A53271697
Bosi Cheng

1. from 1.0 to 5.0: (judged by if, elif, else)
   [211, 343, 1099, 1624, 4137, 8797, 16575, 12883, 4331]

2.

```
import numpy
import urllib
import scipy.optimize
import random
from sklearn import linear_model

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

print "Reading data..."
data = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json"))
print "done"

x=[[1 if d['beer/style']=='Hefeweizen' else 0 for d in data],[d['beer/ABV'] for d in data]]
xx=[]

for i in range(len(data)):
  xx.append([x[0][i],x[1][i]])

y=[d['review/taste'] for d in data]
clf = linear_model.LinearRegression()

clf.fit(xx,y)
print(clf.coef_,clf.intercept_)
```

linear regression coefficient(θ_1 and θ_2) and its intercept (θ_0):
(array([-0.05637406, 0.10877902]), 3.1179508424739546)

3.

linear regression coefficient and its intercept:
(array([-0.03573098, 0.11672256]), 2.996914661970776)

week1.py | Extension: Python | beer_50000.json

```python
import numpy
import urllib
import scipy.optimize
import random
from sklearn import linear_model

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

print "Reading data..."
data = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json"))
print "done"

x=[[1 if d['beer/style']=='Hefeweizen' else 0 for d in data],[d['beer/ABV'] for d in data]]
xx=[]

for i in range(len(data)):
  xx.append([x[0][i],x[1][i]])

y=[d['review/taste'] for d in data]
clf = linear_model.LinearRegression()

halflen=len(data)/2
xxtrain=xx[:halflen]
xxtest=xx[halflen:]
ytrain=y[:halflen]
ytest=y[halflen:]

clf.fit(xxtrain,ytrain)
print(clf.coef_,clf.intercept_)
print(clf.coef_[0],clf.coef_[1])
#(array([-0.03573098,  0.11672256]), 2.996914661970776)
sumtrain,sumtest=0,0
for i in range(halflen):
  val1=ytrain[i]-(clf.coef_[0]*xxtrain[i][0]+clf.coef_[1]*xxtrain[i][1]+clf.intercept_)
  sumtrain+=val1*val1
  val2=ytest[i]-(clf.coef_[0]*xxtest[i][0]+clf.coef_[1]*xxtest[i][1]+clf.intercept_)
  sumtest+=val2*val2
MSE1=sumtrain/halflen
MSE2=sumtest/halflen
print(MSE1,MSE2)
```

MSE of training and MSE of testing:(0.4839680560133444, 0.4237065211985007)

4.

week1.py | Extension: Python | beer_50000.json

```
import numpy
import urllib
import scipy.optimize
import random
from sklearn import linear_model

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

print "Reading data..."
data = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json"))
print "done"

x=[[1 if d['beer/style']=='Hefeweizen' else 0 for d in data],[d['beer/ABV'] for d in data]]
xx=[]

for i in range(len(data)):
  xx.append([x[0][i],x[1][i]])

y=[d['review/taste'] for d in data]
clf = linear_model.LinearRegression()

halflen=len(data)/2
ranstart=random.sample(range(halflen),1)[0]
xxtrain=xx[ranstart:(ranstart+halflen)]
ytrain=y[ranstart:(ranstart+halflen)]
xxtest=xx[:ranstart]+xx[(ranstart+halflen):]
ytest=y[:ranstart]+y[(ranstart+halflen):]

print(ranstart,(ranstart+halflen))

clf.fit(xxtrain,ytrain)
print(clf.coef_,clf.intercept_)
sumtrain,sumtest=0,0
for i in range(halflen):
  val1=ytrain[i]-(clf.coef_[0]*xxtrain[i][0]+clf.coef_[1]*xxtrain[i][1]+clf.intercept_)
  sumtrain+=val1*val1
  val2=ytest[i]-(clf.coef_[0]*xxtest[i][0]+clf.coef_[1]*xxtest[i][1]+clf.intercept_)
  sumtest+=val2*val2
MSE1=sumtrain/halflen
MSE2=sumtest/halflen
print(MSE1,MSE2)

```

start and end of the training part(14487, 39487)
linear regression coefficient and its intercept:
(array([-1.23361769, 0.09547546]), 3.3326207702584405)
MSE of training and MSE of testing:
(0.3595337172103225, 0.5945455303770442)

One possible reason:

Around the time the training part's ending the beer producer proposed some strategies to help the following selling situation (the test part) more organized or

correlated to some certain variables. However, the random division of the training and testing is unorganized or totally accident.

5.

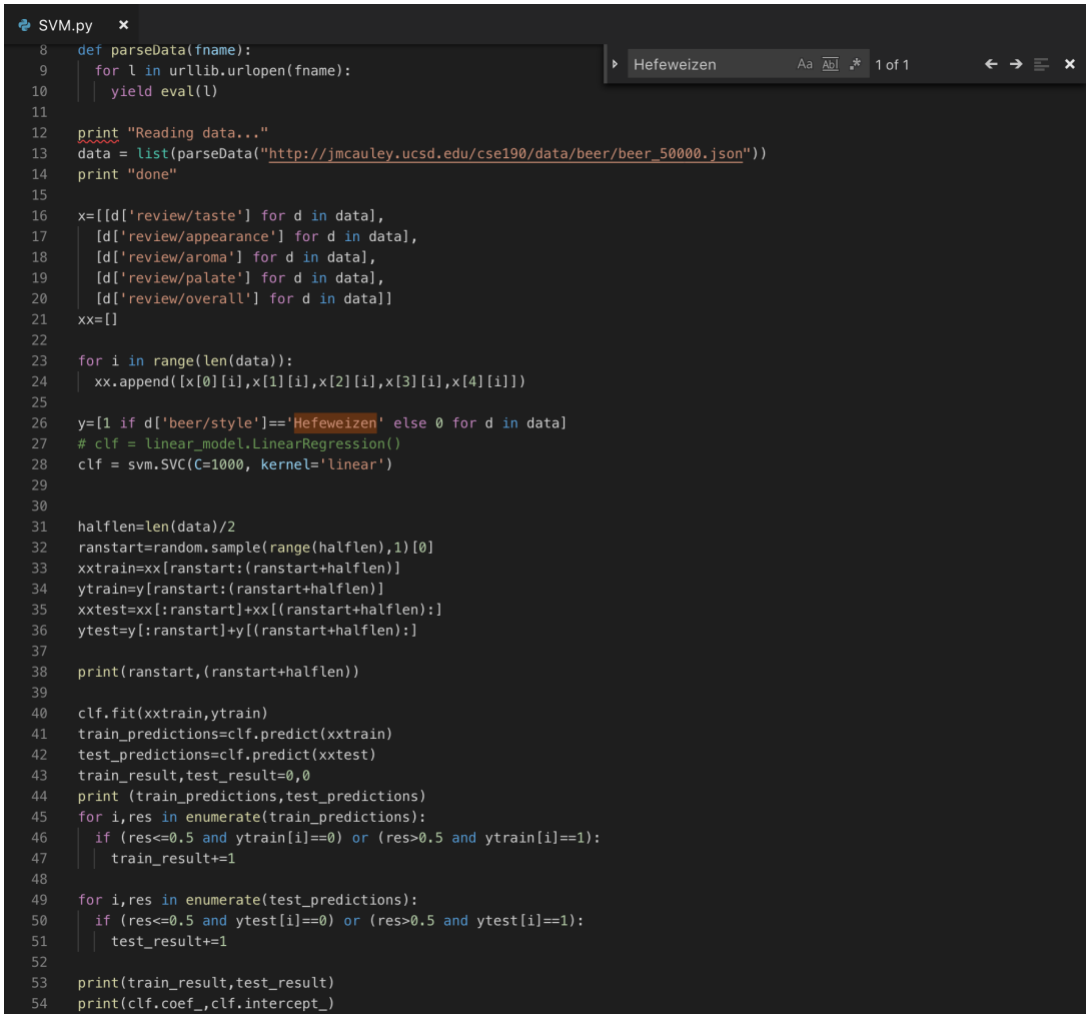
```
import numpy
import urllib
import scipy.optimize
import random
from sklearn import linear_model

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

print "Reading data..."
data = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json"))
print "done"

x=[[d['beer/ABV'] if d['beer/style']=='Hefeweizen' else 0 for d in data],
 [d['beer/ABV'] if d['beer/style']!='Hefeweizen' else 0 for d in data]]
xx=[]

for i in range(len(data)):
  xx.append([x[0][i],x[1][i]])

y=[d['review/taste'] for d in data]
clf = linear_model.LinearRegression()

halflen=len(data)/2
# ranstart=random.sample(range(halflen),1)[0]
xxtrain=xx[:halflen]
ytrain=y[:halflen]
xxtest=xx[halflen:]
ytest=y[halflen:]

# print(ranstart,(ranstart+halflen))

clf.fit(xxtrain,ytrain)
print(clf.coef_,clf.intercept_)
sumtrain,sumtest=0,0
for i in range(halflen):
  val1=ytrain[i]-(clf.coef_[0]*xxtrain[i][0]+clf.coef_[1]*xxtrain[i][1]+clf.intercept_)
  sumtrain+=val1*val1
  val2=ytest[i]-(clf.coef_[0]*xxtest[i][0]+clf.coef_[1]*xxtest[i][1]+clf.intercept_)
  sumtest+=val2*val2
MSE1=sumtrain/halflen
MSE2=sumtest/halflen
print(MSE1,MSE2)
```

linear regression coefficient and its intercept:
(array([0.11005028, 0.1167231]), 2.9969206914347235)
MSE of training and MSE of testing:
(0.4839671462417231, 0.42369661406846515)

6.

Although The model from Question 5 uses the same two features as the model from Questions 2-4 and has the same dimensionality, they are not identical. Moreover, the two variables in Question 5 is more like "yes or no", and the extent it impacts the 'review/taste' is more powerful.

7.



start and end of the training part (1054, 26054)
linear regression coefficient and its intercept:

(array([[6.31252769e-05, 5.16767614e-06, 3.27652087e-05,
4.49432991e-05, -1.99214555e-05]]), array([-1.0004203]))

the number of correct predictions and its percentage of training and testing data:
(24699, 24683) 98.80% 98.73%

8.
My opinion:
The result in the last case is not accurate simply because the percentage of 'Hefeweizen' is so low that the percentage of correct classification seems high. It is not an appropriate case in which the variable 'Hefeweizen' is so weak. Adapting 'linear' as the kernel of svm.SVC is usually in less-variable and fewer-data situation. Besides, the feature 'review/overall' is meaningless because it's somewhat based on other four features.

So when changing the kernel to 'rbf' and deleting the feature 'review/overall', the number of correct predictions and its percentage of training and testing data come to:
(24708, 24675) 98.83% 98.7%

```
8   def parseData(fname):
9     for l in urllib.urlopen(fname):
10      yield eval(l)
11
12  print "Reading data..."
13  data = list(parseData("http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json"))
14  print "done"
15
16  x=[[d['review/taste'] for d in data],
17    [d['review/appearance'] for d in data],
18    [d['review/aroma'] for d in data],
19    [d['review/palate'] for d in data]]
20  xx=[]
21
22  for i in range(len(data)):
23    xx.append([x[0][i],x[1][i],x[2][i],x[3][i]])
24
25  y=[1 if d['beer/style']=='Hefeweizen' else 0 for d in data]
26  # clf = linear_model.LinearRegression()
27  clf = svm.SVC(C=1000, kernel='rbf')
28
29
30  halflen=len(data)/2
31  ranstart=random.sample(range(halflen),1)[0]
32  xxtrain=xx[ranstart:(ranstart+halflen)]
33  ytrain=y[ranstart:(ranstart+halflen)]
34  xxtest=xx[:ranstart]+xx[(ranstart+halflen):]
35  ytest=y[:ranstart]+y[(ranstart+halflen):]
36
37  print(ranstart,(ranstart+halflen))
38
39  clf.fit(xxtrain,ytrain)
40  train_predictions=clf.predict(xxtrain)
41  test_predictions=clf.predict(xxtest)
42  train_result,test_result=0,0
43  # print (train_predictions,test_predictions)
44  for i,res in enumerate(train_predictions):
45    if (res<=0.5 and ytrain[i]==0) or (res>0.5 and ytrain[i]==1):
46      train_result+=1
47
48  for i,res in enumerate(test_predictions):
49    if (res<=0.5 and ytest[i]==0) or (res>0.5 and ytest[i]==1):
50      test_result+=1
51
52  print(train_result,test_result)
53  # print(clf.coef_,clf.intercept_)
```

But after running it I would rather 'linear' kernel because the 'rbf' takes much longer time to finish.