273A hw4

EnYu Huang

February 2021

1 Problem 1

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
# Data Loading
# Problem 1.1
X = np.genfromtxt("data/X_train.txt", delimiter=None)
Y = np.genfromtxt("data/Y_train.txt", delimiter=None)
X,Y = ml.shuffleData(X,Y)
print("X:min",np.min(X,axis=0),"\nX:max",np.max(X,axis=0),"\nX:mean",np.
 ↪mean(X,axis=0),"\nX:var",np.var(X,axis=0))
```

```
X:min [ 1.9350e+02  1.5250e+02  2.1425e+02  1.5250e+02  1.0000e+01  0.0000e+00
  0.0000e+00  0.0000e+00  8.7589e-01  0.0000e+00  0.0000e+00  0.0000e+00
  9.9049e-01 -9.9990e+02]
X:max [2.5300e+02 2.4900e+02 2.5250e+02 2.5250e+02 3.1048e+04 1.3630e+04
 9.2380e+03 1.2517e+02 1.9167e+01 1.3230e+01 6.6761e+01 7.3902e+01
 9.7504e+02 7.9720e+02]
X:mean [2.41601104e+02 2.27376571e+02 2.41554150e+02 2.32826768e+02
 3.08992337e+03 9.28259020e+02 1.38093830e+02 3.24857933e+00
 6.49865290e+00 2.09713912e+00 4.21766041e+00 2.69171845e+00
 1.02715905e+01 5.78148050e+00]
X:var [8.34991711e+01 9.26255931e+01 3.52863398e+01 9.76257317e+01
 1.56515138e+07 3.08176182e+06 4.43951746e+05 8.21948502e+00
 6.40504819e+00 4.36344047e+00 4.08637188e+00 2.19877847e+00
 4.04646245e+02 3.40652055e+03]
```

[3]:
```
# Problem 1.2
Xtr, Xva, Ytr, Yva = ml.splitData(X, Y)
Xt, Yt = Xtr[:5000], Ytr[:5000] # subsample for efficiency (you can go higher)
Xv, Yv = Xva[:1000], Yva[:1000]
XtS, params = ml.rescale(Xt) # Normalize the features
XvS, _ = ml.rescale(Xva, params) # Normalize the features
XvS2, _ = ml.rescale(Xv, params) # Normalize the features
print("XtS:min",np.min(XtS,axis=0),"\nXtS:max",np.max(XtS,axis=0),"\nXtS:
 ↪mean",np.mean(XtS,axis=0),"\nXtS:var",np.var(XtS,axis=0))
```

```
print("\nXvS:min",np.min(XvS,axis=0),"\nXvS:max",np.max(XvS,axis=0),"\nXvS:
 ↪mean",np.mean(XvS,axis=0),"\nXvS:var",np.var(XvS,axis=0))
```

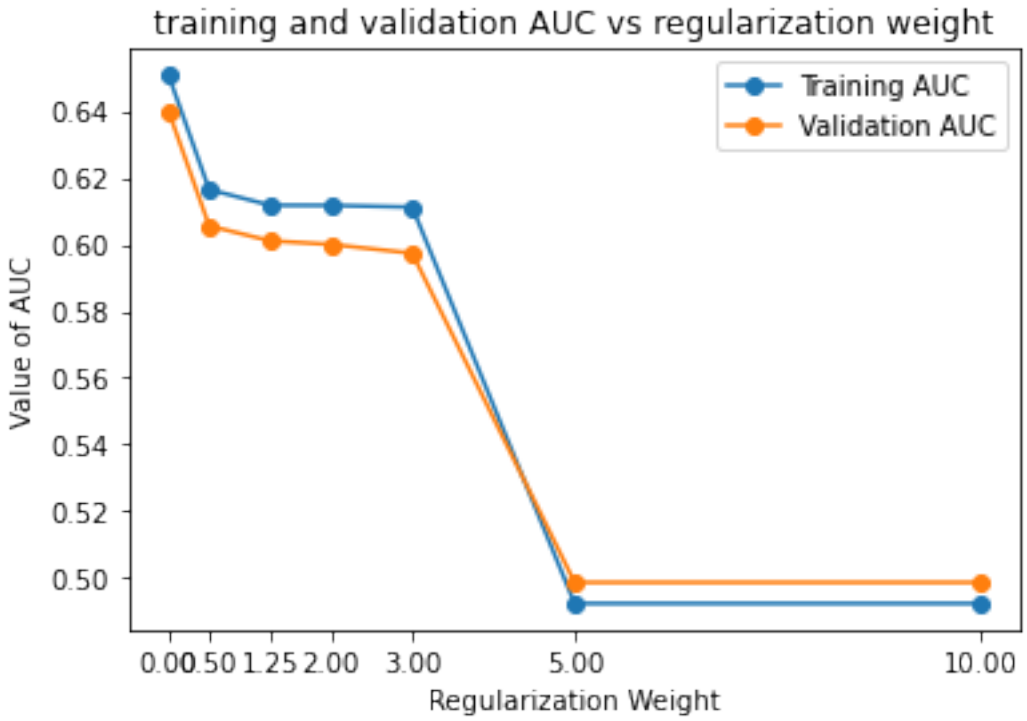
```
XtS:min [ -4.72280605  -3.84784723  -4.34345626  -2.73923756  -0.76829281
  -0.53923572  -0.206148    -1.14481842  -2.11668935  -1.01522092
  -2.13582502  -1.7661784   -0.57189294 -16.70363613]
XtS:max [ 1.23994475  1.82707843  1.76916928  1.93044359  6.93085914  7.23382123
 13.00760532  7.67334018  4.0978637   4.28606737  8.96455791 28.67675514
 29.58313841 11.02101073]
XtS:mean [-1.91133775e-15  1.39060985e-15  6.11330098e-14 -9.51856982e-14
  4.81448215e-17 -3.54188900e-16 -4.79616347e-17  1.87414528e-15
  2.55009347e-15 -1.14103171e-17  5.06645836e-15 -4.73915351e-15
  5.61106717e-17 -2.95213853e-16]
XtS:var [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

XvS:min [ -5.16449129  -3.84784723  -4.51758151  -2.83914516  -0.76829281
  -0.53923572  -0.206148    -1.14481842  -2.13729275  -1.01522092
  -2.13582502  -1.7661784   -0.57289662 -16.70363613]
XvS:max [ 1.23994475  2.23980029  1.80565267  1.95242326  6.93085914  7.23382123
 13.00760532  8.11017474  4.49292686  5.33193239 13.40876787 34.81779809
 61.29976031 13.01737148]
XvS:mean [-0.01822062  0.00768816  0.01081898 -0.00020199  0.00098404
-0.00494321
 -0.00615467 -0.01060382 -0.00484673 -0.00723998 -0.0052791  -0.01121187
  0.01396717  0.0153812 ]
XvS:var [1.0236639  0.98641064 0.97506915 0.9764151  0.96501925 1.01168552
 0.90342826 0.99091286 1.00083575 1.00589363 1.03621362 0.92368499
 1.38082821 0.92400357]
```

[3]:
```
# Problem 1.3
reg = [0.0,0.5,1.25,2.0,3.0,5.0,10.0]
train_auc = np.zeros(len(reg))
validation_auc = np.zeros(len(reg))

for i, _ in enumerate(reg):
    learner = ml.linearC.linearClassify()
    learner.train(XtS, Yt, reg=_, initStep=0.5, stopTol=1e-6, stopIter=100)
    train_auc[i] = learner.auc(XtS, Yt)
    validation_auc[i] = learner.auc(XvS, Yva)

plt.plot(reg, train_auc, marker='o', label = "Training AUC")
plt.plot(reg, validation_auc, marker='o', label = "Validation AUC")

plt.xticks(reg)
plt.xlabel("Regularization Weight")
plt.ylabel("Value of AUC")
plt.title("training and validation AUC vs regularization weight")
```

```
plt.legend()


print ('Training AUC:',train_auc)
print ('Validation AUC:', validation_auc)
```

```
C:\Users\enyu8\mltools\base.py:96: RuntimeWarning: divide by zero encountered in
log
  return - np.mean( np.log( P[ np.arange(M), Y ] ) ) # evaluate
C:\Users\enyu8\mltools\linearC.py:134: RuntimeWarning: invalid value encountered
in double_scalars
  done = (it > stopIter) or ( (it>1) and (abs(Jsur[-1]-Jsur[-2])<stopTol) )
C:\Users\enyu8\mltools\linearC.py:122: RuntimeWarning: overflow encountered in
exp
  sigx  = np.exp(respi) / (1.0+np.exp(respi))
C:\Users\enyu8\mltools\linearC.py:122: RuntimeWarning: invalid value encountered
in true_divide
  sigx  = np.exp(respi) / (1.0+np.exp(respi))
```

```
Training AUC: [0.65073311 0.61644254 0.61172562 0.61168289 0.61116957 0.49231409
 0.49231409]
Validation AUC: [0.63997378 0.60554686 0.60106006 0.59998769 0.59733441
0.49864048
 0.49864048]
```



[4]:
```
# Problem 1.4
degree = 2
XtrP,params = ml.transforms.rescale(ml.transforms.fpoly(Xt, degree, bias=False))
XteP,_ = ml.transforms.rescale(ml.transforms.fpoly(Xva,degree,bias=False),␣
 ↪params)

print ("Number of Features:", XtrP.shape)
```

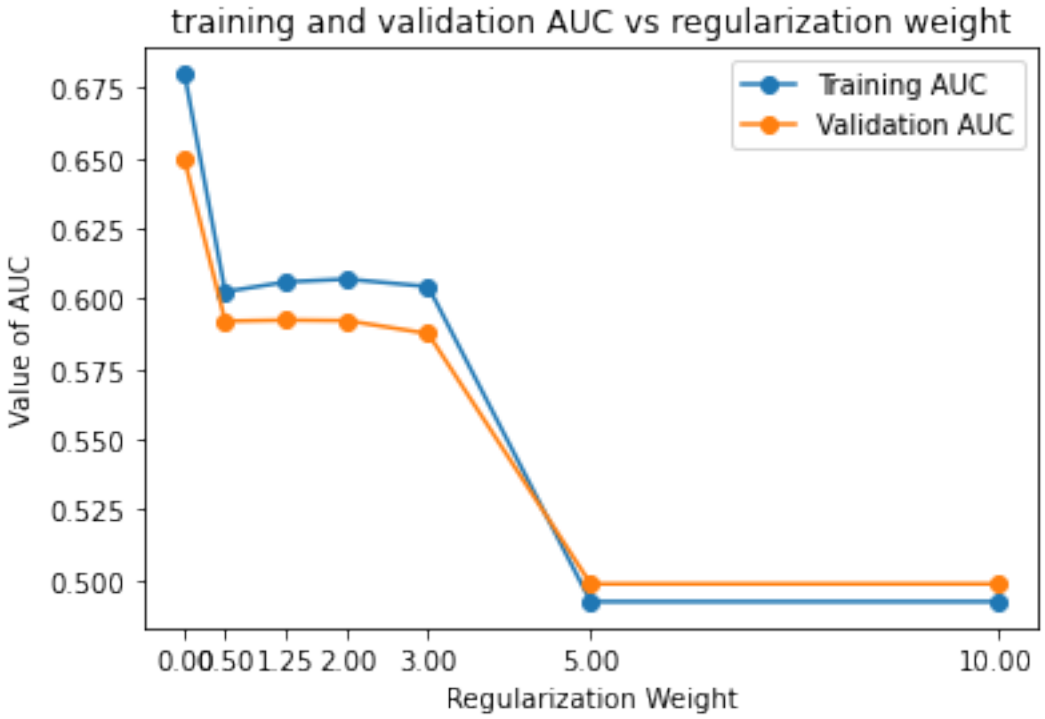
```
Number of Features: (5000, 119)
```

The number of features is : 1.original 14 features 2.14 features at degree 2 3.(14 * 13) / 2 = 91 The total features = 14 + 14 + 91 = 119

[5]:
```
# Problem 1.5
train_auc = np.zeros(len(reg))
validation_auc = np.zeros(len(reg))

for i, _ in enumerate(reg):
    learner = ml.linearC.linearClassify()
    learner.train(XtrP, Yt, reg=_, initStep=0.5, stopTol=1e-6, stopIter=100)
    train_auc[i] = learner.auc(XtrP, Yt)
    validation_auc[i] = learner.auc(XteP, Yva)

plt.plot(reg, train_auc, marker='o', label = "Training AUC")
plt.plot(reg, validation_auc, marker='o', label = "Validation AUC")
plt.xticks(reg)
plt.xlabel("Regularization Weight")
plt.ylabel("Value of AUC")
plt.title("training and validation AUC vs regularization weight")
plt.legend()
plt.show()

print ('Training AUC:',train_auc)
print ('Validation AUC:', validation_auc)
```

```
C:\Users\enyu8\mltools\linearC.py:82: RuntimeWarning: overflow encountered in
exp
  prob = np.exp(resp)
C:\Users\enyu8\mltools\linearC.py:84: RuntimeWarning: invalid value encountered
in true_divide
  prob /= prob + 1.0       # logistic transform (binary classification; C=1)
C:\Users\enyu8\mltools\base.py:96: RuntimeWarning: divide by zero encountered in
log
  return - np.mean( np.log( P[ np.arange(M), Y ] ) ) # evaluate
C:\Users\enyu8\mltools\linearC.py:134: RuntimeWarning: invalid value encountered
in double_scalars
  done = (it > stopIter) or ( (it>1) and (abs(Jsur[-1]-Jsur[-2])<stopTol) )
C:\Users\enyu8\mltools\linearC.py:122: RuntimeWarning: overflow encountered in
```

```
exp
  sigx  = np.exp(respi) / (1.0+np.exp(respi))
C:\Users\enyu8\mltools\linearC.py:122: RuntimeWarning: invalid value encountered
in true_divide
  sigx  = np.exp(respi) / (1.0+np.exp(respi))
```

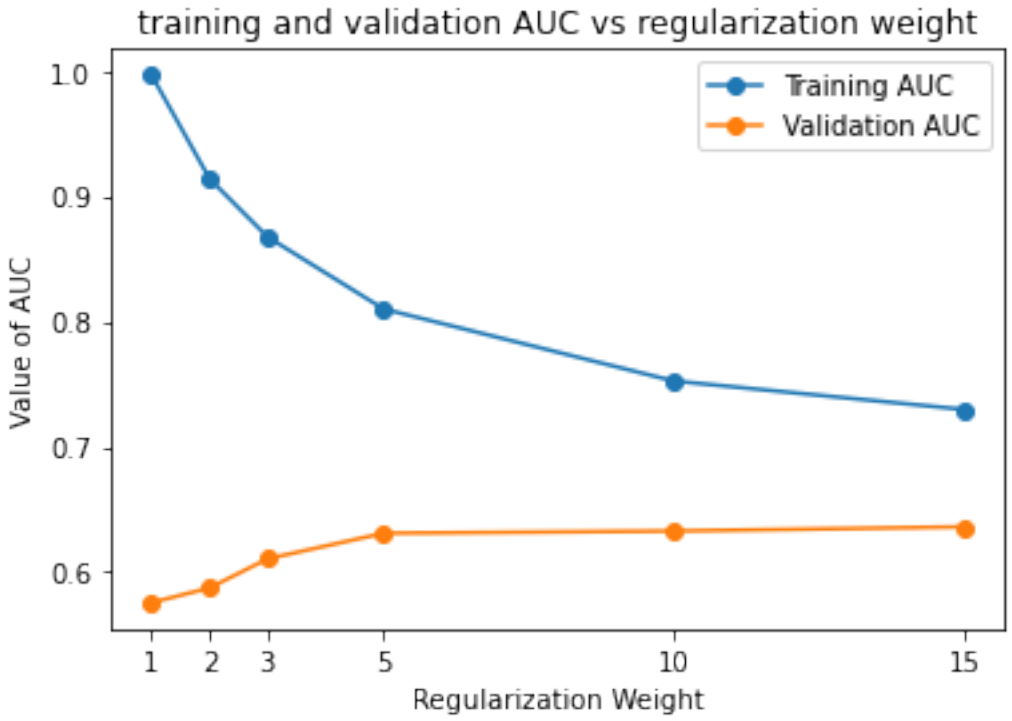



```
Training AUC: [0.67987071 0.60250813 0.60597388 0.60698948 0.60430875 0.49231409
 0.49231409]
Validation AUC: [0.64935148 0.59204991 0.5924052  0.59221903 0.58760309
0.49864048
 0.49864048]
```

2 Problem 2

[11]:
```
# Problem 2.1
K = [1,2,3,5,10,15]
train_auc = np.zeros(len(K))
validation_auc = np.zeros(len(K))
learner = ml.knn.knnClassify()
for i, k in enumerate(K):
    learner.train(XtS, Yt, K=k, alpha=0.0)
    train_auc[i] = learner.auc(XtS, Yt)
    validation_auc[i] = learner.auc(XvS2, Yv)
```

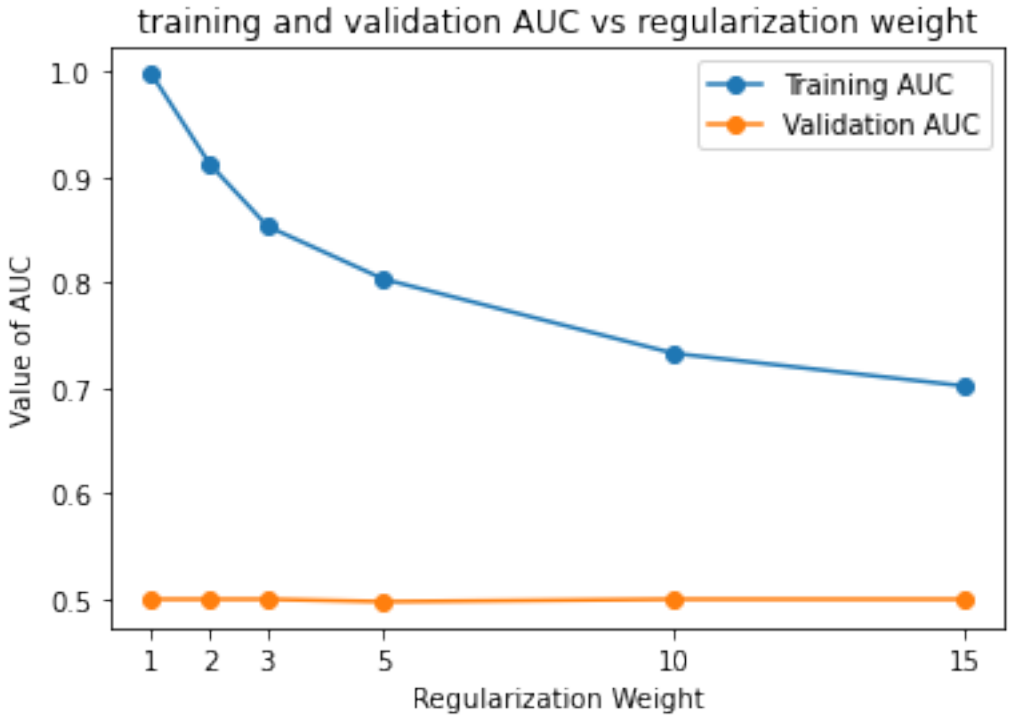
```
plt.plot(K, train_auc, marker='o', label = "Training AUC")
plt.plot(K, validation_auc, marker='o', label = "Validation AUC")
plt.xticks(K)
plt.xlabel("Regularization Weight")
plt.ylabel("Value of AUC")
plt.title("training and validation AUC vs regularization weight")
plt.legend()
plt.show()

print ('Training AUC:',train_auc)
print ('Validation AUC:', validation_auc)
```




```
Training AUC: [0.99674239 0.91399591 0.86758464 0.81004459 0.75258743
0.72986551]
Validation AUC: [0.57565441 0.58742795 0.61065457 0.6310695  0.63282543
0.63616829]
```

[12]:
```
# Problem 2.2
K = [1,2,3,5,10,15]
train_auc = np.zeros(len(K))
validation_auc = np.zeros(len(K))
learner = ml.knn.knnClassify()
```

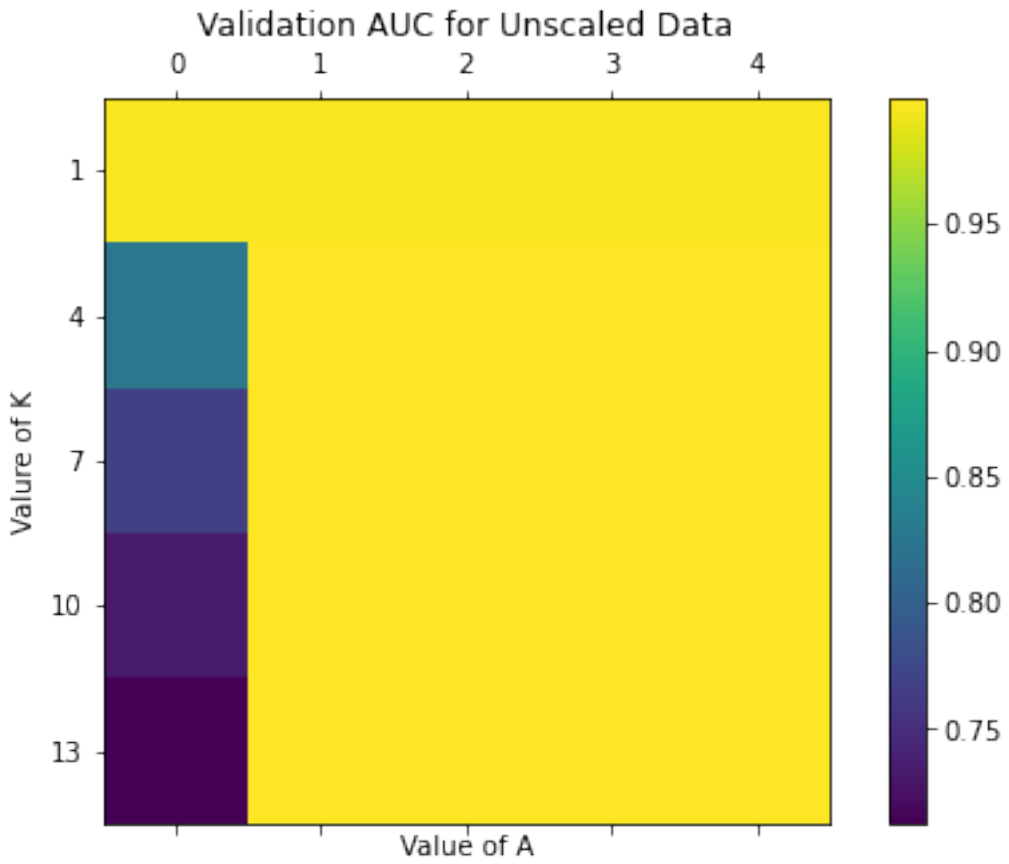
```
for i, k in enumerate(K):
    learner.train(Xt, Yt, K=k, alpha=0.0)
    train_auc[i] = learner.auc(Xt, Yt)
    validation_auc[i] = learner.auc(XvS2, Yv)
plt.plot(K, train_auc, marker='o', label = "Training AUC")
plt.plot(K, validation_auc, marker='o', label = "Validation AUC")

plt.xticks(K)
plt.xlabel("Regularization Weight")
plt.ylabel("Value of AUC")
plt.title("training and validation AUC vs regularization weight")
plt.legend()
plt.show()

print ('Training AUC:',train_auc)
print ('Validation AUC:', validation_auc)
```




```
Training AUC: [0.9973093  0.91232936 0.85316834 0.80312965 0.73286041
0.70202616]
Validation AUC: [0.5        0.5        0.5        0.49768875 0.5        0.5
]
```

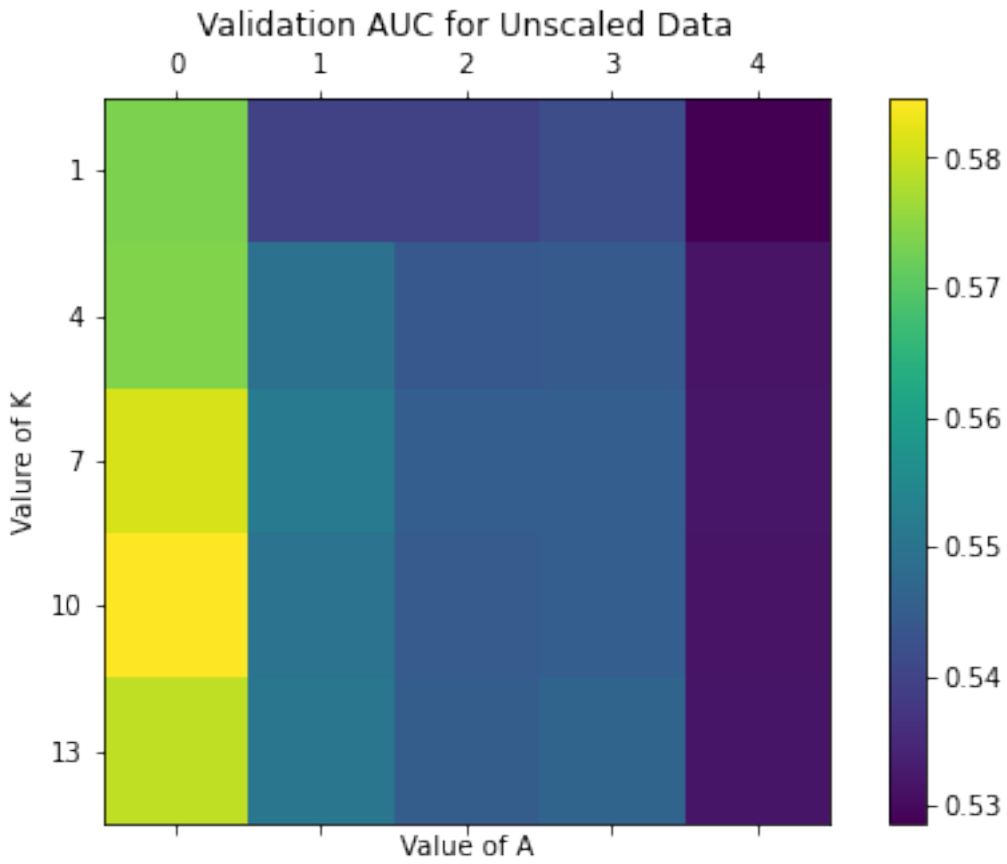
[9]:
```
# Problem 2.3
K = range(1,16,3) # Or something else
A = range(0,5,1) # Or something else
tr_auc = np.zeros((len(K),len(A)))
va_auc = np.zeros((len(K),len(A)))

for i,k in enumerate(K):
    for j,a in enumerate(A):
        learner.train(Xt, Yt, K=k, alpha=a)
        tr_auc[i][j] = learner.auc(Xt, Yt)
        va_auc[i][j] = learner.auc(Xv, Yv)
# Now plot it
f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax1 = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax1)
ax.set_xticklabels([''] + list(A))
ax.set_yticklabels([''] + list(K))
ax.set_xlabel("Value of A")
ax.set_ylabel("Valure of K")
ax.set_title("Validation AUC for Unscaled Data")
plt.show()
print ('Train AUC:',tr_auc)

f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax2 = ax.matshow(va_auc, interpolation='nearest')
f.colorbar(cax2)
ax.set_xticklabels([''] + list(A))
ax.set_yticklabels([''] + list(K))
ax.set_xlabel("Value of A")
ax.set_ylabel("Valure of K")
ax.set_title("Validation AUC for Unscaled Data")
plt.show()
print ('Validation AUC:',va_auc)
```

```
C:\Users\enyu8\mltools\knn.py:103: RuntimeWarning: invalid value encountered in
true_divide
  prob[i,:] = count / count.sum()    # save (soft) results
<ipython-input-9-3ce7752d7885>:16: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels([''] + list(A))
<ipython-input-9-3ce7752d7885>:17: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels([''] + list(K))
```




```
<ipython-input-9-3ce7752d7885>:27: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels([''] + list(A))
<ipython-input-9-3ce7752d7885>:28: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels([''] + list(K))
```

```
Train AUC: [[0.9973093  0.9973093  0.9973093  0.9973093  0.9973093 ]
 [0.82625556 0.99997668 0.99998166 0.99998184 0.99998237]
 [0.76641688 0.99997668 0.99998166 0.99998184 0.99998237]
 [0.73286041 0.99997668 0.99998166 0.99998184 0.99998237]
 [0.71249001 0.99997668 0.99998166 0.99998184 0.99998237]]
```



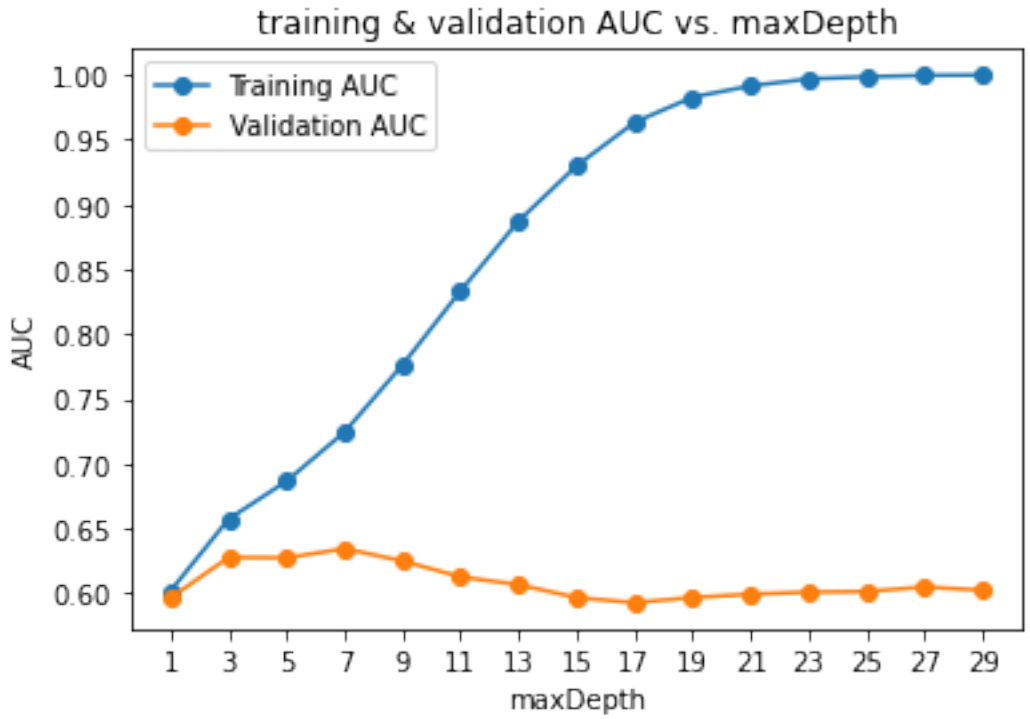

```
Validation AUC: [[0.57353193 0.53971703 0.53954363 0.54194487 0.52865904]
 [0.57391823 0.54947563 0.54403663 0.54448439 0.53159145]
 [0.5813195  0.55149496 0.54532944 0.54548088 0.53205457]
 [0.58463163 0.54998266 0.54476973 0.54528554 0.53158267]
 [0.5792036  0.55058626 0.54519555 0.54666833 0.53189874]]
```

From the plot above by considering the AUC value, I believe a = 0, k = 10 will be the best choice.

3 Problem 3

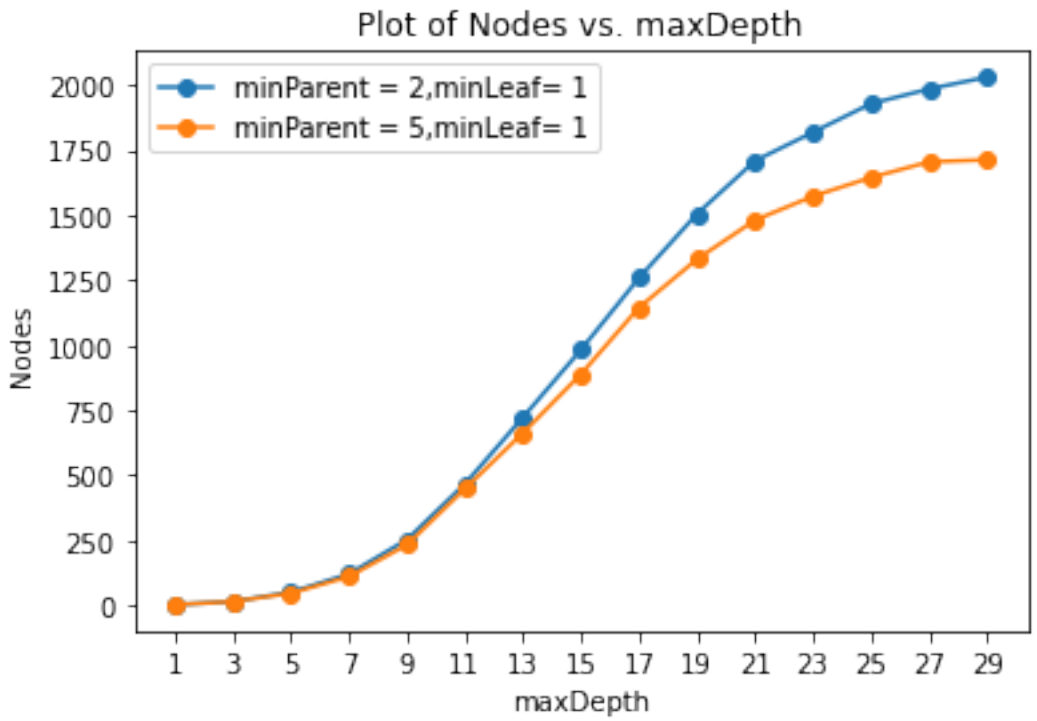
[13]:
```
# Problem 3.1
maxd = np.array(range(1,30,2))
auc_tr = np.zeros(maxd.shape[0])
auc_va = np.zeros(maxd.shape[0])

learner = ml.dtree.treeClassify()
for i, d in enumerate(maxd):
    learner.train(XtS, Yt, maxDepth = d, minParent = 2, minLeaf= 1)
    auc_tr[i] = learner.auc(XtS, Yt)
```

```
    auc_va[i] = learner.auc(XvS, Yva)

plt.plot(maxd, auc_tr, marker='o', label = "Training AUC")
plt.plot(maxd, auc_va, marker='o', label = "Validation AUC")
plt.xticks(maxd)
plt.xlabel("maxDepth")
plt.ylabel("AUC")
plt.title("training & validation AUC vs. maxDepth")
plt.legend()
plt.show()

print ('Training AUC:', auc_tr)
print ('Validation AUC:', auc_va)
```




```
Training AUC: [0.60197166 0.65728946 0.68664687 0.72452667 0.77634293 0.83336716
 0.88726723 0.92981846 0.96334991 0.98279881 0.99145731 0.99679999
 0.99841909 0.9995255  0.9998013 ]
Validation AUC: [0.59650737 0.62784387 0.62747399 0.6345369  0.62499593
0.61277488
 0.60674941 0.59677851 0.5928878  0.59688484 0.5994348  0.60102278
 0.60151668 0.60499844 0.60245903]
```

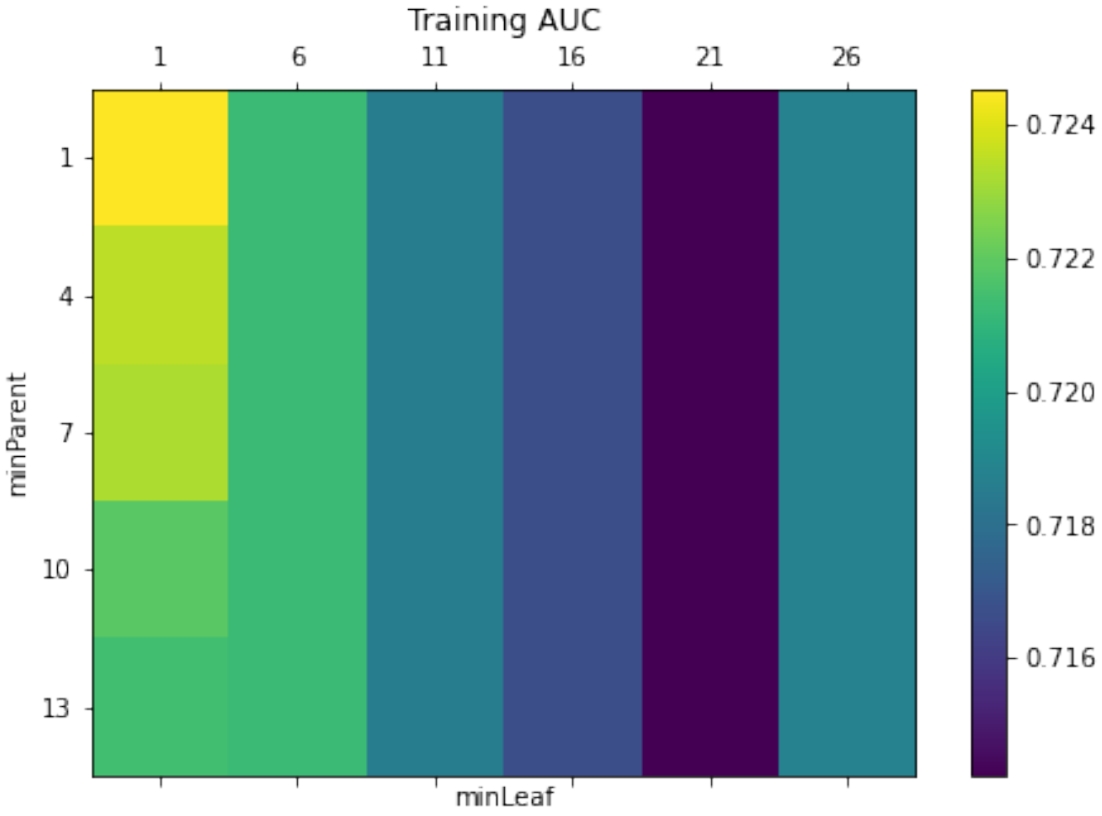
[14]:
```
# Problem 3.2
nodes = np.zeros(maxd.shape[0])

for i, d in enumerate(maxd):
    learner.train(Xt, Yt, maxDepth = d, minParent = 2, minLeaf= 1)
    nodes[i] = learner.sz
plt.plot(maxd, nodes, marker='o', label = "minParent = 2,minLeaf= 1")

for i, d in enumerate(maxd):
    learner.train(Xt, Yt, maxDepth = d, minParent = 5, minLeaf= 1)
    nodes[i] = learner.sz
plt.plot(maxd, nodes, marker='o', label = "minParent = 5,minLeaf= 1")



plt.xticks(maxd)
plt.xlabel("maxDepth")
plt.ylabel("Nodes")
plt.title("Plot of Nodes vs. maxDepth")
plt.legend()
plt.show()
```



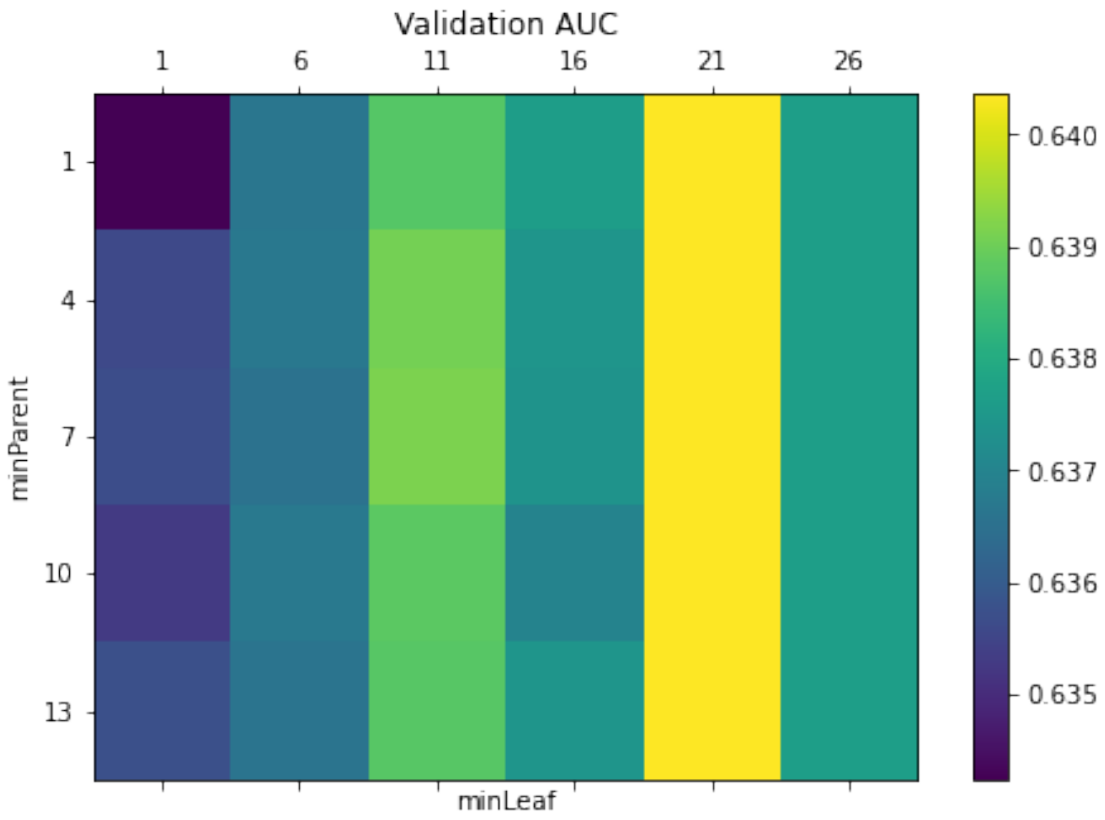
[15]:
```
# Problem 3.3
# From Problem 3.1 , I choose maxDepth = 7, because there exists less␣
 ↪overfitting problem
P = range(1,15,3)
L = range(1,30,5)

tr_auc = np.zeros((len(P),len(L)))
va_auc = np.zeros((len(P),len(L)))

for i,p in enumerate(P):
    for j,l in enumerate(L):
        learner = ml.dtree.treeClassify()
        learner.train(Xt, Yt, maxDepth = 7, minParent = p, minLeaf= l)
        tr_auc[i][j] = learner.auc(Xt, Yt) # train learner using k and a
        va_auc[i][j] = learner.auc(Xva, Yva)

# Now plot it
f, ax = plt.subplots(1, 1, figsize=(8, 5))

cax1 = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax1)
ax.set_xticklabels([''] + list(L))
ax.set_yticklabels([''] + list(P))
ax.set_xlabel("minLeaf")
ax.set_ylabel("minParent")
ax.set_title("Training AUC")

plt.show()

print ('Train AUC:',tr_auc)

f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax2 = ax.matshow(va_auc, interpolation='nearest')
f.colorbar(cax2)
ax.set_xticklabels([''] + list(L))
ax.set_yticklabels([''] + list(P))
ax.set_xlabel("minLeaf")
ax.set_ylabel("minParent")
ax.set_title("Validation AUC")

plt.show()

print ('Validation AUC:',va_auc)
```

```
<ipython-input-15-afc8fb965624>:21: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels([''] + list(L))
```

```
<ipython-input-15-afc8fb965624>:22: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels([''] + list(P))
```




```
<ipython-input-15-afc8fb965624>:34: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels([''] + list(L))
<ipython-input-15-afc8fb965624>:35: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels([''] + list(P))

Train AUC: [[0.72452667 0.7212452  0.71859429 0.71670776 0.71423927 0.7187922 ]
 [0.72350858 0.7212452  0.71859429 0.71670776 0.71423927 0.7187922 ]
 [0.72326616 0.7212452  0.71859429 0.71670776 0.71423927 0.7187922 ]
 [0.72190666 0.7212452  0.71859429 0.71670776 0.71423927 0.7187922 ]
 [0.72141613 0.7212452  0.71859429 0.71670776 0.71423927 0.7187922 ]]
```




```
Validation AUC: [[0.63423895 0.63663654 0.63876278 0.63764286 0.64035751
0.63766262]
 [0.63558583 0.63668781 0.63907537 0.63744459 0.64036171 0.63766262]
 [0.63567996 0.6365473  0.63914675 0.63738018 0.64035751 0.63766262]
 [0.6352704  0.63671545 0.63883405 0.63697061 0.64036764 0.63766262]
 [0.63572661 0.63658608 0.63877176 0.63742332 0.64036171 0.63766262]]
```

From the plot above, when maxDepth = 7, I recommend minLeaf = 21 and minParent = 7 because validation AUC value is highest at this point.

4 Problem 4

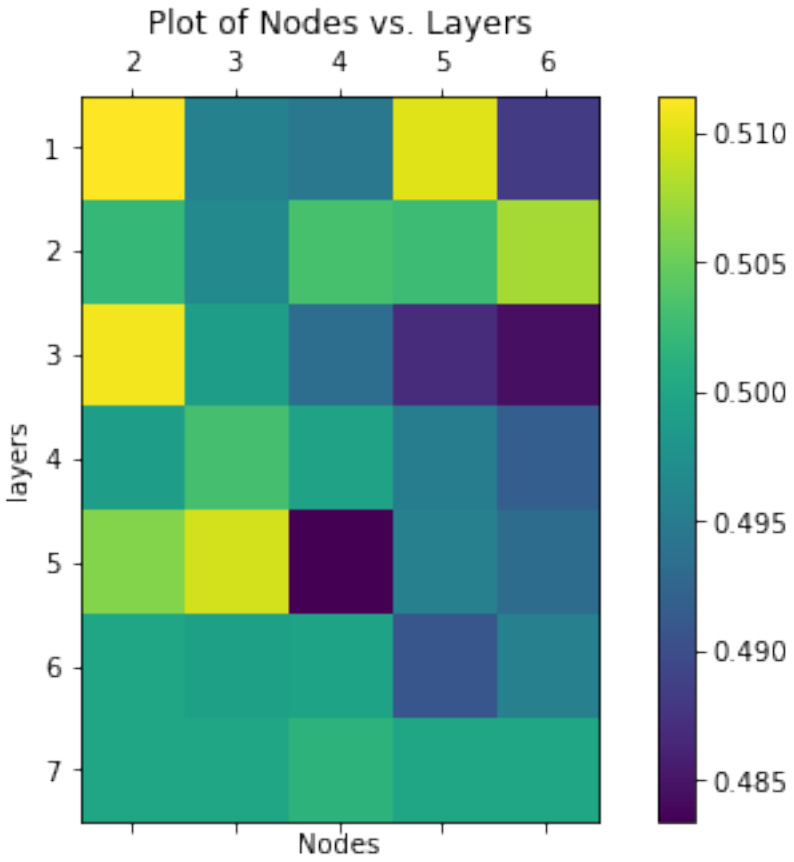
[11]:
```
# Problem 4.1
layers = range(1,8,1)
nodes = range(2,7,1)

tr_auc = np.zeros((len(layers),len(nodes)))
va_auc = np.zeros((len(layers),len(nodes)))


for i, layer in enumerate(layers):
```

```
    for j, node in enumerate(nodes):
        nn = ml.nnet.nnetClassify()
        size = [XtS.shape[1]] + layer*[node] + [2]
        nn.init_weights(size, 'random', XtS, Yt)
        nn.train(XtS, Yt, stopTol=1e-8, stepConstant=.25, stopIter=300)
        tr_auc[i][j] = nn.auc(XtS, Yt)
        print ("layer=",layer,"node=",node,tr_auc[i][j])
        va_auc[i][j] = nn.auc(Xv, Yv)

f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax1 = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax1)
ax.set_xticklabels([""] + list(nodes))
ax.set_yticklabels([""] + list(layers))
plt.xlabel("Nodes")
plt.ylabel("layers")
plt.title("Plot of Nodes vs. Layers")
plt.show()

f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax2 = ax.matshow(va_auc, interpolation='nearest')
f.colorbar(cax2)
ax.set_xticklabels([""] + list(nodes))
ax.set_yticklabels([""] + list(layers))
plt.xlabel("Nodes")
plt.ylabel("layers")
plt.title("Plot of Nodes vs. Layers")
plt.show()

print ("tr_auc:", tr_auc)
print ("va_auc:", va_auc)
```

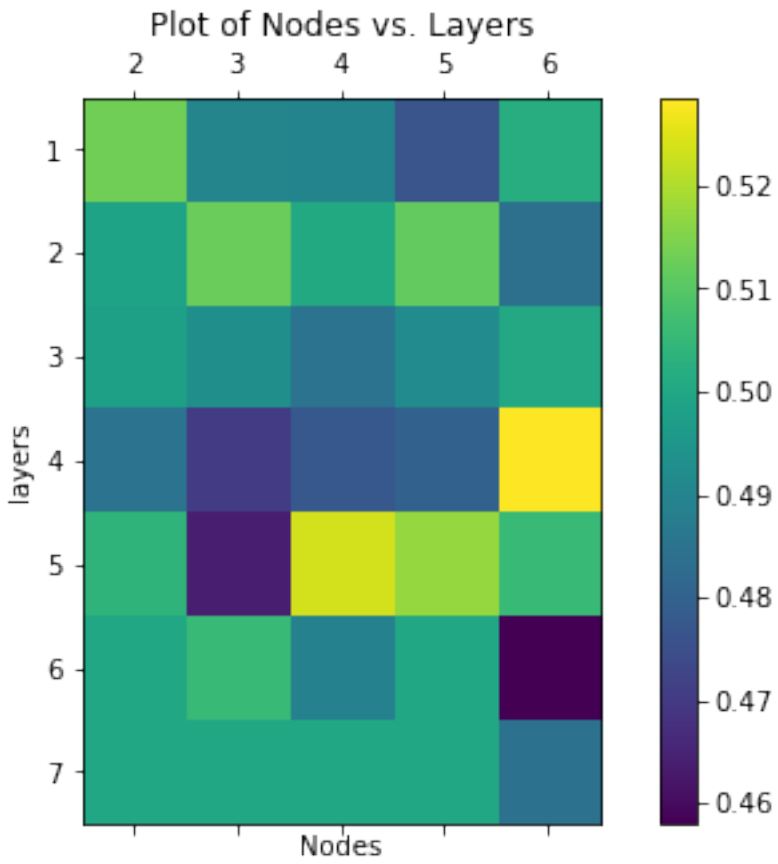
```
layer= 1 node= 2 0.5114397630384232
layer= 1 node= 3 0.4957748808236019
layer= 1 node= 4 0.49465232291054456
layer= 1 node= 5 0.5100394935961319
layer= 1 node= 6 0.48844489351493003
layer= 2 node= 2 0.502121432161018
layer= 2 node= 3 0.49656587069041397
layer= 2 node= 4 0.5032443449769881
layer= 2 node= 5 0.502669757161231
layer= 2 node= 6 0.5076092959010133
layer= 3 node= 2 0.5109453832809497
layer= 3 node= 3 0.4991335932642266
layer= 3 node= 4 0.4935363307013733
layer= 3 node= 5 0.48698597636630525
layer= 3 node= 6 0.4845148761104912
```

```
layer= 4 node= 2 0.49911904224480347
layer= 4 node= 3 0.503162983484238
layer= 4 node= 4 0.4995625821600243
layer= 4 node= 5 0.49517775667288455
layer= 4 node= 6 0.49181238980264563
layer= 5 node= 2 0.5062311191650413
layer= 5 node= 3 0.5095554060231283
layer= 5 node= 4 0.48342026680181366
layer= 5 node= 5 0.4956369123162668
layer= 5 node= 6 0.4932983682983683
layer= 6 node= 2 0.5
layer= 6 node= 3 0.4992912588832199
layer= 6 node= 4 0.4997086247086247
layer= 6 node= 5 0.4909025961857874
layer= 6 node= 6 0.49565004372403887
layer= 7 node= 2 0.5
layer= 7 node= 3 0.5
layer= 7 node= 4 0.5015040785442735
layer= 7 node= 5 0.5
layer= 7 node= 6 0.5
```

```
<ipython-input-11-2fe9eb3c0e91>:22: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels([""] + list(nodes))
<ipython-input-11-2fe9eb3c0e91>:23: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels([""] + list(layers))
```




```
<ipython-input-11-2fe9eb3c0e91>:32: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels([""] + list(nodes))
<ipython-input-11-2fe9eb3c0e91>:33: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels([""] + list(layers))
```




```
tr_auc: [[0.51143976 0.49577488 0.49465232 0.51003949 0.48844489]
 [0.50212143 0.49656587 0.50324434 0.50266976 0.5076093 ]
 [0.51094538 0.49913359 0.49353633 0.48698598 0.48451488]
 [0.49911904 0.50316298 0.49956258 0.49517776 0.49181239]
 [0.50623112 0.50955541 0.48342027 0.49563691 0.49329837]
 [0.5        0.49929126 0.49970862 0.4909026  0.49565004]
 [0.5        0.5        0.50150408 0.5        0.5       ]]
va_auc: [[0.51322186 0.49018965 0.48971384 0.47667602 0.50196386]
 [0.49879026 0.51246774 0.50085063 0.51169566 0.48411177]
 [0.49809898 0.49319493 0.48500954 0.49196274 0.50060824]
 [0.48488385 0.4705308  0.47749523 0.47980922 0.52849736]
 [0.50397711 0.46395017 0.5235574  0.51727528 0.50555942]
 [0.5        0.50557064 0.48893502 0.49989002 0.45804511]
 [0.5        0.5        0.5        0.5        0.48468859]]
```

I recommend Nodes = 5, Layers = 4, since it has highest validation AUC value and there is no overfitting by observing trainging AUC.

[12]:
```
# Problem 4.2
def sig(z): return np.atleast_2d(z)
```

```
def dsig(z): return np.atleast_2d(1)
layer , node = 4, 5 #from previous recommendation
nn.set_activation('custom', sig, dsig)
nn = ml.nnet.nnetClassify()
size = [XtS.shape[1]] + layer*[node] + [2]
nn.init_weights(size, 'random', XtS, Yt)
nn.train(XtS, Yt, stopTol=1e-8, stepConstant=.25, stopIter=100)

print('Custom Train AUC',nn.auc(XtS, Yt))
print('Custom Validation AUC', nn.auc(XvS, Yva))
```

```
Custom Train AUC 0.5125451081602117
Custom Validation AUC 0.5025592964632191
```

[13]:
```
nn.set_activation('logistic', sig, dsig)
nn = ml.nnet.nnetClassify()
nn.init_weights(size, 'random', XtS, Yt)
nn.train(XtS, Yt, stopTol=1e-8, stepConstant=.25, stopIter=100)

print('Logistic Train AUC',nn.auc(XtS, Yt))
print('Logistic Validation AUC', nn.auc(XvS, Yva))
```

```
Logistic Train AUC 0.49957660082507827
Logistic Validation AUC 0.4997831857598815
```

[14]:
```
nn.set_activation('htangent', sig, dsig)
nn = ml.nnet.nnetClassify()
nn.init_weights([XtS.shape[1], 5, 2], 'random', XtS, Yt)
nn.train(XtS, Yt, stopTol=1e-8, stepConstant=.25, stopIter=100)

print('htangent Train AUC',nn.auc(XtS, Yt))
print('htangent Validation AUC', nn.auc(XvS, Yva))
```

```
htangent Train AUC 0.5061813440315267
htangent Validation AUC 0.49737403808789976
```

htangent is better in this case

[5]:
```
# Problem 4.3
Xte = np.genfromtxt('data/X_test.txt', delimiter=None)
Xte,Y = ml.shuffleData(Xte,Y)
Xte_tr, Xte_va, Yte_tr, Yte_va = ml.splitData(Xte, Y)
```

[19]:
```
maxd = range(1,30,3)

for d in maxd:
    learner = ml.dtree.treeClassify()
    learner.train(Xt, Yt, maxDepth = d, minParent = 2, minLeaf= 1)
```

```
    print("when Depth=", d ,"train AUC" ,learner.auc(Xt, Yt))
    print("when Depth=", d ,"validation AUC" ,learner.auc(Xva, Yva))
    print("when Depth=", d ," - mean AUC" ,(learner.auc(Xt, Yt)+learner.auc(Xva,␣
 ↪Yva))/2)
```

```
when Depth= 1 train AUC 0.6049942250735344
when Depth= 1 validation AUC 0.6035318117806983
when Depth= 1  - mean AUC 0.6042630184271163
when Depth= 4 train AUC 0.6757339002578376
when Depth= 4 validation AUC 0.6359300946871778
when Depth= 4  - mean AUC 0.6558319974725078
when Depth= 7 train AUC 0.7406514410314651
when Depth= 7 validation AUC 0.6242001615335818
when Depth= 7  - mean AUC 0.6824258012825235
when Depth= 10 train AUC 0.8268231073592716
when Depth= 10 validation AUC 0.6111851105403466
when Depth= 10  - mean AUC 0.7190041089498092
when Depth= 13 train AUC 0.9023005647608068
when Depth= 13 validation AUC 0.5980345518100573
when Depth= 13  - mean AUC 0.750167558285432
when Depth= 16 train AUC 0.9508929186115356
when Depth= 16 validation AUC 0.593483890806416
when Depth= 16  - mean AUC 0.7721884047089758
when Depth= 19 train AUC 0.9787144918914096
when Depth= 19 validation AUC 0.5943835956844883
when Depth= 19  - mean AUC 0.786549043787949
when Depth= 22 train AUC 0.9933893677336199
when Depth= 22 validation AUC 0.601323969413309
when Depth= 22  - mean AUC 0.7973566685734644
when Depth= 25 train AUC 0.9972629226832209
when Depth= 25 validation AUC 0.60226054680517
when Depth= 25  - mean AUC 0.7997617347441954
when Depth= 28 train AUC 0.999471235251317
when Depth= 28 validation AUC 0.6057764489537333
when Depth= 28  - mean AUC 0.8026238421025251
```

[20]:
```
maxd = range(30,70,4)

for d in maxd:
    learner = ml.dtree.treeClassify()
    learner.train(Xt, Yt, maxDepth = d, minParent = 2, minLeaf=1)

    print("when Depth=", d ,"train AUC" ,learner.auc(Xt, Yt))
    print("when Depth=", d ,"validation AUC" ,learner.auc(Xva, Yva))
    print("when Depth=", d ," - mean AUC" ,(learner.auc(Xt, Yt)+learner.auc(Xva,␣
 ↪Yva))/2)
```

```
when Depth= 30 train AUC 0.9996869791367705
when Depth= 30 validation AUC 0.6054842455218857
when Depth= 30  - mean AUC 0.8025856123293281
when Depth= 34 train AUC 0.9999015607053939
when Depth= 34 validation AUC 0.6075141320791816
when Depth= 34  - mean AUC 0.8037078463922878
when Depth= 38 train AUC 0.9999739819848044
when Depth= 38 validation AUC 0.6053817970490383
when Depth= 38  - mean AUC 0.8026778895169213
when Depth= 42 train AUC 0.9999739819848044
when Depth= 42 validation AUC 0.6063177819163099
when Depth= 42  - mean AUC 0.8031458819505571
when Depth= 46 train AUC 0.9999739819848044
when Depth= 46 validation AUC 0.6077808902571775
when Depth= 46  - mean AUC 0.8038774361209909
when Depth= 50 train AUC 0.9999739819848044
when Depth= 50 validation AUC 0.6057567346754179
when Depth= 50  - mean AUC 0.8028653583301111
when Depth= 54 train AUC 0.9999739819848044
when Depth= 54 validation AUC 0.6071969454794036
when Depth= 54  - mean AUC 0.803585463732104
when Depth= 58 train AUC 0.9999739819848044
when Depth= 58 validation AUC 0.6068368501083109
when Depth= 58  - mean AUC 0.8034054160465576
when Depth= 62 train AUC 0.9999739819848044
when Depth= 62 validation AUC 0.6061881897116311
when Depth= 62  - mean AUC 0.8030810858482178
when Depth= 66 train AUC 0.9999739819848044
when Depth= 66 validation AUC 0.6053039015740448
when Depth= 66  - mean AUC 0.8026389417794246
```

[23]:
```
parent = range(1,30,3)

for p in parent:
    learner = ml.dtree.treeClassify()
    learner.train(Xt, Yt, maxDepth = 46, minParent = p, minLeaf= 1)

    print("when minParent=", p ,"train AUC" ,learner.auc(Xt, Yt))
    print("when minParent=", p ,"validation AUC" ,learner.auc(Xva, Yva))
    print("when minParent=", p ," - mean AUC" ,(learner.auc(Xt, Yt)+learner.
 ↪auc(Xva, Yva))/2)
```

```
when minParent= 1 train AUC 0.9999739819848044
when minParent= 1 validation AUC 0.6061277397147105
when minParent= 1  - mean AUC 0.8030508608497574
when minParent= 4 train AUC 0.9984255077402254
when minParent= 4 validation AUC 0.6066804957742881
when minParent= 4  - mean AUC 0.8025530017572567
```

```
when minParent= 7 train AUC 0.9910239635754939
when minParent= 7 validation AUC 0.6048173903186128
when minParent= 7  - mean AUC 0.7979206769470534
when minParent= 10 train AUC 0.9826584117137217
when minParent= 10 validation AUC 0.6049614612154814
when minParent= 10  - mean AUC 0.7938099364646016
when minParent= 13 train AUC 0.9721879934817272
when minParent= 13 validation AUC 0.6005955707485335
when minParent= 13  - mean AUC 0.7863917821151304
when minParent= 16 train AUC 0.9614755451221229
when minParent= 16 validation AUC 0.6006325639874728
when minParent= 16  - mean AUC 0.7810540545547978
when minParent= 19 train AUC 0.9531913554384536
when minParent= 19 validation AUC 0.6022242682036161
when minParent= 19  - mean AUC 0.7777078118210348
when minParent= 22 train AUC 0.942678646755643
when minParent= 22 validation AUC 0.6021041744317569
when minParent= 22  - mean AUC 0.7723914105937
when minParent= 25 train AUC 0.9333843143376788
when minParent= 25 validation AUC 0.6024507333218919
when minParent= 25  - mean AUC 0.7679175238297853
when minParent= 28 train AUC 0.9270352033580584
when minParent= 28 validation AUC 0.6037477557714885
when minParent= 28  - mean AUC 0.7653914795647735
```

[25]:
```
leaf = range(1,30,3)

for l in leaf:
    learner = ml.dtree.treeClassify()
    learner.train(Xt, Yt, maxDepth = 46, minParent = 4, minLeaf= l)

    print("when minLeaf=", l ,"train AUC" ,learner.auc(Xt, Yt))
    print("when minLeaf=", l ,"validation AUC" ,learner.auc(Xva, Yva))
    print("when minLeaf=", l ," - mean AUC" ,(learner.auc(Xt, Yt)+learner.
 ↪auc(Xva, Yva))/2)
```

```
when minLeaf= 1 train AUC 0.9986237274642413
when minLeaf= 1 validation AUC 0.6055115474452936
when minLeaf= 1  - mean AUC 0.8020676374547675
when minLeaf= 4 train AUC 0.9724950239428326
when minLeaf= 4 validation AUC 0.5995695054919383
when minLeaf= 4  - mean AUC 0.7860322647173854
when minLeaf= 7 train AUC 0.9303896497295646
when minLeaf= 7 validation AUC 0.6026499312078952
when minLeaf= 7  - mean AUC 0.76651979046873
when minLeaf= 10 train AUC 0.896553623308136
when minLeaf= 10 validation AUC 0.6039242975709109
when minLeaf= 10  - mean AUC 0.7502389604395234
```

```
when minLeaf= 13 train AUC 0.8638572038340719
when minLeaf= 13 validation AUC 0.6081044461493795
when minLeaf= 13  - mean AUC 0.7359808249917257
when minLeaf= 16 train AUC 0.8411534009658674
when minLeaf= 16 validation AUC 0.6065506024505557
when minLeaf= 16  - mean AUC 0.7238520017082115
when minLeaf= 19 train AUC 0.8271209286947236
when minLeaf= 19 validation AUC 0.6084580376264438
when minLeaf= 19  - mean AUC 0.7177894831605838
when minLeaf= 22 train AUC 0.8109362032902865
when minLeaf= 22 validation AUC 0.611338468947933
when minLeaf= 22  - mean AUC 0.7111373361191098
when minLeaf= 25 train AUC 0.7973646343467484
when minLeaf= 25 validation AUC 0.6128447247351254
when minLeaf= 25  - mean AUC 0.7051046795409369
when minLeaf= 28 train AUC 0.7885572232715856
when minLeaf= 28 validation AUC 0.6148577402995475
when minLeaf= 28  - mean AUC 0.7017074817855666
```

For problem 4.3, since the training data is massive, I choose decision tree to train the model in ideal time rather than other classifier such as neural network, knn, etc. After the training section, I test the every single value of maxDepth, minParent and minLeaf, chossing them according to the validation AUC and mean AUC to find the value that can fit data the most. The result show that when Depth = 46, Parent = 4, Leaf = 25 will be the best value to predict this data.

5 Statement of Collaboration

I finished this assigment alone, only discussed the plot with Josh Ho, Will and Hamza. I did not discuss the solutions to this homework with any person.