Unit7: For Live Session

DS6306

Garrity

PART 1 - Titanic Survival Classification

70/30 train/test split

```
for_NB <- df %>% select(age,pclass,survived) # just
the data we need for initial NB classifier

model = naiveBayes(survived~.,data = for_NB)
predictions <- predict(model,for_NB[,c(1,2)])

# calculate the posterior probability:
predict(model,df, type = "raw") #gives probabilities

             No       Yes
  [1,] 0.7527590 0.2472410
  [2,] 0.7653859 0.2346141
  [3,] 0.7902548 0.2097452
  [4,] 0.5890190 0.4109810
  [5,] 0.2832819 0.7167181
  [6,] 0.7786641 0.2213359
  [7,] 0.6011130 0.3988870
  [8,] 0.5585488 0.4414512
  [9,] 0.7735716 0.2264284
 [10,] 0.6111656 0.3888344
```

PART 1 - Titanic Survival Classification

Performance Metrics

```
dfClean = df %>% select(age,pclass,survived) %>%
filter(!is.na(age) & !is.na(pclass))
set.seed(4)
trainIndices =
sample(seq(1:length(dfClean$age)),round(.7*length(dfCl
ean$age)))
trainTitanic = dfClean[trainIndices,]
testTitanic = dfClean[-trainIndices,]

head(trainTitanic)
head(testTitanic)

model = naiveBayes(survived~.,data = trainTitanic)

confusionMatrix(table(predict(model,testTitanic[,c(1,2
)]),testTitanic$survived))
```

```
Confusion Matrix and Statistics


       No Yes
  No  105  53
  Yes  16  40

               Accuracy : 0.6776
                 95% CI : (0.6105,
0.7397)
    No Information Rate : 0.5654
    P-Value [Acc > NIR] : 0.0005134

                  Kappa : 0.3122

 Mcnemar's Test P-Value : 1.465e-05

            Sensitivity : 0.8678
            Specificity : 0.4301
         Pos Pred Value : 0.6646
         Neg Pred Value : 0.7143
             Prevalence : 0.5654
         Detection Rate : 0.4907
   Detection Prevalence : 0.7383
      Balanced Accuracy : 0.6489

       'Positive' Class : No
```

PART 1 - Titanic Survival Classification

Compare to knn

Naive Bayes

```
Confusion Matrix and Statistics


       No Yes
  No  105  53
  Yes  16  40

               Accuracy : 0.6776
                 95% CI : (0.6105,
0.7397)
    No Information Rate : 0.5654
    P-Value [Acc > NIR] : 0.0005134

                  Kappa : 0.3122

 Mcnemar's Test P-Value : 1.465e-05

            Sensitivity : 0.8678
            Specificity : 0.4301
         Pos Pred Value : 0.6646
         Neg Pred Value : 0.7143
             Prevalence : 0.5654
         Detection Rate : 0.4907
   Detection Prevalence : 0.7383
      Balanced Accuracy : 0.6489

       'Positive' Class : No
```

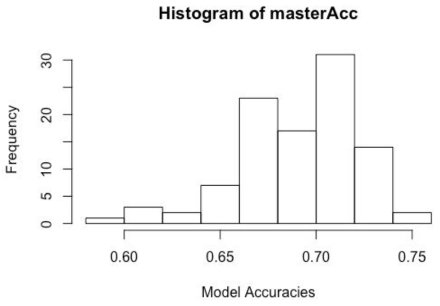
k-nn

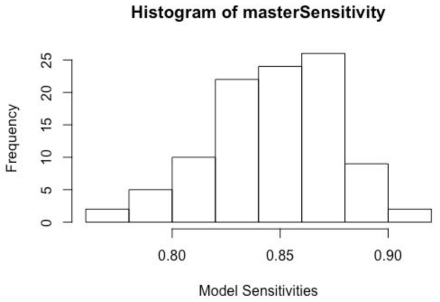
```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  371 159
       Yes  53 131

               Accuracy : 0.7031
                 95% CI : (0.6681,
0.7364)
    No Information Rate : 0.5938
    P-Value [Acc > NIR] : 8.964e-10

                  Kappa : 0.3468

 Mcnemar's Test P-Value : 5.537e-13

            Sensitivity : 0.8750
            Specificity : 0.4517
         Pos Pred Value : 0.7000
         Neg Pred Value : 0.7120
             Prevalence : 0.5938
         Detection Rate : 0.5196
   Detection Prevalence : 0.7423
      Balanced Accuracy : 0.6634

       'Positive' Class : No
```

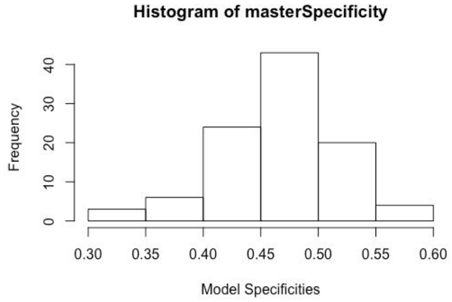
PART 1 - Titanic Survival Classification

Iterative model fitting

```
dfClean = df %>% select(age,pclass,survived) %>% filter(!is.na(age) &
!is.na(pclass))

iterations = 100
masterAcc = matrix(nrow = iterations)
masterSensitivity = matrix(nrow = iterations)
masterSpecificity = matrix(nrow = iterations)
splitPerc = 0.70 # Train/Test split

for(j in 1:iterations)
{
  #set.seed(floor(runif(1,1,100)))...this created issues for me...for discussion
  trainIndices = sample(seq(1:length(dfClean$age)),round(.7*length(dfClean$age)))
  trainTitanic = dfClean[trainIndices,]
  testTitanic = dfClean[-trainIndices,]

  model = naiveBayes(trainTitanic[,c(1,2)],trainTitanic$survived)
  table(predict(model,testTitanic[,c(1,2)]),testTitanic$survived)
  CM =
confusionMatrix(table(predict(model,testTitanic[,c(1,2)]),testTitanic$survived))
  masterAcc[j] = CM$overall[1]
  masterSensitivity[j] = CM$byClass[1]
  masterSpecificity[j] = CM$byClass[2]
}

> MeanAcc
[1] 0.6509813
> MeanSensitivity
[1] 0.8001224
> MeanSpecificity
[1] 0.4637028
```







PART 1 - Titanic Survival Classification

Add gender as a predictor variable

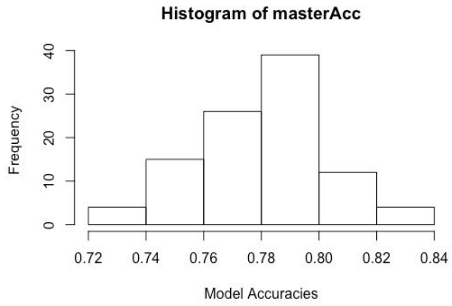
```
dfClean = df %>% select(age,pclass,sex,survived) %>%
filter(!is.na(age) & !is.na(pclass) & !is.na(sex))

dfClean$sexN = ifelse(dfClean$sex == "male",0,1)
dfClean <- dfClean[-3]
dfClean <- dfClean[c(1,2,4,3)]
str(dfClean)

set.seed(4)
trainIndices =
sample(seq(1:length(dfClean$age)),round(.7*length(dfClean$age)))
trainTitanic = dfClean[trainIndices,]
testTitanic = dfClean[-trainIndices,]

head(trainTitanic)
head(testTitanic)

model = naiveBayes(survived~.,data = trainTitanic)

confusionMatrix(table(predict(model,testTitanic[,c(1,3)]),testTita
nic$survived))
```

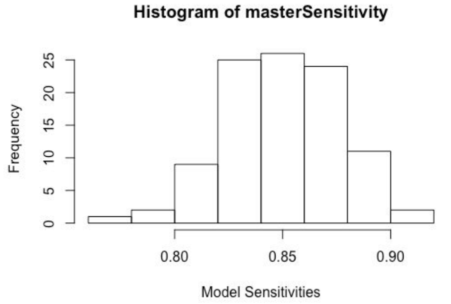
```
Confusion Matrix and Statistics


        No Yes
  No   104  29
  Yes   17  64

               Accuracy : 0.785
                 95% CI : (0.7239,
0.8381)
    No Information Rate : 0.5654
    P-Value [Acc > NIR] : 1.34e-11

                  Kappa : 0.556

 Mcnemar's Test P-Value : 0.1048

            Sensitivity : 0.8595
            Specificity : 0.6882
         Pos Pred Value : 0.7820
         Neg Pred Value : 0.7901
             Prevalence : 0.5654
         Detection Rate : 0.4860
   Detection Prevalence : 0.6215
      Balanced Accuracy : 0.7738

       'Positive' Class : No
```

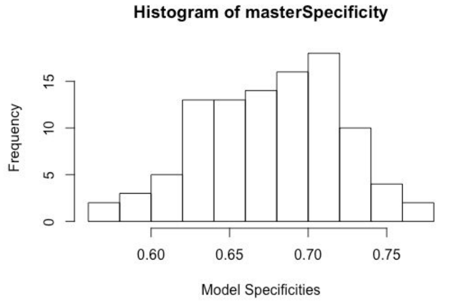
PART 1 - Titanic Survival Classification

Add gender as a predictor variable and iterate

```
iterations = 100
masterAcc = matrix(nrow = iterations)
masterSensitivity = matrix(nrow = iterations)
masterSpecificity = matrix(nrow = iterations)
splitPerc = 0.70 # Train/Test split

for(j in 1:iterations)
{
  # set.seed(floor(runif(1,1,100)))
  trainIndices =
sample(seq(1:length(dfClean$age)),round(.7*length(dfClean$age)))
  trainTitanic = dfClean[trainIndices,]
  testTitanic = dfClean[-trainIndices,]

  model = naiveBayes(trainTitanic[,c(1,3)],trainTitanic$survived)
  table(predict(model,testTitanic[,c(1,3)]),testTitanic$survived)
  CM =
confusionMatrix(table(predict(model,testTitanic[,c(1,3)]),testTita
nic$survived))
  masterAcc[j] = CM$overall[1]
  masterSensitivity[j] = CM$byClass[1]
  masterSpecificity[j] = CM$byClass[2]
}

> MeanAcc
[1] 0.78
> MeanSensitivity
[1] 0.8498545
> MeanSpecificity
[1] 0.6768692
```



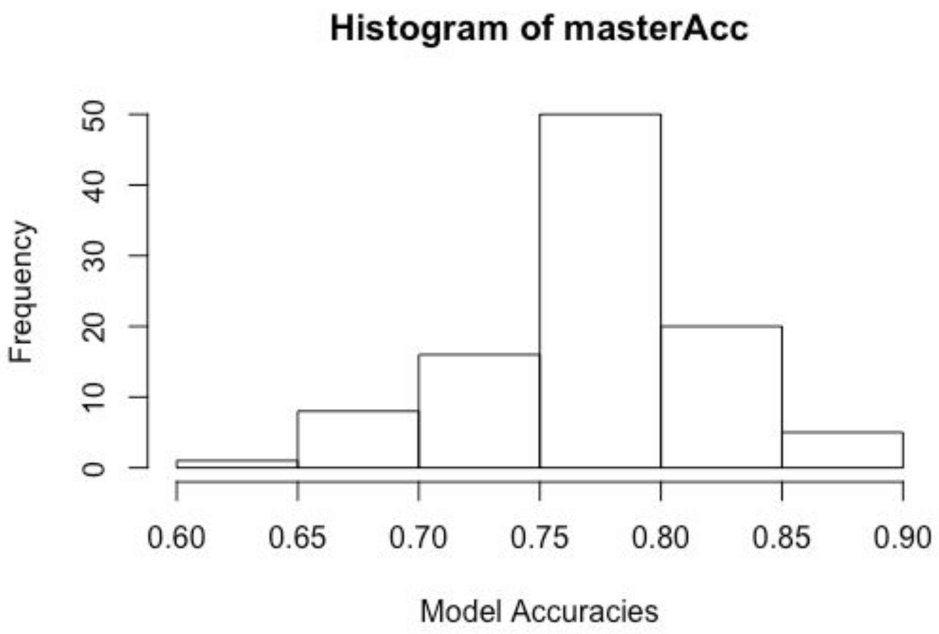




PART 2 - Iris Species Classification
70/30 train/test split

```
iris_forNB = iris %>% select(Sepal.Length, Sepal.Width,
Species)
summary(iris_forNB)

iterations = 100
masterAcc = matrix(nrow = iterations)
splitPerc = 0.70 # Train/Test split

for(j in 1:iterations)
{
  trainIndices = sample(1:dim(iris_forNB)[1],round(splitPe
* dim(iris_forNB)[1]))
  train = iris_forNB[trainIndices,]
  test = iris_forNB[-trainIndices,]

  model = naiveBayes(train[,c(1,2)],train$Species)
  table(predict(model,test[,c(1,2)]),test$Species)
  CM =
confusionMatrix(table(predict(model,test[,c(1,2)]),test$Sp
es))
  masterAcc[j] = CM$overall[1]
}

MeanAcc = colMeans(masterAcc)
> MeanAcc
[1] 0.7768889

hist(masterAcc, xlab="Model Accuracies")
```

k-nn Accuracy ----> 0.8267 (slightly higher compared to NB)



Takeaways & Questions

I’m having a tough time wrapping my head around the ultimate purpose of the train/test split iterations. At the end of the day we end up training/testing our model with a single subset of train and test data, right? Is there anything that we can do with the ensemble of models that we fit with multiple train test splits to improve the overall accuracy?

Would it be advisable to also iterate on the train/test ratio?

The train/test iterations provide us with a sample of accuracy metrics. Is there any value in reporting those metrics? For example, if we find that there is a wide range of accuracies that come from the iteration, then we know that our model is highly sensitive to the exact make up of the training and testing subsets (probably not a great situation). Then what? Collect more data? Try a different train/test ratio? Something else?

I’d appreciate spending a few minutes about the use of a random seed. I understand that we would want to use a fixed seed to ensure the same outcome, but the use of a random seed is less clear.