# Carbon

1. Load data

```python
#load module data
import data
#create a dataset object and read data from file sample.txt
sample = data.DataSet()
#To read nominal data, you have to add argument 'nominal',
default is 'numeric'
#Read data from 'sample.txt'
sample.read('sample.txt', 'numeric')
#view the data
sample.x
array([[  5.,    1.,    1., ...,    3.,    1.,    1.],
       [  5.,    4.,    4., ...,    3.,    2.,    1.],
       [  3.,    1.,    1., ...,    3.,    1.,    1.],
       ...,
       [  5.,   10.,   10., ...,    8.,   10.,    2.],
       [  4.,    8.,    6., ...,   10.,    6.,    1.],
       [  4.,    8.,    8., ...,   10.,    4.,    1.]])
#view class labels
sample.y[:10]
['2', '2', '2', '2', '2', '4', '2', '2', '2', '2']
#view dataset dimension
sample.dim()
(699, 9)
#view features
sample.label
['Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of
Cell Shape', 'Marginal Adhesion', 'Single Epithelial Cell Size',
'Bare Nuclei', 'Bland Chromatin', 'Normal Nucleoli', 'Mitoses']
#view subject ids
sample.key[:5]
['1000025', '1002945', '1015425', '1016277', '1017023']
#create train dataset and test dataset using 1:10 hold out
train,test = data.holdOut(sample,0.1)
```

2. kNN
(this kNN is using Euclidean distance as default distance function)

```python
#load kNN module
import kNN
#create a classifier
clf = kNN.build()
#train the classifier with train data and k=4
clf.train(train,4)
#to view the model by plot 2 features with class label
clf.view('Uniformity of Cell Size','Uniformity of Cell Shape')
```
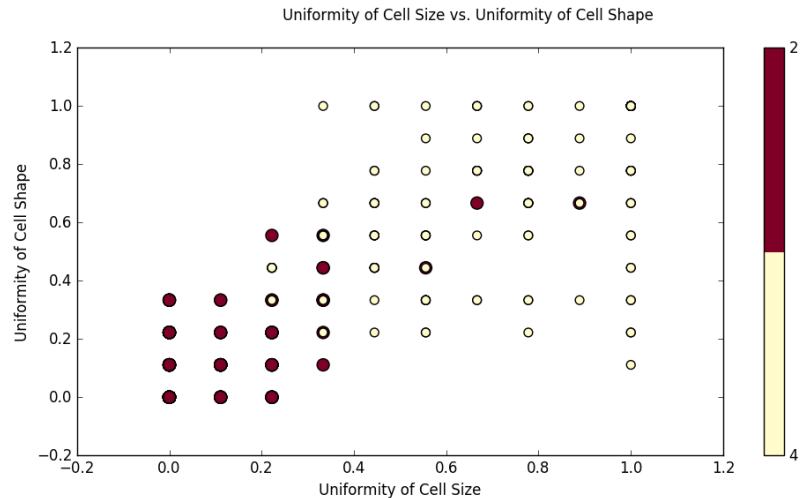
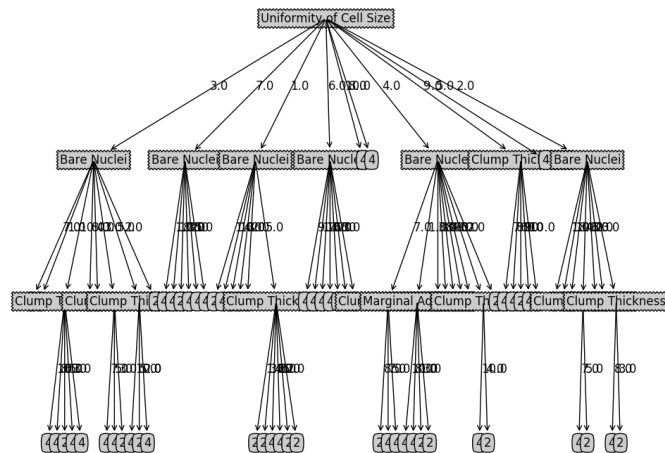Uniformity of Cell Size vs. Uniformity of Cell Shape

```
#to classify a new subject
clf.classify([5, 4, 4, 5, 7, 10, 3, 2, 1])
'4'
#test the classifier with test data
#and save the predicted labels in result
result = clf.test(test)
the total error rate is: 0.130435
result
['2', '4', '2', '4', '2', '4', '2', '2', '2', '2', '2', '2', '2',
'2', '4', '2', '2', '2', '4', '2', '4', '4', '2', '2', '2', '4',
'2', '2', '2', '2', '2', '2', '4', '2', '2', '2', '4', '2', '4',
'4', '4', '4', '4', '4', '4', '2', '4', '2', '2', '4', '4', '2',
'4', '4', '4', '4', '4', '4', '4', '2', '4', '2', '4', '2', '2',
'4', '2', '4', '4']
#to save the classifier in folder models/ as model.knn
clf.save('model')
#to load saved classifier
clf2 = kNN.load('model')
```

3. ID3
(works only on nominal data)

```
#convert continuous data to nominal data
train.num2nom()
test.num2nom()
train.type
'nominal'
#load ID3 module
import ID3
#create a classifier
clf = ID3.build()
#train the classifier with train data to build the tree
clf.train(train)
#to view the model
clf.view()
```

Uniformity of Cell Size

Bare Nuclei Bare Nuclei Bare Nuclei Bare Nuclei Bare Nuclei Clump Thic Bare Nuclei

Clump Clump Thi Clump Thick Clump Thick Marginal Ad Clump Th Clump Clump Thickness

```
#to classify a new subject
clf.classify(['5.0', '4.0', '4.0', '5.0', '7.0', '10.0', '3.0',
'2.0', '1.0'])
'4'
#test the classifier with test data
#and save the predicted labels in result
result = clf.test(test)
the total error rate is: 0.115942
result
['2', '4', '2', '4', '2', '4', '4', '2', '2', '2', '2', '2', '4',
'2', '4', '4', '2', '2', '4', '2', '4', '4', '2', '2', '2', '4',
'2', '2', '2', '2', '2', '2', '4', '2', '2', '2', '4', '2', '4',
'4', '2', '2', '4', '4', '4', '2', '4', '2', '2', '4', '4', '4',
'4', '4', '4', '4', '4', '2', '4', '4', '4', '2', '4', '2', '2',
'2', '2', '4', '4']
#to save the classifier in folder models/ as model.id3
clf.save('model')
#to load saved classifier
clf3 = ID3.load('model')
```

4. NB

```
#load NB module
import NB
#create a classifier
clf = NB.build()
#train the classifier with train data
clf.train(train)
#to view the model
#we can view the conditional prob of a certain feature
clf.view('Uniformity of Cell Size')
```
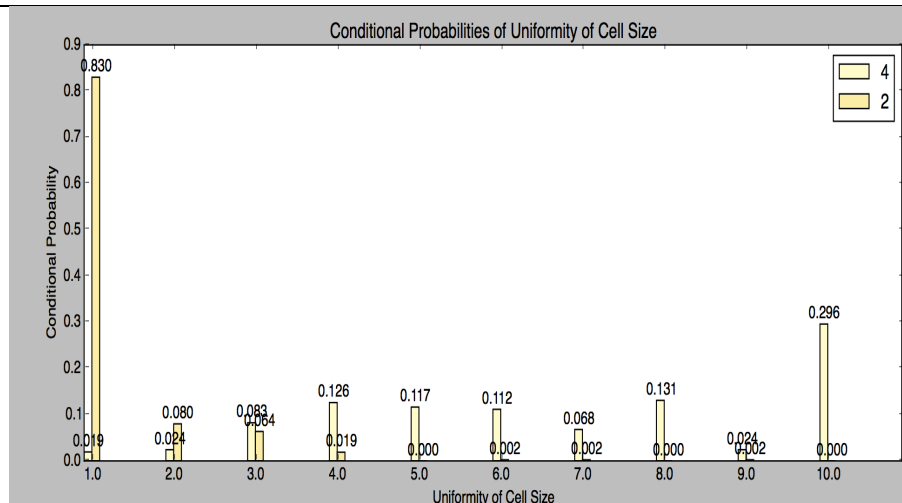
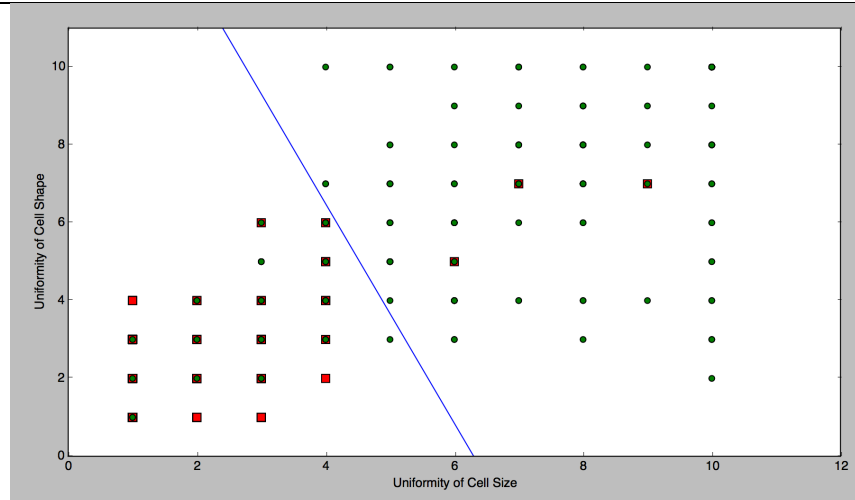Conditional Probabilities of Uniformity of Cell Size

```
#to classify a new subject
clf.classify([5, 4, 4, 5, 7, 10, 3, 2, 1])
'4'
#test the classifier with test data
#and save the predicted labels in result
result = clf.test(test)
the total error rate is: 0.043478
result
['2', '4', '2', '4', '2', '4', '2', '2', '2', '2', '2', '2', '4',
 '2', '4', '4', '2', '2', '4', '2', '4', '4', '2', '4', '2', '4',
 '2', '2', '2', '2', '2', '2', '4', '2', '2', '2', '4', '2', '4',
 '4', '4', '4', '4', '4', '4', '2', '4', '2', '2', '4', '4', '4',
 '4', '4', '4', '4', '4', '4', '4', '4', '4', '2', '4', '4', '2',
 '4', '2', '4', '4']
#to save the classifier in folder models/ as model.nb
clf.save('model')
#to load saved classifier
clf4 = NB.load('model')
```

5. logReg
(Only works for bi-class problems)

```
#load logReg module
import logReg
#create a classifier
clf = logReg.build()
#train the classifier with train data
clf.train(train)
#to view the model
#we can view the model with 2 features over train dataset
#The line in the middle is the possibility of 0.5
clf.view('Uniformity of Cell Size','Uniformity of Cell
Shape',train)
```
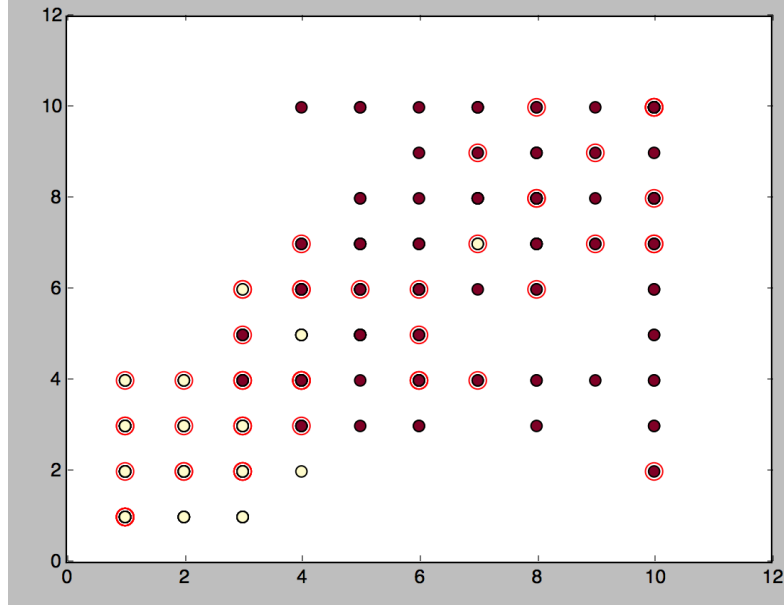
```
#to classify a new subject
clf.classify([5, 4, 4, 5, 7, 10, 3, 2, 1])
'4'
#test the classifier with test data
#and save the predicted labels in result
result = clf.test(test)
the total error rate is: 0.086957
result
['2', '4', '2', '4', '2', '4', '2', '2', '2', '2', '2', '2', '2',
'2', '4', '4', '2', '2', '4', '2', '4', '4', '2', '4', '2', '4',
'2', '2', '2', '2', '2', '2', '4', '2', '2', '2', '4', '2', '4',
'4', '4', '4', '4', '4', '4', '2', '4', '2', '2', '4', '4', '2',
'4', '4', '4', '4', '4', '4', '4', '4', '4', '2', '4', '2', '2',
'4', '2', '4', '4']
#to save the classifier in folder models/ as model.logreg
clf.save('model')
#to load saved classifier
clf5 = logReg.load('model')
```

6. SVM
(Only works for bi-class problems)

```
#load SVM module
import SVM
#create a classifier
clf = SVM.build()
#train the classifier with train data
#Here we set C to 100, tolerance to 0.001 and max iteration
number to 400. And use linear kernel.
clf.train(train,C=100,toler=0.001,maxIter=100,kTup=('lin',0))
#to view the model
#we can view the model with 2 features over train dataset
#circled points are support vectors
clf.view('Uniformity of Cell Size','Uniformity of Cell
Shape',train)
```
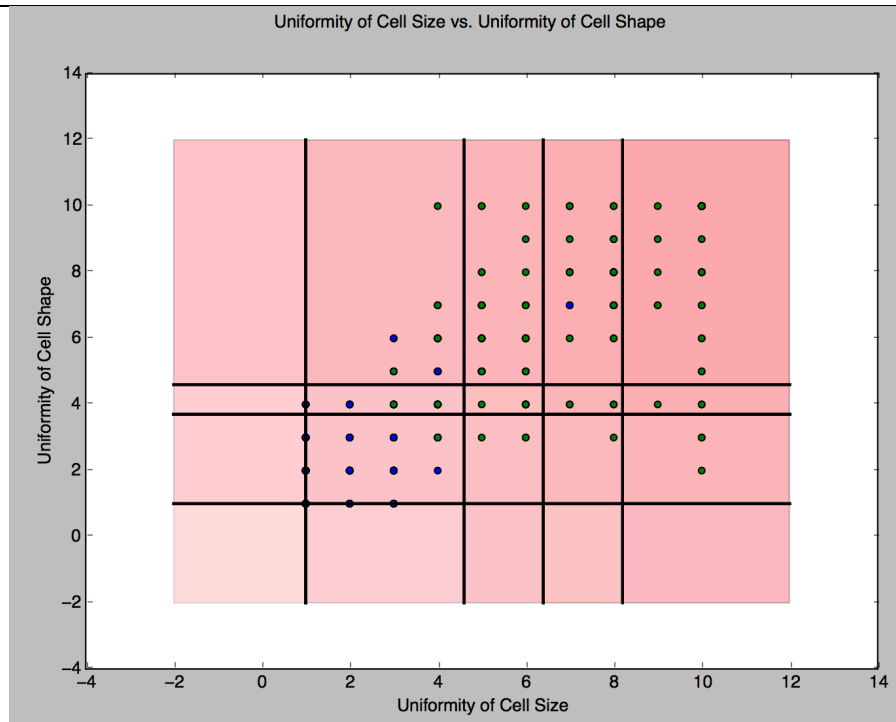
```
#to classify a new subject
clf.classify([5, 4, 4, 5, 7, 10, 3, 2, 1])
'4'
#test the classifier with test data
#and save the predicted labels in result
result = clf.test(test)
the total error rate is: 0.043478
result
['2', '4', '2', '4', '2', '4', '2', '2', '2', '2', '2', '2', '4',
 '2', '4', '4', '2', '2', '4', '2', '4', '4', '2', '4', '2', '4',
 '2', '2', '2', '2', '2', '2', '4', '2', '2', '2', '4', '2', '4',
 '4', '4', '4', '4', '4', '4', '2', '4', '2', '2', '4', '4', '4',
 '4', '4', '4', '4', '4', '4', '4', '4', '4', '2', '4', '4', '2',
 '4', '2', '4', '4']
#to save the classifier in folder models/ as model.logreg
clf.save('model')
#to load saved classifier
clf6 = SVM.load('model')
```

6. AdaBoost
(Only works for bi-class problems)

```
#load Ada module
import Ada
#create a classifier
clf = Ada.build()
#train the classifier with train data
#maximum number of iteration is 50
clf.train(train,numIt=50)
#to view the model
#we can view the model with 2 features over train dataset
#circled points are support vectors
clf.view('Uniformity of Cell Size','Uniformity of Cell
Shape',train)
```

Uniformity of Cell Size vs. Uniformity of Cell Shape

```
#to classify a new subject
clf.classify([5, 4, 4, 5, 7, 10, 3, 2, 1])
'4'
#test the classifier with test data
#and save the predicted labels in result
result = clf.test(test)
the total error rate is: 0.130435
result
['2', '4', '2', '4', '2', '4', '2', '2', '2', '2', '2', '2', '4',
'2', '4', '2', '2', '2', '4', '2', '4', '4', '2', '4', '2', '2',
'2', '2', '2', '2', '2', '2', '4', '2', '2', '2', '4', '2', '4',
'4', '4', '4', '4', '2', '4', '2', '4', '2', '2', '4', '4', '4',
'4', '4', '4', '4', '4', '2', '4', '4', '4', '2', '4', '2', '2',
'2', '2', '4', '4']
#to save the classifier in folder models/ as model.logreg
clf.save('model')
#to load saved classifier
clf7 = Ada.load('model')
```

```
#fast start
import data
sample = data.DataSet()
sample.read('sample.txt')
train,test = data.holdOut(sample,0.1)

reload(Ada)
clf = Ada.build()
clf.train(train)
clf.classify([5, 4, 4, 5, 7, 10, 3, 2, 1])
clf.view('Uniformity of Cell Size','Uniformity of Cell
Shape',train)
```