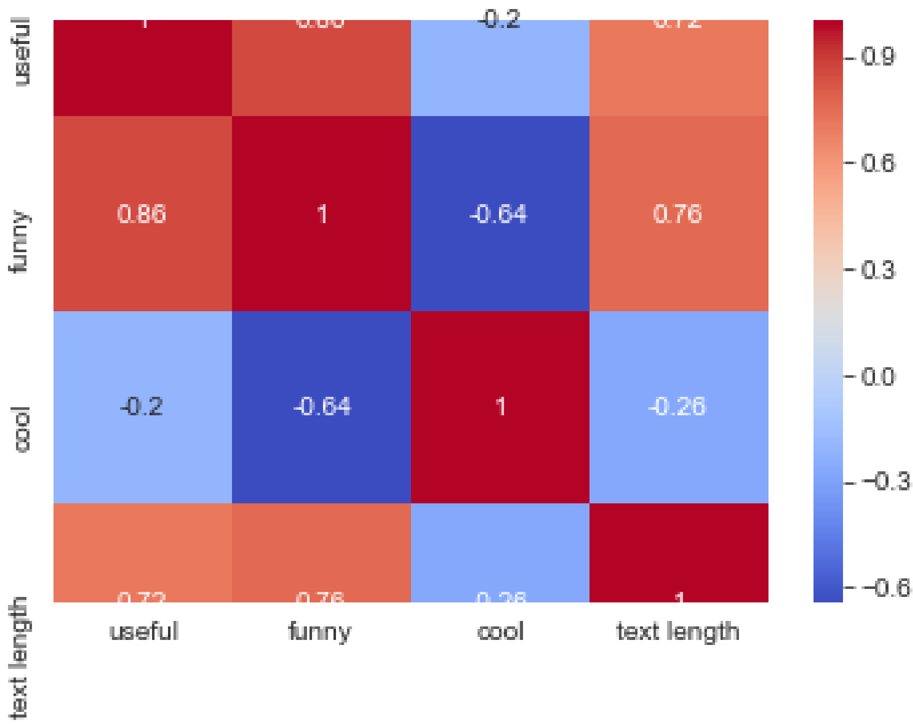
[48]:
```
# calculating the mean values of the numerical columns, grouping it by the␣
↪category, stars
stars = df.groupby('review_stars').mean()
stars
```

[48]:

review_stars	useful	funny	cool	text length
1	1.349011	0.624841	0.210544	611.460820
2	1.303633	0.552373	0.319942	686.313710
3	1.216826	0.465370	0.542514	669.506585
4	1.335686	0.447595	0.723777	580.924491
5	1.037770	0.295098	0.501137	435.432785

[49]:
```
# Visualising the correlation between the dataframe stars
sns.heatmap(stars.corr(),cmap='coolwarm',annot=True)
```

[49]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfea247888>



13 5.2.3 Sentiment Detection

[37]:
```
# Classifying the dataset and splitting it into the reviews and stars.
# Here, we will classify the dataset into 3 types of rating. Rating 1 =␣
↪"negative" , 3 ="Average", and 5 ="Positive".
data_class = df[(df.review_stars==1) | (df.review_stars==3) | (df.
↪review_stars==5)]
```

[38]:
```
# Creating the feature and target. x is the 'text' column of data_class and y is␣
↪the 'stars' column of data_class.
X = data_class['text']
y = data_class['review_stars']
print(X.head())
print(y.head())
```

```
0    Ordered biryani for tonight's supper. I wasn't t…
2    You know this place is good when most of the p…
4    So spicy and flavorful! Shrimp Pakoras really …
5    This is a good take-out place for hakka food i…
6    Th service is really good and the owners are a…
Name: text, dtype: object
0    3
```

```
2    5
4    5
5    3
6    3
Name: review_stars, dtype: int32
```

14 5.2.4 Preparing the data for predictive Modelling

Below I have defined a function that will clean the dataset by removing stop words and punctuations. nltk stands for natural language Toolkit. It is a suite of libraries and programs for symbolic and statistical natural language processing.

[99]:
```
import nltk
from nltk.corpus import stopwords

# CLEANING THE REVIEWS - REMOVAL OF STOPWORDS AND PUNCTUATION
def text_process(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.
 ↪words('english')]
```

[101]:
```
# Using the fit_transform method on the CountVectorizer object and passing the␣
 ↪'text' column. Saved the result by overwriting x.
vocab = CountVectorizer(analyzer=text_process).fit(X)
x = vocab.transform(X)
```

15 5.3 Train Test Split

In order to apply the models, we have to split the data into train and test data. Here, I have divided the data into 70-30. I have taken 70% of the data as a train and remaining 30% of the data as test.

[103]:
```
from sklearn.model_selection import train_test_split
```

[104]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size=0.
 ↪3,random_state=101)
```

16 5.4 Applying the Machine Learning Models

In this case, we are applying classification learning algorithms. It is a supervised learning process. It is a process where the computer program learns from from the input and then uses this to classify new observations.

The basic idea for opting a classification model is when the target variable is categorical. In this case, we are trying to predict the rating of a review. We have got 5 ratings(0,1,2,3,4,5) out of which we need to predict one. Therefore, classification models can help us in this.

In my analysis, I have used Naive Bayes Classifier, Random Forest Classifier, Decision Trees, K-Nearest Neighbor, RNN Model from Neural Networks.

17 5.4.1 Training a Model using Naive Bayes classifier

Firstly, I want to establish a model that will act as a baseline. In general terms a linear model is appropriate and has the advantage of being fast to train.

Here, I have used Multinomial Naive Bayes over Gaussian because with a sparse data, Gaussian Naive Bayes assumptions aren't met. A simple gaussian fit over the data will not give us a good fit or prediction.

Naive Bayes classification technique is based on Bayes' Theorem with the assumption of independence among predictors. This classifier is easy to build. It is known to be a good fit for very large datasets due to its high scalability.

[105]:
```
# Importing the required libraries.
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score,␣
 ↪roc_curve
from sklearn.metrics import classification_report
```

[106]:
```
# Multinomial Naive Bayes
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
predmnb = mnb.predict(x_test)
print("Confusion Matrix for Multinomial Naive Bayes:")
print(confusion_matrix(y_test,predmnb))
print("Score:",round(accuracy_score(y_test,predmnb)*100,2))
print("Classification Report:",classification_report(y_test,predmnb))
```

```
Confusion Matrix for Multinomial Naive Bayes:
[[2215  377  194]
 [ 339 1445  826]
 [ 101  263 9308]]
Score: 86.06
Classification Report:               precision    recall  f1-score   support

           1       0.83      0.80      0.81      2786
           3       0.69      0.55      0.62      2610
           5       0.90      0.96      0.93      9672

    accuracy                           0.86     15068
   macro avg       0.81      0.77      0.79     15068
```

```
weighted avg       0.85      0.86      0.85     15068
```

The performance score of Naive Bayes classifier is 86.06. Since it is high score, I will treat this model as my baseline.

18 5.4.2 Random Forest Classifier

There is no correlation between our feature(text) and target(review_stars) and this is the reason for choosing Random Forest Classifier. The vital thing for a Random Forest Classifier model to make an accurate class prediction is the trees of the forest and more importantly their predictions need to be uncorrelated (or at least have low correlations with each other).

Random forests are an ensemble learning method for classification. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

[107]:
```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
rmfr = RandomForestClassifier()
rmfr.fit(x_train,y_train)
predrmfr = rmfr.predict(x_test)
print("Confusion Matrix for Random Forest Classifier:")
print(confusion_matrix(y_test,predrmfr))
print("Score:",round(accuracy_score(y_test,predrmfr)*100,2))
print("Classification Report:",classification_report(y_test,predrmfr))
```

```
C:\Users\sushb\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
Confusion Matrix for Random Forest Classifier:
[[1774  269  743]
 [ 419  690 1501]
 [ 139  259 9274]]
Score: 77.9
Classification Report:               precision    recall  f1-score   support

           1       0.76      0.64      0.69      2786
           3       0.57      0.26      0.36      2610
           5       0.81      0.96      0.88      9672

    accuracy                           0.78     15068
   macro avg       0.71      0.62      0.64     15068
weighted avg       0.76      0.78      0.75     15068
```

The performance score of Random Forest Classifier is 77.9.

19 5.4.3 Decision Tree

Decision Trees are of two types a) classification b) regression. Since the decision variable(target) is categorical/discrete we will be using decision tree classifier. It builds the model in the form of tree structure. The classifier breaks down a data set into smaller subsets and at the same time an associated decision tree is incrementally developed. Decision trees can handle both categorical and numerical data.

[108]:
```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
preddt = dt.predict(x_test)
print("Confusion Matrix for Decision Tree:")
print(confusion_matrix(y_test,preddt))
print("Score:",round(accuracy_score(y_test,preddt)*100,2))
print("Classification Report:",classification_report(y_test,preddt))
```

```
Confusion Matrix for Decision Tree:
[[1814  510  462]
 [ 481 1079 1050]
 [ 366  804 8502]]
Score: 75.62
Classification Report:               precision    recall  f1-score   support

           1       0.68      0.65      0.67      2786
           3       0.45      0.41      0.43      2610
           5       0.85      0.88      0.86      9672

    accuracy                           0.76     15068
   macro avg       0.66      0.65      0.65     15068
weighted avg       0.75      0.76      0.75     15068
```

The performance score of Decision Tree model is 75.62

20 5.4.4 K Nearest Neighbour Algorithm

KNN is known as a non-parametric and lazy learning algorithm. It is a supervised classification technique that uses proximity as a proxy for 'sameness'. This algorithm takes a bunch of labelled points and uses them to learn how to label other points. To label a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors). Closeness is typically expressed in terms of a dissimilarity function. Once it checks with 'k' number of nearest neighbors, it assigns a label based on whichever label the most of the neighbors have.

[109]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
```

```
knn.fit(x_train,y_train)
predknn = knn.predict(x_test)
print("Confusion Matrix for K Neighbors Classifier:")
print(confusion_matrix(y_test,predknn))
print("Score: ",round(accuracy_score(y_test,predknn)*100,2))
print("Classification Report:")
print(classification_report(y_test,predknn))
```

```
Confusion Matrix for K Neighbors Classifier:
[[ 652   60 2074]
 [ 113  137 2360]
 [  56   48 9568]]
Score:  68.74
Classification Report:
              precision    recall  f1-score   support

           1       0.79      0.23      0.36      2786
           3       0.56      0.05      0.10      2610
           5       0.68      0.99      0.81      9672

    accuracy                           0.69     15068
   macro avg       0.68      0.43      0.42     15068
weighted avg       0.68      0.69      0.60     15068
```

The performance score of K Neighbors Classifier is 68.74

21 5.4.5 RNN Model

The RNN is an expressive model that is known to learn highly complex relationships from an arbitrarily long sequence of data. It maintains a vector of activation units for each element in the data sequence, this makes RNN very deep. The depth of RNN leads to two well-known issues, the exploding and the vanishing gradient problems.

There are many ways to implement a neural network in python. Here, I will be using tensorflow/keras.

[110]:
```
# Importing the libraries
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

[111]:
```
tk = Tokenizer(lower = True)
tk.fit_on_texts(X)
X_seq = tk.texts_to_sequences(X)
X_pad = pad_sequences(X_seq, maxlen=100, padding='post')
```

[112]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_pad, y, test_size = 0.25,␣
 ↪random_state = 1)
```

[113]:
```
batch_size = 64
X_train1 = X_train[batch_size:]
y_train1 = y_train[batch_size:]
X_valid = X_train[:batch_size]
y_valid = y_train[:batch_size]
```

[114]:
```
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
vocabulary_size = len(tk.word_counts.keys())+1
max_words = 100
embedding_size = 32
model = Sequential()
model.add(Embedding(vocabulary_size, embedding_size, input_length=max_words))
model.add(LSTM(200))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
```

[115]:
```
model.
 ↪fit(X_train1,y_train1,validation_data=(X_valid,y_valid),batch_size=batch_size,epochs=10)
```

```
C:\Users\sushb\AppData\Local\Continuum\anaconda3\lib\site-
packages\tensorflow_core\python\framework\indexed_slices.py:433: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

```
Train on 37604 samples, validate on 64 samples
Epoch 1/10
37604/37604 [==============================] - 241s 6ms/step - loss: -211.7108 -
accuracy: 0.1830 - val_loss: -331.5008 - val_accuracy: 0.2656
Epoch 2/10
37604/37604 [==============================] - 253s 7ms/step - loss: -548.4435 -
accuracy: 0.1831 - val_loss: -620.0129 - val_accuracy: 0.2656
Epoch 3/10
37604/37604 [==============================] - 253s 7ms/step - loss: -880.8150 -
accuracy: 0.1831 - val_loss: -908.4469 - val_accuracy: 0.2656
Epoch 4/10
37604/37604 [==============================] - 253s 7ms/step - loss: -1213.9509
- accuracy: 0.1831 - val_loss: -1196.3496 - val_accuracy: 0.2656
Epoch 5/10
37604/37604 [==============================] - 253s 7ms/step - loss: -1545.9127
- accuracy: 0.1831 - val_loss: -1483.9255 - val_accuracy: 0.2656
Epoch 6/10
```

```
37604/37604 [==============================] - 213s 6ms/step - loss: -1877.2001
- accuracy: 0.1831 - val_loss: -1770.8112 - val_accuracy: 0.2656
Epoch 7/10
37604/37604 [==============================] - 173s 5ms/step - loss: -2208.1054
- accuracy: 0.1831 - val_loss: -2057.5815 - val_accuracy: 0.2656
Epoch 8/10
37604/37604 [==============================] - 171s 5ms/step - loss: -2538.9139
- accuracy: 0.1831 - val_loss: -2344.1816 - val_accuracy: 0.2656
Epoch 9/10
37604/37604 [==============================] - 171s 5ms/step - loss: -2869.9704
- accuracy: 0.1831 - val_loss: -2632.5930 - val_accuracy: 0.2656
Epoch 10/10
37604/37604 [==============================] - 171s 5ms/step - loss: -3204.0844
- accuracy: 0.1831 - val_loss: -2921.8779 - val_accuracy: 0.2656
```

[115]:
```
<keras.callbacks.callbacks.History at 0x16b2363ba08>
```

The performance score of RNN Model is 26.56

22 6. Score of the above classifier models-- Results

Multinomial Naive Bayes -- 86.06, Random Forest Classifier -- 77.9, Decision Tree -- 75.62, K Nearest Neighbour Classifier -- 68.74, RNN -- 26.56.

Multinomial Naive Bayes has the best score. We will use this model to predict a random positive, negative and average review.

23 7. Validation

24 7.1 Predict positive review

[134]:
```
# POSITIVE REVIEW
pre = df['text'][20]
print(pre)
print("Actual Rating: ",df['review_stars'][20])
pre_t = vocab.transform([pre])
print("Predicted Rating:")
mnb.predict(pre_t)[0]
```

```
Unexpectedly wonderful Indian cuisine, I had the Tandoori Mixed Grill. Huge
portion with a mix of plump juicy shrimp, amazingly tender chicken and delicious
fish, tons of vegetables and great flavors! They served rice with Tikka Masala
and another garlic sauce, just tremendous. I ordered hot and it was not too
spicy, next time I might order the next level hotter, but it was excellent in
any case. Very attentive service, too!
```

```
Actual Rating:  5
Predicted Rating:
```

[134]: 5

25 7.2 Predict Average Review

[235]:
```python
# AVERAGE REVIEW
ar = df['text'][6]
print(ar)
print("Actual Rating: ",df['review_stars'][6])
ar_t = vocab.transform([ar])
print("Predicted Rating:")
mnb.predict(ar_t)[0]
```

```
Th service is really good and the owners are awesome. Very friendly and seems
like a family run establishment.

I thought the food was mediocre though not as good as federicks in Markham but
maybe I ordered the wrong thing.

Will give it another try. Not bad if you're in the area for lunch looking for
cheap eats. The $10 lunch special is well worth the money.
Actual Rating:  3
Predicted Rating:
```

[235]: 3

26 7.3 Predict Negative Review

[229]:
```python
# NEGATIVE REVIEW
nr = df['text'][58]
print(nr)
print("Actual Rating: ",df['review_stars'][10])
nr_t = vocab.transform([nr])
print("Predicted Rating:")
mnb.predict(nr_t)[0]
```

```
The more I remember this, the worse it gets. This is the second worst indian
food I've ever had in my life (first being Maezo). We had takeout: The lamb
roganjosh was very bland, and the meat did not even taste like lamb. It looked
like chicken. I think they messed up our order.

The chicken masala was terrible compared to other chicken masalas I've had at
```

```
other restaurants. It was overly oily, and the peppers in it tasted rancid.

The naan was wet and soggy. They didn't pack it properly to prevent the steam
collecting inside the aluminum. Other restaurants are able to pack it to-go
without making the naan soggy. The tandoor paneer was actually alright, it was
the only acceptable part of our meal.

I later told my Indian friend about this, and he said "oh Aroma? why on earth
would you go there?" Well, now I know…
Actual Rating:  4
Predicted Rating:
```

[229]: 1

27 8. Summary:

From the datasets, we have found that: 1 -- Mon Ami Gabi is the merchant who has got the maximum number of positive reviews. 2 -- The category of top most reviwed business is restaurants. 3 -- Casinos have got the most negative reviews.

From the machine learning models for sentiment analysis, it is clear that Multinomial Naive Bayes performs the best. The validation proves that the model works fine for positive and average reviews. But it seems to be not working for the negative reviews.

28 9. Future Scope:

I believe the reason for which the model fails to predict the negative review is that there are more positive reviews as compared to the negative ones in the dataset(the collected data,df). That means the dataset is not normally distributed. I can suggests two ways to improve it:

1 -- We can normalize the data. So that the positive and negative ratings are equally distributed over the dataset.

2 -- While collecting the data, we can also check with other business categories (Shopping, food, home services,etc). It might be possible that only the reviews for ”Indian Restaurants' 'contain mostly positive ones.

[ ]: