COURSE4CAPSTONE-YassY

January 27, 2020

1 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.

1.1 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!

1.1.1 As Reviewer:

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

1.1.2 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.

[70]:
```python
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
path = "amazon_reviews_us_Home_Improvement_v1_00.tsv.gz"
```

2 Task 1: Data Processing

2.0.1 TODO 1: Read the data and Fill your dataset

[71]:
```
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'])
    dataset.append(d)
```

2.0.2 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%.

[72]:
```
#YOUR CODE HERE
N = len(dataset)
random.shuffle(dataset)

trainingSet = dataset[:int(N*0.8)]
testSet = dataset[int(N*0.8):]

print("TrainingSet data entries = " + str(len(trainingSet)), " | TestSet data␣
 ↪entries = " + str(len(testSet)))
```

```
TrainingSet data entries = 2107824  | TestSet data entries = 526957
```

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.

[73]:
```
del dataset
```

2.0.3 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?

```
[74]: print("\nSample of the dataset Structure : ")
      trainingSet[0]
```

```
Sample of the dataset Structure :
```

```
[74]: {'marketplace': 'US',
       'customer_id': '8654584',
       'review_id': 'RO6RZ4AAWSOHI',
       'product_id': 'B00SSNPI80',
       'product_parent': '848836235',
       'product_title': 'Hyperikon T8, 4ft LED Light Tube, Dual-End Powered, 18W,
      Clear',
       'product_category': 'Home Improvement',
       'star_rating': 5,
       'helpful_votes': 0,
       'total_votes': 0,
       'vine': 'N',
       'verified_purchase': 'Y',
       'review_headline': 'Five Stars',
       'review_body': 'They are bright work great',
       'review_date': '2015-05-17'}
```

```
[75]: print ("Training dataset contain : " + str(N) + " Entries")
      Ntr = len(trainingSet)
      Vpursh = [d['verified_purchase'] for d in trainingSet if d['verified_purchase']␣
       ↪=='Y']
      Pints = [d['star_rating'] for d in trainingSet if d['star_rating']  >= 4 and␣
       ↪d['verified_purchase'] =='Y']

      pVpursh = (len(Vpursh)/Ntr)*100
      pVpurshPcat= ((len(Pints))/Ntr)*100

      print("Verified Purshases % = " + str('%.0f'% pVpursh + "%"))
      print("Verified Purshades with more than 4 stars rating % = " + str('%.0f'%␣
       ↪pVpurshPcat + "%"))
```

```
Training dataset contain : 2634781 Entries
Verified Purshases % = 90%
Verified Purshades with more than 4 stars rating % = 72%
```

3 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.

3.0.1 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!

[89]:
```
#extracting the necessary feature for the classification process
def feature(d):
    feat = [1, d['helpful_votes'], d['total_votes']]
    return feat
```

3.0.2 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.

[90]:
```
#YOUR CODE HERE
X = [feature(d) for d in trainingSet]
y = [d['verified_purchase'] for d in trainingSet]
model = linear_model.LogisticRegression()
model.fit(X,y)
```

[90]:
```
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)
```

3.0.3 TODO 3: Compute Accuracy of Your Model

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

[97]:
```
#YOUR CODE HERE
predictionTrain=model.predict(X)
correctionpredictionTrain = predictionTrain == y
MAccuracy = (sum(correctionpredictionTrain) /␣
 ↪len(correctionpredictionTrain))*100 ## Train accuracy
print("Model Accuracy = " + str('%.0f'%MAccuracy) + "%")
```

```
Model Accuracy = 90%
```

4 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.

[107]:
```
#CHANGE PATH
path = "amazon_reviews_us_Home_Improvement_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)
```

4.0.1 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**.

[108]:
```
#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]
```

[100]:
```
reg_dataset[0]
```

[100]:
```
{'marketplace': 'US',
 'customer_id': '48881148',
 'review_id': 'R215C9BDXTDQOW',
 'product_id': 'B00FR4YQYK',
 'product_parent': '381800308',
 'product_title': 'SadoTech Model C Wireless Doorbell Operating at over 500-feet
Range with Over 50 Chimes, No Batteries Required for Receiver, (Various
Colors)',
 'product_category': 'Home Improvement',
 'star_rating': 4,
 'helpful_votes': '0',
 'total_votes': '0',
 'vine': 'N',
 'verified_purchase': 'Y',
 'review_headline': 'Four Stars',
 'review_body': 'good product',
 'review_date': '2015-08-31'}
```

[109]:
```
#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.
for d in reg_dataset:
    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))
```

```
605966
```

[111]:
```
#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.
for d in reg_dataset:
    r=''.join([c for c in d['review_body'].lower() if not c in punctuation])
    for w in r.split():
        w=stemmer.stem(w)
        wordCountStem[w]+=1
print(len(wordCountStem))
```

```
532509
```

4.0.2 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.

```
[114]: #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)
```

```
[117]: #YOUR CODE HERE
X_reg = [feature_reg(d) for d in sampleSet]
model = linear_model.Ridge(1.0, fit_intercept=False)
model.fit(X_reg,y_reg)

predictions = model.predict(X_reg)
differences = [(x-y)**2 for (x,y) in zip(predictions, y_reg)]

MSE = sum(differences) / len(differences)
print("MSE = " + '%.3f' %MSE)
```

```
MSE = 9.235
```

5 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**.

[129]:
```
#GIVEN
attribute_1 = defaultdict(set)
attribute_2 = defaultdict(set)
```

5.0.1 TODO 1: Fill your Dictionaries

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

[130]:
```
#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']
```

[131]:
```
#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]
```

5.0.2 TODO 1: Fill your Dictionaries

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

[132]:
```
trainingSet[0]
```

[132]:
```
{'marketplace': 'US',
 'customer_id': '8654584',
 'review_id': 'RO6RZ4AAWSOHI',
 'product_id': 'B00SSNPI80',
 'product_parent': '848836235',
 'product_title': 'Hyperikon T8, 4ft LED Light Tube, Dual-End Powered, 18W,
Clear',
 'product_category': 'Home Improvement',
 'star_rating': 5,
 'helpful_votes': 0,
 'total_votes': 0,
 'vine': 'N',
 'verified_purchase': 'Y',
 'review_headline': 'Five Stars',
 'review_body': 'They are bright work great',
 'review_date': '2015-05-17'}
```

[133]:
```
query = trainingSet[10]['product_id']
query
```

[133]:
```
'B002ZRPMLS'
```

[134]:
```
itemNames[query]
```

[134]:
```
'Malibu 8304-9105-01 11 Watt Pro Light, Aged Verde'
```

[137]:
```
print("10 Most similar entries by ProductID : \n")
mostSimilar(query, 10)
```

```
10 Most similar entries by ProductID :
```

[137]:
```
[(1.0, 'B01MTQH2NF'),
 (1.0, 'B01615SUBI'),
 (1.0, 'B015X6T8R6'),
 (1.0, 'B014FJ7W7K'),
 (1.0, 'B014BYP7TO'),
 (1.0, 'B014B78KPY'),
 (1.0, 'B0149K8OF4'),
 (1.0, 'B0149IF0CG'),
 (1.0, 'B0149A0N8U'),
 (1.0, 'B0147BE274')]
```

[138]:
```
print("10 Most similar entries by ProductName : \n")
[itemNames[x[1]] for x in mostSimilar(query, 10)]
```

```
10 Most similar entries by ProductName :
```

[138]:
```
['Samsung DA29-00020B Aqua-Pure Plus Refrigerator Water Filter',
 'Fibaro Z-Wave Motion Sensor - FGMS-001',
 'SHARP OEM PCOVPB073MRP0 WAVEGUIDE COVER by Sharp',
 'PowerHalo Sliding Gate Opener Sliding Gate Opener Kit Sliding Gate Opener
Remote Auto Close Particularly Simple Installation with Comprehensive
Interface',
 'Antique Gold Swing Arm Floor Lamp 58"',
 'REEGE Premium Multifunction Toilet Handheld Bidet Shattaf Cloth Diaper Sprayer
Shower with Brass Material',
 'Joy Bidet C-1. Cold Water. Non-Electric. Toilet Attachment. (Metal Hose)',
 'Double Cylinder Satin Nickel Finish Deadbolt Lock w/ Keys - Fits All Doors',
 'Alarm Detects One Drop of Water! Leak Detector for your basement. The only
water sensor equipment on the market that detects a single drop of water and
small amounts of moisture.',
 'Classic with Nylon FBA(Ready to Ships)']
```

5.1 Finished!

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