```
from pybrain.supervised.trainers import BackpropTrainer
from pybrain.tools.shortcuts import buildNetwork
from pybrain.datasets import ClassificationDataSet
from pybrain.tools.validation import Validator
import csv


def closeto(i):
    if i>0.5:
        return 1
    else:
        return 0

def intify(s):
    if s == ’Meat’:
        return 0
    elif s == ’Skin’:
        return 1
    else:
        return False

#read file
f = open("Skin-VS-Meat-train.csv", "rb")
csv.reader(f).next() #skip first line
names = csv.reader(f).next()
rdr = csv.DictReader(f, fieldnames=names)

ds = ClassificationDataSet(4, 1, 2)

i = 0
for row in rdr:
    ds.addSample([row[’ WL0 ’], row[’ WL1 ’], row[’ WL2 ’], row[’ WL3 ’]], intify(row["
        Label"]))
    i = i+1
    if i >10000:#because I can’t wait all day
        break
print ds.getLength()
```

```
10001
```

```
#validation data
f = open("Skin-VS-Meat-test.csv", "rb")
csv.reader(f).next() #skip first line
names = csv.reader(f).next()
rdr = csv.DictReader(f, fieldnames=names)

vds = ClassificationDataSet(4, 1, 2)

i = 0
for row in rdr:
    vds.addSample([row[’ WL0 ’], row[’ WL1 ’], row[’ WL2 ’], row[’ WL3 ’]], intify(row["
        Label"
    i   i
    if i            #because I can't wait all day
        break
print vds getLength
```

```
10001
```

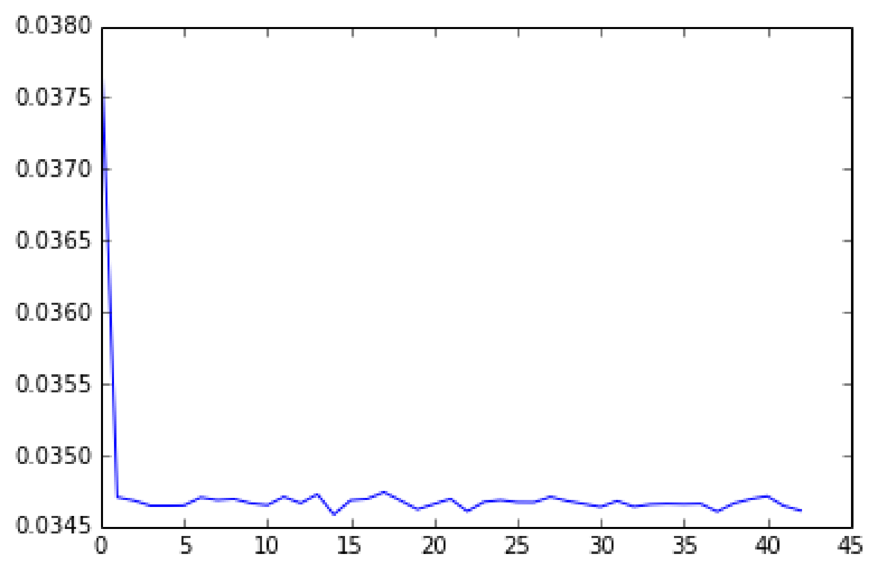
```
#create NN
net = buildNetwork(4, 2, 1, bias=True)

trainer = BackpropTrainer(net, ds)
error = []


#train
for _ in range(20):
    error.append(trainer.train())
```

```
#process outputs
plot(error)
```

```
[<matplotlib.lines.Line2D at 0x3fa64d0>]
```



```
output = net.activateOnDataset(vds)
target = vds.data['target']
target = target[:len(output)]#has to be same shape

#Make output binary
output = map(closeto, output)
```

```
predict = [[0, 0], [0, 0]]
for i in range(size(output)):
    if output[i] == 1 == target[i]:
        predict[0][0] = predict[0][0] +1
    elif output[i] == 1 != target[i]:
        predict[0][1] = predict[0][1] +1

    elif output[i] == 0 == target[i]:
        predict[1][0] = predict[1][0] +1
    elif output[i] == 0 != target[i]:
        predict[1][1] = predict[1][1] +1
    else:
        pass

print ’Confusion matrix:’
print predict

print ’Almost all assignments match because NNs are awesome’
```

```
Confusion matrix:
[[0, 2], [9999, 0]]
Almost all assignments match because NNs are awesome
```

```
#RANDOMIZED DATA
```

```
#randomized data
f = open("Skin-VS-Workpieces-ID7.csv", "rb")
csv.reader(f).next() #skip first line
names = csv.reader(f).next()
rdr = csv.DictReader(f, fieldnames=names)

rds = ClassificationDataSet(4, target=1, nb_classes=2)

print rdr.fieldnames

i=0
for row in rdr:
    rds.addSample([row[’ WL0 (830nm) ’], row[’ WL1 (1060nm) ’], row[’ WL2 (1300nm) ’],
        row[’ WL3 (1550nm) ’]]
                 , intify(row[" Label"]))
    i = i+1
    if i >1000:#because I can’t wait all day
        break

print rds.getLength()
```

```
[’Distance (mm) ’, ’ Q_0_1’, ’ Q_0_2’, ’ Q_0_3’, ’ Q_1_2’, ’ Q_1_3’, ’ Q_2_3 ’, ’ WL0 (830nm) ’
1001
```

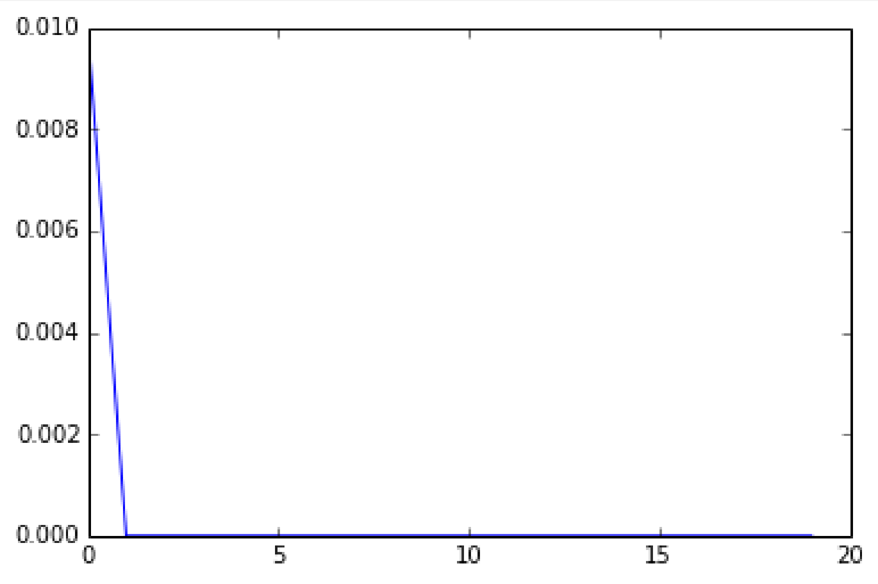
```
from pybrain.structure.modules import SoftmaxLayer

#split data
tstdata, trndata = rds.splitWithProportion( 0.25 )
#trndata._convertToOneOfMany( )
#tstdata._convertToOneOfMany( )
```

```
#create NN
fnn = buildNetwork(4, 2, 1)
trainer = BackpropTrainer(fnn, trndata)

error = []
for _ in range(20):
    error.append(trainer.train())
plot(error)
```

```
[<matplotlib.lines.Line2D at 0x3b89490>]
```



```
output = fnn.activateOnDataset(tstdata)
target = rds.data[’target’]
target = target[:len(output)]#has to be same shape

#Make output binary
output = map(closeto, output)

predict = [[0, 0], [0, 0]]
for i in range(size(output)):
    if output[i] == 1 == target[i]:
        predict[0][0] = predict[0][0] +1
    elif output[i] == 1 != target[i]:
        predict[0][1] = predict[0][1] +1

    elif output[i] == 0 == target[i]:
        predict[1][0] = predict[1][0] +1
    elif output[i] == 0 != target[i]:
        predict[1][1] = predict[1][1] +1
    else:
        pass

print 'Confusion matrix:'
print predict

print 'Almost all assignments match because NNs are awesome'
```

```
Confusion matrix:
[[0, 0], [250, 0]]
Almost all assignments match because NNs are awesome
```