TDDE01 exam 2021-08-25 Daniel Bissessar danbi675 981120-3893

I solemnly swear that I wrote the exam honestly, I did not use any unpermitted aids, nor

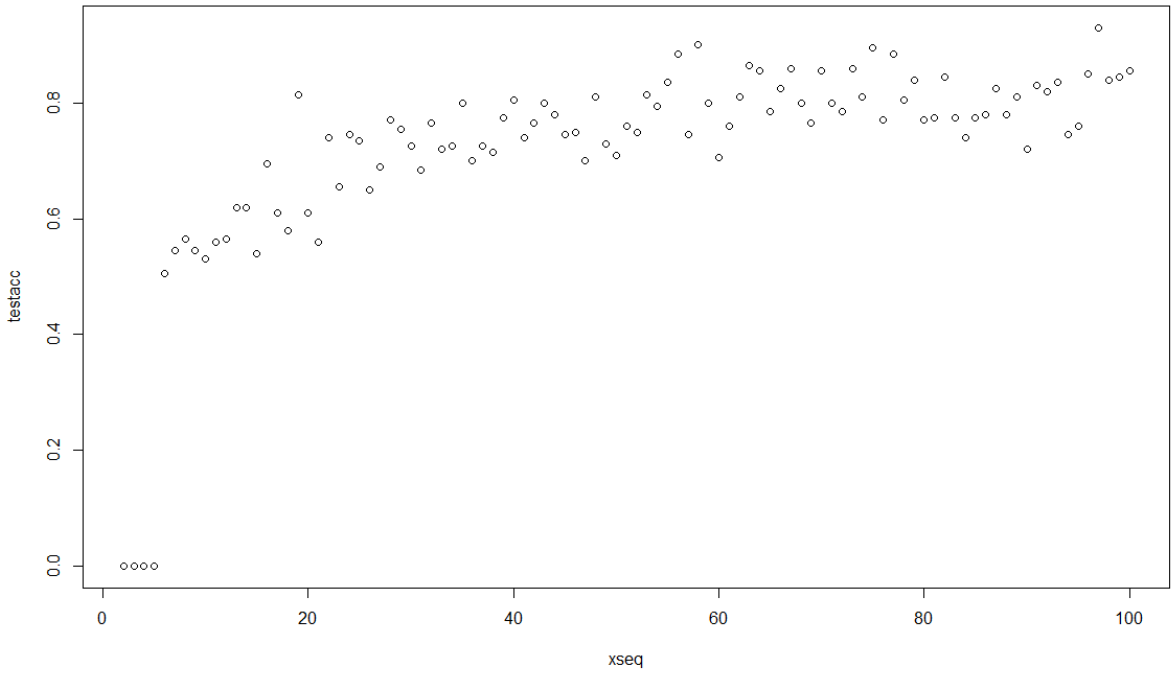
did I communicate with anybody except of the course examiners.

Daniel Bissessar

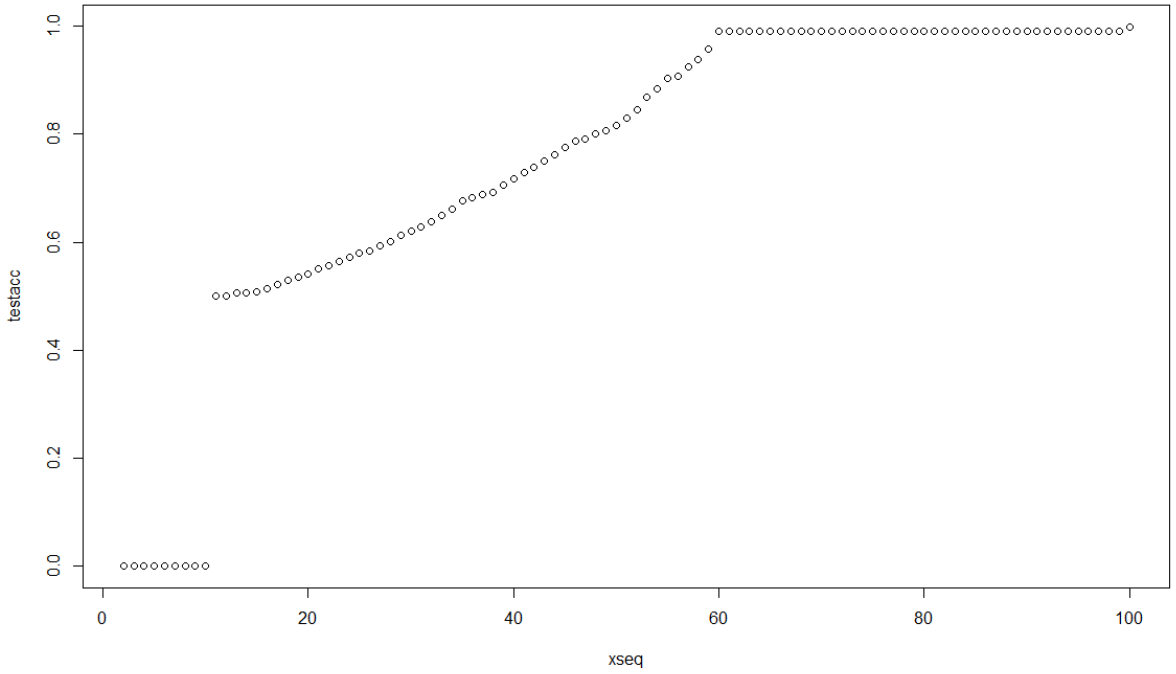
Q1.

Part 1:

a) When the amount of features increases the dimensions of the datasets increase. This creates an issue for knn as the Euclidian distance to all the vectors will become almost the same for every vector, which sort of removes the points of knn looking at the “nearest” neighbour since all vectors will be pretty much as close to the search vector. This is known as the “curse of dimensionality”.



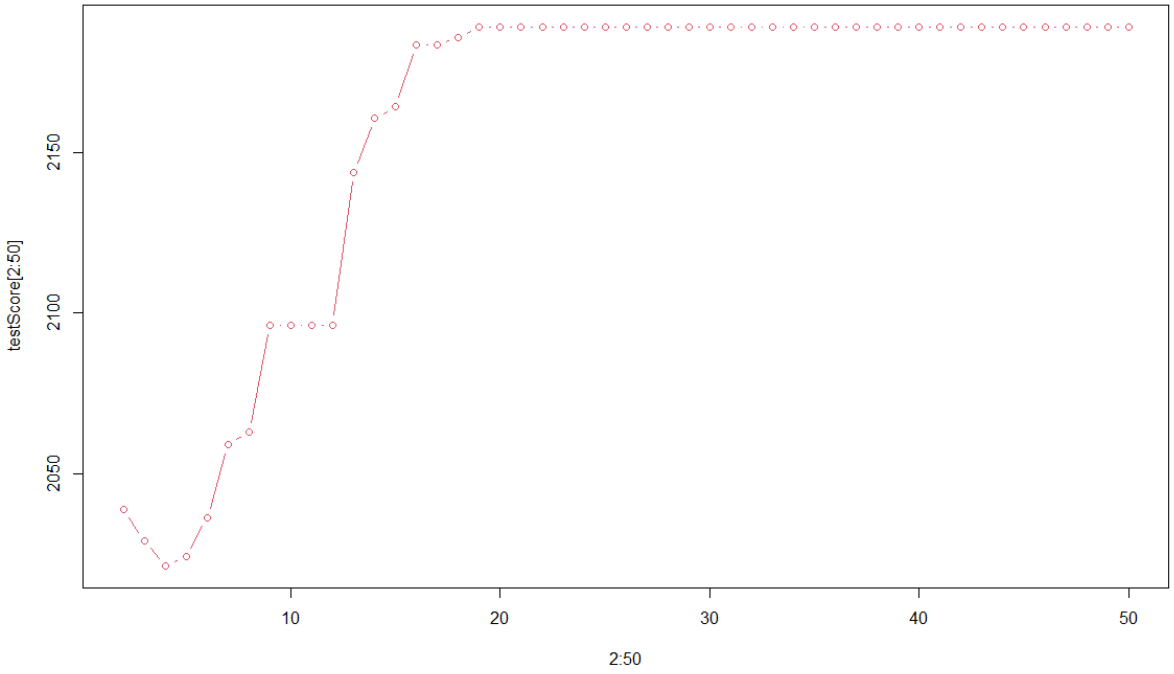
b) When we increase the value of K we increase the accuracy of the model. However we risk overfitting our model substantially, however I think this is mitigated in part in this case due to our test and training data having exactly the same distribution.



Part 2:

1.

a)



b) 2 features were selected by the tree, “Blood.systolic” and “cholesterol”

c)

Fit	0	1
0	2125	0
1	375	0

d) The misclassification rate is 0.15

2.

Fit	0	1
0	2013	22
1	358	17

The test misclassification rate is 0.152.

In logistic regression all features are chosen by the model but not all are significant.

Here I would prioritize the logistic regression model since it has a possibility to classify both 0 and 1. The problem we run into here is that the number of samples to be classified as 1 is very low in comparison to 0. This makes it difficult for a model to classify confidently and correctly a 1.

Q2.

Part 1:

a) See appendix for implementation
b) The value for h is chosen to be 1, due to it having the lowest average error on the final test fold

Part 2:

The estimated mean squared error for the neural network is approximately 2.573e-05

Appendix:

```
library(glmnet)
library(cvTools)
library(tree)
library(rpart)
library(party)
library(randomForest)
library(boot)
library(fastICA)
library(ggplot2)
library(dplyr)
library(tidyr)
library(kernlab)
library(neuralnet)
library(nnet)
library(splines2)
library(fields)
library(mgcv)
library(gam)
library(pamr)
library(klaR)
library(mboost)
library(mvtnorm)
library(e1071)
library(caret)
library(kknn)
set.seed(12345)
#Q1
myDataFunction <- function(n,p) {
  dataToReturn <- data.frame(nrow = n, ncol=p+1)
  for (i in 1:n) {
```

```
   for (j in 1:p) {
     dataToReturn[i,j] <- runif(1,0,1)
   }
  dataToReturn[i,p+1] <- ifelse(0.5*p+rnorm(1,0,0.1) > sum(dataToReturn[i,1:p]), 1,0)
  }

  return(dataToReturn)
}
missclass = function(y,y_h){
  l = length(y)
  return(1-sum(diag(table(y,y_h)))/l)
}
testacc <- rep(0,99)
set.seed(12345)
for (p in 2:100) {
  generatedTrain <- myDataFunction(100,p)
  generatedTest <- myDataFunction(200,p)
  knn_fit <- kknn(generatedTrain[,p+1]~., generatedTrain, generatedTest, k = 3)
  testacc[p-1] <- missclass(generatedTest[,p+1],knn_fit$fitted.values)
}
xseq <- seq(2,100,1)
plot(xseq,testacc)
#Q1B
set.seed(12345)
generatedTrain <- myDataFunction(100,3)
generatedTest <- myDataFunction(1000,3)
testacc <- rep(0,99)
for (k in 1:99) {
  knn_fit <- kknn(generatedTrain[,4]~., generatedTrain, generatedTest, k = k)
  testacc[k] <- missclass(generatedTest[,4],knn_fit$fitted.values)
}
```

```
plot(xseq,testacc)
```

```
#Q2
data=read.csv(file= 'C:/Users/Daniel Bissessar/Desktop/TDDE01/women.csv')

n=dim(data)[1]
data[,13] <- as.factor(data[,13])
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]

treemindev <- tree(Death~.,data=train,mindev=0.003)
testScore1 <- rep(0,50)
for (i in 2:50) {
  prunedTree <- prune.tree(treemindev, best = i)
  pred <- predict(prunedTree, newdata = test, type = 'tree')
  testScore1[i] <- deviance(pred)
}
treemindev <- tree(Death~.,data=test,mindev=0.003)
testScore2 <- rep(0,50)
for (i in 2:50) {
  prunedTree <- prune.tree(treemindev, best = i)
  pred <- predict(prunedTree, newdata = train, type = 'tree')
  testScore2[i] <- deviance(pred)
}
testScore <- (testScore1+testScore2)/2
```

```
plot(2:50, testScore[2:50], type='b', col = 2)
#graphically 4 is best
treemindev <- tree(Death~.,data=train,mindev=0.003)
bestTree <- prune.tree(treemindev, best=4)
summary(bestTree)
fit <- predict(bestTree, newdata = test, type = 'class')
confusionmatrix <- data.matrix(table(test$Death, fit))
missclass <- 1-sum(diag(confusionmatrix))/sum(confusionmatrix)
confusionmatrix
missclass

#Q2b)
logreg <- glm(Death~., family = 'binomial', data = train)
fit <- predict(logreg, newdata = test, type='response')
fit <- ifelse(fit > 0.5, 1, 0)

confusionmatrix <- data.matrix(table(test$Death, fit))
missclass <- 1-sum(diag(confusionmatrix))/sum(confusionmatrix)
confusionmatrix
missclass

#Q3
data=read.table(file= 'C:/Users/Daniel Bissessar/Desktop/TDDE01/dataKernel.txt')
n=dim(data)[1]

id=sample(1:n, floor(n*0.5))
d1=data[id,]
d2=data[-id,]

n=dim(d1)[1]
id=sample(1:n, floor(n*0.5))
```

```
d11=d1[id,]
d12=d1[-id,]

n=dim(d2)[1]
id=sample(1:n, floor(n*0.5))
d21=d2[id,]
d22=d2[-id,]

classify <- function(t, h, data){
 cc1 <- 0
 n1 <- 0
 cc2 <- 0
 n2 <- 0
 for(i in 1:nrow(data)){
  if(data[i,2]==1){
   cc1 <- cc1+dnorm(x=data[i,1]-t,sd=h)
   n1 <- n1+1
  }
  else
  {
   cc2 <- cc2+dnorm(x=data[i,1]-t,sd=h)
   n2 <- n2+1
  }
 }

 cc1 <- cc1/n1
 cc2 <- cc2/n2
 c <- 1
 if(cc1*n1/nrow(data)<cc2*n2/nrow(data))
  c <- 2
```

```
  return (c)
}
errd1 <- rep(0,8)
err <- 0
for(i in 1:nrow(d12)) {
  if(classify(d12[i,1],0.5,d11)==d12[i,2]) {
    err <- err+1
  }
}
errd1[1] <- 1-err/nrow(d12)

err <- 0
for(i in 1:nrow(d12)) {
  if(classify(d12[i,1],1,d11)==d12[i,2]) {
    err <- err+1
  }
}
errd1[2] <- 1-err/nrow(d12)

err <- 0
for(i in 1:nrow(d12)) {
  if(classify(d12[i,1],5,d11)==d12[i,2]) {
    err <- err+1
  }
}
errd1[3] <- 1-err/nrow(d12)

err <- 0
for(i in 1:nrow(d12)) {
  if(classify(d12[i,1],10,d11)==d12[i,2]) {
    err <- err+1
```

```
  }
}
errd1[4] <- 1-err/nrow(d12)

err <- 0
for(i in 1:nrow(d11)) {
  if(classify(d11[i,1],0.5,d12)==d11[i,2]) {
    err <- err+1
  }
}
errd1[5] <- 1-err/nrow(d11)

err <- 0
for(i in 1:nrow(d11)) {
  if(classify(d11[i,1],1,d12)==d11[i,2]) {
    err <- err+1
  }
}
errd1[6] <- 1-err/nrow(d11)

err <- 0
for(i in 1:nrow(d11)) {
  if(classify(d11[i,1],5,d12)==d11[i,2]) {
    err <- err+1
  }
}
errd1[7] <- 1-err/nrow(d11)

err <- 0
for(i in 1:nrow(d11)) {
  if(classify(d11[i,1],10,d12)==d11[i,2]) {
```

```
   err <- err+1
  }
}
errd1[8] <- 1-err/nrow(d11)

errd2 <- rep(0,8)
err <- 0
for(i in 1:nrow(d22)) {
  if(classify(d22[i,1],0.5,d21)==d22[i,2]) {
    err <- err+1
  }
}
errd2[1] <- 1-err/nrow(d22)

err <- 0
for(i in 1:nrow(d22)) {
  if(classify(d22[i,1],1,d21)==d22[i,2]) {
    err <- err+1
  }
}
errd2[2] <- 1-err/nrow(d22)

err <- 0
for(i in 1:nrow(d22)) {
  if(classify(d22[i,1],5,d21)==d22[i,2]) {
    err <- err+1
  }
}
errd2[3] <- 1-err/nrow(d22)

err <- 0
```

```
for(i in 1:nrow(d22)) {
  if(classify(d22[i,1],10,d21)==d22[i,2]) {
    err <- err+1
  }
}
errd2[4] <- 1-err/nrow(d22)

err <- 0
for(i in 1:nrow(d21)) {
  if(classify(d21[i,1],0.5,d22)==d21[i,2]) {
    err <- err+1
  }
}
errd2[5] <- 1-err/nrow(d21)

err <- 0
for(i in 1:nrow(d21)) {
  if(classify(d21[i,1],1,d22)==d21[i,2]) {
    err <- err+1
  }
}
errd2[6] <- 1-err/nrow(d21)

err <- 0
for(i in 1:nrow(d21)) {
  if(classify(d21[i,1],5,d22)==d21[i,2]) {
    err <- err+1
  }
}
errd2[7] <- 1-err/nrow(d21)
```

```
err <- 0
for(i in 1:nrow(d21)) {
  if(classify(d21[i,1],10,d22)==d21[i,2]) {
    err <- err+1
  }
}
errd2[8] <- 1-err/nrow(d21)
```

```
err1 <- c((errd1[1]+errd1[5])/2,(errd1[2]+errd1[6])/2,(errd1[3]+errd1[7])/2,(errd1[4]+errd1[8])/2)
err2 <- c((errd2[1]+errd2[5])/2,(errd2[2]+errd2[6])/2,(errd2[3]+errd2[7])/2,(errd2[4]+errd2[8])/2)
#for fold1 h=0.5 gives best result here
#for fold2 h=1 gives the best result here

err <- 0
for(i in 1:nrow(d2)) {
  if(classify(d2[i,1],0.5,d1)==d2[i,2]) {
    err <- err+1
  }
}
errfold1 <- 1-err/nrow(d2)

err <- 0
for(i in 1:nrow(d1)) {
  if(classify(d1[i,1],1,d2)==d1[i,2]) {
    err <- err+1
  }
}
errfold2 <- 1-err/nrow(d1)
```

```
#Q4
set.seed(1234567890)

Var <- runif(50,0,10)
tr <- data.frame(Var, Sin=sin(Var))
tr1 <- tr[1:25,]
tr2 <- tr[26:50,]

winit <- runif(10,-1,1)
nn <- neuralnet(formula=Sin~Var, data=tr1, hidden = 10, startweights = winit, threshold = 0.001)
predictions1 <- predict(nn,tr2)
MSEtr1 <- 0
for (i in nrow(predictions1)) {
  MSEtr1 <- (predictions1[i]-tr2[i,2])^2
}
MSEtr1 <- MSEtr1/nrow(predictions1)

winit <- runif(10,-1,1)
nn <- neuralnet(formula=Sin~Var, data=tr2, hidden = 10, startweights = winit, threshold = 0.001)
predictions2 <- predict(nn,tr1)
MSEtr2 <- 0
for (i in nrow(predictions2)) {
  MSEtr2 <- (predictions2[i]-tr1[i,2])^2
}
MSEtr2 <- MSEtr2/nrow(predictions2)

MSE <- (MSEtr1+MSEtr2)/2
```