In [11]:
```
import numpy
import urllib.request
import scipy.optimize
import random
from statistics import mean
from sklearn import svm

def parseData(fname):
    for l in urllib.request.urlopen(fname):
        yield eval(l)


print("Reading data...")
data = list(parseData("http://jmcauley.ucsd.edu/cse258/data/beer/beer_50
000.json"))
print("done")

# Q1:
print("1 star:", len([d for d in data if d["review/taste"] == 1]))
print("1.5 star:", len([d for d in data if d["review/taste"] == 1.5]))
print("2 star:", len([d for d in data if d["review/taste"] == 2]))
print("2.5 star:", len([d for d in data if d["review/taste"] == 2.5]))
print("3 star:", len([d for d in data if d["review/taste"] == 3]))
print("3.5 star:", len([d for d in data if d["review/taste"] == 3.5]))
print("4 star:", len([d for d in data if d["review/taste"] == 4]))
print("4.5 star:", len([d for d in data if d["review/taste"] == 4.5]))
print("5 star:", len([d for d in data if d["review/taste"] == 5]))

# Answer: 1 star: 211       reviews
#         1.5 star: 343     reviews
#         2 star: 1099      reviews
#         2.5 star: 1624    reviews
#         3 star: 4137      reviews
#         3.5 star: 8797    reviews
#         4 star: 16575     reviews
#         4.5 star: 12883   reviews
#         5 star: 4331      reviews


# Q2:
count = {}
rating = {}
for d in data:
    count[d["beer/name"]] = count.get(d["beer/name"], 0) + 1

for d in data:
    if count[d["beer/name"]] >= 5:
        rating[d["beer/name"]] = rating.get(d["beer/name"], 0) + d["revi
ew/taste"]

for key in rating:
    rating[key] = rating[key] / count[key]

print([key for key in rating if rating[key] == max(rating.values())])

# Answer: 'Founders CBS Imperial Stout' has the highest average rating a
```

```
mong beers with ≥ 5 reviews


# Q3:
data2 = [d for d in data]
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)
print("theta0: %s theta1: %s theta2: %s" %(theta[0],theta[1],theta[2]))

#Answer: theta0: 3.117950842474128 theta1: -0.056374057703859136 theta2:
 0.10877901639207486
#        theta0 represents the predicted review taste for beer which is
 not Hefeweizen and the ABV = 0 is about 3.12
#        theta1 represents the effect that whether the beer is Hefeweize
n has on the review taste. Keeping ABV the same, if the beer is Hefeweiz
en, then the review taste is predicted to decrease by 0.05.
#        theta2 represents the effect tha beer ABV has on the review tas
te. Keeping whether the beer is Hefeweizenthe the same, if the beer ABV
 increase by one unit, then the review taste is predicted to increase by
 0.1.


#Q4:
data_train = [data[i] for i in range(25000)]
data_test = [data[i] for i in range(25000,50000)]

X_train = [feature(d) for d in data_train]
Y_train = [d["review/taste"] for d in data_train]
theta,residuals,rank,s = numpy.linalg.lstsq(X_train,Y_train)
mse_train = numpy.mean((Y_train-numpy.dot(X_train,theta))**2)
print(mse_train)

X_test = [feature(d) for d in data_test]
Y_test = [d["review/taste"] for d in data_test]
mse_test = numpy.mean((Y_test-numpy.dot(X_test,theta))**2)
print(mse_test)

#Answer: MSE on the training set is 0.48396805601342435
#        MSE on the testing set is  0.4237065211986184


#Q5:
data1 = [d for d in data]
random.shuffle(data1)
data_train = data1[:25000]
data_test = data1[25000:]

X_train = [feature(d) for d in data_train]
```

```
Y_train = [d["review/taste"] for d in data_train]
theta,residuals,rank,s = numpy.linalg.lstsq(X_train,Y_train)
mse_train = numpy.mean((Y_train-numpy.dot(X_train,theta))**2)
print(mse_train)

X_test = [feature(d) for d in data_test]
Y_test = [d["review/taste"] for d in data_test]
mse_test = numpy.mean((Y_test-numpy.dot(X_test,theta))**2)
print(mse_test)

#Answer: MSE on the training set is 0.44778450800373054
#        MSE on the testing set is  0.45158340650228146
#        The reason may be that the previous data set is in some kind of
 order so that it is not very random.
#        But this experiment randomly splits the data set so that it is
 more random and then get different results.


#Q6:
data1 = [d for d in data]
random.shuffle(data1)
data_train = data1[:25000]
data_test = data1[25000:]

def feature(datum):
        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

X_train = [feature(d) for d in data_train]
Y_train = [d["beer/style"] == 'Hefeweizen' for d in data_train]
clf = svm.SVC(C=1000, kernel='linear')
clf.fit(X_train, Y_train)
train_predictions = clf.predict(X_train)
train_correct = (train_predictions == Y_train)
train_acc = sum(train_correct)/len(train_correct)
print("accuracy on train data",train_acc *100 ,"%")

X_test = [feature(d) for d in data_test]
Y_test = [d["beer/style"] == 'Hefeweizen' for d in data_test]
test_predictions = clf.predict(X_test)
test_correct = (test_predictions == Y_test)
test_acc = sum(test_correct)/len(test_correct)
print("accuracy on test data",test_acc *100 ,"%")

#Answer: The accuracy on train data is 98.756 %
#        The accuracy on test data is 98.772 %


#Q7:
data1 = [d for d in data]
random.shuffle(data1)
data_train = data1[:25000]
```

```
data_test = data1[25000:]

def feature(datum):
        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"])
        feat.append(datum["beer/beerId"])
        feat.append(datum["beer/brewerId"])
        if "Hefeweizen" in datum["review/text"]:
                feat.append(1)
        else: feat.append(0)
        return feat

X_train = [feature(d) for d in data_train]
Y_train = [d["beer/style"] == 'Hefeweizen' for d in data_train]
clf = svm.SVC(C=1000, kernel='rbf')
clf.fit(X_train, Y_train)
train_predictions = clf.predict(X_train)
train_correct = (train_predictions == Y_train)
train_acc = sum(train_correct)/len(train_correct)
print("accuracy on train data",train_acc *100 ,"%")

X_test = [feature(d) for d in data_test]
Y_test = [d["beer/style"] == 'Hefeweizen' for d in data_test]
test_predictions = clf.predict(X_test)
test_correct = (test_predictions == Y_test)
test_acc = sum(test_correct)/len(test_correct)
print("accuracy on test data",test_acc *100 ,"%")

#Answer: feature vector: [‘review/taste’, ‘review/appearance’, ‘review/a
roma’, ‘review/palate’, ‘review/overall’, 'beer/beerId','beer/brewerI
d',"review/text"].
#        The accuracy on train data is 100.0%
#        The accuracy on test data is 99.908%

#Q8:
data1 = [d for d in data]
random.shuffle(data1)
data_train = data1[:25000]
data_test = data1[25000:]

def feature(datum):
        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"])
        feat.append(datum["beer/beerId"])
        feat.append(datum["beer/brewerId"])
        if "Hefeweizen" in datum["review/text"]:
                feat.append(1)
        else: feat.append(0)
        return feat
```

```
X_train = [feature(d) for d in data_train]
Y_train = [d["beer/style"] == 'Hefeweizen' for d in data_train]
X_test = [feature(d) for d in data_test]
Y_test = [d["beer/style"] == 'Hefeweizen' for d in data_test]

for C in [0.1, 10, 1000, 100000]:
        clf = svm.SVC(C, kernel='rbf')
        clf.fit(X_train, Y_train)
        train_predictions = clf.predict(X_train)
        train_correct = (train_predictions == Y_train)
        train_acc = sum(train_correct)/len(train_correct)
        print("C=",C,"accuracy on train data",train_acc *100 ,"%")

        test_predictions = clf.predict(X_test)
        test_correct = (test_predictions == Y_test)
        test_acc = sum(test_correct)/len(test_correct)
        print("C=",C,"accuracy on test data",test_acc *100 ,"%")

#Answer: C= 0.1 The accuracy on train data is 99.47200000000001%
#        C= 0.1 The accuracy on test data 99.496%
#        C= 10 The accuracy on test data is 99.996%
#        C= 10 The accuracy on test data is 99.9%
#        C= 1000 The accuracy on train data 100.0%
#        C= 1000 The accuracy on test data 99.9%
#        C= 100000 accuracy on train data 100.0%
#        C= 100000 accuracy on test data 99.9%
#        In my experiment, as C gets larger, the accuracy gets a little
 bit higher but ther is no obvious change
#        because the features are few and the variablity is small.
#        In theory, As C gets larger, the training/test performance beco
mes better, the accuracy will be higher.
```