Module 4 Assignment | R

Ryan S Dunn

11/17/2021

Data Science Using Python and R: Chapter 13 - Page 195: Questions #13, 14, 15, 16, & 17

13. Create a logistic regression model to predict whether or not a customer has a store credit card, based on whether they have a web account and the days between purchases. Obtain the summary of the model.

```
#import the clothing sales training
clothing_test <- read.csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied
Data Mining 502/Module 4/Datasets/clothing_sales_test.csv")

#import the clothing sales test
clothing_train <- read.csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied
Data Mining 502/Module 4/Datasets/clothing_sales_training.csv")

#subset the training set to only return the columns needed for the model
clothing_train <- clothing_train[c("Days","Web", "CC")]
clothing_test <- clothing_test[c("Days","Web", "CC")]

#feature scaling
clothing_train[ ,1] <- scale(clothing_train[ ,1])
clothing_test[ ,1] <- scale(clothing_test[ ,1])

#head(clothing_train)

#build logistic regression model
classifier = glm(formula = CC ~ .,
                 family = binomial,
                 data = clothing_train)

#predict test set results with the probabilities in a vector
probability_pred <- predict(classifier, type = 'response', clothing_test[-3])

#return the vector of 1's and 0's
y_pred <- ifelse(probability_pred > 0.5, 1, 0)

#create confusion matrix - real / predictions
cm <- table(clothing_test[,3], y_pred)
cm

##    y_pred
##       0   1
```

```
##   0 411 306
##   1 219 459
```

```
summary(classifier)
```

```
## 
## Call:
## glm(formula = CC ~ ., family = binomial, data = clothing_train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.9035  -1.1458  -0.6078   1.0895   2.1044  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -0.13136    0.05544  -2.369  0.01782 *  
## Days        -0.52384    0.06200  -8.449  < 2e-16 ***
## Web          1.25370    0.33067   3.791  0.00015 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 2009.9  on 1450  degrees of freedom
## Residual deviance: 1903.6  on 1448  degrees of freedom
## AIC: 1909.6
## 
## Number of Fisher Scoring iterations: 4
```

14. Are there any variables that should be removed from the model? If so, remove them and rerun the model

There should not be any variables removed from the model, as all p-values (Days: p-value <2e-16, Web: p-value 0.00015) point to each co-efficient being statistically significant and therefore relevant for the logistic regression model.

15. Write the descriptive form of the logistic regression model using the coefficients obtained from Question #13.

CC = -0.13 -0.12136(Days) + 1.2537(Web)

16. Validate the model using the test data set.

```
#predict test set results with the probabilities in a vector
probability_pred <- predict(classifier, type = 'response', clothing_test[-3])
```

17. Obtain the predicted values of the response variaable for each record in the data set.

```
#return the vector of 1's and 0's
y_pred <- ifelse(probability_pred > 0.5, 1, 0)

#create confusion matrix - real / predictions
```

```
cm <- table(clothing_test[,3], y_pred)
colnames(cm) <- c("Predicted 0", "Predicted 1")
row.names(cm) <- c("Actual 0", "Actual 1")
cm

##           y_pred
##             Predicted 0 Predicted 1
##   Actual 0          411         306
##   Actual 1          219         459
```

Data Science Using Python and R: Chapter 9 - Page 138: Questions #24, 25, 26, 27, 28, 29, & 30

```
#install the neural net tools packages
#install.packages("nnet")
#install.packages("NeuralNetTools")

library(NeuralNetTools)
library(nnet)

#import training and test data sets
bank_test <- read.csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied Data Mining 502/Module 4/Datasets/bank_marketing_test")

#import the test data set
bank_train <- read.csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied Data Mining 502/Module 4/Datasets/bank_marketing_training")
```

24. Prepare the data set for neural network modeling, including standardizing the variables.

```
# for the training data set:

#convert the binary and ordinal variables to factors for the ANN algorithm
bank_train$response <- as.factor(bank_train$response)
bank_train$education <- as.factor(bank_train$education)
bank_train$job <- as.factor(bank_train$job)
bank_train$marital <- as.factor(bank_train$marital)
bank_train$default <- as.factor(bank_train$default)
bank_train$housing <- as.factor(bank_train$housing)
bank_train$loan <- as.factor(bank_train$loan)
bank_train$contact <- as.factor(bank_train$contact)
bank_train$month <- as.factor(bank_train$month)
bank_train$day_of_week <- as.factor(bank_train$day_of_week)
bank_train$campaign <- as.factor(bank_train$campaign)
bank_train$days_since_previous <- as.factor(bank_train$days_since_previous)
bank_train$previous <- as.factor(bank_train$previous)
bank_train$previous_outcome <- as.factor(bank_train$previous_outcome)

#standardize the quantitative variables for feature scaling

bank_train$age.mm <- (bank_train$age - min(bank_train$age)) /
```

```
(max(bank_train$age) - min(bank_train$age))
bank_train$duration.mm <- (bank_train$duration - min(bank_train$duration)) /
(max(bank_train$duration) - min(bank_train$duration))

#create vector for the training actual responses
y_train_ann = bank_train$response

#convert the binary and ordinal variables to factors for the ANN algorithm
bank_test$response <- as.factor(bank_test$response)
bank_test$education <- as.factor(bank_test$education)
bank_test$job <- as.factor(bank_test$job)
bank_test$marital <- as.factor(bank_test$marital)
bank_test$default <- as.factor(bank_test$default)
bank_test$housing <- as.factor(bank_test$housing)
bank_test$loan <- as.factor(bank_test$loan)
bank_test$contact <- as.factor(bank_test$contact)
bank_test$month <- as.factor(bank_test$month)
bank_test$day_of_week <- as.factor(bank_test$day_of_week)
bank_test$campaign <- as.factor(bank_test$campaign)
bank_test$days_since_previous <- as.factor(bank_test$days_since_previous)
bank_test$previous <- as.factor(bank_test$previous)
bank_test$previous_outcome <- as.factor(bank_test$previous_outcome)

#standardize the quantitative variables for feature scaling

bank_test$age.mm <- (bank_test$age - min(bank_test$age)) /
(max(bank_test$age) - min(bank_test$age))
bank_test$duration.mm <- (bank_test$duration - min(bank_test$duration)) /
(max(bank_test$duration) - min(bank_test$duration))

#create vector for the test actual responses
y_test_ann <- bank_test$response
```

25. Using the training data set, create a neural network model to predict a customer's Response using whichever predictors you think appropriate. Obtain the predicted responses.

```
#full data set ANN with all variables and 5 hidden layers
#nnet_bank <- nnet(response ~ job + marital + education + default + housing +
loan + previous_outcome + age.mm + duration.mm, #data = bank_train, size = 5)

#ANN with less variables and smaller hidden layer
nnet_bank <- nnet(response ~  marital + education + previous_outcome + age.mm
+ duration.mm, data = bank_train, size = 1)

## # weights:  17
## initial  value 15838.503495
## iter  10 value 9119.505553
## iter  20 value 7909.378125
## iter  30 value 6803.919048
## iter  40 value 6607.278953
## iter  50 value 6565.651461
```

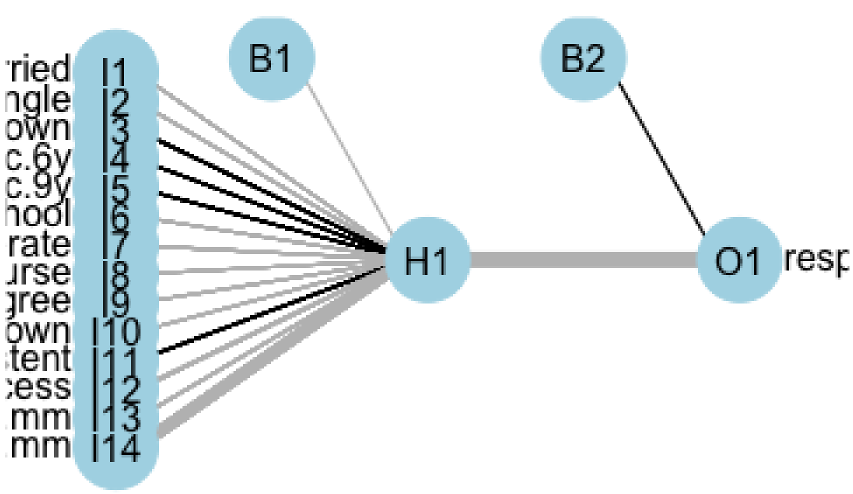
```
## iter  60 value 6556.393510
## iter  70 value 6551.576387
## iter  80 value 6544.851022
## iter  90 value 6543.284752
## iter 100 value 6543.247779
## final  value 6543.247779
## stopped after 100 iterations
```

```
#view the output of the ANN
nnet_bank
```

```
## a 14-1-1 network with 17 weights
## inputs: maritalmarried maritalsingle maritalunknown educationbasic.6y
educationbasic.9y educationhigh.school educationilliterate
educationprofessional.course educationuniversity.degree educationunknown
previous_outcomenonexistent previous_outcomesuccess age.mm duration.mm
## output(s): response
## options were - entropy fitting
```

26. Plot the neural network.

```
#plot the neural network
plotnet(nnet_bank)
```



```r
#obtain the weights
nnet_weight_array <- nnet_bank$wts
print(nnet_weight_array)
```

```
##  [1]  -0.71333899  -0.02401265  -0.26142170   0.44925341   0.08901240
##  [6]   0.12032944  -0.05731878  -0.42747145  -0.09932309  -0.22541408
## [11]  -0.16222320   0.31737613  -1.65275800  -0.63160166 -12.65503122
## [16]   0.96457214 -19.50792368
```

27. Evaluate the neural network model using the test data set. Construct a contingency table to compare the actual and predicted values of Response.

```r
#predict.nnet(object = nnet_bank, newdata = y_test)
cm_ann <- table(bank_test$response, predict(nnet_bank, type = "class"))
#colnames(cm_ann) <- c("Predicted No", "Predicted Yes")
#row.names(cm_ann) <- c("Actual No", "Actual Yes")

#confusion matrix for the ANN
cm_ann
```

```
##
##          no   yes
##   no  23214   672
##   yes  1866  1122
```

```r
#develop the model 1 metrics to input into the confusion matrix
TP_ann <- cm_ann[1,1]
FN_ann <- cm_ann[1,2]
FP_ann <- cm_ann[2,1]
TN_ann <- cm_ann[2,2]

#totals for the corresponding row and column values of confusion matrix
TAN_ann <- FP_ann + TN_ann
TAP_ann <- TP_ann + FN_ann
TPN_ann <- TN_ann + FN_ann
TPP_ann <- FP_ann + TP_ann
GT_ann <- TP_ann + FN_ann + FP_ann + TN_ann

#generate the evaluation metrics for model 1
accuracy_ann <- round((TN_ann + TP_ann) / (GT_ann),4)
error_rate_sensitivity_ann <- (1 - accuracy_ann)
specificity_ann <- round((TN_ann/TAN_ann),4)
precision_ann <- round((TP_ann/TPP_ann),4)
f1_ann <- round(2* ((precision_ann * specificity_ann) / (precision_ann +
specificity_ann)),4)

model_eval_table_ann <- matrix(c( accuracy_ann,  specificity_ann,
precision_ann,
                                 f1_ann), ncol = 1, nrow =4, byrow = TRUE)

#develop the column and row names
```

```r
colnames(model_eval_table_ann) <- c("ANN Model Output ")
row.names(model_eval_table_ann) <-
c("Accuracy","Specificity","Precision","F1")

model_eval_table_ann

##             ANN Model Output
## Accuracy              0.9056
## Specificity           0.3755
## Precision             0.9256
## F1                    0.5343
```

28. Which baseline model do we compare your neural network model against? Did it outperform the baseline according to accuracy?

We would use a baseline model of an all negative model. Provide the output of our model performs better than the all negative model, then it is sufficient for use.

The all negative model has an accuracy of: (23,886 / 26,874) = 0.8888 = 88.88%

Therefore, with an accuracy of 0.9055, or 90.55%, our model outperforms the baseline model on accuracy.

29. Using the same predictors you used for your neural network model, build models to predict Response using the following algorithms:

a. CART

```r
library(rpart)

#develop the CART algorithm
cart <- rpart(formula = response ~  marital + education + loan +
previous_outcome + age.mm + duration.mm, data = bank_train,method = "class")

#subset the predictor variables from the test data set into own data frame
test.X_cart <- subset( x = bank_test, select = c("marital", "education",
"loan", "previous_outcome", "age.mm", "duration.mm"))

#predict the outputs of the test data set with CART
y_pred_cart <- predict(object = cart, newdata = test.X_cart, type = "class")

#return the confusion matrix for the CART algorithm
cm_cart <- table(bank_test$response , y_pred_cart)
cm_cart

##      y_pred_cart
##          no   yes
##   no  23336   550
##   yes  1933  1055
```

```
#cart_test <- rpart(formula = response ~  marital + education + loan +
#previous_outcome + age.mm + duration.mm, data = bank_test, method = "class")

#develop the CART metrics to input into the confusion matrix
TP_cart <- cm_cart[1,1]
FN_cart <- cm_cart[1,2]
FP_cart <- cm_cart[2,1]
TN_cart <- cm_cart[2,2]

#totals for the corresponding row and column values of confusion matrix
TAN_cart <- FP_cart + TN_cart
TAP_cart <- TP_cart + FN_cart
TPN_cart <- TN_cart + FN_cart
TPP_cart <- FP_cart + TP_cart
GT_cart <- TP_cart + FN_cart + FP_cart + TN_cart

#generate the evaluation metrics for model 1
accuracy_cart <- round((TN_cart + TP_cart) / (GT_cart),4)
error_rate_sensitivity_cart <- (1 - accuracy_cart)
specificity_cart <- round((TN_cart/TAN_cart),4)
precision_cart <- round((TP_cart/TPP_cart),4)
f1_cart <- round(2* ((precision_cart * specificity_cart) / (precision_cart +
specificity_cart)),4)

model_eval_table_cart <- matrix(c( accuracy_cart,  specificity_cart,
precision_cart,
                                 f1_ann), ncol = 1, nrow =4, byrow = TRUE)

#develop the column and row names
colnames(model_eval_table_cart) <- c("CART Model Output ")
row.names(model_eval_table_cart) <-
c("Accuracy","Specificity","Precision","F1")

model_eval_table_cart

##             CART Model Output
## Accuracy                0.9076
## Specificity             0.3531
## Precision               0.9235
## F1                      0.5343
```

b. C5.0

```
library(C50)

#train the C5 algorithm
C5 <- C5.0(formula = response ~  marital + education + loan +
previous_outcome + age.mm + duration.mm, data = bank_train, control =
C5.0Control(minCases = 100))

#subset the predictor variables from the test data set into own data frame
```

```
test.X_c5 <- subset( x = bank_test, select = c("marital", "education",
"loan", "previous_outcome", "age.mm", "duration.mm"))

#predict the outputs of the test data set with C5
y_pred_c5 <- predict(object = C5, newdata = test.X_c5)

#return the confusion matrix for the C5 algorithm
cm_c5 <- table(bank_test$response , y_pred_c5)
cm_c5

##      y_pred_c5
##          no   yes
##   no  23336   550
##   yes  1933  1055

#develop the C5 metrics to input into the confusion matrix
TP_c5 <- cm_c5[1,1]
FN_c5 <- cm_c5[1,2]
FP_c5 <- cm_c5[2,1]
TN_c5 <- cm_c5[2,2]

#totals for the corresponding row and column values of confusion matrix
TAN_c5 <- FP_c5 + TN_c5
TAP_c5 <- TP_c5 + FN_c5
TPN_c5 <- TN_c5 + FN_c5
TPP_c5 <- FP_c5 + TP_c5
GT_c5 <- TP_c5 + FN_c5 + FP_c5 + TN_c5

#generate the evaluation metrics for model 1
accuracy_c5 <- round((TN_c5 + TP_c5) / (GT_c5),4)
error_rate_sensitivity_c5 <- (1 - accuracy_c5)
specificity_c5 <- round((TN_c5/TAN_c5),4)
precision_c5 <- round((TP_c5/TPP_c5),4)
f1_c5 <- round(2* ((precision_c5 * specificity_c5) / (precision_c5 +
specificity_c5)),4)

model_eval_table_c5 <- matrix(c( accuracy_c5,  specificity_c5,  precision_c5,
                                 f1_ann), ncol = 1, nrow =4, byrow = TRUE)

#develop the column and row names
colnames(model_eval_table_c5) <- c("C5 Model Output ")
row.names(model_eval_table_c5) <-
c("Accuracy","Specificity","Precision","F1")

model_eval_table_c5

##             C5 Model Output
## Accuracy              0.9076
## Specificity           0.3531
```

```
## Precision                0.9235
## F1                       0.5343
```

c. Naive Bayes

```
library(e1071)

#run the Naive Bayes algorithm
nb <- naiveBayes(formula = response ~  marital + education + loan +
previous_outcome + age.mm + duration.mm, data = bank_train)

#subset the predictor varialbes from teh test data set into own data frame
test.X_nb <- subset( x = bank_test, select = c("marital", "education",
"loan", "previous_outcome", "age.mm", "duration.mm"))

#predict the outputs of test dataset with NB
y_pred_nb <- predict(object = nb , newdata = test.X_nb)

#return the confusion matrix for the NB model
cm_nb <- table(bank_test$response , y_pred_nb)
cm_nb

##      y_pred_nb
##          no   yes
##   no  23127   759
##   yes  1911  1077

#develop the CART metrics to input into the confusion matrix
TP_nb <- cm_nb[1,1]
FN_nb <- cm_nb[1,2]
FP_nb <- cm_nb[2,1]
TN_nb <- cm_nb[2,2]

#totals for the corresponding row and column values of confusion matrix
TAN_nb <- FP_nb + TN_nb
TAP_nb <- TP_nb + FN_nb
TPN_nb <- TN_nb + FN_nb
TPP_nb <- FP_nb + TP_nb
GT_nb <- TP_nb + FN_nb + FP_nb + TN_nb

#generate the evaluation metrics for model 1
accuracy_nb <- round((TN_nb + TP_nb) / (GT_nb),4)
error_rate_sensitivity_nb <- (1 - accuracy_nb)
specificity_nb <- round((TN_nb/TAN_nb),4)
precision_nb <- round((TP_nb/TPP_nb),4)
f1_nb <- round(2* ((precision_nb * specificity_nb) / (precision_nb +
specificity_nb)),4)

model_eval_table_nb <- matrix(c( accuracy_nb,  specificity_nb,  precision_nb,
                                 f1_nb), ncol = 1, nrow =4, byrow = TRUE)
```

```r
#develop the column and row names
colnames(model_eval_table_nb) <- c("Naive Bayes Model Output ")
row.names(model_eval_table_nb) <-
c("Accuracy","Specificity","Precision","F1")

model_eval_table_nb

##             Naive Bayes Model Output
## Accuracy                      0.9006
## Specificity                   0.3604
## Precision                     0.9237
## F1                            0.5185
```

30. Compare the results of your neural netwrok model with the three models from the previous exercise, according to the following criteria. Discuss in detail which model performed best and worst according to each criterion.

```r
print(model_eval_table_ann)

##             ANN Model Output
## Accuracy              0.9056
## Specificity           0.3755
## Precision             0.9256
## F1                    0.5343

print(model_eval_table_cart)

##             CART Model Output
## Accuracy               0.9076
## Specificity            0.3531
## Precision              0.9235
## F1                     0.5343

print(model_eval_table_c5)

##             C5 Model Output
## Accuracy             0.9076
## Specificity          0.3531
## Precision            0.9235
## F1                   0.5343

print(model_eval_table_nb)

##             Naive Bayes Model Output
## Accuracy                      0.9006
## Specificity                   0.3604
## Precision                     0.9237
## F1                            0.5185
```

a. Accuracy

When evaluating the overall measure of the proportion of correct classifications being made by the models, the CART and C5 models were slightly more accurate than the ANN model, with an accuracy score of 0.9076. Conversely, the ANN model was just slightly lower at 0.9055, and Naive Bayes at 0.9006.

b. Sensitivity (precision)

When evaluating the ability of the model to classify a record positively, the ANN model was marginally better than the CART, C5, and Naive Bayes models, however not by much. The ANN model peaked at a 0.9256 precision, with CART and C5 at 0.9235, and Naive Bayes at 0.9237.

c. Specificity

When evaluating the ability of the model to classify a record negatively, the ANN model was marginally better than the CART, C5 and Naive Bayes models. Specifically, the ANN model predicted approximately 2% more positive records than the CART and C5 model, and 1.5% more than Naive Bayes.

Data Science Using Python and R: Chapter 6 - Page 93: Questions #19 & 20

19. use random forests on the training data set to predict income using marital status and capital gains losses.

```
library(randomForest)

## randomForest 4.6-14

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

#import training data
training_data <- read.csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied
Data Mining 502/Module 3/Datasets/adult_ch6_training")

#import test data
test_data <- read.csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied Data
Mining 502/Module 3/Datasets/adult_ch6_test")

#preprocess the dataset for the random forest algorithm
training_data$Income <- factor(training_data$Income)
training_data$Marital.status <- factor(training_data$Marital.status)

#run the random forest algorithm
rf_1 <- randomForest(formula = Income ~ ., data = training_data, ntree = 100,
type = "classification")
#rf_1$predicted

#create the confusion matrix for the predicted and actuals
cm1 <- table(training_data[,2], rf_1$predicted)
colnames(cm1) <- c("Predicted <=50K", "Predicted >50K")
row.names(cm1) <- c("Actual <=50K", "Actual >50K")

cm1
```

```
##
##                  Predicted <=50K Predicted >50K
##   Actual <=50K            14153            118
##   Actual >50K              3266           1224
```

```
#develop the model 1 metrics to input into the confusion matrix
TP_1 <- cm1[1,1]
FN_1 <- cm1[1,2]
FP_1 <- cm1[2,1]
TN_1 <- cm1[2,2]

#totals for the corresponding row and column values of confusion matrix
TAN_1 <- FP_1 + TN_1
TAP_1 <- TP_1 + FN_1
TPN_1 <- TN_1 + FN_1
TPP_1 <- FP_1 + TP_1
GT_1<- TP_1 + FN_1 + FP_1 + TN_1

#generate the evaluation metrics for model 1
accuracy_1 <- round((TN_1 + TP_1) / (GT_1),4)
error_rate_sensitivity_1 <- (1 - accuracy_1)
specificity_1 <- round((TN_1/TAN_1),4)
precision_1 <- round((TP_1/TPP_1),4)
f1_1 <- round(2* ((precision_1 * specificity_1) / (precision_1 +
specificity_1)),4)
```

20. use random forests using the test data set that utilizes the same target and predictor variables. Does the test data result match the training data result?

```
#preprocess the test data
test_data$Income <- factor(test_data$Income)
test_data$Marital.status <- factor(test_data$Marital.status)

#run the random forest algorithm against the test data set
rf_2 <- randomForest(formula = Income ~ ., data = test_data, ntree = 100,
type = "classification")

#create the confusion matrix for the predicted and actuals from the test data
cm2 <- table(test_data[,2], rf_2$predicted)
colnames(cm2) <- c("Predicted <=50K", "Predicted >50K")
row.names(cm2) <- c("Actual <=50K", "Actual >50K")

cm2

##
##                  Predicted <=50K Predicted >50K
##   Actual <=50K             4630             44
##   Actual >50K              1087            394

#develop the model 2 metrics to input into the confusion matrix
TP_1 <- cm2[1,1]
```

```r
FN_1 <- cm2[1,2]
FP_1 <- cm2[2,1]
TN_1 <- cm2[2,2]

#totals for the corresponding row and column values of confusion matrix
TAN_1 <- FP_1 + TN_1
TAP_1 <- TP_1 + FN_1
TPN_1 <- TN_1 + FN_1
TPP_1 <- FP_1 + TP_1
GT_1<- TP_1 + FN_1 + FP_1 + TN_1

#generate the evaluation metrics for model 2
accuracy_2 <- round((TN_1 + TP_1) / (GT_1),4)
error_rate_sensitivity_2 <- (1 - accuracy_1)
specificity_2 <- round((TN_1/TAN_1),4)
precision_2 <- round((TP_1/TPP_1),4)
f1_2 <- round(2* ((precision_1 * specificity_1) / (precision_1 +
specificity_1)),4)

#create a matrix to compare the model side by side
model_eval_table <- matrix(c(accuracy_1, accuracy_2, (accuracy_1 -
accuracy_2),
                             specificity_1, specificity_2, (specificity_1 -
specificity_2),
                             precision_1, precision_2, (precision_1 -
precision_2),
                           f1_1, f1_2, f1_1 - f1_2),
                           ncol = 3, nrow =4, byrow = TRUE)

#develop the column and row names
colnames(model_eval_table) <- c("Model 1", "Model 2", "Difference Between
Models")
row.names(model_eval_table) <- c("Accuracy","Specificity","Precision","F1")

print(model_eval_table)

##             Model 1 Model 2 Difference Between Models
## Accuracy     0.8196  0.8162                    0.0034
## Specificity  0.2726  0.2660                    0.0066
## Precision    0.8125  0.8099                    0.0026
## F1           0.4082  0.4082                    0.0000

print(cm1)

##
##                  Predicted <=50K Predicted >50K
##   Actual <=50K             14153            118
##   Actual >50K               3266           1224

print(cm2)
```

```
##
##                 Predicted <=50K Predicted >50K
##   Actual <=50K             4630             44
##   Actual >50K              1087            394
```

The models are fairly similar, and the results of the training and test model are similar.