import numpy

import urllib

import scipy.optimize

import random

from sklearn import svm

import os

import yaml

import random

import operator

def parseData(fname):

for l in urllib.request.urlopen(fname):

yield eval(l)

data = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer\_50000.json"))

#Question 1

print('QUESTION 1')

ratings = {'1-star':0, '2-star':0, '3-star':0, '4-star':0, '5-star':0}

for d in data:

ratings[str(int(d['review/taste']))+'-star'] += 1

print('Rounded down ratings: {}'.format(ratings))

#Question 2

print('QUESTION 2')

listOfBeers = set()

ratings = dict()

averageRatings = dict()

for d in data:

listOfBeers.add(d['beer/name'])

for beerName in listOfBeers:

ratings[beerName] = 0

averageRatings[beerName] = 0

for d in data:

ratings[d['beer/name']] += 1

averageRatings[d['beer/name']] += d['review/taste']

for key in ratings.keys():

averageRatings[key] = -1 if ratings[key] < 5 else averageRatings[key]/ratings[key]

maximum = max(averageRatings.values())

result = []

for item in averageRatings.items():

if item[1] == maximum:

result.append(item)

print(result)

#Question 3

print('QUESTION 3')

data2 = [d for d in data if 'beer/ABV' in d.keys() and 'beer/style' in d.keys()]

def feature(datum):

feat = [1]

if datum['beer/style'] == 'Hefeweizen':

feat.append(1)

else:

feat.append(0)

feat.append(datum['beer/ABV'])

return feat

X = [feature(d) for d in data2]

y = [d['review/taste'] for d in data2]

theta,residuals,rank,s = numpy.linalg.lstsq(X, y, rcond=None)

print(theta)

#theta[0] shows us the offset. Theta[1] is the weighting of the beer being a

#Hefeweizen on the taste score, in this case it has negative impact on the score.

#Theta[2] shows the impact of the beer alcohol content on the taste score, in this case

#higher is better

#Question 4

print('QUESTION 4')

train = data2[:len(data2)//2]

X\_train = [feature(d) for d in train]

y\_train = [d['review/taste'] for d in train]

theta,residuals,rank,s = numpy.linalg.lstsq(X\_train, y\_train, rcond=-1)

predictions = numpy.matmul(theta, numpy.transpose(X\_train))

print('Error for train data =', numpy.sum(numpy.square(numpy.subtract(predictions, y\_train))))

test = data2[len(data2)//2:]

X\_test = [feature(d) for d in test]

y\_test = [d['review/taste'] for d in test]

predictions = numpy.matmul(theta, numpy.transpose(X\_test))

print('Error for test data =', numpy.sum(numpy.square(numpy.subtract(predictions, y\_test))))

#Question 5

print('QUESTION 5')

dataCopy = data2.copy()

for i in range(10):

random.shuffle(dataCopy)

train = dataCopy[:len(dataCopy)//2]

X\_train = [feature(d) for d in train]

y\_train = [d['review/taste'] for d in train]

theta,residuals,rank,s = numpy.linalg.lstsq(X\_train, y\_train, rcond=-1)

predictions = numpy.matmul(theta, numpy.transpose(X\_train))

print('Error for train data =', numpy.sum(numpy.square(numpy.subtract(predictions, y\_train))))

test = dataCopy[len(dataCopy)//2:]

X\_test = [feature(d) for d in test]

y\_test = [d['review/taste'] for d in test]

predictions = numpy.matmul(theta, numpy.transpose(X\_test))

print('Error for test data =', numpy.sum(numpy.square(numpy.subtract(predictions, y\_test))))

#The data that is pulled from the website might be in some certain order and dividing it before shuffling

#might result in skewed data

#Question 6

print('QUESTION 6')

def feature(datum):

#feat = [1]

feat = []

feat.append(datum['review/taste'])

feat.append(datum['review/appearance'])

feat.append(datum['review/aroma'])

feat.append(datum['review/palate'])

feat.append(datum['review/overall'])

return feat

dataCopy = data2.copy()

for i in range(10):

random.shuffle(dataCopy)

train = dataCopy[:len(dataCopy)//2]

X\_train = [feature(d) for d in train]

y\_train = [1 if d['beer/style'] == 'Hefeweizen' else 0 for d in train]

clf = svm.SVC(C=1000, kernel='rbf', gamma='scale', degree=2)

clf.fit(X\_train, y\_train)

train\_predictions = clf.predict(X\_train)

print('Accuracy for train data =', 100-(100\*numpy.sum(numpy.square(numpy.subtract(train\_predictions, y\_train)))/len(y\_train)), '%')

test = dataCopy[len(dataCopy)//2:]

X\_test = [feature(d) for d in test]

y\_test = [1 if d['beer/style'] == 'Hefeweizen' else 0 for d in test]

test\_predictions = clf.predict(X\_test)

print('Accuracy for test data =', 100-(100\*numpy.sum(numpy.square(numpy.subtract(test\_predictions, y\_test)))/len(y\_test)), '%')

#Question 7

print('QUESTION 7')

def feature(datum):

feat = [1]

feat.append(datum['beer/ABV'])

feat.append(1 if 'malt' in datum['review/text'] or 'malty' in datum['review/text'] else 0)

feat.append(datum['review/timeUnix'])

return feat

train = dataCopy[:len(dataCopy)//2]

X\_train = [feature(d) for d in train]

y\_train = [1 if d['beer/style'] == 'Hefeweizen' else 0 for d in train]

clf2 = svm.SVC(C=1000, kernel='rbf', gamma='scale')

clf2.fit(X\_train, y\_train)

train\_predictions = clf2.predict(X\_train)

print('Accuracy for train data =', 100-(100\*numpy.sum(numpy.square(numpy.subtract(train\_predictions, y\_train)))/len(y\_train)), '%')

test = dataCopy[len(dataCopy)//2:]

X\_test = [feature(d) for d in test]

y\_test = [1 if d['beer/style'] == 'Hefeweizen' else 0 for d in test]

test\_predictions = clf2.predict(X\_test)

print('Accuracy for test data =', 100-(100\*numpy.sum(numpy.square(numpy.subtract(test\_predictions, y\_test)))/len(y\_test)), '%')

#The features used are:

#[1, beer/ABV, Boolean indicating if review text contains the word 'malt' or 'malty', review/timeUnix]

#Question 8

print('QUESTION 8')

c = [0.1, 10, 1000, 100000]

for reg in c:

clf = svm.SVC(C=reg, kernel='rbf', gamma='scale')

clf.fit(X\_train, y\_train)

train\_predictions = clf.predict(X\_train)

print('--------For regularization factor =', reg, '-----------')

print('Accuracy for train data =', 100-(100\*numpy.sum(numpy.square(numpy.subtract(train\_predictions, y\_train)))/len(y\_train)), '%')

test\_predictions = clf.predict(X\_test)

print('Accuracy for test data =', 100-(100\*numpy.sum(numpy.square(numpy.subtract(test\_predictions, y\_test)))/len(y\_test)), '%')

Printed Results

QUESTION 1

Rounded down ratings: {'1-star': 554, '2-star': 2723, '3-star': 12934, '4-star': 29458, '5-star': 4331}

QUESTION 2

[('Founders CBS Imperial Stout', 4.6970172684458396)]

QUESTION 3

[ 3.11795084 -0.05637406 0.10877902]

QUESTION 4

Error for train data = 12099.201400335607

Error for test data = 10592.663029965459

QUESTION 5

Error for train data = 11435.60803063151

Error for test data = 11053.513210952766

QUESTION 6

Accuracy for train data = 98.776 %

Accuracy for test data = 98.712 %

QUESTION 7

Accuracy for train data = 99.172 %

Accuracy for test data = 97.968 %

QUESTION 8

--------For regularization factor = 0.1 -----------

Accuracy for train data = 98.748 %

Accuracy for test data = 98.78 %

--------For regularization factor = 10 -----------

Accuracy for train data = 98.872 %

Accuracy for test data = 98.56 %

--------For regularization factor = 1000 -----------

Accuracy for train data = 99.172 %

Accuracy for test data = 97.968 %

--------For regularization factor = 100000 -----------

Accuracy for train data = 99.592 %

Accuracy for test data = 97.508 %