HW1-CSCI544

September 9, 2021

[1]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
import re
from bs4 import BeautifulSoup
import contractions
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Perceptron
from sklearn.metrics import confusion_matrix as cm
from sklearn.svm import LinearSVC as SVC
from sklearn.linear_model import LogisticRegression as LR
from sklearn.naive_bayes import MultinomialNB
import warnings
warnings.filterwarnings('ignore')
```

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

[2]:
```
#pip install bs4 # in case you don't have it installed

# Dataset: https://s3.amazonaws.com/amazon-reviews-pds/tsv/
 ↪amazon_reviews_us_Kitchen_v1_00.tsv.gz
```

0.1 Read Data

[3]:
```
df = pd.read_csv("../amazon_reviews_us_Kitchen_v1_00.tsv", sep = "\t",␣
 ↪error_bad_lines=False, warn_bad_lines=False)
```

0.2 Keep Reviews and Ratings

[4]:
```
df = df[['star_rating', 'review_body']]
print("REVIEWS SAMPPLE WITH RATING")
print(df.head(3))
print("STATISIICS OF THE REVIEWS: ")
print("rating 1: ", len(df[df.star_rating == 1]))
print("rating 2: ", len(df[df.star_rating == 2]))
print("rating 3: ", len(df[df.star_rating == 3]))
print("rating 4: ", len(df[df.star_rating == 4]))
print("rating 5: ", len(df[df.star_rating == 5]))
df = df.dropna()
```

```
REVIEWS SAMPPLE WITH RATING
   star_rating                                        review_body
0          5.0                Beautiful.  Looks great on counter.
1          5.0  I personally have 5 days sets and have also bo…
2          5.0  Fabulous and worth every penny. Used for clean…
STATISIICS OF THE REVIEWS:
rating 1:  426900
rating 2:  241948
rating 3:  349547
rating 4:  731733
rating 5:  3124759
```

1 Labelling Reviews:

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

[5]:
```
df['label'] = df.apply(lambda row: 1 if row.star_rating > 3 else 0, axis = 1)
r0 = len(df[df.label == 0])
r1 = len(df[df.label == 1])
r3 = len(df[df.star_rating == 3])
df = df[df.star_rating != 3]
print(r0, ", ", r1, ", ", r3)
```

```
1018348 ,  3856296 ,  349539
```

We select 200000 reviews randomly with 100,000 positive and 100,000 negative reviews.

[6]:
```
positive = df[df['label'] == 1]
negative = df[df['label'] == 0]
positive = positive.sample(n = 100000, random_state = 200)
negative = negative.sample(n = 100000, random_state=200)
dataset = pd.concat([positive, negative])
dataset = dataset.reset_index(drop = True)
train=dataset.sample(frac=0.8,random_state=200)
test = dataset.drop(train.index)
```

```
train = train.reset_index(drop = True)
test = test.reset_index(drop = True)
```

2 Data Cleaning

2.1 Convert the all reviews into the lower case.

[7]:
```
aclbdc = (sum(train.review_body.str.len()) + sum(test.review_body.str.len()))/
↪200000
review_sample = list(train.review_body.head(3))
# convert all the reviews to lowercase
train['review_body'] = train['review_body'].str.lower()
test['review_body'] = test['review_body'].str.lower()
```

2.2 remove the HTML and URLs from the reviews

[8]:
```
def remove_html_and_url(x):
    x = re.sub(r'(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)', '',␣
↪x, flags=re.MULTILINE)
    soup = BeautifulSoup(x, 'html.parser')
    x = soup.get_text()
    return x

train.review_body = train.review_body.apply(lambda x: remove_html_and_url(x))
test.review_body = test.review_body.apply(lambda x: remove_html_and_url(x))
```

2.3 remove non-alphabetical characters

[9]:
```
train.review_body = train.review_body.apply(lambda x: re.sub("[^a-zA-Z']+", "␣
↪", x))
test.review_body = test.review_body.apply(lambda x: re.sub("[^a-zA-Z']+", " ",␣
↪x))
```

2.4 Remove the extra spaces between the words

[10]:
```
train.review_body = train.review_body.apply(lambda x: re.sub(' +', ' ', x))
test.review_body = test.review_body.apply(lambda x: re.sub(' +', ' ', x))
```

2.5 perform contractions on the reviews.

[11]:
```
def contractionfunction(s):
    s = contractions.fix(s)
    s = re.sub("[^a-zA-Z]+", " ", s)
    return s
train.review_body = train.review_body.apply(lambda x: contractionfunction(x))
test.review_body = test.review_body.apply(lambda x: contractionfunction(x))
```

```
acladc = (sum(train.review_body.str.len()) + sum(test.review_body.str.len()))/
↪200000
```

3 Pre-processing

3.1 remove the stop words

[12]:
```
aclbdp = (sum(train.review_body.str.len()) + sum(test.review_body.str.len()))/
↪200000
def remove_stopwords(s):
    text_tokens = word_tokenize(s)
    tokens_without_sw = [word for word in text_tokens if not word in stopwords.
↪words('english')]
    return tokens_without_sw
train.review_body = train.review_body.apply(remove_stopwords)
test.review_body = test.review_body.apply(remove_stopwords)
print(train.review_body)
```

```
0         [put, south, facing, window, parts, shades, di…
1         [recipient, loved, gift, makes, satisfied, cus…
2                     [exactly, expected, ordered, product]
3                                                   [love]
4         [handy, gadget, monitor, temps, fridge, differ…
                               …
159995                          [vegetable, peeler, thing]
159996    [pans, best, ever, used, gave, little, cooking…
159997    [boyfriend, moved, new, apartment, needed, dis…
159998                                              [stars]
159999    [worked, days, ring, tightens, bottle, snapped…
Name: review_body, Length: 160000, dtype: object
```

3.2 perform lemmatization

[13]:
```
lemmatizer = WordNetLemmatizer()
def lemmatize(x):
    lemmatize_tokens = [lemmatizer.lemmatize(word) for word in x]
    x = ' '.join(lemmatize_tokens)
    return x

train.review_body = train.review_body.apply(lemmatize)
test.review_body = test.review_body.apply(lemmatize)
review_sample_2 = list(train.review_body.head(3))
acladp = (sum(train.review_body.str.len()) + sum(test.review_body.str.len()))/
↪200000

print("STATISTICS OF THREE CLASES: ")
print("label 0: ", r0, ", label 1: ", r1, ", label 3: ", r3)
```

```
print("Average length of reviews before data cleaning: ", aclbdc)
print("Average length of reviews after data cleaning: ", acladc)

print("Average length of reviews before data pre-processing: ", aclbdp)
print("Average length of reviews after data pre-processing: ", acladp)

print("Sample Reviews before data cleaning \n")
for ele in review_sample:
    print(ele)
print("Sample Reviews after data cleaning and pre-processing:\n")
for ele in review_sample_2:
    print(ele)
```

```
STATISTICS OF THREE CLASES:
label 0:  1018348 , label 1:  3856296 , label 3:  349539
Average length of reviews before data cleaning:  322.619475
Average length of reviews after data cleaning:  301.630745
Average length of reviews before data pre-processing:  301.630745
Average length of reviews after data pre-processing:  184.227195
Sample Reviews before data cleaning

I put these on a south facing window and parts of the shades disintigrated over
time due to the intense sun where I live - Colorado.  Hunter Douglas would not
cover them with their warranty as it states that exposure to the elements voids
the warranty.  They consider the sun an element…I guess you need to use these
shades where the sun doesn't shine. These are way too expensive to be replacing
them every several years.  Just a heads up if you are considering buying these.
I won't spend the money for these again. As a follow up- 8/29/07 - I wrote to
the corporate office and Hunter Douglas is replacing the Sillouettes under
warranty.
The recipient loved her gift and that makes for a very satisfied customer.
Thanks.
Exactly what I expected when I ordered the product.
Sample Reviews after data cleaning and pre-processing:

put south facing window part shade disintigrated time due intense sun live
colorado hunter douglas would cover warranty state exposure element void
warranty consider sun shine way expensive replacing every several year head
considering buying spend money follow wrote corporate office hunter douglas
replacing sillouettes warranty
recipient loved gift make satisfied customer thanks
exactly expected ordered product
```

4 TF-IDF Feature Extraction

[14]:
```
tfidf_vect = TfidfVectorizer(min_df = 0.001)
X_train = tfidf_vect.fit_transform(train['review_body'])
X_train = pd.DataFrame(X_train.toarray(), columns = tfidf_vect.
 ↪get_feature_names())
X_test = tfidf_vect.transform(test['review_body'])
X_test = pd.DataFrame(X_test.toarray(), columns = tfidf_vect.
 ↪get_feature_names())
Y_train = train['label']
Y_test = test['label']
```

5 Perceptron

[21]:
```
def metrics(true, pred):
    tn, fp, fn, tp = cm(true, pred).ravel()
    acc = (tp + tn)/(tn + fp + fn + tp)
    prec = tp/(tp + fp)
    rec = tp / (tp + fn)
    f1 = 2*(rec * prec) / (rec + prec)
    return [acc, prec, rec, f1]

def print_seq(score_list):
    print("%.2f" % score_list[0], "\n%.2f" % score_list[1], "\n%.2f" %␣
 ↪score_list[2], "\n%.2f" % score_list[3])

percept = Perceptron(max_iter = 80, tol = 1e-5, random_state = 200, eta0 = 0.01)
percept.fit(X_train, Y_train)
Y_train_pred = percept.predict(X_train)
train_score = metrics(Y_train, Y_train_pred)

# Predicting on test dataset
Y_test_pred = percept.predict(X_test)
test_score = metrics(Y_test, Y_test_pred)
# print(train_score)
# print(test_score)
# print("TRAIN_SCORES:")
print_seq(train_score)
# print("TEST_SCORES:")
print_seq(test_score)
```

```
0.87
0.84
0.91
0.88
0.86
0.84
```

```
0.90
0.87
```

6 SVM

[22]:
```
svc = SVC(max_iter = 5000)
model_svc = svc.fit(X_train, Y_train)
pred_train_svm = model_svc.predict(X_train)
pred_test_svm = model_svc.predict(X_test)
train_svm_score = metrics(Y_train, pred_train_svm)
test_svm_score = metrics(Y_test, pred_test_svm)

# print(train_svm_score)
# print(test_svm_score)
# print("TRAIN_SCORES:")
print_seq(train_svm_score)
# print("TEST_SCORES:")
print_seq(test_svm_score)
```

```
0.90
0.90
0.89
0.90
0.89
0.90
0.88
0.89
```

7 Logistic Regression

[23]:
```
log_reg = LR(solver='liblinear',random_state=0, C=5, penalty='l2',max_iter=1000)
model = log_reg.fit(X_train, Y_train)
pred_train_LR = model.predict(X_train)
pred_test_LR = model.predict(X_test)
train_LR_score = metrics(Y_train, pred_train_LR)
test_LR_score = metrics(Y_test, pred_test_LR)

# print(train_LR_score)
# print(test_LR_score)
# print("TRAIN_SCORES:")
print_seq(train_LR_score)
# print("TEST_SCORES:")
print_seq(test_LR_score)
```

```
0.90
0.90
0.89
```

```
0.90
0.89
0.90
0.88
0.89
```

8 Naive Bayes

[24]:
```
MNB = MultinomialNB()
MNB.fit(X_train, Y_train)
pred_train_MNB = MNB.predict(X_train)
pred_test_MNB = MNB.predict(X_test)
train_MNB_score = metrics(Y_train, pred_train_MNB)
test_MNB_score = metrics(Y_test, pred_test_MNB)

# print(train_MNB_score)
# print(test_MNB_score)
# print("TRAIN_SCORES:")
print_seq(train_MNB_score)
# print("TEST_SCORES:")
print_seq(test_MNB_score)
```

```
0.87
0.86
0.87
0.87
0.86
0.86
0.86
0.86
```

[ ]:
```
```