Homework 4

Julissa Duenas

3/12/2021

Question 1: a) $(1 - \frac{1}{n})^n$

b) $(1 - \frac{1}{1000})^{1000} \approx 0.3677$

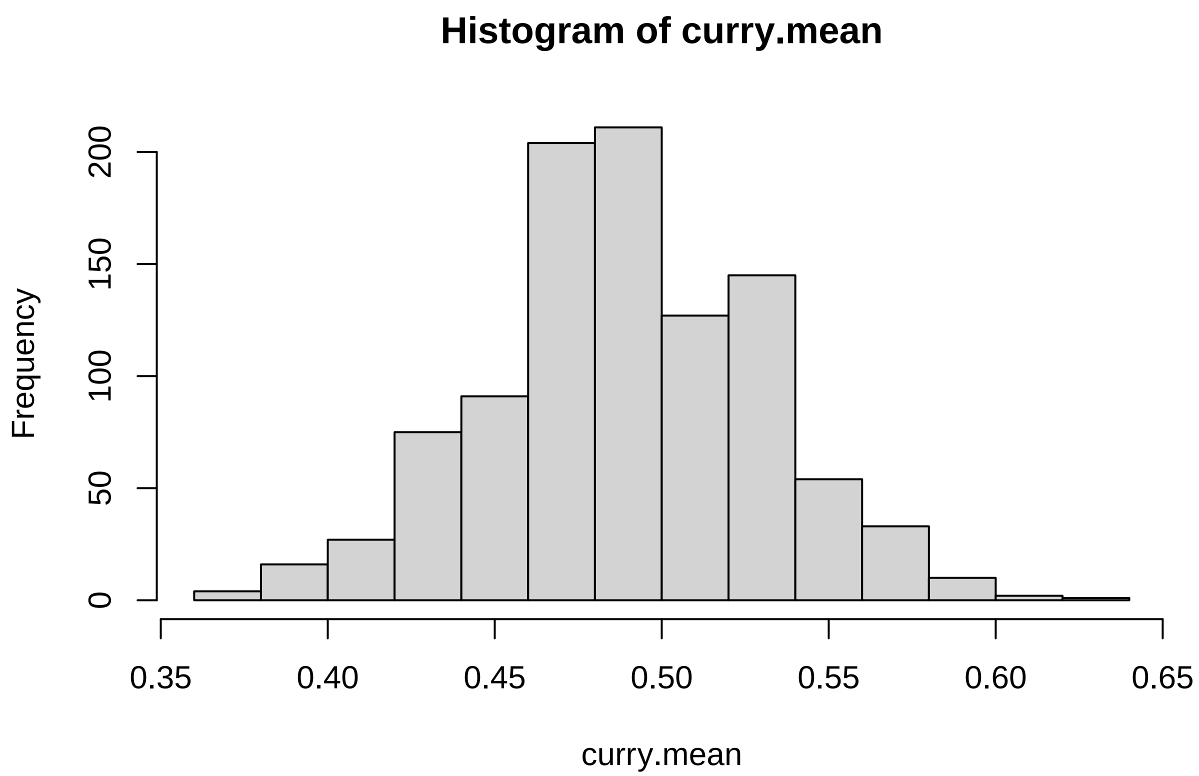
c)

```
use_samp <- sample(seq(1,1000),replace=TRUE)
num.unique <- length(unique(use_samp))
num.missing <- 1000-num.unique
num.missing/1000
```

```
## [1] 0.38
```

d)

```
curry <- c(rep(1,62),rep(0,64))
curry.mean <- c()
for (i in 1:1000) {
  curry.sample <- sample(curry,replace = TRUE)
  curry.mean[i] <- mean(curry.sample)
}
hist(curry.mean)
```




```r
low <- quantile(curry.mean,0.025)
upper <- quantile(curry.mean,0.975)
c(low,upper)
```

```
##      2.5%     97.5%
## 0.4047619 0.5714286
```

11/19 is near the beginning or the season. According to the phenomenon, it is possible that as the season continues, Curry's percentage will drop. There is little probability that it will stay at such a high rate.

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)
}
```

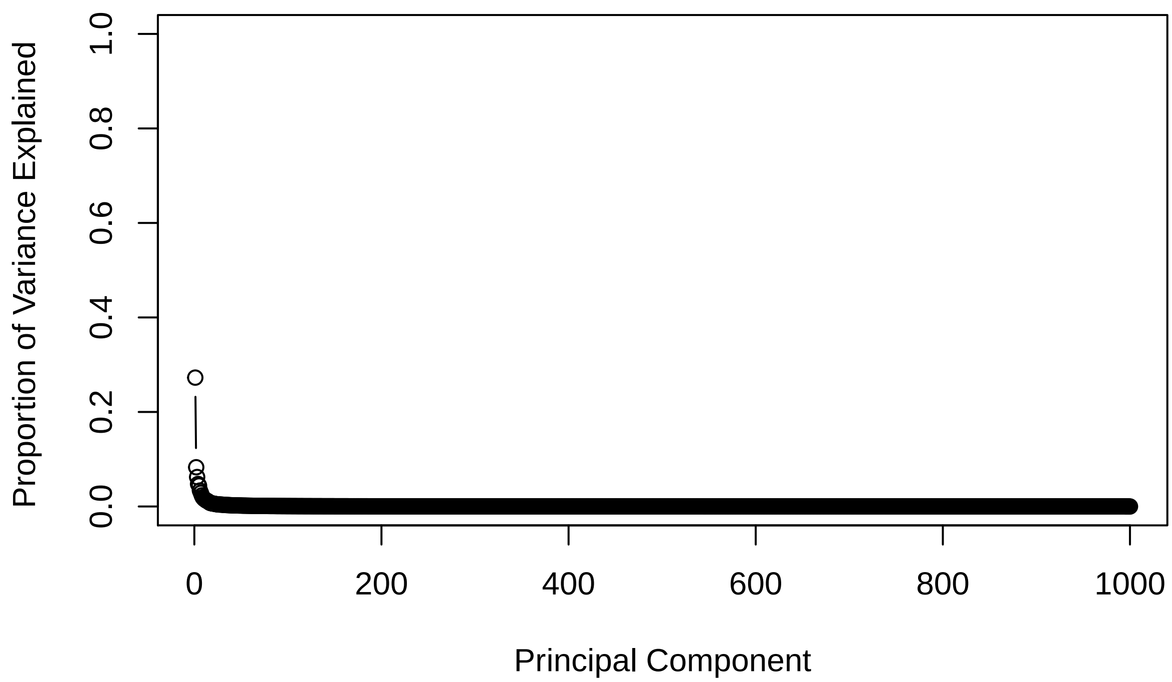
a)

```r
avg.face <- colMeans(face_mat)
plot_face(avg.face)
```

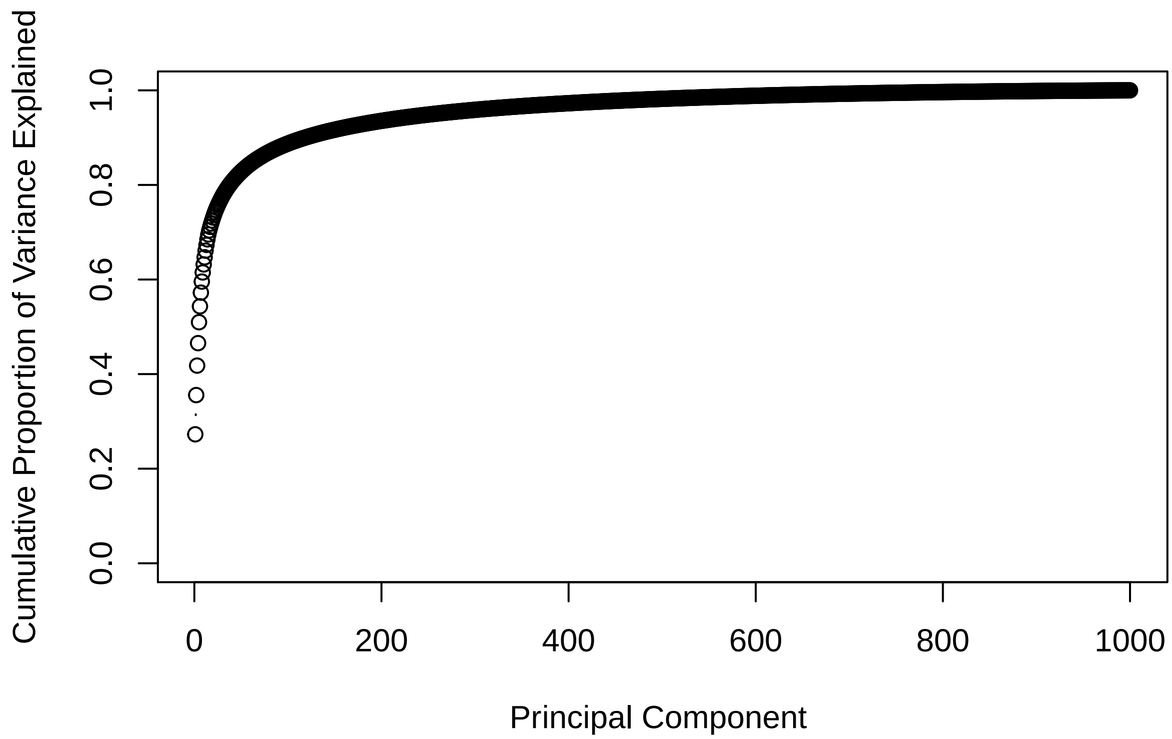
b)

```r
pr.face <- prcomp(face_mat,center=TRUE, scale=FALSE)
```

```r
pr.face.var <- pr.face$sdev^2
pve.face <- pr.face.var/sum(pr.face.var)
plot(pve.face,xlab="Principal Component",
ylab="Proportion of Variance Explained ", ylim=c(0,1),type='b')
```




```r
plot(cumsum(pve.face), xlab="Principal Component ",
ylab=" Cumulative Proportion of Variance Explained ", ylim=c(0,1), type='b')
```




```r
pve.face[1]+pve.face[2]+pve.face[3]+pve.face[4]+pve.face[5]
```

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

at least 5 PCs

c)

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

d)

```r
low.pc1 <- order(pr.face$x[,1])[1:5]
high.pc1 <- order(pr.face$x[,1],decreasing = TRUE)[1:5]
par(mfrow=c(1,5))
for (i in high.pc1) {
  plot_face(face_mat[i,])
}
```

```r
for (i in low.pc1) {
  plot_face(face_mat[i,])
}
```

It seems as if the aspect is lighting. The higher PC1 values have a light or white background while the lower values have a dark or blackened background

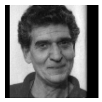
e)

```r
low.pc5 <- order(pr.face$x[,5])[1:5]
high.pc5 <- order(pr.face$x[,5],decreasing = TRUE)[1:5]
par(mfrow=c(1,5))
for (i in high.pc5) {
  plot_face(face_mat[i,])
```

```
}
```

```
for (i in low.pc5) {
  plot_face(face_mat[i,])
}
```

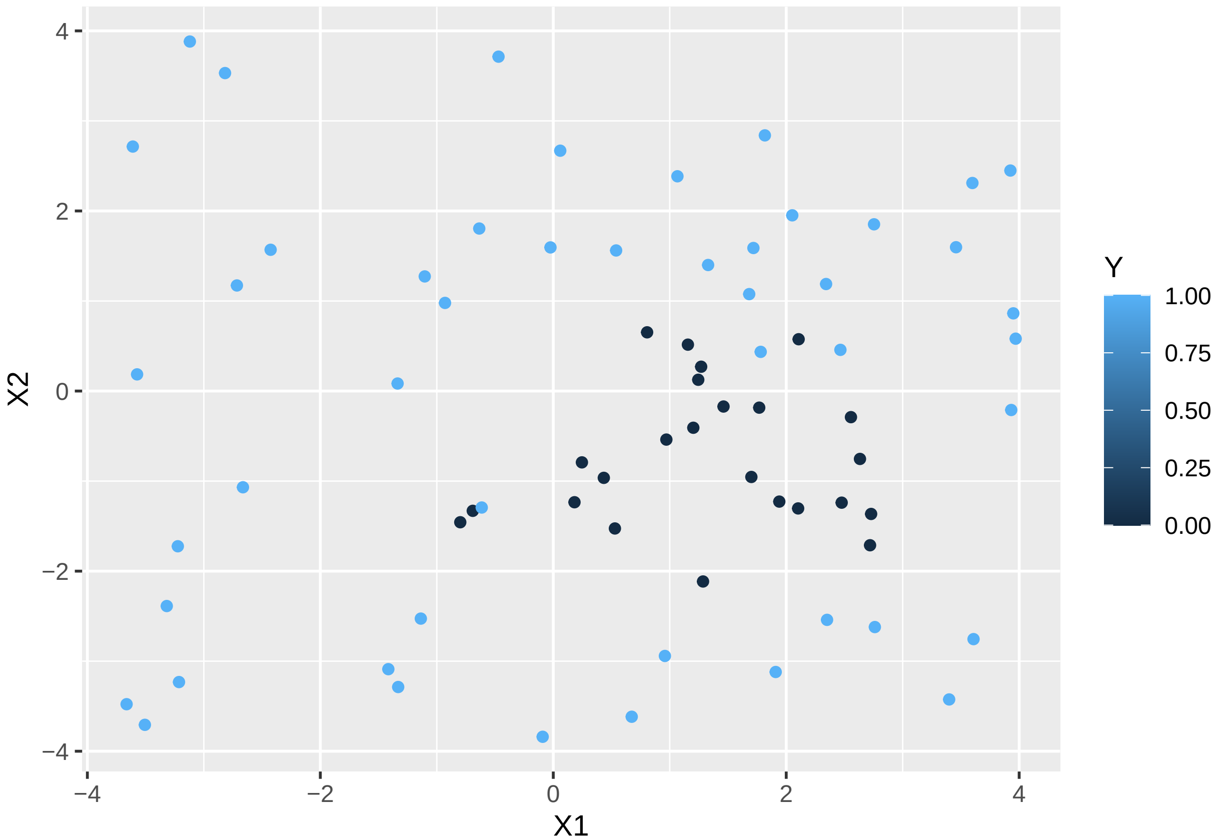
     It seems that the component here is hair length. Those with the higher pc5 value have longer hair surrounding the face and those with the lower values have less/short hair. PC5 would be better because hair is a good indentifier of a person and background darkness is not.

Question 3 a)

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

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

```
ggplot(nonlinear,aes(x=X1,y=X2,color=Y))+geom_point()
```

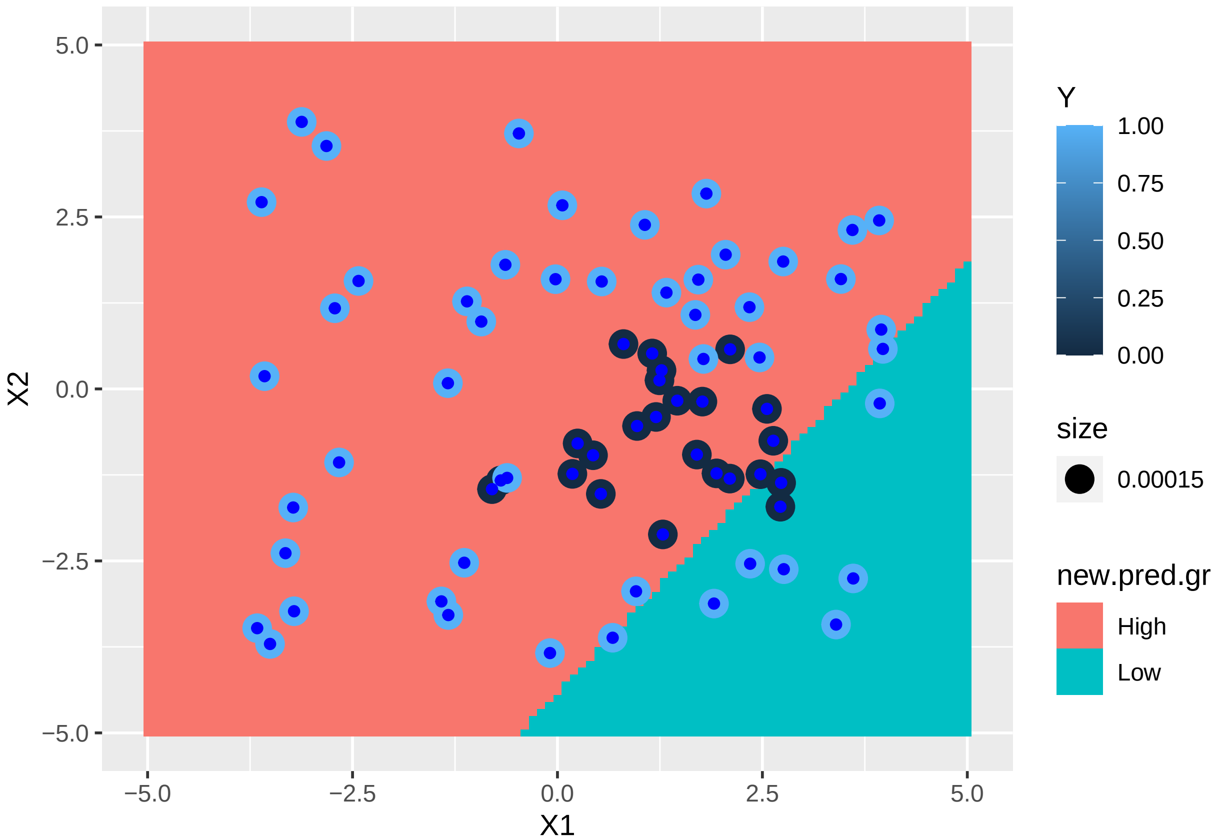


b)

```
# 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.glm <- glm(Y~X1+X2,data = nonlinear,family = binomial)
pred.gr <- predict(nonlinear.glm,gr,type='response')
new.pred.gr <- c()
for (i in c(1:range(length(pred.gr)))) {
  if (pred.gr[i]>0.5){
    new.pred.gr[i] <- 'High'
  }
  else{
    new.pred.gr[i] <- 'Low'
  }
}
```

```
## Warning in 1:range(length(pred.gr)): numerical expression has 2 elements: only
## the first used
```

```
nonlinear.raster <- ggplot(gr,aes(X1,X2),alpha=0.5)+geom_raster(aes(fill=new.pred.gr))+geom_point(data=
nonlinear.raster
```



c)

```r
nonlinear.poly <- glm(Y~poly(X1,2)*poly(X2,2),data=nonlinear,family=binomial)
```

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

```r
summary(nonlinear.poly)
```

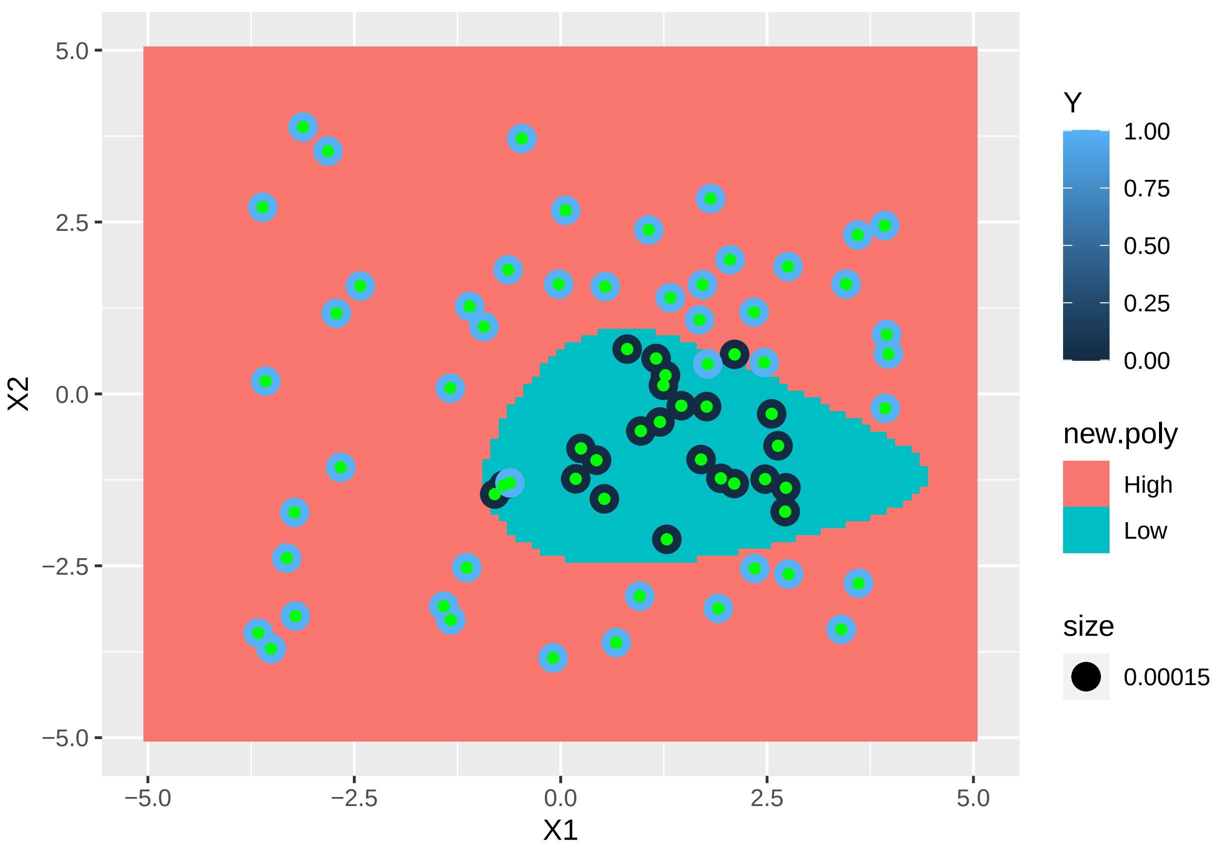
```
##
## Call:
## glm(formula = Y ~ poly(X1, 2) * poly(X2, 2), family = binomial,
##     data = nonlinear)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.57091  -0.09697   0.00000   0.01295   1.89656
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    14.37      15.55   0.924    0.355
## poly(X1, 2)1                  -51.11     147.45  -0.347    0.729
## poly(X1, 2)2                  103.41     134.54   0.769    0.442
## poly(X2, 2)1                   99.60     134.17   0.742    0.458
## poly(X2, 2)2                  113.85     117.09   0.972    0.331
## poly(X1, 2)1:poly(X2, 2)1   -181.63    1294.32  -0.140    0.888
## poly(X1, 2)2:poly(X2, 2)1    583.71    1165.55   0.501    0.617
## poly(X1, 2)1:poly(X2, 2)2    108.28    1072.06   0.101    0.920
## poly(X1, 2)2:poly(X2, 2)2    445.15    1127.08   0.395    0.693
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 12.561  on 63  degrees of freedom
## AIC: 30.561
##
## Number of Fisher Scoring iterations: 14
```

```
pred.poly <- predict(nonlinear.poly,gr,type='response')
new.poly <- c()
for (i in c(1:range(length(pred.poly)))) {
  if(pred.poly[i]>0.5){
    new.poly[i] <- 'High'
  }
  else{
    new.poly[i] <- 'Low'
  }
}
```

```
## Warning in 1:range(length(pred.poly)): numerical expression has 2 elements: only
## the first used
```

```
poly.raster <- ggplot(gr,aes(X1,X2),alpha=0.5)+geom_raster(aes(fill=new.poly))+geom_point(data=nonlinea
poly.raster
```



d)

```
poly5 <- glm(Y~poly(X1,5)+poly(X2,5),data=nonlinear,family=binomial)
```

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

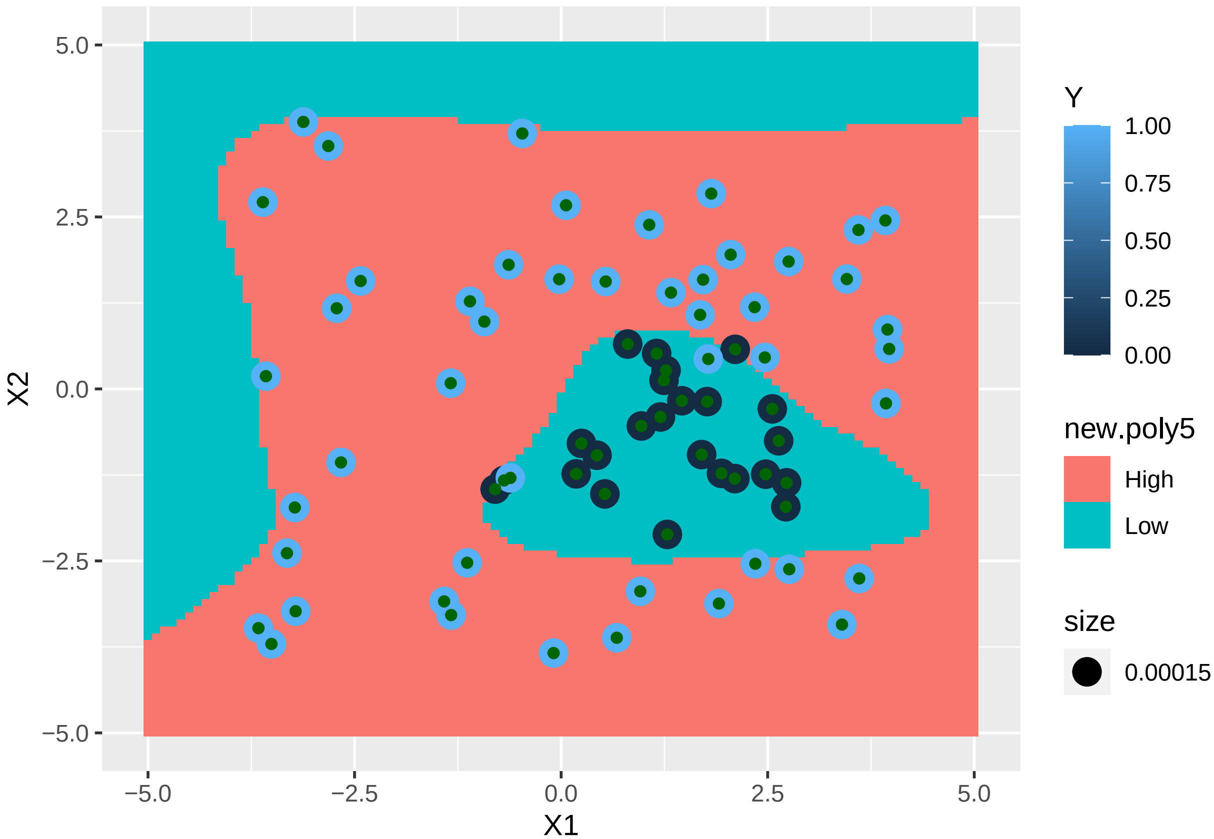
```r
summary(poly5)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 5) + poly(X2, 5), family = binomial,
##     data = nonlinear)
##
## 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
```

```r
pred.poly5 <- predict(poly5,gr,type='response')
new.poly5 <- c()
for (i in c(1:range(length(pred.poly5)))) {
  if(pred.poly5[i]>0.5){
    new.poly5[i] <- 'High'
  }
  else{
    new.poly5[i] <- 'Low'
  }
}
```

```
## Warning in 1:range(length(pred.poly5)): numerical expression has 2 elements:
## only the first used
```

```r
poly5.raster <- ggplot(gr,aes(X1,X2),alpha=0.5)+geom_raster(aes(fill=new.poly5))+geom_point(data=nonlin
poly5.raster
```



The region depicting a low classification is overfitting. A large p results in high variance and low bias

e) The magnitudes in the polynomial models are higher than the ones in the linear model. A larger p constitutes a higher variance and lower bias, we can see this in the polynomial models where in the 5th degree polynomial, there is overfitting.

Question 4:

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

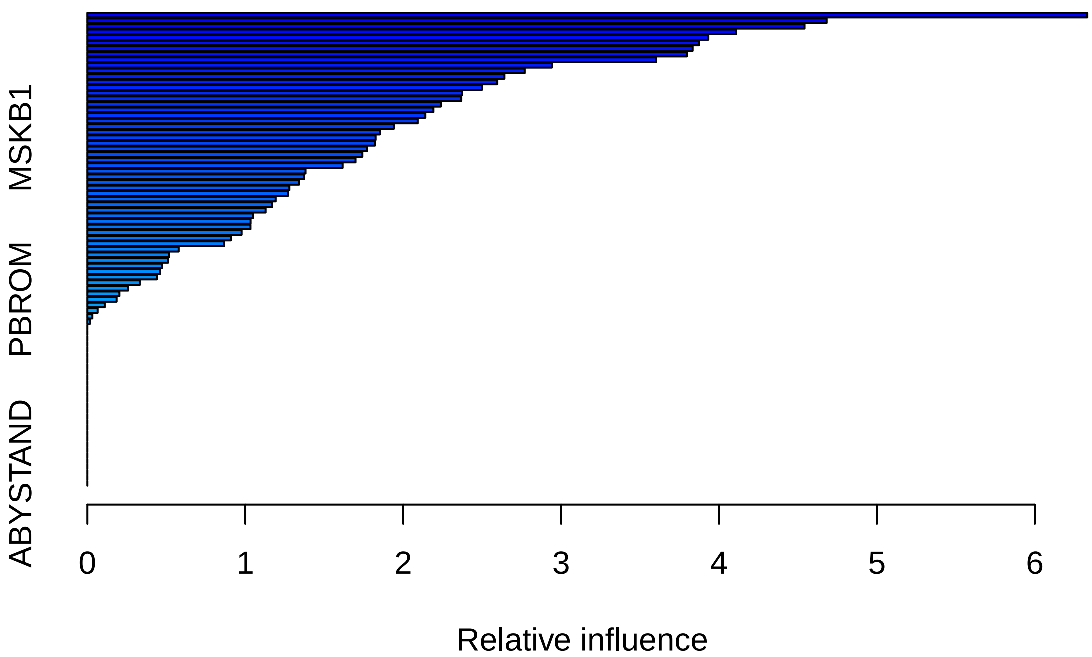
b)

```
caravan.boost <- gbm(ifelse(Purchase=='Yes',1,0)~.,data=caravan.train,distribution='bernoulli',n.trees=5
```

```
## 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(caravan.boost)
```




```
##                 var    rel.inf
## PPERSAUT PPERSAUT 6.33239053
## MOSTYPE   MOSTYPE 4.68157435
## MGODPR     MGODPR 4.54160776
## MKOOPKLA MKOOPKLA 4.10749602
## MGODGE     MGODGE 3.93225063
## MAUT2       MAUT2 3.87330240
## MINK3045 MINK3045 3.83299932
## MBERARBG MBERARBG 3.79724341
## PBRAND     PBRAND 3.60108320
## MBERMIDD MBERMIDD 2.94131196
## MOPLHOOG MOPLHOOG 2.76944933
## MSKC         MSKC 2.64111452
## MAUT1       MAUT1 2.59598525
## MSKB2       MSKB2 2.49838352
## MFALLEEN MFALLEEN 2.37117727
## PWAPART   PWAPART 2.36801957
## MHKOOP     MHKOOP 2.23837464
## MSKA         MSKA 2.19137728
## MOPLMIDD MOPLMIDD 2.14041374
## ABRAND     ABRAND 2.09180295
## MBERHOOG MBERHOOG 1.94026991
## MFGEKIND MFGEKIND 1.85280282
## MSKB1       MSKB1 1.82538850
## MBERARBO MBERARBO 1.82116486
## MZFONDS   MZFONDS 1.77182697
## MRELOV     MRELOV 1.74139713
## MINKM30   MINKM30 1.69827686
## MRELGE     MRELGE 1.61647196
## MINKGEM   MINKGEM 1.38156010
## MSKD         MSKD 1.37251149
## MAUT0       MAUT0 1.34079070
## MZPART     MZPART 1.27924230
```

```
## MINK7512 MINK7512 1.27132905
## MBERZELF MBERZELF 1.19247657
## MGODRK     MGODRK 1.17039835
## MHHUUR     MHHUUR 1.12871995
## MRELSA     MRELSA 1.04862164
## MINK4575 MINK4575 1.03352955
## APERSAUT APERSAUT 1.03286707
## MOPLLAAG MOPLLAAG 0.97681679
## MGODOV     MGODOV 0.90979881
## MFWEKIND MFWEKIND 0.86603360
## MGEMLEEF MGEMLEEF 0.57815650
## PLEVEN     PLEVEN 0.51729082
## MBERBOER MBERBOER 0.51149470
## MGEMOMV   MGEMOMV 0.47158590
## PFIETS     PFIETS 0.46155173
## MOSHOOFD MOSHOOFD 0.44006245
## PMOTSCO   PMOTSCO 0.33159909
## MINK123M MINK123M 0.25847589
## MAANTHUI MAANTHUI 0.20297135
## PBROM       PBROM 0.18523128
## PBYSTAND PBYSTAND 0.10995672
## PWALAND   PWALAND 0.06510213
## ALEVEN     ALEVEN 0.03149891
## PTRACTOR PTRACTOR 0.01536989
## PWABEDR   PWABEDR 0.00000000
## PBESAUT   PBESAUT 0.00000000
## PVRAAUT   PVRAAUT 0.00000000
## PAANHANG PAANHANG 0.00000000
## PWERKT     PWERKT 0.00000000
## PPERSONG PPERSONG 0.00000000
## PGEZONG   PGEZONG 0.00000000
## PWAOREG   PWAOREG 0.00000000
## PZEILPL   PZEILPL 0.00000000
## PPLEZIER PPLEZIER 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
## 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
```

The most important are PPERSAUT, MGODGE, MOSTYPE, MAUT2, MKOOPKLA, MBERHOOG, MSKC, MGODPR, MAUT1, PBRAND

c)

```
bag.caravan <- randomForest(Purchase~.,data=caravan.train,mtry=10,importance=TRUE)
print(bag.caravan)
```

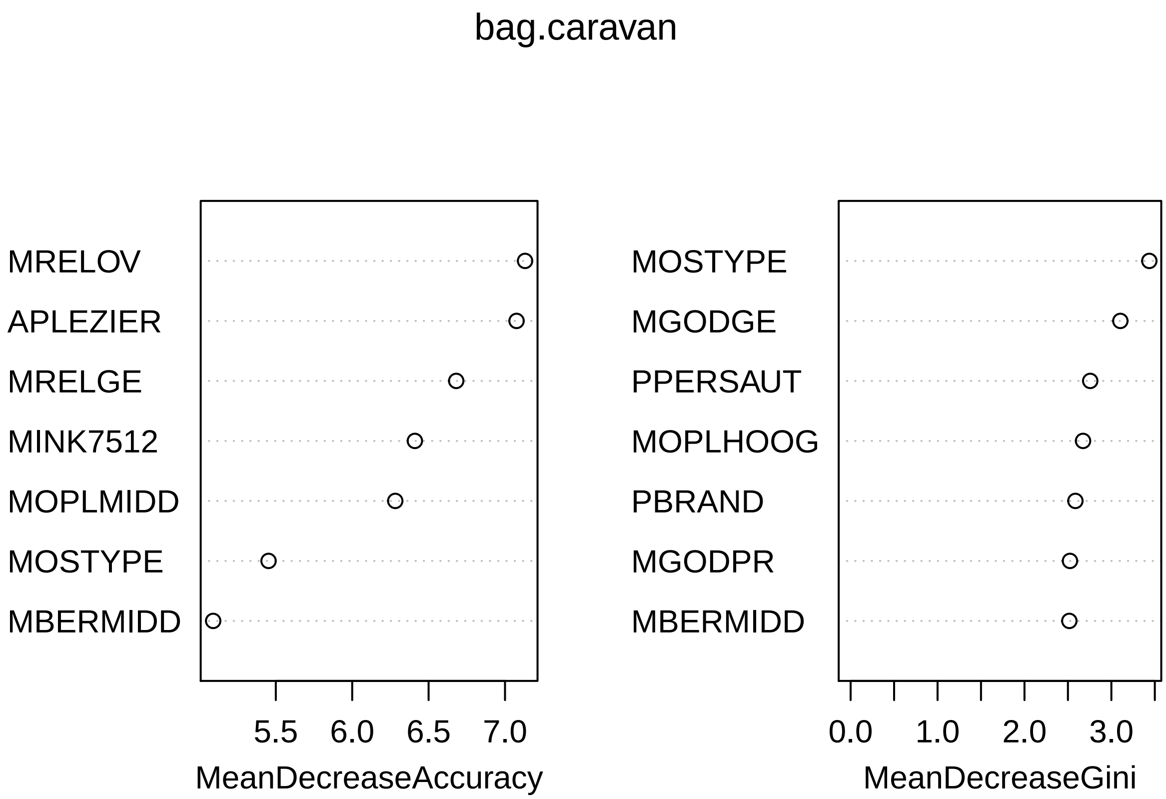
```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = caravan.train, mtry = 10,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 6%
## Confusion matrix:
##      No Yes class.error
## No  938   3 0.003188098
## Yes  57   2 0.966101695
```

```
importance(bag.caravan)
```

```
##                  No         Yes MeanDecreaseAccuracy MeanDecreaseGini
## MOSTYPE   4.22586946  3.64315143           5.45251007     3.4366119895
## MAANTHUI  2.24592703 -0.10813186           2.15493521     0.6192573429
## MGEMOMV   2.66343916 -2.05649894           2.31785636     1.0245675067
## MGEMLEEF  3.60507138 -0.89411341           3.23002571     1.1337613499
## MOSHOOFD  3.10121308  1.75738770           3.47148556     2.0485654454
## MGODRK    2.29875935  2.36310291           2.79800041     1.1791369906
## MGODPR    2.19726548  2.61516057           2.89962103     2.5235524574
## MGODOV    2.98230909  3.34828308           3.87492325     1.6258708262
## MGODGE    3.47478547  5.59673150           5.01516921     3.1031895457
## MRELGE    6.58644990 -0.01856565           6.68067230     2.1381398297
## MRELSA    2.71777704  1.21628606           3.02826517     1.3746425402
## MRELOV    7.41656946 -1.19473238           7.13121025     1.6921727018
## MFALLEEN  4.97351373 -0.86882657           4.70318639     1.5845846031
## MFGEKIND  2.27403058 -0.56256273           1.97402781     1.8733226987
## MFWEKIND  2.96359312  0.03560781           2.87025657     2.3079492469
## MOPLHOOG  2.60702718  7.53544654           5.00513307     2.6738090747
## MOPLMIDD  6.04408850  0.11452175           6.28174575     2.4806640378
## MOPLLAAG  3.77115615 -0.22701102           3.83494247     1.8519381828
## MBERHOOG  1.67561807  1.36456425           1.97691634     1.7608496339
## MBERZELF  1.63007447 -0.42633843           1.42765540     0.6854204973
## MBERBOER  1.26048009  2.53338890           1.86021084     0.5000772144
## MBERMIDD  4.00732361  4.61225047           5.09004849     2.5162927063
## MBERARBG  3.63297615 -0.92517692           3.42569411     2.2092151560
## MBERARBO  1.78385884  1.86845143           2.21146850     1.8903425277
## MSKA      2.41942302  3.58898553           3.28570980     2.0469477530
## MSKB1     3.25284681  1.27554044           3.57608697     2.1037065842
## MSKB2     1.30216491 -1.92211111           0.80362741     1.8642096672
## MSKC      4.41706137  1.42577366           4.75235981     2.2439975574
## MSKD      2.85710895 -0.64809070           2.78717280     1.0372156074
## MHHUUR    1.53819445  2.14612031           2.26099875     2.1848714893
## MHKOOP    0.93655713  2.31153138           1.52041634     2.0830810016
## MAUT1     3.24412483 -0.69194662           2.94013588     1.7861019199
```

```
## MAUT2     3.72275708  2.53974914           4.32493634    1.7769373928
## MAUT0     5.54541139 -1.37256183           4.97671650    1.5998312547
## MZFONDS   4.43670773 -1.58893839           4.12688138    1.9153719057
## MZPART    3.91220756 -0.16571571           3.96041450    2.0824073347
## MINKM30   1.67300832  1.64559250           2.10041063    1.9223211750
## MINK3045  2.19398147  0.35078794           2.19724688    2.0780230812
## MINK4575  0.48640169  2.15539133           1.04555495    1.7669555994
## MINK7512  6.49931491  0.18887338           6.41010518    1.7311970790
## MINK123M  1.16613535 -1.53465241           0.64934078    0.3740222121
## MINKGEM   3.64655137  2.11532463           4.11454870    1.5080182029
## MKOOPKLA  3.16319770  4.52351827           4.37328059    2.3962418748
## PWAPART  -0.99593876  5.60068750           0.58094260    2.2137315281
## PWABEDR  -2.50276997  0.00000000          -2.50345517    0.1714360869
## PWALAND  -1.29529432  0.00000000          -1.29203342    0.0778211748
## PPERSAUT  2.67979491  4.65681558           3.83274599    2.7566202356
## PBESAUT   0.00000000  0.00000000           0.00000000    0.0136190476
## PMOTSCO  -2.12826332  1.92940338          -1.44304772    0.7660871007
## PVRAAUT   0.00000000  0.00000000           0.00000000    0.0000000000
## PAANHANG -1.25710232 -0.38801582          -1.36248884    0.2364318860
## PTRACTOR  2.42051363 -1.00100150           2.37525664    0.2318642854
## PWERKT    0.00000000  0.00000000           0.00000000    0.0000000000
## PBROM     5.26591919 -2.77568231           4.51198606    0.5085966710
## PLEVEN   -0.97476142 -1.92161176          -1.39502662    0.7118244102
## PPERSONG  0.00000000  0.00000000           0.00000000    0.0141666667
## PGEZONG  -0.15809927 -1.41705050          -0.41858174    0.7694920872
## PWAOREG   2.81484570  2.02497635           2.90884524    0.8812189040
## PBRAND   -3.58089002  2.95864867          -2.40935106    2.5860106532
## PZEILPL   0.00000000  0.00000000           0.00000000    0.3281145859
## PPLEZIER  2.16667589  6.52194564           5.01787587    1.9724610502
## PFIETS   -0.41822088 -1.00100150          -0.84730243    0.1492519751
## PINBOED   0.01038308  0.00000000           0.00774900    0.0717968597
## PBYSTAND  0.64454693  0.12679360           0.59637233    0.7399999754
## AWAPART   0.45557405  3.01992523           1.27308269    1.2784046535
## AWABEDR   1.55057816  0.00000000           1.56112425    0.0945483407
## AWALAND  -0.02084676  0.00000000          -0.01159185    0.0489112049
## APERSAUT  1.44026542  0.63893233           1.59321121    1.9354389527
## ABESAUT   0.00000000  0.00000000           0.00000000    0.0060555556
## AMOTSCO  -0.37711214 -1.44266732          -0.78390848    1.0545432975
## AVRAAUT   0.00000000  0.00000000           0.00000000    0.0000000000
## AAANHANG  1.43250801 -1.06379001           1.18633521    0.2027798700
## ATRACTOR  0.60592718  0.00000000           0.62272584    0.0696899284
## AWERKT    0.00000000  0.00000000           0.00000000    0.0026666667
## ABROM     3.72014180 -1.27114277           3.42592355    0.3839851753
## ALEVEN   -1.03108340 -1.65622411          -1.49922615    0.2910754316
## APERSONG  0.00000000  0.00000000           0.00000000    0.0006666667
## AGEZONG  -1.49365484 -1.00100150          -1.57689488    0.4790929279
## AWAOREG   3.09092090  2.19038646           3.52845439    0.6990032789
## ABRAND   -0.91105441  0.49191323          -0.70446225    1.9973897707
## AZEILPL   0.00000000  0.00000000           0.00000000    0.3672022742
## APLEZIER  3.76255651  7.48986277           7.07551891    1.6974487337
## AFIETS   -1.46754558  0.00000000          -1.45887530    0.3540342422
## AINBOED  -0.91311040 -1.00100150          -1.17901932    0.1056886577
## ABYSTAND  0.16507441  1.45740185           0.47880017    0.5062010183
```

```r
varImpPlot(bag.caravan,n.var=7)
```



The OOB estimate error is 6.2%. 10 variables were subsampled, 500 trees used to fit the data. No the order of importance is not similar

d)

```r
caravan.test.boost <- predict(caravan.boost,newdata = caravan.test,type='response')
```

```
## Using 500 trees...
```

```r
new.test.boost <- c()
for (i in c(1:4822)) {
  if (caravan.test.boost[i]>0.2){
    new.test.boost[i] <- 'Yes'
  }
  else{
    new.test.boost[i] <- 'No'
  }
}
error <- table(pred=new.test.boost,truth=caravan.test$Purchase)
error
```

```
##      truth
## pred    No  Yes
##    No  4335  260
##    Yes  198   29
```

```r
test.error <- 1-sum(diag(error))/sum(error)
test.error
```

```
## [1] 0.09498134
```

```r
caravan.random.forest.test <- predict(bag.caravan,newdata=caravan.test,type='prob')
yes <- caravan.random.forest.test[,2]
new.rf <- c()
for (i in c(1:4822)) {
  if (yes[i]>0.2){
    new.rf[i] <- 'Yes'
  }
  else{
    new.rf[i] <- 'No'
  }
}
rf.error <- table(pred=new.rf,truth=caravan.test$Purchase)
rf.error
```

```
##      truth
## pred    No  Yes
##   No  4282  244
##   Yes  251   45
```

```r
test.rf.error <- 1-sum(diag(rf.error))/sum(rf.error)
test.rf.error
```

```
## [1] 0.1026545
```

$46/309 \approx 0.149$ is the fraction of people who actually make a purchase out of those predicted to make a purchase

Question 5

```r
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.
```

a)

```r
drug_use <- drug_use%>%mutate(recent_cannabis_use=factor(ifelse(Cannabis>='CL3','Yes','No'),levels=c('No
drug_use_sub <- drug_use%>%select(Age:SS,recent_cannabis_use)
drug.samp <- sample(1:nrow(drug_use_sub),1500)
drug.train <- drug_use_sub[drug.samp,]
drug.test <- drug_use_sub[-drug.samp,]
drug.svm <- svm(recent_cannabis_use~.,data=drug.train,kernal='radial',cost=1)
drug.pred <- predict(drug.svm,drug.test)
table(predict=drug.pred,truth=drug.test$recent_cannabis_use)
```

```
##        truth
## predict  No Yes
##     No  156  37
##     Yes  28 164
```

b)

```r
drug.tune <- tune(svm,recent_cannabis_use~.,data=drug.train,kernel='radial',ranges = list(c(0.001,0.01,0
summary(drug.tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   Var1
##  0.001
##
## - best performance: 0.1926667
##
## - Detailed performance results:
##    Var1     error dispersion
## 1 1e-03 0.1926667 0.02988868
## 2 1e-02 0.1926667 0.02988868
## 3 1e-01 0.1926667 0.02988868
## 4 1e+00 0.1926667 0.02988868
## 5 1e+01 0.1926667 0.02988868
## 6 1e+02 0.1926667 0.02988868
```

```r
bestmodel <- drug.tune$best.model
tune.pred <- predict(bestmodel,drug.test)
table(predict=tune.pred,truth=drug.test$recent_cannabis_use)
```

```
##        truth
## predict  No Yes
##     No  156  37
##     Yes  28 164
```