PSTAT 131/231 HW #3

Lash Tan (231) and Jacobo Pereira-Pacheco (131)

06/03/2018

Question 1

Fundamentals of the Bootstrap

a)

For a bootstrap sample b of sample size n,

$$\mathbb{P}\{j \notin b\} = (\frac{n-1}{n}) \times (\frac{n-1}{n}) \times ... \times (\frac{n-1}{n})$$
$$= \prod_{k=0}^{n} (\frac{n-1}{n})$$
$$= (\frac{n-1}{n})^n$$

b)

For $n = 1000$,

$$\mathbb{P}\{j \notin b\} = (\frac{999}{1000})^{1000} \approx 0.36769542477$$

c)

```r
set.seed(1)
resample <- sample(1:1000, replace = T)
resample_unique <- length(unique(resample))
cat("Number of missing observations from resampling (out of 1000):",
    (1000-resample_unique))
```

```
## Number of missing observations from resampling (out of 1000): 361
```

d)

```r
set.seed(1)
shots <- c(rep(1, 50), rep(0, 51))
shots_bs_means <- c()
for(i in 1:1000){
  shots_resample <- sample(shots, length(shots), replace = T)
  shots_avg <- mean(shots_resample)
  shots_bs_means <- c(shots_bs_means, shots_avg)
}
shots_confint <- quantile(shots_bs_means, c(.025, .975))
cat("95% confidence interval for the true 3PT%:", shots_confint)
```

```
## 95% confidence interval for the true 3PT%: 0.4059 0.5941
```

Robert Covington's end-of-season 3PT% is most likely lower than what he had earlier in the season, and this is due to *regression toward the mean.* This phenomenon states that extreme values will tend to be followed by less extreme values. Essentially, Covington's 3PT% is expected to converge toward the league average as his number of shot attempts increases. As it turns out, his end-of-season average was within 1% of the league average (around 36%).

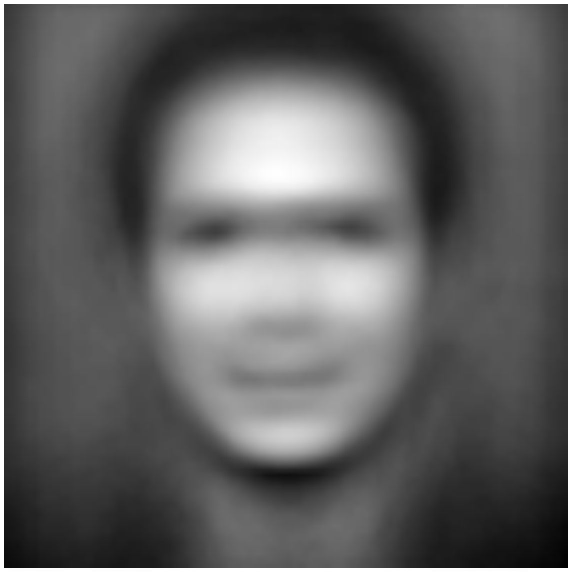
Question 2

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)
}
```

a)

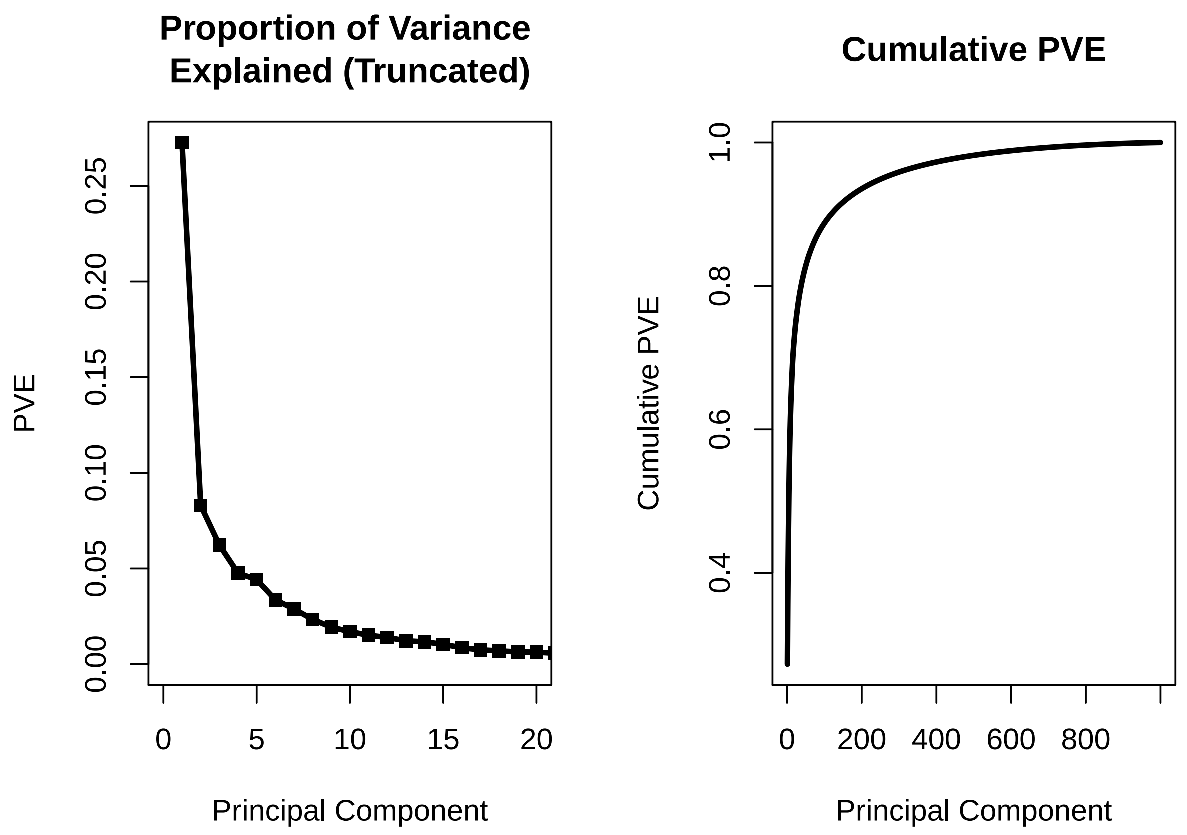
```r
avg_face <- colMeans(face_mat)
plot_face(avg_face)
```



b)

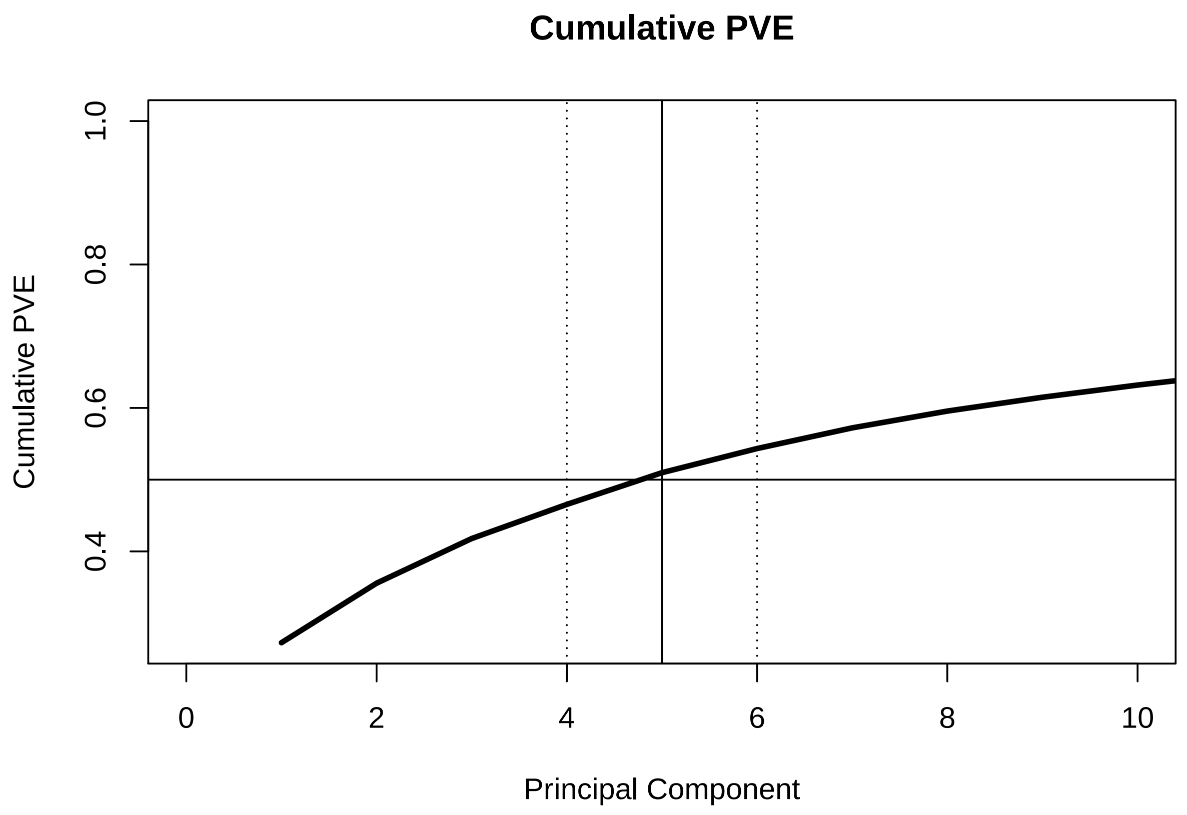
```
face_pr_out <- prcomp(face_mat, center = T, scale = F)
face_pr_var <- face_pr_out$sdev^2
face_pve <- face_pr_var/sum(face_pr_var)
face_cumulative_pve <- cumsum(face_pve)
```

```
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')
```



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

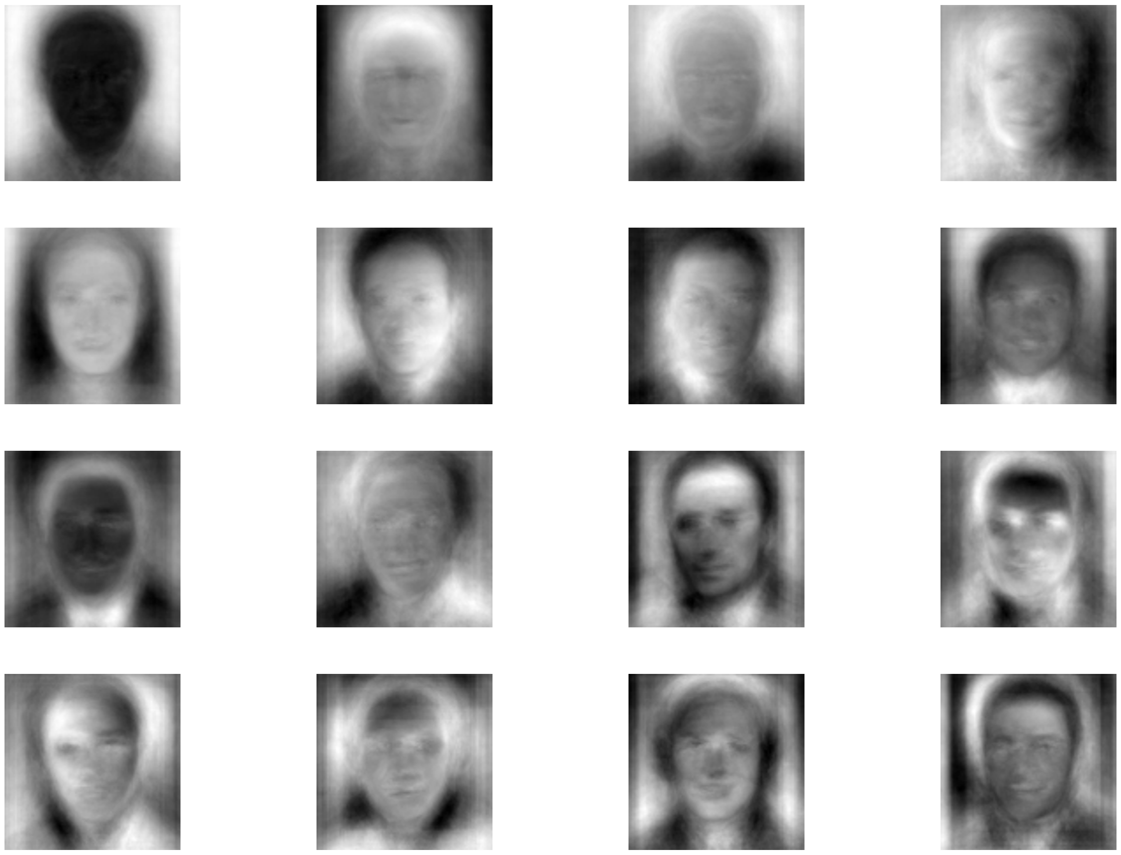
```
pc_50 <- which(face_cumulative_pve >= .5)[1]
plot(face_cumulative_pve, type="l", xlim = c(0,pc_50+5), lwd=3, xlab = 'Principal Component',
     ylab = 'Cumulative PVE', main = 'Cumulative PVE')
abline(h=.5, v=pc_50)
abline(v = c(pc_50 - 1, pc_50 + 1), lty = 3)
```



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.

c)

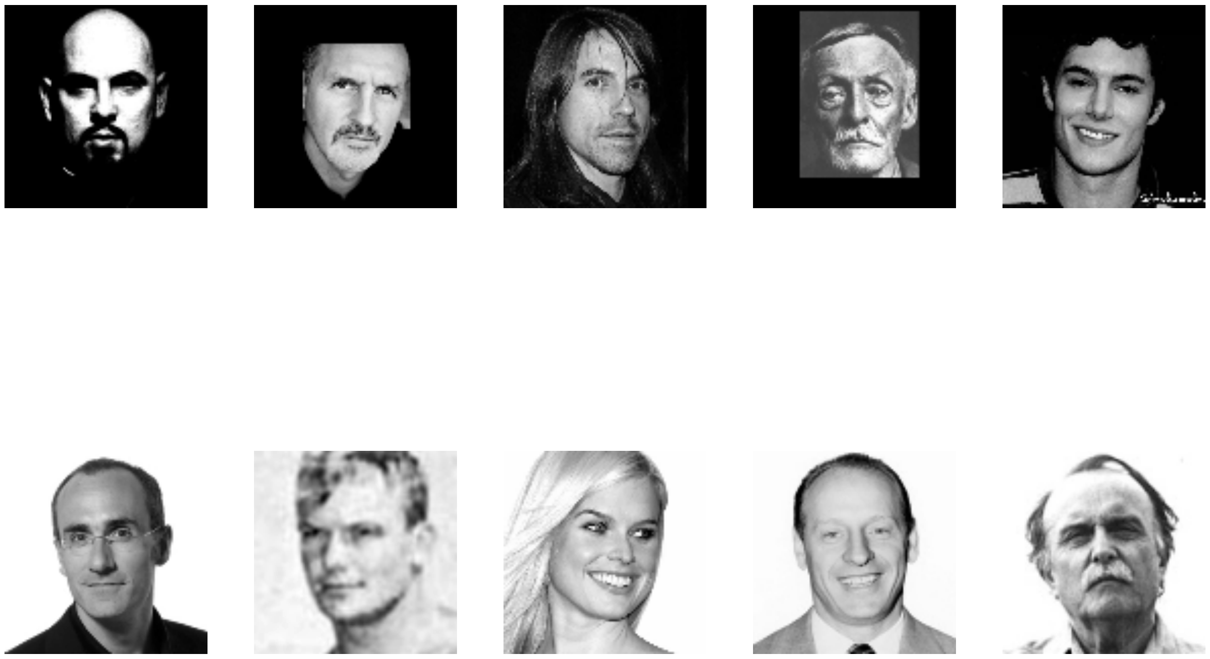
```
par(mfrow=c(4,4), mar=c(1,1,1,1))
for (i in 1:16){
  plot_face(face_pr_out$rotation[ ,i])
}
```



We can see that there are significantly higher amounts of lighter regions opposed to darker regions, albeit both light and dark regions showcase regions of high contrast. The contrast decreases through the 16 principal components and faces become more noticeable.

d)

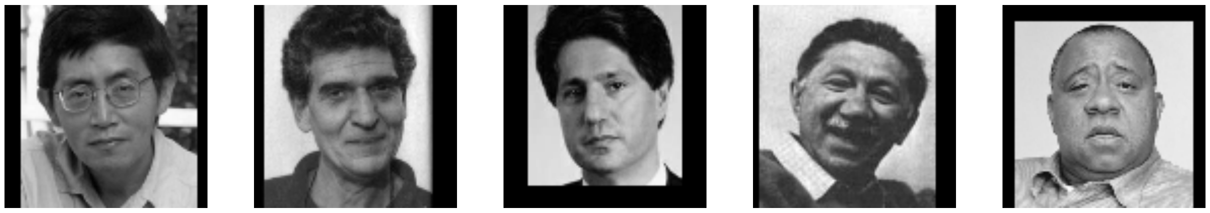
```
min_pc1 <- head(order(face_pr_out$x[ ,1]), n = 5)
max_pc1 <- tail(order(face_pr_out$x[ ,1]), n = 5)

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

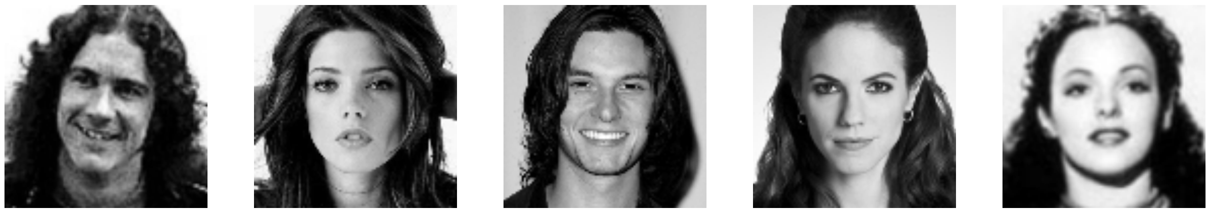


Note that the top row goes from the lowest value to the fifth lowest value from left to right while the bottom row goes from the fifth highest value to the highest value from left to right. The most obvious variation between the top row and the bottom row of the plot above is the contrast of the background 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.

e)

```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, ])
```

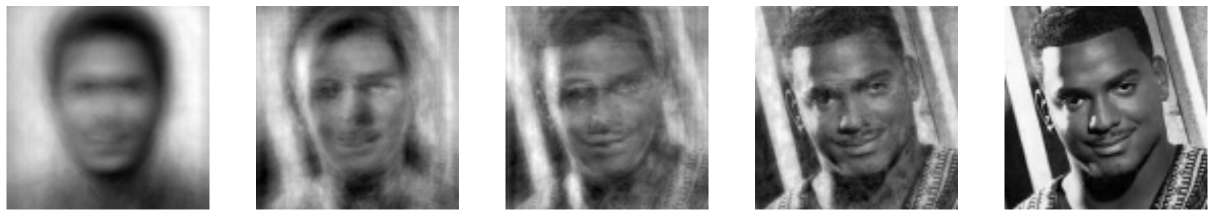




Here we can see that the bottom row is marked by beautiful people while the top row is filled by people who are even more beautiful. Unfortunately, this isn't something that PCA can detect. While there can be multiple interpretations of variation between these two rows, the the feature that stands out most to us is that the bottom row has longer, dark hair that wraps around their relatively lighter faces, whereas the top row has slightly darker faces with shorter hair. An interesting addition would be that the top row has a small black border around each image, which may contribute as well. PC5 more relates to distinctions of physical characteristics of the face and therefore has more importance in *facial* recognition than PC1.

f)

```r
par(mfrow=c(1,5), mar=c(1,1,1,1))
k <- c(10,50,100,300)

## 4 compressed face images
for(i in k){
  k_face <- ((face_pr_out$x[ ,1:i]) %*% (t(face_pr_out$rotation)[1:i, ]))[281, ] + avg_face
  plot_face(k_face)
}

## original image
plot_face(face_mat[281, ])
```



The plot above shows the four compressed images followed by the original image.

Question 3

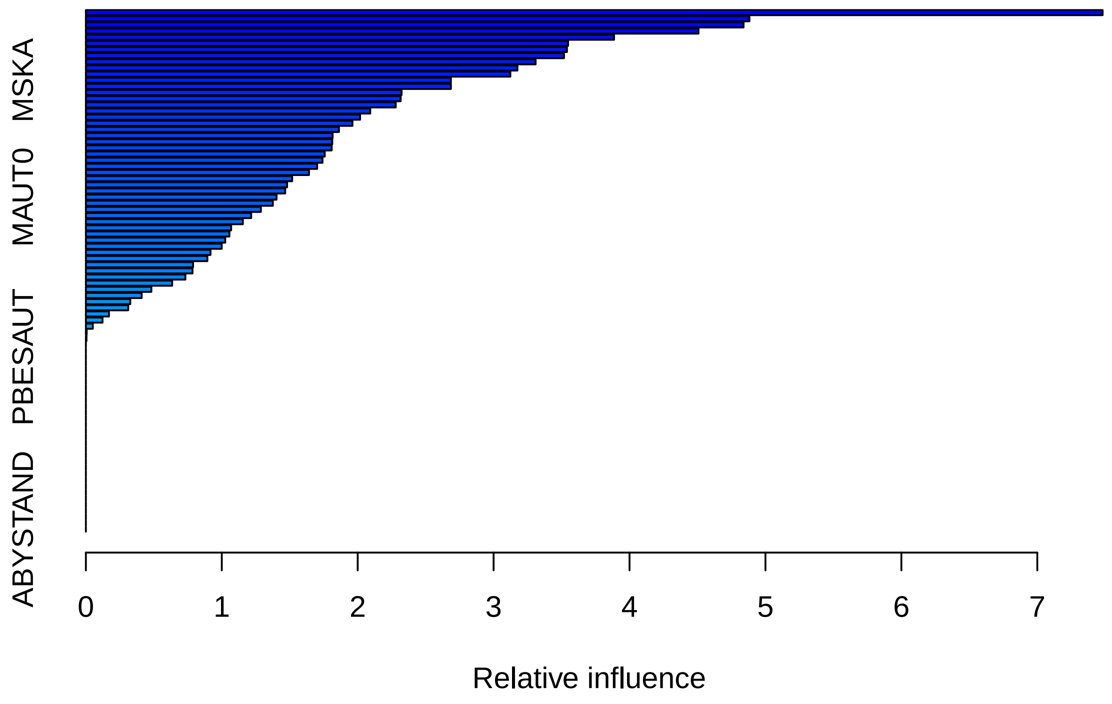
Predicting insurance policy purchases

a)

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

b)

```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)
summary(caravan_boost)
```



```
##                var  rel.inf
## PPERSAUT PPERSAUT 7.480819
## MOPLHOOG MOPLHOOG 4.882054
## MGODGE     MGODGE 4.838870
## MKOOPKLA MKOOPKLA 4.507280
## MOSTYPE   MOSTYPE 3.886044
## MGODPR     MGODPR 3.547892
## PBRAND     PBRAND 3.539488
## MBERMIDD MBERMIDD 3.518083
## MBERARBG MBERARBG 3.309005
## MINK3045 MINK3045 3.175314
## MSKC         MSKC 3.123008
## MSKA         MSKA 2.685845
## MAUT2       MAUT2 2.685548
## MAUT1       MAUT1 2.322786
## PWAPART   PWAPART 2.316252
## MSKB1       MSKB1 2.279820
## MRELOV     MRELOV 2.092410
## MFWEKIND MFWEKIND 2.017651
## MBERHOOG MBERHOOG 1.961379
## MBERARBO MBERARBO 1.862074
## MRELGE     MRELGE 1.815276
## MINK7512 MINK7512 1.812894
## MINKM30   MINKM30 1.808781
## MOPLMIDD MOPLMIDD 1.757785
## MFGEKIND MFGEKIND 1.741173
## MGODOV     MGODOV 1.701539
## MZFONDS   MZFONDS 1.641659
## MFALLEEN MFALLEEN 1.517764
## MSKB2       MSKB2 1.480398
## MINK4575 MINK4575 1.466411
## MAUT0       MAUT0 1.403097
## ABRAND     ABRAND 1.375697
## MHHUUR     MHHUUR 1.287673
## MINKGEM   MINKGEM 1.216352
## MHKOOP     MHKOOP 1.154971
## MGEMLEEF MGEMLEEF 1.068800
## MGODRK     MGODRK 1.056067
## MRELSA     MRELSA 1.025383
## MZPART     MZPART 0.999706
## MSKD         MSKD 0.917078
## MGEMOMV   MGEMOMV 0.893758
## MBERZELF MBERZELF 0.788935
## APERSAUT APERSAUT 0.784653
## MOPLLAAG MOPLLAAG 0.732211
## MOSHOOFD MOSHOOFD 0.634998
## PMOTSCO   PMOTSCO 0.481824
## PLEVEN     PLEVEN 0.410808
## PBYSTAND PBYSTAND 0.326852
## MBERBOER MBERBOER 0.311572
## MINK123M MINK123M 0.169710
## MAANTHUI MAANTHUI 0.122660
## ALEVEN     ALEVEN 0.051158
## PAANHANG PAANHANG 0.006040
## PFIETS     PFIETS 0.004694
## PWABEDR   PWABEDR 0.000000
## PWALAND   PWALAND 0.000000
## PBESAUT   PBESAUT 0.000000
## PVRAAUT   PVRAAUT 0.000000
## PTRACTOR PTRACTOR 0.000000
## PWERKT     PWERKT 0.000000
## PBROM       PBROM 0.000000
## PPERSONG PPERSONG 0.000000
## PGEZONG   PGEZONG 0.000000
## PWAOREG   PWAOREG 0.000000
## PZEILPL   PZEILPL 0.000000
## PPLEZIER PPLEZIER 0.000000
## PINBOED   PINBOED 0.000000
## AWAPART   AWAPART 0.000000
## AWABEDR   AWABEDR 0.000000
## AWALAND   AWALAND 0.000000
## ABESAUT   ABESAUT 0.000000
## AMOTSCO   AMOTSCO 0.000000
## AVRAAUT   AVRAAUT 0.000000
## AAANHANG AAANHANG 0.000000
## ATRACTOR ATRACTOR 0.000000
## AWERKT     AWERKT 0.000000
## ABROM       ABROM 0.000000
## APERSONG APERSONG 0.000000
## AGEZONG   AGEZONG 0.000000
## AWAOREG   AWAOREG 0.000000
## AZEILPL   AZEILPL 0.000000
## APLEZIER APLEZIER 0.000000
## AFIETS     AFIETS 0.000000
## AINBOED   AINBOED 0.000000
## ABYSTAND ABYSTAND 0.000000
```

The PPERSAUT, MKOOPKLA, and MOPLHOOG appear to be the most important preditors in this data set. The plot from the `summary` call does not display enough labels and is therefore not particularly useful in interpreting the most important predictors.

c)

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

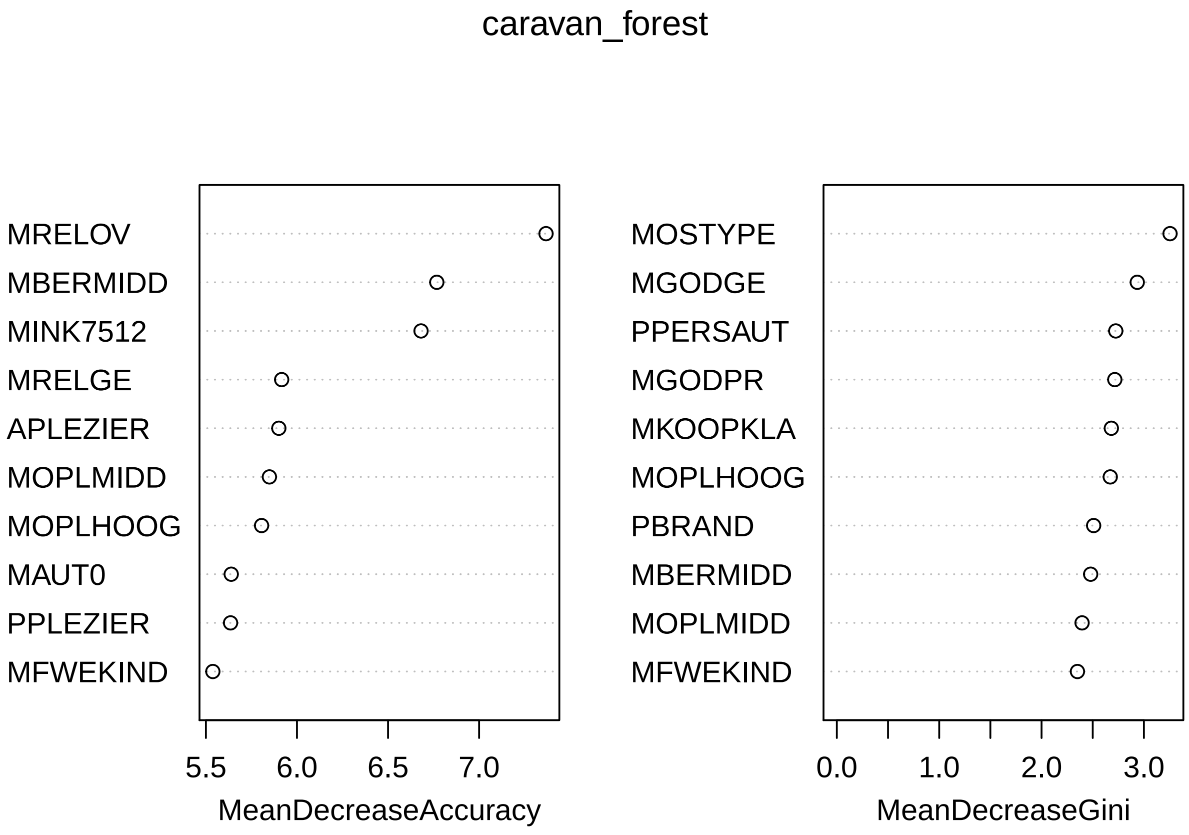
```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = caravan_train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.3%
## Confusion matrix:
##      No Yes class.error
## No  935   6    0.006376
## Yes  57   2    0.966102
```

The OOB estimate of error rate is 6.3% with 9 variables subsampled at each split. The default number of trees selected was 500.

```
importance(caravan_forest)
```

```
##                 No      Yes MeanDecreaseAccuracy MeanDecreaseGini
## MOSTYPE   4.043383  3.24970              5.04221         3.255245
## MAANTHUI  1.876463  0.13093              1.79001         0.576859
## MGEMOMV   3.177171 -1.88863              2.75282         1.027170
## MGEMLEEF  3.976232  0.48653              4.05013         1.146330
## MOSHOOFD  2.197056  3.06240              3.04611         2.081519
## MGODRK    2.284745  0.51375              2.41551         1.203095
## MGODPR    4.713712  0.60129              4.79497         2.715466
## MGODOV    3.677310  0.95518              3.88995         1.396801
## MGODGE    3.082207  4.85747              4.49752         2.934914
## MRELGE    6.081500 -0.22864              5.91600         2.134516
## MRELSA    2.389497  2.06001              3.07267         1.154983
## MRELOV    7.580048 -0.96778              7.36744         1.829240
## MFALLEEN  5.767452 -1.27965              5.37547         1.512680
## MFGEKIND  3.734183  0.15553              3.74371         2.046707
## MFWEKIND  5.387744  0.41168              5.53803         2.350489
## MOPLHOOG  4.131317  6.39460              5.80574         2.671215
## MOPLMIDD  5.649369  0.41420              5.84869         2.395750
## MOPLLAAG  3.159975  0.49089              3.35842         1.886745
## MBERHOOG  1.867565  0.72397              2.13966         1.865654
## MBERZELF  0.878758  0.19990              0.83285         0.796573
## MBERBOER  0.220905  2.28798              0.76335         0.395807
## MBERMIDD  5.970627  3.99496              6.76773         2.479754
## MBERARBG  3.061477  2.33322              3.52131         2.159380
## MBERARBO  4.172088  1.86403              4.49688         1.890482
## MSKA      3.123273  2.44180              3.81334         1.993318
## MSKB1     1.812501  1.94275              2.29306         2.079278
## MSKB2     2.414631 -1.74065              1.91411         1.822485
## MSKC      3.986717  2.36321              4.43900         2.241639
## MSKD      3.081803  1.06357              3.20384         0.982757
## MHHUUR    1.011521  3.62119              2.09854         2.040758
## MHKOOP    1.442019  4.96761              3.12319         2.075577
## MAUT1     0.380937 -1.88047             -0.07395         1.796856
## MAUT2     2.837544  2.80096              3.57332         1.756160
## MAUT0     5.944609 -1.70595              5.63904         1.643492
## MZFONDS   4.805464 -0.72449              4.67261         1.856166
## MZPART    4.131418 -1.22092              4.03015         2.099766
## MINKM30   2.372481  1.58908              2.65986         2.005408
## MINK3045  1.670683 -0.56080              1.53184         2.234686
## MINK4575  0.359900  2.67551              1.17139         1.827598
## MINK7512  6.159991  1.72080              6.68118         1.662366
## MINK123M  0.534217  0.43854              0.69572         0.466279
## MINKGEM   4.720976  1.90984              5.25437         1.355576
## MKOOPKLA  2.662370  5.75286              4.21124         2.681638
## PWAPART  -2.274070  4.79082             -0.41352         1.883658
## PWABEDR   0.831764 -1.00100              0.68124         0.194242
## PWALAND  -0.667704 -1.00100             -0.86429         0.084260
## PPERSAUT  2.551685  4.08249              3.70357         2.724887
## PBESAUT   0.000000  0.00000              0.00000         0.007933
## PMOTSCO  -2.144057  0.58728             -1.92033         0.855990
## PVRAAUT   0.000000  0.00000              0.00000         0.000000
## PAANHANG -1.076968 -1.99334             -1.51223         0.245065
## PTRACTOR  1.121981 -1.41705              0.66776         0.273213
## PWERKT    0.000000  0.00000              0.00000         0.000000
## PBROM     4.767655 -1.38424              4.38234         0.448979
## PLEVEN   -0.003854 -1.22642             -0.36949         0.661391
## PPERSONG  1.001002  0.00000              1.00100         0.015746
## PGEZONG  -2.094201 -1.40275             -2.28071         0.708724
## PWAOREG   3.839223  2.34922              3.95818         0.890749
## PBRAND   -2.142055  3.14813             -0.87263         2.508951
## PZEILPL   0.000000  0.00000              0.00000         0.283419
## PPLEZIER  2.807874  6.90049              5.63536         2.095369
## PFIETS   -1.445154 -1.00100             -1.51789         0.178616
## PINBOED   1.600529  0.00000              1.61196         0.091472
## PBYSTAND  1.815391  0.30398              1.67294         0.757818
## AWAPART  -2.313577  4.75200             -0.75904         1.349995
## AWABEDR  -0.878316 -1.00100             -1.06932         0.086424
## AWALAND  -1.163959 -1.00100             -1.17436         0.097075
## APERSAUT  1.053795  0.94001              1.22402         2.049289
## ABESAUT   0.000000  0.00000              0.00000         0.005000
## AMOTSCO   0.099507 -2.16502             -0.52447         0.924992
## AVRAAUT   0.000000  0.00000              0.00000         0.000000
## AAANHANG  0.135346 -0.06143              0.15389         0.154633
## ATRACTOR  2.463825 -1.00100              1.90388         0.039102
## AWERKT    0.000000  0.00000              0.00000         0.000000
## ABROM     3.616923 -2.19023              3.04715         0.467420
## ALEVEN    0.178038 -1.00100              0.04560         0.296917
## APERSONG  0.000000  0.00000              0.00000         0.002867
## AGEZONG   0.159732 -1.40907             -0.09563         0.431334
## AWAOREG   3.257284  3.02048              3.71672         0.827628
## ABRAND   -1.470959  2.78951             -0.60089         1.828329
## AZEILPL   0.000000  0.00000              0.00000         0.308437
## APLEZIER  2.649997  7.45749              5.90039         1.326720
## AFIETS   -1.517904  0.00000             -1.51441         0.248922
## AINBOED   0.392180 -1.41170             -0.32999         0.092794
## ABYSTAND  0.840457  0.82691              1.02024         0.471116
```

```r
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 Gini, MOSTYPE, MGODGE, and PPERSAUT were determined to be the most important.

d)

```r
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))
```

```
##              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  4274  243
##            Yes  259   46
```

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

```
## [1] 0.1508
```

For the random forest model, the fraction of people predicted to make a purchase that do in fact make one is $\frac{(true\ positives)}{(total\ predicted\ yes)} = \frac{44}{255}$, or 0.1508197.

Question 4

An SVMs prediction of drug use

```
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'))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ID = col_integer(),
##   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()
## )
```

```
## See spec(...) for full column specifications.
```

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

a)

```
set.seed(1)
drug_train_indicies <- sample(nrow(drug_use), 1500)
drug_train <- drug_use[drug_train_indicies, ]
drug_test <- drug_use[-drug_train_indicies, ]
```

```
drug_svm <- svm(recent_cannabis_use~., data = drug_train, kernel = "radial", cost = 1)
drug_svm_predict <- predict(drug_svm, newdata = drug_test)

table(Prediction = drug_svm_predict, Truth = drug_test$recent_cannabis_use)
```

```
##           Truth
## Prediction  No Yes
##        No  145  30
##        Yes  43 167
```

b)

```
set.seed(1)
drug_svm_tune <- tune(svm, recent_cannabis_use~., data = drug_train, kernel = "radial",
                      ranges=list(cost=c(0.001, 0.01, 0.1, 1, 10, 100)))
summary(drug_svm_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1793
##
## - Detailed performance results:
##    cost  error dispersion
## 1 1e-03 0.4653    0.04733
## 2 1e-02 0.2213    0.03011
## 3 1e-01 0.1793    0.02542
## 4 1e+00 0.1847    0.02611
## 5 1e+01 0.2080    0.01772
## 6 1e+02 0.2427    0.01546
```

We see that `cost=0.1` results in the lowest cross-validation error rate of 0.1793333.

```
drug_best_model <- drug_svm_tune$best.model
drug_best_predict <- predict(drug_best_model, drug_test)
table(Prediction = drug_best_predict, Truth = drug_test$recent_cannabis_use)
```

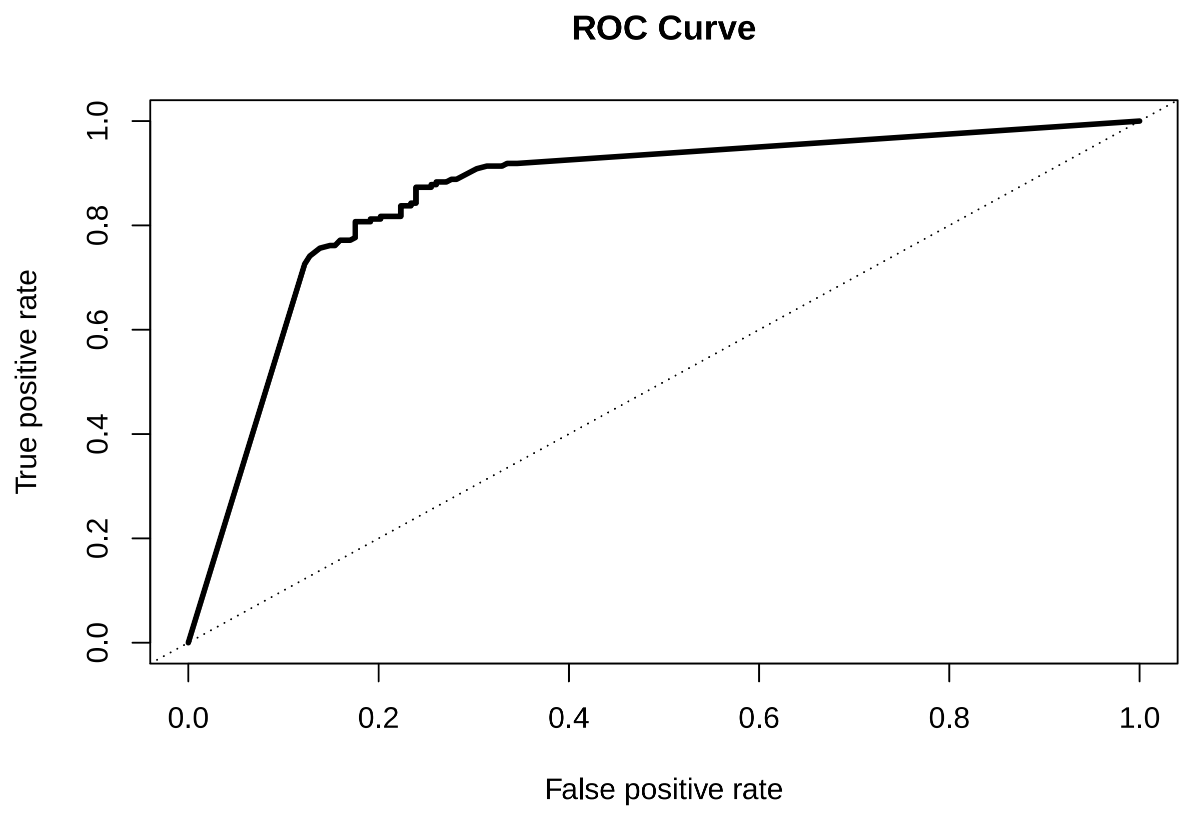
```
##           Truth
## Prediction  No Yes
##        No  148  35
##        Yes  40 162
```

c)

```
set.seed(1)
drug_responses <- rep(0,nrow(drug_test))

for(i in 1:200){
  temp_drug <- drug_train[sample(nrow(drug_train), replace=T), ]
  temp_svm <- svm(recent_cannabis_use~., data = temp_drug, kernel = "radial", cost = .1)
  temp_predict <- ifelse(predict(temp_svm, newdata = drug_test) == 'Yes', 1, 0)
  drug_responses <- drug_responses + temp_predict
}

drug_boot_probs <- drug_responses / 200
drug_boot_yhat <- factor(ifelse(drug_boot_probs >= .5, 'Yes', 'No'))
table(Prediction = drug_boot_yhat, Truth = drug_test$recent_cannabis_use)
```

```
##           Truth
## Prediction  No Yes
##        No  146  34
##        Yes  42 163
```

```
drug_boot_prediction <- prediction(drug_boot_probs, drug_test$recent_cannabis_use)
drug_boot_perf <- performance(drug_boot_prediction, measure = "tpr", x.measure = "fpr")
plot(drug_boot_perf, lwd=3, main="ROC Curve")
abline(a=0, b=1, lty=3)
```



Question 5

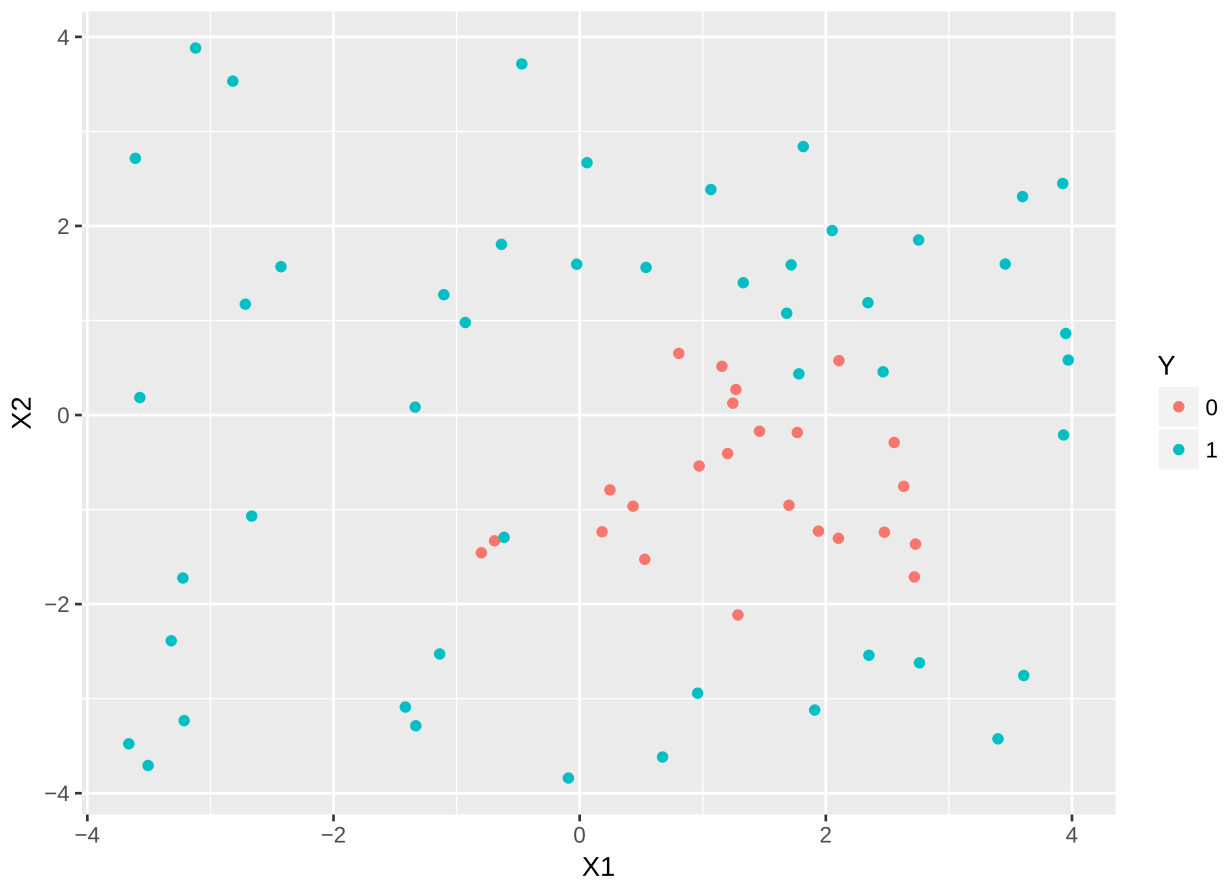
Logistic regression with polynomial features

a)

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

```
## Parsed with column specification:
## cols(
##   Z = col_integer(),
##   X1 = col_double(),
##   X2 = col_double(),
##   Y = col_integer()
## )
```

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



b)

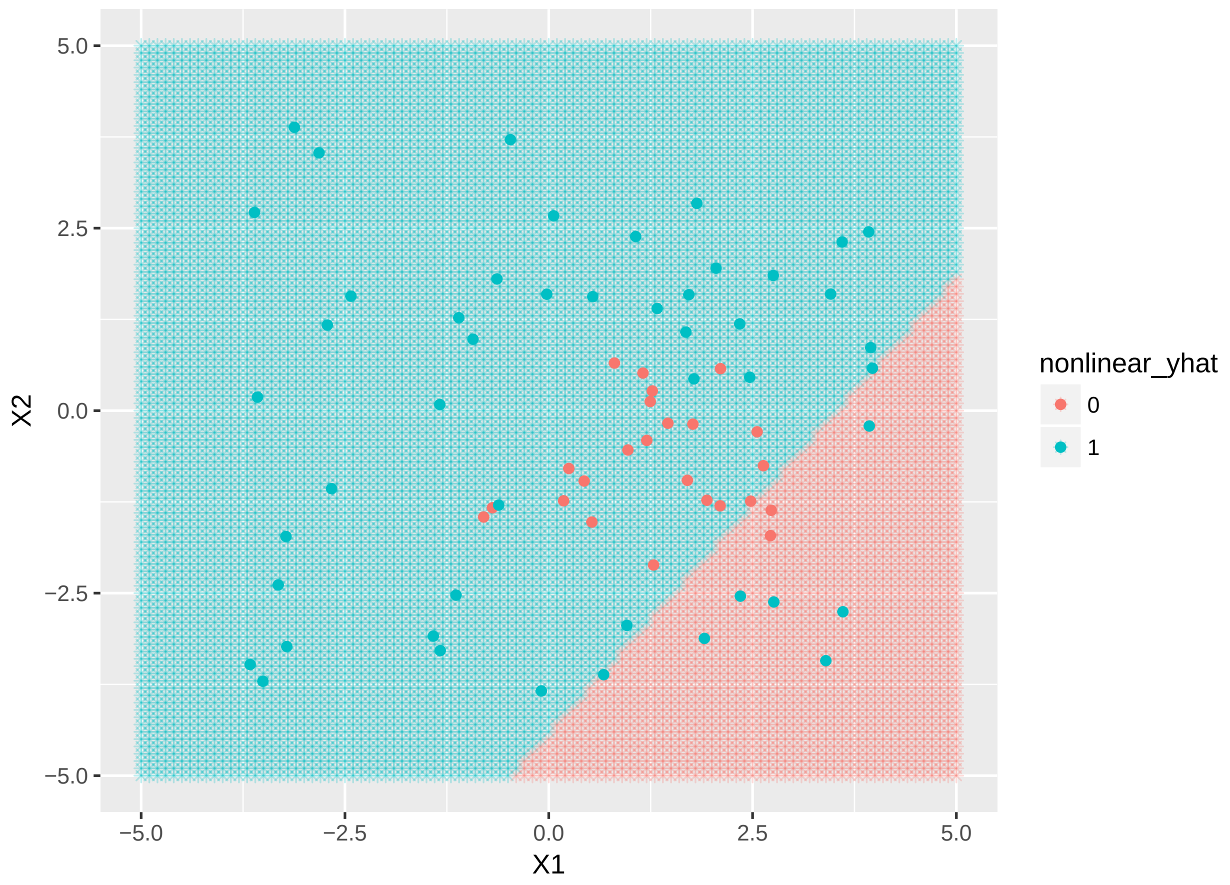
```
summary(nonlinear_fit <- glm(Y ~ X1 + X2, data = nonlinear_data, family="binomial"))
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = "binomial", data = nonlinear_data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.594  -1.248   0.626   0.915   1.511
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.022      0.314    3.26   0.0011 **
## X1            -0.289      0.136   -2.13   0.0334 *
## X2             0.232      0.144    1.62   0.1056
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 84.523  on 69  degrees of freedom
## AIC: 90.52
```

```
##
## Number of Fisher Scoring iterations: 4
```

```
# 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(mapping = aes(x=X1,y=X2)) +
  geom_point(data = gr, shape = 8, alpha = .25, aes(col = nonlinear_yhat)) +
  geom_point(data = nonlinear_data, aes(col=Y))
```

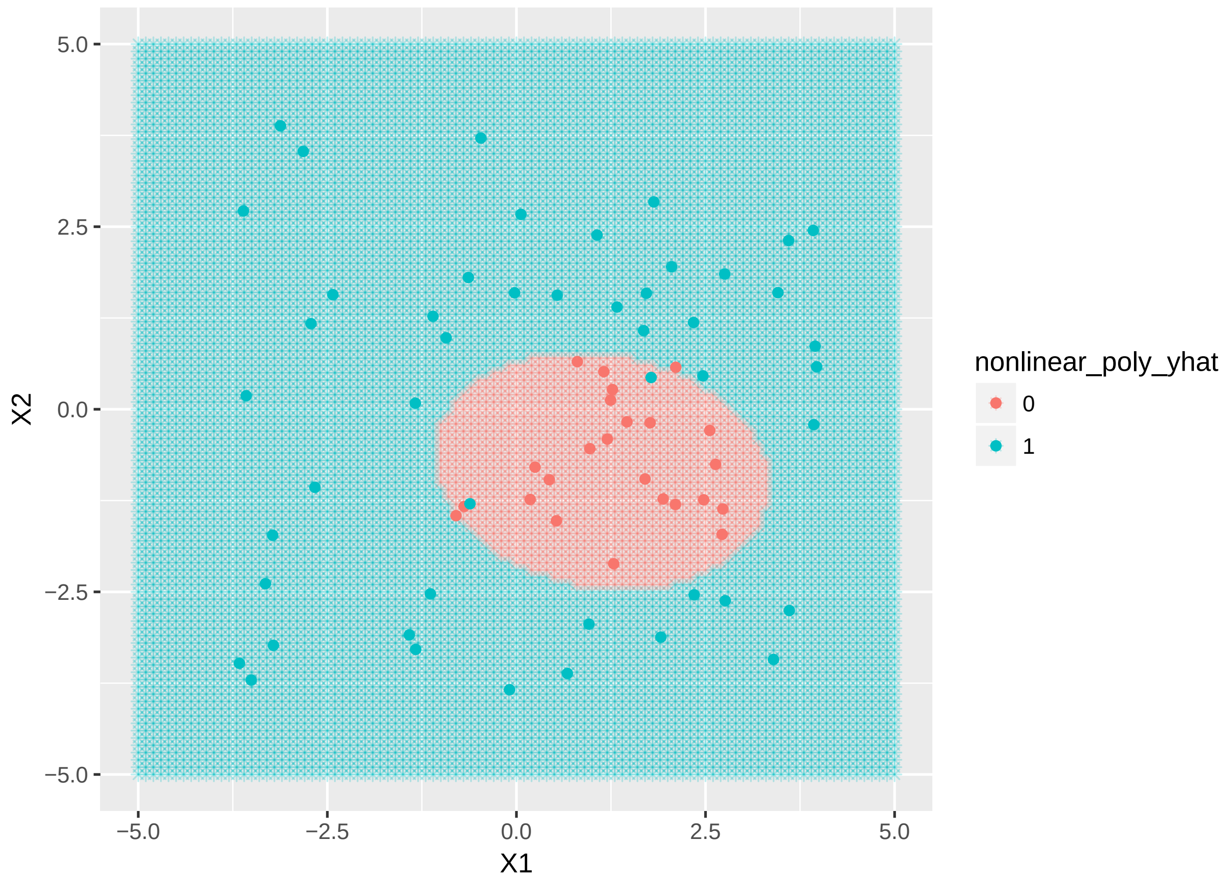


c)

```
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"))
```

```
##
## 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
## -1.3908  -0.0827   0.0000   0.0093   1.9007
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        11.800      4.809    2.45    0.014 *
## poly(X1, degree = 2, raw = F)1   -47.270     28.205   -1.68    0.094 .
## poly(X1, degree = 2, raw = F)2    57.777     29.043    1.99    0.047 *
## poly(X2, degree = 2, raw = F)1    45.071     26.911    1.67    0.094 .
## poly(X2, degree = 2, raw = F)2    96.311     39.733    2.42    0.015 *
## X1:X2                              0.501      0.737    0.68    0.496
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 13.852  on 66  degrees of freedom
## AIC: 25.85
##
## Number of Fisher Scoring iterations: 10
```

```
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 = .25, aes(col = nonlinear_poly_yhat)) +
  geom_point(data = nonlinear_data, aes(col = Y))
```



Because this model is fitting a 2nd degree polynomial with interaction terms, the decision boundary accurately captures where the red points lie by forming a oval-shaped region enclosing these points. This showcases that the 2nd degree polynomial logistic regression model is able to accurately predict the decision boundary, with the exception of two misclassfied points that are either overlapping or very adjacent to the blue points.

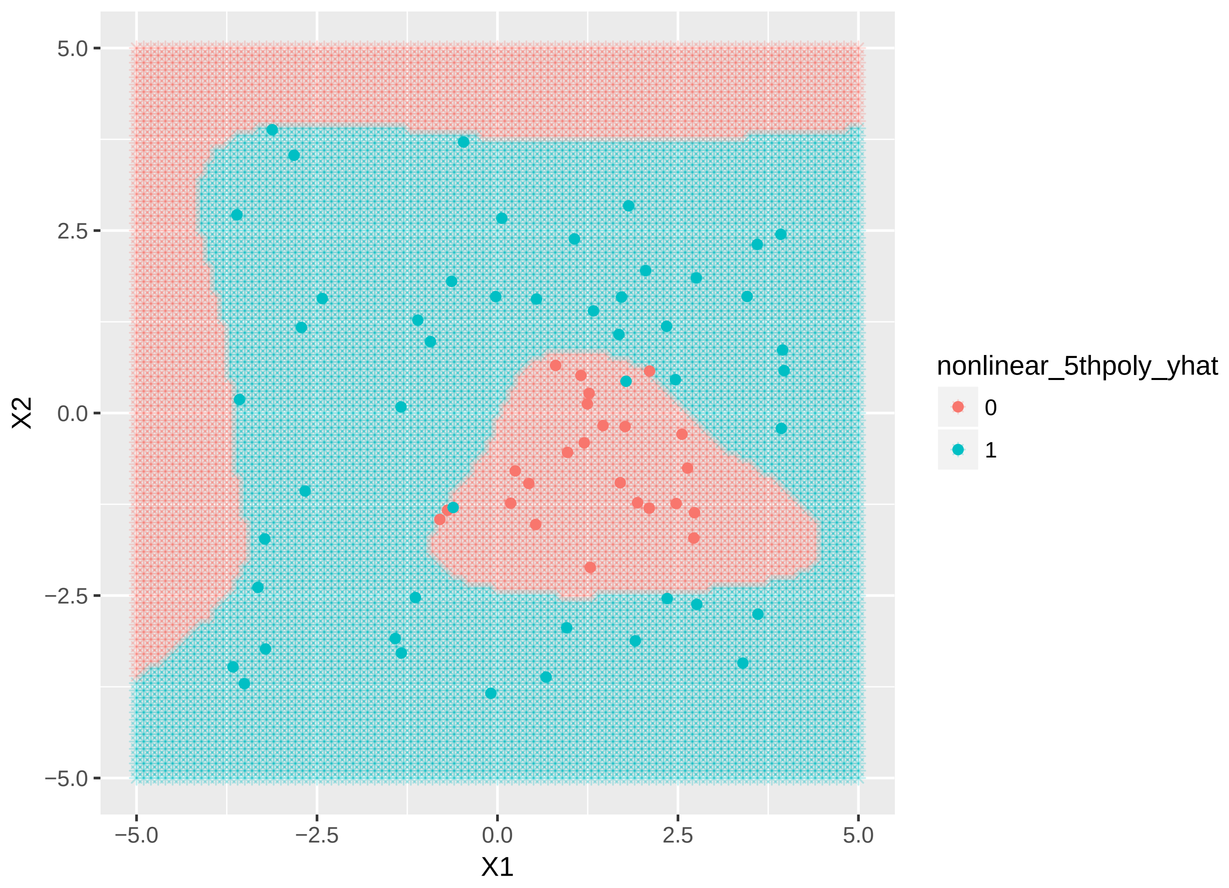
Inspecting the summary output (listed above), three of the coeffecients are significant in the interaction terms of the model; this indicates that because interactions were included in our model, the prediction was able to yield stronger predictions.

d)

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

```
##
## Call:
## glm(formula = Y ~ poly(X1, degree = 5) + poly(X2, degree = 5),
##     family = "binomial", data = nonlinear_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2441  -0.0209   0.0000   0.0008   1.8548
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)               25.4       41.1    0.62     0.54
## poly(X1, degree = 5)1    -49.3       88.4   -0.56     0.58
## poly(X1, degree = 5)2     25.9       36.9    0.70     0.48
## poly(X1, degree = 5)3     36.2       61.0    0.59     0.55
## poly(X1, degree = 5)4    -34.7       64.8   -0.54     0.59
## poly(X1, degree = 5)5     12.7       37.7    0.34     0.74
## poly(X2, degree = 5)1   -174.4      386.2   -0.45     0.65
## poly(X2, degree = 5)2    266.1      480.1    0.55     0.58
## poly(X2, degree = 5)3   -229.0      422.7   -0.54     0.59
## poly(X2, degree = 5)4     90.7      219.1    0.41     0.68
## poly(X2, degree = 5)5   -101.3      203.2   -0.50     0.62
##
## (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: 34.49
##
## 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 = .25, aes(col = nonlinear_5thpoly_yhat)) +
  geom_point(data = nonlinear_data,aes(col = Y))
```



The lack of an interaction plot gives us some undesireable resuls. 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)

From comparing coeffecients throughout the linear model and the two polynomial models, it is clear that the coefficients among the higher-order polynomial fits are larger in magnitude. This is related to the bias/variance trade-off - 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 flucuate tremendously in magnitude, represented by these coefficients. This trade-off is more clear when looking at the second-degree polynomial model where the degree is much smaller resembling lesser fluctuations, yielding smaller coeffecients. Finally, with the linear model, a first-degree polynomial is simply a line, resembling no fluctuation and therefore contains smaller coefficients.

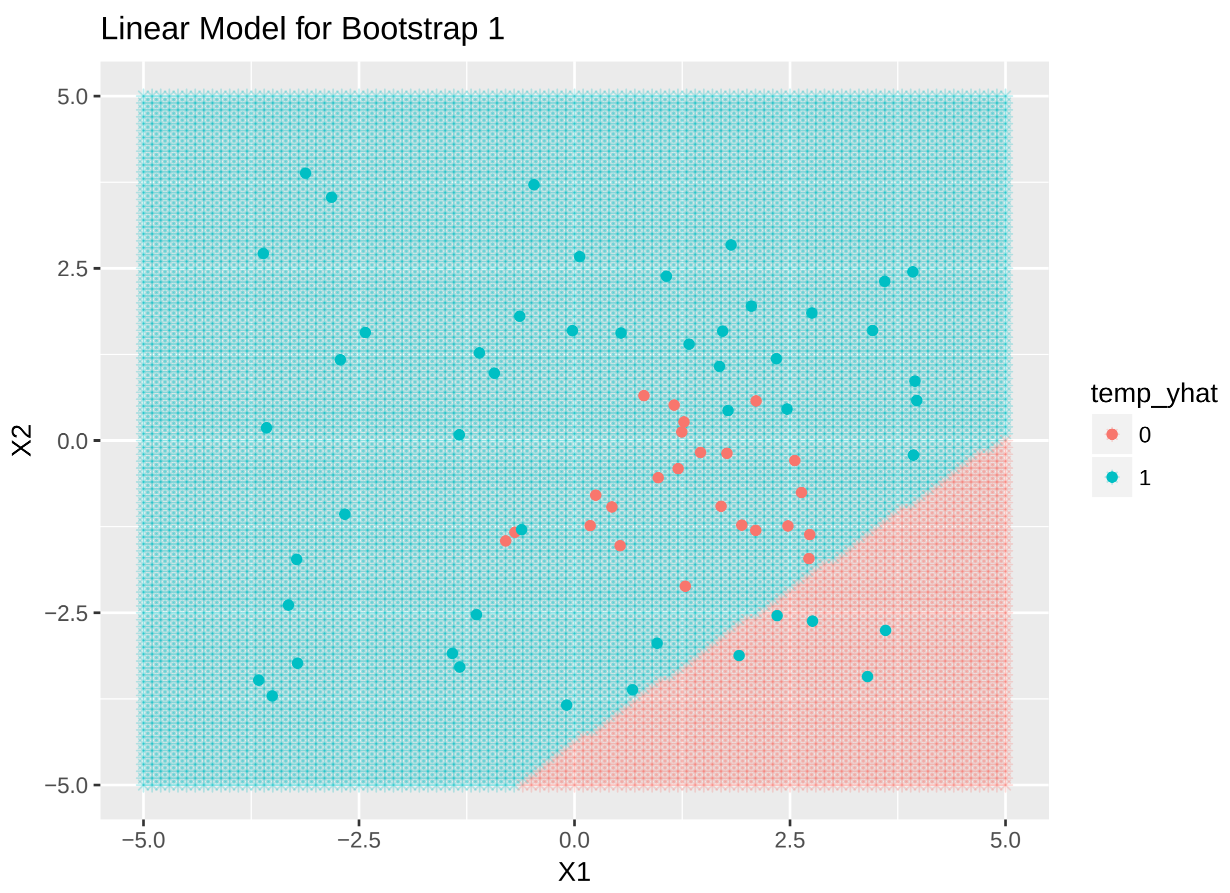
It is worth noting that the `poly()` function creates orthogonal vectors which changes the outcomes of the coefficients. It may be more accurate to either use raw values for this polynomial or to simply fit the model with explicit predictor variables, but we will obtain the same conclusion nonetheless.

f)

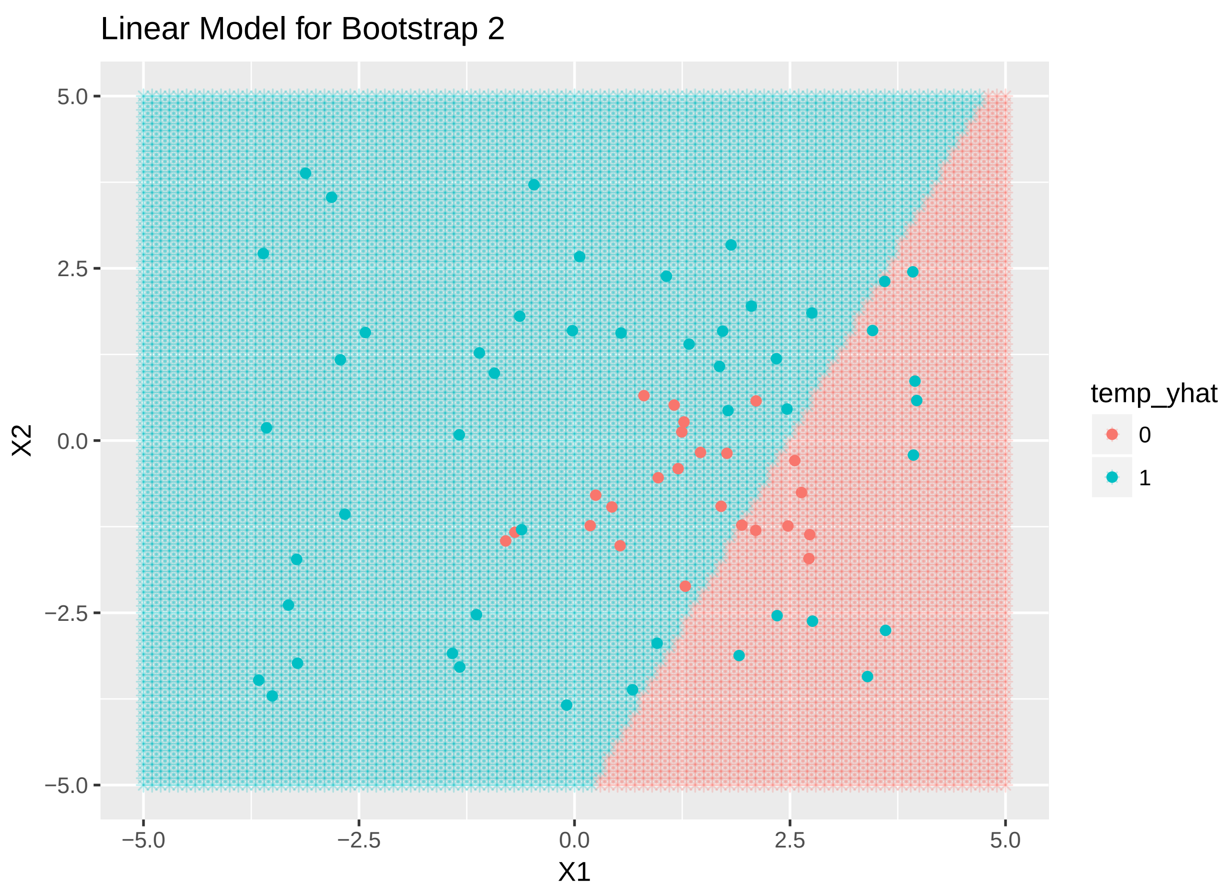
```r
set.seed(1)
boot1 <- nonlinear_data[sample(nrow(nonlinear_data), replace = T), ]
boot1_lm <- glm(Y ~ X1 + X2, data = boot1, family = "binomial")
```

```r
boot1_5th_fit <- glm(Y ~ poly(X1, degree = 5, raw = F)
                                  + poly(X2, degree = 5, raw = F),
                                  data = boot1, family = "binomial")
set.seed(2)
boot2 <- nonlinear_data[sample(nrow(nonlinear_data), replace = T), ]
boot2_lm <- glm(Y ~ X1 + X2, data = boot2, family = "binomial")
boot2_5th_fit <- glm(Y ~ poly(X1, degree = 5, raw = F)
                                  + poly(X2, degree = 5, raw = F),
                                  data = boot2, family = "binomial")
set.seed(3)
boot3 <- nonlinear_data[sample(nrow(nonlinear_data), replace = T), ]
boot3_lm <- glm(Y ~ X1 + X2, data = boot3, family = "binomial")
boot3_5th_fit <- glm(Y ~ poly(X1, degree = 5, raw = F)
                                  + poly(X2, degree = 5, raw = F),
                                  data = boot3, family = "binomial")
```

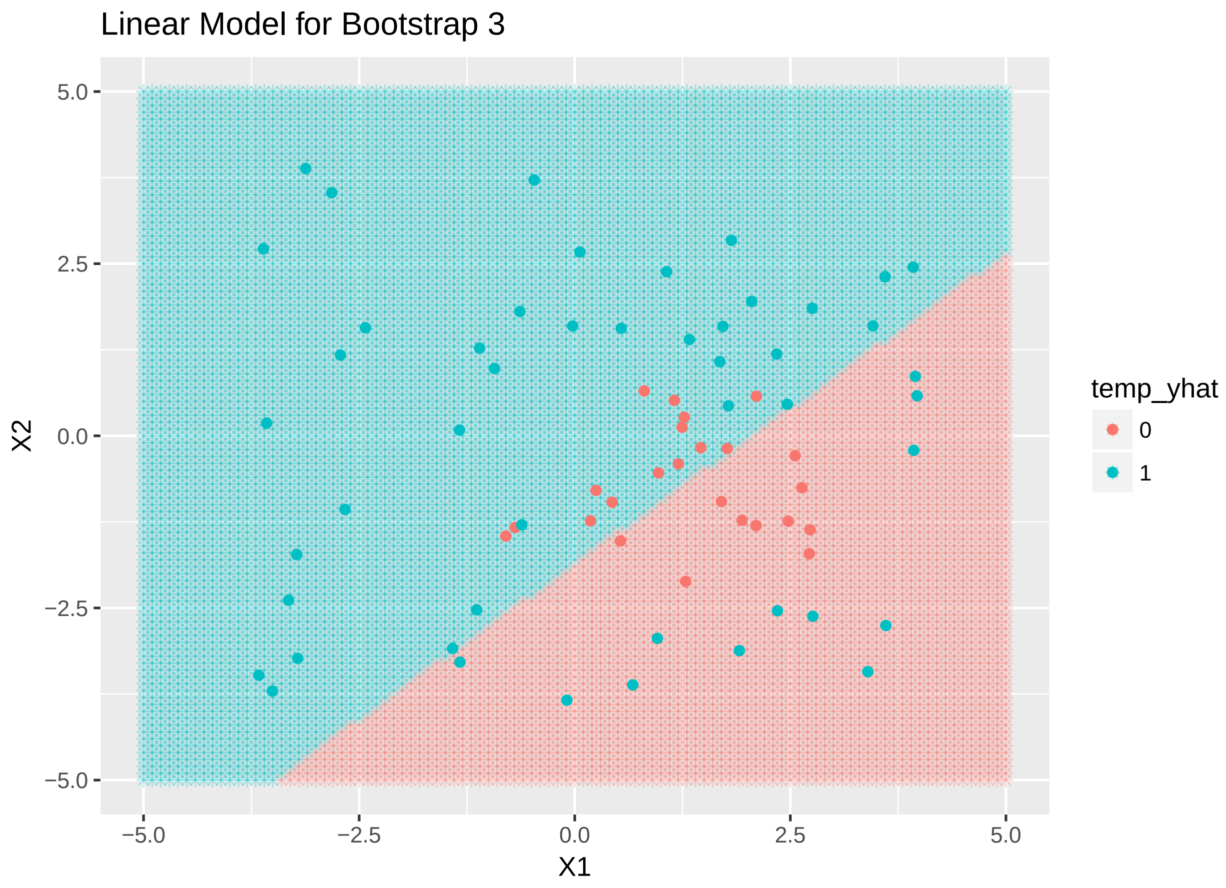
```r
plot_glm <- function(glm_fit, title){
  temp_yhat <- factor(ifelse(predict(glm_fit, gr, type = "response") >= .5, 1, 0))

  ggplot(mapping = aes(x=X1, y=X2)) +
    geom_point(data = gr, shape = 8 , alpha = .25, aes(col = temp_yhat)) +
    geom_point(data = nonlinear_data, aes(col=Y)) +
    labs(title = title)
}

plot_glm(boot1_lm, "Linear Model for Bootstrap 1")
```



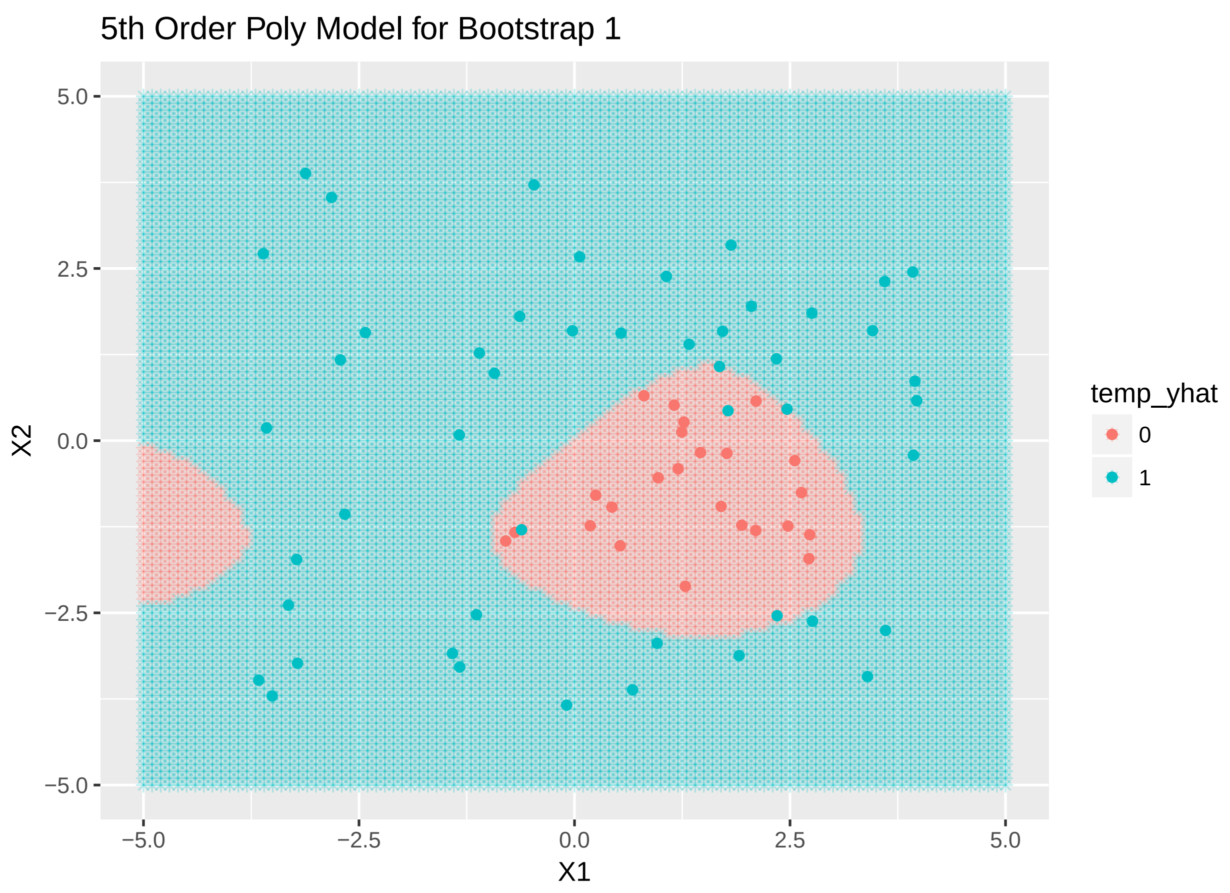
```r
plot_glm(boot2_lm, "Linear Model for Bootstrap 2")
```



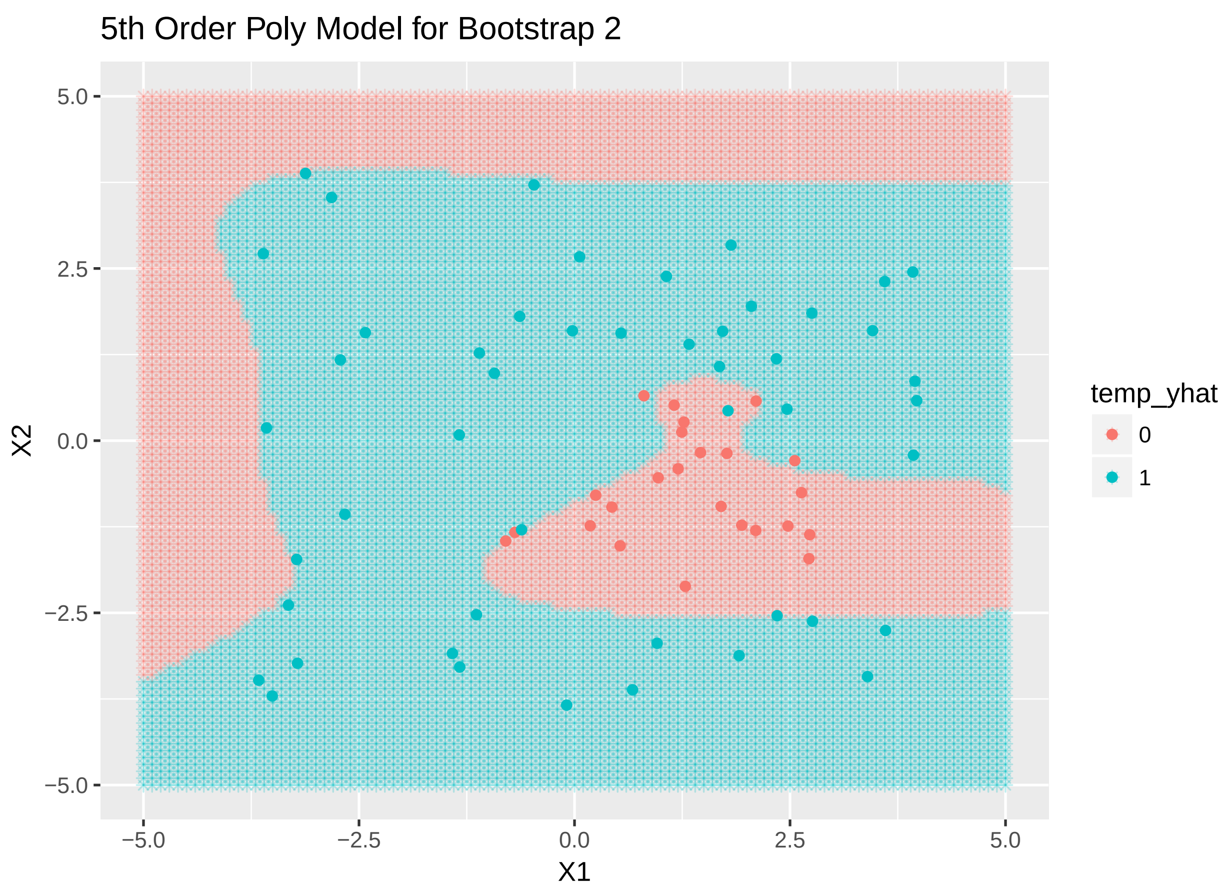
```r
plot_glm(boot3_lm, "Linear Model for Bootstrap 3")
```



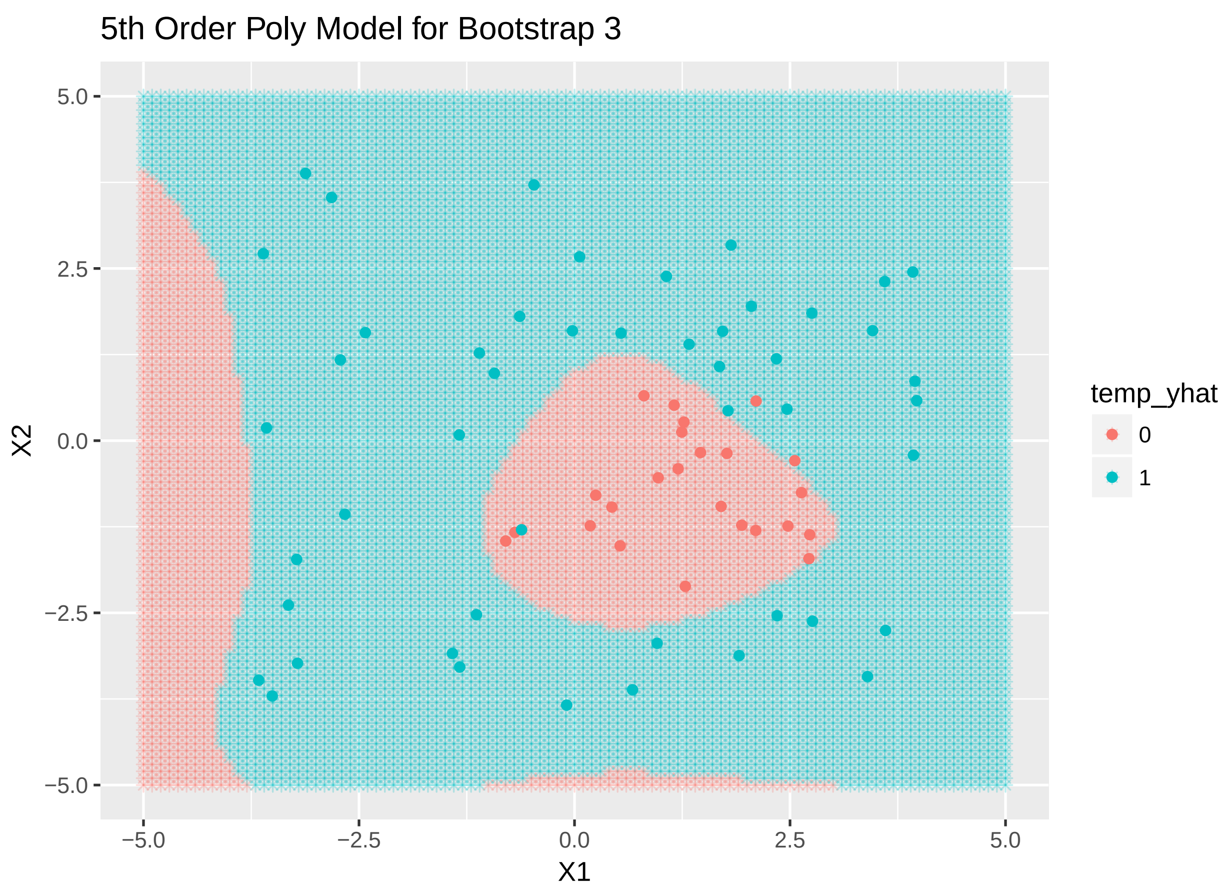
```
plot_glm(boot1_5th_fit, "5th Order Poly Model for Bootstrap 1")
```



```r
plot_glm(boot2_5th_fit, "5th Order Poly Model for Bootstrap 2")
```



```r
plot_glm(boot3_5th_fit, "5th Order Poly Model for Bootstrap 3")
```



As expected, the linear model has much more variation than the 5th-order polynomial model. The decision boundary for the linear model changes in slope and location fairly significantly. The 5th-order polynomial model does vary in the region where there are no true values (it cannot predict this region accurately) but the overall decision boundary will misclassify less values across different bootstrap samples.