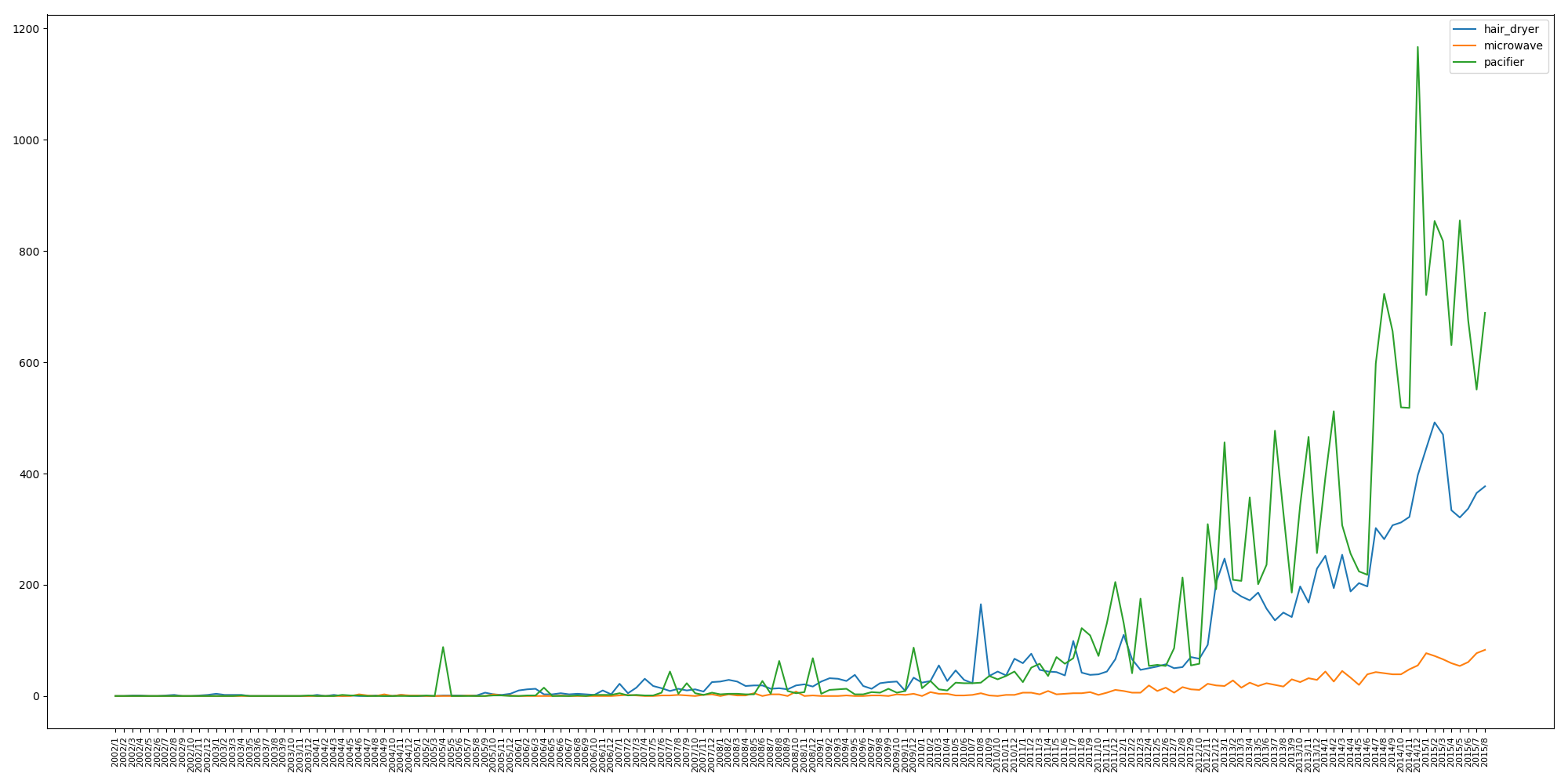
appendix 1:



|  |  |  |  |
| --- | --- | --- | --- |
| code | anylisis | remark | the code of data anylisis |
| import pandas as pd  import matplotlib.pyplot as plt  from tqdm import tqdm  from my\_util import pre\_process  def null\_process():  hair\_dryer=pd.read\_csv('../Data/hair\_dryer.tsv',sep='\t',encoding='utf-8')  microwave=pd.read\_csv('../Data/microwave.tsv',sep='\t',encoding='utf-8')  pacifier=pd.read\_csv('../Data/pacifier.tsv',sep='\t',encoding='utf-8')  # print(hair\_dryer.head())  for column in hair\_dryer.columns:  if len(set(hair\_dryer[column]))==1:  print('hair\_dryer:',column)  if len(set(microwave[column]))==1:  print('microwave:',column)  if len(set(pacifier[column]))==1:  print('pacifier:',column)  print(set(pacifier['product\_category']))  print(set(pacifier['marketplace']))  print(set(microwave['product\_category']))  print(set(microwave['marketplace']))  #'product\_category', 'marketplace' 每一列的值都相同, 直接删掉  hair\_dryer['review\_date'] = pd.to\_datetime (hair\_dryer['review\_date'], format='%m/%d/%Y')  hair\_dryer['year'] = hair\_dryer['review\_date'].dt.year  hair\_dryer['month'] = hair\_dryer['review\_date'].dt.month  pacifier['review\_date'] = pd.to\_datetime (pacifier['review\_date'], format='%m/%d/%Y')  pacifier['year'] = pacifier['review\_date'].dt.year  pacifier['month'] = pacifier['review\_date'].dt.month  microwave['review\_date'] = pd.to\_datetime (microwave['review\_date'], format='%m/%d/%Y')  microwave['year'] = microwave['review\_date'].dt.year  microwave['month'] = microwave['review\_date'].dt.month  del hair\_dryer['product\_category']  del hair\_dryer['marketplace']  del pacifier['product\_category']  del pacifier['marketplace']  del microwave['product\_category']  del microwave['marketplace']  del hair\_dryer['product\_title']  del pacifier['product\_title']  del microwave['product\_title']  tmp1 = pacifier[pacifier['product\_id'] == 'b0042i2bwg']  print (tmp1)  tmp2 = pacifier[pacifier['product\_id'] == 'b00db5f114']  print (tmp2)  dic1 = {}  for idx in hair\_dryer.index:  i = hair\_dryer.loc[idx, 'product\_id']  j = hair\_dryer.loc[idx, 'product\_parent']  if i not in dic1:  dic1[i] = [j]  else:  dic1[i].append (j)  for i in dic1:  if len (set (dic1[i])) != 1:  print ('hair\_dryer')  dic2 = {}  for idx in microwave.index:  i = microwave.loc[idx, 'product\_id']  j = microwave.loc[idx, 'product\_parent']  if i not in dic2:  dic2[i] = [j]  else:  dic2[i].append (j)  for i in dic2:  if len (set (dic2[i])) != 1:  print ('microwave')  dic3 = {}  for idx in pacifier.index:  i = pacifier.loc[idx, 'product\_id']  j = pacifier.loc[idx, 'product\_parent']  if i not in dic3:  dic3[i] = [j]  else:  dic3[i].append (j)  for i in dic3:  if len (set (dic3[i])) != 1:  print (i, dic3[i])  print ('pacifier')  # 以上发现除了pacifier里边的两个异常值例外, 其他所有数据, 只要product\_id一样, product\_parent也一样故只保留一个字段  del hair\_dryer['product\_parent']  del microwave['product\_parent']  del pacifier['product\_parent']  print(hair\_dryer['product\_id'].count()) #11470  print(microwave['product\_id'].count())#1615  print(pacifier['product\_id'].count())#18939  hair\_dryer=hair\_dryer.dropna()  microwave=microwave.dropna()  pacifier=pacifier.dropna()  print(hair\_dryer['product\_id'].count())#11468  print(microwave['product\_id'].count())#1615  print(pacifier['product\_id'].count())#18937  # 只有四行数据有缺失值, 不影响整体, 故直接删除之后重新保存  reviewer\_body = []  for i in tqdm (hair\_dryer['review\_body'].values):  sent = ''  for j in pre\_process (i):  sent = sent + ' ' + j  reviewer\_body.append (sent)  hair\_dryer['review\_body'] = reviewer\_body  reviewer\_body = []  for i in tqdm (microwave['review\_body'].values):  sent = ''  for j in pre\_process (i):  sent = sent + ' ' + j  reviewer\_body.append (sent)  microwave['review\_body'] = reviewer\_body  reviewer\_body = []  for i in tqdm (pacifier['review\_body'].values):  sent = ''  try:  for j in pre\_process (i):  sent = sent + ' ' + j  except:  print (i)  sent = i  reviewer\_body.append (sent)  pacifier['review\_body'] = reviewer\_body  hair\_dryer = hair\_dryer.dropna ()  microwave = microwave.dropna ()  pacifier = pacifier.dropna ()  print (hair\_dryer['product\_id'].count ()) # 11468  print (microwave['product\_id'].count ()) # 1615  print (pacifier['product\_id'].count ()) # 18937  hair\_dryer.to\_csv('../Data/hair\_dryer.csv',encoding='utf-8',index=None)  microwave.to\_csv('../Data/microwave.csv',encoding='utf-8',index=None)  pacifier.to\_csv('../Data/pacifier.csv',encoding='utf-8',index=None)  # 打印有缺失值的行  # print(hair\_dryer[hair\_dryer.isnull().values==True])  # print(microwave[microwave.isnull().values==True])  # print(pacifier[pacifier.isnull().values==True])  # null\_process()  hair\_dryer=pd.read\_csv('../Data/hair\_dryer.csv',encoding='utf-8')  microwave=pd.read\_csv('../Data/microwave.csv',encoding='utf-8')  pacifier=pd.read\_csv('../Data/pacifier.csv',encoding='utf-8')  print(hair\_dryer.columns)  def fig\_star\_rating\_count():  tmp1=hair\_dryer.groupby(by='star\_rating').count()['customer\_id']  plt.subplot(221)  plt.bar(tmp1.index.values,tmp1.values)  plt.ylim(0,8000)  for a, b in zip(tmp1.index.values, tmp1.values):  plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)  plt.title('hair\_dryer')  tmp2=microwave.groupby(by='star\_rating').count()['customer\_id']  plt.subplot(222)  plt.ylim(0,800)  plt.bar(tmp2.index.values,tmp2.values)  for a, b in zip(tmp2.index.values, tmp2.values):  plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)  plt.title('microwave')  tmp3=pacifier.groupby(by='star\_rating').count()['customer\_id']  plt.subplot(212)  plt.ylim(0,14000)  plt.bar(tmp3.index.values,tmp3.values)  for a, b in zip(tmp3.index.values, tmp3.values):  plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)  plt.title('pacifier')  plt.show()  # fig\_star\_rating\_count()  def fig\_time():  # print(hair\_dryer.groupby('product\_id').count()['customer\_id'].describe())  # hair\_dryer1=hair\_dryer[hair\_dryer['review\_date']>pd.to\_datetime('1/1/2013',format='%m/%d/%Y')]  y1=hair\_dryer.groupby(['year','month']).count()['customer\_id']  y2 = microwave.groupby (['year','month']).count ()['customer\_id']  y3 = pacifier.groupby (['year','month']).count ()['customer\_id']  x=[]  for i in range(2002,2016):  for j in range(1,13):  x.append((i,j))  x.pop(-1)  x.pop(-1)  x.pop(-1)  x.pop(-1)  tmp=[]  for i in x:  if i in list(y1.index.values):  tmp.append(y1.loc[i])  else:  tmp.append(0)  y1=tmp  tmp = []  for i in x:  if i in list(y2.index.values):  tmp.append (y2[i])  else:  tmp.append (0)  y2=tmp  tmp = []  for i in x:  if i in list(y3.index.values):  tmp.append (y3[i])  else:  tmp.append (0)  y3=tmp  x=[str(item[0])+'/'+str(item[1]) for item in x]  plt.figure(figsize=(20,10))  plt.plot(x,y1)  plt.plot(x,y2)  plt.plot(x,y3)  plt.xticks (size='small', rotation=90, fontsize=8)  plt.legend(['hair\_dryer','microwave','pacifier'],loc = 'best')  plt.show ()  print (tmp)  fig\_time()  # test1=hair\_dryer[hair\_dryer['product\_id']=='B003V264WW']  # print()  # print(hair\_dryer.info())  # print(microwave.info())  # print(pacifier.info())  #  # print(hair\_dryer.describe())  # print(microwave.describe())  # print(pacifier.describe())  # # 以上结论: 我们需要对数据取样, 因为要想准确显示评价真实性, 要结合helpful\_votes/verified\_purchase两个指标来看, 需要排除一些没有意义的评价  # print(hair\_dryer[hair\_dryer['verified\_purchase']=='Y'].count())  print() | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| code | T1 | remark | | the code of question 1 |
| from textblob import TextBlob  #  # text = "five stars".replace('.','')  # blob = TextBlob (text)  # # 分句  # print ("blob对象")  # print (blob)  # print (blob.sentiment)  from sklearn.model\_selection import train\_test\_split  import pandas as pd  import lightgbm as lgb  from tqdm import tqdm  import numpy as np  cat\_cols = ['customer\_id', 'review\_id', 'product\_id', 'vine', 'verified\_purchase']  def pre\_process(prod,data):  del data['review\_headline']  dtime = pd.to\_datetime (data['review\_date'])  v = (dtime.values - np.datetime64 ('2000-01-01T08:00:00Z')) / np.timedelta64 (1, 'ms')  data['review\_date'] = v  data[cat\_cols].astype('category')  def map\_value(x):  x\_set=set(x.values)  dic={}  n=0  for i in x\_set:  if i not in dic:  dic[i]=n  n+=1  new\_x=[]  for i in x.values:  new\_x.append(dic[i])  return new\_x  for col in cat\_cols:  data[col] = map\_value(data[col])  def get\_sentment(col):  new\_col=[]  # for i in tqdm(col):  # out\_put = emotion\_eng.getMoodValue(i)  # new\_col.append(out\_put['all\_value'])  for i in tqdm(col):  out\_put = TextBlob (i)  new\_col.append(out\_put.sentiment.polarity)  return new\_col  data['review\_body']=get\_sentment(data['review\_body'])  def anylisis(data):  not\_pair = data[((data['star\_rating'] == 1) & (data['review\_body'] > 0.6)) | ((data['star\_rating'] == 5) & (data['review\_body'] < -0.6))]  # print(not\_pair)  return not\_pair.index.values  abnormal\_product = {}  abnormal\_product[prod] = (list (anylisis (data))) # 8  print(abnormal\_product)  # abnormal\_product['microwave'] = (list (anylisis (microwave))) # 3  # abnormal\_product['pacifier'] = (list (anylisis (pacifier))) # 18  data = data[~data.index.isin (abnormal\_product[prod])]  return data  hair\_dryer=pd.read\_csv('../Data/hair\_dryer.csv',encoding='utf-8')  hair\_dryer=hair\_dryer.dropna()  hair\_dryer=pre\_process('hair\_dryer',hair\_dryer)  hair\_dryer.to\_csv('../Data/new\_hair\_dryer.csv')  microwave=pd.read\_csv('../Data/microwave.csv',encoding='utf-8')  microwave=microwave.dropna()  microwave=pre\_process('microwave',microwave)  microwave.to\_csv('../Data/new\_microwave.csv')  pacifier=pd.read\_csv('../Data/pacifier.csv',encoding='utf-8')  pacifier=pacifier.dropna()  pacifier=pre\_process('pacifier',pacifier)  pacifier.to\_csv('../Data/new\_pacifier.csv')  def get\_X\_y(prod,data):  cols\_x=['customer\_id', 'review\_id', 'product\_id', 'helpful\_votes', 'total\_votes', 'vine', 'verified\_purchase','review\_body', 'review\_date', 'year', 'month','rate']  star=[]  for i in data['star\_rating']:  if i<4:  star.append(0)  else:  star.append(1)  data['star\_rating']=star  X=data[cols\_x]  X[cat\_cols]=X[cat\_cols].astype('category')  # X['customer\_id']=X['customer\_id'].astype('category')  y=data['star\_rating']  # min\_max\_scaler = MinMaxScaler ()  # X= min\_max\_scaler.fit\_transform (X)  # da=pd.DataFrame(X,columns=cols\_x)  return X,y  def gen\_rate(data):  tmp = data.groupby ('product\_id').count ()['customer\_id']  sums = {}  for i in tmp.index.values:  sums[i] = tmp[i]  rate = {}  for i in sums:  cnt = data[(data['product\_id'] == i) & (data['star\_rating'] < 4)].count ()[0]  rate[i] = cnt / sums[i]  rates = []  for i in data['product\_id'].values:  rates.append (rate[i])  data['rate'] = rates  return data  hair\_dryer=pd.read\_csv('../Data/new\_hair\_dryer.csv',encoding='utf-8',index\_col=0)  microwave=pd.read\_csv('../Data/new\_microwave.csv',encoding='utf-8',index\_col=0)  pacifier=pd.read\_csv('../Data/new\_pacifier.csv',encoding='utf-8',index\_col=0)  hair\_dryer=gen\_rate(hair\_dryer)  microwave=gen\_rate(microwave)  pacifier=gen\_rate(pacifier)  def model1():  X,y=get\_X\_y('hair\_dryer',hair\_dryer)  print(len(y))  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2, random\_state=0,shuffle=True)  print("Train data length:", len(X\_train))  print("Test data length:", len(X\_test))  print('开始训练!')  # 训练 cv and train  gbm = lgb.sklearn.LGBMClassifier(boosting\_type='gbdt', num\_leaves=64, max\_depth=-1, learning\_rate=0.09, n\_estimators=10, max\_bin=255, subsample\_for\_bin=200000, objective=None, min\_split\_gain=0.0, min\_child\_weight=0.001, min\_child\_samples=20, subsample=1.0, subsample\_freq=1, colsample\_bytree=1.0, reg\_alpha=0.0, reg\_lambda=0.0, random\_state=None, n\_jobs=-1, silent=True)  gbm.fit(X\_train,y\_train,sample\_weight=None, init\_score=None,  eval\_set=None, eval\_names=None, eval\_sample\_weight=None,  eval\_class\_weight=None, eval\_init\_score=None, eval\_metric=None,  early\_stopping\_rounds=None, verbose=True,  feature\_name='auto', categorical\_feature='auto', callbacks=None)  print ('Start predicting...')  # 预测数据集  y\_pred = gbm.predict (X\_test)  from sklearn.metrics import precision\_score, recall\_score, roc\_auc\_score  print('importance:',list(zip(X\_train.columns.values,gbm.feature\_importances\_)))  precision=precision\_score(y\_test, y\_pred)  recall=recall\_score(y\_test, y\_pred)  print(list(zip(y\_test.values,y\_pred)))  print ('正确率：', precision)  print ('召回率：', recall)  print ('auc值：', roc\_auc\_score (y\_test, y\_pred))  print ('F1值：', 2 \* (precision \* recall) / (precision + recall))  def model2():  X, y = get\_X\_y ('microwave',microwave)  print (len (y))  X\_train, X\_test, y\_train, y\_test = train\_test\_split (X, y, test\_size=0.2, random\_state=0, shuffle=True)  print ("Train data length:", len (X\_train))  print ("Test data length:", len (X\_test))  print ('开始训练!')  # 训练 cv and train  gbm = lgb.sklearn.LGBMClassifier (boosting\_type='gbdt', num\_leaves=64, max\_depth=-1, learning\_rate=0.1,  n\_estimators=10, max\_bin=255, subsample\_for\_bin=200000, objective=None,  min\_split\_gain=0.0, min\_child\_weight=0.001, min\_child\_samples=20, subsample=1.0,  subsample\_freq=1, colsample\_bytree=1.0, reg\_alpha=0.0, reg\_lambda=0.0,  random\_state=None, n\_jobs=-1, silent=True)  gbm.fit (X\_train, y\_train, sample\_weight=None, init\_score=None,  eval\_set=None, eval\_names=None, eval\_sample\_weight=None,  eval\_class\_weight=None, eval\_init\_score=None, eval\_metric=None,  early\_stopping\_rounds=None, verbose=True,  feature\_name='auto', categorical\_feature='auto', callbacks=None)  print ('Start predicting...')  # 预测数据集  y\_pred = gbm.predict (X\_test)  from sklearn.metrics import precision\_score, recall\_score, roc\_auc\_score  print ('importance:', list (zip (X\_train.columns.values, gbm.feature\_importances\_)))  precision = precision\_score (y\_test, y\_pred)  recall = recall\_score (y\_test, y\_pred)  print (list (zip (y\_test.values, y\_pred)))  print ('正确率：', precision)  print ('召回率：', recall)  print ('auc值：', roc\_auc\_score (y\_test, y\_pred))  print ('F1值：', 2 \* (precision \* recall) / (precision + recall))  def model3():  X, y = get\_X\_y ('pacifier',pacifier)  print (len (y))  X\_train, X\_test, y\_train, y\_test = train\_test\_split (X, y, test\_size=0.2, random\_state=0, shuffle=True)  print ("Train data length:", len (X\_train))  print ("Test data length:", len (X\_test))  print ('开始训练!')    gbm = lgb.sklearn.LGBMClassifier (boosting\_type='gbdt', num\_leaves=64, max\_depth=-1, learning\_rate=0.09,  n\_estimators=10, max\_bin=255, subsample\_for\_bin=200000, objective=None,  min\_split\_gain=0.0, min\_child\_weight=0.001, min\_child\_samples=20, subsample=1.0,  subsample\_freq=1, colsample\_bytree=1.0, reg\_alpha=0.0, reg\_lambda=0.0,  random\_state=None, n\_jobs=-1, silent=True)  gbm.fit (X\_train, y\_train, sample\_weight=None, init\_score=None,  eval\_set=None, eval\_names=None, eval\_sample\_weight=None,  eval\_class\_weight=None, eval\_init\_score=None, eval\_metric=None,  early\_stopping\_rounds=None, verbose=True,  feature\_name='auto', categorical\_feature='auto', callbacks=None)  print ('Start predicting...')  # 预测数据集  y\_pred = gbm.predict (X\_test)  from sklearn.metrics import precision\_score, recall\_score, roc\_auc\_score  print ('importance:', list (zip (X\_train.columns.values, gbm.feature\_importances\_)))  precision = precision\_score (y\_test, y\_pred)  recall = recall\_score (y\_test, y\_pred)  print (list (zip (y\_test.values, y\_pred)))  print ('正确率：', precision)  print ('召回率：', recall)  print ('auc值：', roc\_auc\_score (y\_test, y\_pred))  print ('F1值：', 2 \* (precision \* recall) / (precision + recall))  # hair\_dryer  model1()  #microwave  model2()  #pacifier  model3() | | | | |
| code | T2\_a | | remark | the code of question 2-a |
| from sklearn.decomposition import pca  from sklearn.preprocessing import StandardScaler, MinMaxScaler  import numpy as np  import pandas as pd  import math  from textblob import TextBlob  # blob = TextBlob ("text")  # print(blob.sentiment.polarity)  # out\_put = emotion\_eng.getMoodValue("great")#out\_put['all\_value']  # all\_low=hair\_dryer[(hair\_dryer['star\_rating']<2) & (hair\_dryer['review\_body']<-0.6)]  # all\_high=hair\_dryer[(hair\_dryer['star\_rating']>3) & (hair\_dryer['review\_body']>0.2)]  # all\_mid=hair\_dryer[(hair\_dryer['star\_rating']>=2) & (hair\_dryer['star\_rating']<=3) & (0.2>=hair\_dryer['review\_body']) & (hair\_dryer['review\_body']>=-0.6)]  # not\_pair=hair\_dryer[((hair\_dryer['star\_rating']<2) & (hair\_dryer['review\_body']>0.8)) | ((microwave['star\_rating']==5) & (hair\_dryer['review\_body']<-0.6))]  #  # a=all\_low.count()['star\_rating']  # b=all\_high.count()['star\_rating']  # c=all\_mid.count()['star\_rating']  # d=hair\_dryer.count()['star\_rating']  # e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  # print()  # all\_low=microwave[(microwave['star\_rating']<2) & (microwave['review\_body']<-0.6)]  # all\_high=microwave[(microwave['star\_rating']>3) & (microwave['review\_body']>0.2)]  # all\_mid=microwave[(microwave['star\_rating']>=2) & (microwave['star\_rating']<=3) & (0.2>=microwave['review\_body']) & (microwave['review\_body']>=-0.6)]  # not\_pair=microwave[((microwave['star\_rating']<2) & (microwave['review\_body']>0.8)) | ((microwave['star\_rating']==5) & (microwave['review\_body']<-0.6))]  #  # a=all\_low.count()['star\_rating']  # b=all\_high.count()['star\_rating']  # c=all\_mid.count()['star\_rating']  # d=microwave.count()['star\_rating']  # e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  # print()  # all\_low=pacifier[(pacifier['star\_rating']<2) & (pacifier['review\_body']<-0.6)]  # all\_high=pacifier[(pacifier['star\_rating']>3) & (pacifier['review\_body']>0.2)]  # all\_mid=pacifier[(pacifier['star\_rating']>=2) & (pacifier['star\_rating']<=3) & (0.2>=pacifier['review\_body']) & (pacifier['review\_body']>=-0.6)]  #  # not\_pair=pacifier[((pacifier['star\_rating']<2) & (pacifier['review\_body']>0.8)) | ((pacifier['star\_rating']==5) & (pacifier['review\_body']<-0.6))]  # a=all\_low.count()['star\_rating']  # b=all\_high.count()['star\_rating']  # c=all\_mid.count()['star\_rating']  # d=pacifier.count()['star\_rating']  # e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  hair\_dryer=pd.read\_csv('../Data/new\_hair\_dryer.csv',encoding='utf-8',index\_col=0)  microwave=pd.read\_csv('../Data/new\_microwave.csv',encoding='utf-8',index\_col=0)  pacifier=pd.read\_csv('../Data/new\_pacifier.csv',encoding='utf-8',index\_col=0)  def anylisis(data):  all\_low=data[(data['star\_rating']<2) & (data['review\_body']<-0.6)]  all\_high=data[(data['star\_rating']>3) & (data['review\_body']>0.2)]  all\_mid=data[(data['star\_rating']>=2) & (data['star\_rating']<=3) & (0.2>=data['review\_body']) & (data['review\_body']>=-0.6)]  # not\_pair 表示好评却1星或5星却差评  not\_pair=data[((data['star\_rating']==1) & (data['review\_body']>0.6)) | ((data['star\_rating']==5) & (data['review\_body']<-0.6))]  a=all\_low.count()['star\_rating']  b=all\_high.count()['star\_rating']  c=all\_mid.count()['star\_rating']  d=data.count()['star\_rating']  e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  # print('好评却1星或5星却差评数量:',e)  return not\_pair.index.values  abnormal\_product={}  abnormal\_product['hair\_dryer']=(list(anylisis(hair\_dryer)))#8  abnormal\_product['microwave']=(list(anylisis(microwave)))#3  abnormal\_product['pacifier']=(list(anylisis(pacifier)))#18  print(abnormal\_product)  def scaler(X):  """  注：这里的归一化是按照列进行的。也就是把每个特征都标准化，就是去除了单位的影响。  """  min\_max\_scaler = MinMaxScaler ()  x\_train= min\_max\_scaler.fit\_transform (X)  x=pd.DataFrame(x\_train,columns=X.columns.values)  return x  def cal\_weight(x):  '''熵值法计算变量的权重'''  # 标准化  x = x.apply (lambda x: ((x - np.min (x)) / (np.max (x) - np.min (x))))  # 求k  rows = x.index.size # 行  cols = x.columns.size # 列  k = 1.0 / math.log (rows)  lnf = [[None] \* cols for i in range (rows)]  # 矩阵计算--  # 信息熵  # p=array(p)  x = np.array (x)  lnf = [[None] \* cols for i in range (rows)]  lnf = np.array (lnf)  for i in range (0, rows):  for j in range (0, cols):  if x[i][j] == 0:  lnfij = 0.0  else:  p = x[i][j] / x.sum (axis=0)[j]  lnfij = math.log (p) \* p \* (-k)  lnf[i][j] = lnfij  lnf = pd.DataFrame (lnf)  E = lnf  # 计算冗余度  d = 1 - E.sum (axis=0)  # 计算各指标的权重  w = [[None] \* 1 for i in range (cols)]  for j in range (0, cols):  wj = d[j] / sum (d)  w[j] = wj  # 计算各样本的综合得分,用最原始的数据  w = pd.DataFrame (w)  return w  def get\_eval(prod,data):  data=data[~data['product\_id'].isin(abnormal\_product[prod])]  x=data[['star\_rating','review\_body']]  # x=scaler(x)  w = cal\_weight (x) # 调用cal\_weight  w.index = x.columns  w.columns = ['weight']  wei={'star\_rating':w.loc['star\_rating','weight'],'review\_body':w.loc['review\_body','weight']}  return wei  # wei=get\_eval('hair\_dryer',hair\_dryer) #{'star\_rating': 0.8529774897515476, 'review\_body': 0.1470225102484523}  # print(wei)  def gen\_score(prod,data):  x=data['star\_rating'].values  y=data['review\_body'].values  wei = get\_eval (prod, data)  print(prod,wei)  score=np.array(x)\*wei['star\_rating']+np.array(y)\*wei['review\_body']  data['score']=score  return data  data1=gen\_score('hair\_dryer',hair\_dryer)  data2=gen\_score('microwave',microwave)  data3=gen\_score('pacifier',pacifier)  print(data1.head())  print(data2.head())  print(data3.head()) | | | | |
| code | T2\_b | | remark | the code of question 2-b |
| from sklearn.decomposition import pca  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler, MinMaxScaler  import numpy as np  import pandas as pd  import math  import matplotlib.pyplot as plt  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import LogisticRegression  from sklearn.linear\_model import SGDClassifier  from sklearn.metrics import classification\_report  from textblob import TextBlob  # blob = TextBlob ("text")  # print(blob.sentiment.polarity)  # out\_put = emotion\_eng.getMoodValue("great")#out\_put['all\_value']  # all\_low=hair\_dryer[(hair\_dryer['star\_rating']<2) & (hair\_dryer['review\_body']<-0.6)]  # all\_high=hair\_dryer[(hair\_dryer['star\_rating']>3) & (hair\_dryer['review\_body']>0.2)]  # all\_mid=hair\_dryer[(hair\_dryer['star\_rating']>=2) & (hair\_dryer['star\_rating']<=3) & (0.2>=hair\_dryer['review\_body']) & (hair\_dryer['review\_body']>=-0.6)]  # not\_pair=hair\_dryer[((hair\_dryer['star\_rating']<2) & (hair\_dryer['review\_body']>0.8)) | ((microwave['star\_rating']==5) & (hair\_dryer['review\_body']<-0.6))]  #  # a=all\_low.count()['star\_rating']  # b=all\_high.count()['star\_rating']  # c=all\_mid.count()['star\_rating']  # d=hair\_dryer.count()['star\_rating']  # e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  # print()  # all\_low=microwave[(microwave['star\_rating']<2) & (microwave['review\_body']<-0.6)]  # all\_high=microwave[(microwave['star\_rating']>3) & (microwave['review\_body']>0.2)]  # all\_mid=microwave[(microwave['star\_rating']>=2) & (microwave['star\_rating']<=3) & (0.2>=microwave['review\_body']) & (microwave['review\_body']>=-0.6)]  # not\_pair=microwave[((microwave['star\_rating']<2) & (microwave['review\_body']>0.8)) | ((microwave['star\_rating']==5) & (microwave['review\_body']<-0.6))]  #  # a=all\_low.count()['star\_rating']  # b=all\_high.count()['star\_rating']  # c=all\_mid.count()['star\_rating']  # d=microwave.count()['star\_rating']  # e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  # print()  # all\_low=pacifier[(pacifier['star\_rating']<2) & (pacifier['review\_body']<-0.6)]  # all\_high=pacifier[(pacifier['star\_rating']>3) & (pacifier['review\_body']>0.2)]  # all\_mid=pacifier[(pacifier['star\_rating']>=2) & (pacifier['star\_rating']<=3) & (0.2>=pacifier['review\_body']) & (pacifier['review\_body']>=-0.6)]  #  # not\_pair=pacifier[((pacifier['star\_rating']<2) & (pacifier['review\_body']>0.8)) | ((pacifier['star\_rating']==5) & (pacifier['review\_body']<-0.6))]  # a=all\_low.count()['star\_rating']  # b=all\_high.count()['star\_rating']  # c=all\_mid.count()['star\_rating']  # d=pacifier.count()['star\_rating']  # e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  hair\_dryer=pd.read\_csv('../Data/new\_hair\_dryer.csv',encoding='utf-8',index\_col=0)  microwave=pd.read\_csv('../Data/new\_microwave.csv',encoding='utf-8',index\_col=0)  pacifier=pd.read\_csv('../Data/new\_pacifier.csv',encoding='utf-8',index\_col=0)  def gen\_rate(data):  tmp = data.groupby ('product\_id').count ()['customer\_id']  sums = {}  for i in tmp.index.values:  sums[i] = tmp[i]  rate = {}  for i in sums:  cnt = data[(data['product\_id'] == i) & (data['star\_rating'] < 4)].count ()[0]  rate[i] = cnt / sums[i]  rates = []  for i in data['product\_id'].values:  rates.append (rate[i])  data['rate'] = rates  return data  hair\_dryer=gen\_rate(hair\_dryer)  microwave=gen\_rate(microwave)  pacifier=gen\_rate(pacifier)  def anylisis(data):  all\_low=data[(data['star\_rating']<2) & (data['review\_body']<-0.6)]  all\_high=data[(data['star\_rating']>3) & (data['review\_body']>0.2)]  all\_mid=data[(data['star\_rating']>=2) & (data['star\_rating']<=3) & (0.2>=data['review\_body']) & (data['review\_body']>=-0.6)]  # not\_pair 表示好评却1星或5星却差评  not\_pair=data[((data['star\_rating']==1) & (data['review\_body']>0.6)) | ((data['star\_rating']==5) & (data['review\_body']<-0.6))]  a=all\_low.count()['star\_rating']  b=all\_high.count()['star\_rating']  c=all\_mid.count()['star\_rating']  d=data.count()['star\_rating']  e=not\_pair.count()['star\_rating']  # print(a,b,c,e,d-a-b-c)  # print('好评却1星或5星却差评数量:',e)  return not\_pair.index.values  abnormal\_product={}  abnormal\_product['hair\_dryer']=(list(anylisis(hair\_dryer)))#8  abnormal\_product['microwave']=(list(anylisis(microwave)))#3  abnormal\_product['pacifier']=(list(anylisis(pacifier)))#18  print(abnormal\_product)  def scaler(X):  """  注：这里的归一化是按照列进行的。也就是把每个特征都标准化，就是去除了单位的影响。  """  min\_max\_scaler = MinMaxScaler ()  x\_train= min\_max\_scaler.fit\_transform (X)  x=pd.DataFrame(x\_train,columns=X.columns.values)  return x  def cal\_weight(x):  '''熵值法计算变量的权重'''  # 标准化  x = x.apply (lambda x: ((x - np.min (x)) / (np.max (x) - np.min (x))))  # 求k  rows = x.index.size # 行  cols = x.columns.size # 列  k = 1.0 / math.log (rows)  lnf = [[None] \* cols for i in range (rows)]  # 矩阵计算--  # 信息熵  # p=array(p)  x = np.array (x)  lnf = [[None] \* cols for i in range (rows)]  lnf = np.array (lnf)  for i in range (0, rows):  for j in range (0, cols):  if x[i][j] == 0:  lnfij = 0.0  else:  p = x[i][j] / x.sum (axis=0)[j]  lnfij = math.log (p) \* p \* (-k)  lnf[i][j] = lnfij  lnf = pd.DataFrame (lnf)  E = lnf  # 计算冗余度  d = 1 - E.sum (axis=0)  # 计算各指标的权重  w = [[None] \* 1 for i in range (cols)]  for j in range (0, cols):  wj = d[j] / sum (d)  w[j] = wj  # 计算各样本的综合得分,用最原始的数据  w = pd.DataFrame (w)  return w  def get\_eval(prod,data):  data=data[~data['product\_id'].isin(abnormal\_product[prod])]  x=data[['star\_rating','review\_body']]  # x=scaler(x)  w = cal\_weight (x) # 调用cal\_weight  w.index = x.columns  w.columns = ['weight']  wei={'star\_rating':w.loc['star\_rating','weight'],'review\_body':w.loc['review\_body','weight']}  return wei  # wei=get\_eval('hair\_dryer',hair\_dryer) #{'star\_rating': 0.8529774897515476, 'review\_body': 0.1470225102484523}  # print(wei)  def gen\_score(prod,data):  x=data['star\_rating'].values  y=data['review\_body'].values  wei = get\_eval (prod, data)  score=np.array(x)\*wei['star\_rating']+np.array(y)\*wei['review\_body']  data['score']=score  return data  def fig(prod, D):  data=gen\_score(prod,D)[['review\_date','year','month','score']]  # print(data.describe())  good=data[(data['score']>4) & (data['year']>2009)].groupby(['year','month']).count()['score']  bad=data[(data['score']<1) & (data['year']>2009)].groupby(['year','month']).count()['score']  all\_of=data[(data['year']>2009)].groupby(['year','month']).count()['review\_date']  good=pd.DataFrame(good,index=good.index.values)  bad=pd.DataFrame(bad,index=bad.index.values)  all\_of=pd.DataFrame(all\_of,index=all\_of.index.values)  bad.rename(columns={'score':'score\_bad'},inplace=True)  x=[str(i[0])+'/'+str(i[1]) for i in good.index.values]  good['time']=x  x=[str(i[0])+'/'+str(i[1]) for i in bad.index.values]  bad['time']=x  x=[str(i[0])+'/'+str(i[1]) for i in all\_of.index.values]  all\_of['time']=x  all=pd.merge(good,bad,how='left')  all=pd.merge(all,all\_of,how='left')  all.fillna(0)  fig = plt.figure(num=1, figsize=(15, 8),dpi=80)  plt.plot(all['time'].values,all['score'].values/all['review\_date'].values)  # plt.show()  plt.plot(all['time'].values,all['score\_bad'].values/all['review\_date'].values)  # plt.plot(all['time'].values,all['review\_date'].values)  plt.legend(['good','bad'],loc = 'best')  plt.xticks (size='small', rotation=90, fontsize=13)  plt.show()  # fig('hair\_dryer',hair\_dryer)  # fig('microwave',microwave)  # fig('pacifier',pacifier)  def classify(prod,data):  data=gen\_score(prod, data)  cols\_x = ['helpful\_votes', 'total\_votes', 'verified\_purchase', 'review\_body', 'review\_date', 'month', 'rate','score']  x=data[cols\_x]  scores=data['star\_rating'].values  y=[]  for score in scores:  if score>=4:  y.append(1)  else:  y.append(0)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=0)  ss = StandardScaler ()  X\_train = ss.fit\_transform (X\_train)  X\_test = ss.fit\_transform (X\_test)  lr = LogisticRegression()  lr.fit (X\_train, y\_train)  lr\_y\_predict = lr.predict (X\_test)  print(lr\_y\_predict)  print ('Accuracy of LR Classifier:', lr.score (X\_test, y\_test))  print()  classify('hair\_dryer',hair\_dryer)  print() | | | | |
| code | T2\_c | | remark | the code of question 2-c |
|  | | | | |
| code | T2\_d | | remark | the code of question 2-d |
| import pandas as pd  del\_cols = [&apos;customer\_id&apos;,&apos;review\_id&apos;,&apos;product\_id&apos;,&apos;helpful\_votes&apos;,&apos;total\_votes&apos;,&apos;vine&apos;,&apos;verified\_purchase&apos;,&apos;review\_date&apos;,&apos;year&apos;,&apos;month&apos;]  hair\_dryer=pd.read\_csv(&apos;../Data/new\_hair\_dryer.csv&apos;,encoding=&apos;utf-8&apos;,index\_col=0) hair\_dryer=hair\_dryer.drop(del\_cols,axis=1) microwave=pd.read\_csv(&apos;../Data/new\_microwave.csv&apos;,encoding=&apos;utf-8&apos;,index\_col=0) microwave=microwave.drop(del\_cols,axis=1) pacifier=pd.read\_csv(&apos;../Data/new\_pacifier.csv&apos;,encoding=&apos;utf-8&apos;,index\_col=0) pacifier=pacifier.drop(del\_cols,axis=1) print(&apos;hair\_dryer\n&apos;,hair\_dryer.corr(&apos;spearman&apos;)) print() print(&apos;microwave\n&apos;,microwave.corr(&apos;spearman&apos;)) print() print(&apos;pacifier\n&apos;,pacifier.corr(&apos;spearman&apos;)) print() | | | | |
| code | T2\_e | | remark | the code of question 2-e |
| import collections  import pickle  import numpy as np  import matplotlib.pyplot as plt  import jieba.analyse  import seaborn as sns  from tqdm import tqdm  from my\_util import pre\_process  import pandas as pd  import wordcloud  # 基于TF - IDF：jieba.analyse.extract\_tags (sentence, topK=20, withWeight=False, allowPOS=())  # 基于TextRank：jieba.analyse.textrank (sentence, topK=20, withWeight=False, allowPOS=('ns', 'n', 'vn', 'v'))  hair\_dryer=pd.read\_csv('../Data/hair\_dryer.csv',encoding='utf-8')  microwave=pd.read\_csv('../Data/microwave.csv',encoding='utf-8')  pacifier=pd.read\_csv('../Data/pacifier.csv',encoding='utf-8')  hair\_dryer = hair\_dryer.dropna ()  microwave = microwave.dropna ()  pacifier = pacifier.dropna ()  def try1():  def gen\_star\_sent(n):  tmp1=hair\_dryer[hair\_dryer['star\_rating']==n]['review\_body']  tmp2=microwave[microwave['star\_rating']==n]['review\_body']  tmp3=pacifier[pacifier['star\_rating']==n]['review\_body']  star\_str=''  for i in tqdm(tmp1.values):  for j in pre\_process(i):  star\_str=star\_str+' '+j  for i in tqdm(tmp2.values):  for j in pre\_process (i):  star\_str = star\_str + ' ' + j  for i in tqdm(tmp3.values):  for j in pre\_process (i):  star\_str = star\_str + ' ' + j  print()  return star\_str  one\_star\_sen=gen\_star\_sent(1)  two\_star\_sen=gen\_star\_sent(2)  three\_star\_sen=gen\_star\_sent(3)  four\_star\_sen=gen\_star\_sent(4)  five\_star\_sen=gen\_star\_sent(5)  # one\_star\_sen=hair\_dryer[hair\_dryer['star\_rating']==1]['review\_body']+microwave[microwave['star\_rating']==1]['review\_body']+pacifier[pacifier['star\_rating']==1]['review\_body']  # keywords=jieba.analyse.extract\_tags(one\_star\_sen, topK=20, withWeight=False, allowPOS=())  # print(keywords)  w = wordcloud.WordCloud(max\_words=50)  w.generate(one\_star\_sen)  w.to\_file('output1.png')  # keywords=jieba.analyse.extract\_tags(two\_star\_sen, topK=20, withWeight=False, allowPOS=())  # print(keywords)  w = wordcloud.WordCloud(max\_words=50)  w.generate(two\_star\_sen)  w.to\_file('output2.png')  keywords=jieba.analyse.extract\_tags(three\_star\_sen, topK=20, withWeight=False, allowPOS=())  print(keywords)  w = wordcloud.WordCloud(max\_words=50)  w.generate(three\_star\_sen)  w.to\_file('output3.png')  # keywords=jieba.analyse.extract\_tags(four\_star\_sen, topK=20, withWeight=False, allowPOS=())  # print(keywords)  w = wordcloud.WordCloud(max\_words=50)  w.generate(four\_star\_sen)  w.to\_file('output4.png')  # keywords=jieba.analyse.extract\_tags(five\_star\_sen, topK=20, withWeight=False, allowPOS=())  # print(keywords)  w = wordcloud.WordCloud(max\_words=50)  w.generate(five\_star\_sen)  w.to\_file('output5.png')  def try2():  # words\_list=set()  # with open('emotion\_dict/words\_list.txt','r',encoding='utf-8') as f:  # for line in f:  # words\_list.add(line.replace('\n',''))  # def gen\_star\_sent(n):  # tmp1 = hair\_dryer[hair\_dryer['star\_rating'] == n]['review\_body']  # tmp2 = microwave[microwave['star\_rating'] == n]['review\_body']  # tmp3 = pacifier[pacifier['star\_rating'] == n]['review\_body']  #  # star\_str = ''  # for i in tqdm(tmp1.values):  # for j in pre\_process(i):  # if j in words\_list:  # star\_str = star\_str + ' ' + j  # for i in tqdm(tmp2.values):  # for j in pre\_process (i):  # if j in words\_list:  # star\_str = star\_str + ' ' + j  # for i in tqdm(tmp3.values):  # for j in pre\_process (i):  # if j in words\_list:  # star\_str = star\_str + ' ' + j  # print()  #  # return star\_str  #  # one\_star\_sen = gen\_star\_sent (1)  # two\_star\_sen = gen\_star\_sent (2)  # three\_star\_sen = gen\_star\_sent (3)  # four\_star\_sen = gen\_star\_sent (4)  # five\_star\_sen = gen\_star\_sent (5)  # star\_sent = {}  # star\_sent['one'] = one\_star\_sen  # star\_sent['two'] = two\_star\_sen  # star\_sent['three'] = three\_star\_sen  # star\_sent['four'] = four\_star\_sen  # star\_sent['five'] = five\_star\_sen  # pickle.dump (star\_sent, open ('star\_sent\_cloud.pkl', 'wb'))  star\_sent=pickle.load(open('star\_sent\_cloud.pkl','rb'))  w = wordcloud.WordCloud (max\_words=50)  w.generate (star\_sent['one'])  w.to\_file ('output1.png')  w = wordcloud.WordCloud (max\_words=50)  w.generate (star\_sent['two'])  w.to\_file ('output2.png')  w = wordcloud.WordCloud (max\_words=50)  w.generate (star\_sent['three'])  w.to\_file ('output3.png')  w = wordcloud.WordCloud (max\_words=50)  w.generate (star\_sent['four'])  w.to\_file ('output4.png')  w = wordcloud.WordCloud (max\_words=50)  w.generate (star\_sent['five'])  w.to\_file ('output5.png')  def try3(num):  words\_list = set ()  with open ('emotion\_dict/words\_list.txt', 'r', encoding='utf-8') as f:  for line in f:  words\_list.add (line.replace ('\n', ''))  def gen\_star\_sent1(n):  tmp1 = hair\_dryer[hair\_dryer['star\_rating'] == n]['review\_body']  tmp2 = microwave[microwave['star\_rating'] == n]['review\_body']  tmp3 = pacifier[pacifier['star\_rating'] == n]['review\_body']  star\_str = []  for i in tqdm (tmp1.values):  for j in pre\_process (i):  if j in words\_list:  star\_str .append(j)  for i in tqdm (tmp2.values):  for j in pre\_process (i):  if j in words\_list:  star\_str .append(j)  for i in tqdm (tmp3.values):  for j in pre\_process (i):  if j in words\_list:  star\_str .append(j)  print()  return star\_str  # one\_star\_sen = gen\_star\_sent1 (1)  # two\_star\_sen = gen\_star\_sent1 (2)  # three\_star\_sen = gen\_star\_sent1 (3)  # four\_star\_sen = gen\_star\_sent1 (4)  five\_star\_sen = gen\_star\_sent1 (5)  # star\_sent={}  # star\_sent['one']=one\_star\_sen  # star\_sent['two']=two\_star\_sen  # star\_sent['three']=three\_star\_sen  # star\_sent['four']=four\_star\_sen  # star\_sent['five']=five\_star\_sen  # pickle.dump(star\_sent,open('star\_sent\_count.pkl','wb'))  star\_sent=pickle.load(open('star\_sent\_count.pkl','rb'))  words\_set=[]  words\_dict={}  word\_counts = collections.Counter (star\_sent['five']) # 对分词做词频统计  word\_counts\_top20 = word\_counts.most\_common (num) # 获取前20最高频的词  # print (word\_counts\_top20) # 输出检查  word\_counts\_top20 = dict (word\_counts\_top20)  words\_set += list (word\_counts\_top20.keys ())  words\_dict['5'] = word\_counts\_top20  y5 = list(word\_counts\_top20.values())  x5 = [5 for \_ in range (len (y5))]  word\_counts = collections.Counter (star\_sent['four']) # 对分词做词频统计  word\_counts\_top20 = word\_counts.most\_common (num) # 获取前20最高频的词  # print (word\_counts\_top20) # 输出检查  word\_counts\_top20 = dict (word\_counts\_top20)  words\_set += list (word\_counts\_top20.keys ())  words\_dict['4'] = word\_counts\_top20  y4 = list(word\_counts\_top20.values())  x4 = [4 for \_ in range (len (y4))]  word\_counts = collections.Counter (star\_sent['three']) # 对分词做词频统计  word\_counts\_top20 = word\_counts.most\_common (num) # 获取前20最高频的词  # print (word\_counts\_top20) # 输出检查  word\_counts\_top20 = dict (word\_counts\_top20)  words\_set += list (word\_counts\_top20.keys ())  words\_dict['3'] = word\_counts\_top20  y3 = list(word\_counts\_top20.values())  x3 = [3 for \_ in range (len (y3))]  word\_counts = collections.Counter (star\_sent['two']) # 对分词做词频统计  word\_counts\_top20 = word\_counts.most\_common (num) # 获取前20最高频的词  # print (word\_counts\_top20) # 输出检查  word\_counts\_top20 = dict (word\_counts\_top20)  words\_set += list (word\_counts\_top20.keys ())  words\_dict['2'] = word\_counts\_top20  y2 = list(word\_counts\_top20.values())  x2 = [2 for \_ in range (len (y2))]  word\_counts = collections.Counter (star\_sent['one']) # 对分词做词频统计  word\_counts\_top20 = word\_counts.most\_common (num) # 获取前20最高频的词  word\_counts\_top20=dict(word\_counts\_top20)  words\_set+=list(word\_counts\_top20.keys())  words\_dict['1']=word\_counts\_top20  # print (word\_counts\_top20) # 输出检查  y1 = list(word\_counts\_top20.values())  x1=[1 for \_ in range(len(y1))]  words\_set=set(words\_set)  sorted(words\_set)  dic={'1':[],'2':[],'3':[],'4':[],'5':[]}  for i in words\_set:  for j in range(1,6):  if i in words\_dict[str(j)].keys():  dic[str(j)].append(words\_dict[str(j)][i])  else:  dic[str(j)].append(0)  data=pd.DataFrame(dic,index=words\_set)  data.to\_csv('../Data/word\_count'+str(num)+'.csv',encoding='utf-8')  cmap = sns.cubehelix\_palette (start=1.5, rot=3, gamma=0.8, as\_cmap=True)  sns.heatmap (data,linewidths = 0.05, vmax=5000, vmin=50, cmap=cmap)  plt.show()  plt.savefig('words\_hot'+str(num)+'.png')    try3(10)  # try2() | | | | |