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!

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
#YOUR CODE HERE
f = gzip.open('./amazon_reviews_us_Apparel_v1_00.tsv.gz', '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'] = d['verified_purchase'] == 'Y'
    dataset.append(d)
```

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
from sklearn.model_selection import train_test_split
trainingSet,testSet = train_test_split(dataset,test_size=0.2, random_state=42,shu
print(len(trainingSet), len(testSet))
```

```
4725066 1181267
```

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. Number of Entries: " + str(len(trainingSet)))
numVerRatings = [d['verified_purchase'] for d in trainingSet].count(True)
print("2. Verified purchases: " +"%{0:.2f}".format(100*(numVerRatings / (len(tra
num5Ratings = [d['star_rating'] for d in trainingSet].count(5)
print ("5. Number of 5-Star Ratings: " +"%{0:.2f}".format(100*(num5Ratings / (le
```

```
1. Number of Entries: 4725066
2. Verified purchases: %90.86
5. Number of 5-Star Ratings: %56.04
```

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 [6]:
```
#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 [7]:
```
trainingSet[0:2]
```

Out[7]:
```
[{'marketplace': 'US',
  'customer_id': '32158956',
  'review_id': 'R1KKOXHNI8MSXU',
  'product_id': 'B01KL6O72Y',
  'product_parent': '24485154',
  'product_title': 'Easy Tool Stainless Steel Fruit Pineapple Corer Slicer Peel
er Cut (One size, sliver)',
  'product_category': 'Apparel',
  'star_rating': 4,
  'helpful_votes': 0,
  'total_votes': 0,
  'vine': 'N',
  'verified_purchase': True,
  'review_headline': '★ THESE REALLY DO WORK GREAT WITH SOME TWEAKING ★',
  'review_body': "These Really Do Work Great, But You Do Need To Know a Few Thi
ngs.  I've Been Using Mine For a Few Years Now.  First, I Paid a Few Dollars Le
ss For Mine and The Price Has Jumped.  They're All Imported, so Try to Find Eit
her a Cheaper One or One That's Extremely Well Made.  This One is Made Well Eno
ugh, If You're Careful.  This Thing Can Cut You, So Don't Let Some Kid Use It,
&#34;Because It Looks Like Fun!!&#34;  You Need A Pineapple That's Big Enough.
I Can't Tell You How Many Times I Went to Wal-Mart or the Grocery Store and The
ir Pineapples were Just Too Small of a Diameter.  It HAS to Be Big Enough.  I
t's Better To Have Some Waste on The Inside Of The Husk.<br /><br />When I'm Fi
nished Using The Pineapple Corer, Then I Cut Up the Husk Of The Pineapple To Ge
t The Rest Of The Pineapple Cut Up and I Save The Core of the Pineapple To Go I
nto My Iced Tea Pitcher or Drink Pitcher (That's How They Do It In Hawaii)  Whe
n I Was In Hawaii, They Never Threw Away The Core, They Always Used It For Some
thing.  Sometimes, They Even Grilled It, Although I've Never Tried That.  But,
I Know They Did It With Honey and Some Kind of Teriyaki  People Have Also Left
Chunks of Pineapple on the Husk and Grilled it Meat Side Down on the Grill.  Th
en, They Eat The Chunks of Grilled Pineapple and Use The Husks Like a Little Bo
wl..  You Don't Grill The Husk Part.  You Can Also Keep The Husk Whole, Like a
Bowl and Take the Center Core Out, Then Fill It Full Of Fruit Salad or Use it a
s a Drink Container or Dessert Container.<br /><br />Someone Told Me They Like
To Make Pineapple Greek Yogurt and Fill It, Then Freeze It a Short Time.  I've
Never Tried That.  I've Made Pineapple Cheesecake and That's Delicious, but I K
now It's Tricky Working With Pineapple Sometimes.  You Know, If You've Ever Tri
ed to Make Jello & Put Pineapple In.  YIKES!!! The Bromelain in the Pineapple K
eeps Your Gelatin From Setting.  It is a Great Anti-Inflammatory & Helps Your D
igestive Tract & Tummy Feel Better, But That Same Enzyme Interferes with Your J
ello Setting Up.<br /><br />When Washing These, Be Extra Carful and Wash Them S
eparate.  I Use Tooth Picks, to Get the Pineapple Out That Gets Stuck In The Li
ttle Crevices.  I Always Put The Parts Back Into The Box & Make Sure Everything
is Dry.  So, That's It.  Make Sure Your Pineapple is Big Enough.  Get All Your
Pineapple To Work For You.  Clean Your Corer and Protect Your Hands.  That's I
t.  Thank You For Reading.Same Enzyme Interferes with Your Jello Setting Up.<br
/><br />When Washing These, Be Extra Carful and Wash Them Separate.  I Use Toot
h Picks, to Get the Pineapple Out That Gets Stuck In The Little Crevices.  I Al
ways Put The Parts Back Into The Box & Make Sure Everything is Dry.  So, That's
It.  Make Sure Your Pineapple is Big Enough.  Get All Your Pineapple To Work Fo
r You.  Clean Your Corer and Protect Your Hands.  That's It.  Thank You For Rea
ding.",
  'review_date': '2013-01-14'},
 {'marketplace': 'US',
  'customer_id': '2714559',
  'review_id': 'R26SP2OPDK4HT7',
  'product_id': 'B01ID3ZS5W',
  'product_parent': '363128556',
  'product_title': 'V28 Women Cowl Neck Knit Stretchable Elasticity Long Sleeve
Slim Fit Sweater Dress',
  'product_category': 'Apparel',
  'star_rating': 5,
  'helpful_votes': 1,
  'total_votes': 2,
  'vine': 'N',
  'verified_purchase': True,
  'review_headline': 'Favorite for winter. Very warm!',
  'review_body': 'I love this dress. Absolute favorite for winter. Heavy materi
al. Stretchy, shows shape well. I am 5ft 7, 120 lbs. Ordered 2-8. Fits fine. No
t tight at all. But not to loose. Very confortable. I live in ND, and it keeps
you warm during winter. If its not cold. You will get hot. I got it just for wi
nter. So its perfect. About to order anothwr in different color.<br />;0)',
  'review_date': '2014-03-04'}]
```

In [8]:

```
#YOUR CODE HERE
X = [feature(d) for d in trainingSet] #List of our "features"
y = [d['verified_purchase'] for d in trainingSet] # if d['verified_purchase'] ==
#List of all star ratings
modelLin = linear_model.LogisticRegression()
modelLin.fit(X, y)
```

Out[8]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', 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 [9]:

```
#YOUR CODE HERE
predictions = modelLin.predict(X)
correct = predictions == y
correct
#calculate accuracy
accuracy = sum(correct) / len(correct)
print("Accuracy = " + str(accuracy))
```

```
Accuracy = 0.9084480513076431
```

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 [11]:
```
#CHANGE PATH
path = "./amazon_reviews_us_Apparel_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)
```

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 [12]:
```
#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 [14]:
```
#YOUR CODE HERE
for d in sampleSet:
    r = ''.join([c for c in d['review_body'].lower() if not c in punctuation])
    for w in r.split():
        wordCount[w] += 1
print(len(wordCount))


for d in sampleSet:
    rs = ''.join([c for c in d['review_body'].lower() if not c in punctuation])
    for w in rs.split():
        w = stemmer.stem(w) # with stemming
        wordCountStem[w] += 1
print(len(wordCountStem))
```

```
189258
162311
```

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 [16]:
```
#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 [17]:
```
#YOUR CODE HERE
X_reg = [feature_reg(datum) for datum in sampleSet] #List of our "features"
y_reg = [d['star_rating'] for d in sampleSet]
model = linear_model.Ridge(1.0, fit_intercept=True)
model.fit(X_reg, y_reg)
predictions = model.predict(X_reg)
labels = y_reg
print("MSE: " + str(MSE(predictions, labels)))
```

```
MSE: 1.2403558372110477
```

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 [18]:
```
#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 [19]:
```
#YOUR CODE HERE
itemNames = {}
for d in trainingSet:
    user,item = d['customer_id'], d['product_id']
    attribute_1[item].add(user)
    attribute_2[user].add(item)
    itemNames[item] = d['product_title']
```

In [20]:
```
#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 [25]:
```
#YOUR CODE HERE
query = testSet[0]['product_id']
print(query)
mostSimilar(query,10)
```

```
B0043GTHFG
```

Out[25]:
```
[(1.0,
  'BellyLady Belly Dance Tribal Sequined Bra Top, 34A/34B/36A, Christmas Gift I
dea'),
 (1.0, 'B01KL6O72Y'),
 (1.0, 'B01ID3ZS5W'),
 (1.0, 'B01I497BGY'),
 (1.0, 'B01HDXFZK6'),
 (1.0, 'B01G6MBEBY'),
 (1.0, 'B01FWRXN0Y'),
 (1.0, 'B01EXNH1HE'),
 (1.0, 'B01E7OL09O'),
 (1.0, 'B01DXHX81O')]
```

Finished!

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