MLHW7\_JF

March 13, 2022

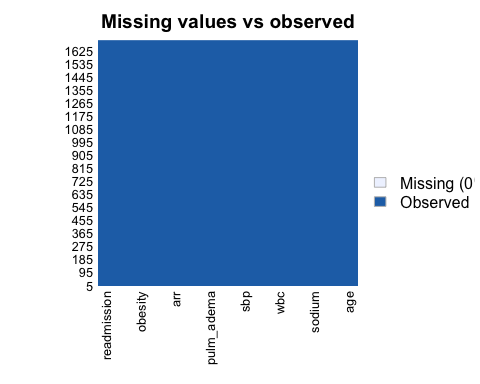
# Our main focus is to predict readmission for myocardial infarction.

# Data Prep: Stripping the ID variable, checking for missing data, looking at the frequency of the outcome variable (readmission) to check for balance, and partitioning the data into training and testing using a 70/30 split.

mi = read.csv("/Users/judyfordjuoh/Desktop/Machine Learning/mi.data.csv") %>%  
 janitor::clean\_names() %>%  
 mutate(readmission = recode(readmission,  
 "0" = "No",  
 "1" = "Yes"))   
  
mi$readmission <- factor(mi$readmission, levels = c("No", "Yes"))  
  
#Stripping off ID Variable  
mi <- mi[,2:16]  
  
#Check distributions, missing data etc, omitting the NAs  
summary(mi)

## age sex sodium alt   
## Min. :26.00 Min. :0.0000 Min. :117.0 Min. :0.0300   
## 1st Qu.:54.00 1st Qu.:0.0000 1st Qu.:133.0 1st Qu.:0.2300   
## Median :63.00 Median :1.0000 Median :136.0 Median :0.3800   
## Mean :61.87 Mean :0.6282 Mean :136.6 Mean :0.4736   
## 3rd Qu.:70.00 3rd Qu.:1.0000 3rd Qu.:140.0 3rd Qu.:0.6100   
## Max. :92.00 Max. :2.0000 Max. :169.0 Max. :3.0000   
## wbc esr sbp dbp   
## Min. : 2.000 Min. : 1.00 Min. : 0.0 Min. : 0.0   
## 1st Qu.: 6.400 1st Qu.: 5.00 1st Qu.:120.0 1st Qu.: 70.0   
## Median : 8.000 Median : 10.00 Median :140.0 Median : 80.0   
## Mean : 8.804 Mean : 13.48 Mean :138.8 Mean : 82.1   
## 3rd Qu.:10.500 3rd Qu.: 19.00 3rd Qu.:160.0 3rd Qu.: 90.0   
## Max. :27.900 Max. :140.00 Max. :260.0 Max. :190.0   
## pulm\_adema fc arr diab   
## Min. :0.00000 Min. :0.000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.00000 1st Qu.:0.000 1st Qu.:0.00000 1st Qu.:0.0000   
## Median :0.00000 Median :2.000 Median :0.00000 Median :0.0000   
## Mean :0.06529 Mean :1.216 Mean :0.02471 Mean :0.1341   
## 3rd Qu.:0.00000 3rd Qu.:2.000 3rd Qu.:0.00000 3rd Qu.:0.0000   
## Max. :1.00000 Max. :4.000 Max. :1.00000 Max. :1.0000   
## obesity asthma readmission  
## Min. :0.00000 Min. :0.00000 No :1541   
## 1st Qu.:0.00000 1st Qu.:0.00000 Yes: 159   
## Median :0.00000 Median :0.00000   
## Mean :0.02471 Mean :0.02235   
## 3rd Qu.:0.00000 3rd Qu.:0.00000   
## Max. :1.00000 Max. :1.00000

missmap(mi, main = "Missing values vs observed")



#Since there is no missing data we won't do na.omit(mi)  
  
summary(mi$readmission) #Notice that the data is unbalanced so we will have to upsize or downsize. For simplicity sake, I will downsize.

## No Yes   
## 1541 159

#tidyverse way to create data partition  
train\_indices <- createDataPartition(y = mi$readmission,p = 0.7,list = FALSE)  
train\_data <- mi[train\_indices, ]  
test\_data <- mi[-train\_indices, ]

# REGULARIZED REGRESSION: ELASTIC NET

#REGULARIZED REGRESSION: ELASTIC NET  
  
set.seed(150)  
  
#Creating 10-fold cross-validation and using down-sampling because of imbalance in data  
en.model <- train(  
 readmission ~., data = train\_data, method = "glmnet",  
 trControl = trainControl("cv", number = 10, sampling = "up"), preProc = c("center", "scale"), tuneLength = 10)  
  
#Print the values of alpha and lambda that gave best prediction  
en.model$bestTune %>% knitr::kable() # 0.3(alpha)| 0.029915(lambda)|

|  |  |  |
| --- | --- | --- |
|  | alpha | lambda |
| 4 | 0.1 | 0.0023071 |

#Print all of the options examined. This is a logistic regression = we are using the Accuracy.   
en.model$results %>% knitr::kable()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| alpha | lambda | Accuracy | Kappa | AccuracySD | KappaSD |
| 0.1 | 0.0001871 | 0.6204494 | 0.0839520 | 0.0230133 | 0.0670202 |
| 0.1 | 0.0004323 | 0.6204494 | 0.0839520 | 0.0230133 | 0.0670202 |
| 0.1 | 0.0009987 | 0.6204494 | 0.0839520 | 0.0230133 | 0.0670202 |
| 0.1 | 0.0023071 | 0.6204494 | 0.0839520 | 0.0230133 | 0.0670202 |
| 0.1 | 0.0053297 | 0.6187616 | 0.0828908 | 0.0243147 | 0.0662985 |
| 0.1 | 0.0123123 | 0.6153930 | 0.0686779 | 0.0259739 | 0.0777095 |
| 0.1 | 0.0284430 | 0.6145525 | 0.0717482 | 0.0311387 | 0.0811243 |
| 0.2 | 0.0001871 | 0.6028652 | 0.0748713 | 0.0286024 | 0.0659578 |
| 0.2 | 0.0004323 | 0.6028652 | 0.0748713 | 0.0286024 | 0.0659578 |
| 0.2 | 0.0009987 | 0.6028652 | 0.0748713 | 0.0286024 | 0.0659578 |
| 0.2 | 0.0023071 | 0.6037056 | 0.0754084 | 0.0287962 | 0.0663833 |
| 0.2 | 0.0053297 | 0.6028511 | 0.0803849 | 0.0285895 | 0.0645541 |
| 0.2 | 0.0123123 | 0.6070529 | 0.0893857 | 0.0303224 | 0.0683449 |
| 0.2 | 0.0284430 | 0.6104212 | 0.0917633 | 0.0332416 | 0.0711599 |
| 0.3 | 0.0001871 | 0.6179421 | 0.0799219 | 0.0273573 | 0.0733326 |
| 0.3 | 0.0004323 | 0.6179421 | 0.0799219 | 0.0273573 | 0.0733326 |
| 0.3 | 0.0009987 | 0.6179421 | 0.0799219 | 0.0273573 | 0.0733326 |
| 0.3 | 0.0023071 | 0.6162685 | 0.0788439 | 0.0266195 | 0.0732337 |
| 0.3 | 0.0053297 | 0.6129282 | 0.0715713 | 0.0286767 | 0.0720519 |
| 0.3 | 0.0123123 | 0.6095525 | 0.0660118 | 0.0363906 | 0.0862700 |
| 0.3 | 0.0284430 | 0.6070245 | 0.0680503 | 0.0332530 | 0.0845307 |
| 0.4 | 0.0001871 | 0.6003020 | 0.0650847 | 0.0285668 | 0.0710781 |
| 0.4 | 0.0004323 | 0.6003020 | 0.0650847 | 0.0285668 | 0.0710781 |
| 0.4 | 0.0009987 | 0.6003020 | 0.0650847 | 0.0285668 | 0.0710781 |
| 0.4 | 0.0023071 | 0.6019897 | 0.0693893 | 0.0261139 | 0.0618940 |
| 0.4 | 0.0053297 | 0.6019826 | 0.0756690 | 0.0279920 | 0.0597539 |
| 0.4 | 0.0123123 | 0.5994475 | 0.0737246 | 0.0271009 | 0.0631676 |
| 0.4 | 0.0284430 | 0.5935653 | 0.0734885 | 0.0259700 | 0.0639367 |
| 0.5 | 0.0001871 | 0.6045530 | 0.0701228 | 0.0316928 | 0.0726315 |
| 0.5 | 0.0004323 | 0.6045530 | 0.0701228 | 0.0316928 | 0.0726315 |
| 0.5 | 0.0009987 | 0.6045530 | 0.0701228 | 0.0316928 | 0.0726315 |
| 0.5 | 0.0023071 | 0.6037127 | 0.0665223 | 0.0317897 | 0.0768557 |
| 0.5 | 0.0053297 | 0.6011988 | 0.0675616 | 0.0270278 | 0.0756403 |
| 0.5 | 0.0123123 | 0.6037197 | 0.0751485 | 0.0268636 | 0.0777295 |
| 0.5 | 0.0284430 | 0.5953517 | 0.0748662 | 0.0174128 | 0.0748885 |
| 0.6 | 0.0001871 | 0.6179284 | 0.0900890 | 0.0283683 | 0.0634133 |
| 0.6 | 0.0004323 | 0.6179284 | 0.0900890 | 0.0283683 | 0.0634133 |
| 0.6 | 0.0009987 | 0.6187617 | 0.0906696 | 0.0289868 | 0.0635240 |
| 0.6 | 0.0023071 | 0.6162477 | 0.0889398 | 0.0263216 | 0.0631098 |
| 0.6 | 0.0053297 | 0.6137408 | 0.0870131 | 0.0235724 | 0.0609198 |
| 0.6 | 0.0123123 | 0.6120813 | 0.0892810 | 0.0241301 | 0.0627555 |
| 0.6 | 0.0284430 | 0.5936006 | 0.0754604 | 0.0234330 | 0.0558654 |
| 0.7 | 0.0001871 | 0.6061563 | 0.0641418 | 0.0289687 | 0.0656559 |
| 0.7 | 0.0004323 | 0.6061563 | 0.0641418 | 0.0289687 | 0.0656559 |
| 0.7 | 0.0009987 | 0.6061563 | 0.0641418 | 0.0289687 | 0.0656559 |
| 0.7 | 0.0023071 | 0.6078370 | 0.0651347 | 0.0282932 | 0.0662246 |
| 0.7 | 0.0053297 | 0.6078230 | 0.0684604 | 0.0276572 | 0.0686847 |
| 0.7 | 0.0123123 | 0.6086703 | 0.0712188 | 0.0284592 | 0.0724728 |
| 0.7 | 0.0284430 | 0.5851548 | 0.0676293 | 0.0273870 | 0.0663660 |
| 0.8 | 0.0001871 | 0.6036844 | 0.0795785 | 0.0296863 | 0.0571519 |
| 0.8 | 0.0004323 | 0.6036844 | 0.0795785 | 0.0296863 | 0.0571519 |
| 0.8 | 0.0009987 | 0.6036915 | 0.0794684 | 0.0276443 | 0.0567619 |
| 0.8 | 0.0023071 | 0.6112547 | 0.0865501 | 0.0252507 | 0.0615017 |
| 0.8 | 0.0053297 | 0.6078933 | 0.0839715 | 0.0239944 | 0.0565157 |
| 0.8 | 0.0123123 | 0.6003090 | 0.0736258 | 0.0313329 | 0.0649154 |
| 0.8 | 0.0284430 | 0.5759957 | 0.0635366 | 0.0297446 | 0.0571491 |
| 0.9 | 0.0001871 | 0.6154355 | 0.0719695 | 0.0254808 | 0.0816568 |
| 0.9 | 0.0004323 | 0.6154355 | 0.0719695 | 0.0254808 | 0.0816568 |
| 0.9 | 0.0009987 | 0.6145951 | 0.0713977 | 0.0250731 | 0.0811752 |
| 0.9 | 0.0023071 | 0.6104003 | 0.0686262 | 0.0233040 | 0.0808735 |
| 0.9 | 0.0053297 | 0.6079003 | 0.0643349 | 0.0228614 | 0.0797606 |
| 0.9 | 0.0123123 | 0.6036985 | 0.0649309 | 0.0190958 | 0.0684633 |
| 0.9 | 0.0284430 | 0.5885723 | 0.0699025 | 0.0251545 | 0.0539176 |
| 1.0 | 0.0001871 | 0.6112552 | 0.0697865 | 0.0239689 | 0.0806748 |
| 1.0 | 0.0004323 | 0.6112552 | 0.0697865 | 0.0239689 | 0.0806748 |
| 1.0 | 0.0009987 | 0.6104077 | 0.0692585 | 0.0238280 | 0.0804655 |
| 1.0 | 0.0023071 | 0.6095461 | 0.0726335 | 0.0232342 | 0.0690485 |
| 1.0 | 0.0053297 | 0.6061563 | 0.0745003 | 0.0300543 | 0.0595528 |
| 1.0 | 0.0123123 | 0.6053302 | 0.0703088 | 0.0251328 | 0.0674696 |
| 1.0 | 0.0284430 | 0.5809530 | 0.0579036 | 0.0214957 | 0.0745910 |

# Model coefficients  
coef(en.model$finalModel, en.model$bestTune$lambda)

## 15 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) -0.002916564  
## age 0.213851781  
## sex -0.157689984  
## sodium 0.011037274  
## alt 0.092251904  
## wbc 0.282893606  
## esr -0.019312263  
## sbp -0.102366936  
## dbp 0.295024964  
## pulm\_adema -0.096975252  
## fc 0.396647514  
## arr -0.164741520  
## diab 0.100908145  
## obesity 0.089158031  
## asthma -0.047349824

#Confusion Matrix  
confusionMatrix(en.model) #0. |(accuracy)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 56.5 3.9  
## Yes 34.1 5.5  
##   
## Accuracy (average) : 0.6205

The accuracy of the elastic net model was 59%.

# ENSEMBLE: BAGGING

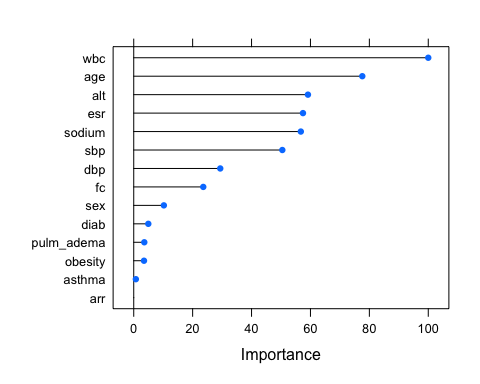
set.seed(150)  
  
#Set our value for mtry hyperparameter (the number of features eligible for selection at each node)  
#Remember, in bagging, all predictor features are eligible for selection at each node  
mtry.val1 <- expand.grid(.mtry = ncol(train\_data) - 1)  
  
trControl\_bag = trainControl("cv", number = 10, sampling = "up")  
  
bag\_readmission <- train(readmission ~., data = train\_data, method = "rf", metric = "Accuracy", trControl = trControl\_bag, tuneGrid = mtry.val1, ntree = 100)  
  
# accuracy results was a little bit better than the classification tree accuracy we ran before  
  
bag\_readmission$results #mtry: 14, Accuracy:

## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 14 0.8824917 -0.02354429 0.01828905 0.05703049

varImp(bag\_readmission) #wbc , age , esr , sodium

## rf variable importance  
##   
## Overall  
## wbc 100.0000  
## age 77.6019  
## alt 59.1468  
## esr 57.4600  
## sodium 56.7310  
## sbp 50.4506  
## dbp 29.3750  
## fc 23.6069  
## sex 10.2265  
## diab 4.9457  
## pulm\_adema 3.5868  
## obesity 3.4839  
## asthma 0.7179  
## arr 0.0000

plot(varImp(bag\_readmission))



confusionMatrix(bag\_readmission) #Accuracy:

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 88.2 9.3  
## Yes 2.4 0.1  
##   
## Accuracy (average) : 0.8825

The accuracy of the bagging model was 52%. The top three most important variables were white blood cell count (100) age at initial MI (81.43) and erythrocyte sedimentation rate (75.29). The least important variables were pulmonary adema (1.55), presence of arrythmia(0.13), and obesity (0.00).

# ENSEMBLE: RANDOM FOREST

set.seed(150)  
  
#Trying three different values of mtry (square root, half)  
#Set our value for mtry hyperparameter (the number of features eligible for selection at each node)  
# since we are not specifying our cross validation, the default is a bootstrap. R is bootstrapping 25 times.  
  
mtry.vals <- c(ncol(train\_data) - 1, sqrt(ncol(train\_data) - 1), 0.5\*ncol(train\_data) - 1)  
  
mtry.grid <- expand.grid(.mtry = mtry.vals)  
  
trControl\_rf = trainControl("cv", number = 10, sampling = "up")  
  
rf\_readmission <- train(readmission ~., data = train\_data, method = "rf", metric = "Accuracy", trControl = trControl\_rf, tuneGrid = mtry.grid, ntree = 100)  
  
confusionMatrix(rf\_readmission) #Accuracy (average) :

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 89.5 9.4  
## Yes 1.1 0.0  
##   
## Accuracy (average) : 0.895

rf\_readmission$results

## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 3.741657 0.8950405 -0.01868554 0.009956691 0.01672471  
## 2 6.500000 0.8950616 -0.01939324 0.007909950 0.01123093  
## 3 14.000000 0.8833105 -0.02575691 0.014815955 0.03310603

rf\_readmission$bestTune #mtry = 6.5

## mtry  
## 2 6.5

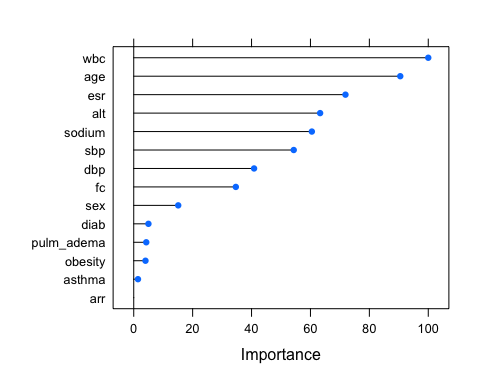
rf\_readmission$finalModel

##   
## Call:  
## randomForest(x = x, y = y, ntree = 100, mtry = min(param$mtry, ncol(x)))   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 6  
##   
## OOB estimate of error rate: 0.74%  
## Confusion matrix:  
## No Yes class.error  
## No 1063 16 0.01482854  
## Yes 0 1079 0.00000000

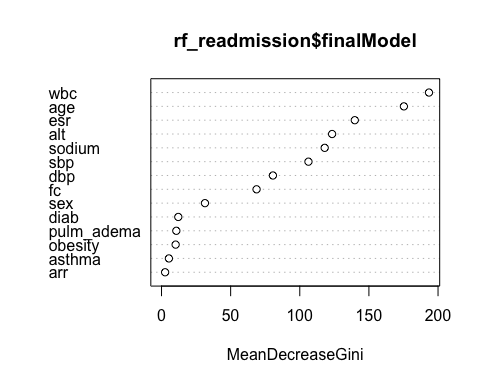
varImp(rf\_readmission) #age, wbc , esr

## rf variable importance  
##   
## Overall  
## wbc 100.000  
## age 90.494  
## esr 71.888  
## alt 63.280  
## sodium 60.474  
## sbp 54.321  
## dbp 40.855  
## fc 34.661  
## sex 15.104  
## diab 4.989  
## pulm\_adema 4.249  
## obesity 3.987  
## asthma 1.439  
## arr 0.000

plot(varImp(rf\_readmission))



varImpPlot(rf\_readmission$finalModel)

 The accuracy of the random forest model was 55%. The top three most important variables were age at initial MI (100), white blood cell count (92.2), and erythrocyte sedimentation rate (66.4). The least important variables were pulmonary adema (2), asthma (0.8), and obesity (0.00).

## Selecting an “optimal” model and calculate final evaluation metrics in the test set.

The optimal model I selected was the random forest model because the accuracy was the highest out of the three models I’ve constructed.

#Checking out info about final model  
rf\_readmission$finalModel

#Make predictions in testset  
rf\_pred\_test <- predict(rf\_readmission, test\_data)  
  
#Get evaluation metrics from test set  
confusionMatrix(rf\_pred\_test, test\_data$readmission, positive = "Yes") #Accuracy #Sensitivity: #Specificity:

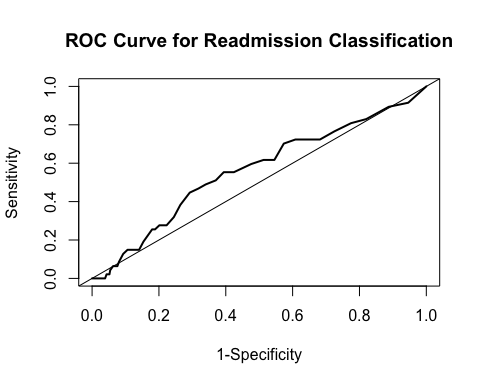
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 456 47  
## Yes 6 0  
##   
## Accuracy : 0.8959   
## 95% CI : (0.866, 0.921)  
## No Information Rate : 0.9077   
## P-Value [Acc > NIR] : 0.8404   
##   
## Kappa : -0.0214   
##   
## Mcnemar's Test P-Value : 3.92e-08   
##   
## Sensitivity : 0.00000   
## Specificity : 0.98701   
## Pos Pred Value : 0.00000   
## Neg Pred Value : 0.90656   
## Prevalence : 0.09234   
## Detection Rate : 0.00000   
## Detection Prevalence : 0.01179   
## Balanced Accuracy : 0.49351   
##   
## 'Positive' Class : Yes   
##

#Create ROC Curve for Analysis  
pred.prob <- predict(rf\_readmission, test\_data, type = "prob")  
  
#Another potential evaluation: Area under the Receiver Operating Curve (AUROC)  
#The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.  
analysis <- roc(response = test\_data$readmission, predictor = pred.prob[,2])

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

## Setting direction: controls < cases

plot(1 - analysis$specificities,analysis$sensitivities,type = "l",  
ylab = "Sensitivity",xlab = "1-Specificity",col = "black",lwd = 2,  
main = "ROC Curve for Readmission Classification")  
abline(a = 0,b = 1)

 The accuracy of the random forest model on the test data was . % with a sensitivity of 0. and a specificity of 0. .