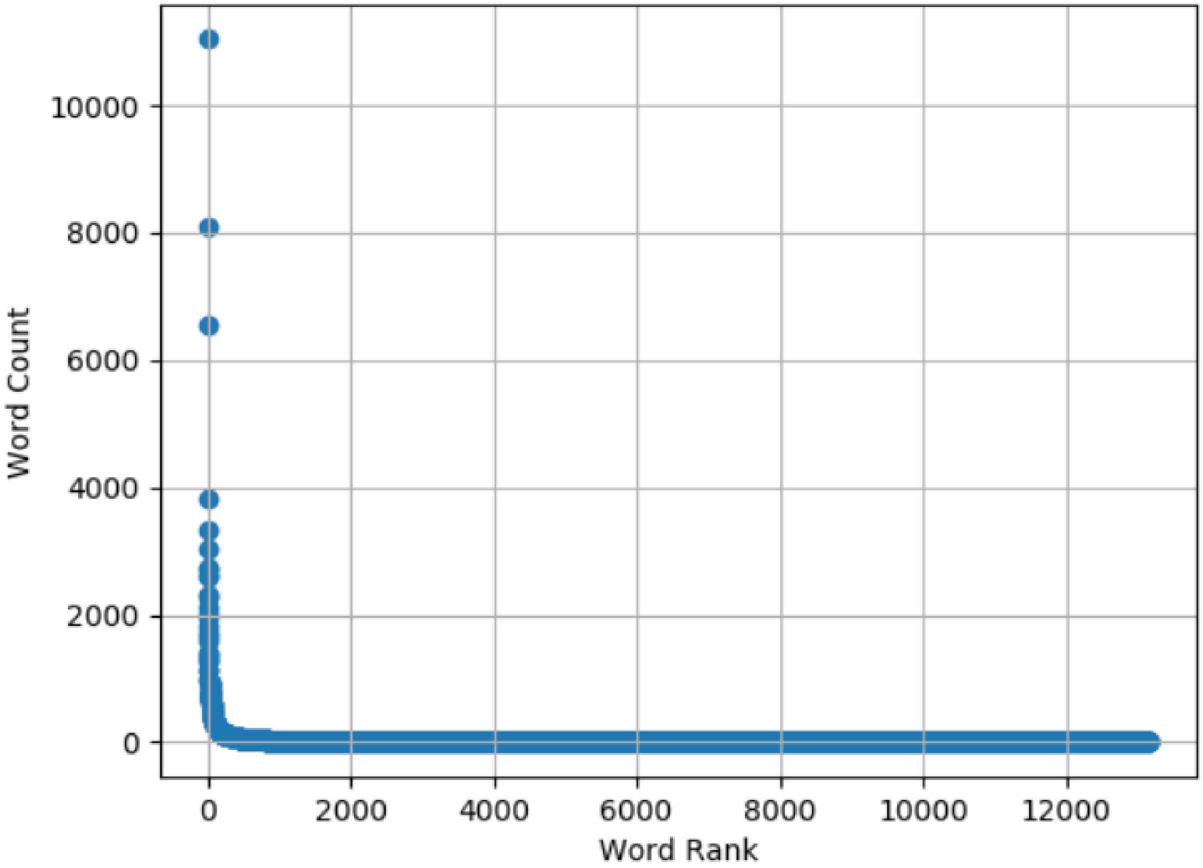
CS498 AMO Homework 7

Team :
Minyuan Gu (minyuan3@illinois.edu, netid minyuan3)
Yanislav Shterev (shterev2@illinois.edu, netid shterev2)

Page 1 Distribution graph (5 points)

Show the distribution graph of words counts vs word rank.



Page 2 Identify the stop words (5 points)

According to the text book page 123, we will examine some of extremely common words and decide if they are good candidates for stop words. First step we will extract them based on certain frequencies.

Choosing the stop words:
From the scatter plot on page 1, we observed a few words have high counts (even more than 10000 times). This indicates we should list out those most frequently occurring words. We tried different max_df settings (e.g. max_df=0.5 means if a word occurs in more than 50% of the documents/samples it is considered to be listed out for our further examination). We have

max_df=0.42, they are:
'but', 'it', 'my', 'with', 'the', 'on', 'was', 'is', 'for', 'have', 'they', 'that', 'of', 'to', 'in', 'and', 'this'

max_df=0.3, on top of those shown above, it contains extra words as below:
'so', 'at', 'are', 'had', 'you', 'place', 'me', 'be', 'not', 'were', 'we'

max_df=0.2, on top of those shown above, it further contains extra words as below:
'food', 'them', 'back', 'go', 'would', 'out', 'will', 'just', 'time', 'as', 'all', 'no', 'what', 'an', 'service', 'can', 'very', 'if', 'one', 'about', 'like', 'their', 'up', 'from', 'great', 'there', 'get', 'good', 'when', 'or', 'here'

From above, from max_df=0.3 we start to see words with some negative meaning (e.g. 'not'), at max_df=0.2 we see more and more words are included as stop words but expressing either positive or negative opinions, e.g. 'great', 'good', 'like', 'no' (and 'not'), and start touching the meaningful terms, like food, service, time and etc. So we think we should stop at max_df=0.3 but with 'not' excluded from the stop list ('not' usually represents negative meaning so keep it away from stop word list).

In order to achieve that, using sklearn we choose max_df=0.42 and supply the **stop word list as below**:
{'you', 'was', 'is', 'for', 'have', 'of', 'to', 'this', 'we', 'so', 'are', 'had', 'me', 'they', 'in', 'the', 'on', 'be', 'that', 'it', 'were', 'at', 'but', 'my', 'with', 'place', 'and'}

The frequency thresholds we choose are: max_df=0.42, min_df=3. The HW requires minimum word occurrence (instead of min_df), however min_df covers and is slightly better than minimum word occurrence:

> e.g. typo or incorrect spelling by one user (review/document), he/she may spell 'fantastic' to 'fentestic' consistently in his review for multiple times, leading such word only occurs in 1 document but word occurrence could be much higher, say 4,5. E.g:
> "Went there last night, fentestic food, fentestic wine, fentestic service and overall 5 stars fentestic restaurant!"
>
> Such noise will not be avoided by setting minimum word occurrence=4, but WILL by min_df=2. Setting higher minimum word occurrence can be used to compensate, however our sample size is just 2000 so higher threshold may cut off some other meaningful words.

Our settings mean two things:
1. If a word appears more than 42% of the documents, it will be excluded from the BOW.

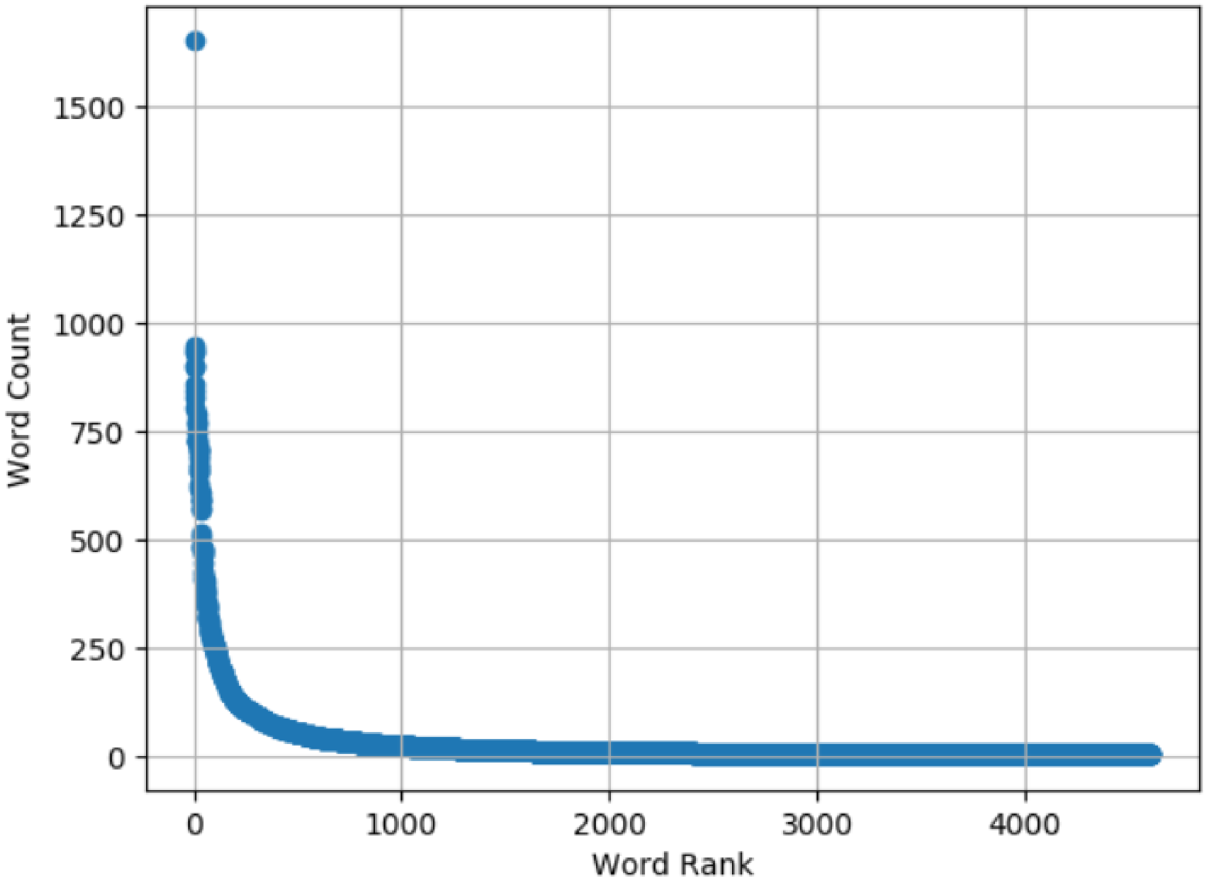
2. If a word appears at a frequency strictly lower than the given threshold, it will be ignored (cutoff). Here we examined min_df of either 2 or 3, which means if a word only occurs in just 1 or 2 documents (reviews in this case), it will be ignored and it is likely to be either a rarely used word, typo or junk characters that isn't useful for our prediction (similar to an outlier). Between 2 and 3, we observed min_df = 3 substantially reduced dimension (around 2000 less) of feature vectors but with slightly better accuracy. So we choose 3. We explored further with min_df=4,5, and 6+, they performed similar however given we just have 2000 samples, we don't want to cut too many low frequency words (e.g. word 'magnificent' only occurs 3 times in 2000 samples, we don't want to set min_df=4 to exclude it).

Using the combination of max_df=0.42, min_df=3 and the stop_word list above, we have input vectors with length of 4606 (words/features).

However, we also observed the vector size 4606 is higher than sample size (2000). We tested with L1 regularization (later pages) we can see that it actually uses only around 300 features – for determining 1 start from 5 star reviews using logistic classifier.
> sum(logistic_clf.coef_[0]==0)
> 4290
We don't extend more discussion for possible lower max_df or higher min_df or much longer stop word list, due to the page & time limitation.

Page 3 Distribution graph again (5 points)

After choosing the stop words, show the distribution graph of words counts vs word rank.



The outstanding point shown above 1500 is the word ‘not’ that we decided to keep it (due to its somewhat negative meaning, so excluded from the stop list). Except that, all other words have their counts lower than 1000.

Page 4 Code snippets (15 points)

Show the snippet of your code that you convert all the reviews into bag-of-words formulation using your chosen stop words and your code for nearest-neighbors with cos-distance.

```
def takeSecond(elem):  #used to sort the words by its count
    return elem[1]

def vectorize_bagOfWd(X, stop_word_lst, maxDf=0.42, minDf=3):
    plt.figure()
    vectorizer = CountVectorizer()
    bagOfWord = vectorizer.fit_transform(X)
    voc_list = vectorizer.vocabulary_.keys()
    wordRank = [[index, item] for index, item in enumerate(bagOfWord.sum(axis=0).tolist()[0])]
    wordRank.sort(key=takeSecond, reverse=True)
    plt.scatter(range(len(wordRank)), np.array(wordRank)[:, 1])
    plt.xlabel('Word Rank')
    plt.ylabel('Word Count')
    plt.grid()
    plt.show()

    vectorizer_cutoff = CountVectorizer(max_df=maxDf, min_df=minDf, stop_words=stop_word_lst)
    bagOfWord_swRemoved = vectorizer_cutoff.fit_transform(X)
    voc_list_swRemoved = vectorizer_cutoff.vocabulary_.keys()
    excluded_word = set(voc_list)-set(voc_list_swRemoved)
    print("Excluded word list size: ", len(excluded_word), "\n Excluded word list:\n", ','.join(excluded_word))
    wordRank_swRemoved = [[index, item] for index, item in enumerate(bagOfWord_swRemoved.sum(axis=0).tolist()[0])]
    wordRank_swRemoved.sort(key=takeSecond, reverse=True)
    plt.figure()
    plt.scatter(range(len(wordRank_swRemoved)), np.array(wordRank_swRemoved)[:, 1])
    plt.xlabel('Word Rank')
    plt.ylabel('Word Count')
    plt.grid()
    plt.show()
    return bagOfWord_swRemoved, vectorizer_cutoff, excluded_word

def text_retrieval(vectorizer, query, Y, query_star='1', top=5, cos_dist_threshold=0.0):
    neigh = NearestNeighbors(metric="cosine")
    neigh.fit(bagOfWord_vectors.toarray(),Y)
    print("Your query is: ", query)
    query = query.lower()
    query_vector = vectorizer.transform([query])
    query_distance, query_results = neigh.kneighbors(query_vector.toarray(), top)
    filtered_index = []
    print(top, " closest results: ")
    for idx, item in enumerate(query_results[0]):
        if query_distance[0,idx]<=(1-cos_dist_threshold):
            print("review #",idx+1, ", cosine distance: ", '%.4f'%(1-query_distance[0,idx]), " : ",
X[query_results[0,idx]][0:199])
            filtered_index.append(query_results[0,idx])
    if len(filtered_index)>0:
        accuracy = sum(Y[filtered_index]==query_star)/len(filtered_index)
    else:
        accuracy = None
    print("Accuracy is", accuracy, ", ", sum(Y[filtered_index]==query_star), "out of total matched",
len(filtered_index), "are correct")
    return accuracy, len(filtered_index)


stop_words = ['you', 'was', 'is', 'for', 'have', 'of', 'to', 'this', 'we', 'so', 'are', 'had', 'me', 'they', 'in',
'the', 'on', 'be', 'that', 'it', 'were', 'at', 'but', 'my', 'with', 'place', 'and']
bagOfWord_vectors, vectorizer, excluded_word = vectorize_bagOfWd(X, stop_words, 0.42, 3)
text_retrieval(vectorizer, "Horrible customer service", Y, query_star='1', top=5)
```

Page 5 Reviews with score (10 points)

Show the original reviews with the distance scores (sklearn returned distance values represent smaller the better match. So we convert it back to cosine distance for printing – larger value means the closer match)

```
Your query is:  Horrible customer service

5  closest results:
review # 1 , cosine distance:  0.6299  :
rogers ...
1) is over priced
2) have horrible customer service
3) faulty and incorrect billing
4) poor customer service
5) not enough options
6) never arrive for an appointment
review # 2 , cosine distance:  0.4576  :
horrible service, horrible customer service, and horrible quality of service!  do not waste your time or money using this company for your pool needs.  dan (602)363-8267 broke my pool filtration syst
review # 3 , cosine distance:  0.4444  :
service was horrible came with a major attitude. payed 30 for lasagna and was no where worth it. won't ever be going back and will never recommend this place. was treated absolutely horrible. horribl
review # 4 , cosine distance:  0.3849  :
customer service was super bad. the pizza was cold by the time they delivered it to me.
review # 5 , cosine distance:  0.3790  :
went to marca today to get a haircut and was given a great service both by front desk - customer service and by georgia, girl who did my hair. i guess i got lucky with her as she has years of experie

Accuracy is 0.8, 4 out of total matched 5 are correct
```

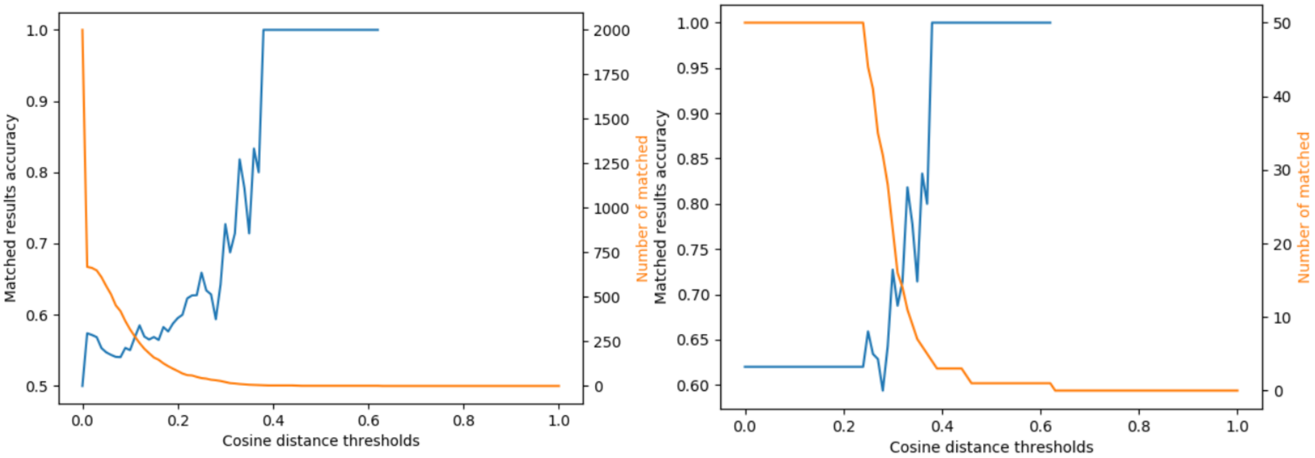
Page 6 Query results (10 points) **Show your document results and explain the reasons that you choose them.**

First of all, among the top 5 matches, first 4 of them are good matches. They have cosine distances: 0.6299, 0.4576, 0.4444 and 0.3849 and their 'star score=1', which matches the query 'Horrible customer service'. They are chosen because:

1. They are all marked as 1 star score, quite negative similar to the query.
2. They have lots of words matching the query 'horrible' – 1st, 2nd & 3rd matches, 'customer' – 1st, 4th matches, 'service' – 1st, 2nd, 3rd, 4th. This will yield high cosine distance to the query.
3. 5th result however is not a good match, but it was returned due it matches 'customer' and 'service' which brings it close enough to the query based on cosine distance. But we don't choose it, since 'customer' and 'service' are not the defining factor for the star of the review, 'horrible' is the word which define it as negative. Later using logistic regression will have the weights adjusted/converged to more important words and correctly classify them (tested 5th review with logistics and it's classified as 5).

And then we look at all other sample with their distances scores retrieved, and plot their accuracy & number of matches against the cosine distance thresholds (the k-nearest returned results must have a cosine distance larger than the threshold in order to be considered as a match).

The left plot we set K=2000 which means we include all the samples. The right plot we zoom in to just nearest 50 samples (K=50). From those plots, we can see, if we raise the threshold the accuracy will increase (tighten the selection) while the number of matches will decrease.



70% accuracy (0.7 in the plot) is achieved at the cosine distance threshold of >=0.33. At cosine distance threshold of 0.33, 11 documents were returned and among them 9 documents are correct matches. They are shown below (wrong one are highlighted in red). We could get more documents however at the cost of low accuracy so we don't go lower than 70% accuracy.

```
review # 1 , cosine distance:  0.6299  :  rogers ...
1) is over priced
2) have horrible customer service
3) faulty and incorrect billing
4) poor customer service
5) not enough options
6) never arrive for an appointment
review # 2 , cosine distance:  0.4576  :  horrible service, horrible customer service, and horrible quality of service!  do not waste your time or money using this company for your pool needs.  dan (602)363-8267 broke my pool filtration syst
review # 3 , cosine distance:  0.4444  :  service was horrible came with a major attitude. payed 30 for lasagna and was no where worth it. won't ever be going back and will never recommend this place. was treated absolutely horrible. horribl
review # 4 , cosine distance:  0.3849  :  customer service was super bad. the pizza was cold by the time they delivered it to me.
review # 5 , cosine distance:  0.3790  :  went to marca today to get a haircut and was given a great service both by front desk - customer service and by georgia, girl who did my hair. i guess i got lucky with her as she has years of experie
review # 6 , cosine distance:  0.3612  :  the service is horrible. it's not bad inside, but really one of the most annoying clubs in vegas. i'm all for vegas clubs, but service here sucks.
review # 7 , cosine distance:  0.3522  :  i come here now for all my car service needs.  in my experiences, the work is always performed in an efficient, effective manner and in good humor.  i appreciate the consistent attention paid to prov
review # 8 , cosine distance:  0.3482  :  horrible customer service!  been with them over 2 years, and after staying with them during my last move they raised my bill almost double for the same services!  sent two emails since i don't have t
review # 9 , cosine distance:  0.3482  :  horrible service....what a mess upon ordering and paying. where is the manager on duty to fix this!
review # 10 , cosine distance:  0.3333  :  they shut down. makes sense, they had terrible service and subpar food..should have listened to your customer base.
review # 11 , cosine distance:  0.3333  :  worse customer service . trying to ask for help .... no one volunteer to help. this is ridiculous
Accuracy is 0.8181818181818182 ,  9 out of total matched 11 are correct
```

Page 7 Accuracy with threshold 0.5 (10 points)

Show your code for creating classifier. Report the accuracy on train and test dataset with threshold 0.5.

```
 def logistic_regression(X_train, X_test, y_train, y_test):
    clf = LogisticRegression(solver='liblinear', max_iter=100)
    #We tested L1 regularisation, due to higher number of features than size of the sample.
    #And it performs similar to L2 (by default)
    #clf = LogisticRegression(solver='liblinear', max_iter=100, penalty='l1')
    clf.fit(X_train, y_train)
    predicted_accurary = clf.score(X_test, y_test)
    train_accuracy = clf.score(X_train, y_train)
    print("Test accuracy is: ", predicted_accurary, " Train accuracy is: ", train_accuracy)
    return clf
```

Test accuracy is: 0.96. Train accuracy is: 0.9994444444444445

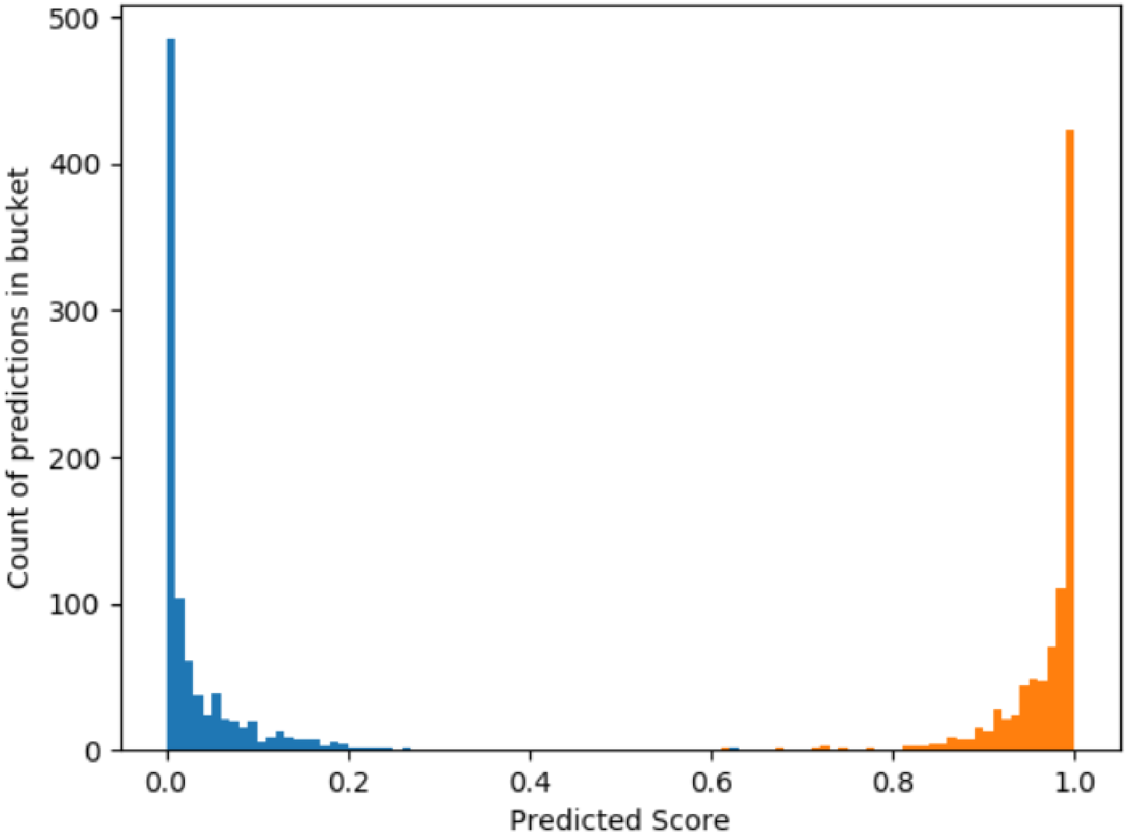
The following code allows using customized probability threshold for prediction.

```
def predict_new_threshold(model, data_X, threshold):
    predicted_p = model.predict_proba(data_X)
    predicted = np.ones(len(data_X)).astype(int).astype(str)
    predicted[predicted_p[:,1]>=threshold]='5'
    return predicted
```

Page 8 Predicted scores (10 points)

Show your code for plotting predicted scores and show the figure.

```
 def plot_histogram(logistic_clf, X_train, y_train):
    predicted_proba = logistic_clf.predict_proba(X_train)
    plt.hist([predicted_proba[y_train=='1'][:,1],predicted_proba[y_train=='5'][:,1]], bins=100,
histtype='stepfilled')
    plt.xlabel("Predicted Score")
    plt.ylabel("Count of predictions in bucket")
    false_positive = predicted_proba[y_train=='1'][predicted_proba[y_train=='1'][:,1]>=0.5]
    false_negative = predicted_proba[y_train=='5'][predicted_proba[y_train=='5'][:,1]<=0.5]
```



Page 9 Accuracy again and curve (20 points) **Report the accuracy on train and test dataset with a different threshold. Explain why you choose that threshold.**

From the above plots, the 2 different classes are actually separated pretty well, and nearly no overlap across 0.5 (the default threshold). In order to further troubleshooting, we list out the top 5 highest predicted scores for negative ones and the top 5 lowest predicted scores for positive samples. See below:

```
Top 5 largest predicted probability for negative reviews:
 0.23707403725589477
 0.24493811118253017
 0.2613864005027665
 0.2732259999116515
 0.6006987072381691
Top 5 smallest predicted probability for positive reviews:
 0.5882788918196694
 0.7301469942797999
 0.739769888043451
 0.7434682520954394
 0.7458043285593453
```

It can be easily seen that there is only one outlier (red) in negative reviews which have a score of 0.6007 causing it to be predicted as a positive review. And let's find it out before we talk about the threshold:

```
X[304]: 'i have friends who love this place, and the food can be good. but my personal interactions with them have been not so great :('
Y[304]: '1'
```

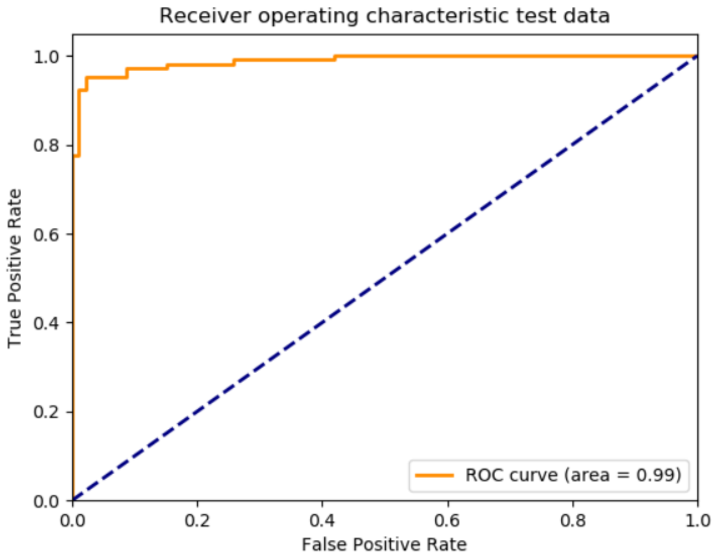
Now we know why it was classified as positive review while it is actually a one star review. Many positive key words like 'love', 'good' and 'great' are included, even there is a 'not' in front of 'so great' but since our model doesn't consider the word sequence, it is beyond our model's capability and tweaking threshold may not be actually useful.

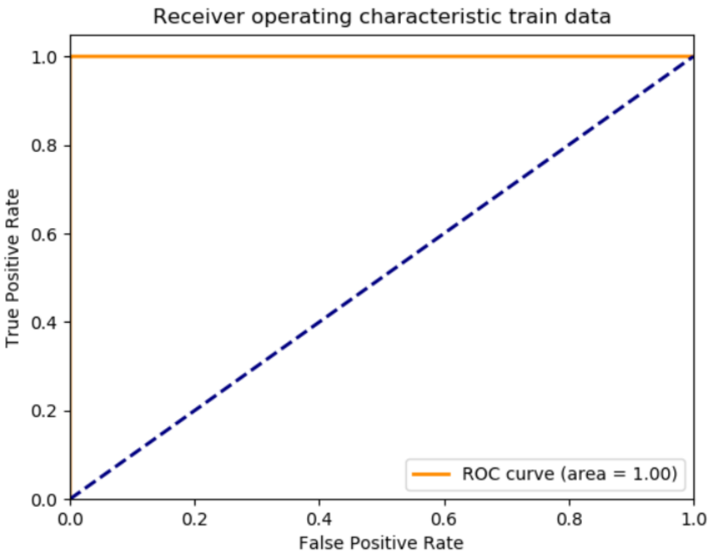
However, for the sake of trying a different threshold, let's ignore this outlier and try to shift the threshold to the mid-point between largest negative review score and smallest positive review score to allow better room for prediction. So we shift our new threshold to this new mid-point 0.43075 (between 0.2732 and 0.5883), we achieved the following accuracy on training and testing data:

```
Training accuracy with threshold changed:  99.94444444444444 %
Test accuracy with threshold changed:  94.0 %
```

It performs very close to the default threshold value (actually slightly worse than the 96% from the default threshold on page 7), so it doesn't suggest it as a better value than the default threshold (0.5).

More discussion on other choices of threshold will be in the next page (ROC). Let's plot the ROC curve.





Page 10 Best threshold (10 points) Choose the threshold that minimizes false positives while maximizing true positives. Explain your reason.

Here we choose test data for the demonstration purpose only, since we explained above training data only misclassified one sample which is beyond this model's capability. From the above ROC plot we can choose a point on the curve with the smallest distance to point [0,1] , or use the Youden index, max(tpr+(1-fpr)), to minimize the false positive while maximizing the true positive.

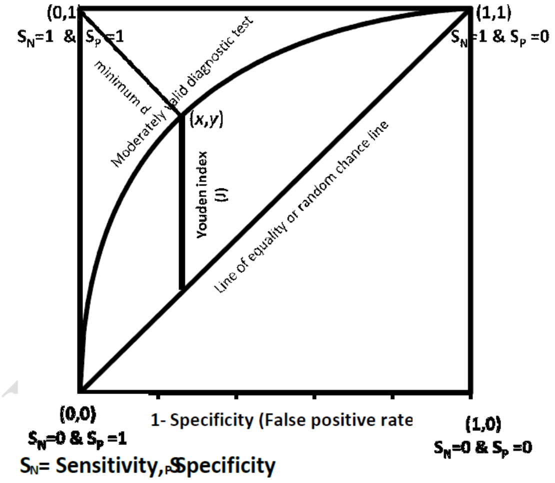

(0,1)
S_N=1 & S_P=1
(1,1)
S_N=1 & S_P =0
minimum d
Moderately valid diagnostic test
(x,y)
Youden index (J)
Line of equality or random chance line
(0,0)
S_N=0 & S_P =1
1- Specificity (False positive rate
(1,0)
S_N=0 & S_P =0
S_N= Sensitivity, S_P Specificity

Figure 3: Finding best cut-off from the ROC curve

The above figure is extracted from the document on http://www.medicalbiostatistics.com/roccurve.pdf

Let's take a look at the below fpr (false positive rate), tpr (true positive rate) and thresholds.

```
fpr
array([0.        , 0.        , 0.        , 0.01075269, 0.01075269,
       0.02150538, 0.02150538, 0.08602151, 0.08602151, 0.15053763,
       0.15053763, 0.25806452, 0.25806452, 0.41935484, 0.41935484,
       1.        ])
tpr
array([0.        , 0.00934579, 0.77570093, 0.77570093, 0.92523364,
       0.92523364, 0.95327103, 0.95327103, 0.97196262, 0.97196262,
       0.98130841, 0.98130841, 0.99065421, 0.99065421, 1.        ,
       1.        ])
thresholds
array([1.99999999e+00, 9.99999990e-01, 8.97555987e-01, 8.88431005e-01,
       6.59130072e-01, 6.45812214e-01, 5.44162755e-01, 4.38586484e-01,
       4.29333589e-01, 1.52490327e-01, 1.47008822e-01, 4.00848079e-02,
       3.82366796e-02, 1.21272114e-02, 1.15356012e-02, 1.38818025e-14])
```

At the point of false positive rate of 0.02150538, we could achieve a true positive rate of 0.95327103 and the associated threshold is 0.544162755. This is very close to the default 0.5. With this new threshold, we have an accuracy similar to the default threshold (0.5):

```
Test accuracy with threshold changed:  96.0 %
```

This concludes the choice of 0.5 and/or 0.54416 are good choices of threshold – minimize false positives and maximize true positive.

Libraries used & Reference:

David Forsyth's book - Applied Machine Learning
Trevor Walker's lecture and sample code – CS-498 Lecture videos
csv – for reading data from csv format: https://docs.python.org/3/library/csv.html
Yelp's reviews dataset - http://courses.engr.illinois.edu/cs498aml/sp2019/homeworks/yelp_2k.csv
Numpy - http://www.numpy.org/
matplotlib - to plot the sum-squared error, mean images and distances plots: https://matplotlib.org/
scikit-learn framework: https://scikit-learn.org
ROC Curve http://www.medicalbiostatistics.com/roccurve.pdf