PSTAT131 HW4

Huiya Li and Yifan Wang 7983851

6/2/2021

```r
knitr::opts_chunk$set(echo = T,cache=TRUE)
```

Question 1

1(a) There are n observations in bootstrap that can be picked from sample with replacement in total n^n ways. If j th obs cannot be picked, there will be (n-1)^n cases. So, probability that j will not be in bootstrap is

$$p = ((n-1)^n)/(n^n) = (1-1/n)^n$$

1(b) Plugging n=1000, p=(1-1/1000)^(1000) = 0.36769

1(c)

```r
set.seed(1)
s =  sample(1:1000, size = 1000, replace = T)
num.miss = 1000-length(unique(s))
num.miss
```

```
## [1] 370
```

```r
p.s=num.miss/1000
p.s
```

```
## [1] 0.37
```

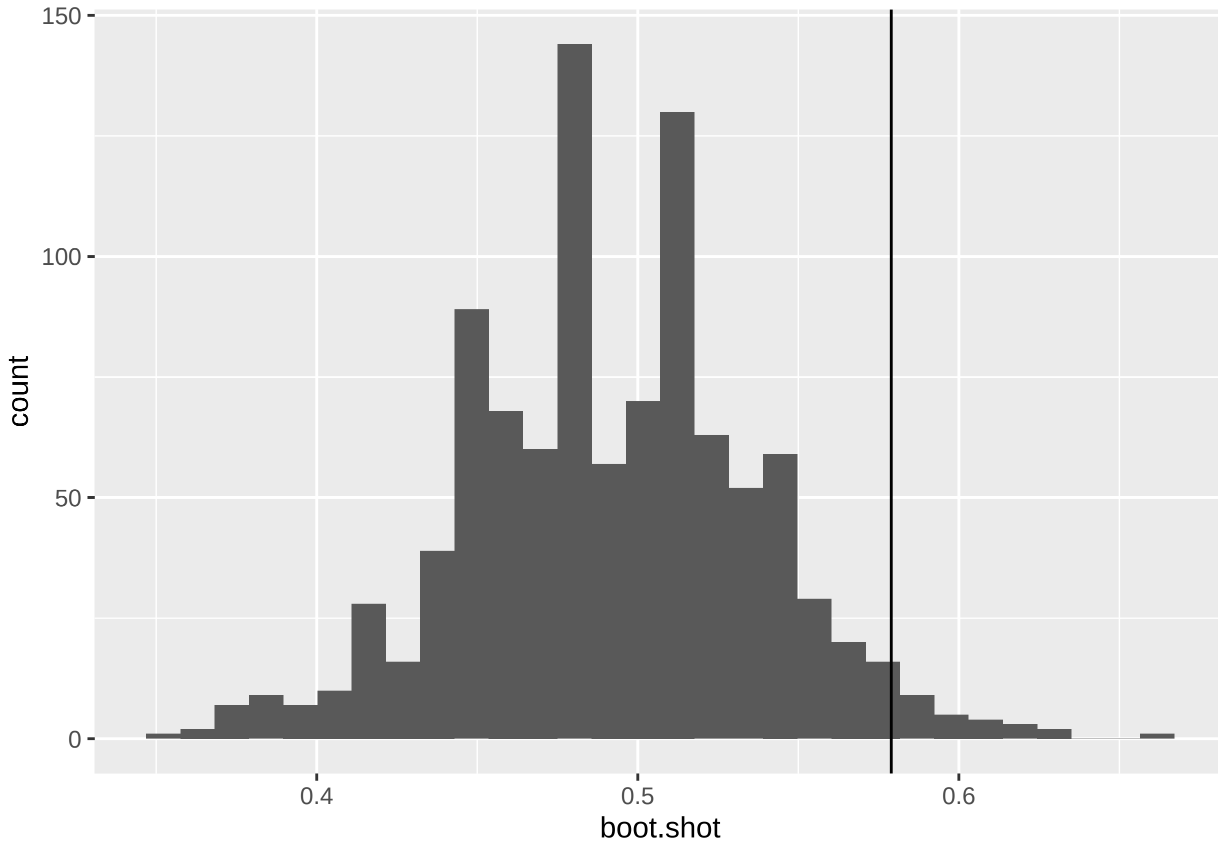
The result is 0.37, which is similar to the theory probability.

1(d)

```r
shots <- c(rep(1,62),rep(0,64))
# bootstraps
boot.shot <- NULL

for(i in 1:1000){
  boot.shot <-c(boot.shot, mean(sample(shots,126,replace = T)))
  boot.shot
}
```

```r
ggplot(as.data.frame(boot.shot), mapping = aes(x = boot.shot)) +
  geom_histogram() +
  geom_vline(xintercept = 11/19)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
# 95%CI
quantile(boot.shot, 0.025)
```

```
##      2.5%
## 0.3968254
```

```r
quantile(boot.shot, 0.975)
```

```
##     97.5%
## 0.5793651
```

so the 95% CI is (0.4047619,0.5793651).

Each shot attempt have 1/2 probability to success(1) and 1/2 probability to fail(0). The expected value E = (1/2)*1 + (1/2)*0 = 1/2. Regression to the mean is the tendency for extreme or unusual socres or events to fall back toward the average. Hence, finally, Curry's three point field goal percentage will go to 0.5 < 11/19.

Question 2

Eigenfaces

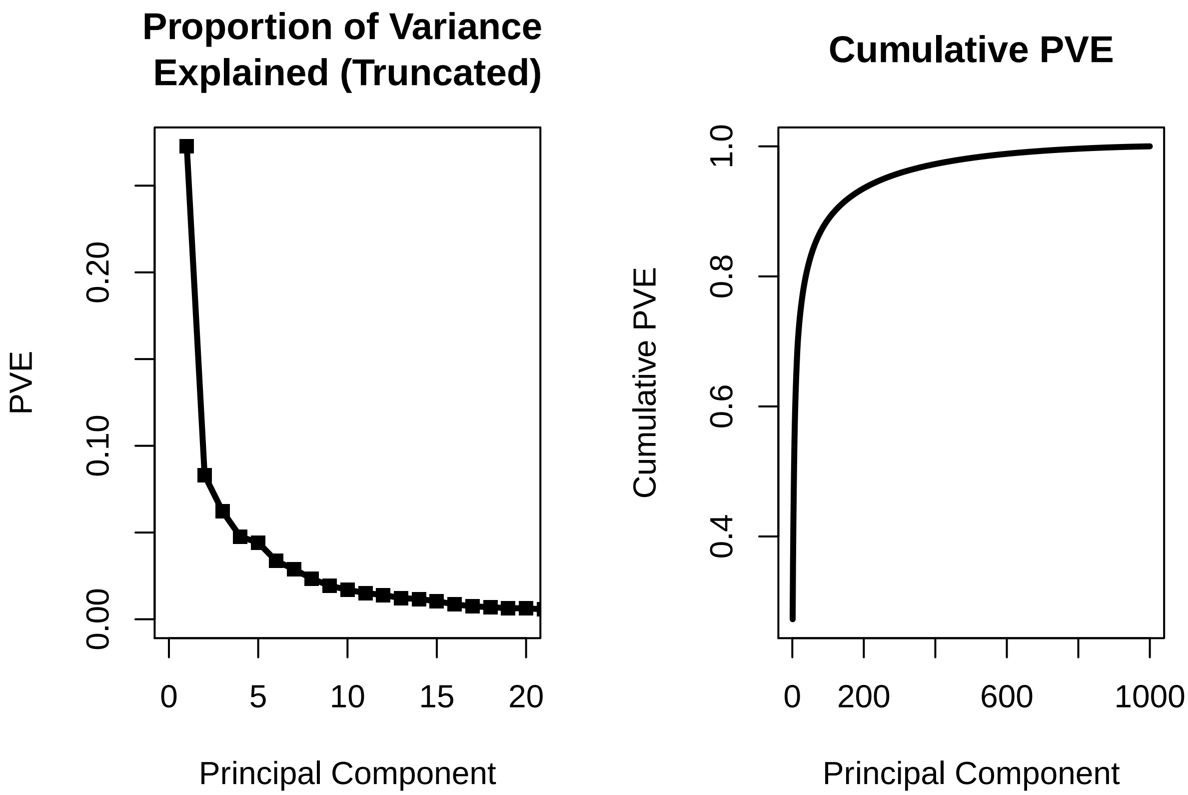
```r
load("faces_array.RData")
face_mat <- sapply(1:1000, function(i) as.numeric(faces_array[, , i])) %>% t
plot_face <- function(image_vector) {
  plot(as.cimg(t(matrix(image_vector, ncol=100))), axes=FALSE, asp=1)
  }
```

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

a) 

```r
face_pr_out <- prcomp(face_mat, center = T, scale = F)
face_pr_var <- face_pr_out$sdev^2
face_pve <- face_pr_var/sum(face_pr_var)
face_cumulative_pve <- cumsum(face_pve)
```

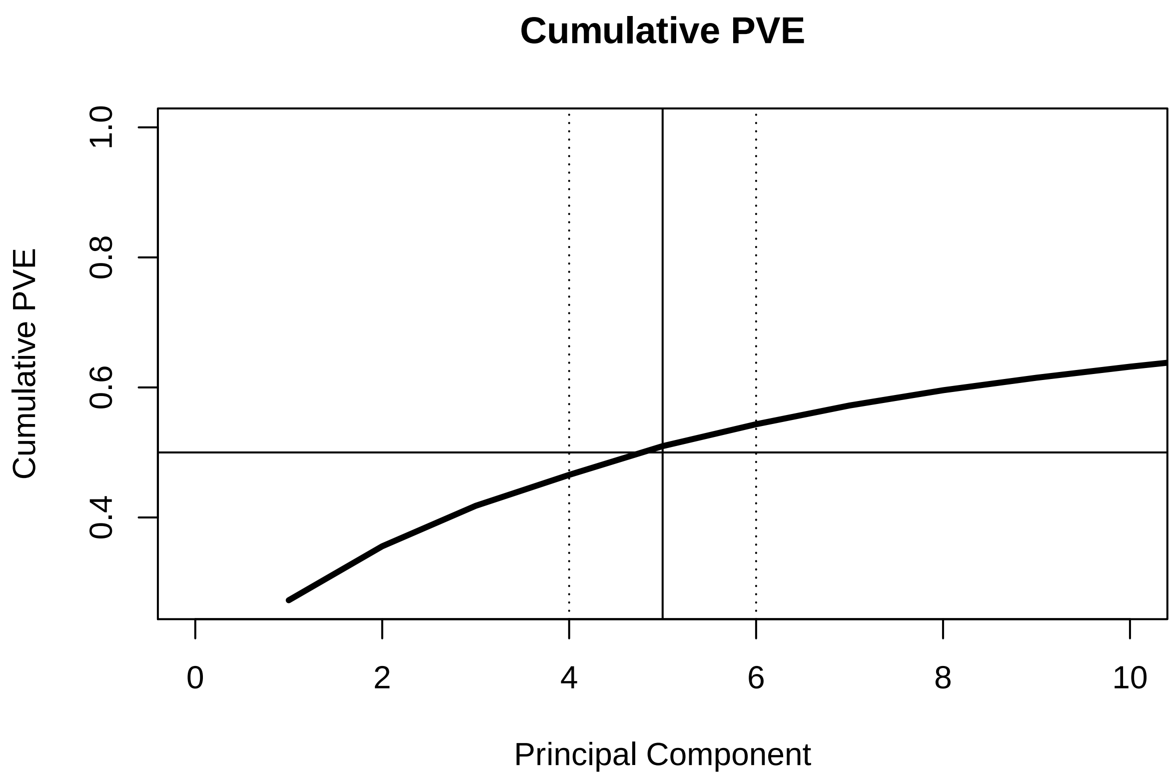
```r
par(mfrow = c(1,2))

plot(face_pve, type="l", lwd=3, xlim = c(0,20),
     xlab = 'Principal Component', ylab = 'PVE', main = 'Proportion of Variance \nExplained (Truncated)
points(face_pve, pch = 15)
plot(face_cumulative_pve, type="l", lwd=3, xlab = 'Principal Component',
     ylab = 'Cumulative PVE', main = 'Cumulative PVE')
```



b)

The PVE plot is truncated to the first 20 principal components to demonstrate where adding components begin to contribute minimally to the explained variance.

```r
pc_50 <- which(face_cumulative_pve >= .5)[1]
plot(face_cumulative_pve, type="l", xlim = c(0,pc_50+5), lwd=3, xlab = 'Principal Component',
     ylab = 'Cumulative PVE', main = 'Cumulative PVE')
abline(h=.5, v=pc_50)
abline(v = c(pc_50 - 1, pc_50 + 1), lty = 3)
```



We can see from the plot above that 5 principal components gives us just over .5 on the cumulative PVE scale, so we need 5 principal components in order to obtain at least 50% of the total variation in the face images.

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

c)

There are significantly higher amounts of lighter regions opposed to darker regions, although both light and dark regions showcase regions of high contrast. The contrast decreases through the 16 principal components and faces become more noticeable.

```r
min_pc1 <- head(order(face_pr_out$x[ ,1]), n = 5)
max_pc1 <- tail(order(face_pr_out$x[ ,1]), n = 5)

par(mfrow=c(2,5), mar=c(1,1,1,1))
for(i in c(min_pc1,max_pc1))
  plot_face(face_mat[i, ])
```

d) 

The top row goes from the lowest value to the fifth lowest value from left to right while the bottom row goes from the fifth highest value to the highest value from left to right. The most obvious variation between the top row and the bottom row of the plot above is the contrast of the background with the face. The top row has completely black backgrounds while the bottom row has completely white backgrounds which greatly contrasts with the individual faces, therefore giving the most variability in the images as a whole.

```r
min_pc5 <- head(order((face_pr_out$x[ ,5])), n = 5)
max_pc5 <- tail(order((face_pr_out$x[ ,5])), n = 5)

par(mfrow=c(2,5), mar=c(1,1,1,1))
for(i in c(min_pc5,max_pc5))
  plot_face(face_mat[i, ])
```

e)

The aspect of variability that is best captured in the 5th principal component is the length/type of the hair on the person's head. It looks like with a lower PC5 value, they person has less hair and with a higher PC5, the person has more/longer hair. I believe PC5 would be better at identifying a person's face because the hair is good indicator of a person's identity. Since PC1 only looks at the background behind the person's face, it is not as strong of an identifier as someone's hair/ hair length.

Question 3

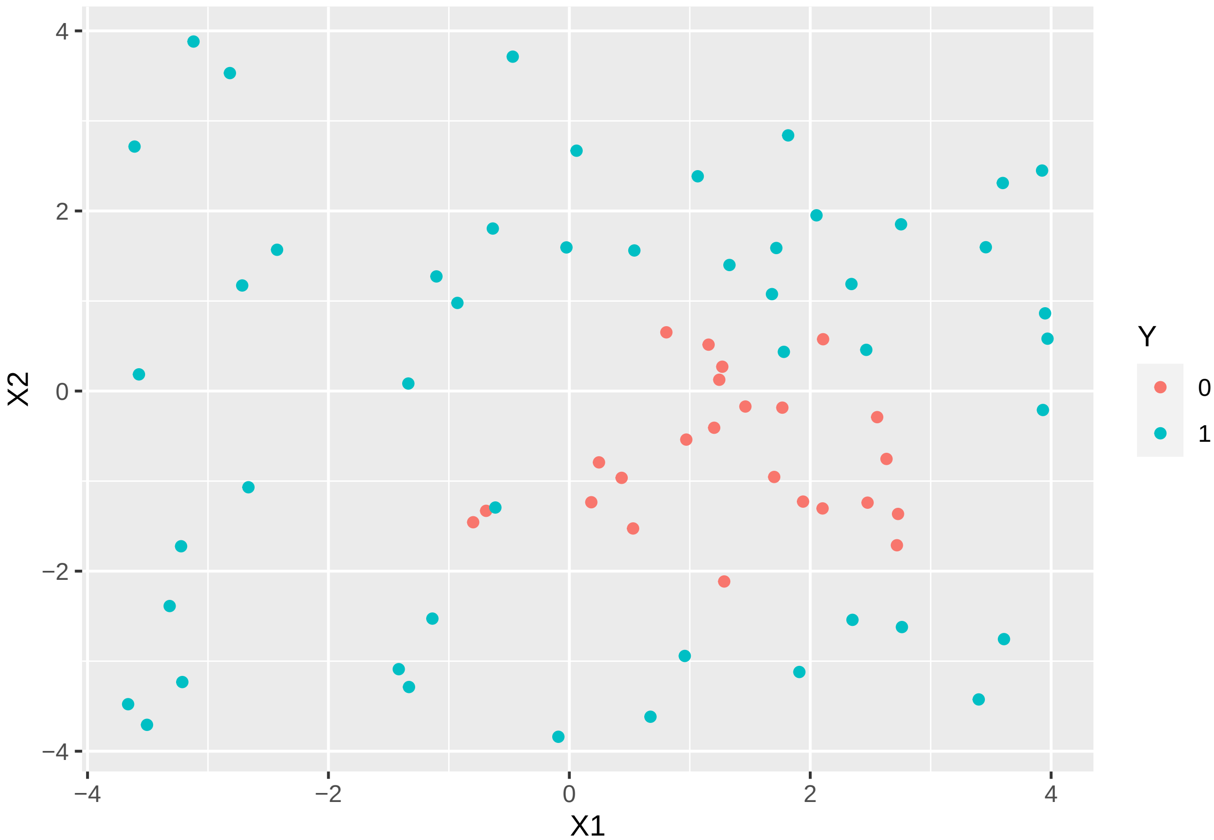
Logistic regression with polynomial features

```
nonlinear_data <- read_csv('nonlinear.csv') %>%
                    mutate(Y = factor(Y))
```

a)

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

```
ggplot(nonlinear_data, aes(x=X1, y=X2, col=Y)) +
  geom_point()
```




```
summary(nonlinear_fit <- glm(Y ~ X1 + X2, data = nonlinear_data, family="binomial"))
```

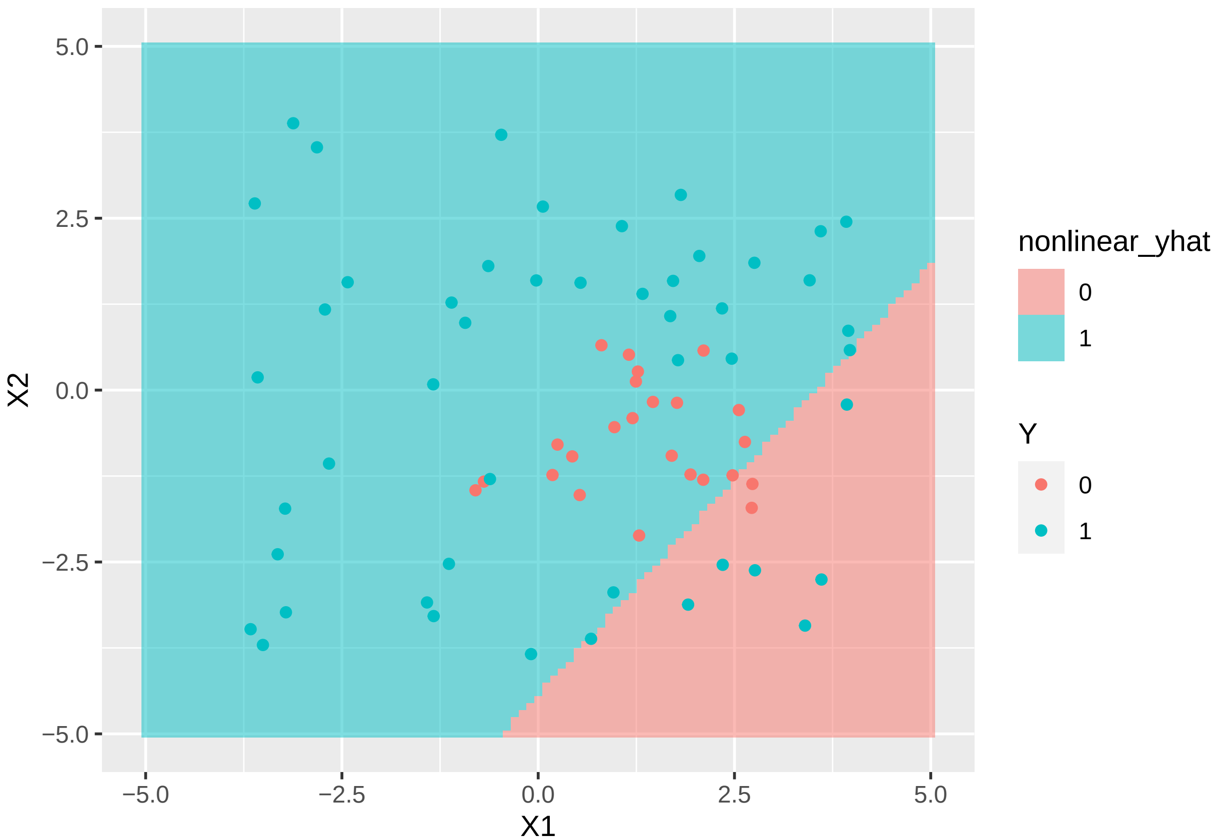
b)

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = "binomial", data = nonlinear_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5940  -1.2476   0.6256   0.9155   1.5108
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0224     0.3137   3.259  0.00112 **
## X1           -0.2893     0.1360  -2.127  0.03341 *
## X2            0.2323     0.1435   1.618  0.10560
## ---
## 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: 90.523
##
## Number of Fisher Scoring iterations: 4
```

```
# grid of points over sample space
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

nonlinear_yhat <- factor(ifelse(predict(nonlinear_fit, gr, type = "response") >= .5, 1, 0))
```

```
ggplot(gr, aes(x=X1,y=X2)) +
  geom_raster(alpha = .5, aes(fill = nonlinear_yhat)) +
  geom_point(data = nonlinear_data, aes(col=Y))
```




```
summary(nonlinear_poly_fit <- glm(Y ~ poly(X1, degree = 2, raw = F)
                                  + poly(X2, degree = 2, raw = F) + X1:X2,
                                  data = nonlinear_data, family = "binomial"))
```

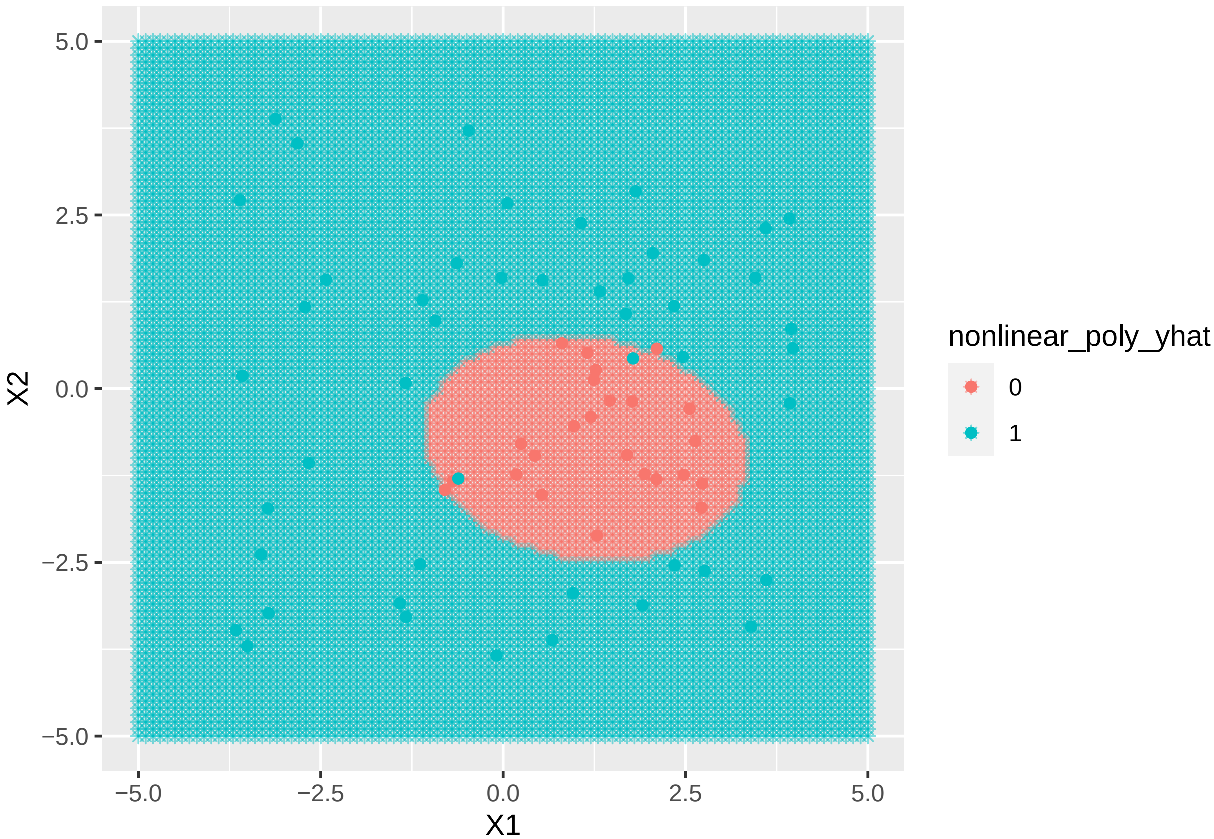
c)

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

```
##
```

```
## Call:
## glm(formula = Y ~ poly(X1, degree = 2, raw = F) + poly(X2, degree = 2,
##     raw = F) + X1:X2, family = "binomial", data = nonlinear_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.39081  -0.08271   0.00000   0.00930   1.90069
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     11.8000     4.8086   2.454   0.0141 *
## poly(X1, degree = 2, raw = F)1 -47.2697    28.2047  -1.676   0.0937 .
## poly(X1, degree = 2, raw = F)2  57.7766    29.0429   1.989   0.0467 *
## poly(X2, degree = 2, raw = F)1  45.0707    26.9112   1.675   0.0940 .
## poly(X2, degree = 2, raw = F)2  96.3106    39.7327   2.424   0.0154 *
## 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
```

```
nonlinear_poly_yhat <- factor(ifelse(predict(nonlinear_poly_fit, gr, type = "response") >= .5, 1, 0))

ggplot(mapping = aes(x=X1, y=X2)) +
  geom_point(data = gr, shape = 8 , alpha = .5, aes(col = nonlinear_poly_yhat)) +
  geom_point(data = nonlinear_data, aes(col = Y))
```




```
summary(nonlinear_5thpoly_fit <- glm(Y ~ poly(X1, degree = 5)
                                   + poly(X2, degree = 5),
                                   data = nonlinear_data, family = "binomial"))
```

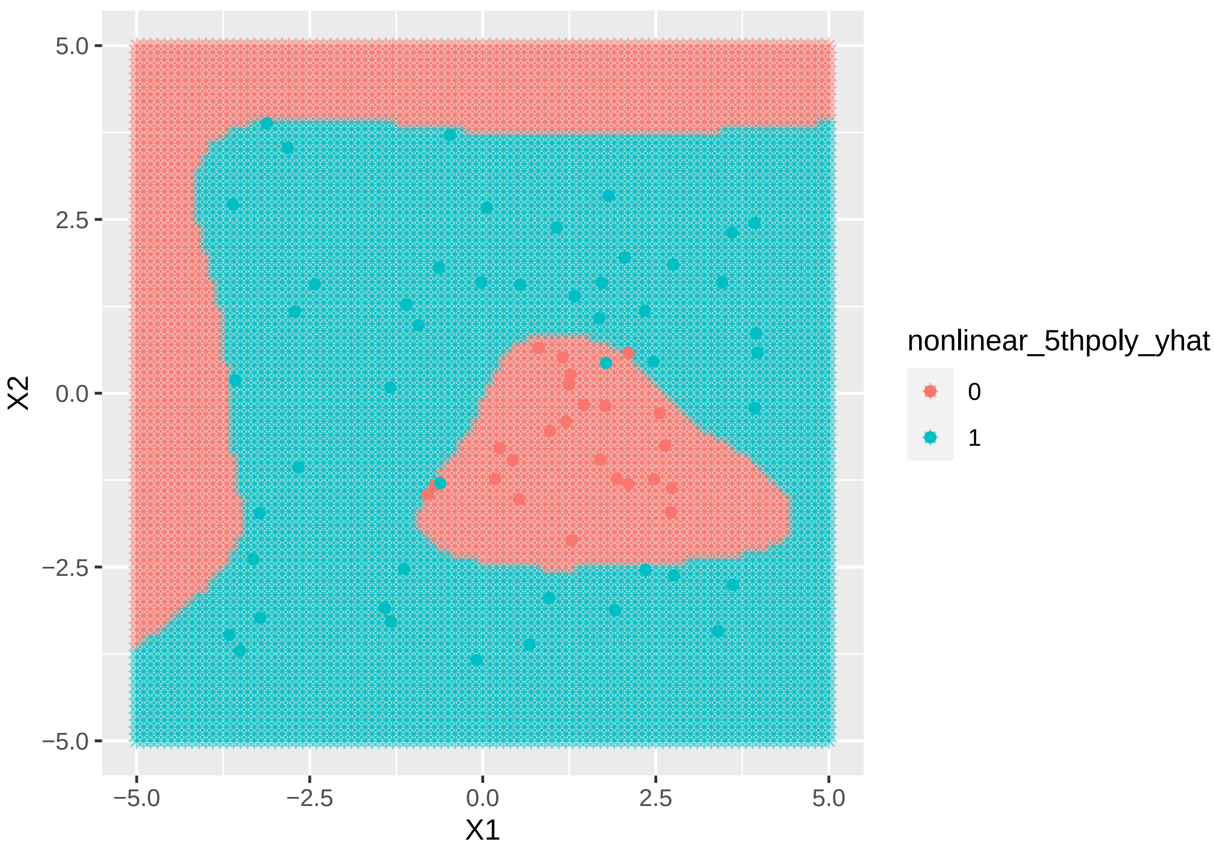
d)

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

```
##
## Call:
## glm(formula = Y ~ poly(X1, degree = 5) + poly(X2, degree = 5),
##     family = "binomial", data = nonlinear_data)
##
## 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, degree = 5)1   -49.29      88.35  -0.558    0.577
## poly(X1, degree = 5)2    25.89      36.92   0.701    0.483
## poly(X1, degree = 5)3    36.24      60.98   0.594    0.552
```

```
## poly(X1, degree = 5)4   -34.71      64.85  -0.535    0.593
## poly(X1, degree = 5)5    12.65      37.72   0.335    0.737
## poly(X2, degree = 5)1  -174.38     386.21  -0.452    0.652
## poly(X2, degree = 5)2   266.09     480.06   0.554    0.579
## poly(X2, degree = 5)3  -228.97     422.75  -0.542    0.588
## poly(X2, degree = 5)4    90.75     219.09   0.414    0.679
## poly(X2, degree = 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
```

```
nonlinear_5thpoly_yhat <- factor(ifelse(predict(nonlinear_5thpoly_fit, gr, type = "response") >= .5, 1,

ggplot(mapping = aes(x=X1, y=X2)) +
  geom_point(data = gr, shape = 8 , alpha = .5, aes(col = nonlinear_5thpoly_yhat)) +
  geom_point(data = nonlinear_data,aes(col = Y))
```



The lack of an interaction plot gives us some undesirable results. A 5th-order polynomial does a fairly reasonable job in creating decision boundaries around the true separation, but we see an added boundary in the upper left corner that is not shown in the true-labeled plot. This region does not contain any actual data points, so it is possible that the model simply did not know what to do for those points.

e) As the degree of the model increases, the model will approach a perfect fit of the data. A perfect fit of several points of data will create an extremely flexible curve that will fluctuate tremendously in magnitude, represented by these coefficients. Looking at the second-degree polynomial model, the degree is much smaller resembling lesser fluctuations, yielding smaller coefficients. Finally, with the linear model, a first-degree polynomial is simply a line, resembling no fluctuation and therefore contains smaller coefficients.

Question 4

Predicting insurance policy purchases

```r
library(ISLR)
caravan_train <- Caravan[1:1000, ]
caravan_test <- Caravan[-(1:1000), ]
```

a)

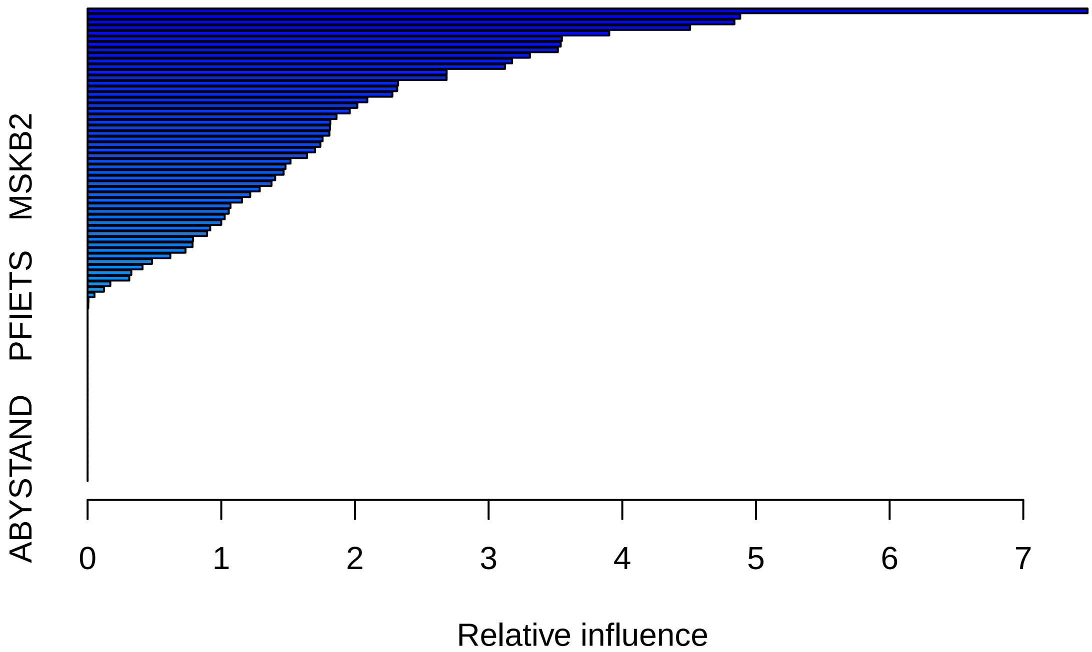
```r
set.seed(1)
caravan_boost <- gbm(ifelse(Purchase == "Yes", 1, 0)~., data = caravan_train,
                     distribution = "bernoulli", n.trees = 1000,
                     shrinkage = .01, interaction.depth = 4)
```

b)

```
## 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.
```

```r
summary(caravan_boost)
```




```
##                   var     rel.inf
## PPERSAUT PPERSAUT 7.480819014
## MOPLHOOG MOPLHOOG 4.882054338
## MGODGE     MGODGE 4.838869962
## MKOOPKLA MKOOPKLA 4.507280400
## MOSTYPE   MOSTYPE 3.902338079
## MGODPR     MGODPR 3.547892360
## PBRAND     PBRAND 3.539487907
## MBERMIDD MBERMIDD 3.518082698
## MBERARBG MBERARBG 3.309004843
## MINK3045 MINK3045 3.175313873
## MSKC         MSKC 3.123008472
## MSKA         MSKA 2.685844523
## MAUT2       MAUT2 2.685548007
## MAUT1       MAUT1 2.322786246
## PWAPART   PWAPART 2.316252267
## MSKB1       MSKB1 2.279820190
## MRELOV     MRELOV 2.092410309
## MFWEKIND MFWEKIND 2.017651081
## MBERHOOG MBERHOOG 1.961378700
## MBERARBO MBERARBO 1.862074416
## MRELGE     MRELGE 1.815276446
## MINK7512 MINK7512 1.812894054
## MINKM30   MINKM30 1.808781053
## MOPLMIDD MOPLMIDD 1.757784665
## MFGEKIND MFGEKIND 1.741172971
## MGODOV     MGODOV 1.701539077
## MZFONDS   MZFONDS 1.641658796
## MFALLEEN MFALLEEN 1.517763739
## MSKB2       MSKB2 1.480397941
## MINK4575 MINK4575 1.466410983
## MAUT0       MAUT0 1.403097259
```

```
## ABRAND     ABRAND 1.375696683
## MHHUUR     MHHUUR 1.287672857
## MINKGEM   MINKGEM 1.216351643
## MHKOOP     MHKOOP 1.154970948
## MGEMLEEF MGEMLEEF 1.068800262
## MGODRK     MGODRK 1.056066524
## MRELSA     MRELSA 1.025383382
## MZPART     MZPART 0.999705745
## MSKD         MSKD 0.917077921
## MGEMOMV   MGEMOMV 0.893757812
## MBERZELF MBERZELF 0.788935429
## APERSAUT APERSAUT 0.784652995
## MOPLLAAG MOPLLAAG 0.732210597
## MOSHOOFD MOSHOOFD 0.618703929
## PMOTSCO   PMOTSCO 0.481824116
## PLEVEN     PLEVEN 0.410808274
## PBYSTAND PBYSTAND 0.326851643
## MBERBOER MBERBOER 0.311571820
## MINK123M MINK123M 0.169710044
## MAANTHUI MAANTHUI 0.122660387
## ALEVEN     ALEVEN 0.051158218
## PAANHANG PAANHANG 0.006040057
## PFIETS     PFIETS 0.004694048
## PWABEDR   PWABEDR 0.000000000
## PWALAND   PWALAND 0.000000000
## PBESAUT   PBESAUT 0.000000000
## PVRAAUT   PVRAAUT 0.000000000
## PTRACTOR PTRACTOR 0.000000000
## PWERKT     PWERKT 0.000000000
## PBROM       PBROM 0.000000000
## PPERSONG PPERSONG 0.000000000
## PGEZONG   PGEZONG 0.000000000
## PWAOREG   PWAOREG 0.000000000
## PZEILPL   PZEILPL 0.000000000
## PPLEZIER PPLEZIER 0.000000000
## PINBOED   PINBOED 0.000000000
## AWAPART   AWAPART 0.000000000
## AWABEDR   AWABEDR 0.000000000
## AWALAND   AWALAND 0.000000000
## ABESAUT   ABESAUT 0.000000000
## AMOTSCO   AMOTSCO 0.000000000
## AVRAAUT   AVRAAUT 0.000000000
## AAANHANG AAANHANG 0.000000000
## ATRACTOR ATRACTOR 0.000000000
## AWERKT     AWERKT 0.000000000
## ABROM       ABROM 0.000000000
## APERSONG APERSONG 0.000000000
## AGEZONG   AGEZONG 0.000000000
## AWAOREG   AWAOREG 0.000000000
## AZEILPL   AZEILPL 0.000000000
## APLEZIER APLEZIER 0.000000000
## AFIETS     AFIETS 0.000000000
## AINBOED   AINBOED 0.000000000
## ABYSTAND ABYSTAND 0.000000000
```

The PPERSAUT, MKOOPKLA, and MOPLHOOG appear to be the most important preditors in this data set.

```
set.seed(1)
caravan_forest <- randomForest(Purchase ~ ., data=caravan_train, importance=TRUE)
caravan_forest
```

c)

```
##
## 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
```

The OOB estimate of error rate is 6.1% with 9 variables subsampled at each split. The default number of trees selected was 500.

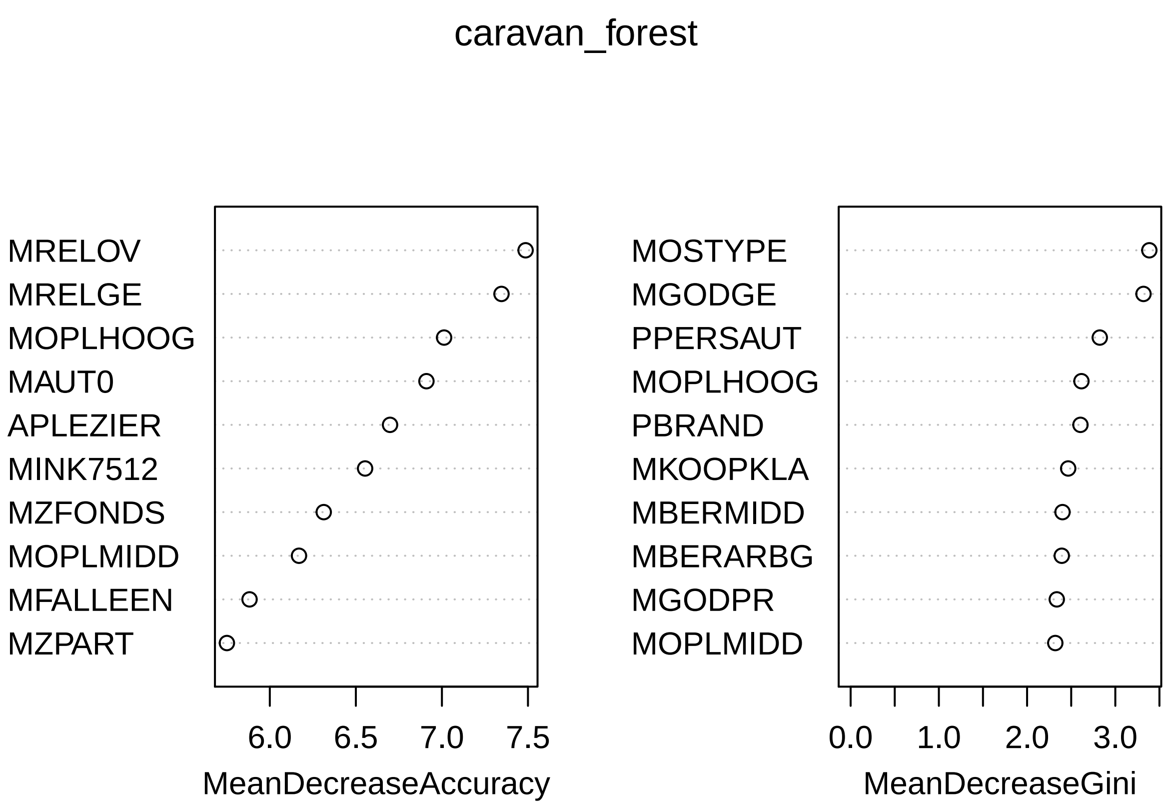
```
importance(caravan_forest)
```

```
##                  No          Yes MeanDecreaseAccuracy MeanDecreaseGini
## MOSTYPE   2.45154591  2.244153819            3.0134507     3.3855613917
## MAANTHUI  1.60618826 -0.290842854            1.4538716     0.6143667918
## MGEMOMV   3.38493902 -2.110795715            2.9068779     1.0071021582
## MGEMLEEF  2.55973289  1.473727894            2.9719722     1.0794485718
## MOSHOOFD  3.36995556  4.426788277            4.4229915     1.9738057922
## MGODRK    4.65845598  0.737034753            4.7409588     1.2387385868
## MGODPR    4.47209929  0.908551685            4.4602577     2.3366146232
## MGODOV    2.50722507 -0.487383289            2.3893495     1.5670299537
## MGODGE    1.28224362  5.639817197            3.4991594     3.3194250040
## MRELGE    7.36265693  0.269752923            7.3461530     1.9588421148
## MRELSA    4.70194669  0.452166675            4.6543739     1.3126338786
## MRELOV    7.46288700  0.496195580            7.4861695     1.7935856470
## MFALLEEN  6.11338942 -0.244730246            5.8823286     1.5595153222
## MFGEKIND  3.55145579 -0.416353528            3.3704460     2.0285631796
## MFWEKIND  1.77176524 -1.029635115            1.5662212     2.2381811379
## MOPLHOOG  5.11644375  6.411930730            7.0125794     2.6160759960
## MOPLMIDD  6.38345887 -0.367223968            6.1696129     2.3197585934
## MOPLLAAG  5.03880399 -0.204079688            4.9848255     1.7436716439
## MBERHOOG  2.84911348  0.640220247            3.1271066     1.8173572113
## MBERZELF  1.81181008  1.029947948            2.0175598     0.7595009487
## MBERBOER  0.24276831  0.500299080            0.3745657     0.4528499609
## MBERMIDD  4.58228798  4.226428926            5.6409776     2.4014556209
```

```
## MBERARBG  4.07720716 -1.126025271            3.7824374     2.3930576749
## MBERARBO  4.70824017  0.507995051            4.8136532     2.0635646903
## MSKA      2.85153495  2.869038666            3.7018567     2.0505017986
## MSKB1     2.02495419  2.811995244            2.7990426     1.9649351466
## MSKB2     3.63933294 -1.140039549            3.2819700     1.9153830076
## MSKC      1.62816010  2.248348320            2.2101073     2.2403166260
## MSKD      1.18991796  0.204816577            1.2780817     1.0344871416
## MHHUUR    2.17495496  4.339104961            3.4333327     2.0183314106
## MHKOOP    2.14404191  4.794262902            3.4505946     2.1058102134
## MAUT1     0.85874384 -0.745278178            0.7330822     1.9019071060
## MAUT2     2.49271799  1.897193762            2.9121809     1.7206310469
## MAUT0     7.06300777 -0.860204512            6.9101043     1.7461401474
## MZFONDS   5.99837468  0.482100327            6.3134388     2.0298872877
## MZPART    5.87997698  0.265375941            5.7507661     1.8976048036
## MINKM30   3.30483243  1.053640398            3.4325492     1.7861887910
## MINK3045  1.42380610  0.694473622            1.5478581     2.1300576007
## MINK4575  2.22746442  1.000307996            2.4605940     1.6754786425
## MINK7512  6.20976748  1.628500118            6.5538078     1.8321676293
## MINK123M -0.98857123  0.530151554           -0.7682253     0.3776091219
## MINKGEM   2.42617883  1.077510437            2.7107917     1.4937252558
## MKOOPKLA  4.16470811  3.618032553            5.1828355     2.4658506352
## PWAPART  -3.02481682  4.903157985           -1.2806744     2.0032603259
## PWABEDR   0.44448411 -1.001001503            0.2224323     0.1697765794
## PWALAND   1.50266675 -1.001001503            1.2555966     0.0893057652
## PPERSAUT  2.09263491  5.367475427            3.4587027     2.8240358520
## PBESAUT   0.00000000  0.000000000            0.0000000     0.0110000000
## PMOTSCO  -1.58856509 -0.914425433           -1.8397010     0.8164202523
## PVRAAUT   0.00000000  0.000000000            0.0000000     0.0000000000
## PAANHANG -0.05105565 -1.001001503           -0.2110083     0.2010860321
## PTRACTOR  1.34110446  0.000000000            1.3315425     0.2069953044
## PWERKT    0.00000000  0.000000000            0.0000000     0.0001025641
## PBROM     5.14931162 -1.728419270            4.5362542     0.4886392196
## PLEVEN   -0.09157408  0.005157702           -0.1001994     0.6850565571
## PPERSONG  0.00000000  0.000000000            0.0000000     0.0050000000
## PGEZONG  -0.99997680 -0.660121070           -1.1215069     0.7265914692
## PWAOREG   3.50037985  2.555469067            3.7534286     0.8830211907
## PBRAND   -3.46618600  3.065705615           -2.3895592     2.6050140703
## PZEILPL   0.00000000  0.000000000            0.0000000     0.3101046631
## PPLEZIER  3.41283872  5.519866934            5.3475104     1.9481523655
## PFIETS   -1.60373164 -1.001001503           -1.6410071     0.1434584923
## PINBOED   0.33600315 -1.415913679           -0.2448524     0.0622960673
## PBYSTAND  1.62312319  1.116455430            1.7928666     0.7890283949
## AWAPART   0.07474160  5.113356495            2.1664576     1.2282489984
## AWABEDR   1.64885562  0.000000000            1.6533327     0.0886151003
## AWALAND   1.30485240  1.001001503            1.5086644     0.0799057637
## APERSAUT  0.90083143  0.166343909            0.8674609     2.0385345213
## ABESAUT   0.00000000  0.000000000            0.0000000     0.0094285714
## AMOTSCO   0.75583812 -1.979465167            0.3016325     0.9055590600
## AVRAAUT   0.00000000  0.000000000            0.0000000     0.0000000000
## AAANHANG -2.22662144 -1.984140947           -2.5742611     0.1799960163
## ATRACTOR  1.90298837  0.000000000            1.9031806     0.0827505956
## AWERKT    0.00000000  0.000000000            0.0000000     0.0000000000
## ABROM     4.58525585 -2.944440951            3.5474461     0.3956193901
## ALEVEN   -0.35600815  0.096028467           -0.3229689     0.2689196899
```

```
## APERSONG  0.00000000  0.000000000           0.0000000     0.0022666667
## AGEZONG   0.01725175 -2.121294784          -0.5451891     0.4079733602
## AWAOREG   3.42266626  1.976805350           3.9407140     0.8752068226
## ABRAND   -0.86663442  0.401244715          -0.7153231     1.9137179129
## AZEILPL   0.00000000  0.000000000           0.0000000     0.3081311861
## APLEZIER  3.18370968  7.770210840           6.6986818     1.5981874502
## AFIETS   -1.68152736 -0.227241349          -1.6378835     0.2249215377
## AINBOED   0.29965226 -1.327312597          -0.4106905     0.0644346995
## ABYSTAND  1.03929978  1.529758423           1.4094141     0.5114897627
```

```
varImpPlot(caravan_forest, n = 10)
```



The order of variable importance differed between the boosting and random forest models. Actually, even the random forest model had different order of variable importance based on the impurity value chosen. For the mean decrease in accuracy, MRELOV, MBERMIDD, and MINK7512 were the most important, whereas for the mean decrease in MOSTYPE, MGODGE, and PPERSAUT were determined to be the most important.

```
caravan_boost_yhat <- ifelse(predict(caravan_boost, newdata = caravan_test,
                              n.trees = 1000, type = "response") > .2,
                      "Yes", "No")

(caravan_boost_err <- table(Boost_Predict = caravan_boost_yhat, Truth = caravan_test$Purchase))
```

d)

```
##              Truth
## Boost_Predict   No  Yes
##           No  4336  258
##           Yes  197   31
```

```
caravan_forest_yhat <- ifelse(predict(caravan_forest, newdata = caravan_test, type = "prob")[ ,2] > .2,
                              "Yes", "No")
(caravan_forest_err <- table(Forest_Predict = caravan_forest_yhat, Truth = caravan_test$Purchase))
```

```
##               Truth
## Forest_Predict   No  Yes
##            No  4276  242
##            Yes  257   47
```

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

```
## [1] 0.1546053
```

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'))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   ID = col_double(),
##   Age = col_double(),
##   Gender = col_double(),
##   Education = col_double(),
##   Country = col_double(),
##   Ethnicity = col_double(),
##   Nscore = col_double(),
##   Escore = col_double(),
##   Oscore = col_double(),
##   Ascore = col_double(),
##   Cscore = col_double(),
##   Impulsive = col_double(),
##   SS = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
drug_use <- drug_use %>%
  mutate(recent_cannabis_use=factor(ifelse(Cannabis >= "CL3", "Yes", "No"), levels=c("No","Yes")))
drug_use_subset <- drug_use %>% select(Age:SS, recent_cannabis_use)
```

```r
set.seed(1)
#sample 1500 obs as training data
train_index = sample(1:nrow(drug_use_subset),1500)
drug_use_train = drug_use_subset[train_index,]
# the rest as test data
drug_use_test = drug_use_subset[-train_index,]
```

5(a)

```r
svm.drug=svm(recent_cannabis_use~., data = drug_use_train, kernel = "radial", cost = 1)

table(true = drug_use_test$recent_cannabis_use,
      pred = predict(svm.drug, newdata = drug_use_test))
```

```
##      pred
## true   No Yes
##   No  134  31
##   Yes  44 176
```

5(b)

```r
set.seed(1)
tune.svm = tune(svm, recent_cannabis_use ~., data=drug_use_train, kernel="radial",
                ranges=list(cost=c(0.1,1,10,100,1000)))
```

```r
summary(tune.svm)$"best.parameters"
```

```
##   cost
## 2    1
```

```r
summary(tune.svm)$"best.performance"
```

```
## [1] 0.1846667
```

The optimal cost is 1. CV training error is 0.1846667.

```r
best.mod = tune.svm$best.model
table(true = drug_use_test$recent_cannabis_use,
      pred = predict(best.mod, newdata = drug_use_test))
```

```
##      pred
## true   No Yes
##   No  134  31
##   Yes  44 176
```