final\_project\_p8106

Michelle Lui

4/27/2021

## Import dataset

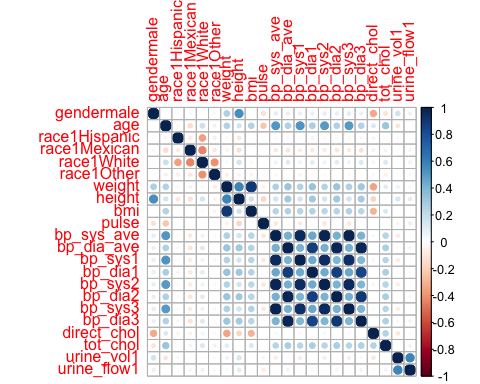
#import dataset  
data("NHANES")  
  
#clean dataset - limit to 2011/12 and only include biological predictors, dropped some biological predictors due to excess missing data  
nhanes = NHANES %>% janitor::clean\_names() %>% filter(survey\_yr == "2011\_12") %>% select(gender, age, race1, weight, height, bmi, pulse:bp\_dia3, direct\_chol:urine\_flow1, diabetes) %>% drop\_na()

# Exploratory Data Analysis

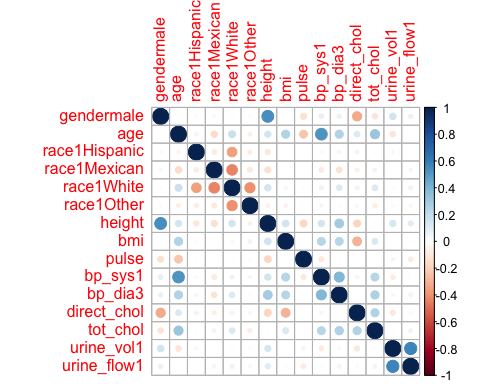
nhanes\_2 =   
 nhanes %>%  
 select(diabetes, everything())  
  
summary(nhanes\_2)

## diabetes gender age race1 weight   
## No :3223 female:1735 Min. : 8.00 Black : 392 Min. : 23.00   
## Yes: 293 male :1781 1st Qu.:25.00 Hispanic: 218 1st Qu.: 62.90   
## Median :42.00 Mexican : 315 Median : 77.10   
## Mean :41.81 White :2319 Mean : 78.25   
## 3rd Qu.:57.00 Other : 272 3rd Qu.: 91.10   
## Max. :80.00 Max. :188.50   
## height bmi pulse bp\_sys\_ave   
## Min. :124.5 Min. :12.90 Min. : 40.00 Min. : 81   
## 1st Qu.:160.3 1st Qu.:22.90 1st Qu.: 64.00 1st Qu.:107   
## Median :167.8 Median :26.60 Median : 72.00 Median :117   
## Mean :167.5 Mean :27.63 Mean : 73.41 Mean :119   
## 3rd Qu.:175.4 3rd Qu.:31.30 3rd Qu.: 82.00 3rd Qu.:128   
## Max. :199.9 Max. :69.00 Max. :136.00 Max. :209   
## bp\_dia\_ave bp\_sys1 bp\_dia1 bp\_sys2   
## Min. : 0.00 Min. : 74.0 Min. : 0.0 Min. : 82.0   
## 1st Qu.: 62.00 1st Qu.:108.0 1st Qu.: 62.0 1st Qu.:108.0   
## Median : 69.00 Median :118.0 Median : 70.0 Median :118.0   
## Mean : 68.44 Mean :119.5 Mean : 69.2 Mean :119.2   
## 3rd Qu.: 77.00 3rd Qu.:128.0 3rd Qu.: 78.0 3rd Qu.:128.0   
## Max. :116.00 Max. :212.0 Max. :110.0 Max. :208.0   
## bp\_dia2 bp\_sys3 bp\_dia3 direct\_chol   
## Min. : 0.00 Min. : 78.0 Min. : 0.00 Min. :0.470   
## 1st Qu.: 62.00 1st Qu.:106.0 1st Qu.: 62.00 1st Qu.:1.110   
## Median : 70.00 Median :116.0 Median : 70.00 Median :1.290   
## Mean : 68.55 Mean :118.7 Mean : 68.33 Mean :1.365   
## 3rd Qu.: 78.00 3rd Qu.:128.0 3rd Qu.: 78.00 3rd Qu.:1.580   
## Max. :116.00 Max. :210.0 Max. :116.00 Max. :4.030   
## tot\_chol urine\_vol1 urine\_flow1   
## Min. : 1.530 Min. : 1 Min. : 0.0110   
## 1st Qu.: 4.110 1st Qu.: 47 1st Qu.: 0.3995   
## Median : 4.780 Median : 88 Median : 0.6800   
## Mean : 4.858 Mean :113 Mean : 0.9642   
## 3rd Qu.: 5.530 3rd Qu.:156 3rd Qu.: 1.2140   
## Max. :10.290 Max. :446 Max. :10.1430

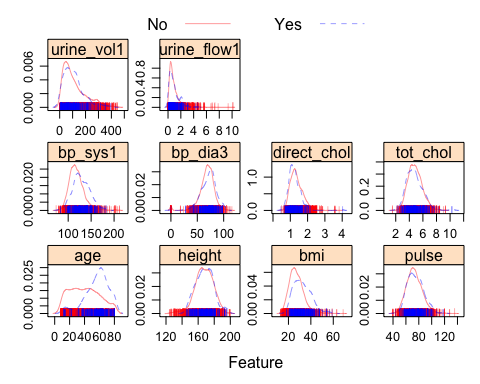
#correlation plots for predictor variables  
corrx = model.matrix(diabetes~., nhanes\_2)[,-1]  
corrplot(cor(corrx))



#remove variables with correlation greater than 0.8  
indexesToDrop = findCorrelation(cor(corrx), cutoff = 0.8)  
corrplot(cor(corrx[,-indexesToDrop]))



#final dataset for analysis using RF and analysis  
nhanes\_3 =   
 nhanes\_2 %>%  
 select(diabetes, age, height, bmi, pulse, bp\_sys1, bp\_dia3, direct\_chol, tot\_chol,  
 urine\_vol1, urine\_flow1, gender, race1)  
  
  
# look at feature plots for continuous predictors (everything but race1 and gender)  
theme1 = transparentTheme(trans =.4)  
trellis.par.set(theme1)  
  
  
featurePlot(x=nhanes\_3[,2:11],  
 y=nhanes\_3$diabetes,  
 scales =list(x=list(relation ="free"),  
 y=list(relation ="free")),  
 plot ="density",pch ="|",  
 auto.key =list(columns =2))



## Create Data Partition

set.seed(2)  
  
train\_rows = createDataPartition(y = nhanes\_3$diabetes,p = 0.7,list = FALSE)  
train = nhanes\_3[train\_rows, ]

## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.  
## Convert to a vector.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

test = nhanes\_3[-train\_rows, ]  
  
x = model.matrix(diabetes ~ ., train)[ ,-1]  
y = train$diabetes  
  
x2 = model.matrix(diabetes ~ ., test)[ ,-1]  
y2 = test$diabetes  
  
control1 = trainControl(method = "cv", selectionFunction = "best", sampling = "down")

## Lasso Model

set.seed(2)  
lasso\_fit = train(x, y,   
 method = "glmnet",  
 tuneGrid = expand.grid(alpha = 1, lambda = exp(seq(-20, 20,length=100))),  
 trControl = control1,  
 preProcess=c("center", "scale"),  
 family = "binomial")   
  
#Print the values of alpha and lambda that gave best prediction  
lasso\_fit$bestTune

## alpha lambda  
## 38 1 0.006406097

#Print all of the options examined  
lasso\_fit$results

## alpha lambda Accuracy Kappa AccuracySD KappaSD  
## 1 1 2.061154e-09 0.7316292 0.2097020 0.03069130 0.04648869  
## 2 1 3.087329e-09 0.7316292 0.2097020 0.03069130 0.04648869  
## 3 1 4.624400e-09 0.7316292 0.2097020 0.03069130 0.04648869  
## 4 1 6.926725e-09 0.7316292 0.2097020 0.03069130 0.04648869  
## 5 1 1.037529e-08 0.7316292 0.2097020 0.03069130 0.04648869  
## 6 1 1.554078e-08 0.7316292 0.2097020 0.03069130 0.04648869  
## 7 1 2.327799e-08 0.7316292 0.2097020 0.03069130 0.04648869  
## 8 1 3.486727e-08 0.7316292 0.2097020 0.03069130 0.04648869  
## 9 1 5.222645e-08 0.7316292 0.2097020 0.03069130 0.04648869  
## 10 1 7.822814e-08 0.7316292 0.2097020 0.03069130 0.04648869  
## 11 1 1.171752e-07 0.7316292 0.2097020 0.03069130 0.04648869  
## 12 1 1.755125e-07 0.7316292 0.2097020 0.03069130 0.04648869  
## 13 1 2.628939e-07 0.7316292 0.2097020 0.03069130 0.04648869  
## 14 1 3.937795e-07 0.7316292 0.2097020 0.03069130 0.04648869  
## 15 1 5.898283e-07 0.7316292 0.2097020 0.03069130 0.04648869  
## 16 1 8.834829e-07 0.7316292 0.2097020 0.03069130 0.04648869  
## 17 1 1.323338e-06 0.7316292 0.2097020 0.03069130 0.04648869  
## 18 1 1.982180e-06 0.7316292 0.2097020 0.03069130 0.04648869  
## 19 1 2.969038e-06 0.7316292 0.2097020 0.03069130 0.04648869  
## 20 1 4.447216e-06 0.7316292 0.2097020 0.03069130 0.04648869  
## 21 1 6.661326e-06 0.7316292 0.2097020 0.03069130 0.04648869  
## 22 1 9.977764e-06 0.7316292 0.2097020 0.03069130 0.04648869  
## 23 1 1.494534e-05 0.7316292 0.2097020 0.03069130 0.04648869  
## 24 1 2.238609e-05 0.7316292 0.2097020 0.03069130 0.04648869  
## 25 1 3.353133e-05 0.7316292 0.2097020 0.03069130 0.04648869  
## 26 1 5.022539e-05 0.7316292 0.2097020 0.03069130 0.04648869  
## 27 1 7.523083e-05 0.7316292 0.2097020 0.03069130 0.04648869  
## 28 1 1.126856e-04 0.7316292 0.2097020 0.03069130 0.04648869  
## 29 1 1.687877e-04 0.7316292 0.2097020 0.03069130 0.04648869  
## 30 1 2.528211e-04 0.7316292 0.2097020 0.03069130 0.04648869  
## 31 1 3.786918e-04 0.7316292 0.2097020 0.03069130 0.04648869  
## 32 1 5.672290e-04 0.7312227 0.2093622 0.03089586 0.04713098  
## 33 1 8.496321e-04 0.7308162 0.2107979 0.03156203 0.04394624  
## 34 1 1.272634e-03 0.7320340 0.2118739 0.03043488 0.04329136  
## 35 1 1.906233e-03 0.7320357 0.2119628 0.03101889 0.04444054  
## 36 1 2.855279e-03 0.7320340 0.2114728 0.02786084 0.04070302  
## 37 1 4.276820e-03 0.7304113 0.2101363 0.02952107 0.04323945  
## 38 1 6.406097e-03 0.7320390 0.2160079 0.02794914 0.04695292  
## 39 1 9.595465e-03 0.7235238 0.2088359 0.02496772 0.04389999  
## 40 1 1.437271e-02 0.7198652 0.2181297 0.02588053 0.03343138  
## 41 1 2.152837e-02 0.7174328 0.2185267 0.02293235 0.03639613  
## 42 1 3.224658e-02 0.7105238 0.2071801 0.02161712 0.03689326  
## 43 1 4.830100e-02 0.7048426 0.2016200 0.01713358 0.03999908  
## 44 1 7.234835e-02 0.6971305 0.1907318 0.01528560 0.03501996  
## 45 1 1.083680e-01 0.6861796 0.1863047 0.02413036 0.04322860  
## 46 1 1.623206e-01 0.6715618 0.1752595 0.03056515 0.04497615  
## 47 1 2.431343e-01 0.4185278 0.0000000 0.43040766 0.00000000  
## 48 1 3.641822e-01 0.4185278 0.0000000 0.43040766 0.00000000  
## 49 1 5.454956e-01 0.4185278 0.0000000 0.43040766 0.00000000  
## 50 1 8.170784e-01 0.4185278 0.0000000 0.43040766 0.00000000  
## 51 1 1.223873e+00 0.4185278 0.0000000 0.43040766 0.00000000  
## 52 1 1.833195e+00 0.4185278 0.0000000 0.43040766 0.00000000  
## 53 1 2.745878e+00 0.4185278 0.0000000 0.43040766 0.00000000  
## 54 1 4.112954e+00 0.4185278 0.0000000 0.43040766 0.00000000  
## 55 1 6.160647e+00 0.4185278 0.0000000 0.43040766 0.00000000  
## 56 1 9.227814e+00 0.4185278 0.0000000 0.43040766 0.00000000  
## 57 1 1.382202e+01 0.4185278 0.0000000 0.43040766 0.00000000  
## 58 1 2.070351e+01 0.4185278 0.0000000 0.43040766 0.00000000  
## 59 1 3.101105e+01 0.4185278 0.0000000 0.43040766 0.00000000  
## 60 1 4.645034e+01 0.4185278 0.0000000 0.43040766 0.00000000  
## 61 1 6.957632e+01 0.4185278 0.0000000 0.43040766 0.00000000  
## 62 1 1.042159e+02 0.4185278 0.0000000 0.43040766 0.00000000  
## 63 1 1.561013e+02 0.4185278 0.0000000 0.43040766 0.00000000  
## 64 1 2.338186e+02 0.4185278 0.0000000 0.43040766 0.00000000  
## 65 1 3.502285e+02 0.4185278 0.0000000 0.43040766 0.00000000  
## 66 1 5.245949e+02 0.4185278 0.0000000 0.43040766 0.00000000  
## 67 1 7.857720e+02 0.4185278 0.0000000 0.43040766 0.00000000  
## 68 1 1.176980e+03 0.4185278 0.0000000 0.43040766 0.00000000  
## 69 1 1.762956e+03 0.4185278 0.0000000 0.43040766 0.00000000  
## 70 1 2.640670e+03 0.4185278 0.0000000 0.43040766 0.00000000  
## 71 1 3.955365e+03 0.4185278 0.0000000 0.43040766 0.00000000  
## 72 1 5.924601e+03 0.4185278 0.0000000 0.43040766 0.00000000  
## 73 1 8.874250e+03 0.4185278 0.0000000 0.43040766 0.00000000  
## 74 1 1.329242e+04 0.4185278 0.0000000 0.43040766 0.00000000  
## 75 1 1.991025e+04 0.4185278 0.0000000 0.43040766 0.00000000  
## 76 1 2.982285e+04 0.4185278 0.0000000 0.43040766 0.00000000  
## 77 1 4.467059e+04 0.4185278 0.0000000 0.43040766 0.00000000  
## 78 1 6.691050e+04 0.4185278 0.0000000 0.43040766 0.00000000  
## 79 1 1.002229e+05 0.4185278 0.0000000 0.43040766 0.00000000  
## 80 1 1.501203e+05 0.4185278 0.0000000 0.43040766 0.00000000  
## 81 1 2.248598e+05 0.4185278 0.0000000 0.43040766 0.00000000  
## 82 1 3.368095e+05 0.4185278 0.0000000 0.43040766 0.00000000  
## 83 1 5.044950e+05 0.4185278 0.0000000 0.43040766 0.00000000  
## 84 1 7.556651e+05 0.4185278 0.0000000 0.43040766 0.00000000  
## 85 1 1.131884e+06 0.4185278 0.0000000 0.43040766 0.00000000  
## 86 1 1.695409e+06 0.4185278 0.0000000 0.43040766 0.00000000  
## 87 1 2.539492e+06 0.4185278 0.0000000 0.43040766 0.00000000  
## 88 1 3.803815e+06 0.4185278 0.0000000 0.43040766 0.00000000  
## 89 1 5.697600e+06 0.4185278 0.0000000 0.43040766 0.00000000  
## 90 1 8.534232e+06 0.4185278 0.0000000 0.43040766 0.00000000  
## 91 1 1.278312e+07 0.4185278 0.0000000 0.43040766 0.00000000  
## 92 1 1.914739e+07 0.4185278 0.0000000 0.43040766 0.00000000  
## 93 1 2.868019e+07 0.4185278 0.0000000 0.43040766 0.00000000  
## 94 1 4.295904e+07 0.4185278 0.0000000 0.43040766 0.00000000  
## 95 1 6.434681e+07 0.4185278 0.0000000 0.43040766 0.00000000  
## 96 1 9.638281e+07 0.4185278 0.0000000 0.43040766 0.00000000  
## 97 1 1.443684e+08 0.4185278 0.0000000 0.43040766 0.00000000  
## 98 1 2.162443e+08 0.4185278 0.0000000 0.43040766 0.00000000  
## 99 1 3.239046e+08 0.4185278 0.0000000 0.43040766 0.00000000  
## 100 1 4.851652e+08 0.4185278 0.0000000 0.43040766 0.00000000

# Model coefficients  
coef(lasso\_fit$finalModel, s = lasso\_fit$bestTune$lambda)

## 16 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -0.04598234  
## age 1.13689677  
## height -0.03059534  
## bmi 0.69433892  
## pulse 0.41099385  
## bp\_sys1 0.09701002  
## bp\_dia3 0.02860917  
## direct\_chol -0.31025394  
## tot\_chol -0.07709907  
## urine\_vol1 0.09126462  
## urine\_flow1 0.11977272  
## gendermale 0.11819933  
## race1Hispanic .   
## race1Mexican -0.12418383  
## race1White -0.31766043  
## race1Other .

# Make predictions  
lasso\_pred = lasso\_fit %>% predict(x2) %>% as.numeric()  
lasso\_pred\_p = ifelse(lasso\_pred-1 > 0.5,1,0)  
  
test\_outcome\_lasso = (as.numeric(y2)-1)  
  
misclasserror\_lasso = mean(lasso\_pred\_p != test\_outcome\_lasso, na.rm=T)  
print(paste('Accuracy Model 1', 1-misclasserror\_lasso))

## [1] "Accuracy Model 1 0.682811016144349"

## Random Forest

mtry\_vals = c(ncol(train)-1, sqrt(ncol(train)-1), 0.5\*ncol(train)-1)  
  
mtry\_grid = expand.grid(.mtry=mtry\_vals)  
  
set.seed(2)  
rf\_fit = train(diabetes ~.,   
 data = train,   
 method="rf",   
 trControl = control1,   
 metric="Accuracy",   
 tuneGrid=mtry\_grid,   
 ntree=100)  
  
rf\_fit$results

## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 3.464102 0.8031067 0.3411511 0.03894405 0.05986997  
## 2 5.500000 0.7981824 0.3334355 0.02040004 0.04224013  
## 3 12.000000 0.7994266 0.3212905 0.01503463 0.03661672

rf\_fit$bestTune

## mtry  
## 1 3.464102

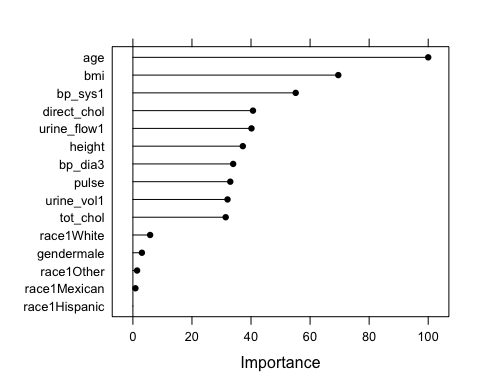
rf\_fit$finalModel

##   
## Call:  
## randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 16.02%  
## Confusion matrix:  
## No Yes class.error  
## No 166 40 0.1941748  
## Yes 26 180 0.1262136

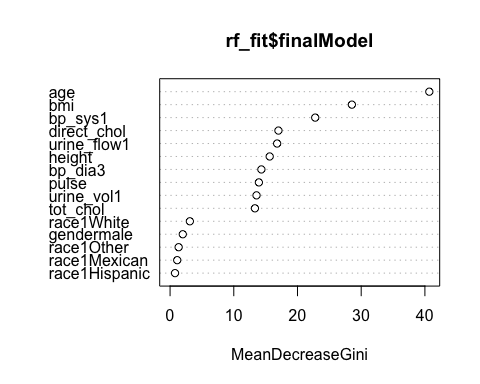
varImp(rf\_fit)

## rf variable importance  
##   
## Overall  
## age 100.0000  
## bmi 69.5471  
## bp\_sys1 55.1292  
## direct\_chol 40.6630  
## urine\_flow1 40.1548  
## height 37.2218  
## bp\_dia3 33.9554  
## pulse 32.9752  
## urine\_vol1 32.0458  
## tot\_chol 31.4214  
## race1White 5.8331  
## gendermale 3.0511  
## race1Other 1.4182  
## race1Mexican 0.8581  
## race1Hispanic 0.0000

plot(varImp(rf\_fit))



varImpPlot(rf\_fit$finalModel)



rf\_pred = predict(rf\_fit, test) %>% as.numeric()  
rf\_pred\_p = ifelse(rf\_pred-1 > 0.5,1,0)  
  
test\_outcome\_rf = (as.numeric(test$diabetes)-1)  
  
misclasserror\_rf = mean(rf\_pred\_p != test\_outcome\_rf, na.rm=T)  
print(paste('Accuracy Model 2', 1-misclasserror\_rf))

## [1] "Accuracy Model 2 0.804368471035138"

## Model Comparisons

resamp = resamples(list(lasso = lasso\_fit, rf = rf\_fit))   
summary(resamp)

##   
## Call:  
## summary.resamples(object = resamp)  
##   
## Models: lasso, rf   
## Number of resamples: 10   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lasso 0.6869919 0.7154472 0.7302261 0.7320390 0.7515492 0.7764228 0  
## rf 0.7439024 0.7786733 0.8008130 0.8031067 0.8401760 0.8536585 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lasso 0.1460028 0.1816105 0.2281826 0.2160079 0.2448435 0.2893161 0  
## rf 0.2792530 0.2945411 0.3216341 0.3411511 0.3837602 0.4595386 0

## Final dataset after selecting variables from RF model

We selected variables from our random forest model since it had the highest accuracy compared to the lasso model

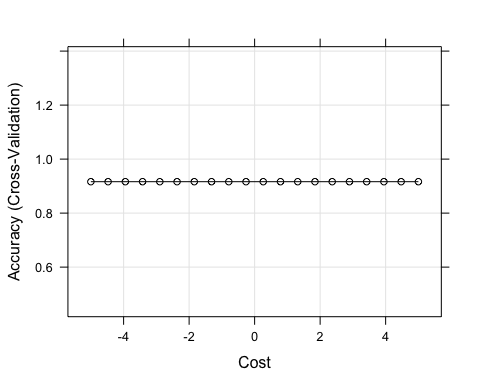
#Final dataset after variable selection  
nhanes\_4 =   
 nhanes\_3 %>%  
 select(diabetes, age, bmi, bp\_sys1, direct\_chol, urine\_flow1, height, bp\_dia3, pulse, urine\_vol1,  
 tot\_chol)  
  
#training and testing  
  
train\_rows2 = createDataPartition(y = nhanes\_4$diabetes,p = 0.7,list = FALSE)  
train2 = nhanes\_4[train\_rows, ]  
test2 = nhanes\_4[-train\_rows, ]

## SVC Analysis

ctrl = trainControl(method = "cv")  
  
#Analysis using linear kernal  
# kernlab  
set.seed(1)  
svml.fit = train(diabetes ~ . ,   
 data = nhanes\_4[train\_rows,],   
 method = "svmLinear",  
 preProcess = c("center", "scale"),  
 tuneGrid = data.frame(C = exp(seq(-5,5,len=20))),  
 trControl = ctrl)  
svml.fit #C = 0.006737

## Support Vector Machines with Linear Kernel   
##   
## 2463 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## Pre-processing: centered (10), scaled (10)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2218, 2217, 2216, 2217, 2216, 2217, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa  
## 6.737947e-03 0.9163666 0   
## 1.140522e-02 0.9163666 0   
## 1.930544e-02 0.9163666 0   
## 3.267802e-02 0.9163666 0   
## 5.531358e-02 0.9163666 0   
## 9.362844e-02 0.9163666 0   
## 1.584834e-01 0.9163666 0   
## 2.682625e-01 0.9163666 0   
## 4.540837e-01 0.9163666 0   
## 7.686205e-01 0.9163666 0   
## 1.301032e+00 0.9163666 0   
## 2.202237e+00 0.9163666 0   
## 3.727693e+00 0.9163666 0   
## 6.309808e+00 0.9163666 0   
## 1.068051e+01 0.9163666 0   
## 1.807874e+01 0.9163666 0   
## 3.060161e+01 0.9163666 0   
## 5.179887e+01 0.9163666 0   
## 8.767916e+01 0.9163666 0   
## 1.484132e+02 0.9163666 0   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was C = 0.006737947.

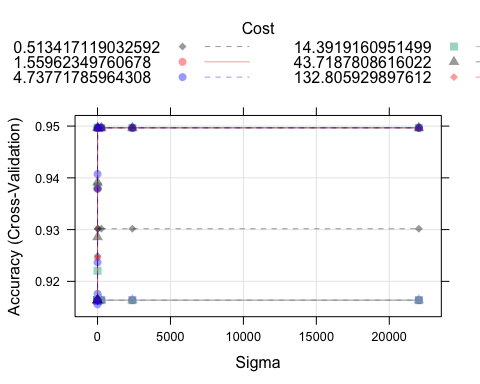
plot(svml.fit, highlight = TRUE, xTrans = log)



#Analysis using radial kernal  
svmr.grid = expand.grid(C = exp(seq(-4,6,len=10)),  
 sigma = exp(seq(-10,10,len=10)))  
  
# tunes over both cost and sigma  
set.seed(1)   
svmr.fit = train(diabetes ~ . , nhanes\_4,  
 subset = train\_rows2,  
 method = "svmRadialSigma",  
 preProcess = c("center", "scale"),  
 tuneGrid = svmr.grid,  
 trControl = ctrl)  
svmr.fit #C = 1.559623, sigma = 0.22026,47 - tune differently??

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 3516 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## Pre-processing: centered (10), scaled (10)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2218, 2217, 2216, 2217, 2216, 2217, ...   
## Resampling results across tuning parameters:  
##   
## C sigma Accuracy Kappa   
## 0.01831564 4.539993e-05 0.9163666 0.000000000  
## 0.01831564 4.189421e-04 0.9163666 0.000000000  
## 0.01831564 3.865920e-03 0.9163666 0.000000000  
## 0.01831564 3.567399e-02 0.9163666 0.000000000  
## 0.01831564 3.291930e-01 0.9163666 0.000000000  
## 0.01831564 3.037732e+00 0.9163666 0.000000000  
## 0.01831564 2.803162e+01 0.9163666 0.000000000  
## 0.01831564 2.586706e+02 0.9163666 0.000000000  
## 0.01831564 2.386965e+03 0.9163666 0.000000000  
## 0.01831564 2.202647e+04 0.9163666 0.000000000  
## 0.05563800 4.539993e-05 0.9163666 0.000000000  
## 0.05563800 4.189421e-04 0.9163666 0.000000000  
## 0.05563800 3.865920e-03 0.9163666 0.000000000  
## 0.05563800 3.567399e-02 0.9163666 0.000000000  
## 0.05563800 3.291930e-01 0.9163666 0.000000000  
## 0.05563800 3.037732e+00 0.9163666 0.000000000  
## 0.05563800 2.803162e+01 0.9163666 0.000000000  
## 0.05563800 2.586706e+02 0.9163666 0.000000000  
## 0.05563800 2.386965e+03 0.9163666 0.000000000  
## 0.05563800 2.202647e+04 0.9163666 0.000000000  
## 0.16901332 4.539993e-05 0.9163666 0.000000000  
## 0.16901332 4.189421e-04 0.9163666 0.000000000  
## 0.16901332 3.865920e-03 0.9163666 0.000000000  
## 0.16901332 3.567399e-02 0.9163666 0.000000000  
## 0.16901332 3.291930e-01 0.9163666 0.000000000  
## 0.16901332 3.037732e+00 0.9163666 0.000000000  
## 0.16901332 2.803162e+01 0.9163666 0.000000000  
## 0.16901332 2.586706e+02 0.9163666 0.000000000  
## 0.16901332 2.386965e+03 0.9163666 0.000000000  
## 0.16901332 2.202647e+04 0.9163666 0.000000000  
## 0.51341712 4.539993e-05 0.9163666 0.000000000  
## 0.51341712 4.189421e-04 0.9163666 0.000000000  
## 0.51341712 3.865920e-03 0.9163666 0.000000000  
## 0.51341712 3.567399e-02 0.9163666 0.000000000  
## 0.51341712 3.291930e-01 0.9248917 0.174496260  
## 0.51341712 3.037732e+00 0.9301697 0.260070418  
## 0.51341712 2.803162e+01 0.9301697 0.260070418  
## 0.51341712 2.586706e+02 0.9301697 0.260070418  
## 0.51341712 2.386965e+03 0.9301697 0.260070418  
## 0.51341712 2.202647e+04 0.9301697 0.260070418  
## 1.55962350 4.539993e-05 0.9163666 0.000000000  
## 1.55962350 4.189421e-04 0.9163666 0.000000000  
## 1.55962350 3.865920e-03 0.9163666 0.000000000  
## 1.55962350 3.567399e-02 0.9163666 0.000000000  
## 1.55962350 3.291930e-01 0.9378851 0.435530439  
## 1.55962350 3.037732e+00 0.9496656 0.544493949  
## 1.55962350 2.803162e+01 0.9496656 0.544493949  
## 1.55962350 2.586706e+02 0.9496656 0.544493949  
## 1.55962350 2.386965e+03 0.9496656 0.544493949  
## 1.55962350 2.202647e+04 0.9496656 0.544493949  
## 4.73771786 4.539993e-05 0.9163666 0.000000000  
## 4.73771786 4.189421e-04 0.9163666 0.000000000  
## 4.73771786 3.865920e-03 0.9163666 0.000000000  
## 4.73771786 3.567399e-02 0.9175861 0.048542441  
## 4.73771786 3.291930e-01 0.9407340 0.536874168  
## 4.73771786 3.037732e+00 0.9496656 0.544493949  
## 4.73771786 2.803162e+01 0.9496656 0.544493949  
## 4.73771786 2.586706e+02 0.9496656 0.544493949  
## 4.73771786 2.386965e+03 0.9496656 0.544493949  
## 4.73771786 2.202647e+04 0.9496656 0.544493949  
## 14.39191610 4.539993e-05 0.9163666 0.000000000  
## 14.39191610 4.189421e-04 0.9163666 0.000000000  
## 14.39191610 3.865920e-03 0.9163666 0.000000000  
## 14.39191610 3.567399e-02 0.9220412 0.211804570  
## 14.39191610 3.291930e-01 0.9383032 0.537637834  
## 14.39191610 3.037732e+00 0.9496656 0.544493949  
## 14.39191610 2.803162e+01 0.9496656 0.544493949  
## 14.39191610 2.586706e+02 0.9496656 0.544493949  
## 14.39191610 2.386965e+03 0.9496656 0.544493949  
## 14.39191610 2.202647e+04 0.9496656 0.544493949  
## 43.71878086 4.539993e-05 0.9163666 0.000000000  
## 43.71878086 4.189421e-04 0.9163666 0.000000000  
## 43.71878086 3.865920e-03 0.9163666 0.000000000  
## 43.71878086 3.567399e-02 0.9285519 0.372296847  
## 43.71878086 3.291930e-01 0.9391146 0.546667639  
## 43.71878086 3.037732e+00 0.9496656 0.544493949  
## 43.71878086 2.803162e+01 0.9496656 0.544493949  
## 43.71878086 2.586706e+02 0.9496656 0.544493949  
## 43.71878086 2.386965e+03 0.9496656 0.544493949  
## 43.71878086 2.202647e+04 0.9496656 0.544493949  
## 132.80592990 4.539993e-05 0.9163666 0.000000000  
## 132.80592990 4.189421e-04 0.9163666 0.000000000  
## 132.80592990 3.865920e-03 0.9163666 0.000000000  
## 132.80592990 3.567399e-02 0.9244984 0.408029906  
## 132.80592990 3.291930e-01 0.9378951 0.544135498  
## 132.80592990 3.037732e+00 0.9496656 0.544493949  
## 132.80592990 2.803162e+01 0.9496656 0.544493949  
## 132.80592990 2.586706e+02 0.9496656 0.544493949  
## 132.80592990 2.386965e+03 0.9496656 0.544493949  
## 132.80592990 2.202647e+04 0.9496656 0.544493949  
## 403.42879349 4.539993e-05 0.9163666 0.000000000  
## 403.42879349 4.189421e-04 0.9163666 0.000000000  
## 403.42879349 3.865920e-03 0.9155569 -0.001557864  
## 403.42879349 3.567399e-02 0.9236805 0.446795521  
## 403.42879349 3.291930e-01 0.9378951 0.544135498  
## 403.42879349 3.037732e+00 0.9496656 0.544493949  
## 403.42879349 2.803162e+01 0.9496656 0.544493949  
## 403.42879349 2.586706e+02 0.9496656 0.544493949  
## 403.42879349 2.386965e+03 0.9496656 0.544493949  
## 403.42879349 2.202647e+04 0.9496656 0.544493949  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 22026.47 and C = 1.559623.

plot(svmr.fit, highlight = TRUE)



## QDA

#qda  
set.seed(1)  
  
ctrl1 = trainControl(method = "repeatedcv", repeats = 5,  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE)  
  
model.qda = train(x = train2[2:11],  
 y = train2$diabetes,  
 method = "qda",  
 metric = "ROC",  
 trControl = ctrl1)

## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.

#evaluate qda model  
qda.pred = predict(model.qda, newdata = test2, type = "prob")[,2]  
  
roc.qda = roc(test2$diabetes, qda.pred)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

qda.auc = roc.qda$auc[1]  
qda.auc

## [1] 0.8200662

plot(roc.qda, legacy.axes = TRUE)

