A simple Data Interview Assignment_ Yiting Duan

- You can find my data on Kaggle-yelp

Data Exploration && Data clearning

```
import numpy as np
import pandas as pd
data=pd.DataFrame(pd.read_excel("pandas处理json.xlsx"))
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
import warnings
import gc


warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=DeprecationWarning)
%matplotlib inline

def null_values(df):
        mis_val = df.isnull().sum()
        mis_val_percent = 100 * df.isnull().sum() / len(df)
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        print ("Dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")
        return mis_val_table_ren_columns

missing_rate = null_values(data.dropna())
```

```
   Dataframe has 16 columns.
   There are 0 columns that have missing values.
```

```
data.head()
```

1	Yzvjg0SayhoZgCljUJRF9Q	Carlos Santo, NMD	8880 E Via Linda, Ste 107	Scottsdale	AZ	85258	33.569404	-111.890264	5.0	4.0	1.0	{'GoodForKids': 'ByAppointmentOr
2	XNoUzKckATkOD1hP6vghZg	Felinus	3554 Rue Notre-Dame O	Montreal	QC	H4C 1P4	45.479984	-73.580070	5.0	5.0	1.0	
3	6OAZjbxqM5ol29BuHsil3w	Nevada House of Hose	1015 Sharp Cir	North Las Vegas	NV	89030	36.219728	-115.127725	2.5	3.0	0.0	{'BusinessAcceptsCredit('True', 'ByA
4	51M2Kk903DFYI6gnB5I6SQ	USE MY GUY SERVICES LLC	4827 E Downing Cir	Mesa	AZ	85205	33.428065	-111.726648	4.5	26.0	1.0	{'BusinessAcceptsCredit('True', 'ByA

```
data.duplicated()
```

```
0          False
1          False
2          False
3          False
4          False
            ...
1048570     True
1048571     True
1048572     True
1048573     True
1048574     True
Length: 1048575, dtype: bool
```

```
data.drop_duplicates()
data.isnull().sum()
```

```
business_id          839182
name                 839183
address              847861
city                 839184
state                839182
postal_code          839691
latitude             839182
longitude            839182
stars                839182
review_count         839182
is_open              839182
attributes           868227
categories           839706
hours                884025
number_categories    839706
number_days          884025
dtype: int64
```

```
data['address'].isnull().value_counts()
```

```
True     847861
False    200714
Name: address, dtype: int64
```

```
data['name']=data['name'].dropna()
```

```
data.isnull().sum()
```

```
business_id          839182
name                 839183
address              847861
city                 839184
state                839182
postal_code          839691
latitude             839182
longitude            839182
stars                839182
review_count         839182
is_open              839182
attributes           868227
categories           839706
hours                884025
number_categories    839706
number_days          884025
dtype: int64
```

```
data.dropna(axis=0, subset = ["stars"])
```

	business_id	name	address	city	state	postal_code	latitude	longitude	stars	review_count	is_open	
0	f9NumwFMBDn751xgFiRbNA	The Range At Lake Norman	10913 Bailey Rd	Cornelius	NC	28031	35.462724	-80.852612	3.5	36.0	1.0	{'BusinessAcce
1	Yzvjg0SayhoZgCljUJRF9Q	Carlos Santo, NMD	8880 E Via Linda, Ste 107	Scottsdale	AZ	85258	33.569404	-111.890264	5.0	4.0	1.0	{'Gooc 'ByAppo
2	XNoUzKckATkOD1hP6vghZg	Felinus	3554 Rue Notre-Dame O	Montreal	QC	H4C 1P4	45.479984	-73.580070	5.0	5.0	1.0	
3	6OAZjbxqM5ol29BuHsil3w	Nevada House of Hose	1015 Sharp Cir	North Las Vegas	NV	89030	36.219728	-115.127725	2.5	3.0	0.0	{'BusinessAcce
4	51M2Kk903DFYI6gnB5I6SQ	USE MY GUY SERVICES LLC	4827 E Downing Cir	Mesa	AZ	85205	33.428065	-111.726648	4.5	26.0	1.0	{'BusinessAcce
...	...	...	...	...	...	...	...	...	...	...	...	
209388	9Q0fPWAjUweoFDk0kafuzQ	Nishi Sushi	9750 Weston Road	Vaughan	ON	L4H 2P2	43.838555	-79.559823	4.0	5.0	0.0	{'Ambience': "{'r

```
data.isnull().sum()
```

```
business_id          839182
name                 839183
address              847861
city                 839184
state                839182
postal_code          839691
latitude             839182
longitude            839182
stars                839182
review_count         839182
is_open              839182
attributes           868227
categories           839706
hours                884025
number_categories    839706
number_days          884025
dtype: int64
```

```
data['stars'].value_counts()
```

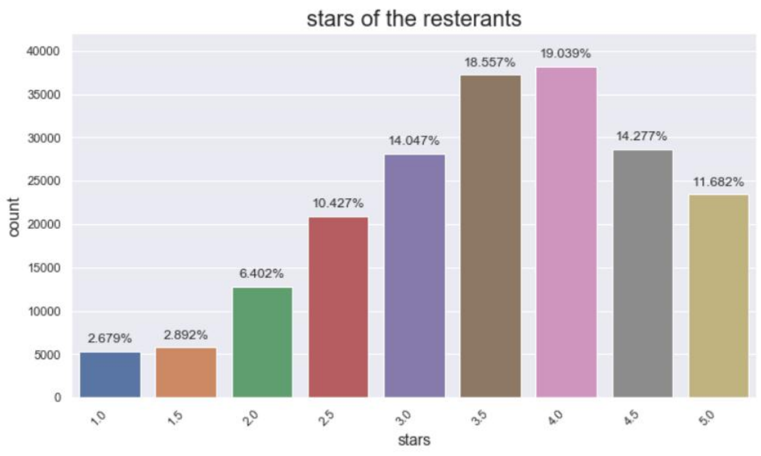
```
4.0    39199
3.5    38079
4.5    29940
3.0    28634
5.0    27080
2.5    21435
2.0    13124
1.5     6004
1.0     5898
Name: stars, dtype: int64
```

plot

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
sns.set_style('whitegrid')
import warnings
import gc
%matplotlib inline
plt.figure(figsize=(11,6))
sns.set(style='darkgrid')
g = sns.countplot(x="stars",data=data)
g.set_xticklabels(g.get_xticklabels()
                   ,fontdict={'horizontalalignment':'right'}
                   ,rotation=45)
g.set_xlabel('stars',fontsize=14)
g.set_ylabel('count',fontsize=15)
g.set_title('stars of the resterants',fontsize=20)

total_data = data.shape[0]
sizes = []
for p in g.patches:
    height = p.get_height()
    sizes.append(height)
    g.text(p.get_x()+p.get_width()/2,height+1000,
        '{:1.3f}%'.format(height/total_data*100),ha='center',fontsize=12)
g.set_ylim(0,max(sizes)*1.1)
```

```
(0, 42034.3)
```



```
data.drop(data[pd.isna(data['address'])].index, inplace=True)  ##为绘制address与stars 的crosstab，先删除address中为NaN的项
data.isnull().sum()
```

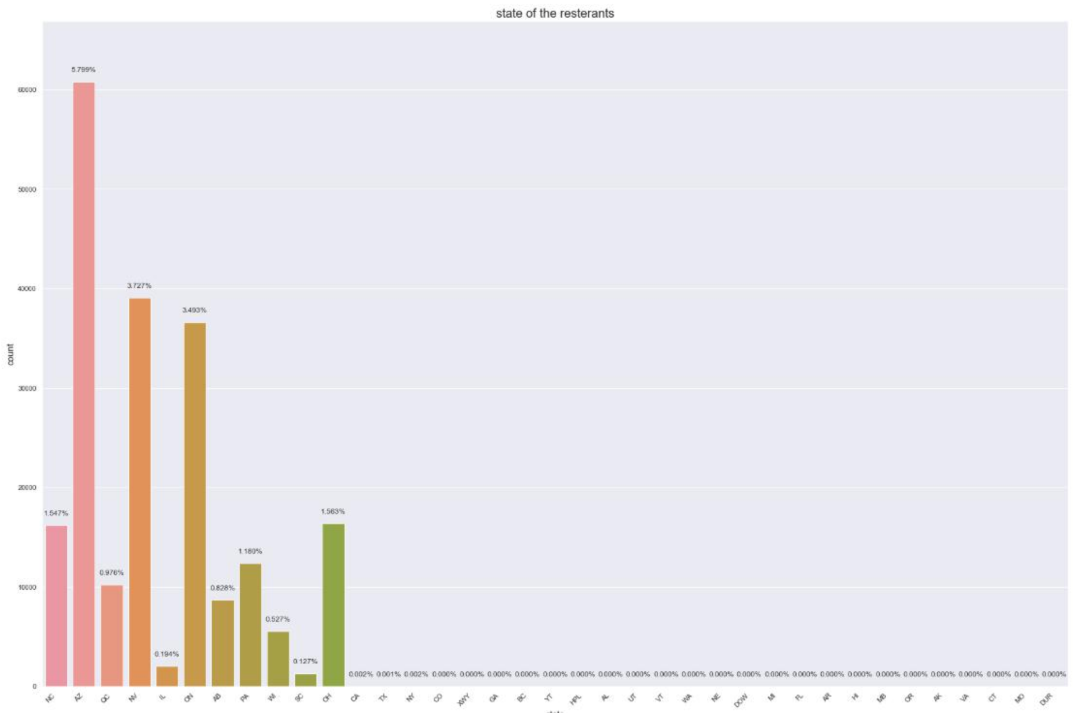
```
business_id              0
name                     1
address                  0
city                     1
state                    0
postal_code            109
latitude                 0
longitude                0
stars                    0
review_count             0
is_open                  0
attributes           27215
categories             496
hours                42719
number_categories      496
number_days          42719
dtype: int64
```

```
data['address'].isnull().value_counts()
```

```
False    200714
Name: address, dtype: int64
```

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
sns.set_style('whitegrid')
import warnings
import gc
plt.figure(figsize=(30,20))
sns.set(style='darkgrid')
g = sns.countplot(x="state",data=data)
g.set_xticklabels(g.get_xticklabels()
                  ,fontdict={'horizontalalignment':'right'}
                  ,rotation=45)
g.set_xlabel('state',fontsize=14)
g.set_ylabel('count',fontsize=15)
g.set_title('state of the resterants',fontsize=20)

total_data = data.shape[0]
sizes = []
for p in g.patches:
    height = p.get_height()
    sizes.append(height)
    g.text(p.get_x()+p.get_width()/2,height+1000,
           '{:1.3f}%'.format(height/total_data*100),ha='center',fontsize=12)
g.set_ylim(0,max(sizes)*1.1)
```

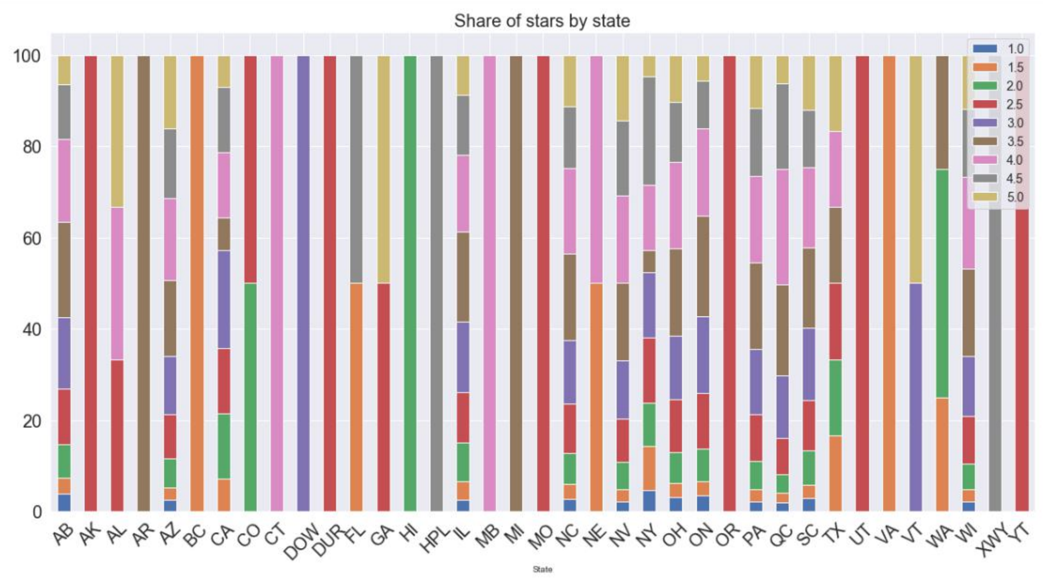


```
#crosstab
tab=pd.crosstab(data['state'],
                data['stars'],
                normalize='index') * 100
```

```
tab.head()
```

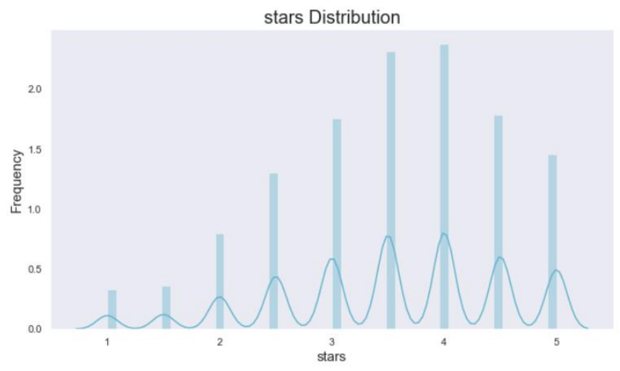
stars	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
state									
AB	3.844340	3.478774	7.334906	12.323113	15.648585	20.813679	18.054245	12.051887	6.450472
AK	0.000000	0.000000	0.000000	100.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AL	0.000000	0.000000	0.000000	33.333333	0.000000	0.000000	33.333333	0.000000	33.333333
AR	0.000000	0.000000	0.000000	0.000000	0.000000	100.000000	0.000000	0.000000	0.000000
AZ	2.466786	2.715227	6.353737	9.708567	12.829052	16.492230	18.005779	15.294076	16.134546

```
fig,ax = plt.subplots(1,1,figsize=(20, 10))
g = tab.plot(kind='bar',stacked=True,rot='45',ax=ax)
ax.set_xlabel('State',fontsize=10)
ax.tick_params(axis='both',labelsize=20)
ax.set_title('Share of stars by state',fontsize=20)
plt.legend(fontsize=14)
```



```
]: plt.figure(figsize=(11,6))
   sns.set(style='dark')
   g = sns.distplot(data["stars"], color='c')
   g.set_xlabel("stars", fontsize=15)
   g.set_ylabel("Frequency", fontsize=15)
   g.set_title("stars Distribution", fontsize=20)
```

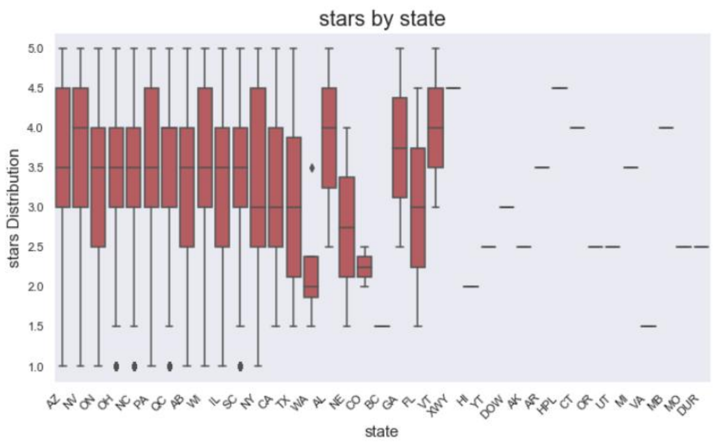
```
Text(0.5, 1.0, 'stars Distribution')
```



```
plt.figure(figsize=(11,6))

g2 = sns.boxplot(x="state", y="stars", data=data, order = data['state'].value_counts().index,color='r')
g2.set_xticklabels(g2.get_xticklabels(),fontdict={'horizontalalignment':'right'},rotation=45)
g2.set_xlabel("state", fontsize=15)
g2.set_ylabel("stars Distribution", fontsize=15)
g2.set_title("stars by state", fontsize=20)

plt.show()
```

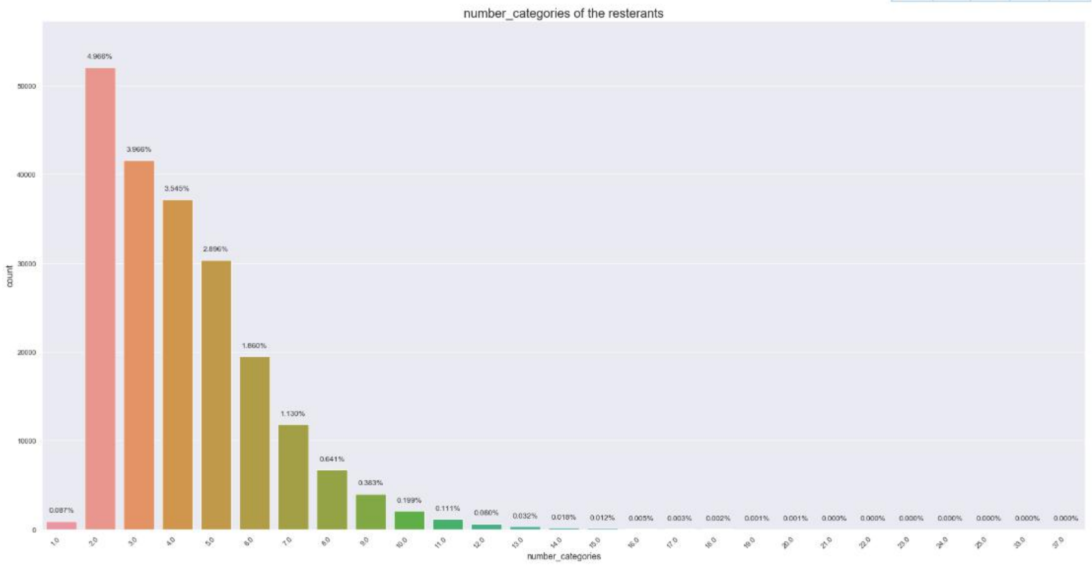


```
data.isnull().sum()
```

```
business_id              0
name                     1
address                  0
city                     1
state                    0
postal_code            109
latitude                 0
longitude                0
stars                    0
review_count             0
is_open                  0
attributes           27215
categories             496
hours                42719
number_categories      496
number_days          42719
dtype: int64
```

```
plt.figure(figsize=(30,15))
sns.set(style='darkgrid')
g = sns.countplot(x="number_categories",data=data)
g.set_xticklabels(g.get_xticklabels()
                  ,fontdict={'horizontalalignment':'right'}
                  ,rotation=45)
g.set_xlabel('number_categories',fontsize=14)
g.set_ylabel('count',fontsize=15)
g.set_title('number_categories of the resterants',fontsize=20)

total_data = data.shape[0]
sizes = []
for p in g.patches:
    height = p.get_height()
    sizes.append(height)
    g.text(p.get_x()+p.get_width()/2,height+1000,
           '{:1.3f}%'.format(height/total_data*100),ha='center',fontsize=12)
g.set_ylim(0,max(sizes)*1.1)
```



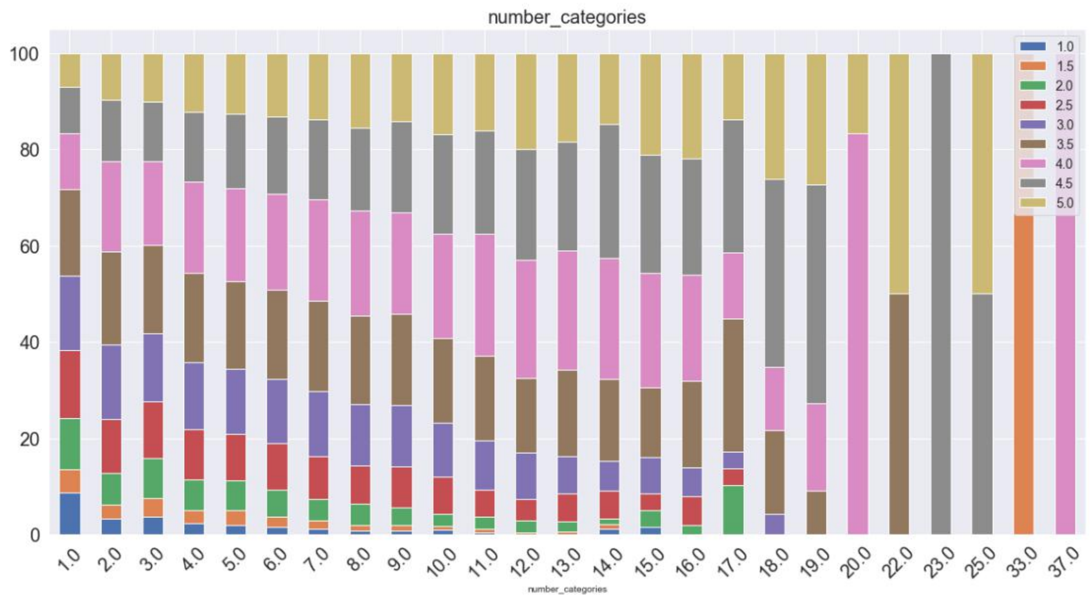
```
tab=pd.crosstab(data['number_categories'],
                data['stars'],
                normalize='index') * 100
```

```
tab.head()
```

stars	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
number_categories									
1.0	9.330406	4.939627	10.208562	13.830955	14.928650	17.672887	11.416026	10.428101	7.244786
2.0	3.612029	2.974499	6.509717	10.982026	15.001152	18.943467	18.401951	12.623857	10.951302
3.0	3.715103	3.945945	8.024142	11.681535	13.872124	17.928680	17.207300	12.268257	11.356914
4.0	2.539546	2.687507	6.284300	10.139352	13.434843	18.212633	18.618853	14.535134	13.547832
5.0	2.130883	3.016830	6.053420	9.432533	13.177222	17.728815	19.121958	15.410203	13.928136

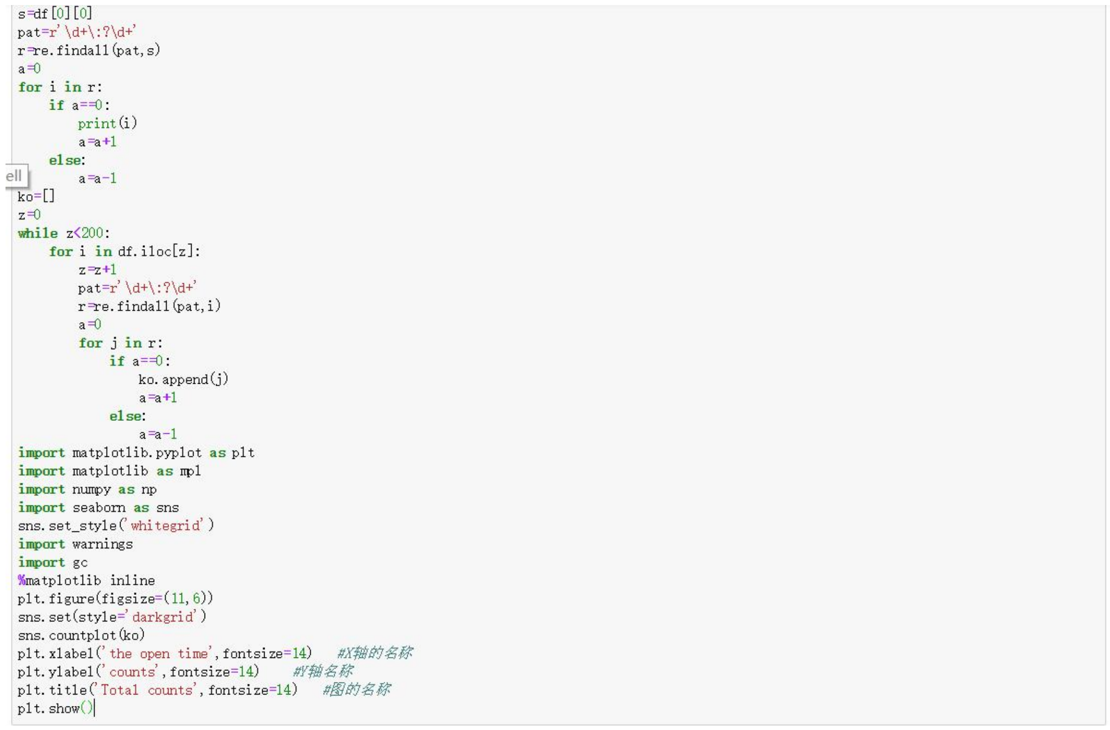
```
fig,ax = plt.subplots(1,1,figsize=(20, 10))
g = tab.plot(kind='bar',stacked=True,rot='45',ax=ax)
ax.set_xlabel('number_categories',fontsize=10)
ax.tick_params(axis='both',labelsize=20)
ax.set_title('number_categories',fontsize=20)
plt.legend(fontsize=14)
```

```
<matplotlib.legend.Legend at 0x1d02a2e6408>
```

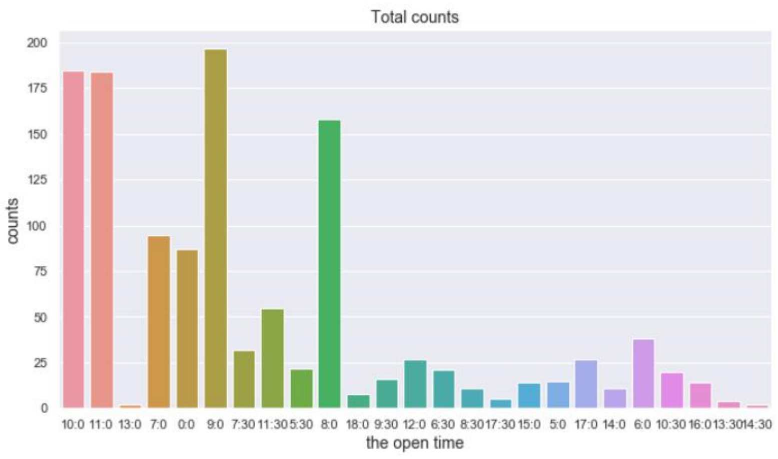


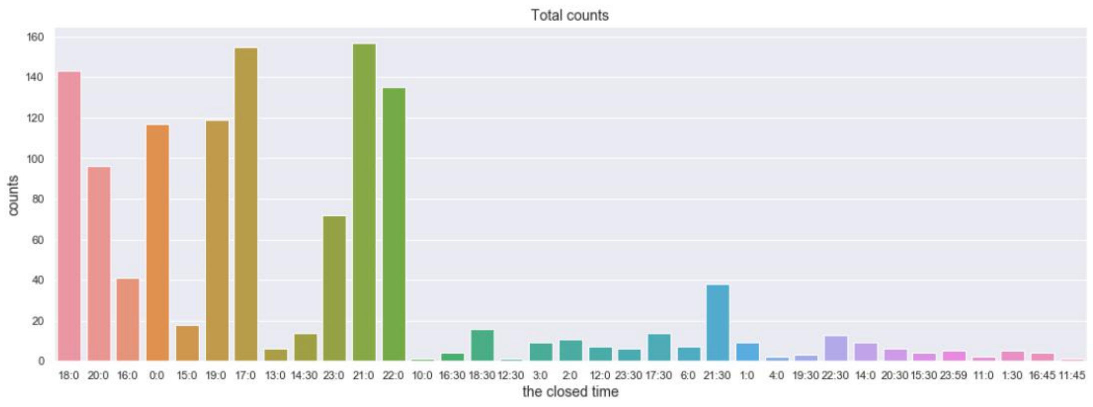
```
hours=data.loc[:,['hours']].values
df=pd.DataFrame(hours)
df
```

	0
0	{'Monday': '10:0-18:0', 'Tuesday': '11:0-20:0'...
1	{'Monday': '7:0-16:0', 'Tuesday': '7:0-16:0', ...
2	{'Monday': '0:0-0:0', 'Tuesday': '9:0-16:0', '...
3	{'Monday': '7:0-18:0', 'Tuesday': '7:0-18:0', ...
4	{'Monday': '7:0-19:0', 'Tuesday': '7:0-19:0', ...
...	...
141132	{'Monday': '5:30-15:0', 'Tuesday': '5:30-15:0'...
141133	{'Monday': '7:0-17:0', 'Tuesday': '7:0-17:0', ...
141134	{'Monday': '10:30-0:0', 'Tuesday': '10:30-0:0'...
141135	{'Monday': '11:0-22:0', 'Tuesday': '11:0-22:0'...
141136	{'Monday': '11:0-22:0', 'Tuesday': '11:0-22:0'...

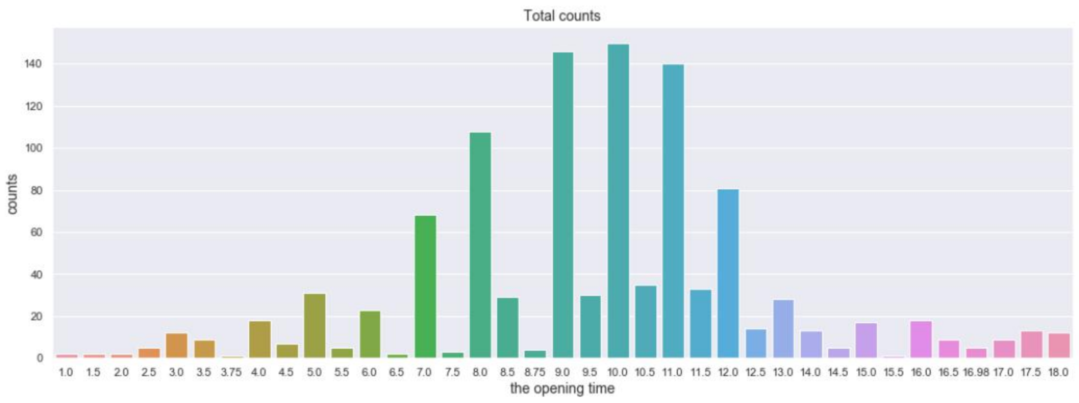


```
s=df[0][0]
pat=r'\d+\:?\d+'
r=re.findall(pat,s)
a=0
for i in r:
    if a==0:
        print(i)
        a=a+1
    else:
        a=a-1
ko=[]
z=0
while z<200:
    for i in df.iloc[z]:
        z=z+1
        pat=r'\d+\:?\d+'
        r=re.findall(pat,i)
        a=0
        for j in r:
            if a==0:
                ko.append(j)
                a=a+1
            else:
                a=a-1
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
import seaborn as sns
sns.set_style('whitegrid')
import warnings
import gc
%matplotlib inline
plt.figure(figsize=(11,6))
sns.set(style='darkgrid')
sns.countplot(ko)
plt.xlabel('the open time',fontsize=14)    #X轴的名称
plt.ylabel('counts',fontsize=14)    #Y轴名称
plt.title('Total counts',fontsize=14)    #图的名称
plt.show()
```





```python
ko['a']=ko[0]
ko=pd.DataFrame(ko)
kc=pd.DataFrame(kc)
sum=pd.concat([ko['a'],kc['b']],axis=1)
sum['a']=sum['a'].astype(str)
sum['b']=sum['b'].astype(str)
sum['ahour']=sum['a'].str.split(':')
sum['bhour']=sum['b'].str.split(':')
sum['amins']=sum['ahour'].apply(lambda x: int(x[0])*60+int(x[1]))
sum['bmins']=sum['bhour'].apply(lambda y: int(y[0])*60+int(y[1]))
sum['mins']=sum['bmins']-sum['amins']
sum['hour']=round(sum['mins']/60,2)
new=[]
for i in sum['hour']:
    if i>0:
        new.append(i)
new=pd.DataFrame(new)
%matplotlib inline
plt.figure(figsize=(18,6))
sns.set(style='darkgrid')
sns.countplot(new[0])
plt.xlabel('the opening time',fontsize=14)   #X轴的名称
plt.ylabel('counts',fontsize=14)    #Y轴名称
plt.title('Total counts',fontsize=14)   #图的名称
plt.show()
```



To analysis the text

```python
#find the most popluar Bussiness_id and analysis it's comments
temp_a=pd.DataFrame(data['business_id'].value_counts())
temp_b=data['user_id'].value_counts()
first=data.loc[data['business_id']=='FaHADZARwnY4yv1vpnsfGA']
text=first.iloc[:,2]
#Use NLTK
```

```python
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

```python
r=[]
for i,v in text.items():
    r.append(v)
g=[]
for i in range(3679):
    g.append(r[i])
type(g)
```

```
list
```

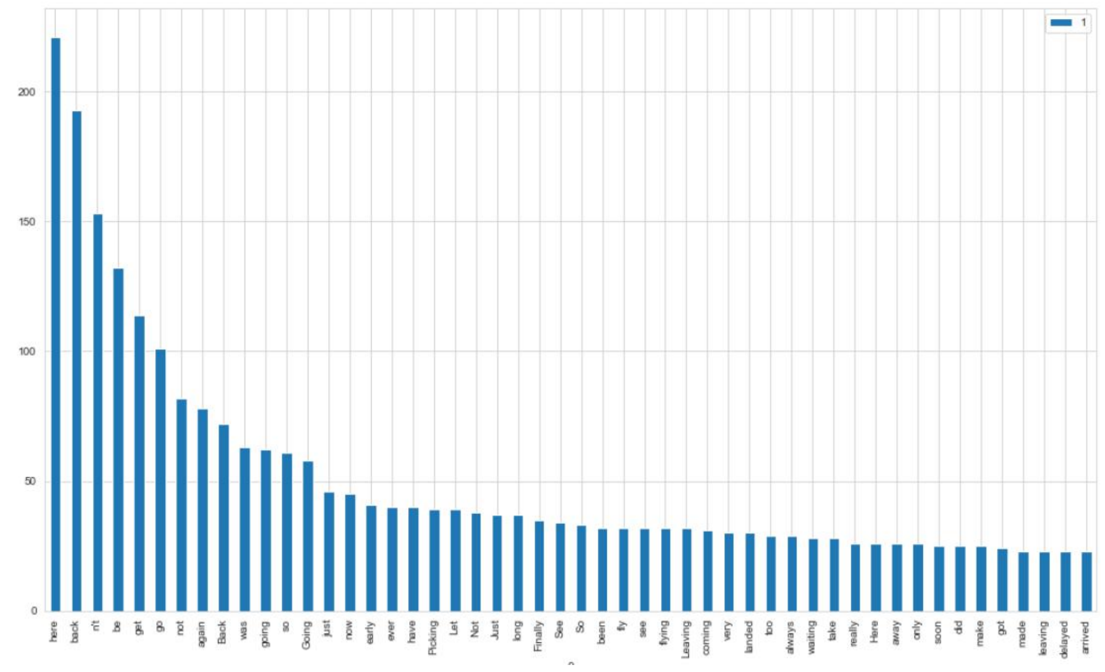
```python
first_list=[]
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
for i in range(3679):
    allnoun=nltk.pos_tag(word_tokenize(g[i]))
    word=[word for word,pos in allnoun if pos in ['RB','RBR','UH','VED','VB','VBD','VBG','VBN']]
    for j in word:
        first_list.append(j)
    i=i+1
```

```python
from collections import Counter
# 统计列表中元素出现的频率
counter = Counter(first_list)

# 将列表中的元素按照频率大小排序
x=pd.DataFrame(counter.most_common())
x
```

```
new=x.iloc[:50]
new.plot(0,kind='bar',figsize=(17,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c58e6d5d08>
```



Use IntensityAnalyzer & calculate the score

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
sid=SentimentIntensityAnalyzer()
```

```
score=[]
for i in range(3679):
    sentence=r[i]
    ss=sid.polarity_scores(sentence)
    score.append(ss)
score=pd.DataFrame(score)
```

```
score
```

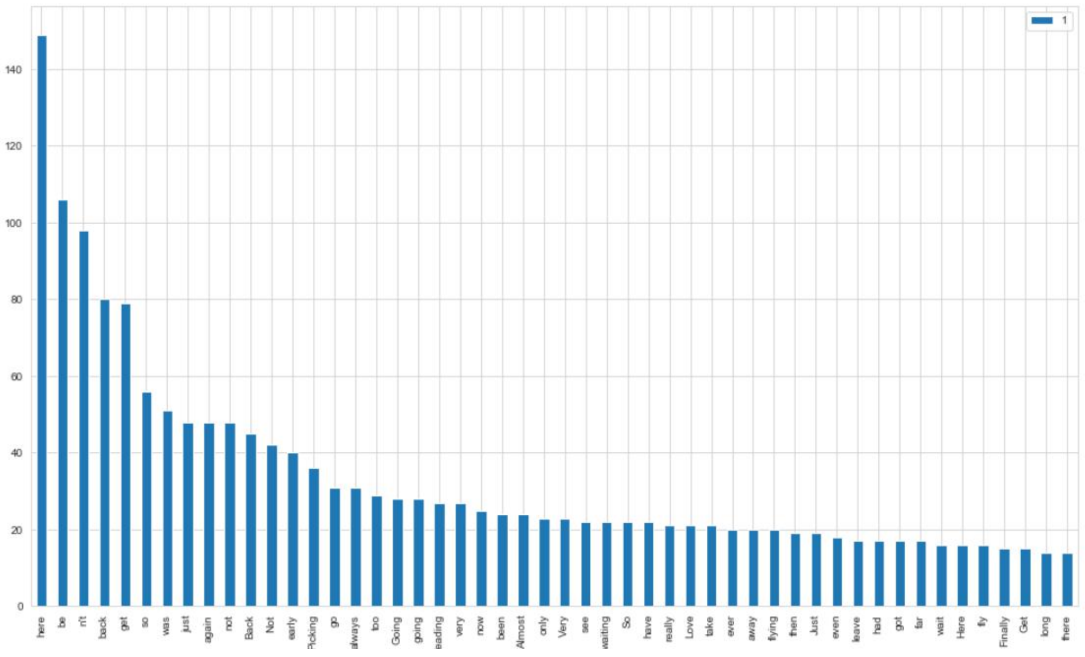
	neg	neu	pos	compound
0	0.000	1.000	0.000	0.0000
1	0.000	1.000	0.000	0.0000
2	0.000	0.882	0.118	0.3680
3	0.000	1.000	0.000	0.0000
4	0.000	0.590	0.410	0.7067
...	...	...	...	...
3674	0.000	1.000	0.000	0.0000
3675	0.534	0.466	0.000	-0.7830
3676	0.000	0.626	0.374	0.5550
3677	0.000	1.000	0.000	0.0000

```
import numpy
from collections import Counter
def count_sentence_emotion(score):
    list1 = score['neg'].values.tolist()
    list2=score['neu'].values.tolist()
    list3=score['pos'].values.tolist()
    list4=score['compound'].values.tolist()
    collection_words1 = Counter(list1)
    most_counterNum1 = collection_words1.most_common(1)
    print(most_counterNum1)
    collection_words2 = Counter(list2)
    most_counterNum2 = collection_words2.most_common(1)
    print(most_counterNum2)
    collection_words3 = Counter(list3)
    most_counterNum3= collection_words3.most_common(1)
    print(most_counterNum3)
    collection_words4 = Counter(list4)
    most_counterNum4= collection_words4.most_common(1)
    print(most_counterNum4)
count_sentence_emotion(score)
```

```
[(0.0, 2906)]
[(1.0, 1811)]
[(0.0, 2293)]
[(0.0, 1816)]
```

```
    # analyse 'business_id'=='JmI9nslLD7KZqRr__Bg6NQ'
    second=data.loc[data['business_id']=='JmI9nslLD7KZqRr__Bg6NQ']
    text2=second.iloc[:,2]
    r=[]
    for i,v in text2.items():
        r.append(v)
    lenr=len(r)
    g=[]
    for i in range(lenr):
        g.append(r[i])
    second_list=[]
    import nltk
    from nltk.tokenize import sent_tokenize, word_tokenize
    for i in range(lenr):
        allnoun=nltk.pos_tag(word_tokenize(g[i]))
        word=[word for word,pos in allnoun if pos in ['RB','RBR','UH','VED','VB','VBD','VBG','VBN']]
        for j in word:
            second_list.append(j)
        i=i+1
    counter = Counter(second_list)
    x=pd.DataFrame(counter.most_common())
    new=x.iloc[:50]
    %matplotlib inline
    new.plot(0,kind='bar',figsize=(17,10))
    sid=SentimentIntensityAnalyzer()
    score=[]
    for i in range(lenr):
        sentence=r[i]
        ss=sid.polarity_scores(sentence)
        score.append(ss)
    score=pd.DataFrame(score)
count_sentence_emotion(score)
```

```
[(0.0, 2048)]
[(1.0, 1200)]
[(0.0, 1467)]
[(0.0, 1204)]
```



use statistical method to analyse

```
text_new=''
for i in range(20000):     #取前20000条进行分析
    text_new=text_new+' '+text.iloc[i]
```

```
words=nltk.word_tokenize(text_new)
```

```
from nltk.stem import SnowballStemmer    #进行词干提取
snowball_stemmer=SnowballStemmer("english")
words_new=[]
for i in words:
    words_new.append(snowball_stemmer.stem(i))
```

```
from nltk.corpus import stopwords    #去停用词
words_new=[word for word in words_new if word not in stopwords.words('english')]
```

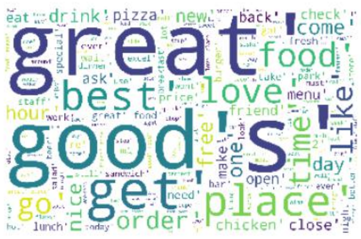
```
words_refresh=[]                    #去掉一些不能提供情感分析的符号
import re
for list in words_new:
    string = re.sub("[\s+\.\!\/_,$%^*(+\"\']+|[+——！，。？：）``；？、~@#￥%……&*（）]+", " ", list)
    if string!=' ':
        words_refresh.append(string)
```

```
import enchant
def is_english_word(word):
    d_en = enchant.Dict("en_US")
    return d_en.check(word)
```

```
words_fin=[]                #对拼写进行检查
for i in words_refresh:
    if is_english_word(i) == True:
        words_fin.append(i)
```

```
freq_dist = nltk.FreqDist(words_fin)
for k,v in freq_dist.items():
    print(str(k)+':'+str(v))
print(freq_dist.most_common(50))  #最常见的50个词
```

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
wordcloud = WordCloud(
        background_color="white",
        width=1500,
        height=960,
        margin=10
        ).generate(str(words_fin))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
wordcloud.to_file('my_test2.png')
```



```
<wordcloud.wordcloud.WordCloud at 0x2821da5dcc8>
```

Use same method to analyze reviews

```
import pandas as pd
```

```
## Bulid up a chunsize to load the data
df=pd.read_json('yelp_academic_dataset_review.json',lines=True,chunksize=20000,nrows=20000)
```

```
for chunksize in df:
    print(chunksize)
```

```
                   review_id                 user_id             business_id  \
0     xQY8N_XvtGbearJ5X4QryQ  OwjRMXRC0KyPrIlcjaXeFQ  -MhfebM0QIsKt87iDN-FNw
1     UmFMZ8PyXZTY2QcwzsfQYA  nIJD_7ZXHq-FX8byPMOkMQ  1brU8StCq3yDfr-QMnGrmQ
2     LG2ZaYiOgpr2DK_90pYjNw  V34qejxNsCbcgD8C0HVk-Q  HQ128KMwrEKHqhFrrDqVNQ
3     i6g_oA9Yf9Y31qt0wibXpw  ofKDkJKXSKZXu5xJNGiiBQ  5JxlZaqCnk1MnbgRirs40Q
4     6TdNDKywdbjoTkizeMce8A  UgMW8bLE0QMJDCkQ1Ax5Mg  IS4cv902ykd8wj1TR0N3-A
...                      ...                     ...                     ...
19995 uFBmIMdpnJC4Tq7n3gWOJg  yBOAoIDR7b6r3v3pAqNnLA  sAEFh7e7fY-W2gMWSbIAWQ
19996 G8xO5iym3dzYAqtANdWTJw  Hprxt1x2u83bKJPYCrGsbQ  SFcmvdfsyjsgmu-b9L91Vw
19997 OsmdU2ukP81eozJpHuIXcw  HvQeFuvIzDfKUZV17rZzRg  1jfwf_DkzQgTZz1cwhL4dg
19998 8jqrnirKWC7--ew7dJ4Oig  TBjAK4U81VytvQvDE0AtAQ  _MNNGWXqkXCPiKQxCHsiuQ
19999 cTzPiVuhrgsFPMHZXk_CFQ  XuXdIy1Z6Or1dbeBQ4HpWQ  uoZwJJap1LsktVrRw0JiQw

       stars  useful  funny  cool  \
0          2       5      0     0
1          1       1      1     0
2          5       1      0     0
3          1       0      0     0
4          4       0      0     0
...      ...     ...    ...   ...
19995      5       0      0     1
19996      3       0      0     0
19997      5       3      0     1
19998      4       0      0     0
19999      5       0      0     0
```

Data Exploration && Data Cleaning

```
data=chunksize
```

```
data.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
19995    False
19996    False
19997    False
19998    False
19999    False
Length: 20000, dtype: bool
```

```
data.isnull().sum()
```

```
review_id      0
user_id        0
business_id    0
stars          0
useful         0
funny          0
cool           0
text           0
date           0
dtype: int64
```

```
data['stars'].value_counts()
```

```
data['useful'].value_counts()
data['funny'].value_counts()
data['cool'].value_counts()
```

```
#通过分析后发现stars数据最为正常，决定首先分析stars评论星级为5的文本
stars=data[data['stars'].isin([5])]
```

```
text=stars.text
```

```
##use NLTK
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
```

```
word=''
for i in range(2000):
    word=word+' '+text.iloc[i]
```

```
word=nltk.word_tokenize(word)
```

```
from nltk.corpus import stopwords
word=[word for word in word if word not in stopwords.words('english')]
```

```
import re
from nltk.corpus import stopwords
from nltk import word_tokenize,pos_tag
from nltk.stem import WordNetLemmatizer
```

```
#对文本进行清洗
import re
from nltk.corpus import stopwords
from nltk import word_tokenize,pos_tag
from nltk.stem import WordNetLemmatizer

def tokenize(sentence):
    '''
        去除多余空白、分词、词性标注
    '''
    sentence = re.sub(r'\s+', ' ', sentence)
    token_words = word_tokenize(sentence)
    token_words = pos_tag(token_words)
    return token_words

wordnet_lematizer = WordNetLemmatizer()

sr = stopwords.words('english')
def delete_stopwords(token_words):
    '''
        去停用词
    '''
    cleaned_words = [word for word in token_words if word not in sr]
    return cleaned_words

def is_number(s):
    '''
        判断字符串是否为数字
    '''
    try:
        float(s)
        return True
    except ValueError:
        pass
    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False

characters = [' ','i','the','we',',', '.','DBSCAN', ':', ';', '?', '(', ')', '[', ']', '&', '!', '*', '@', '#', '$', '%','-','...','^','{','}']
def delete_characters(token_words):
    '''
        去除特殊字符、数字
    '''
    words_list = [word for word in token_words if word not in characters and not is_number(word)]
    return words_list

def to_lower(token_words):
    '''
        统一为小写
    '''
    words_lists = [x.lower() for x in token_words]
    return words_lists

def pre_process(text):
    '''
        文本预处理
    '''
    token_words = tokenize(text)
    token_words = stem(token_words)
    token_words = delete_stopwords(token_words)
    token_words = delete_characters(token_words)
    token_words = to_lower(token_words)
    return token_words
```

```
word = delete_stopwords(word)
word = delete_characters(word)
word = to_lower(word)
```

Analyze

```
from nltk.probability import FreqDist
fdist = FreqDist(word)
tops=fdist.most_common(50)
print(tops)
```

```
[('great', 1057), ("'s", 929), ('place', 906), ("n't", 860), ('food', 848), ('good', 735), ('service', 656), ('time', 583), ('best', 491),
('back', 462), ('one', 430), ('get', 429), ('love', 423), ('like', 420), ('amazing', 409), ('really', 407), ('go', 392), ('would', 382),
("'ve", 378), ('friendly', 375), ('also', 373), ('always', 355), ('nice', 342), ('delicious', 338), ('staff', 326), ('definitely', 317), ('w
ell', 310), ('recommend', 290), ('us', 289), ('got', 285), ('made', 273), ('restaurant', 271), ('try', 266), ('first', 256), ('awesome', 24
2), ("'m", 232), ('even', 232), ('experience', 232), ('new', 229), ('could', 229), ('everything', 224), ('much', 223), ('come', 222), ('eve
r', 221), ('ordered', 220), ('came', 217), ('make', 212), ('went', 212), ('fresh', 209), ('little', 204)]
```

```
#计算高频词出现的频率
def ret (x):
    return x
k=0
for a in map(ret,tops):
    j=0
    for x in a:
        print("tops[%d][%d]="%(k,j),end="")
        if j!=0:
            print(int(x)/87021," ")
        else:
            print(x," ")
        j=j+1
    print
    k+=1
```

```
tops[0][0]=great
tops[0][1]=0.012146493375162317
tops[1][0]='s
tops[1][1]=0.010675584054423645
tops[2][0]=place
tops[2][1]=0.010411280035853414
tops[3][0]=n't
tops[3][1]=0.009882671998712955
tops[4][0]=food
tops[4][1]=0.009744774249893704
tops[5][0]=good
tops[5][1]=0.008446237115179094
tops[6][0]=service
tops[6][1]=0.007538410268785696
tops[7][0]=time
tops[7][1]=0.006699532296801921
tops[8][0]=best
tops[8][1]=0.005642316222521
tops[9][0]=back
tops[9][1]=0.005309063329541145
```

Use Citysearch Corpus to begin the Supervised Learning

```
try_text=pd.read_table('test.txt')     #导入Citysearch Corpus语料库进行监督学习
## the corpus Citysearch Corpus is a restearant's comment set from Citysearch New-York Net
```

```
try_text
```

```
                                                  text
   0          Always a fun place ... the food is deeelish !
   1          The staff is n't the friendliest or most comp...
   2          Great for groups , great for a date , great f...
   3          Another great place to take out-of-towners !
   4                                                    : )
 ...                                                    ...
3323                               Was there Friday night .
3324      Best Pastrami I ever had and great portion wi...
3325                     And I 've been to many NYC delis .
3326     My wife had the fried shrimp which are huge a...
3327      Price no more than a Jersey deli but way bett...

3328 rows × 1 columns
```

```
word=''
for i in range(3327):
    word=word+try_text.iloc[i].values
```

```
str = ','.join(str(i) for i in word)
```

```
word=nltk.word_tokenize(str)
```

```
tops_word=[]
def ret (x):
    return x
k=0
for a in map(ret,tops):
    j=0
    for x in a:
        tops_word.append(x)
        j=j+1
    print
    k+=1
```

```
words=[]     #为stars文本中的50个高频词汇
i=0
while i<100:
    words.append(tops_word[i])
    i=i+2;
```

```
tops_fre=[]      #频率
i=1
while i<100:
    tops_fre.append(tops_word[i]/87021)
    i=i+2;
```

```
len(word)
```

```
54528
```

```
fre= nltk.FreqDist(word)
```

```
fre
```

```
FreqDist({'.': 2993, 'the': 2259, ',': 1943, 'and': 1629, 'a': 1151, 'to': 995, 'I': 959, 'is': 923, 'was': 731, 'of': 695, ...})
```

Calculate IDF

```
#计算IDF
import math
IDF=[]
for i in range(50):
    IDF.append(math.log(1/fre.freq(words[i])))
```

```
IDF
```

```
[5.472747606634748,
 5.1475678363117074,
 5.144418227408811,
 5.271680007019738,
 4.833425076088583,
 5.30066754389299,
 5.6386114511256595,
 6.342121418721152,
 6.135784985723323,
 6.262078711047615,
 6.046657205827316,
 6.102448565455731,
 7.077828213699893,
 5.882589089342711,
 7.217590156075051,
 6.602404516984818,
 5.806603182364789,
 6.054439346269371,
 6.152879419082623,
 6.8121250479668864,
 6.828932166283268,
 6.512020455516549,
 6.616010169040597,
 6.549760783499396,
```

```python
#计算文本的feature= IDF*Tops_fre 最后降序排列
feature=[]
for i in range(50):
    feature.append(IDF[i]*tops_fre[i]*100)
```

```python
#单词与频率组成字典
nvs = zip(words, feature)
FinDict = dict( (name, value) for name, value in nvs)
```

```python
#将字典根据value进行排序
FinDict=sorted(FinDict.items(), key=lambda item:item[1],reverse=True)
```

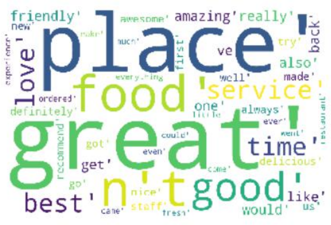
```python
FinDict
```

```
[('great', 6.647469254792439),
("'s", 5.495329311239328),
('place', 5.355997878710176),
("n't", 5.2098284391548875),
('food', 4.710063622025854),
('good', 4.477069494445418),
('service', 4.250616646485828),
('time', 4.248924727496157),
('best', 3.462003916284749),
('love', 3.4404584346250386),
('amazing', 3.392278155657481),
('back', 3.324577245152318),
('really', 3.0879657076025566),
('get', 3.008412262075256),
('one', 2.9878564926922766),
('friendly', 2.9355522149683204),
('also', 2.9271000080712226),
('like', 2.839185274271657),
("'ve", 2.6726748950405437),
('would', 2.6577444872788174),
('definitely', 2.6576227440752116),
('always', 2.656562509863567),
('go', 2.615677189973681),
('nice', 2.600148789156507),
('delicious', 2.5440056363668493),
('awesome', 2.5347443193166543),
('staff', 2.4785044013597117),
('well', 2.337856262722727),
('got', 2.2778862333323615),
('made', 2.256536961619974),
('recommend', 2.2434859156813545),
('us', 2.1505885504405833),
('first', 2.130732166717449),
```

```python
Fin_word=[]    #根据value排序后的字典的特征单词的list
def ret (x):
    return x
k=0
for a in map(ret,FinDict):
    j=0
    for x in a:
        Fin_word.append(x)
        j=j+1
    print
    k+=1
Fin_words=[]
i=0
while i<100:
    Fin_words.append(Fin_word[i])
    i=i+2;
```

```python
del str
```

```python
#将feature word绘制成词云
from wordcloud import WordCloud
import matplotlib.pyplot as plt
wordcloud = WordCloud(
        background_color="white",
        width=1500,
        height=960,
        margin=10
        ).generate(str(Fin_words))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
wordcloud.to_file('Fin_words.png')
```


friendly'
amazing'
really'
place'
back
love'
also'
food
service'
great'
time'
first
ve
well'
made'
one'
always'
definitely'
recommend
got'
get'
delicious'
best'
n't
good
like'
nice'
staff'
would'
us'
go'
new