Instructions

BabyShop.xlsx contains binary data (1 = yes, 0 = no) on several variables on shoppers at a retail store. Other than Implied Gender: M=Male, F=Female, U=Unknown and Home Apt PO Box: A=Apt, H=Home, P=PO Box. Most of the variables are related to purchases. Our goal is to develop a logistic regression model to predict, given the data, the probability of a shopper being PREGNANT. By PREGNANT we mean that the shopper is themself pregnant or purchasing for someone who is. The company is interested in predicting the "pregnancy" of its shoppers so they might, to some degree of accuracy, send the households appropriate maternity related ads and coupons. They also want to minimize the mistake of sending maternity ads to nonpregnant households.

Part 1 – Training the Model

1. Split the data into a training set and test set.
2. Fit a logistic model to the training data. Print the summary and check if there are any variables that are insignificant in the model. Remove any of these variables and re-fit the model to the training data.
3. With your fitted training model, create an ROC curve displaying the sensitivity on the y-axis and 1-specificity on the x-axis. You may use the roc() function. Compute the area under the ROC curve. Using this value, comment on how the model seems to be performing given the test set.
    a. Bonus 1: Write your own code to produce an ROC curve, again with the sensitivity on the y-axis and 1-specificity on the x-axis. In doing this, use 200 threshold values evenly spaced between 0 and 1
    b. Bonus 2: Write your own code to compute the area under the curve from the ROC curve you created with your own code
4. Sensitivity – Specificity Cross Plot
    a. Create a simultaneous cross plot displaying the sensitivity, specificity, and also the positive predictive value and negative predictive value obtained for each threshold value from the roc() function output. Plot each using a different color.
    b. As mentioned, the company wants to avoid sending maternity ads to non-pregnant households. Lets say they would like to be at least 90% sure that they are not. Use the plot to estimate the minimum threshold value that would allow them to do this.
    c. From the roc() output, print a section of the threshold, sensitivity, and specificity values to show the exact value you would choose.
    d. Discussion: (in a paragraph no less than 5 sentences)
        i. What threshold value did you choose and why?
        ii. Overall, as the threshold values increase, how are the values of sensitivity, specificity, ppv, and npv changing? Does this seem to make sense, why?
5. Using your chosen threshold value from 4(b) above, classify (predict) the pregnancy/nonpregnancy of each individual in the test set.
    a. With the predicted values, create and display a confusion matrix.
    b. Compute the overall accuracy rate of the predictions, sensitivity, specificity, positive predictive value, and negative predictive value. Give the values and a statement for each interpreting them

```r
#Reading in file
#install.packages("readxl")
library(readxl)

## Warning: package 'readxl' was built under R version 4.3.3

BabyShop <- read_excel("BabyShop.xlsx")

#1: Training and Testing Set
set.seed(935)
s <- sample(c(1:nrow(BabyShop)), .8*nrow(BabyShop))
train <- BabyShop[s,]
test <- BabyShop[-s,]

#2: Logistic Model
log.train <- glm(PREGNANT ~ ., data = train, family = binomial)
summary(log.train)

##
## Call:
## glm(formula = PREGNANT ~ ., family = binomial, data = train)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -0.3777     0.2148  -1.758 0.078664 .
## Implied_GenderM          -0.3809     0.2335  -1.632 0.102763
## Implied_GenderU           0.1108     0.3863   0.287 0.774274
## Home_Apt__PO_BoxH        -0.1889     0.2299  -0.822 0.411293
## Home_Apt__PO_BoxP         0.1297     0.4238   0.306 0.759558
## Pregnancy_Test            2.8890     0.6371   4.535 5.77e-06 ***
## Birth_Control            -2.4850     0.4450  -5.584 2.34e-08 ***
## Feminine_Hygiene         -2.0973     0.4481  -4.681 2.86e-06 ***
## Folic_Acid                4.7461     1.0623   4.468 7.90e-06 ***
## Prenatal_Vitamins         2.7484     0.4629   5.937 2.90e-09 ***
## Prenatal_Yoga             2.8575     1.2554   2.276 0.022838 *
## Body_Pillow               2.0503     1.0372   1.977 0.048081 *
## Ginger_Ale                1.9496     0.5067   3.848 0.000119 ***
## Sea_Bands                 0.2049     0.8254   0.248 0.803939
## Stopped_buying_ciggies    1.5129     0.4155   3.641 0.000271 ***
## Cigarettes               -1.6471     0.4742  -3.473 0.000514 ***
## Smoking_Cessation         1.4300     0.5770   2.479 0.013193 *
## Stopped_buying_wine       1.4633     0.3893   3.758 0.000171 ***
## Wine                     -2.0320     0.4494  -4.522 6.13e-06 ***
## Maternity_Clothes         2.1547     0.3876   5.559 2.71e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1020.18  on 735  degrees of freedom
## Residual deviance:  529.36  on 716  degrees of freedom
```

```
## AIC: 569.36
##
## Number of Fisher Scoring iterations: 7
```

```
#Insignificant Variables Testing:
train2 <- train[,-c(2,11)]
test2 <- test[,-c(2,11)]
log.train2 <- glm(PREGNANT ~ ., data = train2, family = binomial)
summary(log.train2)
```

```
##
## Call:
## glm(formula = PREGNANT ~ ., family = binomial, data = train2)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -0.44291    0.17497  -2.531 0.011362 *
## Implied_GenderM         -0.39946    0.23164  -1.725 0.084615 .
## Implied_GenderU          0.07583    0.38419   0.197 0.843536
## Pregnancy_Test           2.90162    0.63563   4.565 5.00e-06 ***
## Birth_Control           -2.46199    0.44321  -5.555 2.78e-08 ***
## Feminine_Hygiene        -2.09772    0.44537  -4.710 2.48e-06 ***
## Folic_Acid               4.77194    1.06074   4.499 6.84e-06 ***
## Prenatal_Vitamins        2.77376    0.45812   6.055 1.41e-09 ***
## Prenatal_Yoga            2.79800    1.23943   2.257 0.023977 *
## Body_Pillow              1.96294    1.03040   1.905 0.056778 .
## Ginger_Ale               1.92174    0.50764   3.786 0.000153 ***
## Stopped_buying_ciggies   1.52293    0.41299   3.688 0.000226 ***
## Cigarettes              -1.67554    0.47226  -3.548 0.000388 ***
## Smoking_Cessation        1.45001    0.57220   2.534 0.011274 *
## Stopped_buying_wine      1.44263    0.38896   3.709 0.000208 ***
## Wine                    -2.05039    0.44944  -4.562 5.06e-06 ***
## Maternity_Clothes        2.14614    0.38507   5.573 2.50e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1020.18  on 735  degrees of freedom
## Residual deviance:  530.46  on 719  degrees of freedom
## AIC: 564.46
##
## Number of Fisher Scoring iterations: 7
```

```
#2 cont'd:  Further Testing Insignificant variables
library(car)
```

```
## Loading required package: carData
```

```
vif(log.train)
```

```
##                            GVIF Df GVIF^(1/(2*Df))
## Implied_Gender         1.082302  2        1.019969
## Home_Apt__PO_Box       1.082692  2        1.020061
## Pregnancy_Test         1.065462  1        1.032212
## Birth_Control          1.077396  1        1.037977
## Feminine_Hygiene       1.161295  1        1.077634
## Folic_Acid             1.026576  1        1.013201
## Prenatal_Vitamins      1.111239  1        1.054153
## Prenatal_Yoga          1.046696  1        1.023082
## Body_Pillow            1.048176  1        1.023805
## Ginger_Ale             1.041641  1        1.020608
## Sea_Bands              1.051566  1        1.025459
## Stopped_buying_ciggies 1.038059  1        1.018852
## Cigarettes             1.094802  1        1.046328
## Smoking_Cessation      1.041428  1        1.020504
## Stopped_buying_wine    1.024527  1        1.012189
## Wine                   1.057940  1        1.028562
## Maternity_Clothes      1.121426  1        1.058974
```

```
train3 <- train[,-c(1,2,11)]
test3 <- test[,-c(1,2,11)]
log.train3 <-  glm(PREGNANT ~ ., data = train3, family = binomial)
summary(log.train3)
```

```
##
## Call:
## glm(formula = PREGNANT ~ ., family = binomial, data = train3)
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -0.5945     0.1395  -4.261 2.04e-05 ***
## Pregnancy_Test           2.8909     0.6360   4.546 5.47e-06 ***
## Birth_Control           -2.4542     0.4408  -5.568 2.58e-08 ***
## Feminine_Hygiene        -2.1587     0.4431  -4.872 1.11e-06 ***
## Folic_Acid               4.7057     1.0557   4.457 8.30e-06 ***
## Prenatal_Vitamins        2.7734     0.4497   6.168 6.93e-10 ***
## Prenatal_Yoga            2.5680     1.2268   2.093 0.036329 *
## Body_Pillow              1.9766     1.0253   1.928 0.053883 .
## Ginger_Ale               1.9325     0.5093   3.794 0.000148 ***
## Stopped_buying_ciggies   1.5358     0.4106   3.740 0.000184 ***
## Cigarettes              -1.6870     0.4722  -3.572 0.000354 ***
## Smoking_Cessation        1.4285     0.5681   2.515 0.011910 *
## Stopped_buying_wine      1.4845     0.3874   3.832 0.000127 ***
## Wine                    -2.0470     0.4516  -4.533 5.82e-06 ***
## Maternity_Clothes        2.1730     0.3855   5.637 1.73e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 1020.18  on 735  degrees of freedom
## Residual deviance:  533.92  on 721  degrees of freedom
## AIC: 563.92
##
## Number of Fisher Scoring iterations: 7
```

```
anova(log.train, log.train3, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: PREGNANT ~ Implied_Gender + Home_Apt__PO_Box + Pregnancy_Test +
##     Birth_Control + Feminine_Hygiene + Folic_Acid + Prenatal_Vitamins +
##     Prenatal_Yoga + Body_Pillow + Ginger_Ale + Sea_Bands +
Stopped_buying_ciggies +
##     Cigarettes + Smoking_Cessation + Stopped_buying_wine + Wine +
##     Maternity_Clothes
## Model 2: PREGNANT ~ Pregnancy_Test + Birth_Control + Feminine_Hygiene +
##     Folic_Acid + Prenatal_Vitamins + Prenatal_Yoga + Body_Pillow +
##     Ginger_Ale + Stopped_buying_ciggies + Cigarettes + Smoking_Cessation +
##     Stopped_buying_wine + Wine + Maternity_Clothes
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       716     529.36
## 2       721     533.92 -5  -4.5588   0.4721
```

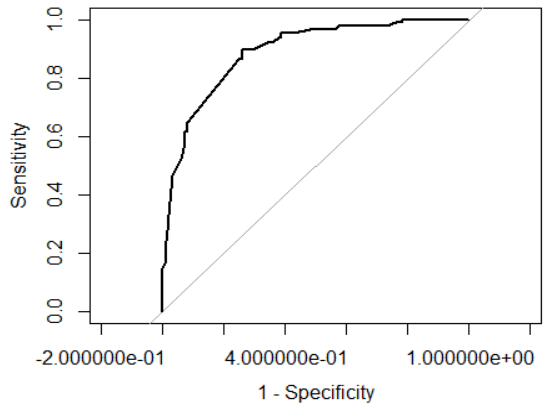
```
#I decided to use vif and anova test to determine if I should remove Implied
Gender, Home_Apt_Po_Box, and Sea Bands
#VIF: The VIF values are low, so multicollinearity is not a problem.
#ANOVA: The p value of 0.4721 >.05. This shows that removing the variables
does not worsen the model, and there is no significant differences.
#Conclusion: Since removing the variables does not hurt the model and makes
all variables significant, I decided to make a model without the variables
Implied Gender, Home_Apt_Po_Box, and Sea Bands.

#3:ROC model and AUC
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

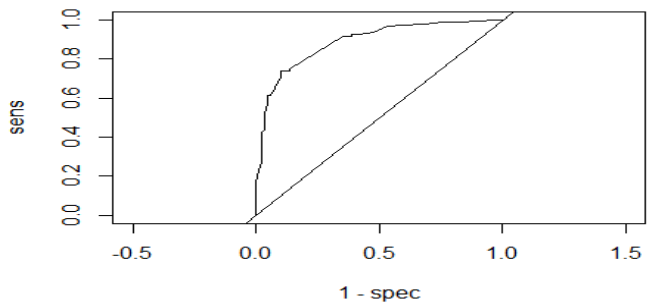
```
test.PREGNANT <- factor(test3$PREGNANT, levels = c("1", "0"))
probs.test <- predict(log.train3, newdata = test3, type = "response")
rocCurve <- roc(response = test.PREGNANT, predictor = probs.test)
```

```
## Setting levels: control = 1, case = 0

## Setting direction: controls > cases
```

```r
plot(rocCurve, legacy.axes = TRUE)
```




```r
auc(rocCurve)

## Area under the curve: 0.8885

##The 0.8885 auc is good for predicting; 1 is the highest it can be, so 0.8885 means the model is very good at predicting but not perfect. (TPR close to 1, FPR Low)

#Bonus 1: ROC Curve
#setting up vectors for loop
thresh <- seq(from = 0, to = 1, by=.005) #I did 1/200=.005
n <- length(thresh)
sens <- numeric(n)
spec <- numeric(n)

#loop for sens and spec values
for(i in 1:n){
  pred.test_bonus <- ifelse(probs.test >= thresh[i], 1, 0)
  pred.test_bonus <- factor(pred.test_bonus, levels = c("1", "0"))
  t <- table(pred.test_bonus, test.PREGNANT)
  sens[i] <- t[1,1]/sum(t[1,1]+t[2,1])
  spec[i] <- t[2,2]/sum(t[1,2]+t[2,2])

}

## Roc Curve by hand:
par(mfrow=c(1,1))
plot(1-spec, sens, ylim=c(0,1), xlim=c(-.5,1.5), type="l")
abline(a=0, b=1)
```




```r
#Bonus 2:
#trapezoidal rule
-sum(diff(1-spec) * ((sens[-1] + sens[-length(sens)]) / 2))
```

```
## [1] 0.8876065
```

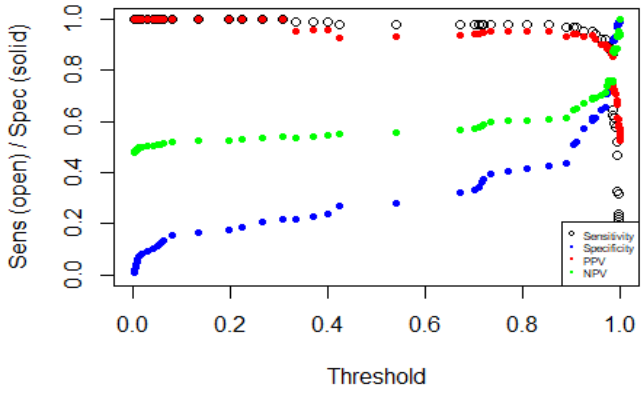
```r
#0.8876065 is similar to 0.8885, but a little off since trapezoidal rule is
more an an estimate. Also, I had to take the negative of the sum. Overall,
the auc is close to 1 so it has good prediction power/ high TPR and low FPR.

#4a: Simultaneous Sens, Spec, PPV, and NPV Crossplot

library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
test.PREGNANT <- factor(test3$PREGNANT, levels = c("1", "0")) #had to set as
factor, was not recognizing PREGNANT as one

#vectors for loop
thresholds <- rocCurve$thresholds
ppv <- numeric(length(thresholds))
npv <- numeric(length(thresholds))

#loop to get ppv and npv values
for (i in seq_along(thresholds)) {
  t <- thresholds[i]
  pred.test <- ifelse(probs.test > t, 1, 0)
  pred.test <- factor(pred.test, levels = c("1", "0"))
  cm <- confusionMatrix(data = pred.test, reference = test.PREGNANT, positive
= "1")
  ppv[i] <- cm$byClass['Pos Pred Value']
  npv[i] <- cm$byClass['Neg Pred Value']
}

#Plot with sens, spec, ppv, and npv
plot(rev(thresholds), rocCurve$sensitivities,
```

```
     ylab = "Sens (open) / Spec (solid)", xlab = "Threshold")
points(rev(thresholds), rocCurve$specificities, col = "blue", pch = 20)
points(rev(thresholds), ppv, col = "red", pch = 20)
points(rev(thresholds), npv, col = "green", pch = 20)
legend("bottomright", legend = c("Sensitivity", "Specificity", "PPV",
"NPV"),col = c("black", "blue", "red", "green"), pch = c(1,20,20,20), cex =
0.5) #had to change pch for sensitivity on the legend so it looked open; also
cex to make legend smaller because it covered graph too much
```




```
#4b: Plot estimation
#using plot: specificity around .95 threshold for .9 spec

#4c: finding actual with subset
threshold_specific <- data.frame(threshold =
rev(rocCurve$thresholds),sensitivity = rocCurve$sensitivities, specificity =
rocCurve$specificities)
print(subset(threshold_specific, specificity >= 0.9))
```

```
##    threshold sensitivity specificity
## 50 0.9859469  0.64772727   0.9166667
## 51 0.9867680  0.62500000   0.9166667
## 52 0.9877573  0.61363636   0.9166667
## 53 0.9884539  0.61363636   0.9270833
## 54 0.9889550  0.59090909   0.9270833
## 55 0.9912556  0.57954545   0.9270833
## 56 0.9935169  0.52272727   0.9375000
## 57 0.9944356  0.46590909   0.9687500
## 58 0.9957077  0.32954545   0.9791667
## 59 0.9963927  0.31818182   0.9791667
## 60 0.9971989  0.23863636   0.9895833
## 61 0.9980254  0.22727273   0.9895833
## 62 0.9986145  0.21590909   0.9895833
## 63 0.9991460  0.19318182   0.9895833
## 64 0.9992185  0.17045455   0.9895833
## 65 0.9993492  0.14772727   1.0000000
```

```
## 66 0.9995327  0.11363636   1.0000000
## 67 0.9997646  0.05681818   1.0000000
## 68 0.9999501  0.03409091   1.0000000
## 69 0.9999744  0.02272727   1.0000000
## 70       Inf  0.00000000   1.0000000

#actual minimum threshold: 0.9859469
```

##Discussion: (in a paragraph no less than 5 sentences) (i) What threshold value did you choose and why? (ii) Overall, as the threshold values increase, how are the values of sensitivity, specificity, ppv, and npv changing? Does this seem to make sense, why?

##I chose 0.9859469 because this is the minimum threshold value makes it at least 90% certain that non-pregnant people are not getting pregnancy ads. Also, this threshold value has the "happy medium" where sensitivity and specificity as well as positive predicitve value and negative predicitve value cross. As threshold values increase, sensitivity and ppv decrease while specificity and npv increase. The sensitivity and ppv decreasing makes sense because increasing the threshold makes the model more restrictive on what will be a positive(1/pregnant) classification. The specificity and npv increasing makes sense because increasing the threshold makes the model more likely to classify a value as negative(0/not pregnant) since it is restrictive with what can be posiitve.

```
#5a: Predict/Confusion Matrix
pred.test2 <- ifelse(probs.test > 0.9859469, 1, 0)
pred.test2 <- factor(pred.test2, levels = c("1", "0"))
c <- confusionMatrix(data=pred.test2, test.PREGNANT, positive = "1")
c

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  0
##          1 21  1
##          0 75 87
##
##                Accuracy : 0.587
##                  95% CI : (0.5122, 0.6589)
##     No Information Rate : 0.5217
##     P-Value [Acc > NIR] : 0.0445
##
##                   Kappa : 0.2004
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.2188
##             Specificity : 0.9886
##          Pos Pred Value : 0.9545
##          Neg Pred Value : 0.5370
##              Prevalence : 0.5217
```

```
##          Detection Rate : 0.1141
##    Detection Prevalence : 0.1196
##       Balanced Accuracy : 0.6037
##
##        'Positive' Class : 1
##
```

```
#5b: Values and Interpretations
#Accuracy: The overall correctness of the model in predicting if people were
pregnant(1) or not pregnant(0) was .587. 58.7% of the test set had the
prediction of pregnant or not pregnant match the actual observation.
mean(pred.test2 == test3$PREGNANT)  #Accuracy=(21+87)/(21+87+1+75)
```

```
## [1] 0.5869565
```

```
#Sensitivity: 21.89% of those that were observed as being pregnant were
predicted as being pregnant. The low sensitivity comes from the fact that the
company wants to be certain they do not send pregnancy coupons to non
pregnant people so less were classified as pregnant(1).
c$byClass[[1]] #Sensitivity=21/(21+75)
```

```
## [1] 0.21875
```

```
#Specificity: 98.86% of those that were observed as being not-pregnant were
predicted as being not pregnant. The model was good at predicting non-
pregnant observations due to the higher threshold which leads to high
specificity.
c$byClass[[2]] # Specificity=87/(87+1)
```

```
## [1] 0.9886364
```

```
#Positive Predictive Value: 95.45% of those predicted to be pregnant were
actually observed as being pregnant. This high value shows that the company
achieved their goal of having high certainty that who they predict to be
pregnant and send coupons to is actually pregnant.
c$byClass[[3]] #PPV=21/(21+1)
```

```
## [1] 0.9545455
```

```
#Negative predictive value: 53.70 % of those predicted to be not-pregnant
were actually observed as being not-pregnant. This lower percentage makes
sense because the company was trying to be cautious/certain so some observed
pregnant people were predicted to be not pregnant.
c$byClass[[4]] #NPV=87/(87+75)
```

```
## [1] 0.537037
```

Part 2 – Investigating the Predictive Ability of the Model

1. Using your threshold value that you chose from Part 1(4), use 10-fold cross validation on the entire data set to compute a set of estimates of the accuracy, sensitivity, specificity, ppv, and npv. In this,
    a. Compute K = 10 values of the accuracy, sensitivity, specificity, ppv, and npv.
    b. Plot these values together in a set of simultaneous box plots.
    c. Compute summary measures for each set of values.
    d. Discussion: (in a paragraph 3-5 sentences) From (b) and (c), comment on the predictive performance of the model in reference to each: accuracy, sensitivity, specificity, ppv, and npv.
2. Looking back at at the roc() output and cross-plot from Part 1(4) above, aside from the threshold value that you chose, choose three other values close to this one (have at least one above and one below).
    a. Repeat problem 1 (a)-(c) above, for each of the three threshold values. Use the same folds as problem 1.
    b. Now, looking at the four total sets of box plots and summary measures, make a more informed decision on the optimal threshold value. Keep in mind, the company would like to err on the side of not sending maternity ads to households that are not pregnant around 90% of the time, but would like to have as high a chance as it can at sending ads to households that are pregnant.
    c. Discussion: (in a paragraph 3-5 sentences) Using what the plots and numbers show, comment on why you are recommending the threshold value you chose.

```r
#1a: k=10 values
library(caret)
#Removing the insignificant variables from part 1
BabyShop2 <- BabyShop[, -c(1, 2, 11)]
BabyShop2$PREGNANT <- factor(BabyShop2$PREGNANT, levels = c(0, 1), labels =
c("NonPregnant", "Pregnant")) #doing this in case it does not recognize as
factor again

#myControl: from DataCamp lesson
myControl <- trainControl(
  method = "cv",
  number = 10,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  savePredictions = "final",
  verboseIter = TRUE
)

#Train() with full data set/cross validation
cv.model <- train(PREGNANT ~ ., data = BabyShop2, method = "glm", family =
binomial(), trControl = myControl)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy"
was not
## in the result set. ROC will be used instead.

## + Fold01: parameter=none
## - Fold01: parameter=none
## + Fold02: parameter=none
## - Fold02: parameter=none
## + Fold03: parameter=none
## - Fold03: parameter=none
## + Fold04: parameter=none
## - Fold04: parameter=none
## + Fold05: parameter=none
## - Fold05: parameter=none
## + Fold06: parameter=none
## - Fold06: parameter=none
## + Fold07: parameter=none
## - Fold07: parameter=none
## + Fold08: parameter=none
## - Fold08: parameter=none
## + Fold09: parameter=none
## - Fold09: parameter=none
## + Fold10: parameter=none
## - Fold10: parameter=none
## Aggregating results
## Fitting final model on full training set
```

```r
#predictions and threshold
predictions <- cv.model$pred
threshold <- 0.9859469

#Converting probabilities to binary predictions
predictions$pred_class <- ifelse(predictions$Pregnant > threshold,
"Pregnant", "NonPregnant")
predictions$pred_class <- factor(predictions$pred_class, levels =
c("NonPregnant", "Pregnant"))

#function to get metrics and confusion matrix for each fold
results <- with(predictions, tapply(seq_along(Pregnant), Resample,
function(idx) {
  data <- predictions[idx, ]
  cm <- confusionMatrix(data = factor(data$pred_class, levels =
c("NonPregnant", "Pregnant")), reference = factor(data$obs, levels =
c("NonPregnant", "Pregnant")), positive = "Pregnant")
c(Accuracy = cm$overall['Accuracy'],
    Sensitivity = cm$byClass['Sensitivity'],
    Specificity = cm$byClass['Specificity'],
    PPV = cm$byClass['Pos Pred Value'],
    NPV = cm$byClass['Neg Pred Value'])
}))

#Turning into data frame for part 1b
library(dplyr)
```

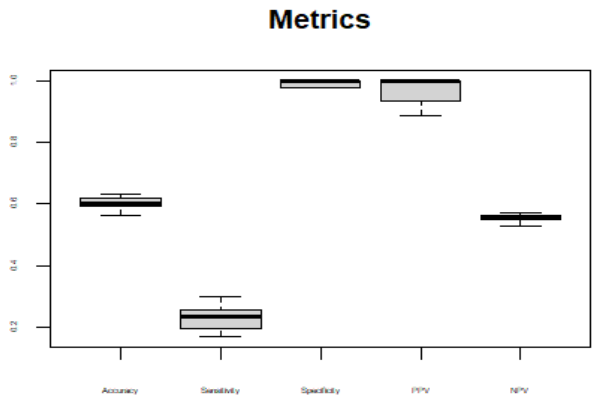
```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:car':
##
##     recode

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
results_df <- bind_rows(results, .id = "Fold")
print(results_df)
```

```
## # A tibble: 10 × 6
##    Fold  Accuracy.Accuracy Sensitivity.Sensitivity Specificity.Specificity
##    <chr>             <dbl>                   <dbl>                   <dbl>
##  1 1                 0.581                   0.170                   1
##  2 2                 0.593                   0.196                   1
##  3 3                 0.609                   0.234                   1
```

```
##  4 4                 0.598                     0.213                     1
##  5 5                 0.630                     0.277                     1
##  6 6                 0.598                     0.234                     0.978
##  7 7                 0.565                     0.170                     0.978
##  8 8                 0.630                     0.298                     0.978
##  9 9                 0.620                     0.255                     1
## 10 10                0.620                     0.255                     1
## # i 2 more variables: `PPV.Pos Pred Value` <dbl>, `NPV.Neg Pred Value`
<dbl>
```

```r
#1b
#Setting columns up as vectors
accuracy <- results_df$Accuracy.Accuracy
sensitivity <- results_df$Sensitivity.Sensitivity
specificity <- results_df$Specificity.Specificity
ppv <- results_df$`PPV.Pos Pred Value`
npv <- results_df$`NPV.Neg Pred Value`

#simultaneous box plots
boxplot(accuracy, sensitivity, specificity, ppv, npv,
        names = c("Accuracy","Sensitivity", "Specificity", "PPV", "NPV"),
        cex.axis = 0.4,
        main = "Metrics")
```




```r
#1c:Summary Measures
#applying summary measures to all columns of dataframe
summary_stats <- sapply(results_df, function(x) {
  c(Mean = mean(x, na.rm = TRUE),
    Median = median(x, na.rm = TRUE),
    SD = sd(x, na.rm = TRUE),
    Min = min(x, na.rm = TRUE),
    Max = max(x, na.rm = TRUE))
})
```

```
## Warning in mean.default(x, na.rm = TRUE): argument is not numeric or
logical:
## returning NA

## Warning in mean.default(sort(x, partial = half + 0L:1L)[half + 0L:1L]):
## argument is not numeric or logical: returning NA

summary_stats <- as.data.frame(t(summary_stats))
summary_stats

##                                      Mean            Median
SD
## Fold                                 <NA>              <NA>
3.02765035409749
## Accuracy.Accuracy       0.604361697208822 0.603260869565217
0.0214498957387521
## Sensitivity.Sensitivity 0.230203515263645 0.234042553191489
0.0430993928797612
## Specificity.Specificity 0.993333333333333                 1
0.0107343531453255
## PPV.Pos Pred Value      0.973888888888889                 1
0.0433546955891704
## NPV.Neg Pred Value      0.554046230827439 0.552777777777778
0.0129091126134787
##                                      Min               Max
## Fold                                   1                 9
## Accuracy.Accuracy       0.565217391304348 0.630434782608696
## Sensitivity.Sensitivity 0.170212765957447 0.297872340425532
## Specificity.Specificity 0.977777777777778                 1
## PPV.Pos Pred Value      0.888888888888889                 1
## NPV.Neg Pred Value      0.530120481927711 0.571428571428571
```

##Discussion: (in a paragraph 3-5 sentences) From (b) and (c), comment on the predictive performance of the model in reference to each: accuracy, sensitivity, specif icity,ppv, and npv

##The values are similar to those found in part one, so the threshold performed well on the full dataset in accordance with prediction of pregnant vs non pregnant. The accuracy boxplot is distributred around a median of .6, sensitivity is once again low and distributed around .2, specificity is high and very close to one, ppv is also high with the distribution staying around .9-1, and npv is centered around .55 in the distribiution. The sensitivity is low due to the high threshold, while specificity is high since more values will be classified as not pregnant. PPV is high, which means that given someone of predicted pregant there is a high chance they actually are and will be sent the right coupons.

```
#2a: value: 0.9851035(below)
#a
library(caret)
#Removing the insignificant variables from part 1
BabyShop2 <- BabyShop[, -c(1, 2, 11)]
```

```r
BabyShop2$PREGNANT <- factor(BabyShop2$PREGNANT, levels = c(0, 1), labels =
c("NonPregnant", "Pregnant"))

#myControl: from DataCamp lesson
myControl <- trainControl(
  method = "cv",
  number = 10,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  savePredictions = "final",
  verboseIter = TRUE
)

#Train() with full data set/cross validation
cv.model <- train(PREGNANT ~ ., data = BabyShop2, method = "glm", family =
binomial(), trControl = myControl)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy"
was not
## in the result set. ROC will be used instead.

## + Fold01: parameter=none
## - Fold01: parameter=none
## + Fold02: parameter=none
## - Fold02: parameter=none
## + Fold03: parameter=none
## - Fold03: parameter=none
## + Fold04: parameter=none
## - Fold04: parameter=none
## + Fold05: parameter=none
## - Fold05: parameter=none
## + Fold06: parameter=none
## - Fold06: parameter=none
## + Fold07: parameter=none
## - Fold07: parameter=none
## + Fold08: parameter=none
## - Fold08: parameter=none
## + Fold09: parameter=none
## - Fold09: parameter=none
## + Fold10: parameter=none
## - Fold10: parameter=none
## Aggregating results
## Fitting final model on full training set
```

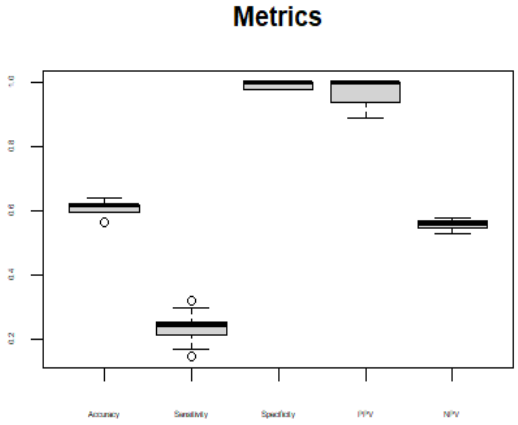
```r
#predictions and threshold
predictions <- cv.model$pred
threshold <- 0.9851035

#Converting probabilities to binary predictions
predictions$pred_class <- ifelse(predictions$Pregnant > threshold,
```

```r
"Pregnant", "NonPregnant")
predictions$pred_class <- factor(predictions$pred_class, levels =
c("NonPregnant", "Pregnant"))

#function to get metrics and confusion matrix for each fold
results <- with(predictions, tapply(seq_along(Pregnant), Resample,
function(idx) {
  data <- predictions[idx, ]
  cm <- confusionMatrix(data = factor(data$pred_class, levels =
c("NonPregnant", "Pregnant")),
                        reference = factor(data$obs, levels =
c("NonPregnant", "Pregnant")),
                        positive = "Pregnant")
  c(Accuracy = cm$overall['Accuracy'],
    Sensitivity = cm$byClass['Sensitivity'],
    Specificity = cm$byClass['Specificity'],
    PPV = cm$byClass['Pos Pred Value'],
    NPV = cm$byClass['Neg Pred Value'])
}))

#Turning into data frame for part 1b
library(dplyr)
results_df <- bind_rows(results, .id = "Fold")
print(results_df)
```

```
## # A tibble: 10 × 6
##    Fold  Accuracy.Accuracy Sensitivity.Sensitivity Specificity.Specificity
##    <chr>             <dbl>                   <dbl>                   <dbl>
##  1 1                 0.565                   0.170                   0.978
##  2 2                 0.630                   0.298                   0.978
##  3 3                 0.641                   0.319                   0.978
##  4 4                 0.620                   0.255                   1
##  5 5                 0.620                   0.255                   1
##  6 6                 0.565                   0.149                   1
##  7 7                 0.598                   0.213                   1
##  8 8                 0.598                   0.213                   1
##  9 9                 0.620                   0.255                   1
## 10 10                0.620                   0.239                   1
## # ℹ 2 more variables: `PPV.Pos Pred Value` <dbl>, `NPV.Neg Pred Value`
<dbl>
```

```r
#b
#Setting columns up as vectors
accuracy <- results_df$Accuracy.Accuracy
sensitivity <- results_df$Sensitivity.Sensitivity
specificity <- results_df$Specificity.Specificity
ppv <- results_df$`PPV.Pos Pred Value`
npv <- results_df$`NPV.Neg Pred Value`

#simultaneous box plots
```

```r
boxplot(accuracy, sensitivity, specificity, ppv, npv,
        names = c("Accuracy","Sensitivity", "Specificity", "PPV", "NPV"),
        cex.axis = 0.4,
        main = "Metrics")
```




```r
#c
#applying summary measures to all columns of dataframe
summary_stats <- sapply(results_df, function(x) {
  c(Mean = mean(x, na.rm = TRUE),
    Median = median(x, na.rm = TRUE),
    SD = sd(x, na.rm = TRUE),
    Min = min(x, na.rm = TRUE),
    Max = max(x, na.rm = TRUE))
})
```

```
## Warning in mean.default(x, na.rm = TRUE): argument is not numeric or
logical:
## returning NA
```

```
## Warning in mean.default(sort(x, partial = half + 0L:1L)[half + 0L:1L]):
## argument is not numeric or logical: returning NA
```

```r
summary_stats <- as.data.frame(t(summary_stats))
summary_stats
```

```
##                                      Mean            Median
SD
## Fold                                <NA>              <NA>
3.02765035409749
## Accuracy.Accuracy       0.607608695652174 0.619565217391304
0.0258493608430971
## Sensitivity.Sensitivity 0.236679000925069 0.247224791859389
0.0525510565800406
## Specificity.Specificity 0.993333333333333                 1
```

```
0.0107343531453255
## PPV.Pos Pred Value      0.975972222222222                 1
0.0407199547142941
## NPV.Neg Pred Value      0.556287039666087            0.5625
0.0167103242361638
##                                       Min               Max
## Fold                                    1                 9
## Accuracy.Accuracy       0.565217391304348 0.641304347826087
## Sensitivity.Sensitivity 0.148936170212766 0.319148936170213
## Specificity.Specificity 0.977777777777778                 1
## PPV.Pos Pred Value      0.888888888888889                 1
## NPV.Neg Pred Value      0.529411764705882 0.578947368421053
```

```
#2a: value:0.9877573
#a
library(caret)
#Removing the insignificant variables from part 1
BabyShop2 <- BabyShop[, -c(1, 2, 11)]
BabyShop2$PREGNANT <- factor(BabyShop2$PREGNANT, levels = c(0, 1), labels =
c("NonPregnant", "Pregnant"))

#myControl: from DataCamp lesson
myControl <- trainControl(
  method = "cv",
  number = 10,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  savePredictions = "final",  # Ensure predictions are saved
  verboseIter = TRUE
)

#Train() with full data set/cross validation
cv.model <- train(PREGNANT ~ ., data = BabyShop2, method = "glm", family =
binomial(), trControl = myControl)
```

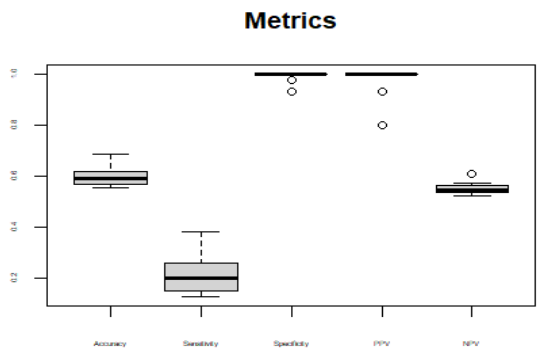
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy"
was not
## in the result set. ROC will be used instead.
```

```
## + Fold01: parameter=none
## - Fold01: parameter=none
## + Fold02: parameter=none
## - Fold02: parameter=none
## + Fold03: parameter=none
## - Fold03: parameter=none
## + Fold04: parameter=none
## - Fold04: parameter=none
## + Fold05: parameter=none
## - Fold05: parameter=none
## + Fold06: parameter=none
```

```
## - Fold06: parameter=none
## + Fold07: parameter=none
## - Fold07: parameter=none
## + Fold08: parameter=none
## - Fold08: parameter=none
## + Fold09: parameter=none
## - Fold09: parameter=none
## + Fold10: parameter=none
## - Fold10: parameter=none
## Aggregating results
## Fitting final model on full training set

#predictions and threshold
predictions <- cv.model$pred
threshold <- 0.9877573

#Converting probabilities to binary predictions
predictions$pred_class <- ifelse(predictions$Pregnant > threshold,
"Pregnant", "NonPregnant")
predictions$pred_class <- factor(predictions$pred_class, levels =
c("NonPregnant", "Pregnant"))

#function to get metrics and confusion matrix for each fold
results <- with(predictions, tapply(seq_along(Pregnant), Resample,
function(idx) {
  data <- predictions[idx, ]
  cm <- confusionMatrix(data = factor(data$pred_class, levels =
c("NonPregnant", "Pregnant")),
                        reference = factor(data$obs, levels =
c("NonPregnant", "Pregnant")),
                        positive = "Pregnant")
  c(Accuracy = cm$overall['Accuracy'],
    Sensitivity = cm$byClass['Sensitivity'],
    Specificity = cm$byClass['Specificity'],
    PPV = cm$byClass['Pos Pred Value'],
    NPV = cm$byClass['Neg Pred Value'])
}))

#Turning into data frame for part 1b
library(dplyr)
results_df <- bind_rows(results, .id = "Fold")
print(results_df)

## # A tibble: 10 × 6
##    Fold  Accuracy.Accuracy Sensitivity.Sensitivity Specificity.Specificity
##    <chr>             <dbl>                  <dbl>                   <dbl>
##  1 1                 0.587                  0.191                       1
##  2 2                 0.570                  0.149                       1
##  3 3                 0.554                  0.128                       1
##  4 4                 0.554                  0.128                       1
```

```
##  5 5                  0.685                     0.383                     1
##  6 6                  0.593                     0.261                     0.933
##  7 7                  0.587                     0.191                     1
##  8 8                  0.598                     0.213                     1
##  9 9                  0.630                     0.298                     0.978
## 10 10                 0.620                     0.255                     1
## # i 2 more variables: `PPV.Pos Pred Value` <dbl>, `NPV.Neg Pred Value`
<dbl>
```

```r
#b
#Setting columns up as vectors
accuracy <- results_df$Accuracy.Accuracy
sensitivity <- results_df$Sensitivity.Sensitivity
specificity <- results_df$Specificity.Specificity
ppv <- results_df$`PPV.Pos Pred Value`
npv <- results_df$`NPV.Neg Pred Value`

#simultaneous box plots
boxplot(accuracy, sensitivity, specificity, ppv, npv,
        names = c("Accuracy","Sensitivity", "Specificity", "PPV", "NPV"),
        cex.axis = 0.4,
        main = "Metrics")
```




```r
#c
#applying summary measures to all columns of dataframe
summary_stats <- sapply(results_df, function(x) {
  c(Mean = mean(x, na.rm = TRUE),
    Median = median(x, na.rm = TRUE),
    SD = sd(x, na.rm = TRUE),
    Min = min(x, na.rm = TRUE),
    Max = max(x, na.rm = TRUE))
})
```

```
## Warning in mean.default(x, na.rm = TRUE): argument is not numeric or
logical:
## returning NA
```

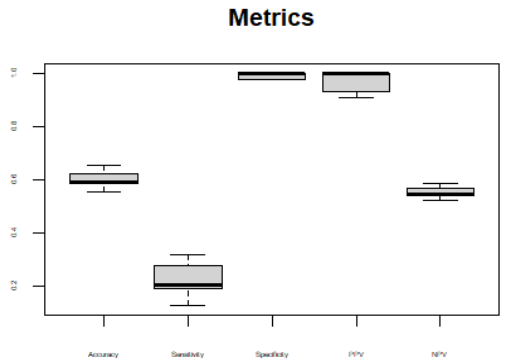
```
## Warning in mean.default(sort(x, partial = half + 0L:1L)[half + 0L:1L]):
## argument is not numeric or logical: returning NA

summary_stats <- as.data.frame(t(summary_stats))
summary_stats

##                                     Mean            Median
SD
## Fold                                <NA>              <NA>
3.02765035409749
## Accuracy.Accuracy       0.597851645782922 0.590181557572862
0.0393194592797178
## Sensitivity.Sensitivity 0.219703977798335 0.202127659574468
0.08118885643284
## Specificity.Specificity 0.991111111111111                 1
0.021468706290651
## PPV.Pos Pred Value      0.973333333333333                 1
0.064406118871953
## NPV.Neg Pred Value      0.550918144452373 0.545474581251837
0.0253281411482071
##                                      Min               Max
## Fold                                   1                 9
## Accuracy.Accuracy       0.554347826086957 0.684782608695652
## Sensitivity.Sensitivity 0.127659574468085 0.382978723404255
## Specificity.Specificity 0.933333333333333                 1
## PPV.Pos Pred Value                    0.8                 1
## NPV.Neg Pred Value      0.523255813953488 0.608108108108108

#2a:0.9884539 (above)
#a
library(caret)
#Removing the insignificant variables from part 1
BabyShop2 <- BabyShop[, -c(1, 2, 11)]
BabyShop2$PREGNANT <- factor(BabyShop2$PREGNANT, levels = c(0, 1), labels =
c("NonPregnant", "Pregnant"))

#my Control: from Data Camp lesson
myControl <- trainControl(
  method = "cv",
  number = 10,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  savePredictions = "final",  # Ensure predictions are saved
  verboseIter = TRUE
)

#Train() with full data set/cross validation
cv.model <- train(PREGNANT ~ ., data = BabyShop2, method = "glm", family =
binomial(), trControl = myControl)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy"
was not
## in the result set. ROC will be used instead.

## + Fold01: parameter=none
## - Fold01: parameter=none
## + Fold02: parameter=none
## - Fold02: parameter=none
## + Fold03: parameter=none
## - Fold03: parameter=none
## + Fold04: parameter=none
## - Fold04: parameter=none
## + Fold05: parameter=none
## - Fold05: parameter=none
## + Fold06: parameter=none
## - Fold06: parameter=none
## + Fold07: parameter=none
## - Fold07: parameter=none
## + Fold08: parameter=none
## - Fold08: parameter=none
## + Fold09: parameter=none
## - Fold09: parameter=none
## + Fold10: parameter=none
## - Fold10: parameter=none
## Aggregating results
## Fitting final model on full training set

#predictions and threshold
predictions <- cv.model$pred
threshold <- 0.9884539

#Converting probabilities to binary predictions
predictions$pred_class <- ifelse(predictions$Pregnant > threshold,
"Pregnant", "NonPregnant")
predictions$pred_class <- factor(predictions$pred_class, levels =
c("NonPregnant", "Pregnant"))

#function to get metrics and confusion matrix for each fold
results <- with(predictions, tapply(seq_along(Pregnant), Resample,
function(idx) {
  data <- predictions[idx, ]
  cm <- confusionMatrix(data = factor(data$pred_class, levels =
c("NonPregnant", "Pregnant")),
                        reference = factor(data$obs, levels =
c("NonPregnant", "Pregnant")),
                        positive = "Pregnant")
  c(Accuracy = cm$overall['Accuracy'],
    Sensitivity = cm$byClass['Sensitivity'],
    Specificity = cm$byClass['Specificity'],
    PPV = cm$byClass['Pos Pred Value'],
```

```
    NPV = cm$byClass['Neg Pred Value'])
}))

#Turning into data frame for part 1b
library(dplyr)
results_df <- bind_rows(results, .id = "Fold")
print(results_df)
```

```
## # A tibble: 10 × 6
##    Fold  Accuracy.Accuracy Sensitivity.Sensitivity Specificity.Specificity
##    <chr>             <dbl>                   <dbl>                   <dbl>
##  1 1                 0.620                   0.277                   0.978
##  2 2                 0.652                   0.319                   1
##  3 3                 0.630                   0.298                   0.978
##  4 4                 0.624                   0.255                   1
##  5 5                 0.593                   0.217                   0.978
##  6 6                 0.587                   0.191                   1
##  7 7                 0.554                   0.128                   1
##  8 8                 0.554                   0.128                   1
##  9 9                 0.587                   0.191                   1
## 10 10                0.587                   0.191                   1
## # ℹ 2 more variables: `PPV.Pos Pred Value` <dbl>, `NPV.Neg Pred Value`
<dbl>
```

```
#b
#Setting columns up as vectors
accuracy <- results_df$Accuracy.Accuracy
sensitivity <- results_df$Sensitivity.Sensitivity
specificity <- results_df$Specificity.Specificity
ppv <- results_df$`PPV.Pos Pred Value`
npv <- results_df$`NPV.Neg Pred Value`

#simultaneous box plots
boxplot(accuracy, sensitivity, specificity, ppv, npv,
        names = c("Accuracy","Sensitivity", "Specificity", "PPV", "NPV"),
        cex.axis = 0.4,
        main = "Metrics")
```




```r
#c
#applying summary measures to all columns of dataframe
summary_stats <- sapply(results_df, function(x) {
  c(Mean = mean(x, na.rm = TRUE),
    Median = median(x, na.rm = TRUE),
    SD = sd(x, na.rm = TRUE),
    Min = min(x, na.rm = TRUE),
    Max = max(x, na.rm = TRUE))
})
```

```
## Warning in mean.default(x, na.rm = TRUE): argument is not numeric or
logical:
## returning NA
```

```
## Warning in mean.default(sort(x, partial = half + 0L:1L)[half + 0L:1L]):
## argument is not numeric or logical: returning NA
```

```r
summary_stats <- as.data.frame(t(summary_stats))
summary_stats
```

```
##                                      Mean            Median
SD
## Fold                                <NA>              <NA>
3.02765035409749
## Accuracy.Accuracy       0.598880163781987 0.590181557572862
0.0321461330191899
## Sensitivity.Sensitivity 0.219611470860315 0.204440333024977
0.066535293184193
## Specificity.Specificity 0.993333333333333                 1
0.0107343531453255
## PPV.Pos Pred Value      0.977099567099567                 1
0.0373670231924949
## NPV.Neg Pred Value      0.551086560651798 0.546084337349398
0.0204549326826289
##                                      Min               Max
## Fold                                   1                 9
## Accuracy.Accuracy       0.554347826086957 0.652173913043478
```

```
## Sensitivity.Sensitivity 0.127659574468085 0.319148936170213
## Specificity.Specificity 0.977777777777778                 1
## PPV.Pos Pred Value      0.909090909090909                 1
## NPV.Neg Pred Value      0.523255813953488 0.584415584415584

#2b: threshold choice = 0.9859469
```

##Discussion: (in a paragraph 3-5 sentences) Using what the plots and numbers show,comment on why you are recommending the threshold value you chose.

##I chose 0.9859469 because it has the highest mean specificity and mean ppv while also being the lowest to achieve the miminum specificity required in part 1. Also, the lower sensitivity matches the information in part 1. Finally, the boxplot distributions for ppv and specificity stay close to the top so there is not a large amount of variation. Accuracy and npv also have distributions that do not vary widely. Other thresholds have higher accuracy, but this is not necesarily important as the company is trying to be certain when classyfying pregnant people correctly. ##