COEN 281 – Data Mining and Pattern Recognition HW #3

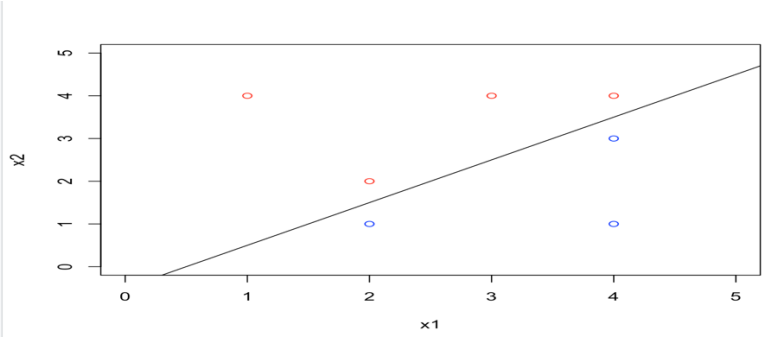
Bhaumik Ichhaporia (1549327) & Sanika Lakka (975020)

Question 1:

a)

```
x1 = c(3, 2, 4, 1, 2, 4, 4)
x2 = c(4, 2, 4, 4, 1, 3, 1)
colors = c("red", "red", "red", "red", "blue", "blue", "blue")
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
```

b)



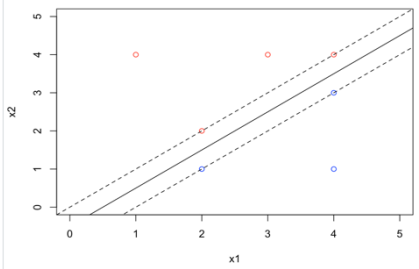
As shown in the plot, the optimal separating hyperplane has to be between the observations (2,1), and (2,2) and between the observations (4,3) and (4,4).So it is a line that passes through points (2,1.5) and (4,3.5) which equation is $X1-X2-0.5=0$.

c) -> The classification rule is "Classify to Red if $X1-X2-0.5<0$, and classify to Blue otherwise.

B0 = -0.5 ,B1= 1 and B2 = -1

d)

```
#d)On your sketch, indicate the margin for the maximal margin hyperplane.
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
abline(-1, 1, lty = 2)
abline(0, 1, lty = 2)
```



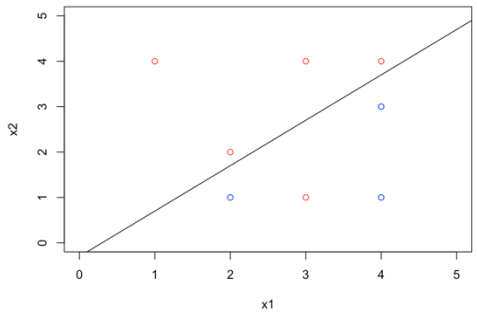
e) Indicate the support vectors for the maximal margin classifier.

The support vectors are the points (2,1), (2,2), (4,3) and (4,4).

f) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.
Looking at the plot in the above observations it is clear that if we moved the observation (4,1) it would not change the maximal margin hyperplane as it is not a support vector.

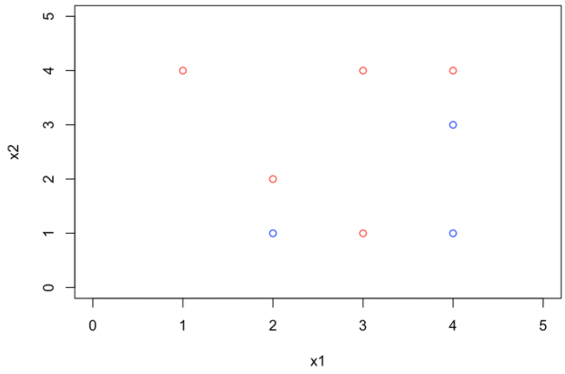
g)

```
#g)
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.3, 1)
```



h)

```
#h)
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
points(c(3), c(1), col = c("red"))
```



Question 2:

```
az_5000 <- read.table("~/Downloads/az-5000.txt",header = T)
indices <- sample(1:nrow(az_char), size = (0.8*nrow(az_char)))##sampling 80% of the data
training_az<- az_char[indices,]  ##putting the sampled data in training vector
test_az <- az_char[-indices,]
```

a)

We are given 18 input feature and 26 output features.

18 input all feed into nH hidden layers. Plus 1 per unit for the bias .
nH weights each feed into 26 output nodes. Plus 1 per unit for the bias.

Total number of weights = 19*nH + 26*nH + 26

According to class heuristic
Total number of weights = n/10, where n is the number of training points
45*nH + 26 = n/10, where n is 4000 observations.
45*nH + 26 = 4000/10
nH = 8.31
nH is between 8 and 9
Approximately we can take 8

b)

```
char_num = as.numeric(training_az$char)

targetMatrix = matrix(data=0, nrow=4000, ncol=26)


for (i in 1:4000){
  char_value = char_num[i]
  targetMatrix[i, char_value] = 1;
}

sum(targetMatrix==1)
```

```
> sum(targetMatrix==1)
[1] 4000
```

c)

```
#install"(nnet")
library(nnet)
for(i in 1:20){
  nnet_model <- nnet(data=training_az, formula(training_az$char~.), size=i, maxit=1000)
}
```

```
iter 870 value 709.057507
iter 880 value 708.224289
iter 890 value 707.359220
iter 900 value 706.501273
iter 910 value 705.779618
iter 920 value 704.861491
iter 930 value 703.940420
iter 940 value 703.178503
iter 950 value 702.329577
iter 960 value 701.480453
iter 970 value 700.686196
iter 980 value 699.891007
iter 990 value 698.951491
iter1000 value 697.813902
final  value 697.813902
stopped after 1000 iterations
```

d)

```
fittedvalue=list()
for (i in 1:20)
{
  fittedvalue[[i]]=nnet_model[[i["fitted.values"]]]
}

library(Metrics)
MSE = 0
for (i in 1:20){
  MSE[i] = mse(targetMatrix, fittedvalue[[i]])
}
```

```
> MSE = 0
> for (i in 1:20){
+   MSE[i] = mse(targetMatrix, fitted_values[[i]])
+ }
> MSE
 [1] 0.033859022 0.027410753 0.022566529 0.027373196 0.013938697 0.011901518 0.011626956 0.010091002 0.007980679
[10] 0.007374720 0.007869668 0.006262514 0.005842655 0.006272883 0.004678093 0.003964182 0.006327985 0.004206130
[19] 0.002333722 0.002431175
>
```

e)

```
az_pred = list()
for (i in 1:20){
  az_pred[[i]] = predict(nnet_model[[i]], test_az)
}


num_test = as.numeric(az5000_test$char)
targetMatrix_test = matrix(data=0, nrow=1000, ncol=26)

for (i in 1:1000){
  char_val_test = num_test[i]
  targetMatrix_test[i, char_val_test] = 1;
}

sum(targetMatrix_test==1)

MSE_test = 0
for (i in 1:20){
  MSE
}
```

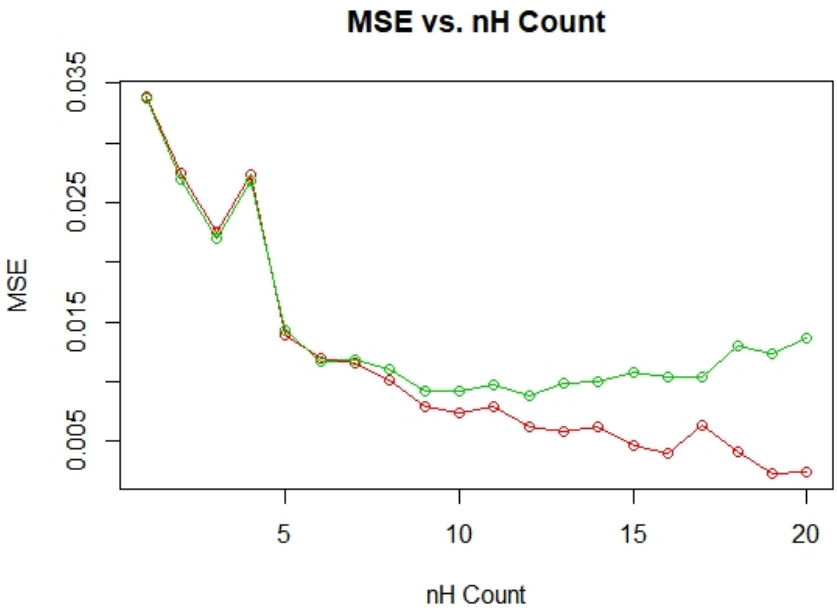
```
> MSE_test = 0
> for (i in 1:20){
+   MSE_test[i] = mse(targetMatrix_test, az5000.pred[[i]])
+ }
> MSE_test
 [1] 0.033659576 0.026896280 0.022003482 0.026853293 0.014346611 0.011759550 0.011831667 0.011054972 0.009281362
[10] 0.009265691 0.009733540 0.008800321 0.009829022 0.010011613 0.010765145 0.010389715 0.010367438 0.012944650
[19] 0.012304219 0.013633450
>
```

f)

```
# Plot d
plot(MSE, col=2, main= "MSE vs. nH Count", xlab="nH Count", ylab= "MSE")
lines(MSE, col=2)

# Plot e on the same figure
par(new=TRUE)
points(MSE_test, col=3)
lines(MSE_test, col=3)

best_net=min(MSE_test)
which(MSE_test==best_net)
```



nh = 9

cm=table(az_red[[12]], az_test$char)

What was the total accuracy on the test set with nH=9

accuracy = sum(diag(cm))/sum(cm)*100

Accuracy = 82.2%

Question 3:

a)

```
#3
install.packages('e1071')
library(e1071)
spam_data<- read.table("~/Downloads/spam.csv", header=TRUE,sep=",")

indices <- sample(1:nrow(az_char), size = (0.85*nrow(az_char)))##sampling 80% of the
levels(training$char)
training_svm<- spam_data[indices,] ##putting the sampled data in training vector
test_svm <- spam_data[-indices,] ##putting the rest dataset in the test_data
help("tune.svm")
help("svm")
t_500 <- sample(1:nrow(training_svm), 500)
training_500=training_svm[t_500,]
```

```
> summary(tuned_parameters)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
 0.001  100

- best performance: 0.09201848

- Detailed performance results:
  gamma cost      error dispersion
1 1e-06   10 0.40305226 0.04836686
2 1e-05   10 0.39875069 0.04765995
3 1e-04   10 0.17298779 0.04171877
4 1e-03   10 0.09835495 0.04162512
5 1e-06  100 0.39875069 0.04765995
6 1e-05  100 0.17298779 0.04171877
7 1e-04  100 0.10025314 0.04948303
8 1e-03  100 0.09201848 0.04133355
```

b)

```
> svm_model_after_tune <- svm(type ~ ., data=training_svm, kernel="radial", cost=100, gamma=0.001)
> summary(svm_model_after_tune)

Call:
svm(formula = type ~ ., data = training_svm, kernel = "radial", cost = 100, gamma = 0.001)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100
      gamma:  0.001

Number of Support Vectors:  773

 ( 385 388 )


Number of Classes:  2

Levels:
 nonspam spam



>
```

c)

```
> cm=table(test_svm$type,y_pred)
> print(cm)
          y_pred
           nonspam spam
  nonspam      427   19
  spam          22  237
> accuracy_test=sum(diag(cm))/sum(cm)
> print(accuracy_test)
[1] 0.941844
>
```

Accuracy is 0.941844

Question 4:

```
#4
house_tree<- read.table("~/Downloads/housetype_data.txt", header=TRUE,sep=",")
dim(house_tree)
head(house_tree)

library(rpart)
indicest <- sample(1:nrow(house_tree), size = (0.9*nrow(house_tree)))##sampling 90% of the

trainingt <- house_tree[indicest,] ##putting the sampled data in training vector
sum(is.na(trainingt))
sum(is.na(house_tree))
testt <- house_tree[-indicest,] ##putting the rest dataset in the test_data
str(house_tree)
```

```
> dim(house_tree)
[1] 9013   14
> head(house_tree)
  ht sex ms age edu ocu inc ba di hs hs2 hhs eth lang
1  1   2  4   7   4   5  NA  5  1  1   0   1   7    1
2  1   2  1   5   4   5   9  5  3  3   0   1   7   NA
3  1   1  1   5   5   5   9  5  3  5   2   1   7    1
4  3   2  1   3   5   1   9  5  2  3   1   2   7    1
5  1   2  5   1   2   6   1  5  1  4   2   3   7    1
6  1   2  5   1   2   6   1  3  1  4   2   3   7    1
>
> library(rpart)
> indicest <- sample(1:nrow(house_tree), size = (0.9*nrow(house_tree)))##sampling 90% of the
>
> trainingt <- house_tree[indicest,] ##putting the sampled data in training vector
> sum(is.na(trainingt))
[1] 2414
> sum(is.na(house_tree))
[1] 2671
> testt <- house_tree[-indicest,] ##putting the rest dataset in the test_data
>
```

a)

```
as.factor((house_tree$ht))
as.factor((house_tree$sex))
as.factor((house_tree$ms))
#as.factor((house_tree$age))
as.factor((house_tree$edu))
as.factor((house_tree$ocu))
#as.factor((house_tree$inc))
#as.factor((house_tree$hs))
#as.factor((house_tree$hs2))
as.factor((housetype_data$eth))
as.factor((housetype_data$lang))
#as.factor((house_tree$ba))
as.factor((house_tree$di))
```

Here the commented ones in green are continuous variables while the remaining ones are categorical.

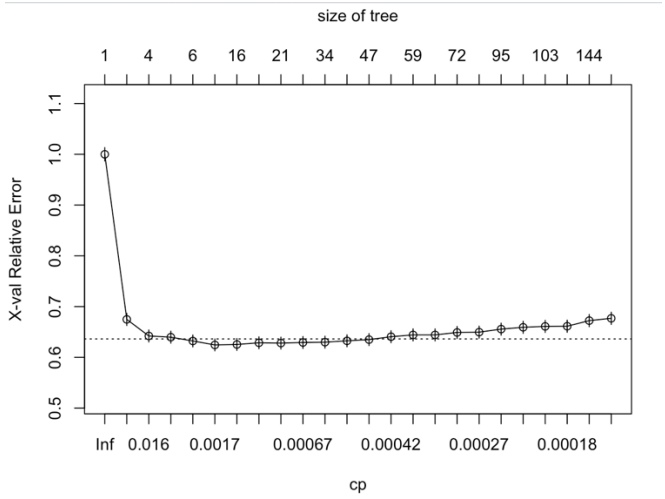
->classifier=rpart(formula = ht ~ . , data = trainingt,method = 'class',cp=.0001)

b) ->View(classifier$cptable)

	CP	nsplit	rel error	xerror	xstd
6	0.0014375562	9	0.6181491	0.6244385	0.01178729
7	0.0011979635	15	0.6085654	0.6253369	0.01179283
9	0.0007487272	20	0.6037736	0.6280323	0.01180939
8	0.0007986423	17	0.6061695	0.6286313	0.01181305
10	0.0005989817	24	0.6007787	0.6292303	0.01181671
11	0.0005390836	33	0.5953878	0.6298293	0.01182037
5	0.0019466906	5	0.6259359	0.6322252	0.01183494
12	0.0004991514	39	0.5911950	0.6325247	0.01183676
13	0.0004492363	46	0.5864031	0.6349206	0.01185124
4	0.0077867625	4	0.6337227	0.6397125	0.01187999
14	0.0003993212	55	0.5822102	0.6406110	0.01188534
3	0.0080862534	3	0.6418089	0.6421084	0.01189424
15	0.0003743636	58	0.5810123	0.6442049	0.01190665
16	0.0003593890	62	0.5795148	0.6442049	0.01190665
17	0.0002994909	71	0.5753220	0.6489967	0.01193480
18	0.0002395927	84	0.5714286	0.6495957	0.01193830
19	0.0002246181	94	0.5690326	0.6555855	0.01197301
20	0.0001996606	99	0.5678347	0.6591794	0.01199362
21	0.0001871818	102	0.5672357	0.6609763	0.01200385
22	0.0001711376	110	0.5657382	0.6612758	0.01200556
23	0.0001497454	143	0.5585505	0.6723570	0.01206773
2	0.0329439952	2	0.6747529	0.6747529	0.01208097
24	0.0001000000	151	0.5573525	0.6768494	0.01209249
1	0.1626235400	0	1.0000000	1.0000000	0.01327409

```
print(classifier$cptable[which.min(classifier$cptable[,"xerror"]),"CP"])
1] 0.001437556
plotcp(classifier)
```

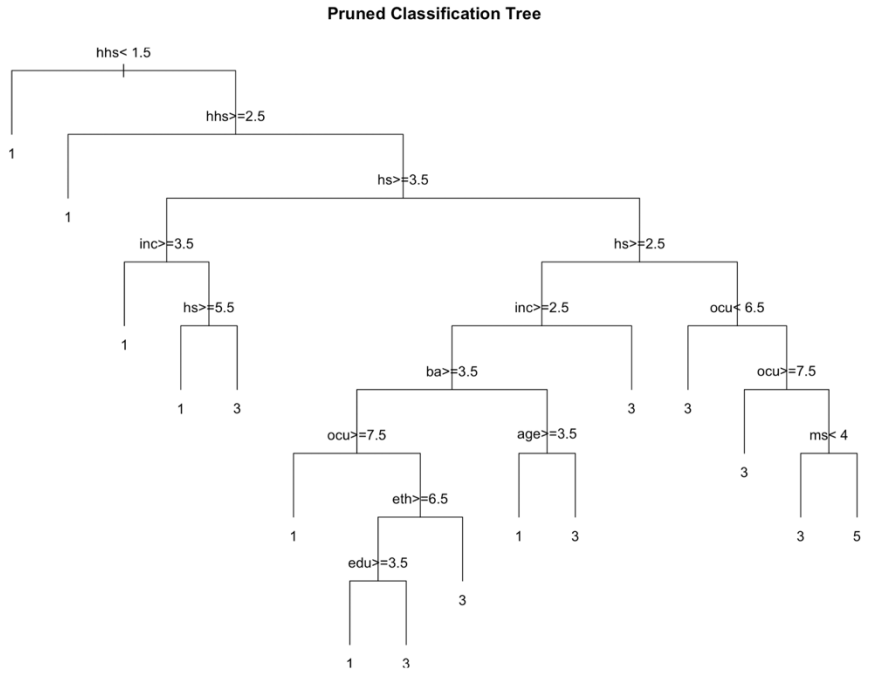
Row with. Minimum cross validation error



c) ->prune_tree=prune(classifier,c= 0.001437556)

d)

```
plot(prune_tree, uniform=TRUE,
     main="Pruned Classification Tree" )
text(prune_tree)
```



➔ From the decision tree we can see that the primary splitter is hhs variable i.e the number of person in household under 18. So we can say that it plays the main deciding factor in buying a

house. Exploring the left side of that node we can deduce that if the number of person in household under 18 are less then 2 i.e 0 or 1 then the would prefer type 1 of home i.e House. Further moving on right side of the decision tree we can see that Number of people in household ,income and hence occupation are the major factors contributing towards buying the house. Age ,Marital Status ,education and ethnicity are less deciding factors. The house that are more Frequently bought are of type 1,3 and 5 and they are House, Apartment and Other .

e) ->Yes, there were many surrogate splits used in the construction of optimal tree. The below image contains the necessary information.

->summary(prune_tree)

```
Node number 11987: 33 observations,    complexity param=0.0001711376
  predicted class=1  expected loss=0.5454545  P(node) =0.004068549
    class counts:    15     9     0     7     2
   probabilities: 0.455 0.273 0.000 0.212 0.061
  left son=23974 (9 obs) right son=23975 (24 obs)
  Primary splits:
      age < 4.5 to the left,  improve=1.5934340, (0 missing)
      ocu < 3.5 to the left,  improve=1.1458330, (1 missing)
      inc < 7.5 to the right, improve=1.0153570, (1 missing)
      sex < 1.5 to the left,  improve=0.7462121, (0 missing)
      edu < 3.5 to the right, improve=0.4545455, (0 missing)
  Surrogate splits:
      ms < 4.5 to the right, agree=0.758, adj=0.111, (0 split)
```

The primary split is on the following

The surrogate split is on the ms variable o.e Maritial status

What does a Surrogate mean?

A surrogate is a mimic or a substitute for the primary splitter of a node. We can't start the process of about surrogates until we have a primary splitter in hand. Once we have that splitter in hand, then we can go looking for surrogates. The ideal surrogate splits the data in exactly the same way as the primary split, in other words, we are looking for clones, close approximations, something else in the data that can do the same work that the primary splitter accomplished.

f)

```
> cm=table(testt$ht,y_pred)
> print(cm)
   y_pred
      1   2   3   4   5
  1 468   6  72   0   1
  2  47   5  21   0   1
  3  35   1 186   0   4
  4   9   0   2   0   0
  5  20   1  18   0   5
> accuracy_test<- sum(diag(cm))/sum(cm)
> print(accuracy_test)
[1] 0.7361419
>
```