Homework Assignment 4

Marissa Santiago

June 04, 2021

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

#install.packages('imager', dependencies = TRUE)
```

Problem 1

1a

Suppose original sample size is n. We are to sample *with replacement* n obsevations from the original sample. When trying to pick a saple probability that isnt j, it becomes $n-1$ observations. Therefore, the probability that observation j is not selected on any of the n draws is $(n-1)^n$ divided by the total observation replacements n^n. This gives us $(1-\frac{1}{n})^n$.

1b

```
n = 1000
bootstrap = (1-1/n)^n
bootstrap
```

```
## [1] 0.3677
```

The probability for n = 1000 is 0.3677, or 36.77%.

1c

```
#HELP IS THIS RIGHT?
set.seed(3)
numbers <- 1:1000
samples <- sample(numbers, size = 1000, replace = TRUE)
```

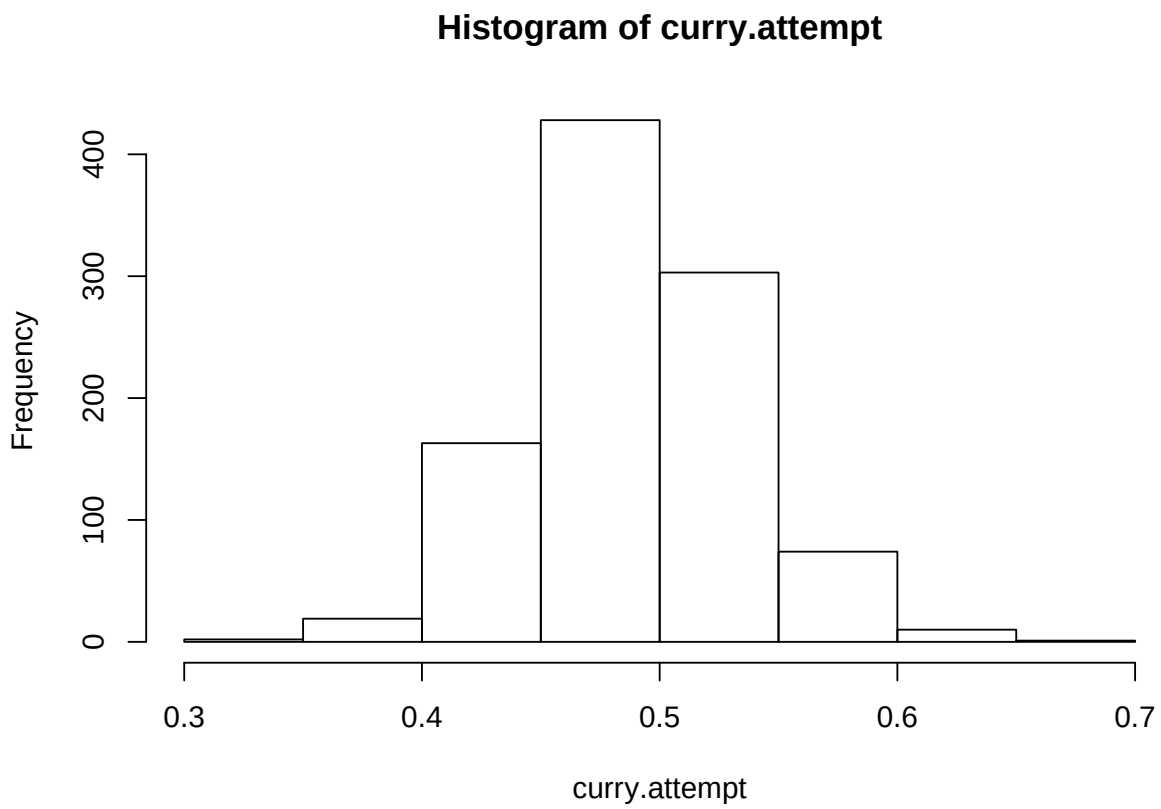
```r
num.unique <- length(unique(samples))
num.missing <- 1000 - num.unique
resample <- num.missing/1000
resample
```

```
## [1] 0.389
```

The difference of 0.022 between our empirical and theoretical result is less than 1.3% of the theoretical result, and so we determine that it is due to sampling error.

1d

```r
curry.stat <- c(rep(1,62), rep(0,64))
curry.attempt <- c()
for (i in 1:1000){
  curry.sample <- sample(curry.stat, size = 126, replace = TRUE)
  curry.made <- sum(curry.sample)
  curry.attempt[i] <- curry.made/126
}
hist(curry.attempt)
```



```r
lower <- quantile(curry.attempt, 0.025)
upper <- quantile(curry.attempt, 0.975)
c(lower, upper)
```

```
##   2.5%  97.5%
## 0.4048 0.5794
```

We are 95% confident the true average is between 0.4048 and 0.5794.

Curry's shooting percentage will go down as the season progresses because of the idea of regression to the mean. His shooting percentage is an outlier to the league's average as well as his own and as time goes on, his stats will regress more towards a mean value instead of an outlier.

Question 2

```r
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)
}
```

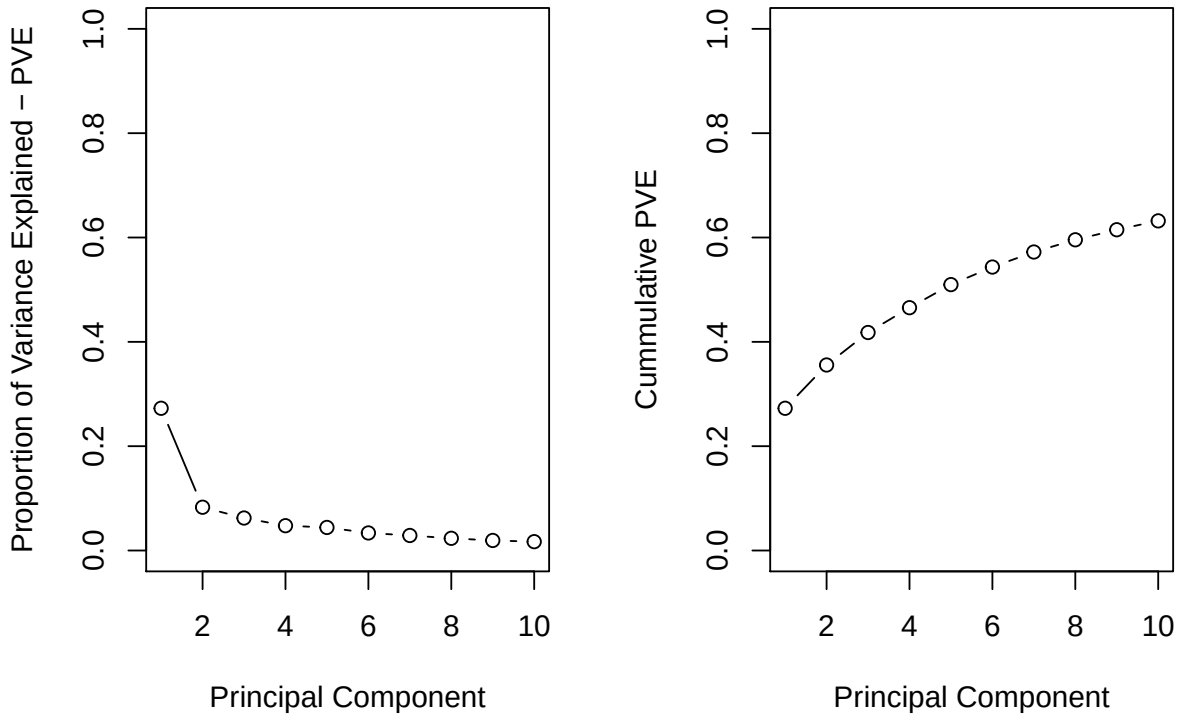
2a

```r
avg_face <- colMeans(face_mat)
plot_face(avg_face)
```

2b

```
pca <- prcomp(face_mat, center = TRUE, scale = FALSE)
```

```r
pca.var <- pca$sdev^2
pve <- pca.var/sum(pca.var)
par(mfrow=c(1, 2))
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained - PVE",
     xlim = c(1,10),
     ylim = c(0,1), type = "b")
plot(cumsum(pve), xlab = "Principal Component",
     ylab = "Cummulative PVE",
     xlim= c(1,10),
     ylim = c(0,1), type = "b")
```




```r
half_var <- 1
for (i in c(1:length(pve))){
  if (cumsum(pve)[i] < 0.5){
    half_var <- half_var + 1
  }
  else{
    break
  }
}
half_var
```

```
## [1] 5
```

We need 5 PCs to explain atleast 50% of the total variation in face images.

2c

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

2d

```
#examine faces that have the highest and lowest values for specific PCs
par(mfrow=c(1,5))
lowest.pc1 <- order(pca$x[,1], decreasing=FALSE)
highest.pc1 <- order(pca$x[,1], decreasing=TRUE)
for(i in 1:5){
  plot_face(face_mat[lowest.pc1[i],])
}
```

```
for (i in 1:5){
  plot_face(face_mat[highest.pc1[i],])
}
```

The first principle component captures the variability in background lighting that surrounds the face images. Faces that are surrounded by a darkened/black background have lower PC1 values and faces surrouned by a lighter/white background have higher PC1 values. ## 2e

```
par(mfrow=c(1,5))
lowest.pc5 <- order(pca$x[,5], decreasing=FALSE)
highest.pc5 <- order(pca$x[,5], decreasing=TRUE)
for(i in 1:5){
  plot_face(face_mat[lowest.pc5[i],])
}
```

```
for (i in 1:5){
  plot_face(face_mat[highest.pc5[i],])
}
```





The 5th priciple component captures the variability in hair length and style. Faces that have shorter hair will have lower PC5 values and are distiguishable by a black/dark border. Faces that have longer hair will have higher PC5 values and do not have a border.

Based on the results, the 5th principle component would be more useful as a feature in a face recognition model because it gives more physical details of the face image. Both length and style of hair are stronger indicators of a person's identity than a dark or light background captured by the first principle component.

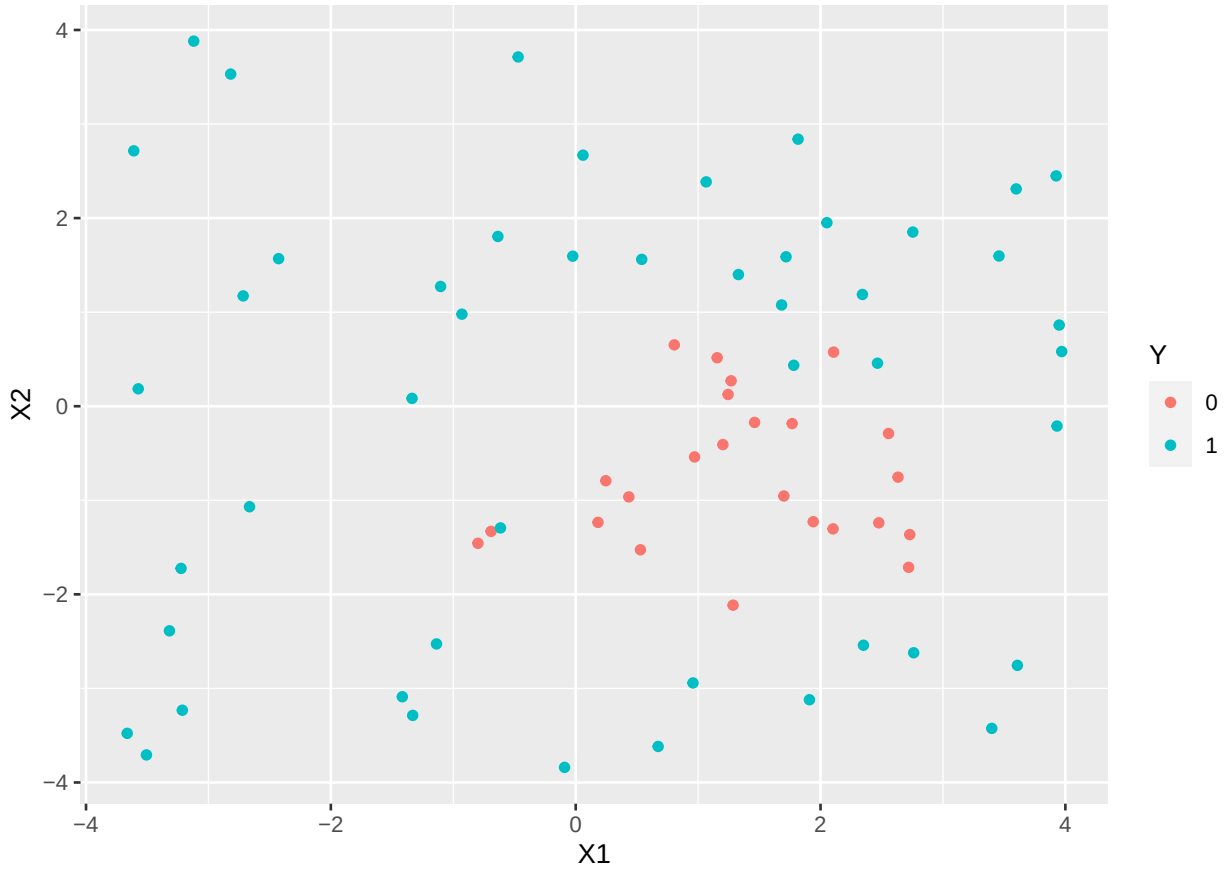
Question 3

3a

```
nonlinear <- read_csv('nonlinear.csv')
```

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

```
nonlinear$Y <- as.factor(nonlinear$Y)
ggplot(nonlinear,aes(x=X1,y=X2,color=Y))+geom_point()
```



3b

```
#A logistic regression model of Y on X1 and X2
gr <- expand.grid(X1=seq(-5, 5, by=0.1), # sample points in X1
                  X2=seq(-5, 5, by=0.1)) # sample points in X2
glm.fit <- glm(Y~X1+X2, data=nonlinear, family=binomial)
summary(glm.fit)
```

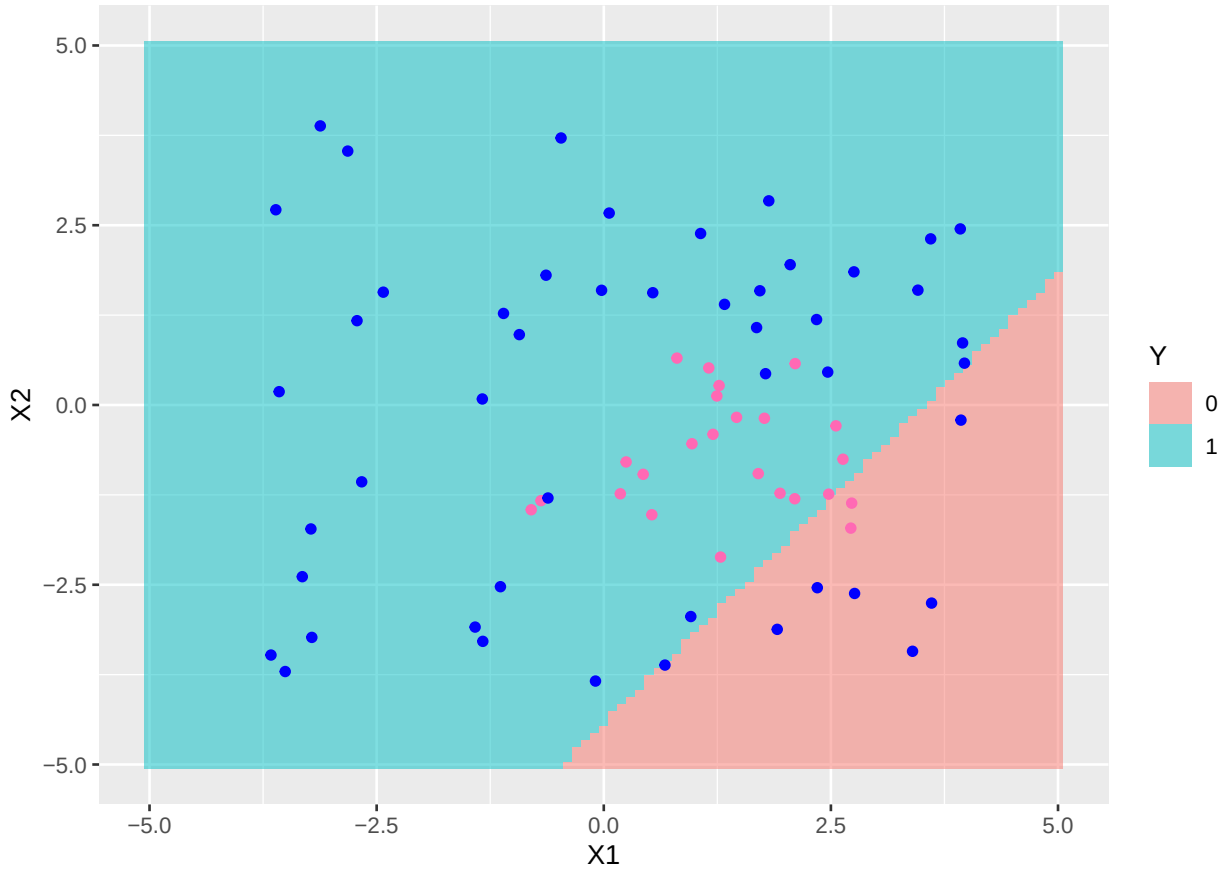
```
##
```

```
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = nonlinear)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.594  -1.248   0.626   0.915   1.511
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.022      0.314    3.26   0.0011 **
## X1            -0.289      0.136   -2.13   0.0334 *
## X2             0.232      0.144    1.62   0.1056
## ---
## 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: 84.523  on 69  degrees of freedom
## AIC: 90.52
##
## Number of Fisher Scoring iterations: 4
```

```
predict.gr = predict(glm.fit, gr, type = "response")

gr <- gr %>% mutate(Y = as.factor(ifelse(predict.gr<=0.5,0,1)))

ggplot(gr, aes(x = X1, y = X2))+geom_raster(aes(fill=Y), alpha=0.5)+geom_point(aes(nonlinear$X1,nonlinea
```



3c

```
#2nd degree polynomial of X1 and X2
nonlinear.poly2 <- glm(Y~poly(X1,X2, degree = 2), data=nonlinear,family=binomial)
```

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

```
summary(nonlinear.poly2)
```

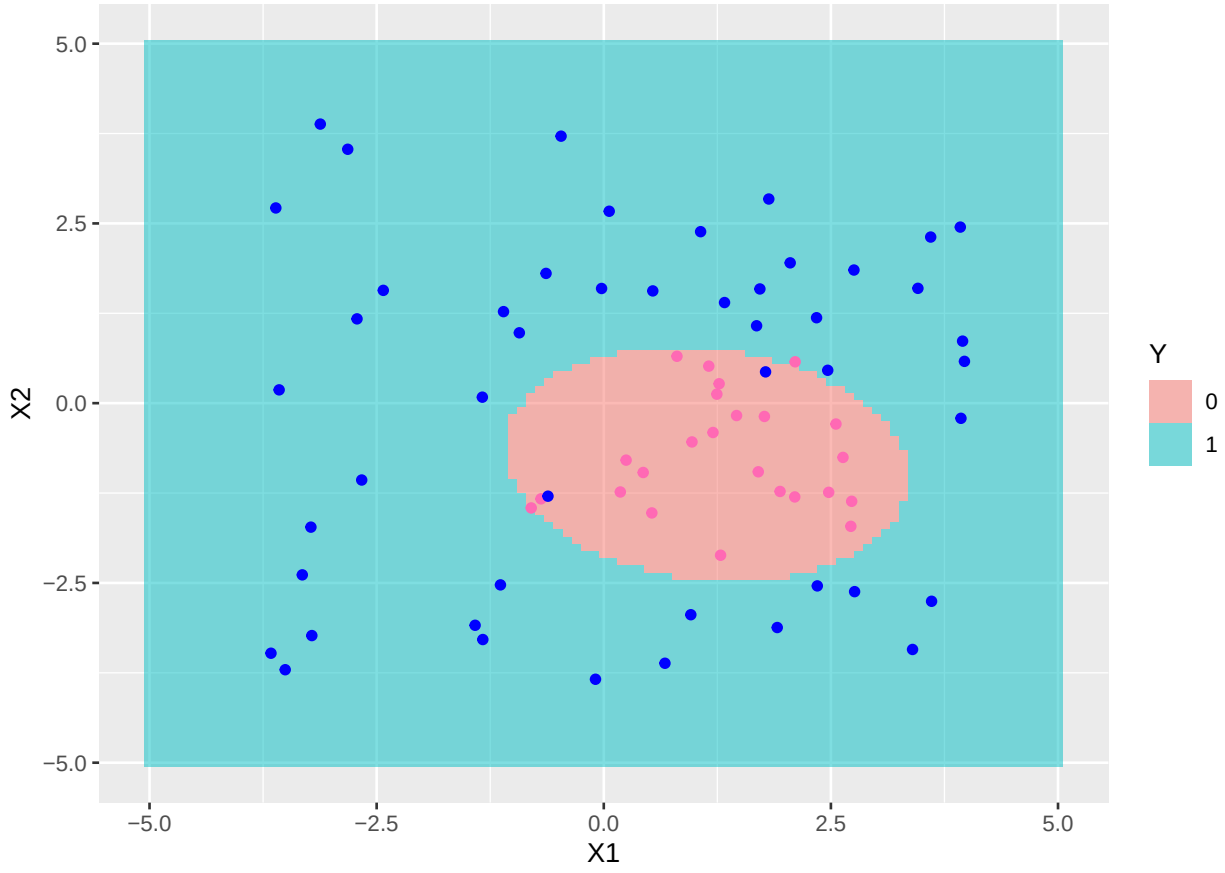
```
##
## Call:
## glm(formula = Y ~ poly(X1, X2, degree = 2), family = binomial,
##     data = nonlinear)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3908  -0.0827   0.0000   0.0093   1.9007
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     11.72       4.79    2.45    0.014 *
## poly(X1, X2, degree = 2)1.0    -49.66      27.06   -1.84    0.066 .
```

```
## poly(X1, X2, degree = 2)2.0    57.78      29.04    1.99    0.047 *
## poly(X1, X2, degree = 2)0.1    50.18      24.27    2.07    0.039 *
## poly(X1, X2, degree = 2)1.1   157.20     231.04    0.68    0.496
## poly(X1, X2, degree = 2)0.2    96.31      39.73    2.42    0.015 *
## ---
## 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: 25.85
##
## Number of Fisher Scoring iterations: 10
```

```
pred.poly2 <- predict(nonlinear.poly2,gr,type='response')

gr <- gr %>% mutate(Y = as.factor(ifelse(pred.poly2<=0.5,0,1)))

ggplot(gr, aes(x = X1, y = X2))+geom_raster(aes(fill=Y), alpha=0.5)+geom_point(aes(nonlinear$X1,nonlinea
```



3d

```
#5th degree polynomial
nonlinear.poly5 <- glm(Y~poly(X1,X2, degree = 5), data=nonlinear,family=binomial)
```

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

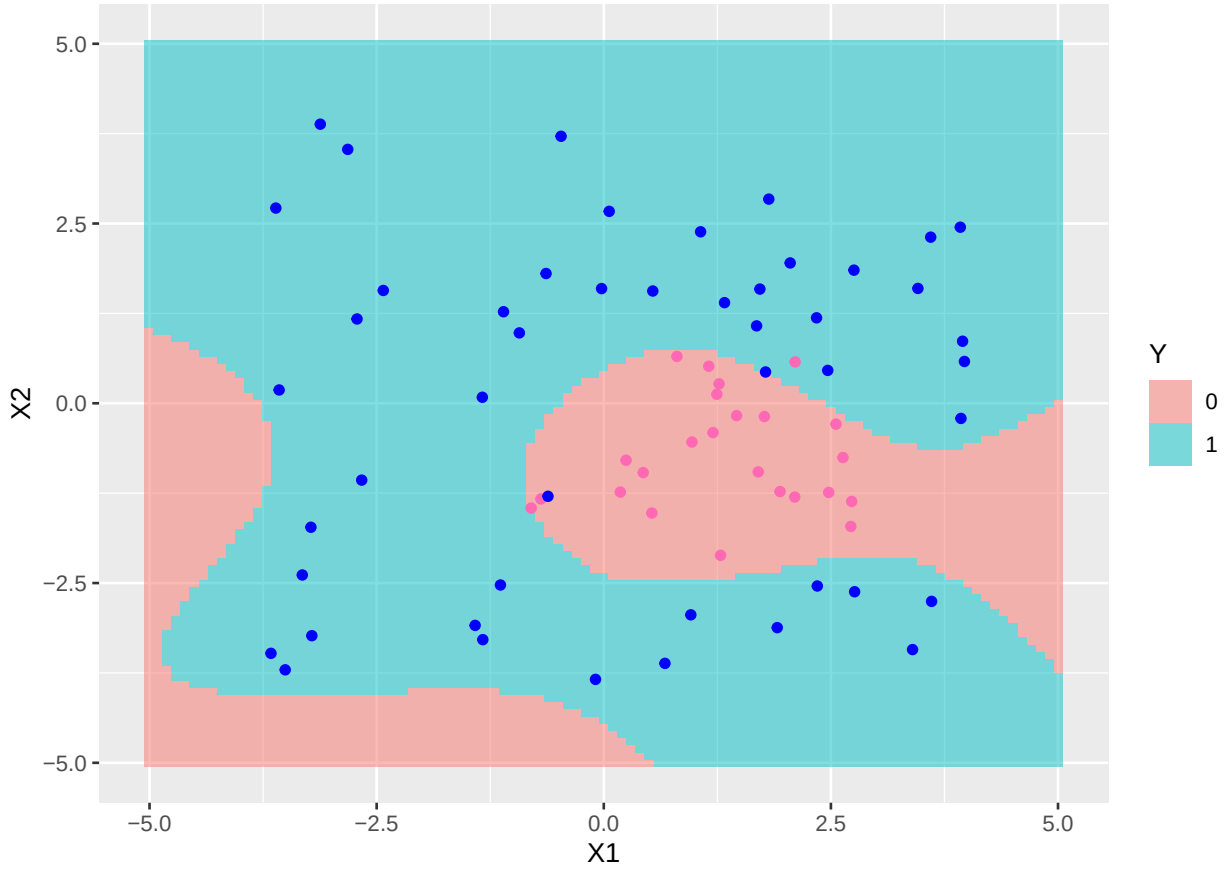
```
summary(nonlinear.poly5)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, X2, degree = 5), family = binomial,
##     data = nonlinear)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -8.49    0.00    0.00    0.00    8.49
##
## Coefficients:
##                              Estimate Std. Error   z value Pr(>|z|)
## (Intercept)                  1.36e+15   1.02e+07 133487956   <2e-16 ***
## poly(X1, X2, degree = 5)1.0  1.15e+15   9.35e+07  12251576   <2e-16 ***
## poly(X1, X2, degree = 5)2.0  5.67e+15   8.49e+07  66773905   <2e-16 ***
## poly(X1, X2, degree = 5)3.0  7.32e+14   7.88e+07   9287024   <2e-16 ***
## poly(X1, X2, degree = 5)4.0 -2.45e+15   8.23e+07 -29842754   <2e-16 ***
## poly(X1, X2, degree = 5)5.0 -5.01e+14   7.82e+07  -6408591   <2e-16 ***
## poly(X1, X2, degree = 5)0.1  4.06e+15   1.19e+08  34082507   <2e-16 ***
## poly(X1, X2, degree = 5)1.1  1.74e+16   8.97e+08  19398167   <2e-16 ***
## poly(X1, X2, degree = 5)2.1  7.02e+15   7.08e+08   9917835   <2e-16 ***
## poly(X1, X2, degree = 5)3.1  4.39e+15   7.59e+08   5788898   <2e-16 ***
## poly(X1, X2, degree = 5)4.1  1.05e+16   7.59e+08  13847620   <2e-16 ***
## poly(X1, X2, degree = 5)0.2  1.43e+16   1.53e+08  93404326   <2e-16 ***
## poly(X1, X2, degree = 5)1.2  5.96e+16   1.15e+09  51950615   <2e-16 ***
## poly(X1, X2, degree = 5)2.2  8.17e+15   7.89e+08  10350179   <2e-16 ***
## poly(X1, X2, degree = 5)3.2 -2.43e+16   8.22e+08 -29538788   <2e-16 ***
## poly(X1, X2, degree = 5)0.3 -2.90e+15   1.53e+08 -18994170   <2e-16 ***
## poly(X1, X2, degree = 5)1.3 -9.58e+15   1.19e+09  -8027227   <2e-16 ***
## poly(X1, X2, degree = 5)2.3 -1.46e+16   8.63e+08 -16931553   <2e-16 ***
## poly(X1, X2, degree = 5)0.4  1.99e+15   1.20e+08  16637650   <2e-16 ***
## poly(X1, X2, degree = 5)1.4  3.11e+16   8.76e+08  35454923   <2e-16 ***
## poly(X1, X2, degree = 5)0.5  2.50e+15   9.17e+07  27241172   <2e-16 ***
## ---
## 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: 288.349  on 51  degrees of freedom
## AIC: 330.3
##
## Number of Fisher Scoring iterations: 21
```

```
pred.poly5 <- predict(nonlinear.poly5,gr,type='response')

gr <- gr %>% mutate(Y = as.factor(ifelse(pred.poly5<=0.5,0,1)))

ggplot(gr, aes(x = X1, y = X2))+geom_raster(aes(fill=Y), alpha=0.5)+geom_point(aes(nonlinear$X1,nonlinea
```



Plot the resulting decision boundary and discuss the result. Explain the reason for any strange behvaior? -

3e

The overall magnitudes of the coefficients in the polynomial models are higher than those of the linear model. Typically, when dealing with polynomial regression, a larger p value means higher variance and lower bias. This is present when looking at the graphs of the corresponding polynomial models. In the 5th degree polynomial model, the decision boundary is overfitting in the bottom left hand corner despite not having any points in that region. This is because of the high variance that comes with a larger degree value.

Question 4

```
library(ISLR)
head(Caravan)
```

```
##   MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE MRELGE
```

```
## 1      33        1       3        2        8      0      5      1      3      7
## 2      37        1       2        2        8      1      4      1      4      6
## 3      37        1       2        2        8      0      4      2      4      3
## 4       9        1       3        3        3      2      3      2      4      5
## 5      40        1       4        2       10      1      4      1      4      7
## 6      23        1       2        1        5      0      5      0      5      0
##   MRELSA MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD MOPLLAAG MBERHOOG
## 1      0      2        1        2        6        1        2        7        1
## 2      2      2        0        4        5        0        5        4        0
## 3      2      4        4        4        2        0        5        4        0
## 4      2      2        2        3        4        3        4        2        4
## 5      1      2        2        4        4        5        4        0        0
## 6      6      3        3        5        2        0        5        4        2
##   MBERZELF MBERBOER MBERMIDD MBERARBG MBERARBO MSKA MSKB1 MSKB2 MSKC MSKD
## 1        0        1        2        5        2    1     1     2    6    1
## 2        0        0        5        0        4    0     2     3    5    0
## 3        0        0        7        0        2    0     5     0    4    0
## 4        0        0        3        1        2    3     2     1    4    0
## 5        5        4        0        0        0    9     0     0    0    0
## 6        0        0        4        2        2    2     2     2    4    2
##   MHHUUR MHKOOP MAUT1 MAUT2 MAUT0 MZFONDS MZPART MINKM30 MINK3045 MINK4575
## 1      1      8     8     0     1       8      1       0        4        5
## 2      2      7     7     1     2       6      3       2        0        5
## 3      7      2     7     0     2       9      0       4        5        0
## 4      5      4     9     0     0       7      2       1        5        3
## 5      4      5     6     2     1       5      4       0        0        9
## 6      9      0     5     3     3       9      0       5        2        3
##   MINK7512 MINK123M MINKGEM MKOOPKLA PWAPART PWABEDR PWALAND PPERSAUT PBESAUT
## 1        0        0       4        3       0       0       0        6       0
## 2        2        0       5        4       2       0       0        0       0
## 3        0        0       3        4       2       0       0        6       0
## 4        0        0       4        4       0       0       0        6       0
## 5        0        0       6        3       0       0       0        0       0
## 6        0        0       3        3       0       0       0        6       0
##   PMOTSCO PVRAAUT PAANHANG PTRACTOR PWERKT PBROM PLEVEN PPERSONG PGEZONG
## 1       0       0        0        0      0     0      0        0       0
## 2       0       0        0        0      0     0      0        0       0
## 3       0       0        0        0      0     0      0        0       0
## 4       0       0        0        0      0     0      0        0       0
## 5       0       0        0        0      0     0      0        0       0
## 6       0       0        0        0      0     0      0        0       0
##   PWAOREG PBRAND PZEILPL PPLEZIER PFIETS PINBOED PBYSTAND AWAPART AWABEDR
## 1       0      5       0        0      0       0        0       0       0
## 2       0      2       0        0      0       0        0       2       0
## 3       0      2       0        0      0       0        0       1       0
## 4       0      2       0        0      0       0        0       0       0
## 5       0      6       0        0      0       0        0       0       0
## 6       0      0       0        0      0       0        0       0       0
##   AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM
## 1       0        1       0       0       0        0        0      0     0
## 2       0        0       0       0       0        0        0      0     0
## 3       0        1       0       0       0        0        0      0     0
## 4       0        1       0       0       0        0        0      0     0
## 5       0        0       0       0       0        0        0      0     0
```

```
## 6       0        1       0       0       0        0        0      0     0
##   ALEVEN APERSONG AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED
## 1      0        0       0       0      1       0        0      0       0
## 2      0        0       0       0      1       0        0      0       0
## 3      0        0       0       0      1       0        0      0       0
## 4      0        0       0       0      1       0        0      0       0
## 5      0        0       0       0      1       0        0      0       0
## 6      0        0       0       0      0       0        0      0       0
##   ABYSTAND Purchase
## 1        0       No
## 2        0       No
## 3        0       No
## 4        0       No
## 5        0       No
## 6        0       No
```

```
data("Caravan")
```

4a

```
caravan_train <- head(Caravan, 1000)
caravan_test<- tail(Caravan, -1000)
print(dim(caravan_train))
```

```
## [1] 1000   86
```

```
print(dim(caravan_test))
```

```
## [1] 4822   86
```

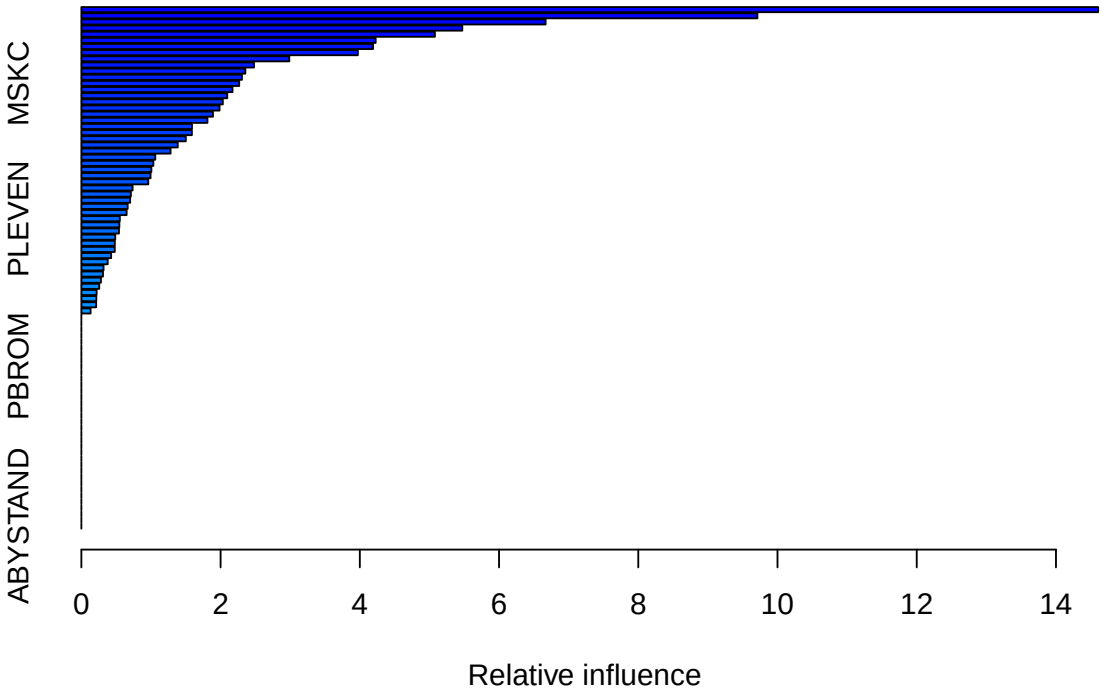
4b

```
set.seed(3)
boost.caravan = gbm(ifelse(Purchase=="Yes",1,0)~., data=caravan_train, n.trees=1000, shrinkage=0.01, di
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 50: PVRAAUT has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 71: AVRAAUT has no variation.
```

```
summary(boost.caravan)
```




```
##                var rel.inf
## PPERSAUT PPERSAUT 14.6059
## MKOOPKLA MKOOPKLA  9.7118
## MOPLHOOG MOPLHOOG  6.6685
## MBERMIDD MBERMIDD  5.4718
## PBRAND     PBRAND  5.0787
## ABRAND     ABRAND  4.2272
## MGODGE     MGODGE  4.1891
## MINK3045 MINK3045  3.9716
## MSKA         MSKA  2.9871
## PWAPART   PWAPART  2.4795
## MAUT1       MAUT1  2.3553
## MOSTYPE   MOSTYPE  2.3064
## MSKC         MSKC  2.2671
## MGODOV     MGODOV  2.1713
## MAUT2       MAUT2  2.0956
## MGODPR     MGODPR  2.0319
## MBERARBG MBERARBG  1.9852
## MBERHOOG MBERHOOG  1.8903
## MFWEKIND MFWEKIND  1.8109
## MINKGEM   MINKGEM  1.5886
## MINK7512 MINK7512  1.5872
## PBYSTAND PBYSTAND  1.5001
## MSKB1       MSKB1  1.3841
## MRELGE     MRELGE  1.2814
```

```
## MFGEKIND MFGEKIND  1.0618
## APERSAUT APERSAUT  1.0345
## MRELOV     MRELOV  1.0056
## MSKD         MSKD  0.9940
## MGODRK     MGODRK  0.9606
## MOPLMIDD MOPLMIDD  0.7361
## MHKOOP     MHKOOP  0.7111
## MHHUUR     MHHUUR  0.7005
## MAUT0       MAUT0  0.6648
## MSKB2       MSKB2  0.6500
## PLEVEN     PLEVEN  0.5525
## MZFONDS   MZFONDS  0.5458
## MBERBOER MBERBOER  0.5398
## MINKM30   MINKM30  0.4865
## MOSHOOFD MOSHOOFD  0.4805
## MGEMOMV   MGEMOMV  0.4794
## MINK4575 MINK4575  0.4276
## PMOTSCO   PMOTSCO  0.3782
## MOPLLAAG MOPLLAAG  0.3186
## MBERARBO MBERARBO  0.3101
## MZPART     MZPART  0.2809
## MGEMLEEF MGEMLEEF  0.2565
## MINK123M MINK123M  0.2189
## MBERZELF MBERZELF  0.2144
## MFALLEEN MFALLEEN  0.2124
## MRELSA     MRELSA  0.1322
## MAANTHUI MAANTHUI  0.0000
## PWABEDR   PWABEDR  0.0000
## PWALAND   PWALAND  0.0000
## PBESAUT   PBESAUT  0.0000
## PVRAAUT   PVRAAUT  0.0000
## PAANHANG PAANHANG  0.0000
## PTRACTOR PTRACTOR  0.0000
## PWERKT     PWERKT  0.0000
## PBROM       PBROM  0.0000
## PPERSONG PPERSONG  0.0000
## PGEZONG   PGEZONG  0.0000
## PWAOREG   PWAOREG  0.0000
## PZEILPL   PZEILPL  0.0000
## PPLEZIER PPLEZIER  0.0000
## PFIETS     PFIETS  0.0000
## PINBOED   PINBOED  0.0000
## AWAPART   AWAPART  0.0000
## AWABEDR   AWABEDR  0.0000
## AWALAND   AWALAND  0.0000
## ABESAUT   ABESAUT  0.0000
## AMOTSCO   AMOTSCO  0.0000
## AVRAAUT   AVRAAUT  0.0000
## AAANHANG AAANHANG  0.0000
## ATRACTOR ATRACTOR  0.0000
## AWERKT     AWERKT  0.0000
## ABROM       ABROM  0.0000
## ALEVEN     ALEVEN  0.0000
## APERSONG APERSONG  0.0000
```

```
## AGEZONG   AGEZONG  0.0000
## AWAOREG   AWAOREG  0.0000
## AZEILPL   AZEILPL  0.0000
## APLEZIER APLEZIER  0.0000
## AFIETS     AFIETS  0.0000
## AINBOED   AINBOED  0.0000
## ABYSTAND ABYSTAND  0.0000
```

The most influential variables appear to be PPERSAUT, MKOOPKLA and MOPLHOOG.

4c

```
rf.caravan = randomForest(Purchase ~ ., data=caravan_train, importance=TRUE)
rf.caravan
```

```
##
## 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    0.004251
## Yes  57   2    0.966102
```

The out-of-bag estimate of error is 6.1%. The number of variables subsampled at each split is 9. 500 trees were used to fit the data.

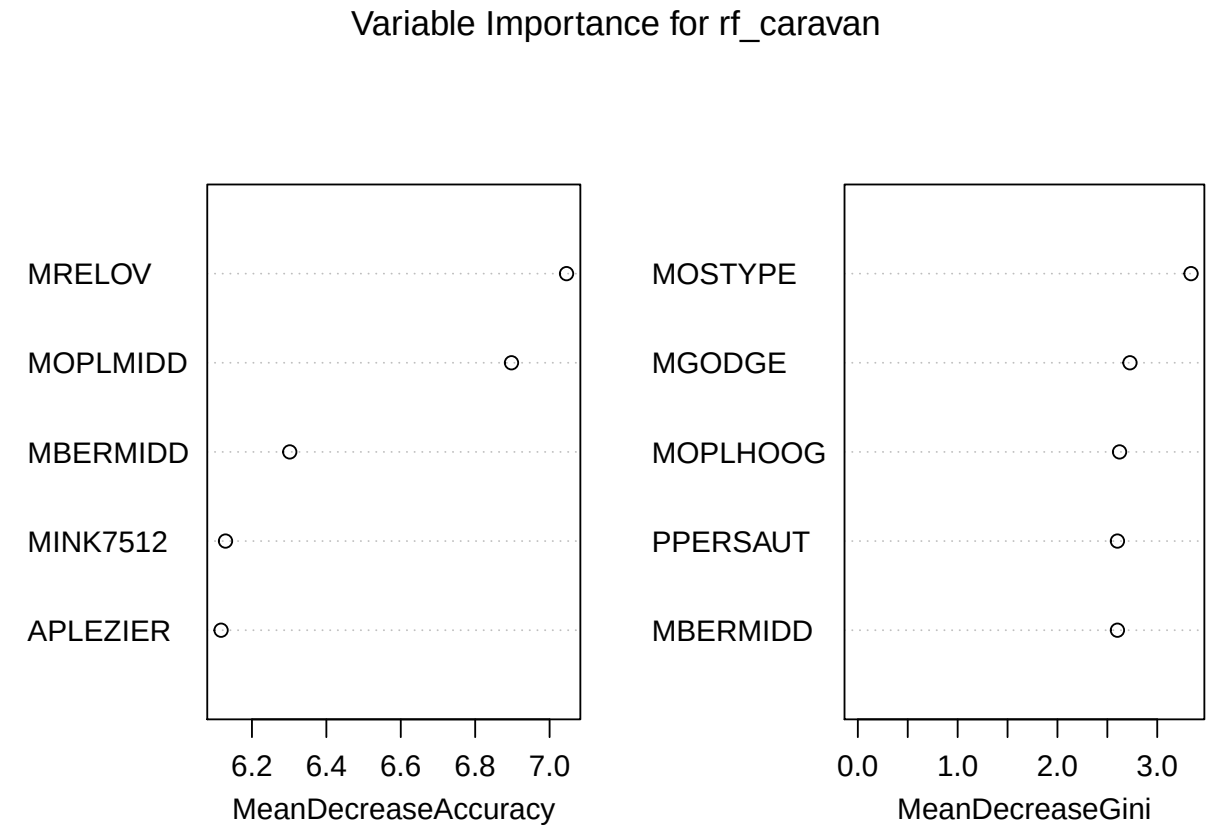
```
#look at the importance variable
importance(rf.caravan)
```

```
##                No       Yes MeanDecreaseAccuracy MeanDecreaseGini
## MOSTYPE   4.37789  2.674655             5.191585         3.338506
## MAANTHUI  0.20401 -1.085781            -0.181948         0.562141
## MGEMOMV   3.86043 -1.446109             3.476580         1.157774
## MGEMLEEF  2.96633 -0.005871             2.960229         1.097615
## MOSHOOFD  1.67481  2.988928             2.428824         1.934117
## MGODRK    2.54743 -1.400777             2.210297         1.202663
## MGODPR    3.06009  2.516898             3.742425         2.465306
## MGODOV    3.48971  1.124712             3.751623         1.707775
## MGODGE    1.94799  4.934913             3.246163         2.726442
## MRELGE    5.99612  1.044278             6.098253         2.173844
## MRELSA    2.56084 -0.983692             2.198714         1.320839
## MRELOV    7.17393 -0.431269             7.045513         1.798614
## MFALLEEN  4.61006  0.560250             4.820852         1.542376
## MFGEKIND  3.17798  0.039715             3.159480         1.983102
## MFWEKIND  3.61450  3.275506             4.263434         2.284061
## MOPLHOOG  3.20206  5.346169             4.828892         2.622144
```

```
## MOPLMIDD  6.74768  1.348328           6.897547        2.331769
## MOPLLAAG  5.29967 -1.262692           5.260369        2.011217
## MBERHOOG  2.29951 -0.420059           2.306480        1.808625
## MBERZELF  1.29833 -0.338496           1.140756        0.802259
## MBERBOER  2.01831 -0.403397           1.780232        0.427601
## MBERMIDD  5.94535  2.161375           6.301594        2.600914
## MBERARBG  4.67959  0.275968           4.718779        1.991778
## MBERARBO  4.58075  0.801875           4.826055        1.833442
## MSKA      1.33596  3.327815           2.456516        2.095991
## MSKB1     1.28160  1.830049           1.887386        2.017862
## MSKB2     2.50064 -1.480251           2.190104        1.859095
## MSKC      3.65377  2.251851           4.334427        2.290168
## MSKD      2.62407  0.483575           2.797060        1.008686
## MHHUUR    0.81301  2.006097           1.505350        2.012583
## MHKOOP    2.48655  2.836937           3.367559        2.221805
## MAUT1     1.80107 -2.383257           1.259501        1.796386
## MAUT2     3.00955  1.795238           3.424876        1.834244
## MAUT0     4.50441 -0.730393           4.424648        1.775899
## MZFONDS   3.66944 -1.158816           3.463061        1.801703
## MZPART    4.57995 -2.409122           3.913823        1.907323
## MINKM30   1.40775  0.071840           1.504562        1.850508
## MINK3045  1.70666  0.404796           1.809014        2.031397
## MINK4575  0.14210  1.164082           0.471646        1.674737
## MINK7512  5.89063  1.060731           6.128957        1.540876
## MINK123M -0.25675 -0.607675          -0.374626        0.404931
## MINKGEM   4.96166  1.012104           5.183579        1.581373
## MKOOPKLA  3.73993  4.235649           4.606043        2.401648
## PWAPART  -2.51030  5.988891          -0.365642        2.326188
## PWABEDR   0.85796  0.000000           0.857691        0.165751
## PWALAND   0.22518 -1.001002          -0.001921        0.102020
## PPERSAUT  2.06600  5.606864           3.794340        2.601023
## PBESAUT   0.00000  0.000000           0.000000        0.007800
## PMOTSCO  -2.89247  1.367929          -2.285359        0.759296
## PVRAAUT   0.00000  0.000000           0.000000        0.000000
## PAANHANG  1.69798 -1.986018           1.182340        0.296813
## PTRACTOR  0.34670 -1.415817          -0.020980        0.235005
## PWERKT    0.00000  0.000000           0.000000        0.000000
## PBROM     3.49404 -2.794955           2.940454        0.439797
## PLEVEN   -0.47249  0.768928          -0.232764        0.658160
## PPERSONG  0.00000  0.000000           0.000000        0.015332
## PGEZONG  -1.09524 -2.245475          -1.508133        0.755760
## PWAOREG   4.47388  1.862765           4.382474        0.699356
## PBRAND   -1.65630  3.281034          -0.522320        2.509488
## PZEILPL   0.00000  0.000000           0.000000        0.345318
## PPLEZIER  2.21005  5.659792           4.469482        1.778333
## PFIETS   -0.02875 -1.704640          -0.550393        0.188156
## PINBOED   0.25400  0.000000           0.257930        0.049467
## PBYSTAND  2.06187 -0.435595           1.787473        0.671613
## AWAPART  -0.60397  3.982363           0.638091        1.198307
## AWABEDR   2.01373 -1.417051           1.806928        0.101795
## AWALAND   2.54179  0.000000           2.543932        0.084690
## APERSAUT  1.31620 -1.108797           0.912106        1.964278
## ABESAUT   0.00000  0.000000           0.000000        0.005594
## AMOTSCO   0.83543 -0.964137           0.533850        0.940846
```

```
## AVRAAUT   0.00000  0.000000            0.000000         0.000000
## AAANHANG  0.65883  0.000000            0.652878         0.199162
## ATRACTOR  2.90047  0.000000            2.891692         0.070581
## AWERKT    0.00000  0.000000            0.000000         0.002667
## ABROM     3.66737 -2.799811            2.927201         0.405920
## ALEVEN   -1.64549 -1.096120           -1.724283         0.288120
## APERSONG  0.00000  0.000000            0.000000         0.005633
## AGEZONG  -0.09647 -1.001002           -0.225035         0.439905
## AWAOREG   2.57939  3.429502            3.536543         0.962611
## ABRAND    0.79783  0.203993            0.799162         1.836447
## AZEILPL   0.00000  0.000000            0.000000         0.340622
## APLEZIER  3.24770  6.494700            6.117007         1.734526
## AFIETS   -0.26689 -0.145524           -0.482189         0.220575
## AINBOED  -1.62937 -1.670503           -1.892040         0.063778
## ABYSTAND  0.79881  1.738251            1.274553         0.615205
```

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



The order of important variables is not the same for Random Forest and Boosting models because the top variable in the Boosted model is PPERSAUT, but it is below MOSTYPE and MGODGE in the random forest model.

#4d

```
#prediction of boosting model
yhat.boost = predict(boost.caravan, newdata = caravan_test, n.trees=1000, type='response')
yhat.boostprob = as.factor(ifelse(yhat.boost>=0.2, "Yes", "No"))
#cofusion matrix
boost.error = table(predicted = yhat.boostprob, true = caravan_test$Purchase)
test.boost.error = 1 - sum(diag(boost.error))/sum(boost.error)
test.boost.error
```

```
## [1] 0.07777
```

```
boost.error
```

```
##          true
## predicted   No  Yes
##       No  4413  255
##       Yes  120   34
```

```
#predition of randomforest model
yhat.rf = predict(rf.caravan, newdata = caravan_test, type='prob')
yhat.rfprob = as.factor(ifelse(yhat.rf[,"Yes"]>=0.2, "Yes", "No"))
# Confusion matrix
rf.error = table(predicted = yhat.rfprob, true = caravan_test$Purchase)
test.rf.err = 1 - sum(diag(rf.error))/sum(rf.error)
test.rf.err
```

```
## [1] 0.1035
```

```
rf.error
```

```
##          true
## predicted   No  Yes
##       No  4279  245
##       Yes  254   44
```

The fraction of the people predicted to make a purchase that do in fact make one is $\frac{44}{44+254}$ which is 0.1477, or 14.77%.

Question 5

```
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',
'Choc','Coke','Crack','Ecstasy','Heroin','Ketamine','Legalh','LSD',
'Meth', 'Mushrooms', 'Nicotine', 'Semer','VSA'))
```

5a

```
drug_use <- drug_use %>% mutate(recent_cannabis_use=as.factor(ifelse(Cannabis>="CL3", "No", "Yes")))
drug_use_subset <- drug_use %>% select(Age:SS, recent_cannabis_use)
train.indices = sample(1:nrow(drug_use_subset), 1500)
drug_use_train <- drug_use_subset[train.indices,]
drug_use_test <- drug_use_subset[-train.indices,]

#SVM
svm.fit = svm(recent_cannabis_use~Age+SS, data=drug_use_train, kernel="radial", cost=1,scale=FALSE)
drug.pred = predict(svm.fit, drug_use_test, type='class')

#Confusion Matrix
conf.test = table(predicted=drug.pred, true=drug_use_test$recent_cannabis_use)
conf.test
```

```
##          true
## predicted  No Yes
##       No  147  46
##       Yes  57 135
```

#5b

```
set.seed(3)
tune.test <- tune(svm,recent_cannabis_use~., data=drug_use_train, kernel="radial",ranges=list(cost=c(0.

bestmod <- tune.test$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = recent_cannabis_use ~ ., data = drug_use_train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##
## Number of Support Vectors:  879
##
##  ( 440 439 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
pred.tune <- predict(bestmod, newdata = drug_use_test)

#use test data
drug.err <- table(true=drug_use_test$recent_cannabis_use, predict= pred.tune)
drug.err
```

```
##      predict
## true   No Yes
##   No  169  35
##   Yes  38 143
```

```
train.drug.err = 1 - sum(diag(drug.err))/sum(drug.err)
train.drug.err
```

```
## [1] 0.1896
```

The optimal cost for this model is 0.1 and the cross validated training error is 0.1896.