CSCI544 Homework1 Report

Mengsha Wen

In [1]:

```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
import re
from bs4 import BeautifulSoup
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\86923\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

1. Data Preparation

Read Data & Keep Reviews and Ratings

In [2]:

```
df = pd.read_csv('C:/Users/86923/Desktop/Gradute/lecture/CS544/HW/amazon_reviews_us_Kitchen_v1_00.ts
                 sep='\t',
                 usecols=['star_rating','review_body'])
```

Samples of Reviews

The following are three samples of reviews along with corresponding ratings.

In [3]:

```
print(df.iloc[0]['star_rating'],' ', df.iloc[0]['review_body'])
print(df.iloc[464650]['star_rating'],' ', df.iloc[464650]['review_body'])
print(df.iloc[949]['star_rating'],' ', df.iloc[949]['review_body'])
```

```
5.0   Beautiful.  Looks great on counter.
3.0   Great mug and very large, but the graphics are too delicate. Left mine soaking
in water and now all the graphics are peeling off like wet paper!  Can't even imagin
e what would happen in a dishwasher. Well I can. You would be left with a very large
black mug.
3.0   Knife sharpener work as advertised. Only reason it didn't core higher is the r
ubber base smells like fresh driveway sealer (really bad) one week later still stink
s to high heaven.
```

Statistics of Reviews

There are total 4,875,088 reviews in this dataset.

In [4]:

```
df.shape[0]
```

Out[4]:

```
4875088
```

After dropping the rows with missing values (eg., Nan), that leaves 4,874,842 items.

In [5]:

```
#df[df['star_rating'].isnull()]
df = df.dropna()
df.shape[0]
```

Out[5]:

```
4874842
```

According to the analysis, the statistics of the ratings is shown in figure 1 below. There are 3,124,740 reviews received 5 rating; 731,718 reviews received 4 stars; 349,552 reviews are 3 stars; and 241,945 reviews and 426,887 reviews received 2 ratings and 1 ratings, respectively.

In [6]:

```
df['star_rating'].value_counts()
```

Out[6]:

```
5.0    3124740
4.0     731718
1.0     426887
3.0     349552
2.0     241945
Name: star_rating, dtype: int64
```

Labelling Reviews:

The reviews with rating 4,5 are labelled to be 1 and 1,2 are labelled as 0. Discard the reviews with rating 3'

In [7]:

```
# discard rating 3
df1 = df[~df['star_rating'].isin([3])]
df1.shape[0]
```

Out[7]:

```
4525290
```

In [8]:

```
# labelling
df1['label'] = df1.star_rating.apply(lambda x: 1 if(x>=4) else 0)
df1['label'].value_counts()
```

```
<ipython-input-8-49ec2d5f87c8>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df1['label'] = df1.star_rating.apply(lambda x: 1 if(x>=4) else 0)
```

Out[8]:

```
1    3856458
0     668832
Name: label, dtype: int64
```

After creating binary labels, there are 3,856,458 items labeled to 1, and 668,832 items labeled to 0. Besides, there are 349,552 reviews with the rating 3 which will be discarded.

In [9]:

```
print("number of Label 0:",df1['label'].value_counts()[0])
print("number of Label 1:",df1['label'].value_counts()[1])
print("number of rating 3:",df['star_rating'].value_counts()[3])
```

```
number of Label 0: 668832
number of Label 1: 3856458
number of rating 3: 349552
```

After doing all of things mentioned above, the dataset is like as following

In [10]:

```
data = df1[['review_body','label']]
data.iloc[-5:]
```

Out[10]:

	review_body	label
4875083	After a month of heavy use, primarily as a chi...	1
4875084	I've used my Le Creuset enameled cast iron coo...	1
4875085	According to my wife, this is \\"the best birt...	1
4875086	Hoffritz has a name of producing a trendy and ...	1
4875087	OK. I was late to snap to the Dead Reckoners. ...	1

I select 200000 reviews randomly with 100,000 positive and 100,000 negative reviews by selecting respectively and then shuffling.

In [11]:

```
data_pos = (data[data['label'] == 1]).sample(n=100000, random_state = 1)
data_neg = (data[data['label'] == 0]).sample(n=100000, random_state = 1)
```

In [12]:

```
from sklearn.utils import shuffle
# merge two sub dataset and shuffle
new_data = shuffle(pd.concat([data_pos, data_neg]))
new_data.head()
```

Out[12]:

	review_body	label
3247950	It only held the vacuum for two days. The rabb...	0
4574560	Great price for a set of wine glasses. You get...	1
785835	Did not work as I had expected so don't use at...	0
2100979	Returned this, hard to open huge spout.	0
2207513	Love this thing! Does exactly what it says wi...	1

train-test split

After randomly selecting 100,000 positive reviews and 100,000 negative reviews from dataset and splitting training and testing dataset, there are 160,000 items in training set and 40,000 in testing set.

In [13]:

```
from sklearn.model_selection import train_test_split
label = new_data['label']
reviews = new_data.drop('label',axis=1)
X_train, X_test, y_train, y_test = train_test_split(reviews, label, random_state=42, test_size=0.2)
#print(len(X_train), len(X_test), len(y_train), len(y_test))
```

2. Data Cleaning

The average length of the reviews in terms of character length in the dataset before data cleaning is shown below.The length is around 324

In [14]:

```
print("The average length of the reviews in terms of character"+
      " length before data cleaning: ",new_data["review_body"].apply(len).mean())
```

```
The average length of the reviews, in terms of character length before data cleanin
g:  324.148235
```

2.1 Perform contractions on the reviews.

In [15]:

```
# using build-in library to deal with contractions on the reviews
import contractions
def contractionfunction(s):
    words = []
    for word in s.split():
        words.append(contractions.fix(word))
    new_str = ' '.join(words)
    return new_str
```

2.2 Convert the all reviews into the lower case.

2.3 Remove the HTML and URLs from the reviews.

2.4 Remove non-alphabetical characters.

2.5 Remove the extra tab, Line break, spaces, etc. between the words.

The following are examples of cleaning X_train and X_test dataset, respectively.

In [16]:

```
import re
# dealing with X_train
reviews_train = []
for r in X_train.review_body:
    # contraction
    r = contractionfunction(r)
    # change all letters into lower case
    r = r.lower()
    # dealing with html
    r = re.sub(r'</?\w+[^>]*>',' ',r)
    # dealing with urls
    r = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+',
               '', r)
    # dealing with non-alphabetical characters
    r = re.sub("[^a-z]+",' ',r)
    # dealing with \n, \b, \r, \t
    r = re.sub(r'\r|\n|\t','',r)
    # dealing with ,/./:/extra spaces....
    r = re.sub(r'[^\w\s]','',r)
    # perform contractions on the reviews
    reviews_train.append(r)
reviews_train[0]
```

Out[16]:

```
'a great whisk can really do a lot to help with not just the mixing of ingredients b
ut determining the consistency of what you are cooking i was offered the easylife st
ainless steel whisk to try in exchange for a review and i can tell you that it is de
finitely designed to help you not just stir your ingredients but help you in prepari
ng the best product possible cleaning some of the whisks i have had in the past has
been an issue but this one cleans easily though i will say i handwashed it as oppose
d to putting it in the dishwasher and without the hassle of ingredients getting stuc
k on it recommend this whisk highly for the price you will not be disappointed '
```

In [17]:

```
# dealing with X_test
reviews_test = []
for r in X_test.review_body:
    #contraction
    r = contractionfunction(r)
    # change all letters into lower case
    r = r.lower()
    # dealing with html
    r = re.sub(r'</?\w+[^>]*>',' ',r)
    # dealing with urls
    r = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+',
               '', r)
    # dealing with non-alphabetical characters
    r = re.sub("[^a-z]+",' ',r)
    # dealing with \n, \b, \r, \t
    r = re.sub(r'\r|\n|\t','',r)
    # dealing with ,/./:/extra spaces....
    r = re.sub(r'[^\w\s]','',r)
    # perform contractions on the reviews
    reviews_test.append(r)
reviews_test[0]
```

Out[17]:

```
'thought it was going to be a little bigger but that is okay the salt comes out real
ly good but not the pepper '
```

The average length of the reviews in terms of character length in the dataset after data cleaning is shown below.

In [18]:

```
df_train = pd.DataFrame(reviews_train, columns=['reviews'])
df_test = pd.DataFrame(reviews_test, columns=['reviews'])

avg_len = (df_train["reviews"].apply(len).mean()*160000
           + df_test["reviews"].apply(len).mean()*40000) / 200000
print("The average length of the reviews in terms of character"
      +" length after data cleaning  and before preprocessing: ",avg_len)
```

```
The average length of the reviews in terms of character length after data cleaning
and before preprocessing:  310.93149
```

3. Pre-processing

3.1 remove the stop words

In [19]:

```
from nltk.corpus import stopwords

def rm_stopwords(review):
    stop_words = set(stopwords.words('english'))
    words = [w for w in review.split(' ') if w not in stop_words]
    s = ' '.join(words)
    return s
```

In [20]:

```
# remove the stop words in train and test dataset respectively
reviews_train_rmstopwords = []
reviews_test_rmstopwords = []
for review in reviews_train:
    reviews_train_rmstopwords.append(rm_stopwords(review))
for review in reviews_test:
    reviews_test_rmstopwords.append(rm_stopwords(review))
```

3.2 perform lemmatization

In [21]:

```
from nltk.stem import WordNetLemmatizer

def per_lemmatize(review):
    lm = WordNetLemmatizer()
    words = [lm.lemmatize(w) for w in review.split(' ')]
    s = ' '.join(words)
    return s
```

In [22]:

```
reviews_train_lemmatize = []
reviews_test_lemmatize = []
for review in reviews_train_rmstopwords:
    reviews_train_lemmatize.append(per_lemmatize(review))
for review in reviews_test_rmstopwords:
    reviews_test_lemmatize.append(per_lemmatize(review))
```

Below are three sample reviews before data cleaning and preprocessin

In [23]:

```
print("Three sample reviews before data cleaning and preprocessing:\n")
print(X_train.iloc[0].review_body,'\n')
print(X_train.iloc[799].review_body,'\n')
print(X_train.iloc[109485].review_body,'\n')
```

```
Three sample reviews before data cleaning and preprocessing:

A great whisk can really do a lot to help with not just the mixing of ingredients bu
t determining the consistency of what you are cooking.<br /><br />I was offered the
1Easylife stainless steel whisk to try in exchange for a review, and I can tell you
that it is definitely designed to help you not just stir your ingredients but help y
ou in preparing the best product possible.<br /><br />Cleaning some of the whisks I
have had in the past has been an issue, but this one cleans easily ( though I will s
ay I handwashed it as opposed to putting it in the dishwasher) and without the hassl
e of ingredients getting stuck on it.<br /><br />Recommend this whisk highly. For th
e price you won't be disappointed.

I am very disappointed to say that the nonstick surface started bubbling up and then
peeling off after only a few uses.  Until that point, I was fairly happy with this s
killet, although it is quite heavy, large, and hogs the space in my cabinet.  Howeve
r, I have also noticed recently that the thermostat does what others have noted in t
heir reviews....It's all or nothing as it goes on and off to maintain temperature.
This is particularly bothersome when frying eggs.  I did get about a dozen acceptabl
e uses out of it, but now it's going in the trash.  Good-bye, Hamilton Beach product
s!  I won't be wasting anymore of my money on them.

Accuracy of reading not satisfactory. Useless with chicken or turkey as I found out
on Chrisdtmas Day - digital reading all laround the place, it stated that the turkey
was cooked after 21/2 hours which was totally misleading as it took a further 2 hour
s total 41/2 hours. so much for the digital thermometer??? Bad purchase and expensiv
e!
```

In [24]:

```
# three samples after revoming stop words
print("Three sample reviews after data cleaning and preprocessing:\n")
print(reviews_train_lemmatize[0],'\n')
print(reviews_train_lemmatize[799],'\n')
print(reviews_train_lemmatize[109485],'\n')
```

```
Three sample reviews after data cleaning and preprocessing:

great whisk really lot help mixing ingredient determining consistency cooking offere
d easylife stainless steel whisk try exchange review tell definitely designed help s
tir ingredient help preparing best product possible cleaning whisk past issue one cl
ean easily though say handwashed opposed putting dishwasher without hassle ingredien
t getting stuck recommend whisk highly price disappointed

disappointed say nonstick surface started bubbling peeling us point fairly happy ski
llet although quite heavy large hog space cabinet however also noticed recently ther
mostat others noted review nothing go maintain temperature particularly bothersome f
rying egg get dozen acceptable us going trash good bye hamilton beach product wastin
g anymore money

accuracy reading satisfactory useless chicken turkey found chrisdtmas day digital re
ading laround place stated turkey cooked hour totally misleading took hour total hou
r much digital thermometer bad purchase expensive
```

Above are three samples after data cleaning and pre-processing.

In [25]:

```
df_train = pd.DataFrame(reviews_train_lemmatize, columns=['reviews'])
df_test = pd.DataFrame(reviews_train_lemmatize, columns=['reviews'])

avg_len = (df_train["reviews"].apply(len).mean()*160000 +
           df_test["reviews"].apply(len).mean()*40000) / 200000
print("The average length of the reviews in terms of character"+
        " length after data cleaning and preprocessing: ",avg_len)
```

```
The average length of the reviews in terms of character length after data cleaning a
nd preprocessing:  191.10123125000004
```

4. TF-IDF Feature Extraction

In [26]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,2))
x_train_new = vectorizer.fit_transform(reviews_train_lemmatize)
x_test_new = vectorizer.transform(reviews_test_lemmatize)
```

In [27]:

```
# deal with the label, convert Series to ndarry type
y_train_new = y_train.values
y_test_new = y_test.values
```

At this point, *x_train_new*, *y_train_new* and *x_test_new*, *y_test_new* are the train and test data set after all of processes

5. Perceptron

In [28]:

```
from sklearn.linear_model import Perceptron

clf = Perceptron()
clf.fit(x_train_new, y_train_new)
# predict
y_train_pred = clf.predict(x_train_new)
y_test_pred = clf.predict(x_test_new)
```

In [29]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import f1_score, confusion_matrix, roc_curve, roc_auc_score
def metric_measure(y_train_new, y_train_pred, y_test_new, y_test_pred):
    a_train = accuracy_score(y_train_new, y_train_pred)
    p_train = precision_score(y_train_new, y_train_pred, average='binary')
    r_train = recall_score(y_train_new, y_train_pred, average='binary')
    f1_train = f1_score(y_train_new, y_train_pred, average='binary')
    print('accuracy of train set is:',a_train)
    print('precision of train set is:',p_train)
    print('recall of train set is:', r_train)
    print('F1-score of train set is:', f1_train)

    a_test = accuracy_score(y_test_new, y_test_pred)
    p_test = precision_score(y_test_new, y_test_pred, average='binary')
    r_test = recall_score(y_test_new, y_test_pred, average='binary')
    f1_test = f1_score(y_test_new, y_test_pred, average='binary')
    print('accuracy of test set is:',a_test)
    print('precision of test set is:',p_test)
    print('recall of test set is:', r_test)
    print('F1-score of test set is:', f1_test)
```

Metrics measure results of Perceptron

In [30]:

```
print("The 8 metrics values of Perceptron:\n")
metric_measure(y_train_new, y_train_pred, y_test_new, y_test_pred)
```

```
The 8 metrics values of Perceptron:

accuracy of train set is: 0.99415
precision of train set is: 0.9943468982816068
recall of train set is: 0.9939747763930991
F1-score of train set is: 0.9941608025153466
accuracy of test set is: 0.8979
precision of test set is: 0.8977829402555426
recall of test set is: 0.8961536522659677
F1-score of test set is: 0.8969675563852868
```

6.1 SVM

In [31]:

```
from sklearn.svm import LinearSVC
clf_svm = LinearSVC()
clf_svm.fit(x_train_new, y_train_new)
y_train_pred = clf_svm.predict(x_train_new)
y_test_pred = clf_svm.predict(x_test_new)
```

Metrics measure results of SVM

In [32]:

```
print("The 8 metrics values of SVM:\n")
metric_measure(y_train_new, y_train_pred, y_test_new, y_test_pred)
```

```
The 8 metrics values of SVM:

accuracy of train set is: 0.99520625
precision of train set is: 0.9941741771647662
recall of train set is: 0.9962700996719185
F1-score of train set is: 0.9952210349232062
accuracy of test set is: 0.9161
precision of test set is: 0.9127473077886301
recall of test set is: 0.9186368906588698
F1-score of test set is: 0.9156826290136174
```

6.2 Logistic Regression

In [33]:

```
from sklearn.linear_model import LogisticRegression
clf_lr = LogisticRegression(random_state = 0).fit(x_train_new, y_train_new)
y_train_pred = clf_lr.predict(x_train_new)
y_test_pred = clf_lr.predict(x_test_new)
```

```
D:\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWar
ning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Metrics measure results of Logictic Regression

In [34]:

```
print("The 8 metrics values of Logictic Regression:\n")
metric_measure(y_train_new, y_train_pred, y_test_new, y_test_pred)
```

```
The 8 metrics values of Logictic Regression:

accuracy of train set is: 0.9440375
precision of train set is: 0.9478271806804603
recall of train set is: 0.940047153923888
F1-score of train set is: 0.9439211363579427
accuracy of test set is: 0.9091
precision of test set is: 0.9090955002272613
recall of test set is: 0.9074456823108333
F1-score of test set is: 0.9082698420707402
```

7. Naive Bayes

In [35]:

```
from sklearn.naive_bayes import MultinomialNB
clf_mnb = MultinomialNB()
clf_mnb.fit(x_train_new, y_train_new)
y_train_pred = clf_mnb.predict(x_train_new)
y_test_pred = clf_mnb.predict(x_test_new)
```

Metrics measure results of Naive Bayes

In [36]:

```
print("The 8 metrics values of Naive Bayes:\n")
metric_measure(y_train_new, y_train_pred, y_test_new, y_test_pred)
```

```
The 8 metrics values of Naive Bayes:

accuracy of train set is: 0.95513125
precision of train set is: 0.9651388074540495
recall of train set is: 0.9445629529832965
F1-score of train set is: 0.95474003417036
accuracy of test set is: 0.897875
precision of test set is: 0.9175148430873622
recall of test set is: 0.8725109643595301
F1-score of test set is: 0.8944471719077025
```