Project Submission

This notebook will be your project submission. All tasks will be listed in the order of the Courses that they appear in. The tasks will be the same as in the Capstone Example Notebook, but in this submission you ***MUST*** use another dataset. Failure to do so will result in a large penalty to your grade in this course.

Finding your dataset

Take some time to find an interesting dataset! There is a reading discussing various places where datasets can be found, but if you are able to process it, go ahead and use it! Do note, for some tasks in this project, each entry will need 3+ attributes, so keep that in mind when finding datasets. After you have found your dataset, the tasks will continue as in the Example Notebook. You will be graded based on the tasks and your results. Best of luck!

- Link of Dataset: link (https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Furniture_v1_00.tsv.gz)

As Reviewer:

Your job will be to verify the calculations made at each "TODO" labeled throughout the notebook.

First Step: Imports

In the next cell we will give you all of the imports you should need to do your project. Feel free to add more if you would like, but these should be sufficient.

In [1]:
```
import gzip
from collections import defaultdict
import random
import numpy
import scipy.optimize
import string
from sklearn import linear_model
from nltk.stem.porter import PorterStemmer # Stemming
```

Task 1: Data Processing

TODO 1: Read the data and Fill your dataset

In [2]:
```
#YOUR CODE HERE
path = "amazon_reviews_us_Furniture_v1_00.tsv.gz"
f = gzip.open(path,'rt', encoding="utf8")

header = f.readline()
header = header.strip().split('\t')

dataset=[]
for line in f:
    fields = line.strip().split('\t')
    d = dict(zip(header, fields))
    d['star_rating'] = int(d['star_rating'])
    d['helpful_votes'] = int(d['helpful_votes'])
    d['total_votes'] = int(d['total_votes'])
    d['verified_purchase'] = int(d['verified_purchase'] == 'Y')
    dataset.append(d)

print("Size of Dataset: ",len(dataset))
dataset[0]
```

```
Size of Dataset:  792113
```

Out[2]:
```
{'marketplace': 'US',
 'customer_id': '24509695',
 'review_id': 'R3VR960AHLFKDV',
 'product_id': 'B004HB5E0E',
 'product_parent': '488241329',
 'product_title': 'Shoal Creek Computer Desk',
 'product_category': 'Furniture',
 'star_rating': 4,
 'helpful_votes': 0,
 'total_votes': 0,
 'vine': 'N',
 'verified_purchase': 1,
 'review_headline': '... desk is very study and it i has a beautiful finish, I
think that it a just a ...',
 'review_body': 'This desk is very study and it i has  a beautiful finish,  I t
hink that it a just a little  pricey for the size.',
 'review_date': '2015-08-31'}
```

TODO 2: Split the data into a Training and Testing set

First shuffle your data, then split your data. Have Training be the first 80%, and testing be the remaining 20%.

In [3]:
```
#YOUR CODE HERE
#YOUR CODE HERE
trainingSize = int(len(dataset)*0.8)
testSize = int(len(dataset)*0.2)
print(trainingSize,testSize)
trainingSet = list(dataset[:trainingSize])
testSet = list(dataset[trainingSize:])
print(len(trainingSet), len(testSet))
print(len(dataset)==len(trainingSet)+len(testSet))
```

```
633690 158422
633690 158423
True
```

Now delete your dataset

You don't want any of your answers to come from your original dataset any longer, but rather your Training Set, this will help you to not make any mistakes later on, especialy when referencing the checkpoint solutions.

In [4]:
```
#YOUR CODE HERE
del dataset
```

TODO 3: Extracting Basic Statistics

Next you need to answer some questions through any means (i.e. write a function or just find the answer) all based on the **Training Set:**

1. How many entries are in your dataset?
2. Pick a non-trivial attribute (i.e. verified purchases in example), what percentage of your data has this atttribute?
3. Pick another different non-trivial attribute, what percentage of your data share both attributes?

In [5]:
```
print("1. Entries in Dataset: ", len(trainingSet))
```

```
1. Entries in Dataset:  633690
```

In [6]:
```
trainingSet[0]
```

Out[6]:
```
{'marketplace': 'US',
 'customer_id': '24509695',
 'review_id': 'R3VR960AHLFKDV',
 'product_id': 'B004HB5E0E',
 'product_parent': '488241329',
 'product_title': 'Shoal Creek Computer Desk',
 'product_category': 'Furniture',
 'star_rating': 4,
 'helpful_votes': 0,
 'total_votes': 0,
 'vine': 'N',
 'verified_purchase': 1,
 'review_headline': '... desk is very study and it i has a beautiful finish, I
think that it a just a ...',
 'review_body': 'This desk is very study and it i has  a beautiful finish,  I t
hink that it a just a little  pricey for the size.',
 'review_date': '2015-08-31'}
```

In [7]:
```
avgRating = sum([d['star_rating'] for d in trainingSet])

avgRating = avgRating/len(trainingSet)
print("2. Avg Star Rating: ",avgRating)
```

```
2. Avg Star Rating:  4.113351954425665
```

In [8]:
```
### fraction of reviews are from verified purchases
fracVerified = sum([1 for d in trainingSet if d['verified_purchase'] == 1])
fracVerified = fracVerified/len(trainingSet)
print("3. fraction of reviews are from verified purchases: ",fracVerified)
```

```
3. fraction of reviews are from verified purchases:  0.9226167368902776
```

In [9]:
```
### How many total users are there?
users = list(set([d['customer_id'] for d in trainingSet]))
print("4. Total Users",len(users))
```

```
4. Total Users 526828
```

In [10]:
```
### How many total items are there?
items = list(set([d['product_id'] for d in trainingSet]))
print("5. Total Items: ",len(items))
```

```
5. Total Items:  111025
```

In [11]:
```
### What fraction of reviews have 5-star ratings?
frac5star = sum([1 for d in trainingSet if d['star_rating'] == 5])
frac5star = frac5star/len(trainingSet)
print("6. fraction of reviews with 5-star ratings: ",frac5star)
```

```
6. fraction of reviews with 5-star ratings:  0.576703119821995
```

Task 2: Classification

Next you will use our knowledge of classification to extract features and make predictions based on them. Here you will be using a Logistic Regression Model, keep this in mind so you know where to get help from.

TODO 1: Define the feature function

This implementation will be based on ***any two*** attributes from your dataset. You will be using these two attributes to predict a third. Hint: Remember the offset!

In [12]:
```
trainingSet[0]
```

Out[12]:
```
{'marketplace': 'US',
 'customer_id': '24509695',
 'review_id': 'R3VR960AHLFKDV',
 'product_id': 'B004HB5E0E',
 'product_parent': '488241329',
 'product_title': 'Shoal Creek Computer Desk',
 'product_category': 'Furniture',
 'star_rating': 4,
 'helpful_votes': 0,
 'total_votes': 0,
 'vine': 'N',
 'verified_purchase': 1,
 'review_headline': '... desk is very study and it i has a beautiful finish, I
think that it a just a ...',
 'review_body': 'This desk is very study and it i has  a beautiful finish,  I t
hink that it a just a little  pricey for the size.',
 'review_date': '2015-08-31'}
```

In [13]:
```
#FIX THIS

def feature(d):
    feat = [1, d['star_rating'], len(d['review_body'])]
    return feat
```

TODO 2: Fit your model

1. Create your **Feature Vector** based on your feature function defined above.
2. Create your **Label Vector** based on the "verified purchase" column of your training set.
3. Define your model as a **Logistic Regression** model.
4. Fit your model.

In [14]:
```
#YOUR CODE HERE
X_train = [feature(d) for d in trainingSet]
X_test = [feature(d) for d in testSet]
Y_train = [d['verified_purchase'] for d in trainingSet]
Y_test = [d['verified_purchase'] for d in testSet]
```

In [15]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,classification_repo

model = LogisticRegression()
model.fit(X_train,Y_train)
```

```
C:\Users\alsrivas\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\l
inear_model\logistic.py:433: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[15]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

TODO 3: Compute Accuracy of Your Model

1. Make **Predictions** based on your model.
2. Compute the **Accuracy** of your model.

In [16]:
```
#YOUR CODE HERE
prediction = model.predict(X_test)
```

In [17]:
```
print("Accuracy of our logistic model on test dataset:", accuracy_score(Y_test,p
```

```
Accuracy of our logistic model on test dataset: 0.8418537712327124
```

Task 3: Regression

In this section you will start by working though two examples of altering features to further differentiate. Then you will work through how to evaluate a Regularaized model.

In [18]:
```
#CHANGE PATH
path = "amazon_reviews_us_Furniture_v1_00.tsv.gz"

#GIVEN
f = gzip.open(path, 'rt', encoding="utf8")
header = f.readline()
header = header.strip().split('\t')
reg_dataset = []
for line in f:
    fields = line.strip().split('\t')
    d = dict(zip(header, fields))
    d['star_rating'] = int(d['star_rating'])
    reg_dataset.append(d)
```

In [19]:
```
reg_dataset[0]
```

Out[19]:
```
{'marketplace': 'US',
 'customer_id': '24509695',
 'review_id': 'R3VR960AHLFKDV',
 'product_id': 'B004HB5E0E',
 'product_parent': '488241329',
 'product_title': 'Shoal Creek Computer Desk',
 'product_category': 'Furniture',
 'star_rating': 4,
 'helpful_votes': '0',
 'total_votes': '0',
 'vine': 'N',
 'verified_purchase': 'Y',
 'review_headline': '... desk is very study and it i has a beautiful finish, I
think that it a just a ...',
 'review_body': 'This desk is very study and it i has  a beautiful finish,  I t
hink that it a just a little  pricey for the size.',
 'review_date': '2015-08-31'}
```

TODO 1: Unique Words in a Sample Set

We are going to work with a new dataset here, as such we are going to take a smaller portion of the set and call it a Sample Set. This is because stemming on the normal training set will take a very long time. (Feel free to change sampleSet -> reg_dataset if you would like to see the difference for yourself)

1. Count the number of unique words found within the 'review body' portion of the sample set defined below, making sure to **Ignore Punctuation and Capitalization**.
2. Count the number of unique words found within the 'review body' portion of the sample set defined below, this time with use of **Stemming, Ignoring Puctuation, *and* Capitalization**.

In [20]:
```
#GIVEN for 1.
wordCount = defaultdict(int)
punctuation = set(string.punctuation)

#GIVEN for 2.
wordCountStem = defaultdict(int)
stemmer = PorterStemmer() #use stemmer.stem(stuff)

#SampleSet and y vector given
sampleSet = reg_dataset[:2*len(reg_dataset)//10]
y_reg = [d['star_rating'] for d in sampleSet]
```

In [21]:
```
#YOUR CODE HERE
for d in sampleSet:
    r = ''.join([stemmer.stem(c) for c in d['review_body'].lower() if not c in pu
    for w in r.split():
        wordCount[w] += 1
```

TODO 2: Evaluating Classifiers

1. Given the feature function and your counts vector, **Define** your X_reg vector. (This being the X vector, simply labeled for the Regression model)
2. **Fit** your model using a **Ridge Model** with (alpha = 1.0, fit_intercept = True).
3. Using your model, **Make your Predictions**.
4. Find the **MSE** between your predictions and your y_reg vector.

In [22]:
```
#GIVEN FUNCTIONS
def feature_reg(datum):
    feat = [0]*len(words)
    r = ''.join([c for c in datum['review_body'].lower() if not c in punctuation
    for w in r.split():
        if w in wordSet:
            feat[wordId[w]] += 1
    return feat

def MSE(predictions, labels):
    differences = [(x-y)**2 for x,y in zip(predictions,labels)]
    return sum(differences) / len(differences)

#GIVEN COUNTS AND SETS
counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

#Note: increasing the size of the dictionary may require a lot of memory
words = [x[1] for x in counts[:100]]

wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

In [23]:
```
#YOUR CODE HERE
X_reg = [feature_reg(d) for d in sampleSet]
```

In [24]:
```
from sklearn.linear_model import Ridge
model2 = Ridge(alpha = 1.0, fit_intercept = True)
model2.fit(X_reg,y_reg)
prediction2 = model2.predict(X_reg)
print(MSE(prediction2,y_reg))
```

```
1.2373840021129838
```

Task 4: Recommendation Systems

For your final task, you will use your knowledge of simple similarity-based recommender systems to make calculate the most similar items.

The next cell contains some starter code that you will need for your tasks in this section. Notice you should be back to using your **trainingSet**.

In [25]:
```
#GIVEN
attribute_1 = defaultdict(set)
attribute_2 = defaultdict(set)
```

TODO 1: Fill your Dictionaries

1. For each entry in your training set, fill your default dictionaries (defined above).

In [26]:
```
#YOUR CODE HERE
itemNames={}
for d in trainingSet:
    #user,item=d['customer_id'],d['product_id']
    attr1,attr2 = d['customer_id'],d['product_id']
    attribute_1[attr2].add(attr1)
    attribute_2[attr1].add(attr2)
    itemNames[attr2] = d['product_title']
```

In [27]:
```
#GIVEN
def Jaccard(s1, s2):
    numer = len(s1.intersection(s2))
    denom = len(s1.union(s2))
    return numer / denom

def mostSimilar(n, m): #n is the entry index
    similarities = []  #m is the number of entries
    users = attribute_1[n]
    for i2 in attribute_1:
        if i2 == n: continue
        sim = Jaccard(users, attribute_1[n])
        similarities.append((sim,i2))
    similarities.sort(reverse=True)
    return similarities[:m]
```

TODO 1: Fill your Dictionaries

1. Calculate the **10** most similar entries to the **first** entry in your dataset, using the functions defined above.

In [28]:
```
#YOUR CODE HERE
trainingSet[0]
```

Out[28]:
```
{'marketplace': 'US',
 'customer_id': '24509695',
 'review_id': 'R3VR960AHLFKDV',
 'product_id': 'B004HB5E0E',
 'product_parent': '488241329',
 'product_title': 'Shoal Creek Computer Desk',
 'product_category': 'Furniture',
 'star_rating': 4,
 'helpful_votes': 0,
 'total_votes': 0,
 'vine': 'N',
 'verified_purchase': 1,
 'review_headline': '... desk is very study and it i has a beautiful finish, I
think that it a just a ...',
 'review_body': 'This desk is very study and it i has  a beautiful finish,  I t
hink that it a just a little  pricey for the size.',
 'review_date': '2015-08-31'}
```

In [29]:
```
query = trainingSet[0]['product_id']
```

In [30]:
```
query
```

Out[30]:
```
'B004HB5E0E'
```

In [31]:
```
print("10 Most Similar entries to the first entry of dataset: ")
similar_10 = mostSimilar(query,10)
similar_10
```

```
10 Most Similar entries to the first entry of dataset:
```

Out[31]:
```
[(1.0, 'B0148QGRBM'),
 (1.0, 'B0148CQMAC'),
 (1.0, 'B0146CTB9S'),
 (1.0, 'B0141L2CTU'),
 (1.0, 'B013WCT134'),
 (1.0, 'B013TNE7WG'),
 (1.0, 'B013S86C5C'),
 (1.0, 'B013PUW1SU'),
 (1.0, 'B013N9GIG4'),
 (1.0, 'B013L4ZZV0')]
```

In [32]:
```
print("10 Most Similar Item Names to the first entry of dataset: ")
[itemNames[x[1]] for x in similar_10 ]
```

```
10 Most Similar Item Names to the first entry of dataset:
```

Out[32]:
```
['The Charlotte - Handmade Table From The Barrel Shack',
 'Amdirect Shoe Rack Closet Portable Shoe Tower Storage Organizer Cabinet',
 'ancheer office chair',
 'Glamour Home Abigail Metal Accent Table Silver Rustic Square Mirrored Glass T
op Grande Size End Table',
 'Classically Elegant Brookline Tufted Dining Chair (Set of 2) in Aqua Blue- 3
8.0 " H X 19.0 " W X 22.0 " D',
 'SIG MPX Stock Adapter With SB15 Tube Buffer USA Made By Veriforce Tactical',
 'Amdirect Salon Spa Equipment 3 Shelves Rolling Trolley Storage Rack',
 'German Furniture Warehouse 4 Piece Breakfast Nook, Modern Dining Set, Kitchen
Nook Corner Sets Mern',
 'Intex Twin Size Pillow Rest Air Bed Mattress Airbed w/ Pump - Model 66775 Ne
w',
 'Arctic Dreams 10" Cooling Gel Mattress Made in the USA']
```

Finished!

Congratulations! You are now ready to submit your work. Once you have submitted make sure to get started on your peer reviews!