HW4-KAI YE

November 24, 2017

1 Problem1

1.1

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        import mltools as ml
        X = np.genfromtxt('X_train.txt', delimiter=None)
        Y = np.genfromtxt('Y_train.txt', delimiter=None)
        X,Y = ml.shuffleData(X,Y)
        for i in range(X.shape[1]):
            print("min is:", np.min(X[:, i]))
            print("max is:", np.max(X[:, i]))
            print("mean is:",np.mean(X[:,i]))
            print("variance is:",np.var(X[:,i]))
```

```
min is: 193.5
max is: 253.0
mean is: 241.6011037
variance is: 83.4991711498
min is: 152.5
max is: 249.0
mean is: 227.3765713
variance is: 92.625593125
min is: 214.25
max is: 252.5
mean is: 241.5541505
variance is: 35.2863398033
min is: 152.5
max is: 252.5
mean is: 232.82676815
variance is: 97.6257317486
min is: 10.0
max is: 31048.0
mean is: 3089.923365
variance is: 15651513.7564
min is: 0.0
max is: 13630.0
```

```
mean is: 928.25902
variance is: 3081761.81695
min is: 0.0
max is: 9238.0
mean is: 138.09383
variance is: 443951.746446
min is: 0.0
max is: 125.17
mean is: 3.2485793303
variance is: 8.21948502491
min is: 0.87589
max is: 19.167
mean is: 6.49865290275
variance is: 6.40504819136
min is: 0.0
max is: 13.23
mean is: 2.09713912048
variance is: 4.36344047061
min is: 0.0
max is: 66.761
mean is: 4.21766040935
variance is: 4.08637188423
min is: 0.0
max is: 73.902
mean is: 2.69171845215
variance is: 2.19877847436
min is: 0.99049
max is: 975.04
mean is: 10.2715904759
variance is: 404.646245041
min is: -999.9
max is: 797.2
mean is: 5.7814805
variance is: 3406.52055098
```

1.2

```
In [6]: import numpy as np
        import matplotlib.pyplot as plt
        import mltools as ml
        X = np.genfromtxt('X_train.txt', delimiter=None)
        Y = np.genfromtxt('Y_train.txt', delimiter=None)
        X,Y = ml.shuffleData(X,Y)
        Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
        Xt, Yt = Xtr[:5000], Ytr[:5000]
        Xv, Yv = Xva[:5000], Yva[:5000]
        XtS, params = ml.transforms.rescale(Xt)
        XvS, _ = ml.rescale(Xv, params)
        for i in range(XtS.shape[1]):
            print("XtS's min is:", np.min(XtS[:, i]))
            print("XtS's max is:", np.max(XtS[:, i]))
            print("XtS's mean is:",np.mean(XtS[:,i]))
            print("XtS's variance is:",np.var(XtS[:,i]))
        for i in range(XvS.shape[1]):
            print("XvS's min is:", np.min(XvS[:, i]))
            print("XvS's max is:", np.max(XvS[:, i]))
            print("XvS's mean is:",np.mean(XvS[:,i]))
            print("XvS's variance is:",np.var(XvS[:,i]))
```

```
XtS's min is: -4.52126957134
XtS's max is: 1.25218967078
XtS's mean is: 3.18287618484e-15
XtS's variance is: 1.0
XtS's min is: -3.91860859565
XtS's max is: 1.8546250746
XtS's mean is: -5.73763259126e-16
XtS's variance is: 1.0
XtS's min is: -4.44384964661
XtS's max is: 1.69854155328
XtS's mean is: -5.44957856619e-14
XtS's variance is: 1.0
XtS's min is: -2.77832525137
XtS's max is: 1.89364918514
XtS's mean is: -8.21351875402e-14
XtS's variance is: 1.0
XtS's min is: -0.786772175108
XtS's max is: 7.35655916225
XtS's mean is: 5.47117906535e-17
XtS's variance is: 1.0
XtS's min is: -0.532268502963
XtS's max is: 7.2964207308
XtS's mean is: 3.19744231092e-18
XtS's variance is: 1.0
XtS's min is: -0.206599234995
XtS's max is: 13.6108766848
XtS's mean is: -1.84741111298e-17
XtS's variance is: 1.0
XtS's min is: -1.17945974003
XtS's max is: 8.47555116736
XtS's mean is: -1.62430069395e-15
XtS's variance is: 1.0
XtS's min is: -1.93339374679
XtS's max is: 4.15006601356
XtS's mean is: -2.61053401118e-15
XtS's variance is: 1.0
```

```
XtS's min is: -1.01139144701
XtS's max is: 4.32615602159
XtS's mean is: -4.44799752586e-16
XtS's variance is: 1.0
XtS's min is: -2.14730737157
XtS's max is: 6.53341798045
XtS's mean is: -5.67723645872e-16
XtS's variance is: 1.0
XtS's min is: -1.9201633731
XtS's max is: 13.4098098436
XtS's mean is: 1.94191329683e-15
XtS's variance is: 1.0
XtS's min is: -0.570186724663
XtS's max is: 29.9918388489
XtS's mean is: 7.07700564817e-16
XtS's variance is: 1.0
XtS's min is: -18.3287075848
XtS's max is: 10.0881593252
XtS's mean is: -2.48689957516e-17
XtS's variance is: 1.0
XvS's min is: -4.73913595784
XvS's max is: 1.25218967078
XvS's mean is: -0.00814820285492
XvS's variance is: 0.997479385864
XvS's min is: -7.85490427991
XvS's max is: 2.27449661425
XvS's mean is: -0.021730241599
XvS's variance is: 1.05025649636
XvS's min is: -4.49979745495
XvS's max is: 1.80535100555
XvS's mean is: -0.0176934096165
XvS's variance is: 1.03198900703
XvS's min is: -8.14962412559
XvS's max is: 1.96571309672
XvS's mean is: -0.0236245801846
XvS's variance is: 1.03716711663
XvS's min is: -0.786772175108
XvS's max is: 7.35655916225
XvS's mean is: 0.0342559393831
XvS's variance is: 1.11859965305
XvS's min is: -0.532268502963
XvS's max is: 7.2964207308
XvS's mean is: 0.0125978141381
XvS's variance is: 1.05802832739
XvS's min is: -0.206599234995
XvS's max is: 13.6108766848
XvS's mean is: 0.0159494774544
XvS's variance is: 1.15618783962
```

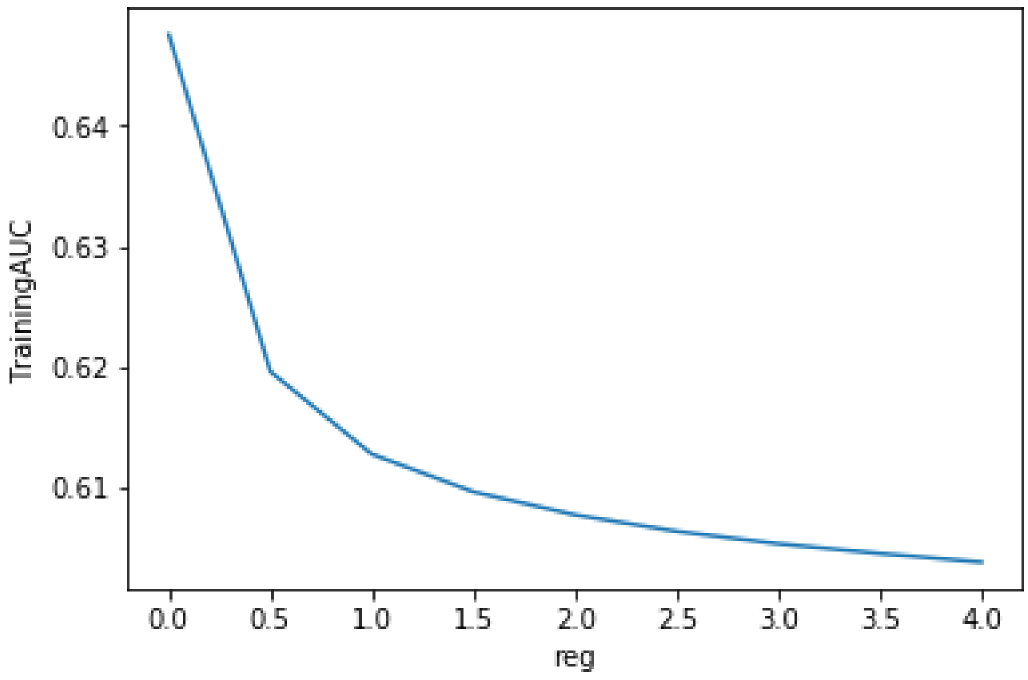
```
XvS's min is: -1.17945974003
XvS's max is: 44.8137551815
XvS's mean is: 0.0250881350382
XvS's variance is: 1.48422650539
XvS's min is: -2.13135249299
XvS's max is: 4.12419165501
XvS's mean is: 0.0204557583377
XvS's variance is: 1.03440387751
XvS's min is: -1.01139144701
XvS's max is: 4.3025917522
XvS's mean is: 0.0191620000848
XvS's variance is: 1.04672639733
XvS's min is: -2.14730737157
XvS's max is: 8.2407837827
XvS's mean is: -0.00489710986497
XvS's variance is: 1.05811164659
XvS's min is: -1.9201633731
XvS's max is: 19.9096899135
XvS's mean is: 0.0114467224141
XvS's variance is: 1.08830311216
XvS's min is: -0.572422026306
XvS's max is: 61.1062245557
XvS's mean is: 0.0214423760733
XvS's variance is: 2.31203252062
XvS's min is: -18.3287075848
XvS's max is: 13.0515128993
XvS's mean is: -0.0284496513995
XvS's variance is: 1.49099410153
```

2 Problem2

2.1

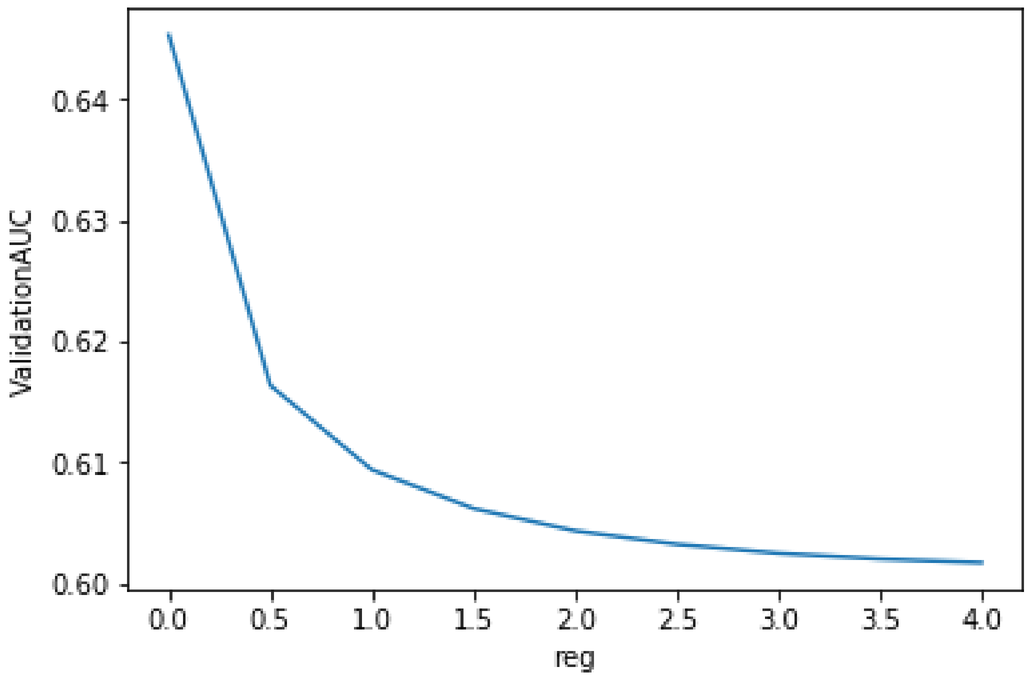
```
In [17]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         regcollection=[0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0]
         auccollection=[]
```

```
learner = ml.linearC.linearClassify()
for x in regcollection:
    learner.train(XtS, Yt, x, initStep=0.1, stopTol=1e-6, stopIter=100)
    auccollection.append(learner.auc(XtS, Yt))# training AUC
plt.plot(regcollection,auccollection)
plt.xlabel('reg')
plt.ylabel('TrainingAUC')
plt.show()
```



```
In [18]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         regcollection=[0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0]
         auccollection=[]
         learner = ml.linearC.linearClassify()
         for x in regcollection:
             learner.train(XtS, Yt, x, initStep=0.1, stopTol=1e-6, stopIter=100)
```

```
        auccollection.append(learner.auc(XvS, Yv))# validation AUC
    plt.plot(regcollection,auccollection)
    plt.xlabel('reg')
    plt.ylabel('ValidationAUC')
    plt.show()
```



2.2

```
In [12]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         Xt2 = xform.fpoly(XtS,2, bias=False)
         print (Xt2.shape[1])
```

```
119
```

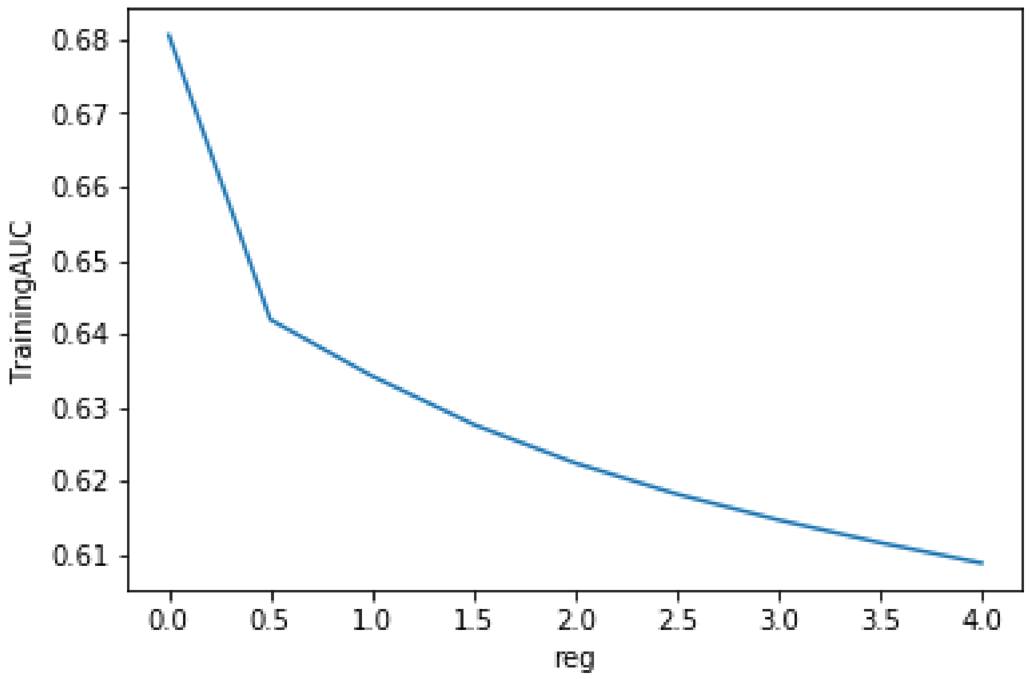
The number of features is 119 because it consists of 14 original data and 14 x1*x1, x2*x2...x14*x14 also other 14*13/2=91 polynomials such as : x1*x2 x2*x3...

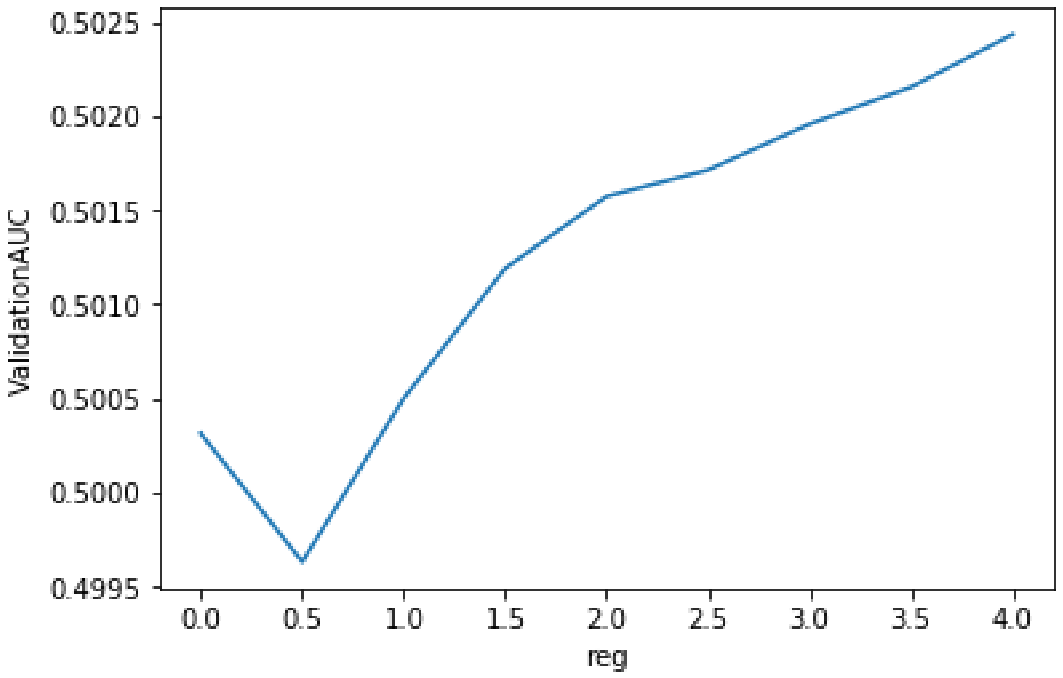
2.3

```
In [19]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         Xt2 = xform.fpoly(Xt,2, bias=False)
         Xv2 = xform.fpoly(Xv,2, bias=False)

         Xt2S, params = ml.transforms.rescale(Xt2)
         Xv2S, _ = ml.rescale(Xv2, params)
         regcollection=[0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0]
         trainingauccollection=[]
         validationauccollection=[]
         learner = ml.linearC.linearClassify()
         for x in regcollection:
             learner.train(Xt2S, Yt, x, initStep=0.1, stopTol=1e-6, stopIter=100)
             trainingauccollection.append(learner.auc(Xt2S, Yt))
         plt.plot(regcollection,trainingauccollection) # training AUC
         plt.xlabel('reg')
         plt.ylabel('TrainingAUC')
         plt.show()
         for x in regcollection:
             learner.train(Xt2S, Yt, x, initStep=0.1, stopTol=1e-6, stopIter=100)
             validationauccollection.append(learner.auc(Xv2S, Yt))
         plt.plot(regcollection,validationauccollection) # validation AUC
         plt.xlabel('reg')
         plt.ylabel('ValidationAUC')
         plt.show()
```

```
C:\Users\admin\Desktop\mltools\base.py:97: RuntimeWarning: divide by zero encountered in log
  return - np.mean( np.log( P[ np.arange(M), Y ] ) ) # evaluate
```

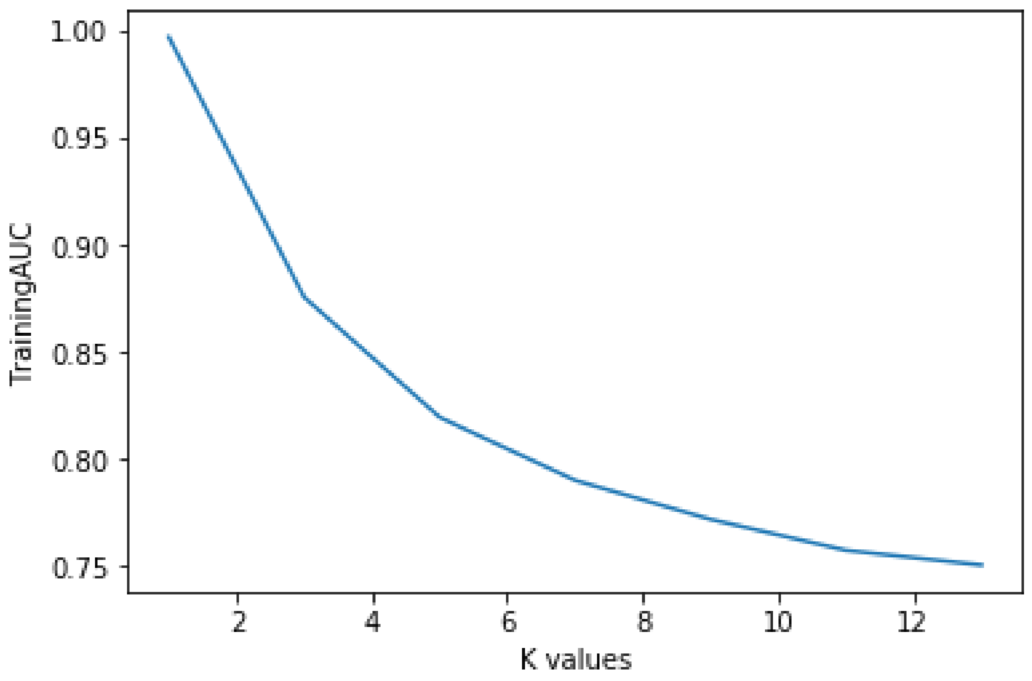

TrainingAUC
0.68
0.67
0.66
0.65
0.64
0.63
0.62
0.61
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
reg

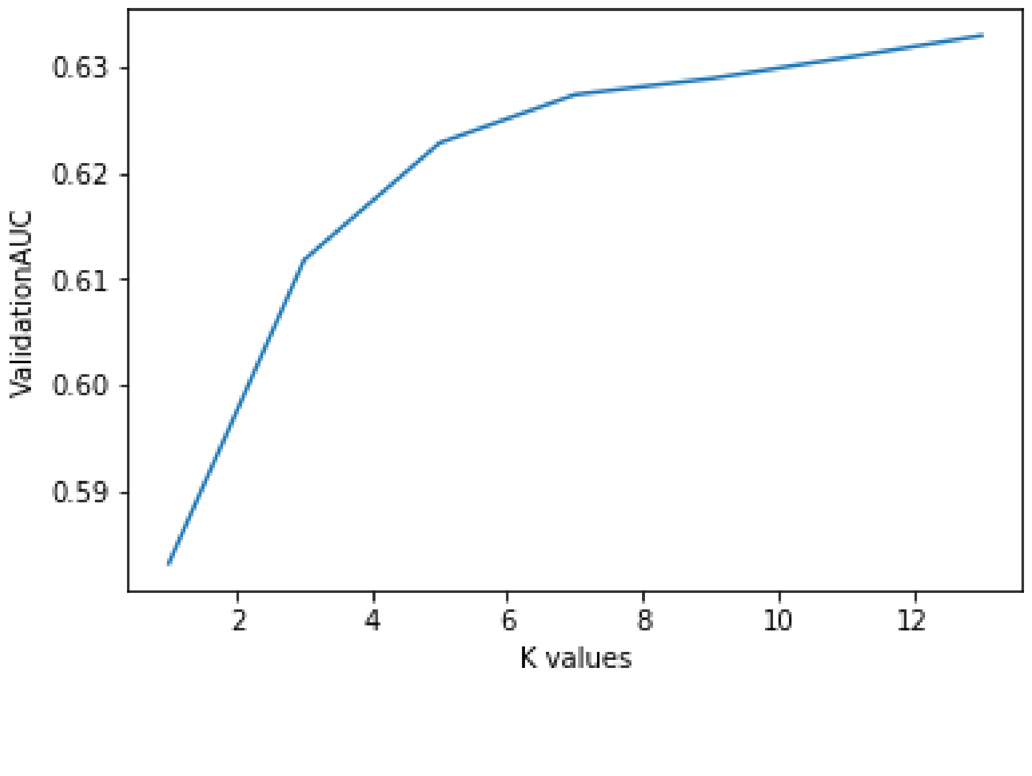

ValidationAUC
0.5025
0.5020
0.5015
0.5010
0.5005
0.5000
0.4995
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
reg

3 Problem3

3.1

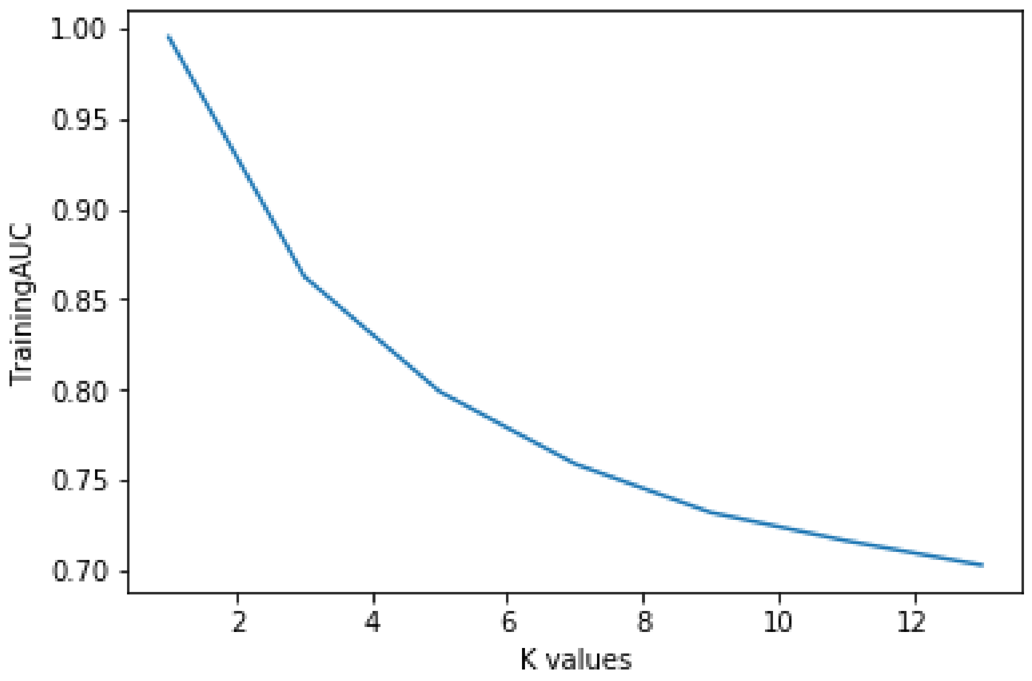
```
In [20]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         kcollection=[1,3,5,7,9,11,13]
         TrainingAUC=[]
         ValidationAUC=[]
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         learner = ml.knn.knnClassify()
         for k in kcollection:
             learner.train(XtS, Yt,k, alpha=0.0)
             TrainingAUC.append(learner.auc(XtS, Yt)) # train AUC
             ValidationAUC.append(learner.auc(XvS,Yv))
         plt.plot(kcollection,TrainingAUC)
         plt.xlabel('K values')
         plt.ylabel('TrainingAUC')
         plt.show()
         plt.plot(kcollection,ValidationAUC)
         plt.xlabel('K values')
         plt.ylabel('ValidationAUC')
         plt.show()
```

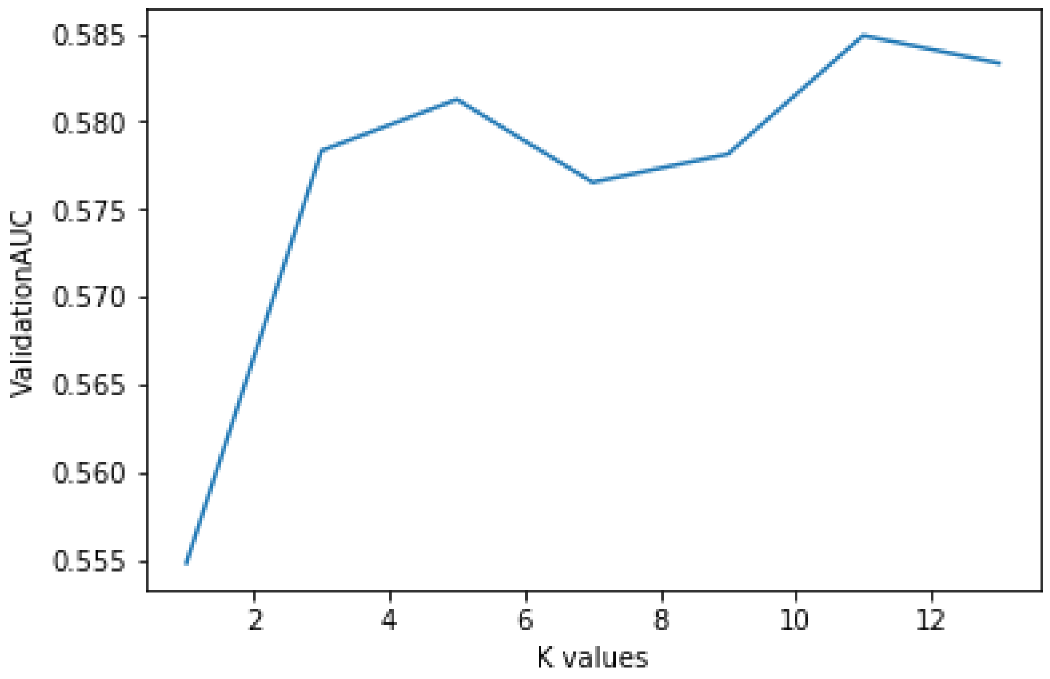

TrainingAUC
1.00
0.95
0.90
0.85
0.80
0.75
2
4
6
8
10
12
K values


ValidationAUC
0.63
0.62
0.61
0.60
0.59
2
4
6
8
10
12
K values

3.2

```
In [23]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         kcollection=[1,3,5,7,9,11,13]
         TrainingAUC=[]
         ValidationAUC=[]
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         learner = ml.knn.knnClassify()
         for k in kcollection:
             learner.train(Xt, Yt,k, alpha=0.0)
             TrainingAUC.append(learner.auc(Xt, Yt)) # train AUC
             ValidationAUC.append(learner.auc(Xv,Yv))
         plt.plot(kcollection,TrainingAUC)
         plt.xlabel('K values')
         plt.ylabel('TrainingAUC')
         plt.show()
         plt.plot(kcollection,ValidationAUC)
         plt.xlabel('K values')
         plt.ylabel('ValidationAUC')
         plt.show()
```


TrainingAUC
1.00
0.95
0.90
0.85
0.80
0.75
0.70
2
4
6
8
10
12
K values

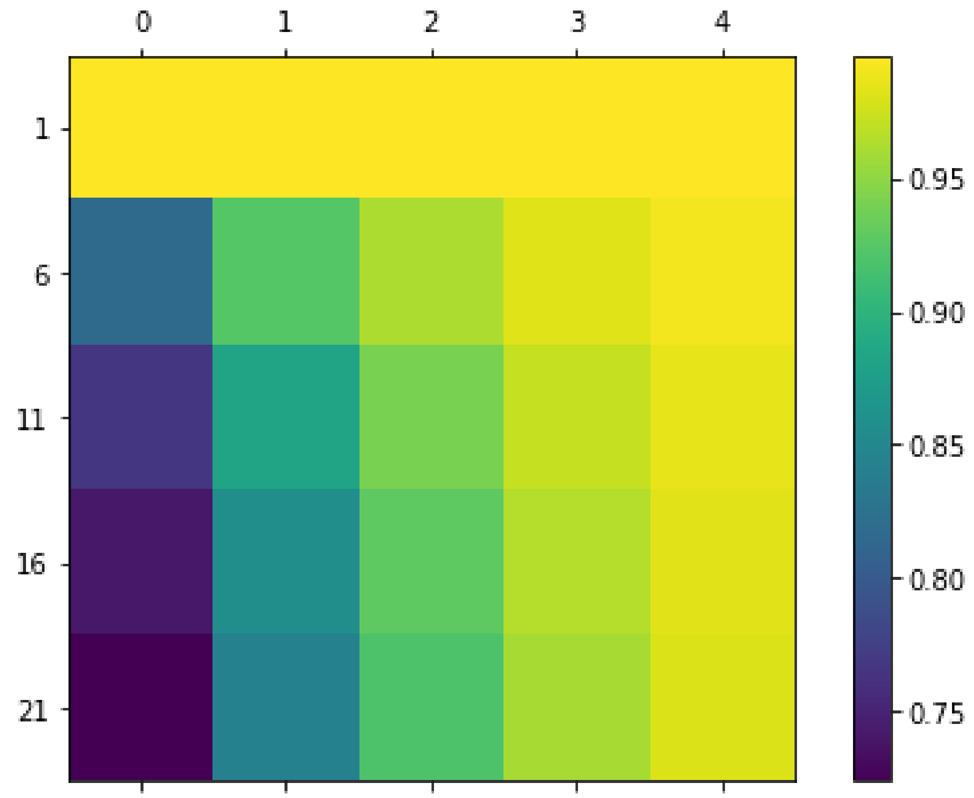

ValidationAUC
0.585
0.580
0.575
0.570
0.565
0.560
0.555
2
4
6
8
10
12
K values

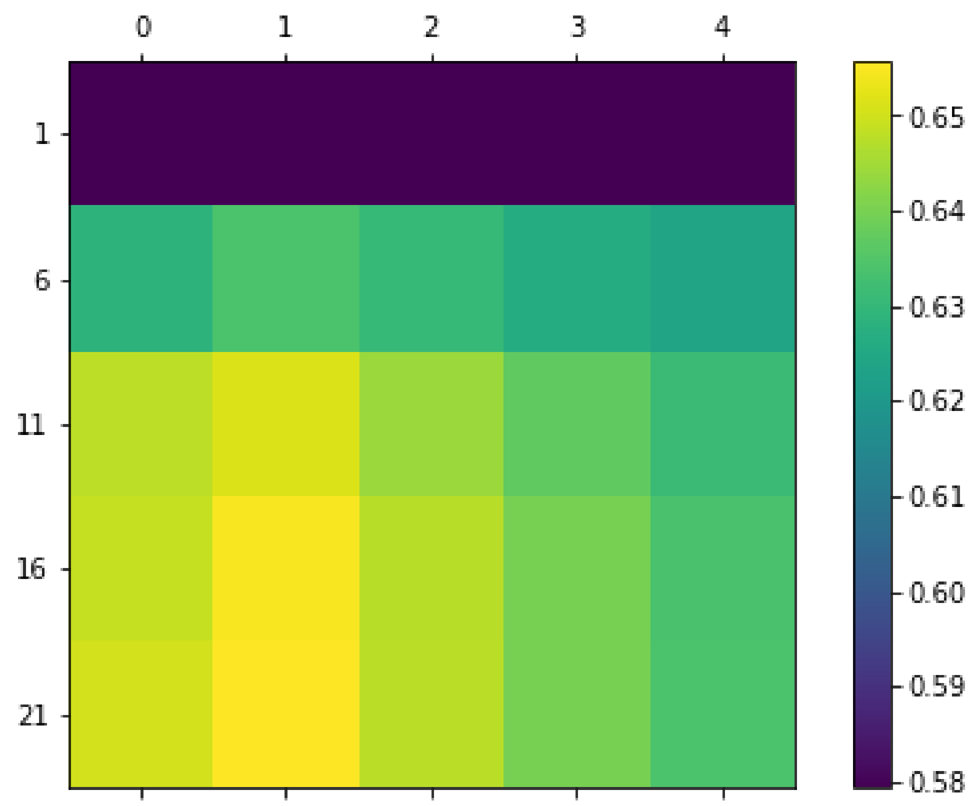
3.3

```
In [21]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         kcollection=[1,3,5,7,9,11,13]
         TrainingAUC=[]
         ValidationAUC=[]
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         K = range(1,25,5) # Or something else
         A = range(0,5,1) # Or something else
         tr_auc = np.zeros((len(K),len(A)))
         va_auc = np.zeros((len(K),len(A)))
         for i,k in enumerate(K):
             for j,a in enumerate(A):
                 learner.train(XtS, Yt,k,a)
                 tr_auc[i][j] =learner.auc(XtS,Yt)
                 va_auc[i][j] =learner.auc(XvS,Yv)
         # Now plot it
         f, ax = plt.subplots(1, 1, figsize=(8, 5))
         cax = ax.matshow(tr_auc, interpolation='nearest')
         f.colorbar(cax)
         ax.set_xticklabels(['']+list(A))
         ax.set_yticklabels(['']+list(K))
         plt.show()

         ff, axx = plt.subplots(1, 1, figsize=(8, 5))
         caxx = axx.matshow(va_auc, interpolation='nearest')
         ff.colorbar(caxx)
         axx.set_xticklabels(['']+list(A))
         axx.set_yticklabels(['']+list(K))
         plt.show()
```

```
C:\Users\admin\Desktop\mltools\knn.py:103: RuntimeWarning: invalid value encountered in true_di
  prob[i,:] = count / count.sum()    # save (soft) results
```


0
1
2
3
4
1
6
11
16
21
0.95
0.90
0.85
0.80
0.75



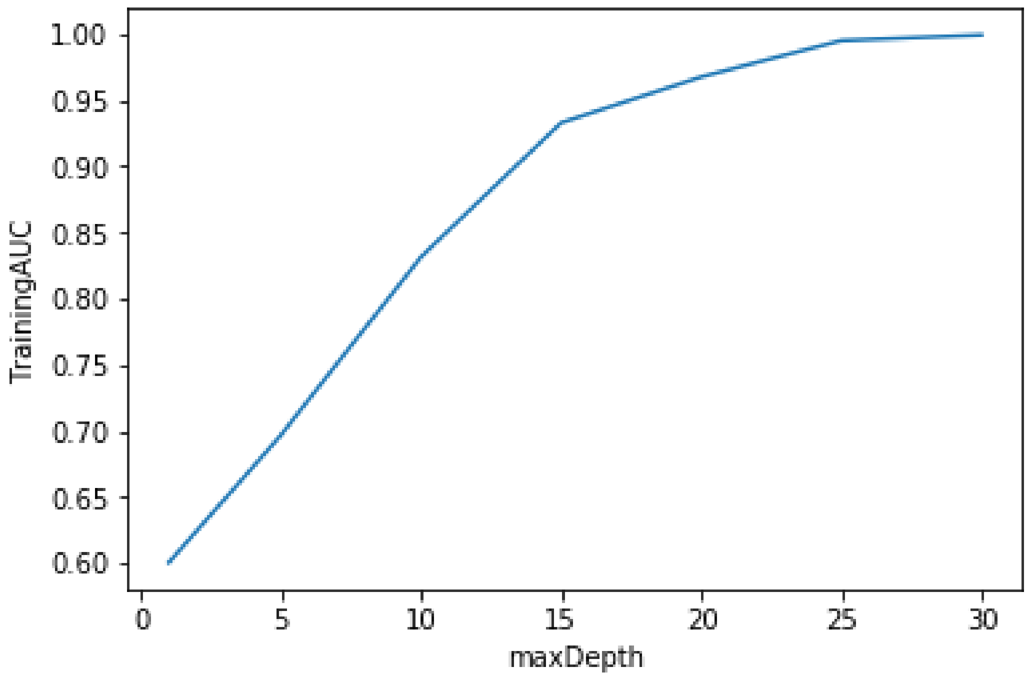
```
In [ ]: I would recommand K=21 and alpha=1
```

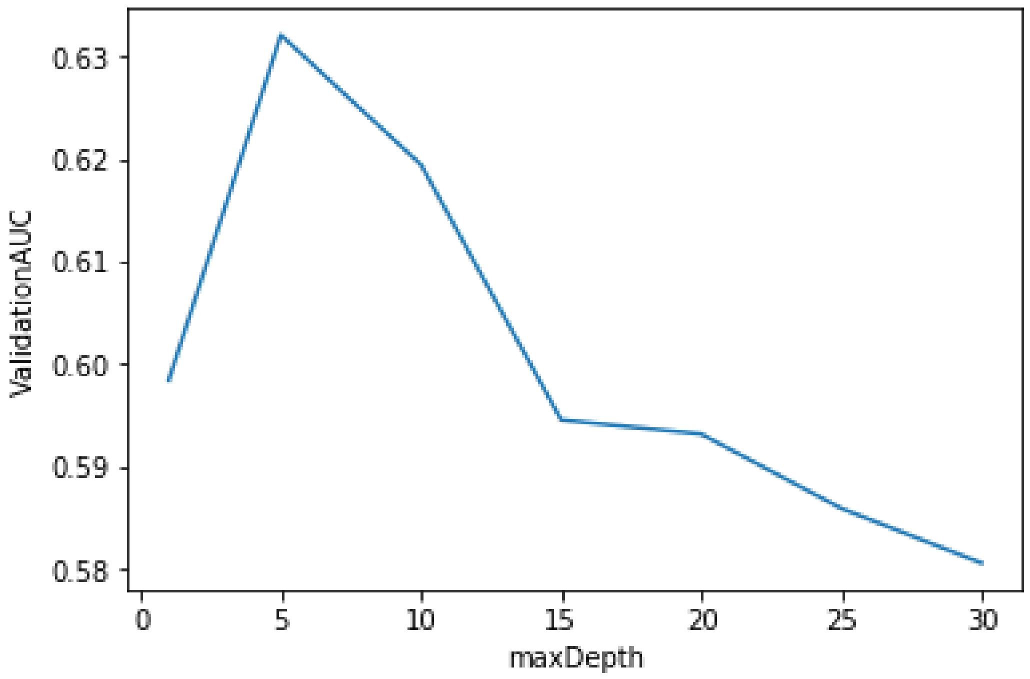
4 Problem4

4.1

```
In [24]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         maxdepthcollection=[1,5,10,15,20,25,30]
         TrainingAUC=[]
         ValidationAUC=[]
         for x in maxdepthcollection:
```

```
    learner = ml.dtree.treeClassify(XtS,Yt,maxDepth=x)
    TrainingAUC.append(learner.auc(XtS,Yt))
    ValidationAUC.append(learner.auc(XvS,Yv))
plt.plot(maxdepthcollection,TrainingAUC)
plt.xlabel('maxDepth')
plt.ylabel('TrainingAUC')
plt.show()
plt.plot(maxdepthcollection,ValidationAUC)
plt.xlabel('maxDepth')
plt.ylabel('ValidationAUC')
plt.show()
```

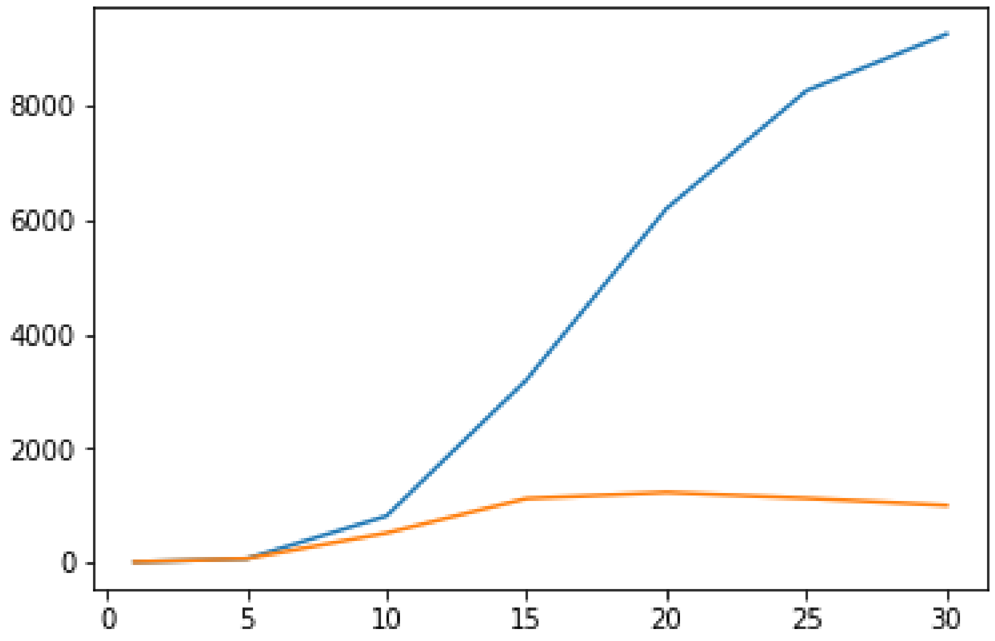




4.2

```
In [25]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)
         maxdepthcollection=[1,5,10,15,20,25,30]
         nodecollection=[]
         nodecollection2=[]
         count=2
         for x in maxdepthcollection:
             learner = ml.dtree.treeClassify(XtS,Yt,maxDepth=x)
             nodecollection.append(learner.sz)
             learner1=ml.dtree.treeClassify(XtS,Yt,maxDepth=x,minLeaf=count)
             nodecollection2.append(learner1.sz)
             count=count+1
```

```
plt.plot(maxdepthcollection,nodecollection)
plt.plot(maxdepthcollection,nodecollection2)
plt.show()
```

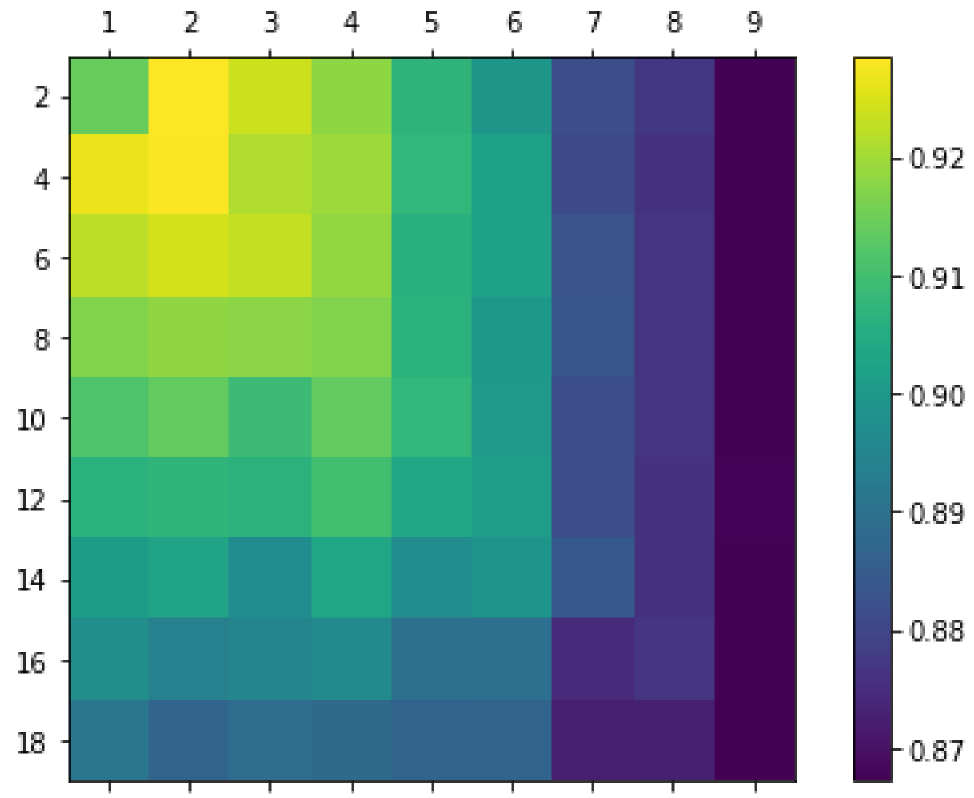


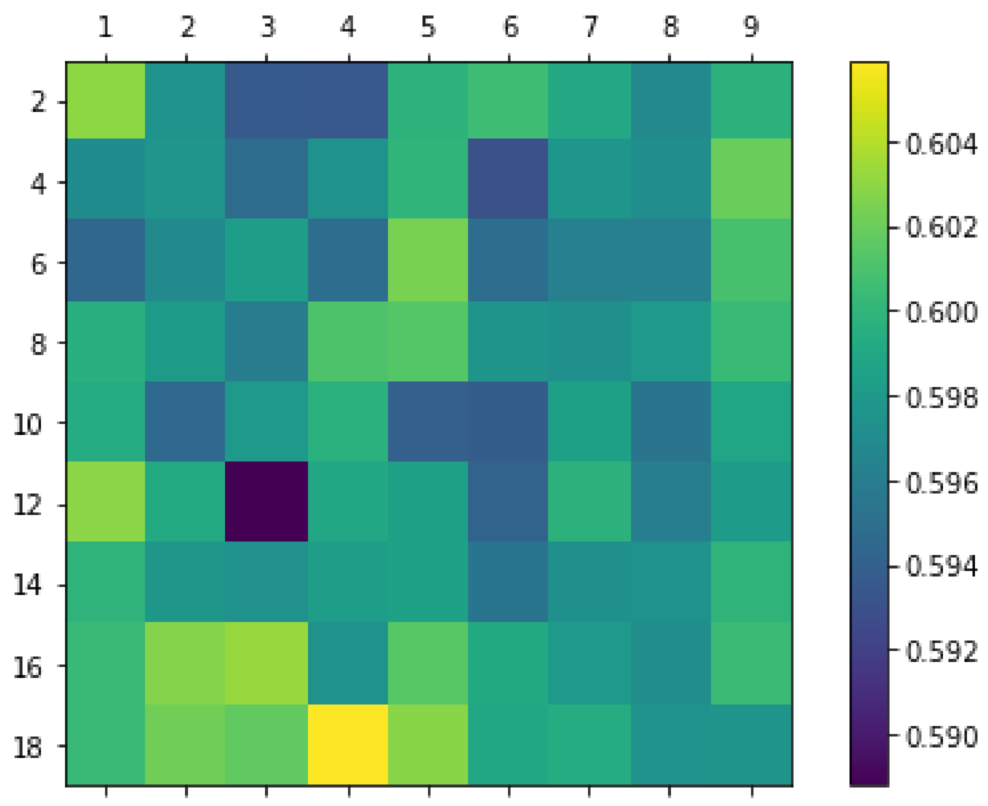
4.3

```
In [27]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)

         Par = range(2,20,2) # Or something else
         L = range(1,10,1) # Or something else
         tr_auc = np.zeros((len(Par),len(L)))
         va_auc = np.zeros((len(Par),len(L)))
         for i,k in enumerate(Par):
             for j,a in enumerate(L):
```

```
        learner = ml.dtree.treeClassify(XtS, Yt, maxDepth=15,minParent=k,minLeaf=a)
        tr_auc[i][j] = learner.auc(XtS,Yt) # train learner using k and a
        va_auc[i][j] = learner.auc(XvS,Yv)
f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax)
ax.set_xticklabels(['']+list(L))
ax.set_yticklabels(['']+list(Par))
plt.show()
ff, axx = plt.subplots(1, 1, figsize=(8, 5))
caxx = axx.matshow(va_auc, interpolation='nearest')
ff.colorbar(caxx)
axx.set_xticklabels(['']+list(L))
axx.set_yticklabels(['']+list(Par))
plt.show()
```





```
In [ ]: I would recommand minParent=18 and minLeaf=4
```

5 Problem5

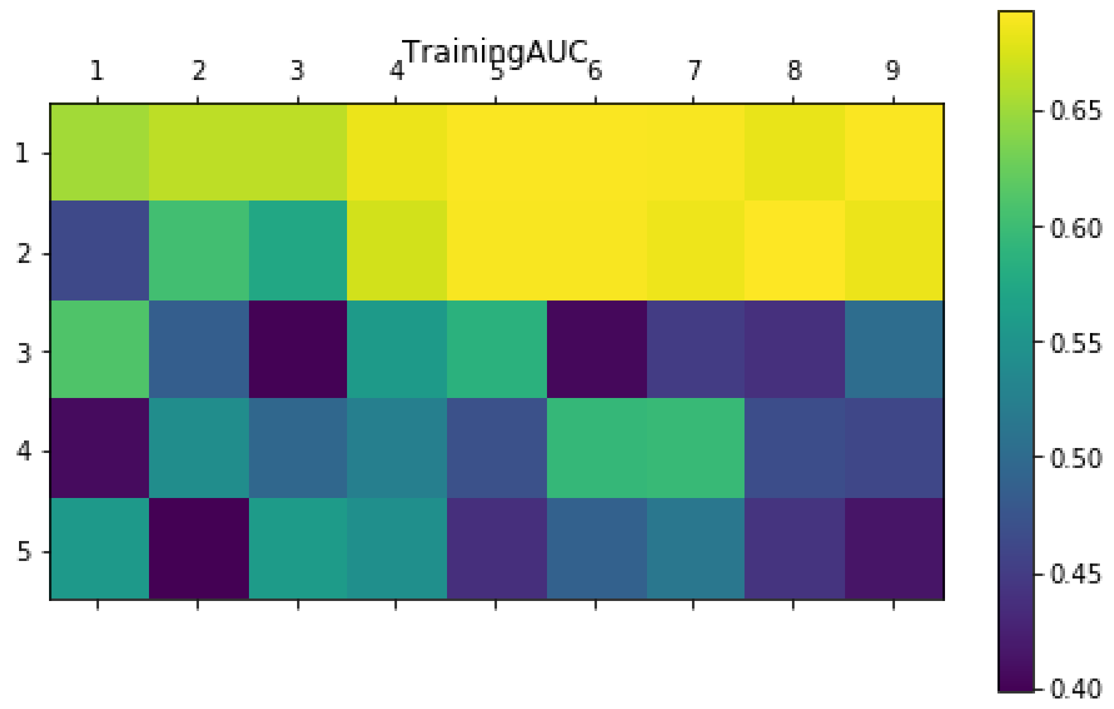
5.1

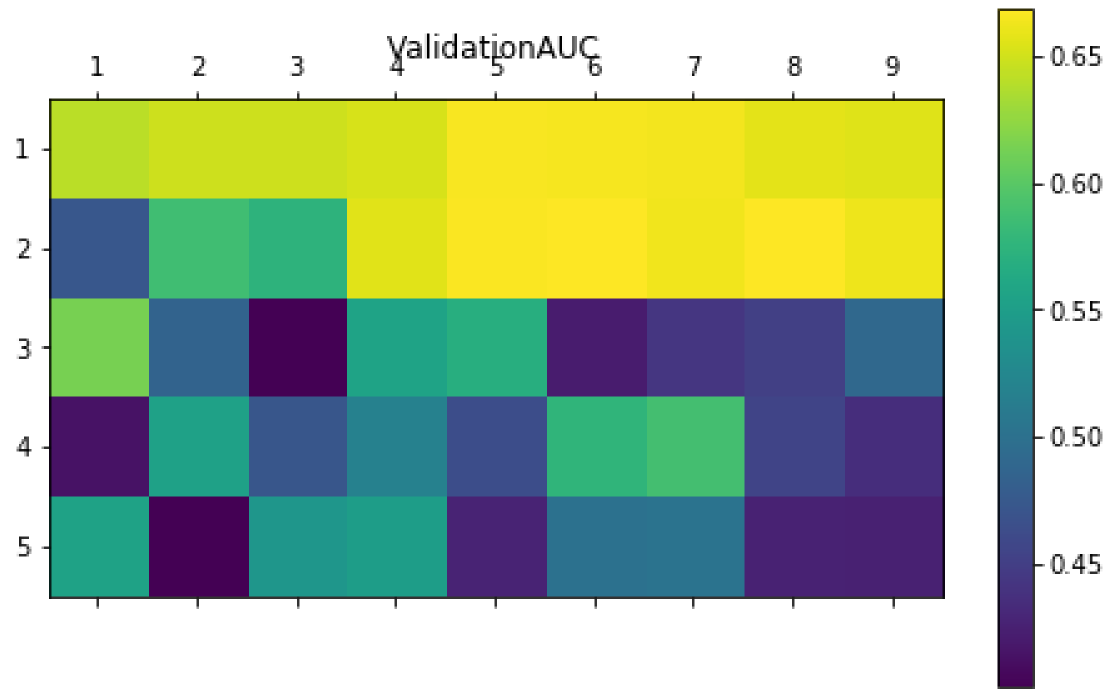
```
In [31]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)

         nn = ml.nnet.nnetClassify()
```

```
L=range(1,6,1)
Nodes = range(1,10,1) # Or something else
tr_auc = np.zeros((len(L),len(Nodes)))
va_auc = np.zeros((len(L),len(Nodes)))
for j,n in enumerate(Nodes):
    nn.init_weights([XtS.shape[1], n,2], 'random', XtS, Yt) # as many layers nodes yo
    nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
    tr_auc[0][j] = nn.auc(XtS,Yt) # train learner using k and a
    va_auc[0][j] = nn.auc(XvS,Yv)
for j,n in enumerate(Nodes):
    nn.init_weights([XtS.shape[1], n,n,2], 'random', XtS, Yt) # as many layers nodes
    nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
    tr_auc[1][j] = nn.auc(XtS,Yt) # train learner using k and a
    va_auc[1][j] = nn.auc(XvS,Yv)
for j,n in enumerate(Nodes):
    nn.init_weights([XtS.shape[1], n,n,n,2], 'random', XtS, Yt) # as many layers node
    nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
    tr_auc[2][j] = nn.auc(XtS,Yt) # train learner using k and a
    va_auc[2][j] = nn.auc(XvS,Yv)
for j,n in enumerate(Nodes):
    nn.init_weights([XtS.shape[1], n,n,n,n,2], 'random', XtS, Yt) # as many layers no
    nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
    tr_auc[3][j] = nn.auc(XtS,Yt) # train learner using k and a
    va_auc[3][j] = nn.auc(XvS,Yv)
for j,n in enumerate(Nodes):
    nn.init_weights([XtS.shape[1], n,n,n,n,n,2], 'random', XtS, Yt) # as many layers
    nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
    tr_auc[4][j] = nn.auc(XtS,Yt)
    va_auc[4][j] = nn.auc(XvS,Yv)
f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax)
ax.set_xticklabels(['']+list(Nodes))
ax.set_yticklabels(['']+list(L))
ax.set_title('TrainingAUC')
plt.show()
ff, axx = plt.subplots(1, 1, figsize=(8, 5))
caxx = axx.matshow(va_auc, interpolation='nearest')
ff.colorbar(caxx)
axx.set_xticklabels(['']+list(Nodes))
axx.set_yticklabels(['']+list(L))
axx.set_title('ValidationAUC')
plt.show()
```

```
it 1 : Jsur = 0.4233348660968459, J01 = 0.3326
it 2 : Jsur = 0.42126080147075673, J01 = 0.3326
it 4 : Jsur = 0.41988859102482273, J01 = 0.3326
it 8 : Jsur = 0.4192004229557035, J01 = 0.3326
it 16 : Jsur = 0.4194798277034388, J01 = 0.3326
it 32 : Jsur = 0.41998947746286175, J01 = 0.3326
it 64 : Jsur = 0.420647286054375, J01 = 0.3326
it 128 : Jsur = 0.421345850901122, J01 = 0.3326
it 256 : Jsur = 0.42178486371340684, J01 = 0.3326
it 1 : Jsur = 0.419013941180813, J01 = 0.3326
it 2 : Jsur = 0.41549126848415635, J01 = 0.3326
it 4 : Jsur = 0.41273294169129665, J01 = 0.3326
it 8 : Jsur = 0.41172466498381144, J01 = 0.3326
it 16 : Jsur = 0.41355409916758024, J01 = 0.3326
it 32 : Jsur = 0.4144123564560883, J01 = 0.3326
it 64 : Jsur = 0.4155810092434125, J01 = 0.3326
it 128 : Jsur = 0.4166447553521401, J01 = 0.3326
it 256 : Jsur = 0.41733046625588394, J01 = 0.3326
it 1 : Jsur = 0.42202111265207987, J01 = 0.3326
it 2 : Jsur = 0.4152906121618487, J01 = 0.3326
it 4 : Jsur = 0.4094528355506526, J01 = 0.3076
it 8 : Jsur = 0.4063973458198177, J01 = 0.3006
it 16 : Jsur = 0.406087907809274, J01 = 0.3002
it 32 : Jsur = 0.407195629606001, J01 = 0.3
it 64 : Jsur = 0.4085635959811339, J01 = 0.3012
it 128 : Jsur = 0.40929689329589464, J01 = 0.3036
it 256 : Jsur = 0.41001476246298235, J01 = 0.3054
it 1 : Jsur = 0.4193047457443165, J01 = 0.3326
it 2 : Jsur = 0.41241050068280566, J01 = 0.3326
it 4 : Jsur = 0.4099215192008522, J01 = 0.3104
it 8 : Jsur = 0.40668564429896054, J01 = 0.3008
it 16 : Jsur = 0.40364265251882425, J01 = 0.2992
it 32 : Jsur = 0.402004342610595, J01 = 0.3012
it 64 : Jsur = 0.40108681403496055, J01 = 0.3006
it 128 : Jsur = 0.4002600478177367, J01 = 0.2998
it 256 : Jsur = 0.3994715928531223, J01 = 0.299
it 1 : Jsur = 0.4183289495647681, J01 = 0.3326
it 2 : Jsur = 0.41202797646523986, J01 = 0.3328
it 4 : Jsur = 0.4081925816294097, J01 = 0.3026
it 8 : Jsur = 0.40503882376730604, J01 = 0.2998
it 16 : Jsur = 0.4024906495572808, J01 = 0.3008
it 32 : Jsur = 0.4006159310804005, J01 = 0.3004
it 64 : Jsur = 0.3996057566842036, J01 = 0.2984
it 128 : Jsur = 0.3975247297287547, J01 = 0.2944
it 256 : Jsur = 0.39416825344450673, J01 = 0.288
it 1 : Jsur = 0.42021986989415083, J01 = 0.3326
it 2 : Jsur = 0.4121816263400037, J01 = 0.3326
it 4 : Jsur = 0.40895518624453137, J01 = 0.3072
it 8 : Jsur = 0.40595618512999365, J01 = 0.3006
it 16 : Jsur = 0.40342422519694676, J01 = 0.3008
it 32 : Jsur = 0.40162270945384926, J01 = 0.2982
it 64 : Jsur = 0.3999056723599175, J01 = 0.2992
it 128 : Jsur = 0.39756039598232484, J01 = 0.2964
it 256 : Jsur = 0.39439356447258195, J01 = 0.2892
it 1 : Jsur = 0.419782954845811, J01 = 0.3326
it 2 : Jsur = 0.41044987413314266, J01 = 0.3256
it 4 : Jsur = 0.40761940315696193, J01 = 0.3046
it 8 : Jsur = 0.40557683859284466, J01 = 0.3018
it 16 : Jsur = 0.40328771521527623, J01 = 0.3024
it 32 : Jsur = 0.4017308126462046, J01 = 0.3018
it 64 : Jsur = 0.4005726012042026, J01 = 0.303
it 128 : Jsur = 0.39920524522688505, J01 = 0.3008
it 256 : Jsur = 0.39530995378196093, J01 = 0.2902
it 1 : Jsur = 0.4172058367389939, J01 = 0.3326
it 2 : Jsur = 0.4089387375345013, J01 = 0.3114
it 4 : Jsur = 0.4056057108647182, J01 = 0.3052
it 8 : Jsur = 0.40344682762158646, J01 = 0.3028
it 16 : Jsur = 0.4019762073775481, J01 = 0.3032
it 32 : Jsur = 0.40106832360808814, J01 = 0.303
it 64 : Jsur = 0.4005994255440285, J01 = 0.3004
it 128 : Jsur = 0.40025010710532094, J01 = 0.2984
it 256 : Jsur = 0.3997671590448177, J01 = 0.3006
it 1 : Jsur = 0.4207188464685596, J01 = 0.3326
it 2 : Jsur = 0.41163360177396335, J01 = 0.3326
it 4 : Jsur = 0.40794540133548673, J01 = 0.3058
it 8 : Jsur = 0.405415244271736, J01 = 0.3002
it 16 : Jsur = 0.4027630210742024, J01 = 0.3018
it 32 : Jsur = 0.4012287573440922, J01 = 0.3006
it 64 : Jsur = 0.40003566739506097, J01 = 0.2994
it 128 : Jsur = 0.39777070271567183, J01 = 0.2958
it 256 : Jsur = 0.3952781845688374, J01 = 0.2914
it 1 : Jsur = 0.4448051729668963, J01 = 0.3326
it 2 : Jsur = 0.4449934412905066, J01 = 0.3326
it 4 : Jsur = 0.44487541361335114, J01 = 0.3326
it 8 : Jsur = 0.44431864801961485, J01 = 0.3326
it 16 : Jsur = 0.44401377943550197, J01 = 0.3326
it 32 : Jsur = 0.4439599463556604, J01 = 0.3326
it 64 : Jsur = 0.4439549062994225, J01 = 0.3326
it 1 : Jsur = 0.444805172681881, J01 = 0.3326
it 2 : Jsur = 0.44499344065696633, J01 = 0.3326
it 4 : Jsur = 0.4448754128388311, J01 = 0.3326
it 8 : Jsur = 0.44431864696932, J01 = 0.3326
it 16 : Jsur = 0.44401377648156976, J01 = 0.3326
it 32 : Jsur = 0.44395993958481184, J01 = 0.3326
it 64 : Jsur = 0.4439548908367099, J01 = 0.3326
it 1 : Jsur = 0.44480517271710723, J01 = 0.3326
it 2 : Jsur = 0.44499344080032444, J01 = 0.3326
it 4 : Jsur = 0.44487541297658534, J01 = 0.3326
it 8 : Jsur = 0.44431864726501136, J01 = 0.3326
it 16 : Jsur = 0.4440137773543157, J01 = 0.3326
it 32 : Jsur = 0.44395994174809483, J01 = 0.3326
it 64 : Jsur = 0.44395489654802184, J01 = 0.3326
it 1 : Jsur = 0.4448051719412546, J01 = 0.3326
it 2 : Jsur = 0.44499343953384723, J01 = 0.3326
it 4 : Jsur = 0.4448754113558445, J01 = 0.3326
it 8 : Jsur = 0.444318643958215, J01 = 0.3326
it 16 : Jsur = 0.4440137650751875, J01 = 0.3326
it 32 : Jsur = 0.4439598965029745, J01 = 0.3326
it 64 : Jsur = 0.44395464499436366, J01 = 0.3326
it 128 : Jsur = 0.44394423125959775, J01 = 0.3326
it 256 : Jsur = 0.403805805797428, J01 = 0.2992
it 1 : Jsur = 0.4448051674800585, J01 = 0.3326
it 2 : Jsur = 0.444993432528206, J01 = 0.3326
it 4 : Jsur = 0.4448754005472486, J01 = 0.3326
it 8 : Jsur = 0.4443186200574859, J01 = 0.3326
it 16 : Jsur = 0.444013618070697, J01 = 0.3326
it 32 : Jsur = 0.44395710065670657, J01 = 0.3326
it 64 : Jsur = 0.40506888525893403, J01 = 0.299
it 128 : Jsur = 0.4007874640675018, J01 = 0.2956
it 256 : Jsur = 0.3943978715525242, J01 = 0.2854
it 1 : Jsur = 0.44480515885208716, J01 = 0.3326
it 2 : Jsur = 0.4449934140902504, J01 = 0.3326
it 4 : Jsur = 0.44487536137765127, J01 = 0.3326
it 8 : Jsur = 0.44431841987364096, J01 = 0.3326
it 16 : Jsur = 0.44400346666756224, J01 = 0.3326
it 32 : Jsur = 0.40484525181339603, J01 = 0.3
it 64 : Jsur = 0.4011161221667335, J01 = 0.2978
it 128 : Jsur = 0.39601731459846207, J01 = 0.289
it 256 : Jsur = 0.3939780638204282, J01 = 0.2876
it 1 : Jsur = 0.4448051698092011, J01 = 0.3326
it 2 : Jsur = 0.44499343642653266, J01 = 0.3326
it 4 : Jsur = 0.4448754067830992, J01 = 0.3326
it 8 : Jsur = 0.44431863437921604, J01 = 0.3326
it 16 : Jsur = 0.4440137190970601, J01 = 0.3326
it 32 : Jsur = 0.44395955760651534, J01 = 0.3326
it 64 : Jsur = 0.44390740293032266, J01 = 0.3326
it 128 : Jsur = 0.4036865152097365, J01 = 0.2996
it 256 : Jsur = 0.3975917218857836, J01 = 0.2926
it 1 : Jsur = 0.4448051500029999, J01 = 0.3326
it 2 : Jsur = 0.4449933840221318, J01 = 0.3326
it 4 : Jsur = 0.4448752604265854, J01 = 0.3326
it 8 : Jsur = 0.44431695114493086, J01 = 0.3326
it 16 : Jsur = 0.4070848694884558, J01 = 0.3038
it 32 : Jsur = 0.40324885356826634, J01 = 0.3
it 64 : Jsur = 0.3992371192591672, J01 = 0.297
it 128 : Jsur = 0.39447397647916416, J01 = 0.2892
it 256 : Jsur = 0.3923499735529565, J01 = 0.2842
it 1 : Jsur = 0.44480516905534023, J01 = 0.3326
it 2 : Jsur = 0.44499343632581934, J01 = 0.3326
it 4 : Jsur = 0.44487540651798263, J01 = 0.3326
it 8 : Jsur = 0.444318631588843, J01 = 0.3326
it 16 : Jsur = 0.44401369787208916, J01 = 0.3326
it 32 : Jsur = 0.4439592497828262, J01 = 0.3326
it 64 : Jsur = 0.43697847307766685, J01 = 0.3326
it 128 : Jsur = 0.4031881199502575, J01 = 0.3014
it 256 : Jsur = 0.3980128754555316, J01 = 0.2914
it 1 : Jsur = 0.4448051729208339, J01 = 0.3326
it 2 : Jsur = 0.4449934412943254, J01 = 0.3326
it 4 : Jsur = 0.44487541360559224, J01 = 0.3326
it 8 : Jsur = 0.44431864801883586, J01 = 0.3326
it 16 : Jsur = 0.44401377944124043, J01 = 0.3326
it 32 : Jsur = 0.44395994635802344, J01 = 0.3326
it 64 : Jsur = 0.4439549063012511, J01 = 0.3326
it 1 : Jsur = 0.4448051729350663, J01 = 0.3326
it 2 : Jsur = 0.4449934413297464, J01 = 0.3326
it 4 : Jsur = 0.44487541362373556, J01 = 0.3326
it 8 : Jsur = 0.4443186480128457, J01 = 0.3326
it 16 : Jsur = 0.4440137794422255, J01 = 0.3326
it 32 : Jsur = 0.4439599463582782, J01 = 0.3326
it 64 : Jsur = 0.4439549063011398, J01 = 0.3326
it 1 : Jsur = 0.4448051727941506, J01 = 0.3326
it 2 : Jsur = 0.4449934412751397, J01 = 0.3326
it 4 : Jsur = 0.4448754136130734, J01 = 0.3326
it 8 : Jsur = 0.44431864802192067, J01 = 0.3326
it 16 : Jsur = 0.4440137794375194, J01 = 0.3326
it 32 : Jsur = 0.4439599463571798, J01 = 0.3326
it 64 : Jsur = 0.44395490630103085, J01 = 0.3326
it 1 : Jsur = 0.44480517293345845, J01 = 0.3326
it 2 : Jsur = 0.4449934413605096, J01 = 0.3326
it 4 : Jsur = 0.44487541362656885, J01 = 0.3326
it 8 : Jsur = 0.44431864800716175, J01 = 0.3326
it 16 : Jsur = 0.4440137794486707, J01 = 0.3326
it 32 : Jsur = 0.44395994636021535, J01 = 0.3326
it 64 : Jsur = 0.44395490630198764, J01 = 0.3326
it 1 : Jsur = 0.4448051726823398, J01 = 0.3326
it 2 : Jsur = 0.44499344132484314, J01 = 0.3326
it 4 : Jsur = 0.4448754136358721, J01 = 0.3326
it 8 : Jsur = 0.44431864801299514, J01 = 0.3326
it 16 : Jsur = 0.444013779443629, J01 = 0.3326
it 32 : Jsur = 0.44395994635920577, J01 = 0.3326
it 64 : Jsur = 0.4439549063018909, J01 = 0.3326
it 1 : Jsur = 0.44480517265139297, J01 = 0.3326
it 2 : Jsur = 0.4449934412803752, J01 = 0.3326
it 4 : Jsur = 0.4448754136224673, J01 = 0.3326
it 8 : Jsur = 0.4443186480190054, J01 = 0.3326
it 16 : Jsur = 0.4440137794376677, J01 = 0.3326
it 32 : Jsur = 0.44395994635658337, J01 = 0.3326
it 64 : Jsur = 0.44395490630014467, J01 = 0.3326
it 1 : Jsur = 0.4448051725671805, J01 = 0.3326
it 2 : Jsur = 0.44499344131781016, J01 = 0.3326
it 4 : Jsur = 0.44487541363640787, J01 = 0.3326
it 8 : Jsur = 0.4443186480109618, J01 = 0.3326
it 16 : Jsur = 0.444013779442478, J01 = 0.3326
it 32 : Jsur = 0.4439599463575624, J01 = 0.3326
it 64 : Jsur = 0.4439549063000031, J01 = 0.3326
it 1 : Jsur = 0.4448051727595348, J01 = 0.3326
it 2 : Jsur = 0.444993441327803, J01 = 0.3326
it 4 : Jsur = 0.44487541361644, J01 = 0.3326
it 8 : Jsur = 0.4443186480090304, J01 = 0.3326
it 16 : Jsur = 0.44401377944583953, J01 = 0.3326
it 32 : Jsur = 0.44395994635759406, J01 = 0.3326
it 64 : Jsur = 0.4439549062993352, J01 = 0.3326
it 1 : Jsur = 0.4448051727185598, J01 = 0.3326
it 2 : Jsur = 0.44499344129104995, J01 = 0.3326
it 4 : Jsur = 0.44487541362102184, J01 = 0.3326
it 8 : Jsur = 0.44431864801776266, J01 = 0.3326
it 16 : Jsur = 0.4440137794390365, J01 = 0.3326
it 32 : Jsur = 0.44395994635653335, J01 = 0.3326
it 64 : Jsur = 0.4439549062996174, J01 = 0.3326
it 1 : Jsur = 0.44480517297570576, J01 = 0.3326
it 2 : Jsur = 0.4449934412939476, J01 = 0.3326
it 4 : Jsur = 0.4448754136120782, J01 = 0.3326
it 8 : Jsur = 0.44431864802023135, J01 = 0.3326
it 16 : Jsur = 0.444013779437932, J01 = 0.3326
it 32 : Jsur = 0.44395994635736613, J01 = 0.3326
it 64 : Jsur = 0.4439549063011374, J01 = 0.3326
it 1 : Jsur = 0.44480517296311955, J01 = 0.3326
it 2 : Jsur = 0.444993441289542, J01 = 0.3326
it 4 : Jsur = 0.44487541360988997, J01 = 0.3326
it 8 : Jsur = 0.4443186480209284, J01 = 0.3326
it 16 : Jsur = 0.4440137794379888, J01 = 0.3326
it 32 : Jsur = 0.4439599463573533, J01 = 0.3326
it 64 : Jsur = 0.4439549063011364, J01 = 0.3326
it 1 : Jsur = 0.4448051729342648, J01 = 0.3326
it 2 : Jsur = 0.44499344130747165, J01 = 0.3326
it 4 : Jsur = 0.44487541361807814, J01 = 0.3326
it 8 : Jsur = 0.44431864801739396, J01 = 0.3326
it 16 : Jsur = 0.44401377943966197, J01 = 0.3326
it 32 : Jsur = 0.44395994635776004, J01 = 0.3326
it 64 : Jsur = 0.44395490630116763, J01 = 0.3326
it 1 : Jsur = 0.44480517245403933, J01 = 0.3326
it 2 : Jsur = 0.4449934412605932, J01 = 0.3326
it 4 : Jsur = 0.44487541362195654, J01 = 0.3326
it 8 : Jsur = 0.44431864802200843, J01 = 0.3326
it 16 : Jsur = 0.4440137794389394, J01 = 0.3326
it 32 : Jsur = 0.4439599463575476, J01 = 0.3326
it 64 : Jsur = 0.44395490630115125, J01 = 0.3326
it 1 : Jsur = 0.4448051728325665, J01 = 0.3326
it 2 : Jsur = 0.4449934413225541, J01 = 0.3326
it 4 : Jsur = 0.444875413624189, J01 = 0.3326
it 8 : Jsur = 0.4443186480131879, J01 = 0.3326
it 16 : Jsur = 0.4440137794429138, J01 = 0.3326
it 32 : Jsur = 0.4439599463584562, J01 = 0.3326
it 64 : Jsur = 0.44395490630122003, J01 = 0.3326
it 1 : Jsur = 0.4448051727473142, J01 = 0.3326
it 2 : Jsur = 0.4449934413493669, J01 = 0.3326
it 4 : Jsur = 0.4448754136360888, J01 = 0.3326
it 8 : Jsur = 0.44431864800743076, J01 = 0.3326
it 16 : Jsur = 0.4440137794463113, J01 = 0.3326
it 32 : Jsur = 0.4439599463592442, J01 = 0.3326
it 64 : Jsur = 0.4439549063012914, J01 = 0.3326
it 1 : Jsur = 0.44480517286675647, J01 = 0.3326
it 2 : Jsur = 0.44499344129859236, J01 = 0.3326
it 4 : Jsur = 0.44487541361656296, J01 = 0.3326
it 8 : Jsur = 0.4443186480184042, J01 = 0.3326
it 16 : Jsur = 0.4440137794398249, J01 = 0.3326
it 32 : Jsur = 0.44395994635779296, J01 = 0.3326
it 64 : Jsur = 0.44395490630120377, J01 = 0.3326
it 1 : Jsur = 0.4448051726331296, J01 = 0.3326
it 2 : Jsur = 0.444993441341823, J01 = 0.3326
it 4 : Jsur = 0.44487541362286576, J01 = 0.3326
it 8 : Jsur = 0.4443186480058293, J01 = 0.3326
it 16 : Jsur = 0.4440137794512214, J01 = 0.3326
it 32 : Jsur = 0.4439599463600902, J01 = 0.3326
it 64 : Jsur = 0.44395490630133255, J01 = 0.3326
it 1 : Jsur = 0.4448051728850087, J01 = 0.3326
it 2 : Jsur = 0.44499344136725666, J01 = 0.3326
it 4 : Jsur = 0.4448754136296473, J01 = 0.3326
it 8 : Jsur = 0.4443186480046673, J01 = 0.3326
it 16 : Jsur = 0.444013779449099, J01 = 0.3326
it 32 : Jsur = 0.4439599463597669, J01 = 0.3326
it 64 : Jsur = 0.44395490630131734, J01 = 0.3326
it 1 : Jsur = 0.44480517290879606, J01 = 0.3326
it 2 : Jsur = 0.44499344128839785, J01 = 0.3326
it 4 : Jsur = 0.4448754136155355, J01 = 0.3326
it 8 : Jsur = 0.4443186480209224, J01 = 0.3326
it 16 : Jsur = 0.44401377943713394, J01 = 0.3326
it 32 : Jsur = 0.44395994635722125, J01 = 0.3326
it 64 : Jsur = 0.44395490630112655, J01 = 0.3326
it 1 : Jsur = 0.44480517297400873, J01 = 0.3326
it 2 : Jsur = 0.4449934412888184, J01 = 0.3326
it 4 : Jsur = 0.4448754136122798, J01 = 0.3326
it 8 : Jsur = 0.44431864802150833, J01 = 0.3326
it 16 : Jsur = 0.44401377943690323, J01 = 0.3326
it 32 : Jsur = 0.44395994635715486, J01 = 0.3326
it 64 : Jsur = 0.4439549063011213, J01 = 0.3326
it 1 : Jsur = 0.4448051728902534, J01 = 0.3326
it 2 : Jsur = 0.44499344130974494, J01 = 0.3326
it 4 : Jsur = 0.4448754136254418, J01 = 0.3326
it 8 : Jsur = 0.44431864801738685, J01 = 0.3326
it 16 : Jsur = 0.44401377943880455, J01 = 0.3326
it 32 : Jsur = 0.44395994635763425, J01 = 0.3326
it 64 : Jsur = 0.44395490630115864, J01 = 0.3326
it 1 : Jsur = 0.4448051727964394, J01 = 0.3326
it 2 : Jsur = 0.4449934413183562, J01 = 0.3326
it 4 : Jsur = 0.44487541362784194, J01 = 0.3326
it 8 : Jsur = 0.44431864801428766, J01 = 0.3326
it 16 : Jsur = 0.444013779441648, J01 = 0.3326
it 32 : Jsur = 0.44395994635822034, J01 = 0.3326
it 64 : Jsur = 0.4439549063012031, J01 = 0.3326
it 1 : Jsur = 0.44480517291999294, J01 = 0.3326
it 2 : Jsur = 0.44499344128698504, J01 = 0.3326
it 4 : Jsur = 0.4448754136107141, J01 = 0.3326
it 8 : Jsur = 0.4443186480208361, J01 = 0.3326
it 16 : Jsur = 0.44401377943812204, J01 = 0.3326
it 32 : Jsur = 0.44395994635738223, J01 = 0.3326
it 64 : Jsur = 0.4439549063011384, J01 = 0.3326
it 1 : Jsur = 0.4448051726476422, J01 = 0.3326
it 2 : Jsur = 0.44499344131864066, J01 = 0.3326
it 4 : Jsur = 0.44487541363133987, J01 = 0.3326
it 8 : Jsur = 0.44431864801237225, J01 = 0.3326
it 16 : Jsur = 0.44401377944366177, J01 = 0.3326
it 32 : Jsur = 0.4439599463586363, J01 = 0.3326
it 64 : Jsur = 0.44395490630123496, J01 = 0.3326
it 1 : Jsur = 0.4448051726485414, J01 = 0.3326
it 2 : Jsur = 0.4449934413020781, J01 = 0.3326
it 4 : Jsur = 0.44487541362052424, J01 = 0.3326
it 8 : Jsur = 0.4443186480152778, J01 = 0.3326
it 16 : Jsur = 0.44401377944373327, J01 = 0.3326
it 32 : Jsur = 0.443959946358546, J01 = 0.3326
it 64 : Jsur = 0.443954906301227, J01 = 0.3326
it 1 : Jsur = 0.44480517277482184, J01 = 0.3326
it 2 : Jsur = 0.4449934413044778, J01 = 0.3326
it 4 : Jsur = 0.4448754136234492, J01 = 0.3326
it 8 : Jsur = 0.4443186480166156, J01 = 0.3326
it 16 : Jsur = 0.44401377944068443, J01 = 0.3326
it 32 : Jsur = 0.44395994635798086, J01 = 0.3326
it 64 : Jsur = 0.44395490630118456, J01 = 0.3326
it 1 : Jsur = 0.4448051725622321, J01 = 0.3326
it 2 : Jsur = 0.44499344127878565, J01 = 0.3326
it 4 : Jsur = 0.4448754136216205, J01 = 0.3326
it 8 : Jsur = 0.4443186480192485, J01 = 0.3326
it 16 : Jsur = 0.44401377944056164, J01 = 0.3326
it 32 : Jsur = 0.44395994635789604, J01 = 0.3326
it 64 : Jsur = 0.4439549063011777, J01 = 0.3326
```





```
In [ ]: I would recommand number of hidden layers=2 and nodes in each layer=8
```

5.2

```
In [33]: import numpy as np
         import matplotlib.pyplot as plt
         import mltools as ml
         import mltools.transforms as xform
         X = np.genfromtxt('X_train.txt', delimiter=None)
         Y = np.genfromtxt('Y_train.txt', delimiter=None)
         X,Y = ml.shuffleData(X,Y)
         Xtr, Xva, Ytr, Yva = ml.splitData(X, Y,0.75)
         Xt, Yt = Xtr[:5000], Ytr[:5000]
         Xv, Yv = Xva[:5000], Yva[:5000]
         XtS, params = ml.transforms.rescale(Xt)
         XvS, _ = ml.rescale(Xv, params)

         nn = ml.nnet.nnetClassify()
         tr_auc = np.zeros((len(L),len(Nodes)))
         va_auc = np.zeros((len(L),len(Nodes)))
         nn.init_weights([XtS.shape[1],8,8,2], 'random', XtS, Yt)
         sig = lambda z: np.atleast_2d(z)
         dsig = lambda z: np.atleast_2d(1)
         nn.setActivation('custom', sig, dsig)
```

```
        nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
        print("Custom activation TrianAUC:",nn.auc(XtS,Yt))
        print("Custom activation ValidateAUC:",nn.auc(XvS,Yv))

        nn = ml.nnet.nnetClassify()
        nn.init_weights([XtS.shape[1],8,8,2], 'random', XtS, Yt)
        nn.setActivation('Logistic', sig, dsig)
        nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
        print("Logistic activation TrianAUC:",nn.auc(XtS,Yt))
        print("Logistic activation ValidateAUC:",nn.auc(XvS,Yv))

        nn = ml.nnet.nnetClassify()
        nn.init_weights([XtS.shape[1],8,8,2], 'random', XtS, Yt)
        nn.setActivation('Htangent', sig, dsig)
        nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
        print("Htangent activation TrianAUC:",nn.auc(XtS,Yt))
        print("Htangent activation ValidateAUC:",nn.auc(XvS,Yv))
```

```
it 1 : Jsur = 0.447634786733653, J01 = 0.337
it 2 : Jsur = 0.44731095050400965, J01 = 0.337
it 4 : Jsur = 0.4473099859714809, J01 = 0.337
it 8 : Jsur = 0.4471946070460831, J01 = 0.337
it 16 : Jsur = 0.4154463436172536, J01 = 0.3162
it 32 : Jsur = 0.4128313242527168, J01 = 0.3116
it 64 : Jsur = 0.412225914322236, J01 = 0.3106
it 128 : Jsur = 0.4120357916805219, J01 = 0.3086
it 256 : Jsur = 0.41198342373228025, J01 = 0.3078
Custom activation TrianAUC: 0.660180816449
Custom activation ValidateAUC: 0.64560973568
it 1 : Jsur = 0.4300751793449095, J01 = 0.337
it 2 : Jsur = 0.4234401712812418, J01 = 0.337
it 4 : Jsur = 0.4254597887545122, J01 = 0.337
it 8 : Jsur = 0.426463039024082, J01 = 0.337
it 16 : Jsur = 0.4265948359740891, J01 = 0.337
it 32 : Jsur = 0.4265814499528768, J01 = 0.337
it 64 : Jsur = 0.4267160600035876, J01 = 0.337
it 128 : Jsur = 0.42695874536031636, J01 = 0.337
it 256 : Jsur = 0.42721318503143557, J01 = 0.337
Logistic activation TrianAUC: 0.646914349396
Logistic activation ValidateAUC: 0.633410605986
it 1 : Jsur = 0.44763472542821764, J01 = 0.337
it 2 : Jsur = 0.4473106459257506, J01 = 0.337
it 4 : Jsur = 0.44730325952541405, J01 = 0.337
it 8 : Jsur = 0.41696818017041104, J01 = 0.319
it 16 : Jsur = 0.4139650848324165, J01 = 0.3144
it 32 : Jsur = 0.41159729692536184, J01 = 0.3096
it 64 : Jsur = 0.40793527268535634, J01 = 0.308
it 128 : Jsur = 0.404105903055012, J01 = 0.3064
it 256 : Jsur = 0.40087454164889896, J01 = 0.303
Htangent activation TrianAUC: 0.687190676316
Htangent activation ValidateAUC: 0.66277164255
```

So our custom activation is better than logistic activation, but worse than Htangent.

6 Problem6

I prefer linear classifier to others because it will not easily overfit and using gradient descent we can get quite

good parameters which gives us accurate predictions. My username is KAI YE, the leaderboard score is 0.54683.

```
In [ ]: learner = ml.linearC.linearClassify()
        learner.train(XtS, Y, 4, initStep=1e-6, stopTol=1e-6, stopIter=100)
```

7 Statement of Collaboration

No disscussion with others.