Assignment 3

Name Student no.

Mark Qu 6.

Q1

a) - b)

```
ttrain <- read.csv("data/ttrain.csv", header=T, row.names=1)
ttest <- read.csv("data/ttest.csv", header=T, row.names=1)
m <- glm(Survived ~ ., data=ttrain,family="binomial")
prob <- predict(m, ttrain, type="response")
pred <- factor(ifelse(prob < .5, "No", "Yes"))
tab <- table(pred, ttrain$Survived)
tab
```

```
##
## pred    No  Yes
##   No  1097  284
##   Yes   96  284
```

```
tab[1,2]/sum(tab[,2])
```

```
## [1] 0.5
```

```
tab[2,1]/sum(tab[,1])
```

```
## [1] 0.0804694
```

```
tab[2,2]/sum(tab[2,])
```

```
## [1] 0.7473684
```

```
mean(pred != ttrain$Survived)
```

```
## [1] 0.2157865
```

50% of survivors are mis classified.

Of those who died 8.0469405% are mis classified.

74.7368421% of predicted survivors actually survived.

Overall error rate is 0.2157865.

c)

```
prob <- predict(m, ttest, type="response")
pred <- factor(ifelse(prob < .5, "No", "Yes"))
tab <- table(pred, ttest$Survived)
tab
```

```
##
## pred   No Yes
##   No  267  78
##   Yes  30  65
```

```
tab[1,2]/sum(tab[,2])
```

```
## [1] 0.5454545
```

```
tab[2,1]/sum(tab[,1])
```

```
## [1] 0.1010101
```

```
tab[2,2]/sum(tab[2,])
```

```
## [1] 0.6842105
```

```
mean(pred != ttest$Survived)
```

```
## [1] 0.2454545
```

54.5454545% of survivors are mis classified.

Of those who died 10.1010101% are mis classified.

68.4210526% of predicted survivors actually survived.

Overall error rate is 0.2454545.

Q2

Recall that the normal probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ^2 is the variance.

Hint: recall the formula

$$P(Y = j|X = x_0) = \frac{\pi_j f_j(x_0)}{C}$$

We need to compute this probability for $j = 1 =$ yes, $x_0 = 4$, $\pi_1 = .8$.

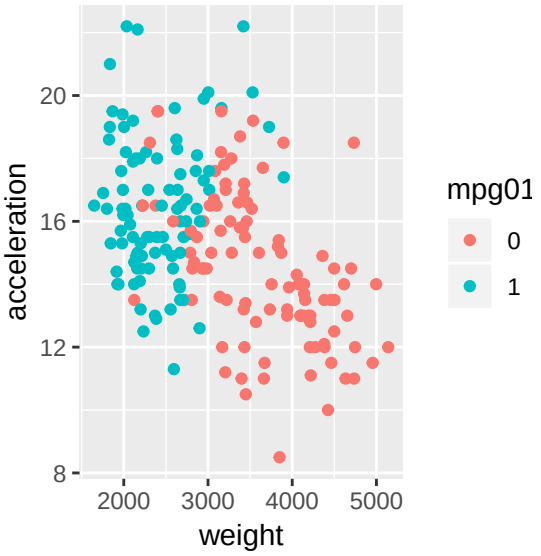
$f_1(4) = 0.0403$ and so $P(Y = 1|X = 4) = .8 \times 0.0403/C = 0.0323/C\backslash$ $f_0(4) = 0.0532$ and so $P(Y = 0|X = 4) = .2 \times 0.0532/C = 0.0106/C\backslash$ As both probabilities must sum to 1, $C = 0.0323 + 0.0106$ and $P(Y = 1|X = 4) = .753$

Q3

```
library(MASS)
library(ISLR)
library(class)
m <- median(Auto$mpg)
Auto$mpg01 <- factor(ifelse(Auto$mpg <= m, 0, 1))
set.seed(1)
s <- sample(nrow(Auto), round(.5*nrow(Auto)))
Atrain <- Auto[s,]
Atest <- Auto[-s,]
```

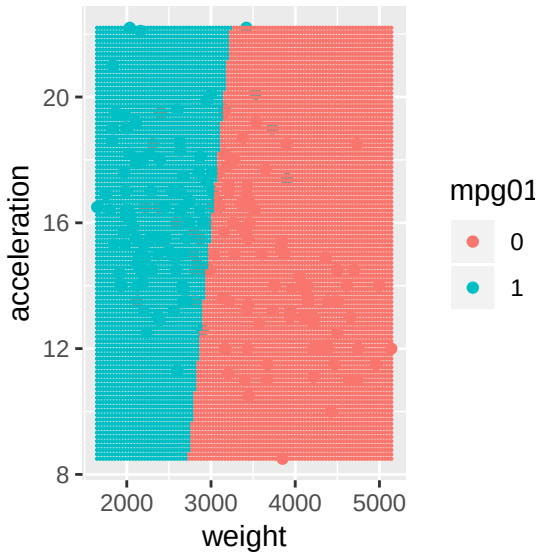
a)

```
library(ggplot2)

ggplot(data=Atrain, aes(x=weight, y=acceleration, color=mpg01))+ geom_point()
```



b)

```
f <- lda(mpg01 ~ weight+acceleration, data=Atrain)
grid <- expand.grid(
  weight = seq(min(Atrain$weight), max(Atrain$weight),length=100),
  acceleration = seq(min(Atrain$acceleration), max(Atrain$acceleration),length=100)
 )
grid$pred <- predict(f, grid)$class

ggplot(data=Atrain, aes(x=weight, y=acceleration, color=mpg01))+ geom_point()+
   geom_point(data=grid, aes(color=pred),size=.1)
```

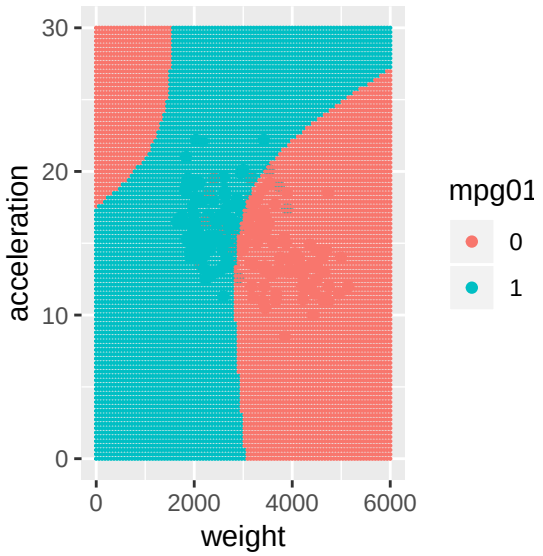



```
pred <- predict(f, Atest)$class
mean(pred!= Atest$mpg01)
```

```
## [1] 0.127551
```

c)

```
f <- qda(mpg01 ~ weight+acceleration, data=Atrain)


grid = expand.grid(weight=seq(0,6000, length=100), acceleration=seq(0,30,length=100))


grid$pred <- predict(f, grid)$class

ggplot(data=Atrain, aes(x=weight, y=acceleration, color=mpg01))+ geom_point()+
   geom_point(data=grid, aes(color=pred),size=.1)
```



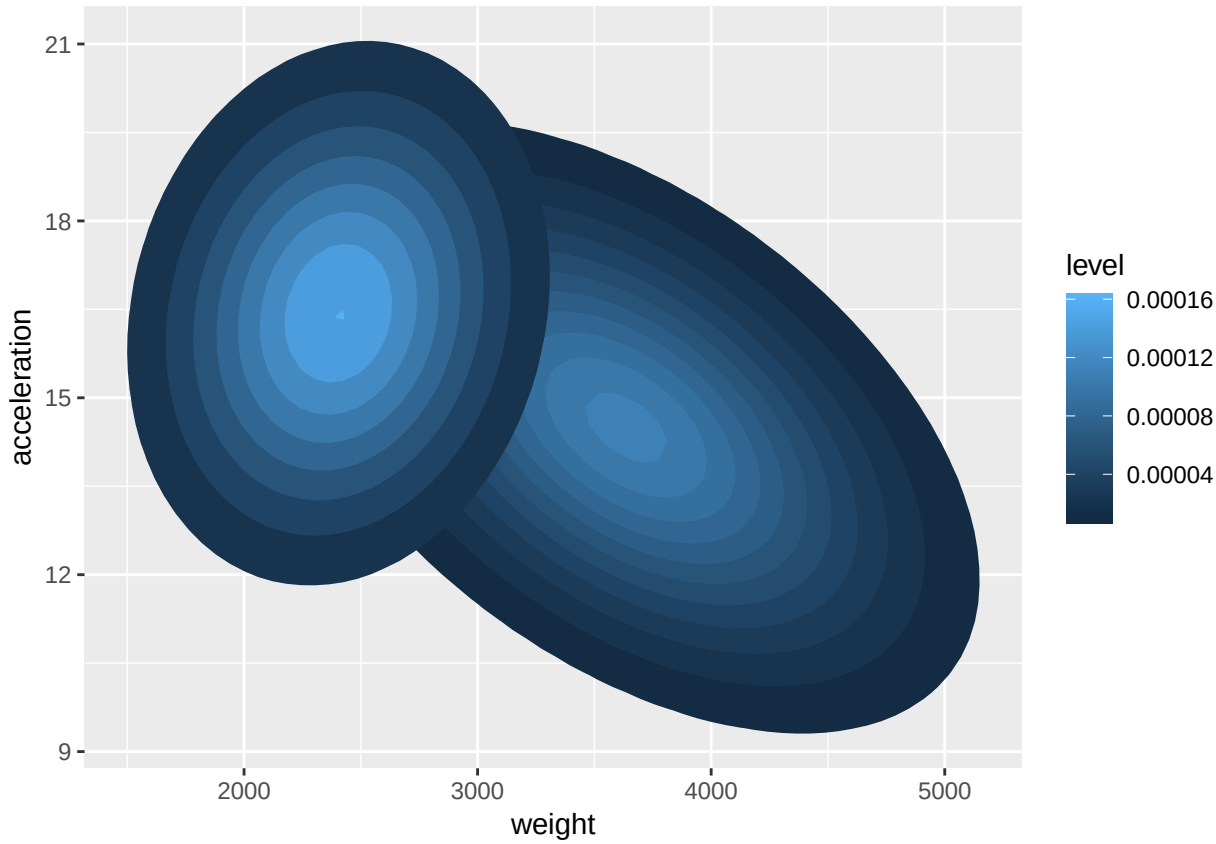

```
pred <- predict(f, Atest)$class
mean(pred!= Atest$mpg01)
```

```
## [1] 0.122449
```

QDA does better

Decision boundary

```
Atrain0 <- Atrain[Atrain$mpg01==0, 5:6]
Atrain1 <- Atrain[Atrain$mpg01==1, 5:6]

mu0 <- apply(Atrain0, 2, mean)
mu1 <- apply(Atrain1, 2, mean)
Sig0<- cov(Atrain0)
Sig1<- cov(Atrain1)
library(mvtnorm)
grid$f1 <- dmvnorm(grid[,1:2], mean=mu0, sigma=Sig0)
grid$f2 <- dmvnorm(grid[,1:2], mean=mu1, sigma=Sig1)
ggplot( data=grid, aes(x=weight, y=acceleration))+
 stat_contour(aes(z=f1,fill = ..level..),geom = "polygon")+
   stat_contour(aes(z=f2,fill = ..level..),geom = "polygon")
```




```
x <- as.matrix(grid[, 1:2])
dec <- function(x, mu, Sig){
  x <- matrix(x, nrow = 1)
  return((x-mu)%*%solve(Sig)%*%t(x-mu))
}

C <- f$prior[1] - f$prior[2] - 1/2*log(det(Sig0)) +  1/2*log(det(Sig1))
grid$pred <- C  - 1/2*(apply(x,1 , dec, mu0, Sig0) - apply(x,1 , dec, mu1, Sig1))



# ggplot(grid, aes(x=weight, y=acceleration, z= pred)) + geom_tile(aes(fill = pred))+
#     scale_fill_gradient(low="white", high="blue")

grid$pred <- as.factor(grid$pred>0)
ggplot(data=Atrain, aes(x=weight, y=acceleration, color=mpg01))+ geom_point()+
   geom_point(data=grid, aes(color=pred),size=.1)
```

d)

```
f <- lda(mpg01 ~ weight+acceleration+displacement+horsepower, data=Atrain)
pred <- predict(f, Atest)$class
mean(pred!= Atest$mpg01)
```

```
## [1] 0.1020408
```

e)

```
f <- qda(mpg01 ~ weight+acceleration+displacement+horsepower, data=Atrain)
pred <- predict(f, Atest)$class
mean(pred!= Atest$mpg01)
```

```
## [1] 0.122449
```

LDA does better

f)

```
sAuto <- scale(Auto[,3:6])
pred <-  knn(sAuto[s,], sAuto[-s,], Auto$mpg01[s], k=30)
mean(pred!= Atest$mpg01)
```

```
## [1] 0.08673469
```

```
pred <-  knn(sAuto[s,], sAuto[-s,], Auto$mpg01[s], k=5)
mean(pred!= Atest$mpg01)
```

```
## [1] 0.07142857
```

k=5 does better, and it beats lda and qda (and logistic too)

```
f <- glm(mpg01 ~ weight+acceleration+displacement+horsepower,
         data=Atrain, family="binomial")
pred <- predict(f, Atest, type="response")
pred <- factor(ifelse(pred < .5, 0,1))
mean(pred!= Atest$mpg01)
```

```
## [1] 0.1173469
```

Q4

```
set.seed(1)
x <- data.frame(Group=factor(c(rep("A",6),rep("B",4))),
                Prob = c(runif(5,.3,1), runif(5,0,.66)))
```

a)

The predicted classes are

```
x$Pred <- factor(ifelse(x$Prob > .5, "A", "B"))
x
```

```
##    Group       Prob Pred
## 1      A 0.48585606    B
## 2      A 0.56048673    A
## 3      A 0.70099735    A
## 4      A 0.93574545    A
## 5      A 0.44117735    B
## 6      A 0.59293719    A
## 7      B 0.62348568    A
## 8      B 0.43612654    B
## 9      B 0.41521527    B
## 10     B 0.04077894    B
```

b)

```
table(x$Pred, x$Group)
```

```
##
##     A B
##   A 4 1
##   B 2 3
```

```
sum(x$Group != x$Pred)/10 # error rate
```

```
## [1] 0.3
```

```
sum(x$Group == "B" & x$Pred =="A")/4 # false positive rate
```

```
## [1] 0.25
```

```
sum(x$Group == "A" & x$Pred =="A")/6 # true positive rate
```

```
## [1] 0.6666667
```

c)

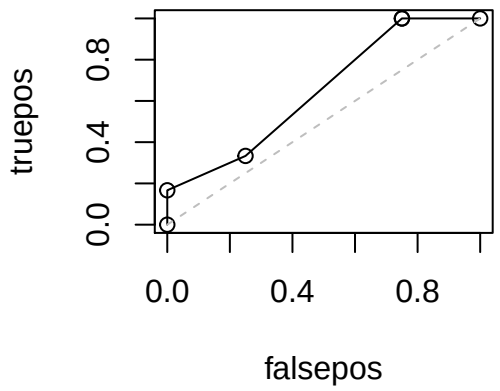
```
n <- 6
thresh <- seq(0,1, length.out=n)
falsepos <- vector("numeric", n)
```

```r
truepos <- vector("numeric", n)
for (i in 1:n){
  Pred <- factor(ifelse(x$Prob > thresh[i], "A", "B"))
  falsepos[i] <- sum(x$Group == "B" & Pred =="A")/4 # false positive rate
  truepos[i] <- sum(x$Group == "A" & Pred =="A")/6 # true positive rate
}
rbind(falsepos,truepos)
```

```
##          [,1] [,2] [,3]      [,4]      [,5] [,6]
## falsepos    1 0.75 0.75 0.2500000 0.0000000    0
## truepos     1 1.00 1.00 0.3333333 0.1666667    0
```

d)

```r
plot(falsepos, truepos)
lines(falsepos, truepos)
lines(c(0,1), c(0,1), col="grey", lty=2)
```

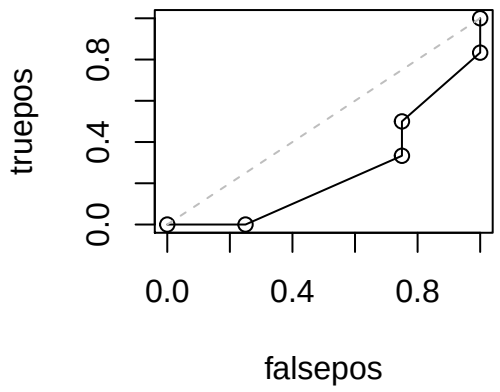


e)

```r
x$Prob <- runif(10,0,1)
n <- 6
thresh <- seq(0,1, length.out=n)
falsepos <- vector("numeric", n)
truepos <- vector("numeric", n)
for (i in 1:n){
  Pred <- factor(ifelse(x$Prob > thresh[i], "A", "B"))
  falsepos[i] <- sum(x$Group == "B" & Pred =="A")/4 # false positive rate
  truepos[i] <- sum(x$Group == "A" & Pred =="A")/6 # true positive rate
}
rbind(falsepos,truepos)
```

```
##          [,1]      [,2] [,3]      [,4] [,5] [,6]
## falsepos    1 1.0000000 0.75 0.7500000 0.25    0
## truepos     1 0.8333333 0.50 0.3333333 0.00    0
```

```
plot(falsepos, truepos)
lines(falsepos, truepos)
lines(c(0,1), c(0,1), col="grey", lty=2)
```



Q5

Any sensible answer. Could subset the training data into training - validation random splits etc.

```
library(class)
library(MASS)
?Pima.tr
head(Pima.tr)
```

```
##   npreg glu bp skin  bmi   ped age type
## 1     5  86 68   28 30.2 0.364  24   No
## 2     7 195 70   33 25.1 0.163  55  Yes
## 3     5  77 82   41 35.8 0.156  35   No
## 4     0 165 76   43 47.9 0.259  26   No
## 5     0 107 60   25 26.4 0.133  23   No
## 6     5  97 76   27 35.6 0.378  52  Yes
```

```
# pairs(Pima.tr[,1:7], col = as.numeric(Pima.tr$type))


Pima.feat.tr <- scale(Pima.tr[,1:7])
means <- attr(Pima.feat.tr,"scaled:center")
  sds<- attr(Pima.feat.tr,"scaled:scale")
Pima.feat.te<- scale(Pima.te[1:7], center=means, scale=sds)

# pairs(Pima.feat.te[,1:7], col = as.numeric(Pima.te$type))

f1 <- glm(type~.,family="binomial",data=Pima.tr)
summary(f1)
```

```
##
## Call:
## glm(formula = type ~ ., family = "binomial", data = Pima.tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9830  -0.6773  -0.3681   0.6439   2.3154
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.773062   1.770386  -5.520 3.38e-08 ***
## npreg        0.103183   0.064694   1.595  0.11073
## glu          0.032117   0.006787   4.732 2.22e-06 ***
## bp          -0.004768   0.018541  -0.257  0.79707
## skin        -0.001917   0.022500  -0.085  0.93211
## bmi          0.083624   0.042827   1.953  0.05087 .
## ped          1.820410   0.665514   2.735  0.00623 **
## age          0.041184   0.022091   1.864  0.06228 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 178.39  on 192  degrees of freedom
```

```
## AIC: 194.39
##
## Number of Fisher Scoring iterations: 5
```

```
f1b<-glm(type~.,family="binomial",data=Pima.tr[, -c(1,3,4)])
summary(f1b)
```

```
##
## Call:
## glm(formula = type ~ ., family = "binomial", data = Pima.tr[,
##     -c(1, 3, 4)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0863  -0.6727  -0.3689   0.6823   2.2092
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.971388   1.527587  -6.528 6.69e-11 ***
## glu          0.031255   0.006627   4.716 2.40e-06 ***
## bmi          0.077030   0.032251   2.388 0.016921 *
## ped          1.719794   0.656088   2.621 0.008760 **
## age          0.058603   0.017574   3.335 0.000854 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 181.08  on 195  degrees of freedom
## AIC: 191.08
##
## Number of Fisher Scoring iterations: 5
```

```
f2 <- lda(type~.,data=Pima.tr)
f3 <- qda(type~.,data=Pima.tr)

msrate <- matrix(0, 5,length(seq(0.05, 0.95, by= 0.05)))
i <- 1
for (thresh in seq(0.05, 0.95, by= 0.05)){

pred1 <- predict(f1, Pima.tr, type="response")
pred1 <- factor(ifelse(pred1 < thresh, "No", "Yes"))
tab1 <- table(Pima.tr$type, pred1)
msrate[1, i] <- 1-sum(diag(tab1))/sum(tab1)


pred1b <- predict(f1b, Pima.tr[, -c(1,3,4)], type="response")
pred1b <- factor(ifelse(pred1b < thresh, "No", "Yes"))
tab5 <- table(Pima.tr$type, pred1b)
msrate[5, i] <- 1-sum(diag(tab5))/sum(tab5)

pred2 <- predict(f2, Pima.tr)$posterior[,2]
pred2 <- factor(ifelse(pred2 < thresh, "No", "Yes"))
tab2 <- table(Pima.tr$type, pred2)
```

```
tab2
msrate[2, i] <- 1-sum(diag(tab2))/sum(tab2)


pred3 <- predict(f3, Pima.tr)$posterior[,2]
pred3 <- factor(ifelse(pred3 < thresh, "No", "Yes"))
tab3 <- table(Pima.tr$type, pred3)
msrate[3, i] <- 1-sum(diag(tab3))/sum(tab3)



pred4 <- knn(Pima.feat.tr, Pima.feat.tr, Pima.tr$type, k=i+1)
tab4 <-table(Pima.tr$type, pred4)
msrate[4, i] <- 1-sum(diag(tab4))/sum(tab4)

i<- i+ 1
}


# matplot(t(msrate[,]), type = "l")

thresh <-0.5
pred1 <- predict(f1, Pima.te, type="response")
pred1 <- factor(ifelse(pred1 < thresh, "No", "Yes"))
tab1 <- table(Pima.te$type, pred1)
1-sum(diag(tab1))/sum(tab1)
```

```
## [1] 0.1987952
```

```
pred1b <- predict(f1b, Pima.te[, -c(1,3,4)], type="response")
pred1b <- factor(ifelse(pred1b < thresh, "No", "Yes"))
tab5 <- table(Pima.te$type, pred1b)
1-sum(diag(tab5))/sum(tab5)
```

```
## [1] 0.2078313
```

```
pred2 <- predict(f2, Pima.te)$posterior[,2]
pred2 <- factor(ifelse(pred2 < thresh, "No", "Yes"))
tab2 <- table(Pima.te$type, pred2)
1-sum(diag(tab2))/sum(tab2)
```

```
## [1] 0.2018072
```

```
pred3 <- predict(f3, Pima.te)$posterior[,2]
pred3 <- factor(ifelse(pred3 < thresh, "No", "Yes"))
tab3 <- table(Pima.te$type, pred3)
1-sum(diag(tab3))/sum(tab3)
```

```
## [1] 0.2289157
```

```
pred4 <- knn(Pima.feat.tr, Pima.feat.te, Pima.tr$type, k=4)
tab4 <-table(Pima.te$type, pred4)
1-sum(diag(tab4))/sum(tab4)
```

```
## [1] 0.2409639
```

```r
pred4 <- knn(Pima.feat.tr, Pima.feat.te, Pima.tr$type, k=4)
tab4 <-table(Pima.te$type, pred4)
1-sum(diag(tab4))/sum(tab4)
```

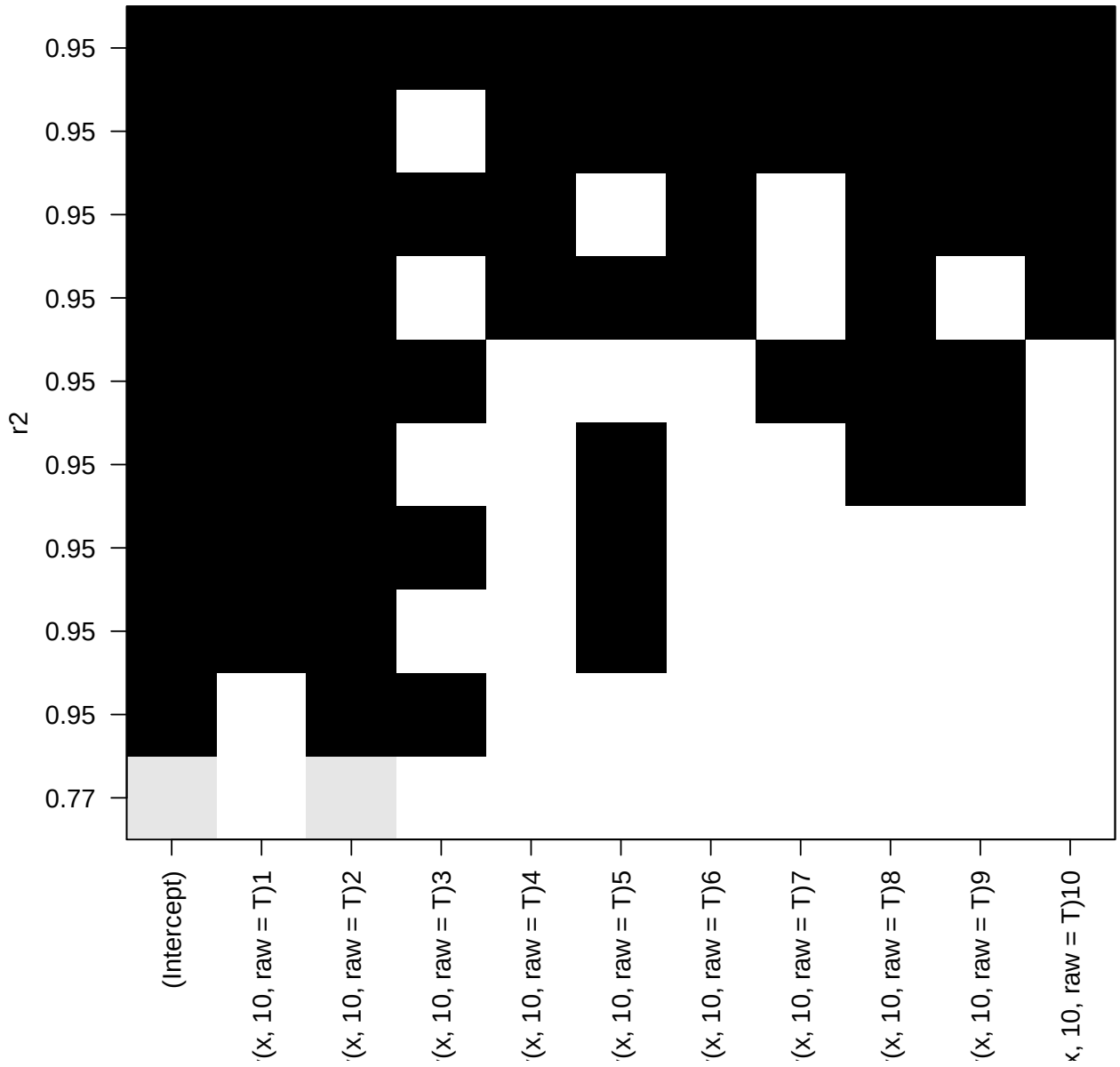
```
## [1] 0.2590361
```

Q6

```
set.seed(1)

x <-rnorm(100)
y <- 1 + .2*x+3*x^2+.6*x^3 + rnorm(100)
d <- data.frame(x=x,y=y)
```

a)

```
library(leaps)
allfits <- regsubsets(y ~ poly(x,10, raw=T), data=d,nvmax=10)
plot(allfits,scale="r2")
```



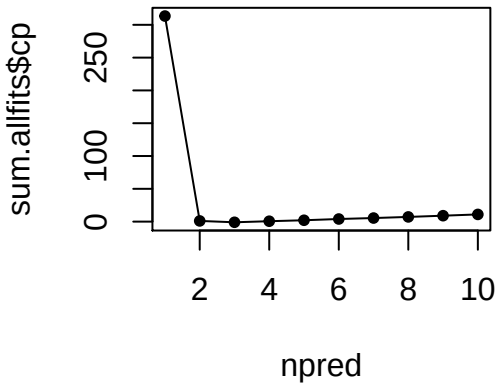
```
which(summary(allfits)$which[3,])
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##                     1                     2                     3
## poly(x, 10, raw = T)5
##                     6
```

Answer, X^2, X^3 and X^6... May depend on seed.

b)

```
npred <-1:10
sum.allfits <- summary(allfits)
plot(npred, sum.allfits$cp, pch=20)
lines(npred, sum.allfits$cp)
```




```
w <- which.min(sum.allfits$cp)
which(sum.allfits$which[w,])
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##                     1                     2                     3
## poly(x, 10, raw = T)5
##                     6
```

c)

```
fw <- regsubsets(y ~ poly(x,10,raw=TRUE), data=d,nvmax=10, method="forward")
sum.fw <- summary(fw)
w <- which.min(sum.fw$cp)
which(sum.fw$which[w,])
```

```
##              (Intercept) poly(x, 10, raw = TRUE)2 poly(x, 10, raw = TRUE)3
##                        1                        3                        4
```

d)

```
bw <- regsubsets(y ~ poly(x,10,raw=TRUE), data=d,nvmax=10, method="backward")
sum.bw <- summary(bw)
w <- which.min(sum.bw$cp)
which(sum.bw$which[w,])
```

```
##              (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2
##                        1                        2                        3
## poly(x, 10, raw = TRUE)5
##                        6
```