Homework Assignment 4

Jasmine Kwok

December 10, 2020

```
library(knitr)
# set global chunk options: images will be 7x5 inches
knitr::opts_chunk$set(fig.width=7, fig.height=5)
options(digits = 4)

## indents are for indenting r code as formatted text
## They may need to be adjusted depending on your OS
# if your output looks odd, increase or decrease indent
indent1 = '    '
indent2 = '        '
indent3 = '      '
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(ROCR)
library(e1071)
library(imager)
```

```
## Loading required package: magrittr

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract

##
## Attaching package: 'imager'

## The following object is masked from 'package:magrittr':
##
##     add

## The following object is masked from 'package:randomForest':
##
##     grow

## The following object is masked from 'package:stringr':
##
##     boundary

## The following object is masked from 'package:tidyr':
##
##     fill

## The following objects are masked from 'package:stats':
##
##     convolve, spectrum
```

```
## The following object is masked from 'package:graphics':
##
##     frame

## The following object is masked from 'package:base':
##
##     save.image
```

1. Fundamentals of the bootstrap

a) Given a sample of size n, what is the probability that any observation j is not in a bootstrap sample? Express your answer as a function of n. The probability of selecting x_j is $\frac{1}{n}$, then the probability of x_j not in a bootstrap sample is $1 - \frac{1}{n}$. FSo for any j-th observation is not in the bootstrap sample is $(1 - \frac{1}{n})^n$

b) Compute the above probability for n=1000.

```
(1-1/1000)**1000
```

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

0.3676954

c) print the number of missing observations

```
set.seed(10)
rand <- sample(1:1000, replace=TRUE)
num_uniq <- length(unique(rand))
num_uniq
```

```
## [1] 639
```

```
non_uniq <- length(rand)-length(unique(rand))
non_uniq # missing observations
```

```
## [1] 361
```

d) By November 19, 2015, Stephen Curry, an NBA basketball player regarded as one of the best players currently in the game, had made 62 out of 126 three point shot attempts (49.2%). His three point field goal percentage of 0.492, if he maintains it, will be one of the best all time for a single season. Use bootstrap resampling on a sequence of 62 1's (makes) and 64 0's (misses). For each bootstrap sample compute and save the sample mean (e.g. bootstrap FG% for the player). Use 1000 bootstrap samples to plot a histogram of those values.

Compute the 95% bootstrap confidence interval for Stephen Curry's "true" end-of-season FG% using the quantile function in R. Print the endpoints of this interval. However, this estimate, and the associated uncertainty, exclude information about his career performance as well as the typical shooting skill for other players in the league. For reference, prior to this year, Stephen Curry had made about 43% of all three point shots in his career. Despite the fact that the bootstrap histogram shows that it is about equally likely that Curry's true skill is greater or elss than 0.492, why do you expect that his end-of-season field goal percentage will in fact be lower than his percentage on 11/19? Hint: look up the phenomenon known as "regression to the mean".

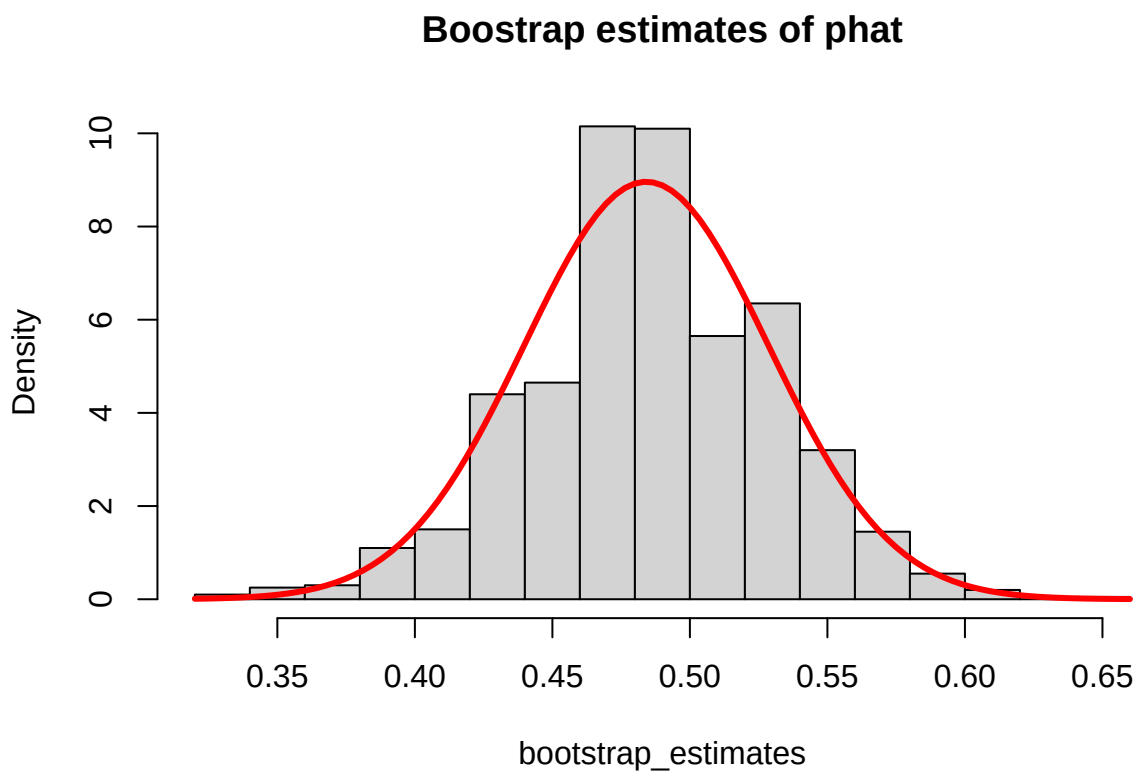
```r
B<-1000
# creating a list of 62 1s and 64 0s (normally distributed)
shootseq <- rbinom(126,1, p=0.492)
phat <- mean(shootseq)
sd_hat <- sqrt(phat*(1-phat)/126)
c(phat,sd_hat)
```

```
## [1] 0.48412698 0.04452109
```

```r
# finding mean of each sample
bootstrap_estimates <- sapply(1:1000,function(i)mean(sample(shootseq,replace=TRUE)))
head(bootstrap_estimates)
```

```
## [1] 0.4761905 0.5317460 0.5000000 0.5158730 0.5714286 0.5396825
```

```r
# another method
# store <- vector()
#for(i in 1:1000){
  #store[i] <- mean(sample(shootseq,replace=TRUE))
#}

# create a histogram
#hist(store, freq=FALSE, breaks = 20, main="Boostrap estimates of phat")
hist(bootstrap_estimates, freq=FALSE, breaks = 20, main="Boostrap estimates of phat")
curve(dnorm(x,phat,sd_hat), add = TRUE, col="red", lwd=3)
```



```r
# compute 95% confidence interval
quantile(bootstrap_estimates,c(0.025,0.975))
```

```
##      2.5%     97.5%
## 0.3968254 0.5714286
```

We expect that Curry's his end-of-season field goal percentage to be lower than the percentage of 0.492 on 11/19 due to the regression to the mean. Regression to the mean suggests a phenomenon in which a future point which is his end-of-season field goal percentage will be closer to the mean which is 0.48413 in this case rather than towards an extreme variable.

2. Eigenfaces

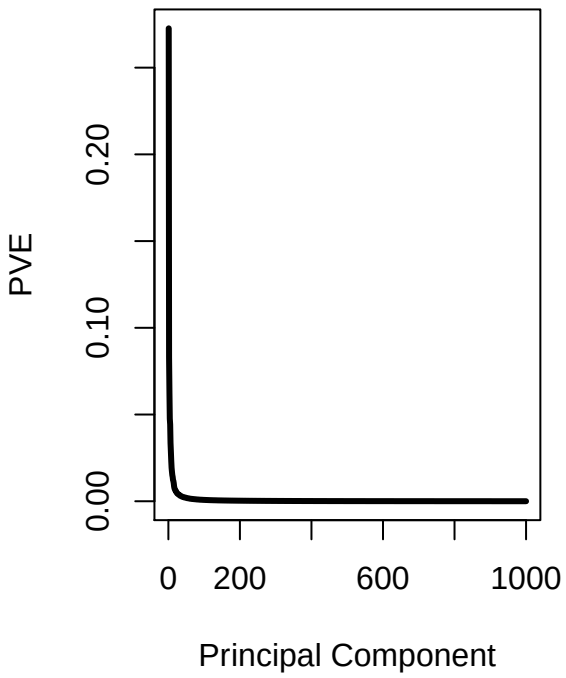
```r
load("faces_array.RData")
face_mat <- sapply(1:1000, function(i) as.numeric(faces_array[, , i])) %>% t
plot_face <- function(image_vector) {
plot(as.cimg(t(matrix(image_vector, ncol=100))), axes=FALSE, asp=1)
}
```

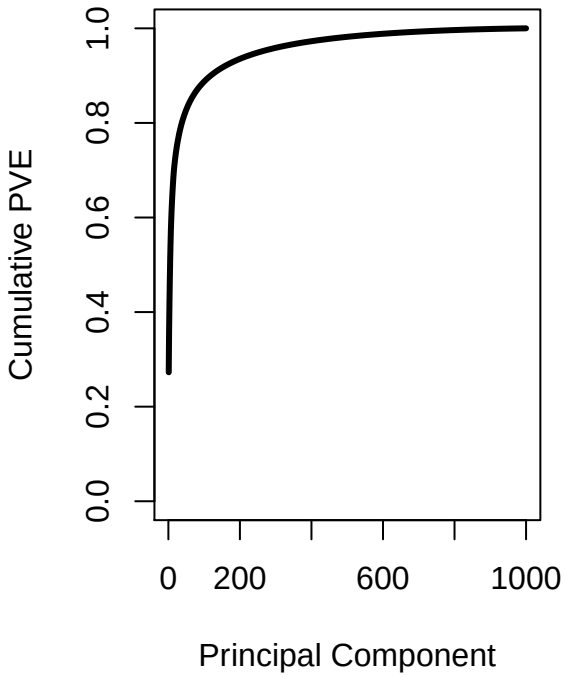
a) Find the "average" face in this dataset by averaging all of the columns in face_mat. Plot the average face by calling plot_face on the average.

```r
# averaging all columns in face_mat to find average face
ave_face <- colMeans(face_mat)

# plotting the average face
plot_face(ave_face)
```

b) Run PCA on face_mat setting center=TRUE and scale=FALSE. In class we mentioned that in general it is best if scale=TRUE because it puts all variables on the same scale and we don't have to worry about the units of the variables (remember, the scale of the variables affects our results). In general, this is good practice, especially when the predictor variables are of mixed types. Here, each variable represents a single pixel intensity (in black & white) and so all variables already have the same units and same scale (minimum of 0 and maximum of 255). In this case, setting scale=FALSE actually seems to give slightly better results. Plot the PVE and cumulative PVE from the PCA. How many PCs do you need to explain at least 50% of the total variation in the face images?

```
facemat_pca <- prcomp(face_mat, scale =FALSE, center = TRUE)
sdev <- facemat_pca$sdev
pve <- sdev^2/sum(sdev^2)
cumulative_pve <- cumsum(pve)

## This will put the next two plots side by side
par(mfrow=c(1, 2))

## Plot proportion of variance explained
plot(pve, type="l", lwd=3, xlab="Principal Component",
     ylab="PVE")
plot(cumulative_pve, type="l", lwd=3, xlab="Principal Component", ylab="Cumulative PVE", ylim=c(0,1))
```






```
# the number of PCs needed to explain at least 50% of total variation in the images
min(which(cumulative_pve>=0.5))
```

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

c) Plot the first 16 principle component directions as faces using the plot_face function (these are the columns of the rotation matrix). Early researchers termed these "eigenfaces" since they are eigenvectors of the matrix of faces. The code below will adjust the margins of you plot and specifies a layout for the 16 images. par(mfrow=c(4,4)) specifies a grid of 4 x 4 images. Each time you call plot_face it will plot the next face in one of the new grid cells. All you need to do is call plot_face 16 times (please use a for loop). Note that these images describe "directions" of maximum variability in the face images. You should interpret light and dark regions in the eigenfaces as regions of high contrast, e.g. your interpretation should not change if you inverted black and white in the images.

```
# plot the first 16 PCs using plot_face function
for (i in 1:16){
  plot_face(facemat_pca$rotation[,i])
}
```

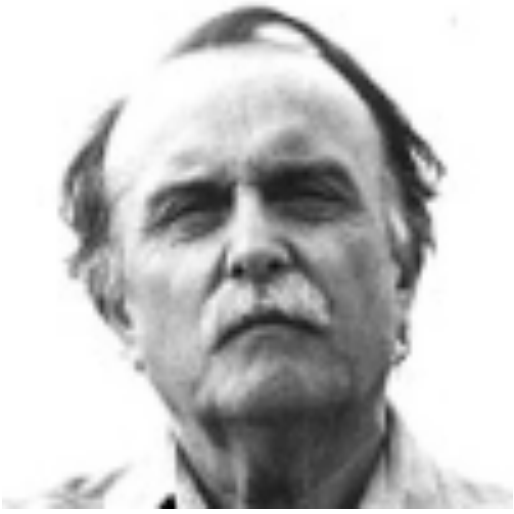
```
par(mfrow=c(4,4))
```

d) In this part, we will examine faces that have the highest and lowest values for specific PCs. Plot the faces with the 5 largest values on PC1 and the 5 smallest values for PC1. Based on the example faces, and the first eigenface from the previous part and the 10 example images, what aspect of variability in the face images is captured by the first component.

```
# plot the faces with 5 largest values on PC1
largest_pc1 <- order(facemat_pca$x[, 1], decreasing = TRUE)[1:5]
largest_pc1
```

```
## [1] 355 358 310 227 644
```

```
for (i in largest_pc1){
  plot_face(face_mat[i,])
}
```



```
# plot the faces with 5 smallest values on PC1
smallest_pc1 <- order(facemat_pca$x[, 1])[1:5]
smallest_pc1
```

```
## [1] 597 435 578 206  69
```

```
for (i in smallest_pc1){
  plot_face(face_mat[i,])
}
```

Based on the example faces and the first eigenface from the previous part and the 10 example images, the first component captures the variability between lightness and darkness in the face images which differentiates the individual and the background. The largest values in PC1 has a light background and darker individual while the smallest values of PC1 has a dark background and lighter individual.

e) Repeat part d) but now display example faces with the largest and smallest values on principal component 5. Again, discuss what aspect of variability in the face images is best captured by this principal component. Based on your results, which principal component, (1 or 5) would be more useful as a feature in a face recognition model (e.g. a model which predicts the identity of the individual in an image)

```
# plot the faces with 5 largest values on PC5
largest_pc5 <- order(facemat_pca$x[, 5], decreasing = TRUE)[1:5]
largest_pc5
```

```
## [1]  64 512 842 683 921
```

```
for (i in largest_pc5){
  plot_face(face_mat[i,])
}
```

```
# plot the faces with 5 smallest values on PC5
smallest_pc5 <- order(facemat_pca$x[, 5])[1:5]
smallest_pc5
```

```
## [1] 818 420 391  54 788
```

```
for (i in smallest_pc5){
  plot_face(face_mat[i,])
}
```

The variability in hair length is captured in this principal component. Faces with largest PC5 have long hair while faces with the smallest PC5 have short hair. Based on these results, I think PC5 would be more useful in facial recognition as it identifies a specific feature, the hair, which is an indicator of identity for the individual in the image.

3. Logistic regression with polynomial features

a) In class, we have used polynomial linear regression several times as an example for model complexity and the bias variance tradeoff. We can also introduce polynomial logistic regression models to derive more sophisticated classification functions by introducing additional features. Use read_csv to load nonlinear.csv and plot the data. Plot each point colored according to its class, Y.

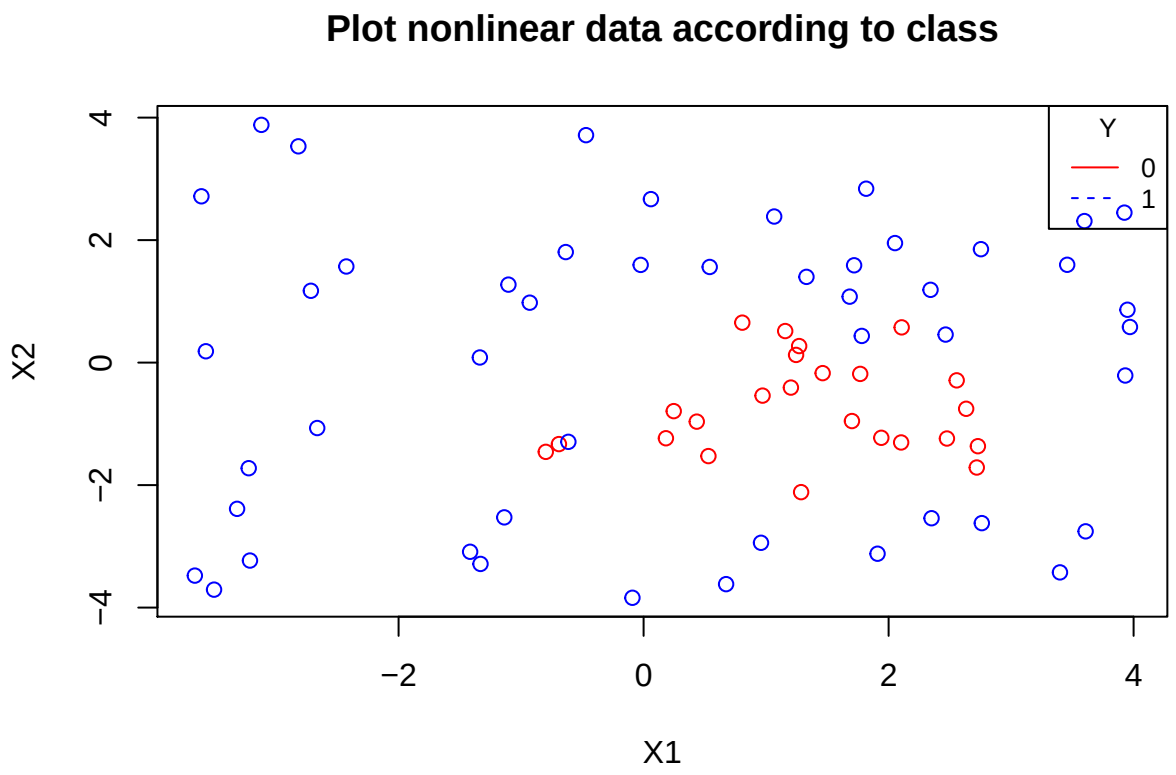
```
# reading in data
nonlinear_data <- read_csv("nonlinear.csv")
```

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

```r
head(nonlinear_data)
```

```
## # A tibble: 6 x 4
##       Z     X1     X2     Y
##   <dbl>  <dbl>  <dbl> <dbl>
## 1     7  2.11   0.575     0
## 2    12  0.971 -0.539     0
## 3    21  0.806  0.652     0
## 4    24  1.24   0.125     0
## 5    28 -0.798 -1.46      0
## 6    41  1.46  -0.172     0
```

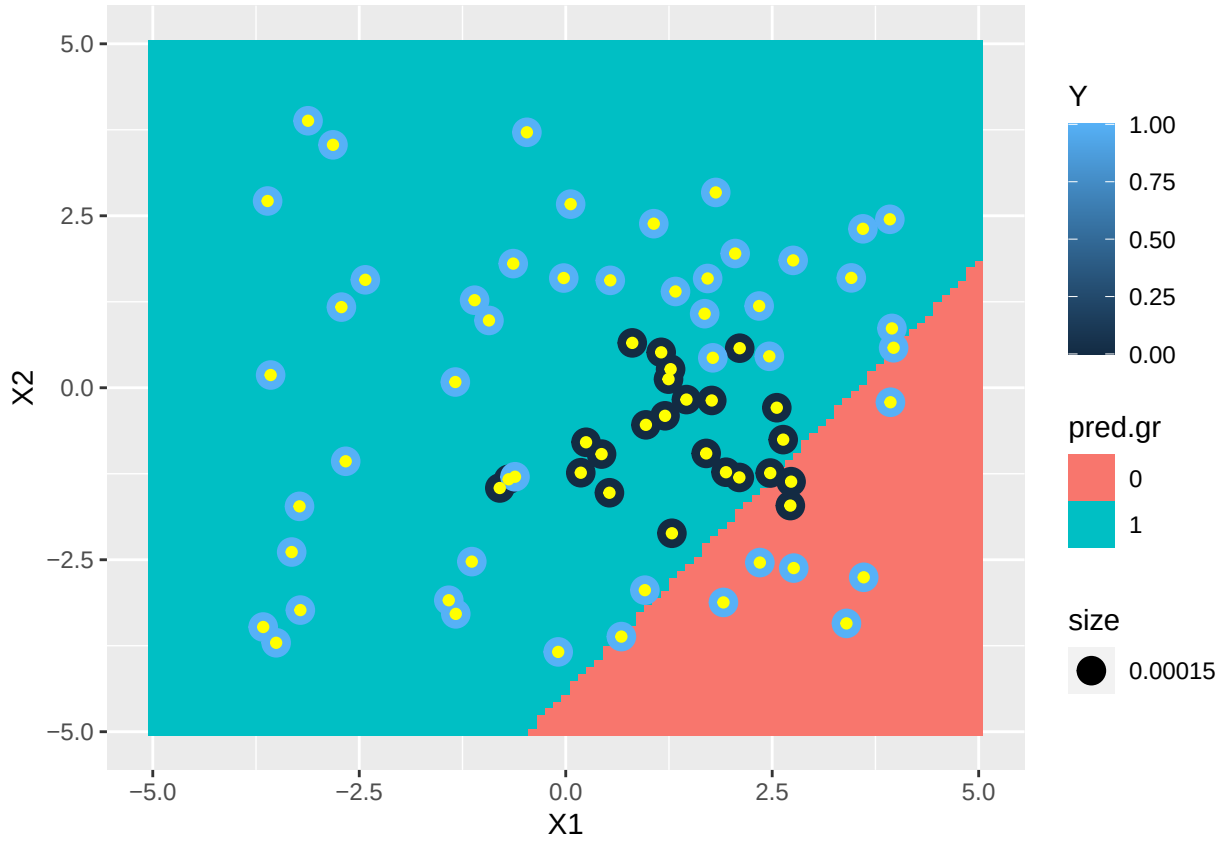
```r
# plotting the non linear data according to class Y
plot(nonlinear_data$X1, nonlinear_data$X2, col=ifelse(nonlinear_data$Y == 0, "red", "blue"),
     main="Plot nonlinear data according to class", xlab="X1", ylab = "X2")
legend("topright",title = "Y", legend=c("0", "1"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```



b) Fit a logistic regression model of Y on X1 and X2. The decision boundary can be visualized by making predictions of class labels over finely sampled grid points that cover your region (sample space) of interest. For each point in gr, predict a class label using the logistic regression model. You should classify based on the probability being greater or less than 1/2. Visualize your predictions at each point on the grid using the geom_raster function. This function colors in rectangles on the defined grid and is a good way to visualize your decision boundary. Set the fill aesthetic to your predicted label

and outside of the aes use alpha=0.5 to set the transparency of your predictions. Plot the observed data, colored by label, over the predictions using geom_point.

```
#fit logistic regression model of Y on X1 and X2
fit_X1X2 <- glm(Y~X1+X2,data = nonlinear_data, family = binomial)

# 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

# predict class label for each point
pred.gr <- predict(fit_X1X2, gr, type = "response" )
newpred.gr <- as.factor(ifelse(pred.gr>=0.5,1, 0))
gr <- gr %>% add_column(pred.gr=newpred.gr)
# create a geom_raster plot
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = pred.gr)) + geom_point(data =
       nonlinear_data,aes(col=Y, size=0.00015)) + geom_point(data = nonlinear_data, colour="yellow")
```

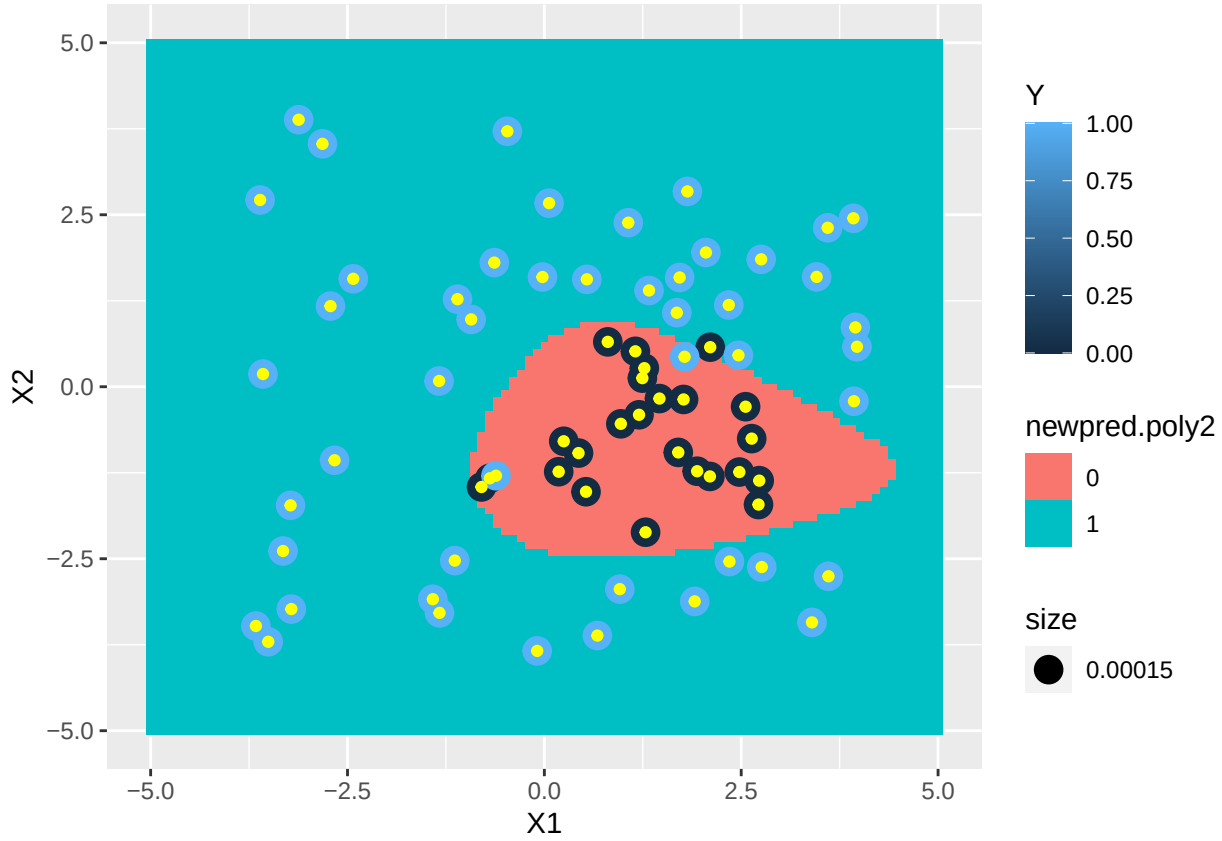


c) Fit a model involving 2nd degree polynomial of X1 and X2 with interaction terms. You should use the poly() function. Inspect result of the fit using summary(). Plot the resulting decision boundary.

```
#fit a model of 2nd degree polynomial of X1 and X2 with interaction terms
fit_2poly<- glm(Y~poly(X1, 2)*poly(X2,2),data = nonlinear_data, family = binomial)
summary(fit_2poly)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 2) * poly(X2, 2), family = binomial,
##     data = nonlinear_data)
##
## 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
```

```r
# plot resulting decision boundary
# prediction values
pred.poly2 <- predict(fit_2poly, gr, type = "response" )
newpred.poly2 <- as.factor(ifelse(pred.poly2<=0.5,0,1))

# create a geom_raster plot
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = newpred.poly2)) + geom_point(data =
       nonlinear_data,aes(col=Y, size=0.00015))+geom_point(data = nonlinear_data, colour="yellow")
```



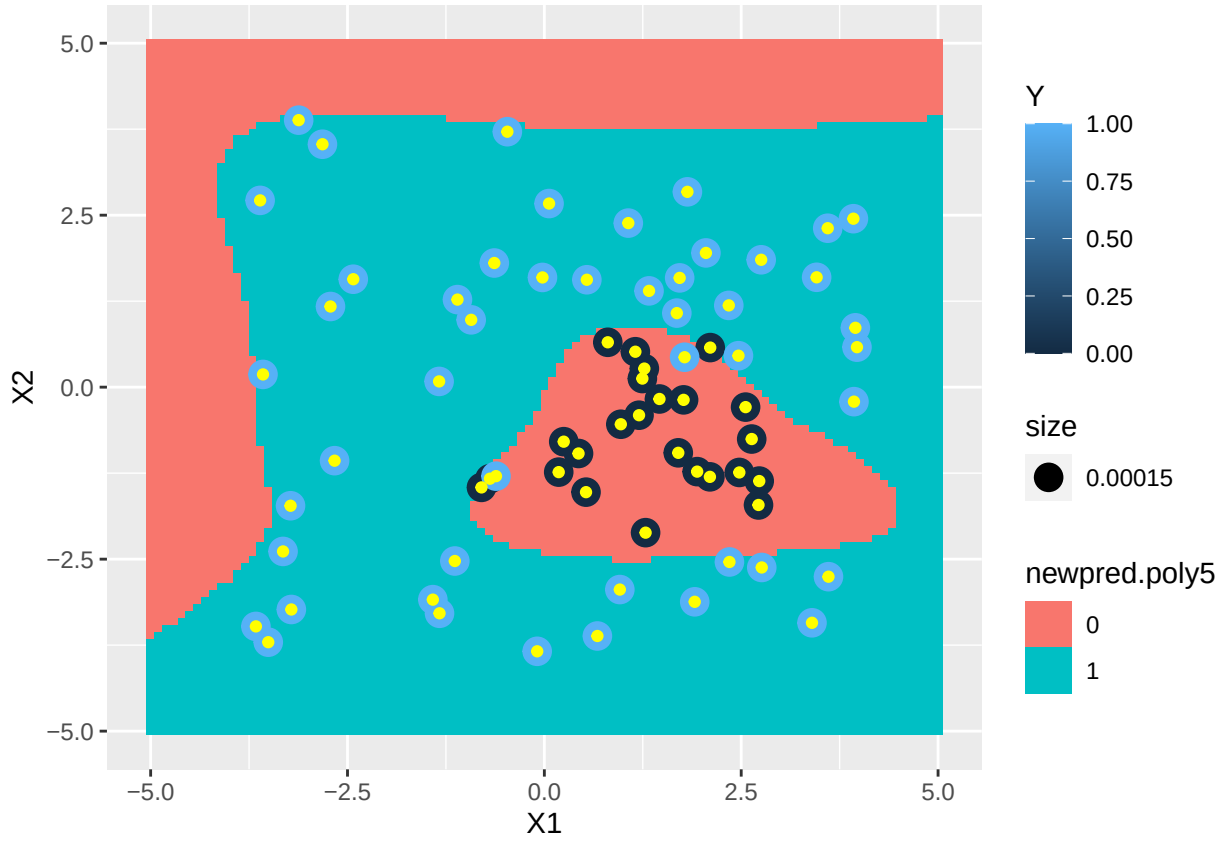
d) Using the same procedure, fit a logistic regression model with 5-th degree polynomials without any interaction terms. Inspect result of the fit using summary(). Plot the resulting decision boundary and discuss the result. Explain the reason for any strange behvaior.

```
# fit 5th degree polynomial with no interaction terms
fit_5poly<- glm(Y~poly(X1, 5)+poly(X2,5),data = nonlinear_data, family = binomial)
summary(fit_5poly)
```

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

```
# plot resulting decision boundary
# prediction values
pred.poly5 <- predict(fit_5poly, gr, type = "response" )
newpred.poly5 <- as.factor(ifelse(pred.poly5<=0.5,0,1))

# create a geom_raster plot
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = newpred.poly5)) + geom_point(data =
       nonlinear_data,aes(col=Y, size=0.00015)) +  geom_point(data = nonlinear_data, colour="yellow")
```



In this graph, there is a pink region on the top without any blue points which may suggest overfitting. I think the logistic regression model with 2nd degree polynomial and interaction terms may be a better model than the logistic regression model with 5th degree polynomial.

e) Qualitatively, compare the relative magnitudes of coefficients of in the two polynomial models and the linear model. What do you notice? Your answer should mention bias, variance and/or overfitting. The magnitudes of coefficients of the two polynomial models such as 14.4, -51.1, and 103.4 are much larger compared to the coefficients of the linear model such as 1.022, -0.289, and 0.232. The magnitude of coeffients of the 2nd degree polynomial is larger than the 5th degree polynomial. The polynomial models and non-linear and would have a higher variance and lower bias as they are more complex in comparison to the linear model which is more simple and has smaller bias and larger variance. The use of 5th degree polynomial model may be overfitting as the model is overly complex and has a strange behavior shown in the previous plot.

f) (231 required, 131 extra credit) Create 3 bootstrap replicates of the original dataset. Fit the linear model and the 5th order polynomial to each of the bootstrap replicates. Plot class predictions on the grid of values for each of both linear and 5th order fits, from each of the bootstrap samples. There should be six plots total. Discuss what you see in the context of your answer to the previous question.

```
set.seed(1)
# create 3 bootstrap replicates of original dataset
bootstrap.data1 <- nonlinear_data[c(sample(nrow(nonlinear_data),replace=TRUE)),]
bootstrap.data2 <- nonlinear_data[c(sample(nrow(nonlinear_data),replace=TRUE)),]
bootstrap.data3 <- nonlinear_data[c(sample(nrow(nonlinear_data),replace=TRUE)),]

# fit linear model to each boostrap replicate
fit_X1X2.boot1 <- glm(Y~X1+X2,data = bootstrap.data1, family = binomial)
fit_X1X2.boot2 <- glm(Y~X1+X2,data = bootstrap.data2, family = binomial)
fit_X1X2.boot3 <- glm(Y~X1+X2,data = bootstrap.data3, family = binomial)

# summary of linear models
summary(fit_X1X2.boot1)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = bootstrap.data1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6188  -1.4254   0.6961   0.8093   1.1416
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.891247   0.311908   2.857  0.00427 **
## X1          -0.007726   0.124429  -0.062  0.95049
## X2           0.210228   0.127022   1.655  0.09791 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 88.632  on 71  degrees of freedom
## Residual deviance: 85.720  on 69  degrees of freedom
## AIC: 91.72
##
## Number of Fisher Scoring iterations: 4
```

```r
summary(fit_X1X2.boot2)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = bootstrap.data2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6047  -1.3280   0.7354   0.8619   1.3654
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0374     0.3143   3.301 0.000964 ***
## X1           -0.1762     0.1282  -1.374 0.169345
## X2            0.2541     0.1477   1.721 0.085294 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.209  on 71  degrees of freedom
## Residual deviance: 86.140  on 69  degrees of freedom
## AIC: 92.14
##
## Number of Fisher Scoring iterations: 4
```

```r
summary(fit_X1X2.boot3)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = bootstrap.data3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7709   0.3040   0.5755   0.7150   1.3391
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.3451     0.3159   4.258 2.07e-05 ***
## X1           -0.2515     0.1460  -1.723   0.0849 .
## X2            0.2940     0.1586   1.854   0.0637 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 76.278  on 71  degrees of freedom
## Residual deviance: 70.180  on 69  degrees of freedom
## AIC: 76.18
##
## Number of Fisher Scoring iterations: 4
```

```r
# fit 5th degree polynomial model to each boostrap replicate
fit_5poly.boot1<- glm(Y~poly(X1, 5)+poly(X2,5),data = bootstrap.data1, family = binomial)
fit_5poly.boot2<- glm(Y~poly(X1, 5)+poly(X2,5),data = bootstrap.data2, family = binomial)
fit_5poly.boot3<- glm(Y~poly(X1, 5)+poly(X2,5),data = bootstrap.data3, family = binomial)

# summary of polynomial models
summary(fit_5poly.boot1)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 5) + poly(X2, 5), family = binomial,
##     data = bootstrap.data1)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.578e-04  -2.000e-08   2.000e-08   2.000e-08   3.261e-04
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1569.2    90327.5   0.017    0.986
## poly(X1, 5)1   7418.4   460840.0   0.016    0.987
## poly(X1, 5)2  13193.2   759789.2   0.017    0.986
## poly(X1, 5)3   8631.3   503700.4   0.017    0.986
## poly(X1, 5)4   1215.5   116559.8   0.010    0.992
## poly(X1, 5)5   6775.3   392919.5   0.017    0.986
## poly(X2, 5)1   -694.2   431852.8  -0.002    0.999
## poly(X2, 5)2   6363.2   409078.7   0.016    0.988
## poly(X2, 5)3  -1569.4   450561.0  -0.003    0.997
## poly(X2, 5)4   3000.7   194193.3   0.015    0.988
## poly(X2, 5)5   -752.3   194415.3  -0.004    0.997
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8.8632e+01  on 71  degrees of freedom
## Residual deviance: 9.6085e-07  on 61  degrees of freedom
## AIC: 22
##
## Number of Fisher Scoring iterations: 25
```

```r
summary(fit_5poly.boot2)
```

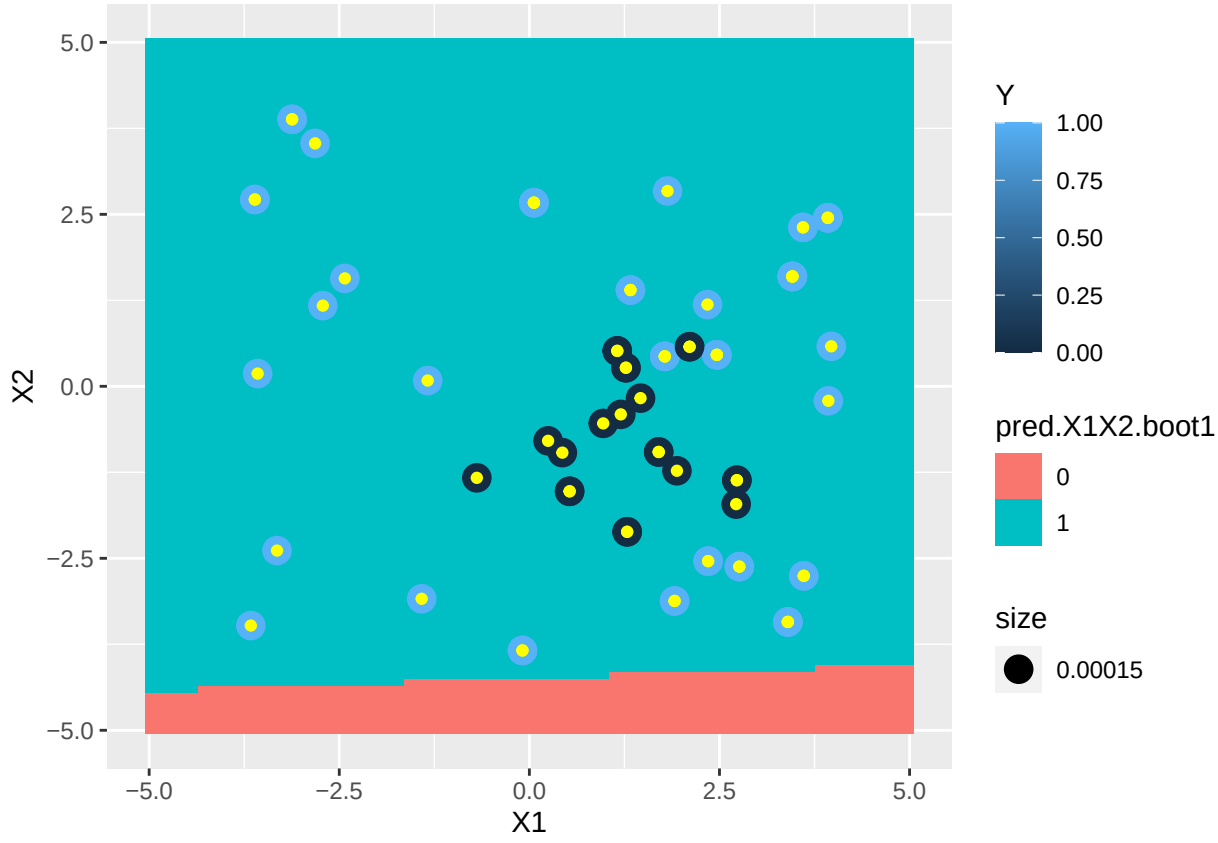
```
##
## Call:
## glm(formula = Y ~ poly(X1, 5) + poly(X2, 5), family = binomial,
##     data = bootstrap.data2)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.01711  -0.00011   0.00000   0.00001   2.28362
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)    73.860     58.785   1.256    0.209
## poly(X1, 5)1 -120.396    121.186  -0.993    0.320
## poly(X1, 5)2   69.353     90.301   0.768    0.442
## poly(X1, 5)3   26.980     92.596   0.291    0.771
## poly(X1, 5)4   -7.482     66.015  -0.113    0.910
## poly(X1, 5)5  -34.808     59.100  -0.589    0.556
## poly(X2, 5)1  450.656   1850.852   0.243    0.808
## poly(X2, 5)2  942.159    809.820   1.163    0.245
## poly(X2, 5)3  307.950   1556.465   0.198    0.843
## poly(X2, 5)4  356.133    321.183   1.109    0.268
## poly(X2, 5)5   83.821    482.491   0.174    0.862
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90.209  on 71  degrees of freedom
## Residual deviance: 11.279  on 61  degrees of freedom
## AIC: 33.279
##
## Number of Fisher Scoring iterations: 18
```

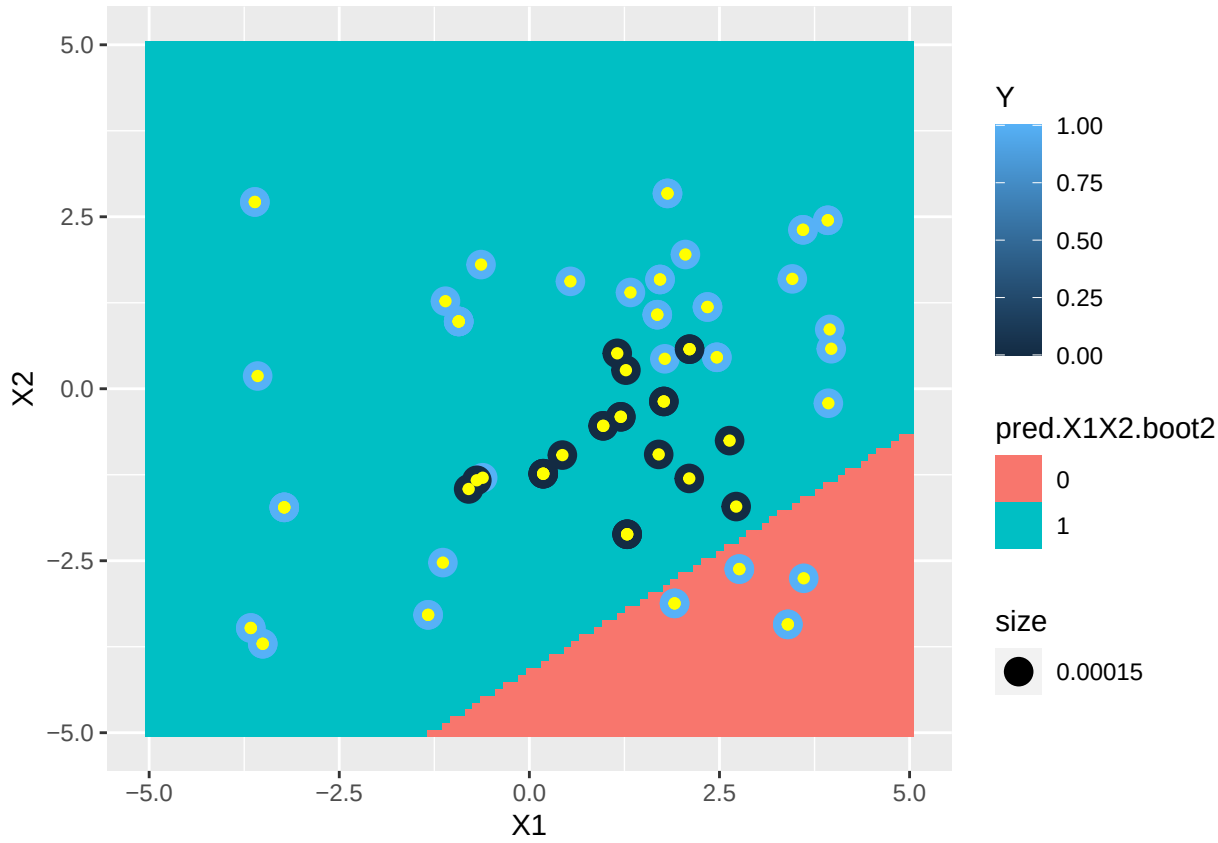
```
summary(fit_5poly.boot3)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 5) + poly(X2, 5), family = binomial,
##     data = bootstrap.data3)
##
## 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)   1.626e+15  7.909e+06  205630252   <2e-16 ***
## poly(X1, 5)1 -5.178e+15  7.752e+07  -66796057   <2e-16 ***
## poly(X1, 5)2  4.617e+15  7.037e+07   65609098   <2e-16 ***
## poly(X1, 5)3  6.218e+15  7.039e+07   88344305   <2e-16 ***
## poly(X1, 5)4 -4.997e+15  7.444e+07  -67124083   <2e-16 ***
## poly(X1, 5)5  1.888e+15  7.235e+07   26097752   <2e-16 ***
## poly(X2, 5)1  5.713e+14  6.901e+07    8278119   <2e-16 ***
## poly(X2, 5)2  9.048e+15  7.421e+07  121924533   <2e-16 ***
## poly(X2, 5)3 -1.047e+16  7.117e+07 -147173441   <2e-16 ***
## poly(X2, 5)4 -4.540e+14  6.900e+07   -6579588   <2e-16 ***
## poly(X2, 5)5  8.515e+15  8.122e+07  104835854   <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:  76.278  on 71  degrees of freedom
## Residual deviance: 144.175  on 61  degrees of freedom
## AIC: 166.17
##
## Number of Fisher Scoring iterations: 19
```

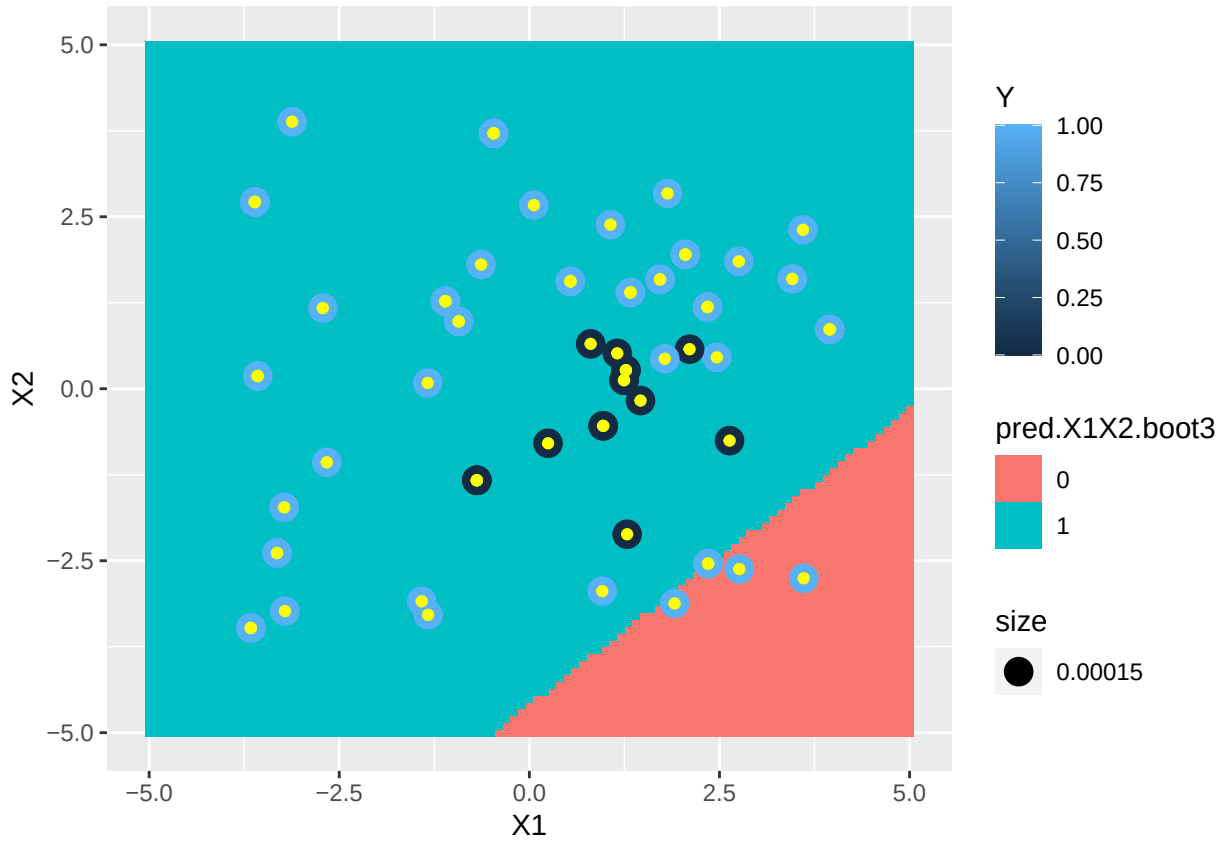
```
# predictions for linear model
# linear model bootstrap 1 predictions
pred.X1X2.boot1 <- predict(fit_X1X2.boot1, gr, type = "response")
pred.X1X2.boot1 <- as.factor(ifelse(pred.X1X2.boot1<=0.5,0,1))
# linear model bootstrap 2 predictions
pred.X1X2.boot2 <- predict(fit_X1X2.boot2, gr, type = "response")
pred.X1X2.boot2<- as.factor(ifelse(pred.X1X2.boot2<=0.5,0,1))
# linear model bootstrap 3 predictions
pred.X1X2.boot3 <- predict(fit_X1X2.boot3, gr, type = "response")
pred.X1X2.boot3<- as.factor(ifelse(pred.X1X2.boot3<=0.5,0,1))

# create plots
# linear model bootstrap 1
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = pred.X1X2.boot1 ))+ geom_point(
  data = bootstrap.data1,aes(col=Y, size=0.00015)) +  geom_point(data = bootstrap.data1, colour="yellow
```



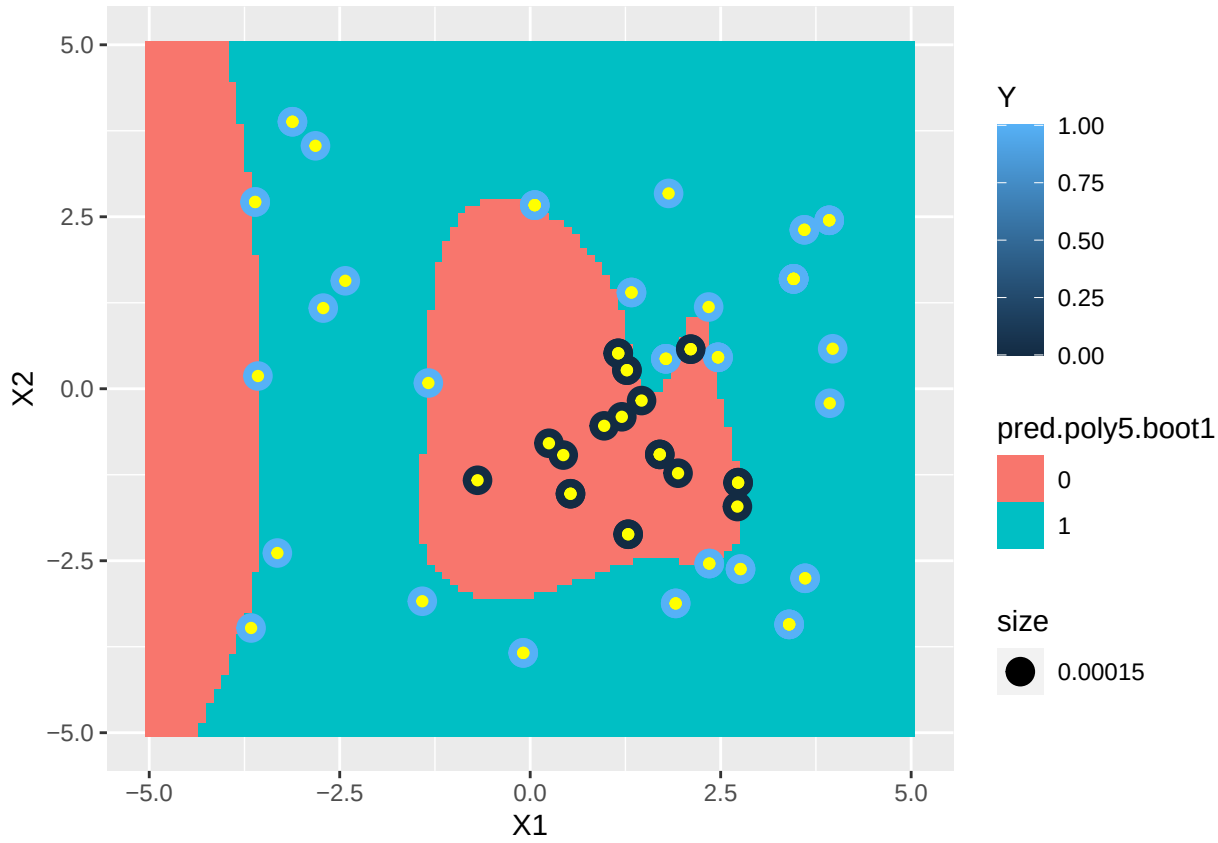

```
# linear model bootstrap 2
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = pred.X1X2.boot2 )) + geom_point(
  data = bootstrap.data2,aes(col=Y, size=0.00015)) +  geom_point(data = bootstrap.data2, colour="yellow
```



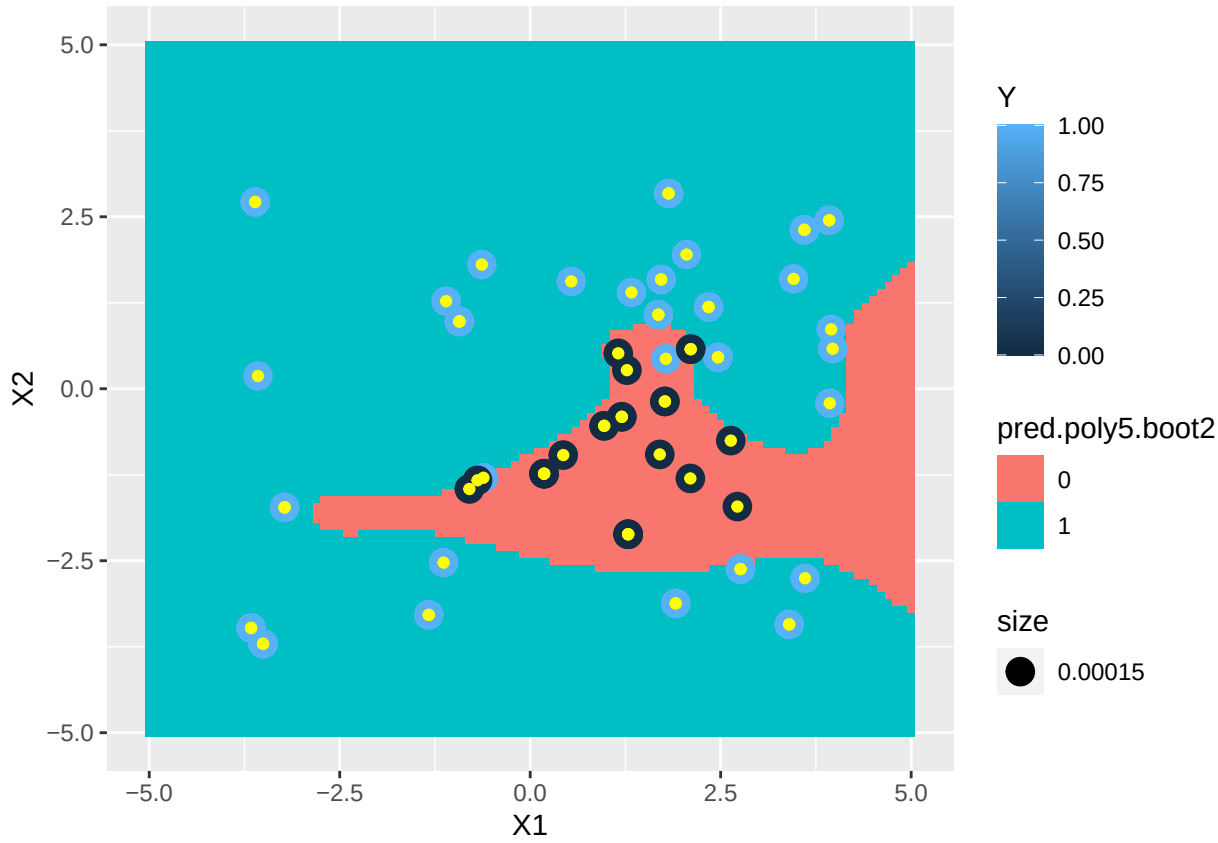

```
# linear model bootstrap 3
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = pred.X1X2.boot3 )) + geom_point(
  data = bootstrap.data3,aes(col=Y, size=0.00015)) +geom_point(data = bootstrap.data3, colour="yellow")
```



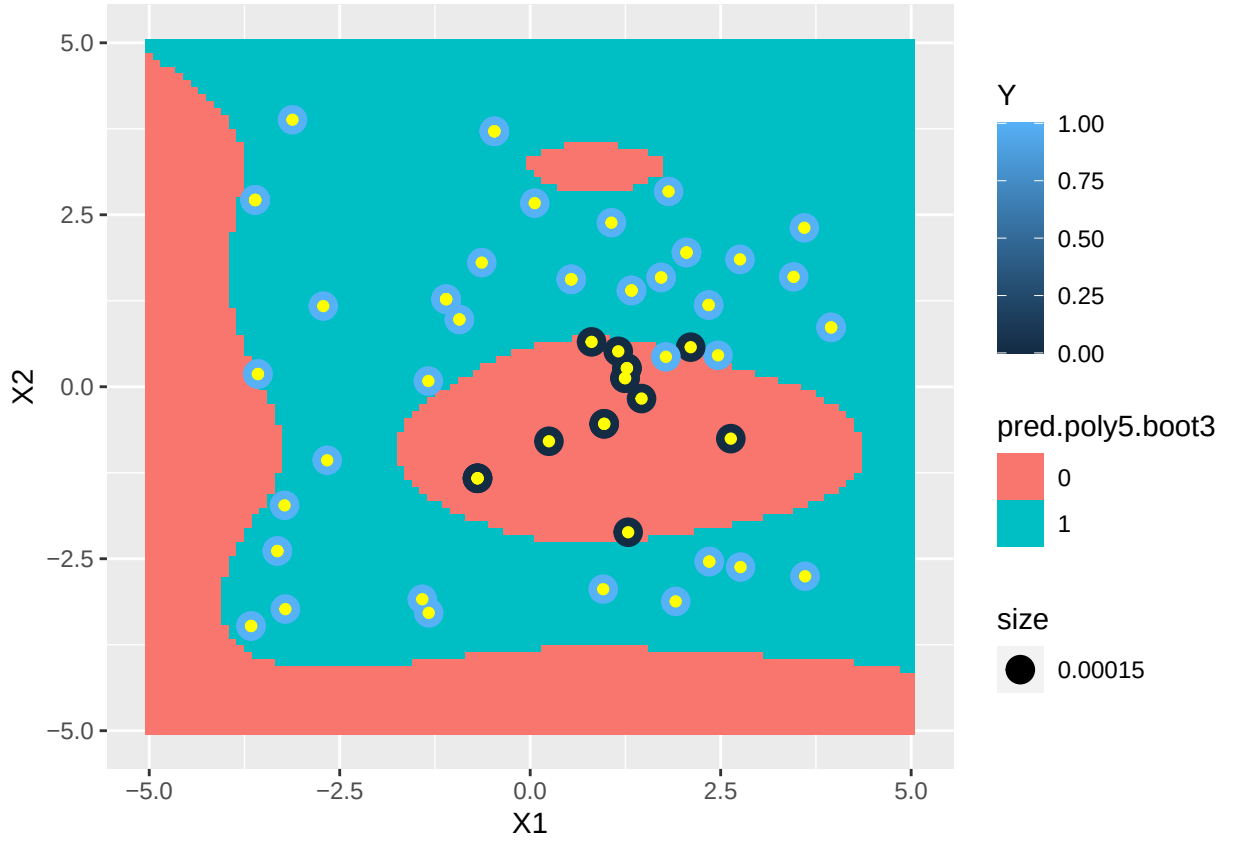

```
# predictions for 5th degree polynomial model
# 5 degree polynomial model predictions bootstrap 1
pred.poly5.boot1 <- predict(fit_5poly.boot1, gr, type = "response")
pred.poly5.boot1<- as.factor(ifelse(pred.poly5.boot1<=0.5,0,1))
# 5 degree polynomial model predictions bootstrap 2
pred.poly5.boot2 <- predict(fit_5poly.boot2, gr, type = "response")
pred.poly5.boot2<- as.factor(ifelse(pred.poly5.boot2<=0.5,0,1))
# 5 degree polynomial model predictions bootstrap 3
pred.poly5.boot3 <- predict(fit_5poly.boot3, gr, type = "response")
pred.poly5.boot3<- as.factor(ifelse(pred.poly5.boot3<=0.5,0,1))

# create plots
# polynomial model bootstrap 1
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = pred.poly5.boot1 )) + geom_point(
  data = bootstrap.data1,aes(col=Y, size=0.00015))+  geom_point(data = bootstrap.data1, colour="yellow"
```




```
# polynomial model bootstrap 2
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = pred.poly5.boot2 )) + geom_point(
  data = bootstrap.data2,aes(col=Y, size=0.00015))+  geom_point(data = bootstrap.data2, colour="yellow"
```




```
# polynomial model bootstrap 3
ggplot(gr, aes(X1, X2)) + geom_raster(aes(fill = pred.poly5.boot3 )) + geom_point(
  data = bootstrap.data3,aes(col=Y, size=0.00015))+geom_point(data = bootstrap.data3, colour="yellow")
```



From the summary tables, magnitude of coefficients for the polynomial models are much larger than the linear models. The model with the largest coefficient values (1.63e+15,-5.18e+15 and more) is the 5th degree polynomial model with bootstrap data 3 while the most inflexible model with the smallest coefficient values (0.89125,-0.00773, and 0.21023) is the linear model with bootstrap data 1. The graphs above show that the polynomial model is better at classifying the dataset in comparison to the linear model. The polynomial model is nonlinear as shown from the graph while the boundaries for linear model are a line. With the bootstrap replicates, we can see that the polynomial models overfit and classifies regions with no points which is a strange behavior. The linear model is much simpler, however, it was much worse at classifying the points for such as shown in graph 1 to 3. With the simpler model, the linear models have a higher bias and lower variance. The polynomial models on the other hand have higher variance and lower bias due to their flexibility.

4. Predicting insurance policy purchase

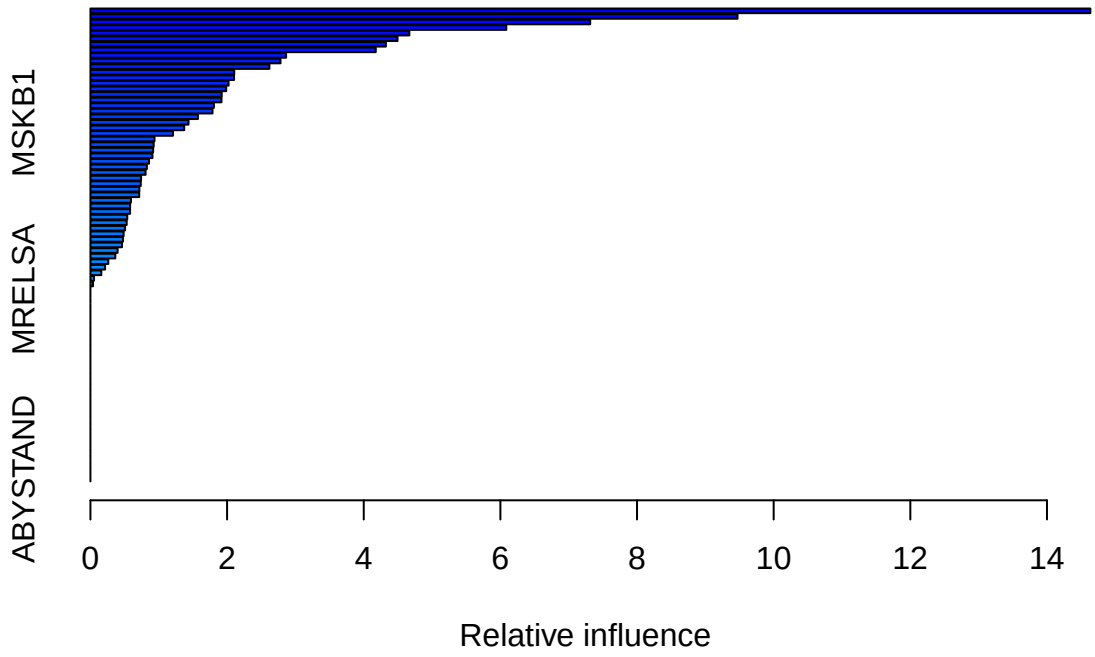
This question involves the use of the "Caravan" data set, which contains 5822 real customer records. Each record consists of 86 variables, containing sociodemographic data (variables 1-43) and product ownership (variables 44-86), grouped by zip code. In this problem we will focus on predicted the variable "Purchase" which indicates whether the customer purchased a caravan insurance policy. For more information see http://www.liacs.nl/~putten/library/ cc2000/data.html.

a) When you load the "ISLR" library, the variable Caravan is automatically loaded into your environment. Split Carvan into a training set consisting of the first 1000 observations and a test set consisting of the remaining observations.

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

# create a training set - first 1000 and test set  - remaining observations
train_caravan <- Caravan[1:1000,]
test_caravan <- Caravan[-(1:1000),]
```

b) Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use the gbm to fit a 1,000 tree boosted model and set the shrinkage value of 0.01. Which predictors appear to be the most important (Hint: use the summary function)?

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




```
##                var     rel.inf
## PPERSAUT PPERSAUT 14.63504779
## MKOOPKLA MKOOPKLA  9.47091649
## MOPLHOOG MOPLHOOG  7.31457416
## MBERMIDD MBERMIDD  6.08651965
## PBRAND     PBRAND  4.66766122
## MGODGE     MGODGE  4.49463264
## ABRAND     ABRAND  4.32427755
```

```
## MINK3045 MINK3045  4.17590619
## MOSTYPE   MOSTYPE  2.86402583
## PWAPART   PWAPART  2.78191075
## MAUT1       MAUT1  2.61929152
## MBERARBG MBERARBG  2.10480508
## MSKA         MSKA  2.10185152
## MAUT2       MAUT2  2.02172510
## MSKC         MSKC  1.98684345
## MINKGEM   MINKGEM  1.92122708
## MGODPR     MGODPR  1.91777542
## MBERHOOG MBERHOOG  1.80710618
## MGODOV     MGODOV  1.78693913
## PBYSTAND PBYSTAND  1.57279593
## MSKB1       MSKB1  1.43551401
## MFWEKIND MFWEKIND  1.37264255
## MRELGE     MRELGE  1.20805179
## MOPLMIDD MOPLMIDD  0.93791970
## MINK7512 MINK7512  0.92590720
## MINK4575 MINK4575  0.91745993
## MGODRK     MGODRK  0.90765539
## MFGEKIND MFGEKIND  0.85745374
## MZPART     MZPART  0.82531066
## MRELOV     MRELOV  0.80731252
## MINKM30   MINKM30  0.74126812
## MHKOOP     MHKOOP  0.73690793
## MZFONDS   MZFONDS  0.71638323
## MAUT0       MAUT0  0.71388052
## MHHUUR     MHHUUR  0.59287247
## APERSAUT APERSAUT  0.58056986
## MOSHOOFD MOSHOOFD  0.58029563
## MSKB2       MSKB2  0.53885275
## PLEVEN     PLEVEN  0.53052444
## MINK123M MINK123M  0.50660603
## MBERARBO MBERARBO  0.48596479
## MGEMOMV   MGEMOMV  0.47614792
## PMOTSCO   PMOTSCO  0.46163590
## MSKD         MSKD  0.39735297
## MBERBOER MBERBOER  0.36417546
## MGEMLEEF MGEMLEEF  0.26166240
## MFALLEEN MFALLEEN  0.21448118
## MBERZELF MBERZELF  0.15906143
## MOPLLAAG MOPLLAAG  0.05263665
## MAANTHUI MAANTHUI  0.03766014
## MRELSA     MRELSA  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
```

We can see that PPERSAUT, MKOOPKLA, MOPLHOOG, MBERMIDD,and PRBRAND are the most important predictors from the summary table.

c) Now fit a random forest model to the same training set from the previous problem. Set importance=TRUE but use the default parameter values for all other inputs to the randomForest function. Print the random forest object returned by the random forest function. What is the out-of-bag estimate of error? How many variables were subsampled at each split in the trees? How many trees were used to fit the data? Look at the variable importance. Is the order of important variables similar for both boosting and random forest models?

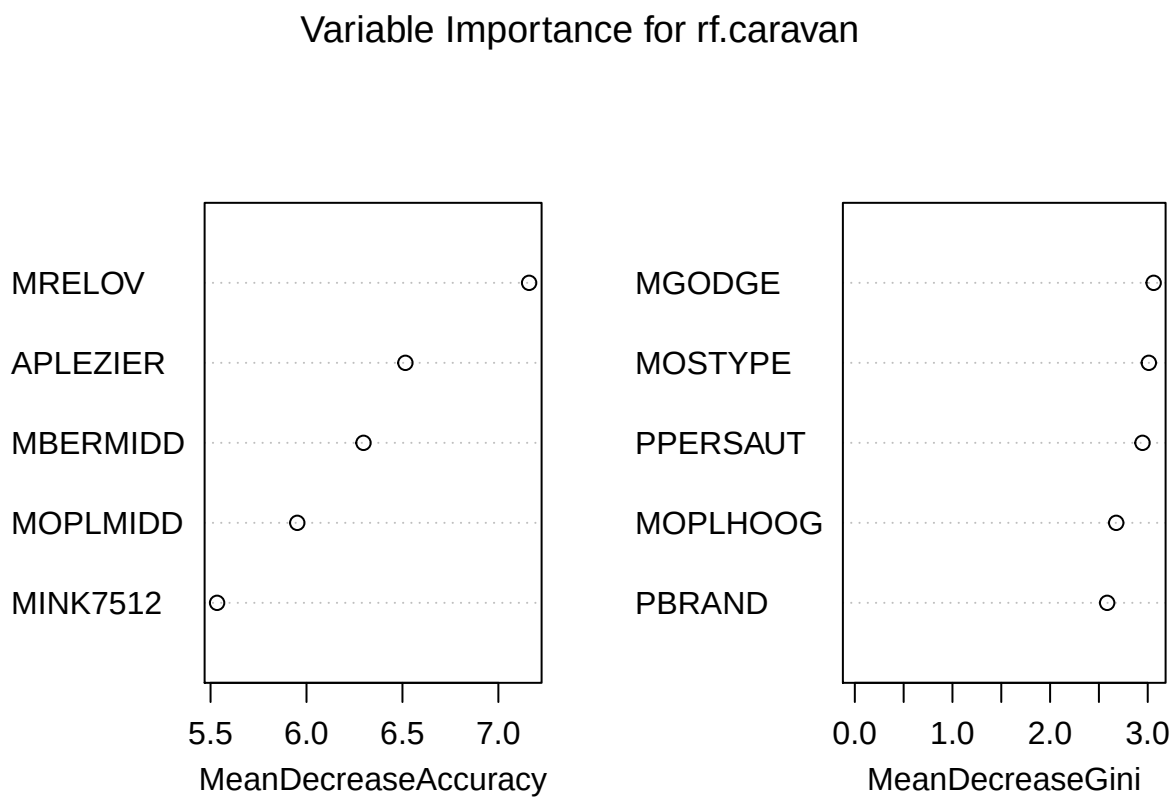
```
rf.caravan = randomForest(Purchase ~ ., data=train_caravan, importance=TRUE)
rf.caravan
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train_caravan, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.2%
## Confusion matrix:
##      No Yes class.error
## No  936   5 0.005313496
## Yes  57   2 0.966101695
```

```r
# finding important variables
head(importance(rf.caravan))
```

```
##                 No        Yes MeanDecreaseAccuracy MeanDecreaseGini
## MOSTYPE  4.3991288  1.3355031            4.6924200        3.0112384
## MAANTHUI 0.6328113 -0.5908229            0.4360337        0.5068821
## MGEMOMV  3.1039173 -1.0614165            2.8583008        1.0401876
## MGEMLEEF 4.6687172  0.1802101            4.7080121        0.9994155
## MOSHOOFD 3.0530073  4.6342592            4.0745604        1.9005616
## MGODRK   1.7768512  0.4954718            1.9216445        1.4122826
```

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



The out-of-bag estimate of error rate is 6.2%. The variables sampled at each split was 9 and a total of 500 trees were used to fit the data. The order of important variables were similar for boosting model ordered by gini index and random forest models and different for boosting model ordered by model accuracy. The top 5 important predictors in the random forest model by Gini Index are MOSTYPE, MGODGE, PPERSAUT, MOPLHOOG, and PBRAND and by Model Accuracy, the predictors are MRELOV, APLEZIER, MBERMIDD, MOPLMIDD, and MINK7512.

d) Use both models to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 %. Print the confusion matrix for both the boosting and random forest models. In the random forest model, what fraction of the people predicted to make a purchase do in fact make one? Note: use the predict function with type="prob" for random forests and type="resonpse" for the boosting algorithm.

```r
set.seed(1)
# predict using boosting model
yhat.boost <- predict(boost.caravan, newdata = test_caravan, type = "response")
```

```
## Using 1000 trees...
```

```r
# predict using random forest model
yhat.rf <- predict(rf.caravan, newdata = test_caravan, type = "prob")

# confusion matrix for boosting model
confmatrix.boost <- table(Truth=test_caravan$Purchase,Prediction=ifelse(yhat.boost>0.2,"Yes", "No"))
confmatrix.boost
```

```
##      Prediction
## Truth   No  Yes
##   No  4410  123
##   Yes  256   33
```

```r
# confusion matrix for random forest model
confmatrix.rf <- table(Truth=test_caravan$Purchase, Prediction=ifelse(yhat.rf[,2]>0.2,"Yes" ,"No"))
confmatrix.rf
```

```
##      Prediction
## Truth   No  Yes
##   No  4283  250
##   Yes  244   45
```

```r
# fraction of people predicted to make purchase and do infact make one (rf model)
sum(confmatrix.rf[2,2])/sum(confmatrix.rf[c(1:2),2])
```

```
## [1] 0.1525424
```

5. An SVMs prediction of drug use

In this problem we return to an analysis of the drug use dataset. Load the drug use data using read_csv:

```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.
```

```
head(drug_use)
```

```
## # A tibble: 6 x 32
##      ID     Age Gender Education Country Ethnicity Nscore Escore  Oscore Ascore
##   <dbl>   <dbl>  <dbl>     <dbl>   <dbl>     <dbl>  <dbl>  <dbl>   <dbl>  <dbl>
## 1     1  0.498   0.482   -0.0592   0.961     0.126  0.313 -0.575 -0.583  -0.917
## 2     2 -0.0785 -0.482    1.98     0.961    -0.317 -0.678  1.94   1.44    0.761
## 3     3  0.498  -0.482   -0.0592   0.961    -0.317 -0.467  0.805 -0.847  -1.62
## 4     4 -0.952   0.482    1.16     0.961    -0.317 -0.149 -0.806 -0.0193  0.590
## 5     5  0.498   0.482    1.98     0.961    -0.317  0.735 -1.63  -0.452  -0.302
## 6     6  2.59    0.482   -1.23     0.249    -0.317 -0.678 -0.300 -1.56    2.04
## # ... with 22 more variables: Cscore <dbl>, Impulsive <dbl>, SS <dbl>,
## #   Alcohol <chr>, Amphet <chr>, Amyl <chr>, Benzos <chr>, Caff <chr>,
## #   Cannabis <chr>, Choc <chr>, Coke <chr>, Crack <chr>, Ecstasy <chr>,
## #   Heroin <chr>, Ketamine <chr>, Legalh <chr>, LSD <chr>, Meth <chr>,
## #   Mushrooms <chr>, Nicotine <chr>, Semer <chr>, VSA <chr>
```

a) Split the data into training and test data. Use a random sample of 1500 observations for the training data and the rest as test data. Use a support vector machine to predict recent_cannabis_use using only the subset of predictors between Age and SS variables as on homework 3. Unlike homework 3, do not bother mutating the features into factors. Use a "radial" kernel and a cost of 1. Generate and print the confusion matrix of the predictions against the test data.

```
# define the column recent cannabis use
drug_use <- drug_use %>% mutate(recent_cannabis_use =
                                  factor(ifelse(Cannabis >= "CL3", "Yes", "No"),
                                         levels=c("No", "Yes")))

# subset of predictors between Age and SS variables
drug_use_subset <- drug_use %>% select(Age:SS,recent_cannabis_use)

# randomly sample to split data into training set (1500) and test set
set.seed(1)
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,]

# cost=1
svmfit=svm(recent_cannabis_use~.,data=drug_use.train, kernel="radial", cost=1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = recent_cannabis_use ~ ., data = drug_use.train, kernel = "radial",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  740
##
##  ( 383 357 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
# predict on test set
svmfit.test <- predict(svmfit, drug_use.test, type = "response")

#confusion matrix
confmatrix.svm <- table(Truth=drug_use.test$recent_cannabis_use,Prediction=svmfit.test)
confmatrix.svm
```

```
##      Prediction
## Truth  No Yes
##   No  134  31
##   Yes  44 176
```

b) Use the tune function to perform cross validation over the set of cost parameters: cost=c(0.001, 0.01, 0.1, 1,10,100). What is the optimal cost and corresponding cross validated training error for this model? Print the confusion matrix for the best model. The best model can be found in the best.model variable returned by tune.

```
# perform cross validation over a set of cost parameters cost=c(0.001, 0.01, 0.1,1,10,100)
tune.out=tune(svm,recent_cannabis_use~., data=drug_use.train,
              kernel="radial",ranges=list(cost=c(0.001, 0.01, 0.1,
1,10,100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
```

```
## - best performance: 0.188
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4806667 0.03477618
## 2 1e-02 0.2020000 0.02667592
## 3 1e-01 0.1880000 0.01932184
## 4 1e+00 0.1900000 0.02042753
## 5 1e+01 0.2106667 0.02017210
## 6 1e+02 0.2440000 0.02688797
```

```
# optimal cost
summary(tune.out)$"best.parameters" # 0.1 - optimal cost
```

```
##   cost
## 3  0.1
```

```
# the best model
summary(tune.out)$"best.model"
```

```
##
## 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:  887
```

```
# confusion matrix
conf.tune <- table(true=drug_use.test$recent_cannabis_use,
                   pred=predict(tune.out$best.model,newdata=drug_use.test))
conf.tune
```

```
##      pred
## true   No Yes
##   No  138  27
##   Yes  35 185
```

```
# training error
train.err <- 1-sum(diag(conf.tune))/sum(conf.tune)
train.err #0.176
```

```
## [1] 0.161039
```

```
#test or train error
```