COURSE4-Submission

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.

[145]:
```
import gzip
import json
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
import pandas as pd
from sklearn.linear_model import Ridge
```

2 Task 1: Data Processing

2.0.1 TODO 1: Read the data and Fill your dataset

[34]:
```
#YOUR CODE HERE
file = open('beer_profile_and_ratings.csv', encoding='utf8')
dataset = []
header = file.readline().strip().split(',')
for line in file:
    line = line.split(',')
    dataset.append(line)

header
```

[34]:
```
['\ufeffName',
 'Style',
 'Brewery',
 'Beer Name (Full)',
 'Description',
 'ABV',
 'Min IBU',
 'Max IBU',
 'Astringency',
 'Body',
 'Alcohol',
 'Bitter',
 'Sweet',
 'Sour',
 'Salty',
 'Fruits',
 'Hoppy',
 'Spices',
 'Malty',
 'review_aroma',
 'review_appearance',
 'review_palate',
 'review_taste',
 'review_overall',
 'number_of_reviews']
```

[37]:
```
dataset[0]
```

[37]:
```
['Amber',
 'Altbier',
 'Alaskan Brewing Co.',
 'Alaskan Brewing Co. Alaskan Amber',
 '"Notes:Richly malty and long on the palate',
 ' with just enough hop backing to make this beautiful amber colored ""alt""
```

```
style beer notably well balanced.\\t"',
 '5.3',
 '25',
 '50',
 '13',
 '32',
 '9',
 '47',
 '74',
 '33',
 '0',
 '33',
 '57',
 '8',
 '111',
 '3.498994',
 '3.636821',
 '3.556338',
 '3.643863',
 '3.847082',
 '497\n']
```

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%.

[40]:
```
#YOUR CODE HERE
random.shuffle(dataset)
```

[49]:
```
N = len(dataset)
Train = dataset[:int(N*0.8)]
Test = dataset[int(N*0.8):]
len(Train), len(Test)
```

[49]:
```
(2557, 640)
```

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.

[105]:
```
#YOUR CODE HERE
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?

[82]:
```
d = pd.read_csv('beer_profile_and_ratings.csv')
d.head()
```

[82]:
```
                             Name    Style  \
0                           Amber  Altbier
1                      Double Bag  Altbier
2                  Long Trail Ale  Altbier
3                    Doppelsticke  Altbier
4  Sleigh'r Dark Doüble Alt Ale  Altbier

                                             Brewery  \
0                                Alaskan Brewing Co.
1                             Long Trail Brewing Co.
2                             Long Trail Brewing Co.
3  Uerige Obergärige Hausbrauerei GmbH / Zum Uerige
4                            Ninkasi Brewing Company

                                    Beer Name (Full)  \
0                    Alaskan Brewing Co. Alaskan Amber
1                   Long Trail Brewing Co. Double Bag
2               Long Trail Brewing Co. Long Trail Ale
3  Uerige Obergärige Hausbrauerei GmbH / Zum Ueri...
4  Ninkasi Brewing Company Sleigh'r Dark Doüble A...

                                         Description  ABV  Min IBU  Max IBU  \
0  Notes:Richly malty and long on the palate, wit...  5.3       25       50
1  Notes:This malty, full-bodied double alt is al...  7.2       25       50
2  Notes:Long Trail Ale is a full-bodied amber al...  5.0       25       50
3                                             Notes:  8.5       25       50
4  Notes:Called 'Dark Double Alt' on the label.Se...  7.2       25       50

   Astringency  Body  ...  Fruits  Hoppy  Spices  Malty  review_aroma  \
0           13    32  ...      33     57       8    111      3.498994
1           12    57  ...      24     35      12     84      3.798337
2           14    37  ...      10     54       4     62      3.409814
3           13    55  ...      49     40      16    119      4.148098
4           25    51  ...      11     51      20     95      3.625000

   review_appearance  review_palate  review_taste  review_overall  \
0           3.636821       3.556338      3.643863        3.847082
1           3.846154       3.904366      4.024948        4.034304
2           3.667109       3.600796      3.631300        3.830239
3           4.033967       4.150815      4.205163        4.005435
```

```
4          3.973958       3.734375      3.765625        3.817708

   number_of_reviews
0                497
1                481
2                377
3                368
4                 96

[5 rows x 25 columns]
```

[83]:
```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3197 entries, 0 to 3196
Data columns (total 25 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                3197 non-null   object
 1   Style               3197 non-null   object
 2   Brewery             3197 non-null   object
 3   Beer Name (Full)    3197 non-null   object
 4   Description         3197 non-null   object
 5   ABV                 3197 non-null   float64
 6   Min IBU             3197 non-null   int64
 7   Max IBU             3197 non-null   int64
 8   Astringency         3197 non-null   int64
 9   Body                3197 non-null   int64
 10  Alcohol             3197 non-null   int64
 11  Bitter              3197 non-null   int64
 12  Sweet               3197 non-null   int64
 13  Sour                3197 non-null   int64
 14  Salty               3197 non-null   int64
 15  Fruits              3197 non-null   int64
 16  Hoppy               3197 non-null   int64
 17  Spices              3197 non-null   int64
 18  Malty               3197 non-null   int64
 19  review_aroma        3197 non-null   float64
 20  review_appearance   3197 non-null   float64
 21  review_palate       3197 non-null   float64
 22  review_taste        3197 non-null   float64
 23  review_overall      3197 non-null   float64
 24  number_of_reviews   3197 non-null   int64
dtypes: float64(6), int64(14), object(5)
memory usage: 624.5+ KB
```

[81]:
```
d.describe()
```

[81]:

```
               ABV      Min IBU      Max IBU  Astringency         Body  \
count  3197.000000  3197.000000  3197.000000  3197.000000  3197.000000
mean      6.526688    21.180482    38.986863    16.515796    46.129496
std       2.546997    13.242242    21.355281    10.410661    25.947842
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       5.000000    15.000000    25.000000     9.000000    29.000000
50%       6.000000    20.000000    35.000000    14.000000    40.000000
75%       7.600000    25.000000    45.000000    21.000000    58.000000
max      57.500000    65.000000   100.000000    81.000000   175.000000

           Alcohol       Bitter        Sweet         Sour        Salty  \
count  3197.000000  3197.000000  3197.000000  3197.000000  3197.000000
mean     17.055990    36.364404    58.270879    33.145449     1.017204
std      17.331334    25.791152    34.281310    35.780172     2.132651
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       6.000000    17.000000    33.000000    11.000000     0.000000
50%      11.000000    31.000000    54.000000    22.000000     0.000000
75%      22.000000    52.000000    77.000000    42.000000     1.000000
max     139.000000   150.000000   263.000000   284.000000    48.000000

            Fruits        Hoppy       Spices        Malty  review_aroma  \
count  3197.000000  3197.000000  3197.000000  3197.000000   3197.000000
mean     38.529559    40.924617    18.345637    75.330935      3.638789
std      32.296646    30.403641    23.756582    39.909338      0.503209
min       0.000000     0.000000     0.000000     0.000000      1.509615
25%      12.000000    18.000000     4.000000    45.000000      3.422559
50%      29.000000    33.000000    10.000000    73.000000      3.720183
75%      60.000000    56.000000    23.000000   103.000000      3.978000
max     175.000000   172.000000   184.000000   239.000000      5.000000

       review_appearance  review_palate  review_taste  review_overall  \
count        3197.000000    3197.000000   3197.000000     3197.000000
mean            3.754393       3.660428      3.702496        3.747522
std             0.403416       0.449937      0.510361        0.444288
min             1.571429       1.285714      1.214286        1.136364
25%             3.604651       3.470021      3.500000        3.566667
50%             3.833333       3.741667      3.791667        3.830239
75%             4.000000       3.965587      4.033333        4.032847
max             4.666667       5.000000      5.000000        5.000000

       number_of_reviews
count        3197.000000
mean          233.284955
std           361.811847
min             1.000000
25%            23.000000
50%            93.000000
```

```
75%           284.000000
max          3290.000000
```

[107]:
```
d['review_overall'] = d['review_overall'].astype('int')
```

[108]:
```
new_data = d[['Body', 'Alcohol', 'review_overall']].values.tolist()
random.shuffle(new_data)
N = len(new_data)
Train = new_data[:int(N*0.8)]
Test = new_data[int(N*0.8):]
len(Train), len(Test)
```

[108]:
```
(2557, 640)
```

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!

[109]:
```
#FIX THIS

def feature(d):
    feat = [1, d[0],(d[1])]
    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.

[110]:
```
#YOUR CODE HERE
X_train = [feature(d) for d in Train]
y_train = [d[-1] for d in Train]
X_test = [feature(d) for d in Test]
y_test = [d[-1] for d in Test]

# Look at first 10 rows of X and y
print("Label: ", y_train[:10], "\nFeatures:", X_train[:10])
```

```
Label:  [3, 3, 3, 3, 3, 3, 3, 3, 3, 4]
Features: [[1, 39, 4], [1, 17, 11], [1, 36, 9], [1, 41, 7], [1, 30, 3], [1, 21,
4], [1, 64, 36], [1, 34, 33], [1, 43, 39], [1, 4, 0]]
```

3.0.3 TODO 3: Compute Accuracy of Your Model

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

[112]:
```
#YOUR CODE HERE
model = linear_model.LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)
predictionsTrain = model.predict(X_train)
predictionsTest = model.predict(X_test)
correctPredictionsTrain = predictionsTrain == y_train
correctPredictionsTest = predictionsTest == y_test
```

[113]:
```
sum(correctPredictionsTrain) / len(correctPredictionsTrain)
```

[113]:
```
0.6390301134141572
```

[114]:
```
sum(correctPredictionsTest) / len(correctPredictionsTest)
```

[114]:
```
0.646875
```

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.

[115]:
```
#CHANGE PATH
path = pd.read_csv('beer_profile_and_ratings.csv')
#GIVEN
header = path.columns
f = path.values.tolist()
reg_dataset = []
for line in f:
    d = dict(zip(header, line))
    d['review_overall'] = int(d['review_overall'])
    reg_dataset.append(d)
```

[182]:
```
reg_dataset[0]
```

[182]:
```
{'Name': 'Amber',
 'Style': 'Altbier',
 'Brewery': 'Alaskan Brewing Co.',
 'Beer Name (Full)': 'Alaskan Brewing Co. Alaskan Amber',
 'Description': 'Notes:Richly malty and long on the palate, with just enough hop
```

```
backing to make this beautiful amber colored "alt" style beer notably well
balanced.\\t',
 'ABV': 5.3,
 'Min IBU': 25,
 'Max IBU': 50,
 'Astringency': 13,
 'Body': 32,
 'Alcohol': 9,
 'Bitter': 47,
 'Sweet': 74,
 'Sour': 33,
 'Salty': 0,
 'Fruits': 33,
 'Hoppy': 57,
 'Spices': 8,
 'Malty': 111,
 'review_aroma': 3.498994,
 'review_appearance': 3.636821,
 'review_palate': 3.556338,
 'review_taste': 3.643863,
 'review_overall': 3,
 'number_of_reviews': 497}
```

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**.

[153]:
```
#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['review_overall'] for d in sampleSet]
```

[154]:
```
#YOUR CODE HERE
for d in sampleSet:
    r = ''.join([c for c in d['Description'].lower() if not c in punctuation])
    for w in r.split():
        wordCount[w] += 1
print('unique words:', len(wordCount))
```

```
unique words: 4333
```

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.

[141]:
```
#GIVEN FUNCTIONS
def feature_reg(datum):
    feat = [0]*len(words)
    r = ''.join([c for c in datum['Description'] 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)
```

[143]:
```
#YOUR CODE HERE
X = [feature_reg(d) for d in reg_dataset] #List of our "features"
y = [d['review_overall'] for d in reg_dataset] #List of all star ratings
y_class = [(rating >= 3) for rating in y] #List of all ratings higher than 3␣
 ↪stars

modelLin = linear_model.LogisticRegression() #Basic Linear Model
```

```
modelLin.fit(X, y_class)
```

[143]: LogisticRegression()

[155]:
```
X_reg = [feature_reg(d) for d in sampleSet]
# Fit a Ridge Model
model = Ridge(1.0, fit_intercept=True)
model.fit(X_reg, y_reg)
# Make your predictions
predictions = model.predict(X_reg)
# MSE
mse = MSE(predictions, y_reg)
print('MSE is {:.2f}'.format(mse))
```

```
MSE is 0.19
```

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**.

[149]:
```
#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).

[168]:
```
#YOUR CODE HERE
for d in Train:
    user,item = d[0], d[1]
    attribute_1[user].add(item)
    attribute_2[item].add(user)


#Getting the respective lengths of our dataset and dictionaries
N = len(Train)
nAt1 = len(attribute_1)
nAt2 = len(attribute_2)
```

[160]:
```
#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.

[172]:
```
#YOUR CODE HERE
query = Train[0][0]
mostSimilar(query,10)
```

[172]:
```
[(1.0, 175),
 (1.0, 166),
 (1.0, 158),
 (1.0, 156),
 (1.0, 155),
 (1.0, 154),
 (1.0, 153),
 (1.0, 147),
 (1.0, 141),
 (1.0, 139)]
```

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!