Homework 4 PSTAT 131, Spring 2021

Nishant Yadav and Tanner Berney

Friday June 4th 2021

```
library(tidyverse)
library(tree)
library(randomForest)
library(gbm)
library(ROCR)
library(e1071)
library(imager)
```

Question 1 a-d

Question 1a

$$(1-\frac{1}{n})^n$$

Question 1b

```
n <- 1000
(1-1/n)^n
```

```
## [1] 0.3676954
```

Question 1c

```
set.seed(100)
bootstrap_sample <- sample(1:1000, replace=TRUE)
length(unique(bootstrap_sample))
```

```
## [1] 634
```

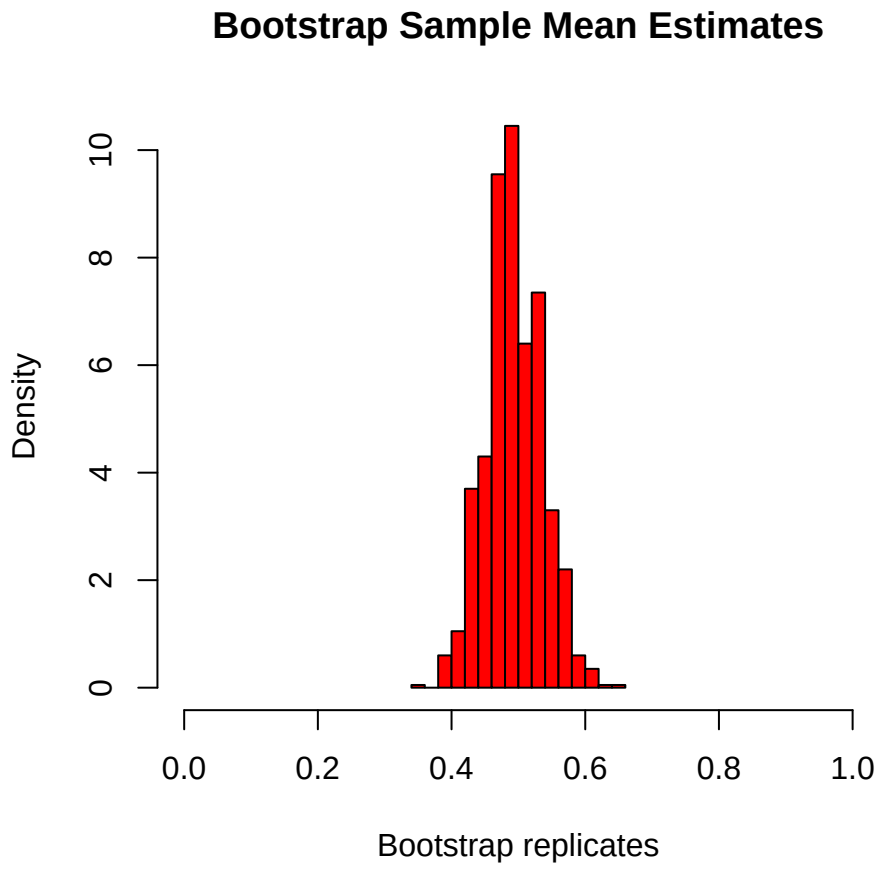
```
print(1000-length(unique(bootstrap_sample)))
```

```
## [1] 366
```

There are 634 unique values and 366 missing values. That would mean about 36% of the observations are out-of-bag.

Question 1d

```
shots <- c(rep(1,62),rep(0,64))
set.seed(100)
bootstrap_mean_estimates <- sapply(1:1000, function(i) mean(sample(shots, replace=TRUE)))
hist(bootstrap_mean_estimates, freq=FALSE, breaks=20, main="Bootstrap Sample Mean Estimates", xlim=c(0,1),
     xlab="Bootstrap replicates",col="red")
```




```
bootstrap_CI <- c(quantile(bootstrap_mean_estimates,probs=.025),quantile(bootstrap_mean_estimates,probs=.97
print(bootstrap_CI)
```

```
##      2.5%     97.5%
## 0.4126984 0.5793651
```

Regression to the mean occurs when repeated measurements are made on the same subject or unit of observation. This is due to the values being observed with random error. In general, when repeated measurements are made on the same subject and are relatively high/low, they are likely to be followed by less extreme ones near the subjects true mean. In this case, it means Curry's end-of-season FG% will be less than his percentage on 11/19 since it is a relatively high observation.

Question 2 a-e

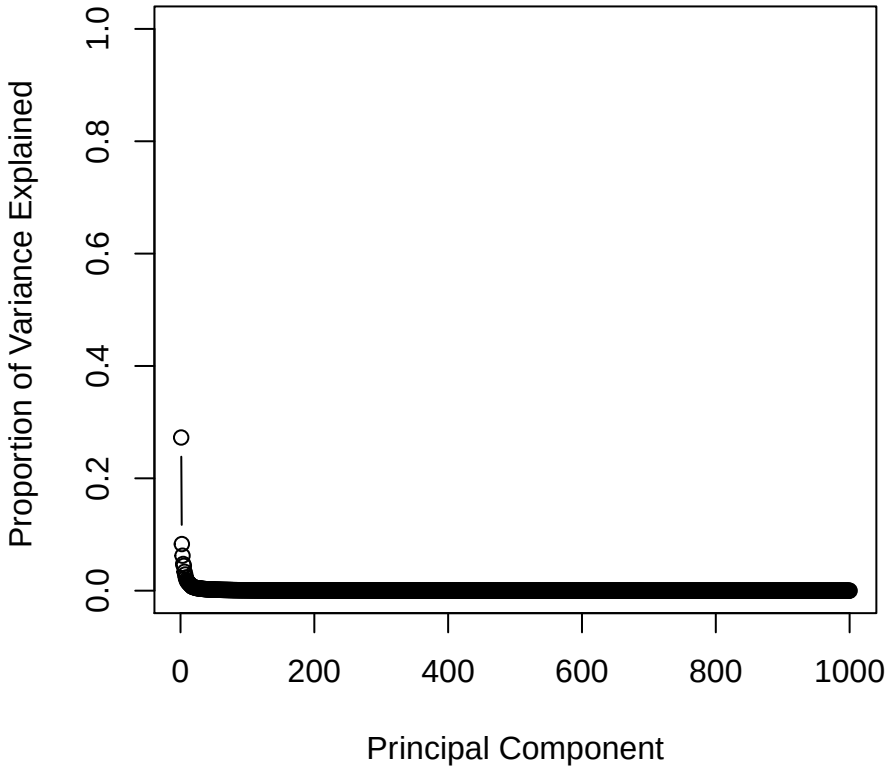
```
load("faces_array.RData")
face_mat <- sapply(1:1000, function(i) as.numeric(faces_array[, , i])) %>% t
 plot_face <- function(image_vector) {
   plot(as.cimg(t(matrix(image_vector, ncol=100))), axes=FALSE, asp=1)
}
```

Question 2a

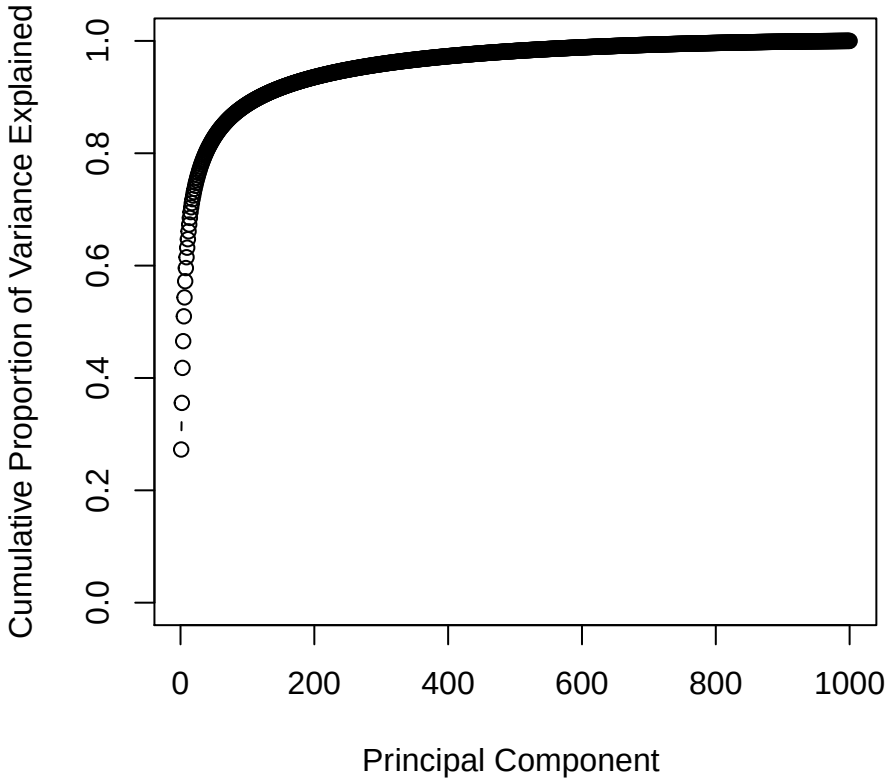
```
face_avg <- colMeans(face_mat)
plot_face(face_avg)
```

Question 2b

```
pr.out <- prcomp(face_mat,center=TRUE,scale=FALSE)
pr.var <- (pr.out$sdev)^2
pve = pr.var/sum(pr.var)
cumulative_pve <-cumsum(pve)
plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained ", ylim=c(0,1),type='b')
```




```
plot(cumulative_pve, xlab="Principal Component ",ylab=" Cumulative Proportion of Variance Explained ", ylim
```




```
sum(pve[1:5])
```

```
## [1] 0.5097349
```

You need at least 5 PCs to explain at least 50% of the total variation in the face images

Question 2c

```
par(mfrow=c(4,4),cex=.1)
for (i in 1:16){
  plot_face(pr.out$rotation[,i])
}
```

Question 2d

```
PC1_sorted <- sort(pr.out$x[,1])
largest_values <- PC1_sorted[996:1000]
smallest_values <-PC1_sorted[1:5]
max_indexs <- c(0,0,0,0,0)
min_indexs <- c(0,0,0,0,0)
for (i in 1:5){
max_indexs[i] <- which(pr.out$x[,1]==largest_values[i])
min_indexs[i] <- which(pr.out$x[,1]==smallest_values[i])
}
par(mfrow=c(1,5),cex=.1)
for (i in max_indexs){
plot_face(face_mat[i,])
}
```

```
par(mfrow=c(1,5),cex=.1)
for (i in min_indexs){
plot_face(face_mat[i,])
}
```

The first principle component seems to capture the face of the person in the image.

Question 2e

```
PC5_sorted <- sort(pr.out$x[,5])
largest_values <- PC5_sorted[996:1000]
smallest_values <-PC5_sorted[1:5]
max_indexs <- c(0,0,0,0,0)
min_indexs <- c(0,0,0,0,0)
for (i in 1:5){
max_indexs[i] <- which(pr.out$x[,5]==largest_values[i])
min_indexs[i] <- which(pr.out$x[,5]==smallest_values[i])
}
par(mfrow=c(1,5),cex=.1)
for (i in max_indexs){
plot_face(face_mat[i,])
}
```

```
par(mfrow=c(1,5),cex=.1)
for (i in min_indexs){
plot_face(face_mat[i,])
}
```

The fifth principle component seems to capture the hair at the side of the face of the individuals in the images. PC1 would be more useful feature in a face recognition model because it actually captures the face/features of a persons face.

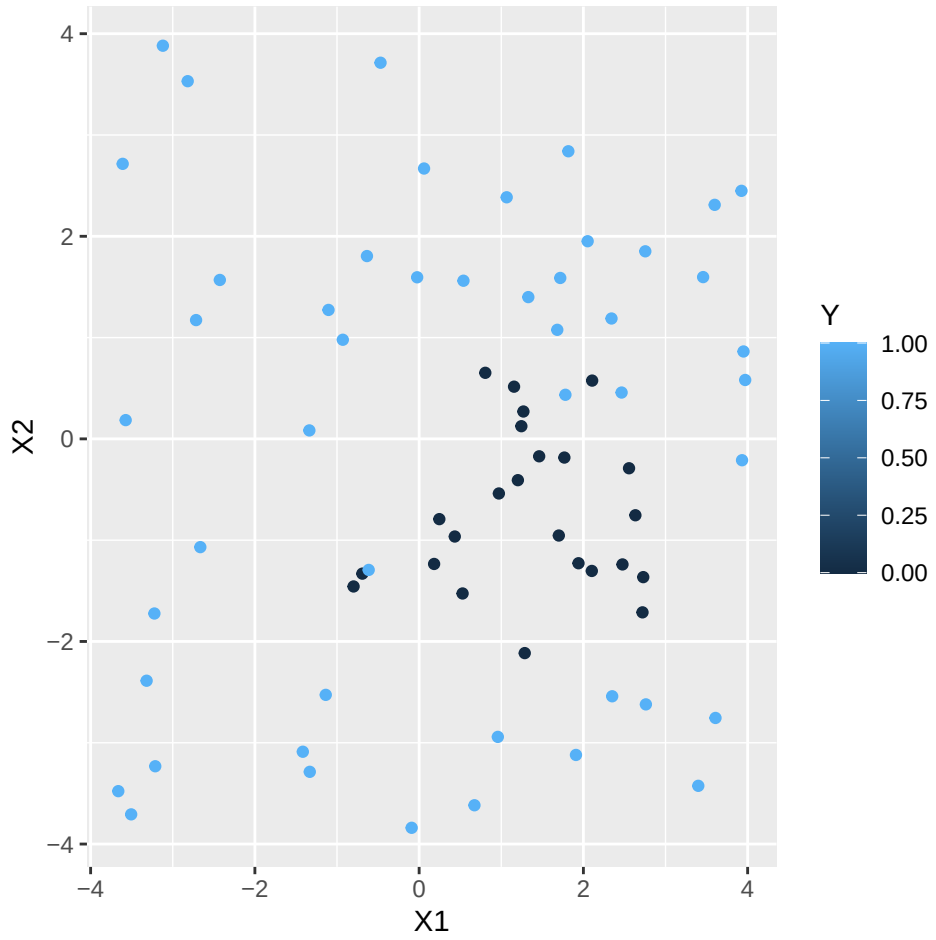
Question 3 a-f

Question 3a

```
nonldata = read_csv('nonlinear.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Z = col_double(),
##   X1 = col_double(),
##   X2 = col_double(),
##   Y = col_double()
## )
```

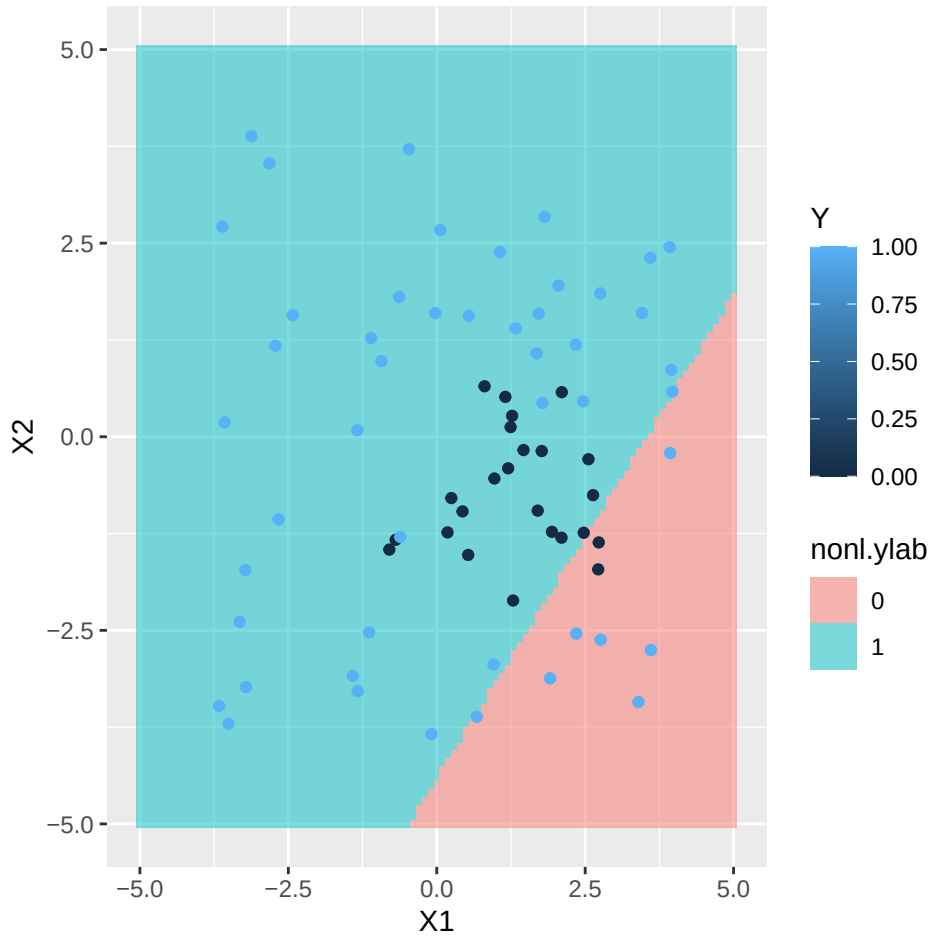
```
plot1 = ggplot(nonldata,aes(X1,X2,color =Y)) + geom_point()
plot1
```



Question 3b

```
grid1 <- expand.grid(X1=seq(-5, 5, by=0.1),  # sample points in X1
                     X2=seq(-5, 5, by=0.1))  # sample points in X2
```

```
nonl.fit = glm(Y~X1+X2, data = nonldata,family = binomial)
nonl.predict = predict(nonl.fit,grid1,type = "response")
nonl.ylab = as.factor(ifelse(nonl.predict<=0.5,0,1))
ggplot(grid1,aes(X1,X2))  + geom_raster(aes(fill=nonl.ylab),alpha = 0.5) +
  geom_point(aes(color=Y),data = nonldata)
```



Question 3c

```
nonl.fit2 <- glm(Y~poly(X1,2) + poly(X2,2) + X1*X2,data = nonldata, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(nonl.fit2)
```

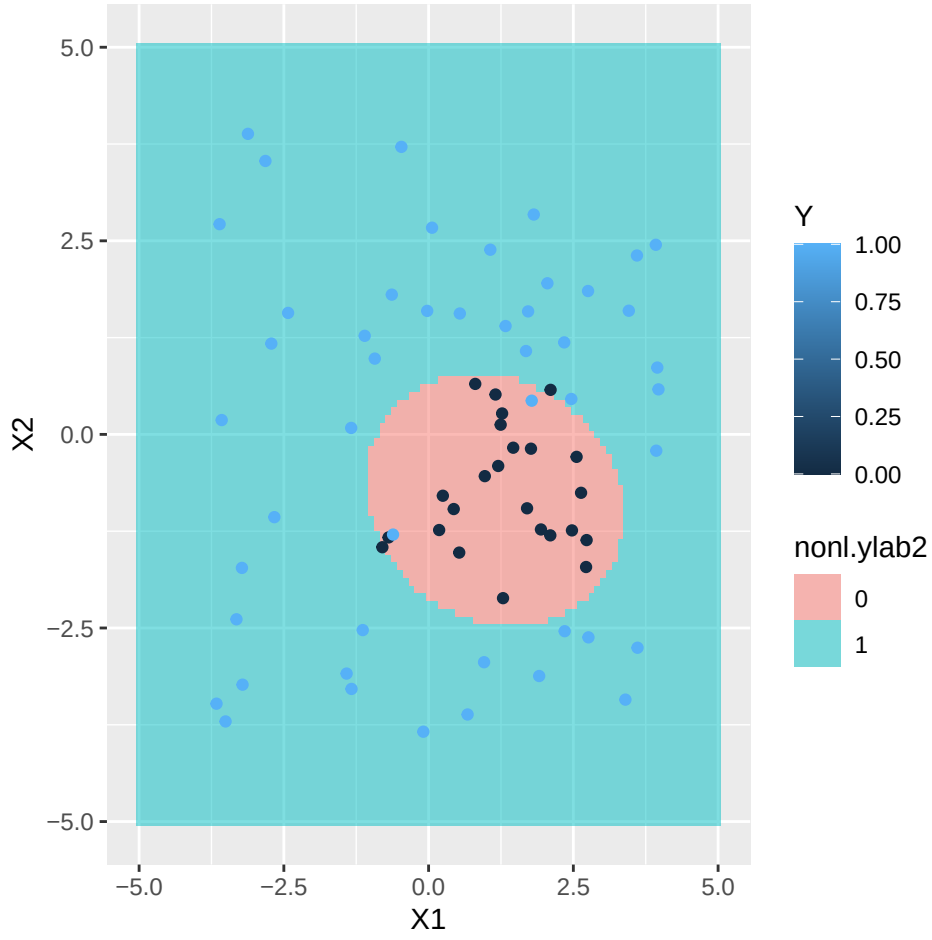
```
##
## Call:
## glm(formula = Y ~ poly(X1, 2) + poly(X2, 2) + X1 * X2, family = "binomial",
##     data = nonldata)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.39081  -0.08271   0.00000   0.00930   1.90069
##
## Coefficients: (2 not defined because of singularities)
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   11.8000     4.8086   2.454   0.0141 *
## poly(X1, 2)1 -47.2697    28.2047  -1.676   0.0937 .
## poly(X1, 2)2  57.7766    29.0429   1.989   0.0467 *
## poly(X2, 2)1  45.0707    26.9112   1.675   0.0940 .
## poly(X2, 2)2  96.3106    39.7327   2.424   0.0154 *
```

```
## X1                  NA         NA      NA       NA
## X2                  NA         NA      NA       NA
## X1:X2           0.5014     0.7369   0.680   0.4963
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 13.852  on 66  degrees of freedom
## AIC: 25.852
##
## Number of Fisher Scoring iterations: 10
```

```
nonl.predict2 <- predict(nonl.fit2,grid1,type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
nonl.ylab2 <- as.factor(ifelse(nonl.predict2<=0.5,0,1))
ggplot(grid1,aes(X1,X2)) + geom_raster(aes(fill=nonl.ylab2),alpha = 0.5) +
  geom_point(aes(color = Y),data = nonldata)
```



Question 3d

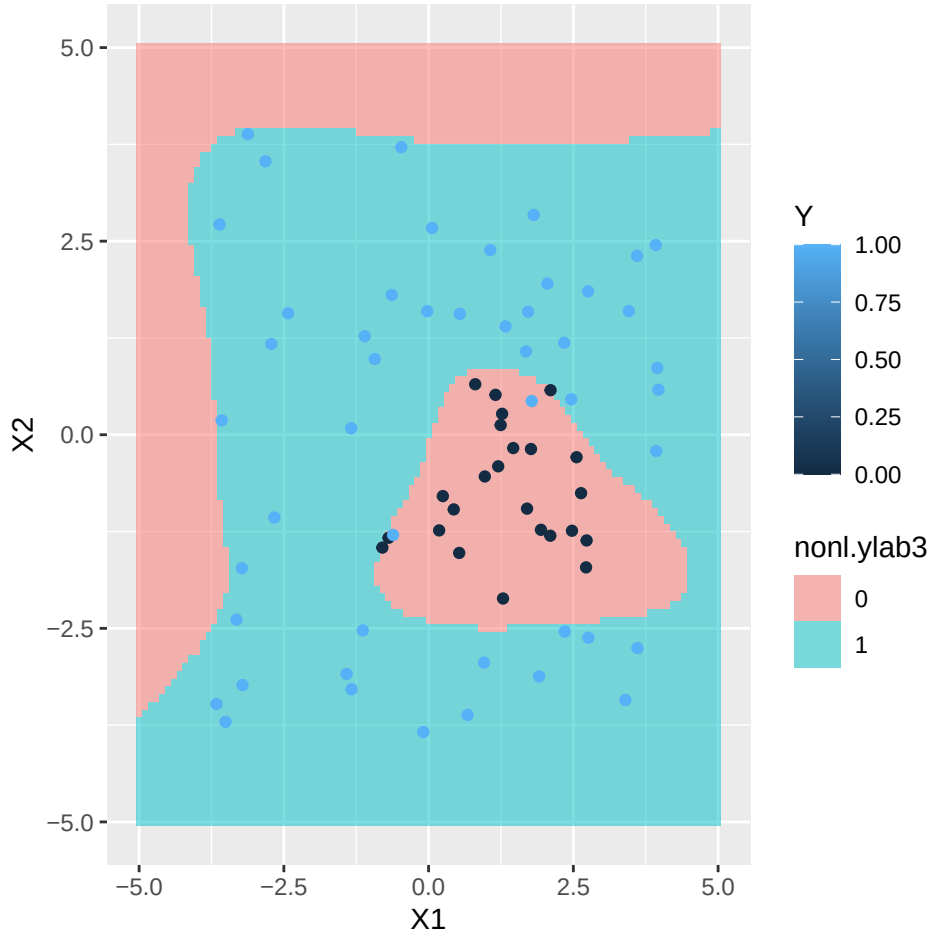
```
nonl.fit3<-glm(Y~poly(X1,5) + poly(X2,5), data = nonldata, family = binomial("logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(nonl.fit3)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 5) + poly(X2, 5), family = binomial("logit"),
##     data = nonldata)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.24411  -0.02088   0.00000   0.00078   1.85481
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)     25.42      41.06   0.619    0.536
## poly(X1, 5)1   -49.29      88.35  -0.558    0.577
## poly(X1, 5)2    25.89      36.92   0.701    0.483
## poly(X1, 5)3    36.24      60.98   0.594    0.552
## poly(X1, 5)4   -34.71      64.85  -0.535    0.593
## poly(X1, 5)5    12.65      37.72   0.335    0.737
## poly(X2, 5)1  -174.38     386.21  -0.452    0.652
## poly(X2, 5)2   266.09     480.06   0.554    0.579
## poly(X2, 5)3  -228.97     422.75  -0.542    0.588
## poly(X2, 5)4    90.75     219.09   0.414    0.679
## poly(X2, 5)5  -101.31     203.20  -0.499    0.618
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 12.494  on 61  degrees of freedom
## AIC: 34.494
##
## Number of Fisher Scoring iterations: 14
```

```
nonl.predict3<-predict(nonl.fit3,grid1,type = "response")
nonl.ylab3 <- as.factor(ifelse(nonl.predict3<=0.5,0,1))
ggplot(grid1,aes(X1,X2)) + geom_raster(aes(fill = nonl.ylab3),alpha = 0.5) +
  geom_point(aes(color = Y),data = nonldata)
```



The plot shows a finer representation compared to previous models. The unexplained red area is due to the model boundaries being tighter and more specific, causing the model to predict some values as 0.

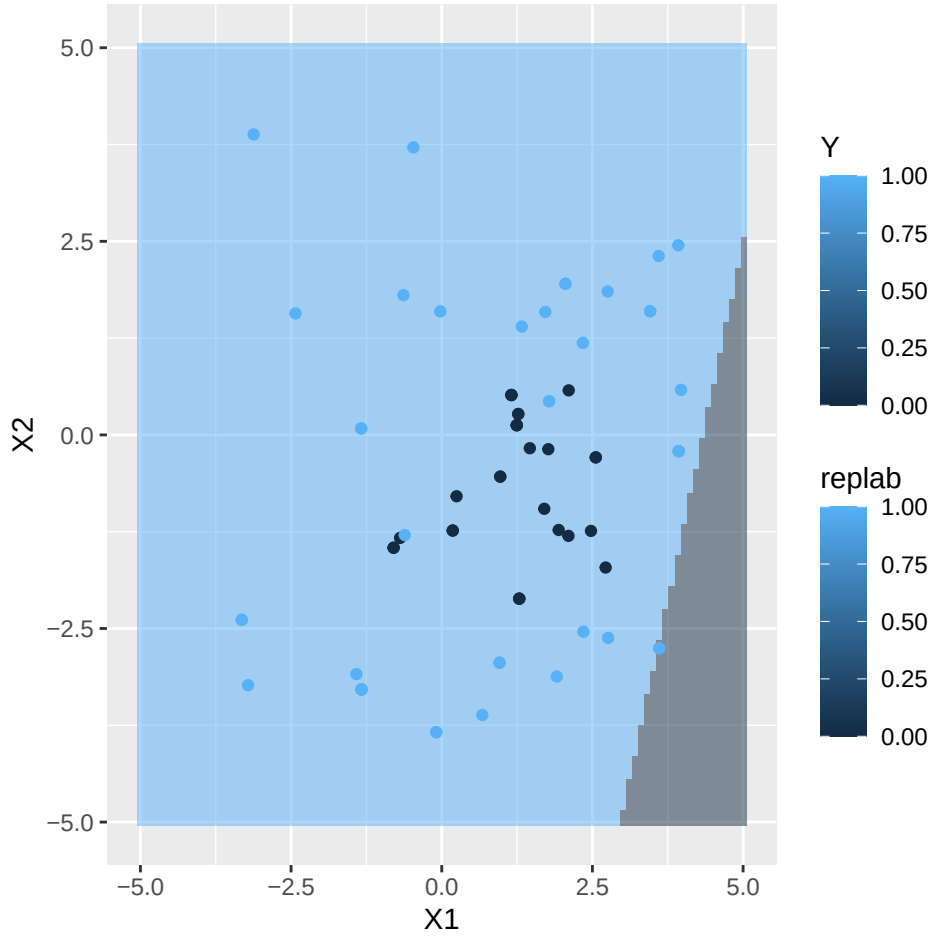
Question 3e

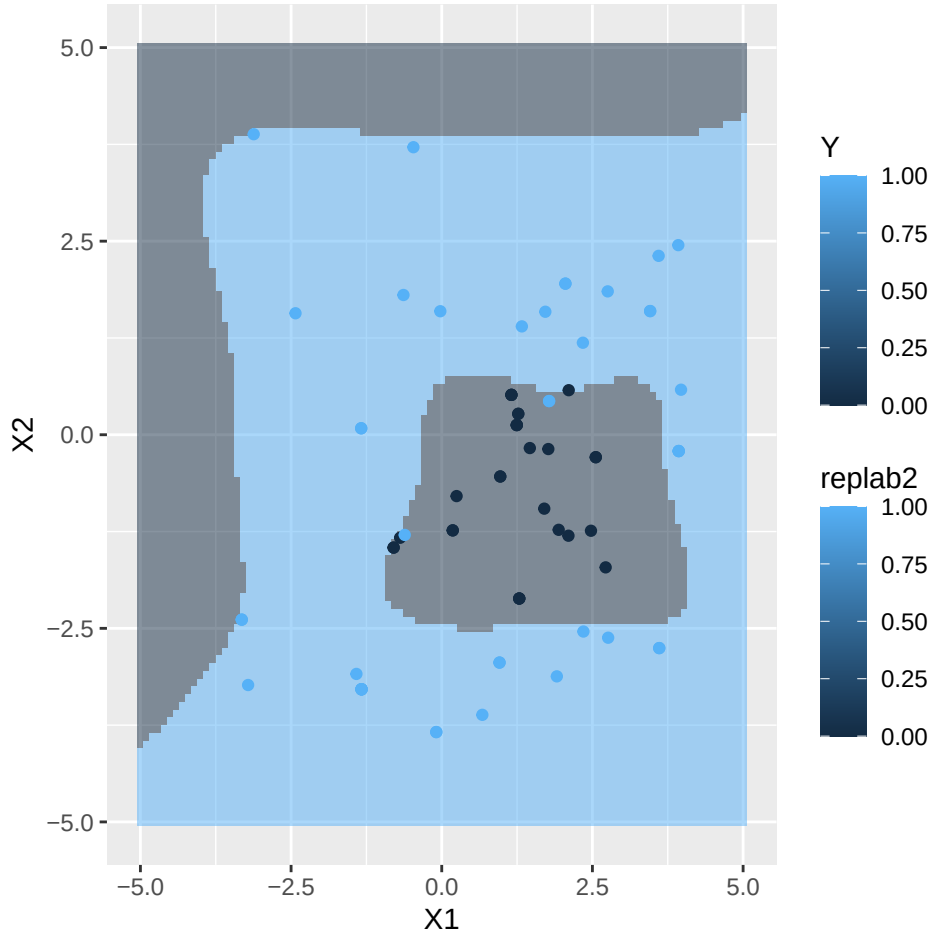
Going from linear to 2nd to 5th degree models, we can see that the 5th degree polynomial has the highest variance and lowest bias, as well as it fitting the data the best. Overfitting may occur however, if we try to predict boundaries.

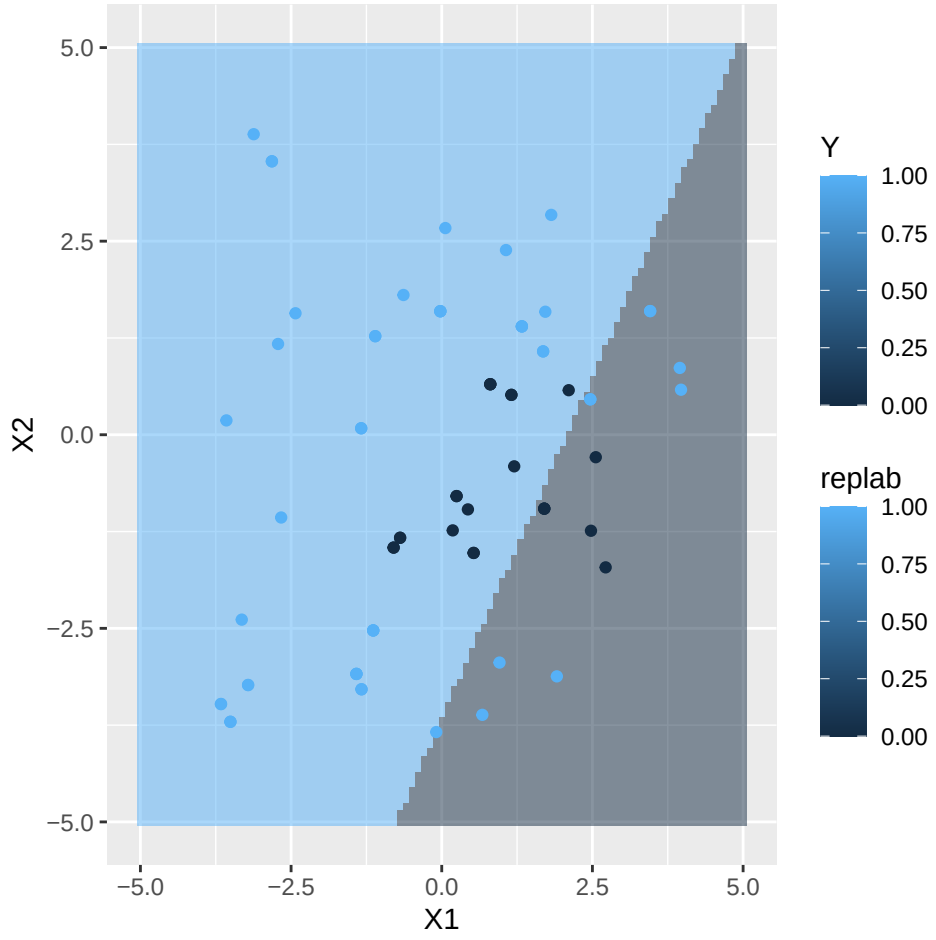
Question 3f

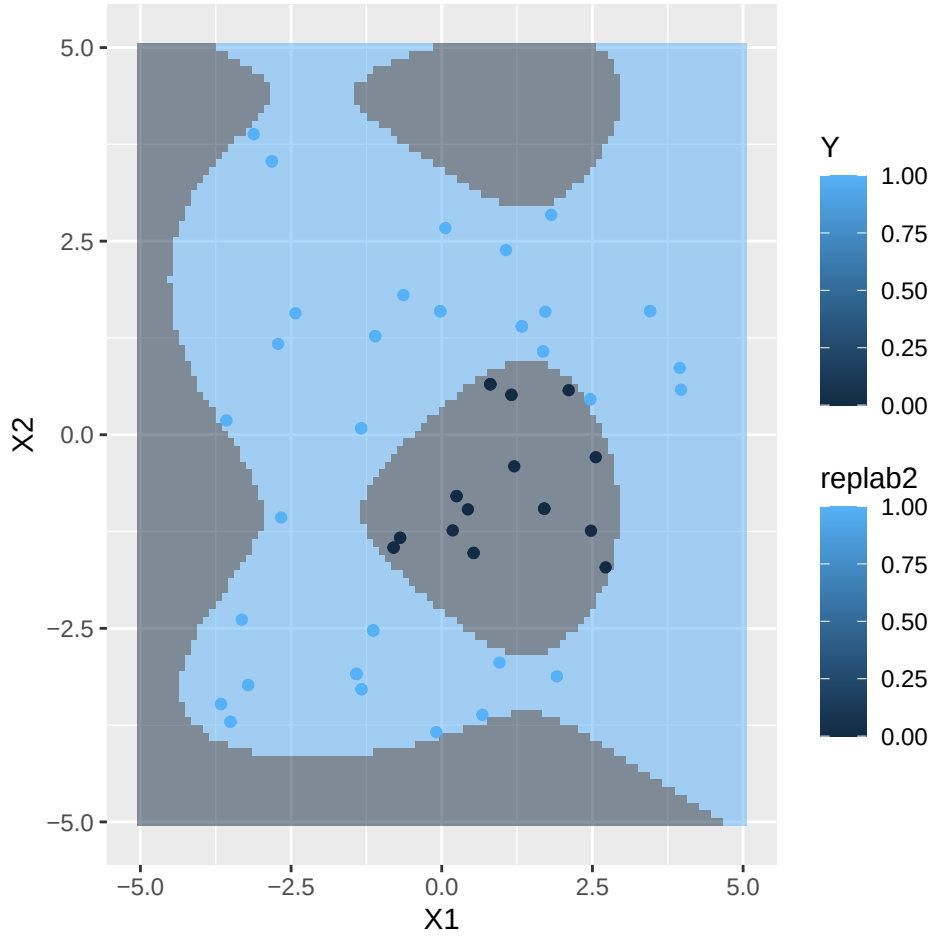
```
for (i in 1:3){
  bootsam = sample(nrow(nonldata), replace=TRUE)
  nonlrep = nonldata[bootsam,]
  fit1 = glm(Y~ X1 + X2, nonlrep, family = binomial('logit'))
  rep.pred = predict(fit1,grid1, type="response")
  replab = ifelse(rep.pred <= 0.5, 0 , 1)
  fit2 = glm(Y~poly(X1,5) + poly(X2,5), nonlrep, family = binomial("logit"))
  rep.pred2 = predict(fit2, grid1, type="response")
  replab2 = ifelse(rep.pred2 <= 0.5, 0 ,1)
  plot1 = ggplot(data = grid1, aes(x = X1, y = X2)) +
    geom_raster(aes(fill = replab), data = grid1, alpha = 0.5) +
    geom_point(aes(color = Y), data = nonlrep)
  plot2 = ggplot(data = grid1, aes(x = X1, y = X2)) +
    geom_raster(aes(fill = replab2),data = grid1, alpha = 0.5) +
    geom_point(aes(color = Y), data = nonlrep)
```

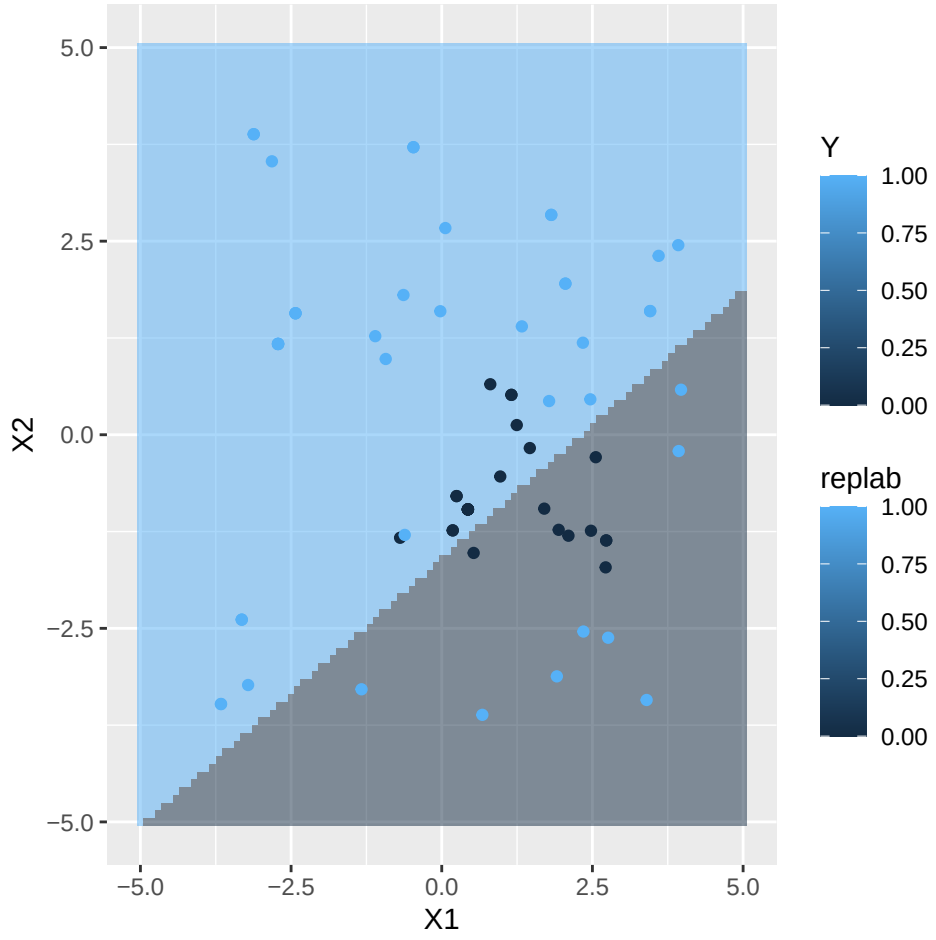
```
  print(plot1)
  print(plot2)
}
```

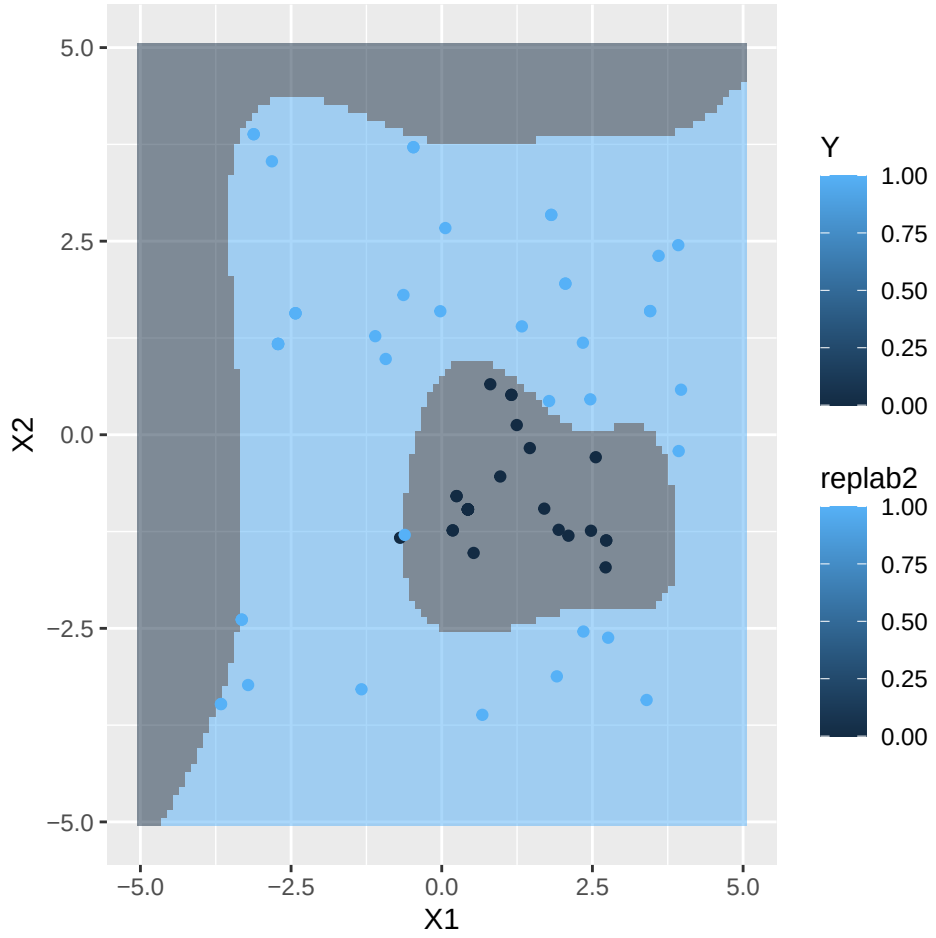



5.0
2.5
0.0
−2.5
−5.0
X2
−5.0
−2.5
0.0
2.5
5.0
X1
Y
1.00
0.75
0.50
0.25
0.00
replab2
1.00
0.75
0.50
0.25
0.00


5.0
2.5
0.0
−2.5
−5.0
X2
−5.0
−2.5
0.0
2.5
5.0
X1
Y
1.00
0.75
0.50
0.25
0.00
replab
1.00
0.75
0.50
0.25
0.00


5.0
2.5
0.0
−2.5
−5.0
X2
−5.0
−2.5
0.0
2.5
5.0
X1
Y
1.00
0.75
0.50
0.25
0.00
replab2
1.00
0.75
0.50
0.25
0.00


5.0
2.5
0.0
−2.5
−5.0
X2
−5.0
−2.5
0.0
2.5
5.0
X1
Y
1.00
0.75
0.50
0.25
0.00
replab
1.00
0.75
0.50
0.25
0.00



You can see that in 5th degree models, the graph has a high variance and the boundaries are much more different compared to linear and lower degree models, these boundaries fit the data even better.

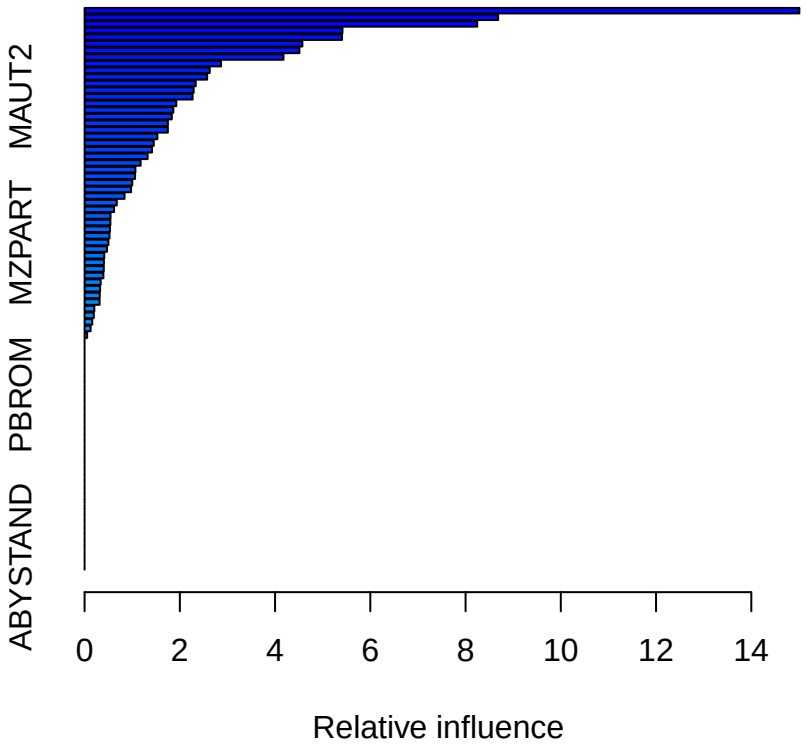
Question 4 a-d

Question 4a

```
library(ISLR)
caravan.train <- Caravan[1:1000,]
caravan.test <- Caravan[1001:5822,]
```

Question 4b

```
boostcaravan = gbm(ifelse(Purchase == "Yes", 1, 0)~.,data = caravan.train,
                   distribution = "bernoulli", shrinkage = 0.01, n.trees = 1000)
summary(boostcaravan)
```




```
##                var     rel.inf
## PPERSAUT PPERSAUT 15.00927118
## MKOOPKLA MKOOPKLA  8.68404061
## MOPLHOOG MOPLHOOG  8.24522353
## MBERMIDD MBERMIDD  5.41562346
## PBRAND     PBRAND  5.40572294
## MGODGE     MGODGE  4.57305583
## MINK3045 MINK3045  4.51086451
## ABRAND     ABRAND  4.17594041
## MOSTYPE   MOSTYPE  2.86521614
## MAUT1       MAUT1  2.62837863
## MSKC         MSKC  2.57363706
## PWAPART   PWAPART  2.33405982
## MSKA         MSKA  2.28961681
## MAUT2       MAUT2  2.26855360
## MBERARBG MBERARBG  1.92644985
## PBYSTAND PBYSTAND  1.86378528
## MGODPR     MGODPR  1.83044757
## MGODOV     MGODOV  1.74930913
## MSKB1       MSKB1  1.74846781
## MAUT0       MAUT0  1.52788194
## MINKGEM   MINKGEM  1.45197493
## MBERHOOG MBERHOOG  1.41522150
## MFWEKIND MFWEKIND  1.32358965
## MRELGE     MRELGE  1.17673240
## MRELOV     MRELOV  1.06718557
```

```
## MGODRK     MGODRK  1.06022342
## MFGEKIND MFGEKIND  1.00190977
## MINK7512 MINK7512  0.97603557
## PMOTSCO   PMOTSCO  0.83929265
## MOSHOOFD MOSHOOFD  0.67586437
## APERSAUT APERSAUT  0.61881702
## MBERARBO MBERARBO  0.54466212
## MOPLMIDD MOPLMIDD  0.54194923
## MINKM30   MINKM30  0.53278118
## MZFONDS   MZFONDS  0.52394202
## MZPART     MZPART  0.49996369
## PLEVEN     PLEVEN  0.47056591
## MINK4575 MINK4575  0.41358419
## MHHUUR     MHHUUR  0.40526167
## MINK123M MINK123M  0.40309642
## MBERBOER MBERBOER  0.39460925
## MSKB2       MSKB2  0.33749463
## MHKOOP     MHKOOP  0.32528917
## MGEMLEEF MGEMLEEF  0.31791606
## MSKD         MSKD  0.31478956
## MBERZELF MBERZELF  0.20483524
## MOPLLAAG MOPLLAAG  0.19399779
## MGEMOMV   MGEMOMV  0.16124516
## MRELSA     MRELSA  0.12734929
## MFALLEEN MFALLEEN  0.05427444
## MAANTHUI MAANTHUI  0.00000000
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PAANHANG PAANHANG  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT     PWERKT  0.00000000
## PBROM       PBROM  0.00000000
## PPERSONG PPERSONG  0.00000000
## PGEZONG   PGEZONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## PPLEZIER PPLEZIER  0.00000000
## PFIETS     PFIETS  0.00000000
## PINBOED   PINBOED  0.00000000
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
## ATRACTOR ATRACTOR  0.00000000
## AWERKT     AWERKT  0.00000000
## ABROM       ABROM  0.00000000
## ALEVEN     ALEVEN  0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
```

```
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

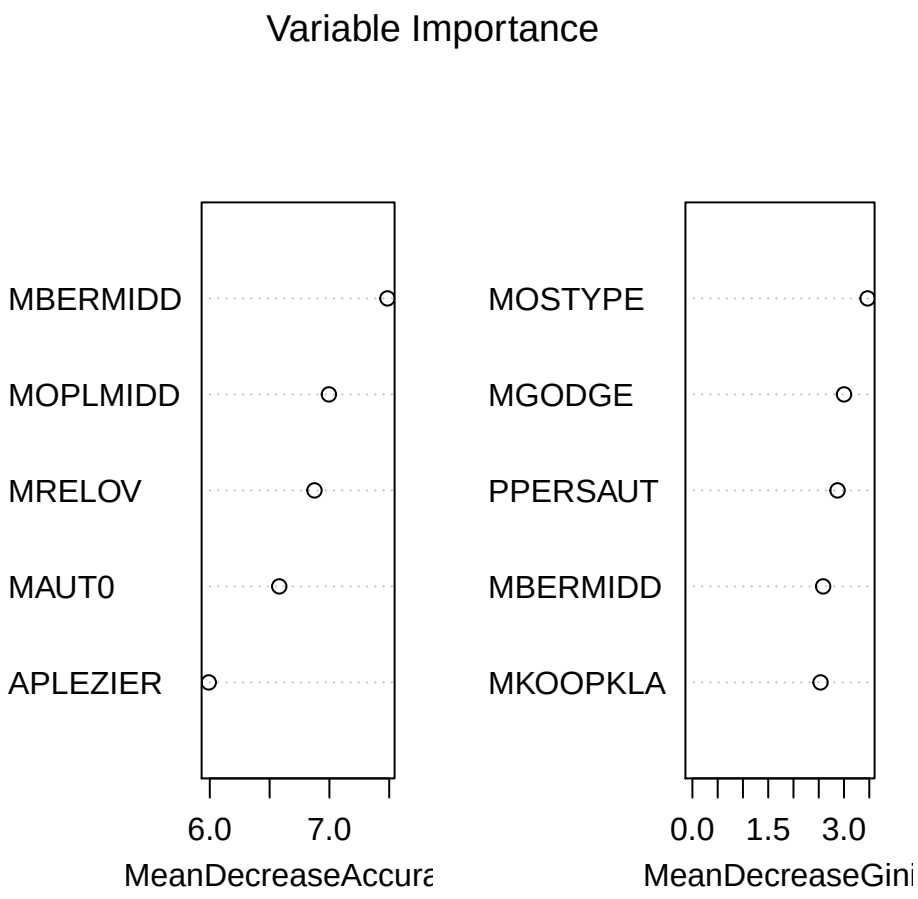
From the summary we can see that PPERSAUT, MKOOPKLA, and MOPLHOOG are the most important.

Question 4c

```
caravan.randf <- randomForest(Purchase~., data = caravan.train, importance = TRUE)
print(caravan.randf)
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = caravan.train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.1%
## Confusion matrix:
##      No Yes class.error
## No  937   4 0.004250797
## Yes  57   2 0.966101695
```

```
varImpPlot(caravan.randf, sort=T, main="Variable Importance", n.var=5)
```



The out of bag error is 6.2%. 9 Variables tried act each split. 500 trees where used and the most important variables are APLEZIER, MRELOV and MINK7512. The important variables are different in the boosting and tree models.

Question 4d

```
boostpredict = predict(boostcaravan, newdata=caravan.test, 1000, type="response")
table(pred=as.factor(ifelse(boostpredict>=.2, "Purchase", "No Purchase")),
                     true=caravan.test$Purchase)
```

```
##              true
## pred            No  Yes
##   No Purchase 4414  258
##   Purchase     119   31
```

```
rfpredict <- predict(caravan.randf, newdata=caravan.test, 500, type="prob")
table(pred=as.factor(ifelse(rfpredict[,2]>=.2, "Yes", "No")),
                     true=caravan.test$Purchase)
```

```
##      true
## pred    No  Yes
##   No  4269  246
##   Yes  264   43
```

Question 5 a-b

```
drug_use = read_csv('drug.csv',
                    col_names = c('ID','Age','Gender','Education','Country','Ethnicity',
                                  'Nscore','Escore','Oscore','Ascore','Cscore','Impulsive',
                                 'SS','Alcohol','Amphet','Amyl','Benzos','Caff','Cannabis',
                                 'Meth', 'Mushrooms', 'Nicotine', 'Semer','VSA'))
```

Question 5a

```
drug_use = drug_use %>%
  mutate(recent_cannabis_use = factor(ifelse(drug_use$Cannabis < 'CL3', "No", "Yes"),
                                              levels = c("No","Yes")))
drugsubset = drug_use %>% select(Age:SS, recent_cannabis_use)
set.seed(6724)
train = sample(nrow(drug_use), 1500)
drug.train = drugsubset[train,]
drug.test =  drugsubset[-train,]
drug.svm = svm(recent_cannabis_use~., data=drug.train, kernel = "radial", cost=1, scale=F)
svm.pred = predict(drug.svm, newdata=drug.test)
table(predict=svm.pred, truth=drug.test$recent_cannabis_use)
```

```
##        truth
## predict  No Yes
##     No  157  31
##     Yes  36 161
```

Question 5b

```
tuner = tune(svm, recent_cannabis_use~., data=drug.train, kernel="radial",
             ranges=list(cost=c(0.001, 0.01, 0.1,1,5,10,100)))
summary(tuner)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1853333
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4620000 0.03428998
## 2 1e-02 0.2280000 0.04711950
## 3 1e-01 0.1853333 0.03367235
## 4 1e+00 0.1920000 0.03410948
## 5 5e+00 0.2053333 0.03139553
## 6 1e+01 0.2113333 0.04037479
## 7 1e+02 0.2373333 0.04276840
```

```
bmodel =  tuner$best.model
summary(bmodel)
```

```
##
## Call:
## best.tune(method = svm, train.x = recent_cannabis_use ~ ., data = drug.train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##
## Number of Support Vectors:  888
##
##  ( 440 448 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

The optimal cost is $0.1 and the CV training error is 0.1866667.

```
best.prediction = predict(bmodel, newdata = drug.test)
table(best.prediction, drug.test$recent_cannabis_use)
```

```
##
```

```
## best.prediction  No Yes
##              No  159  34
##              Yes  34 158
```