````
---
title: "Todd_Garner_DS6306_Week7_FLS"
author: "Todd Garner"
date: "2023-02-12"
output: powerpoint_presentation
editor_options:
  chunk_output_type: inline
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(dev = c('pdf', 'png'),
        fig.align = 'center', fig.height = 5, fig.width = 8.5,
        pdf.options(encoding = "ISOLatin9.enc"))

library(class)
library(caret)
library(e1071)
library(dplyr)
library(jsonlite)
library(ggplot2)
library(ggthemes)
library(tidyverse)
library(gridExtra)
```

# DS 6306 Week 7 FLS

## Part 2

### Question 1 - Now add Sex to the model so that it has Age, Pclass and Sex in the NB model.  Use the trainTitanic(set.seed(4)) dataframe to train the model and create a confusion matrix using the testTitanic dataframe.  In addition, find the Accuracy, Sensitivity and Specificity. (1 slide)


```{r}
# Read in the training set.  Check to "View" the full file to make sure it's what we want.
Titanic <- read.csv(file.choose(), header = TRUE)
View(Titanic)
```
We won't need to run that piece of code again so I'm isolating it.

```{r}
#Titanic$SurvivedF <- factor(Titanic$Survived, labels = c("Died", "Survived"))
Titanic$MF <- factor(Titanic$Sex, labels = c("0", "1")) #male = 1, female = 0
head(Titanic$MF)
head(Titanic)
Titanic_sub <- Titanic %>% filter(!is.na(Age) & !is.na(Pclass) & !is.na(MF))
Titanic_Sub_filter <- Titanic_sub %>% select(Age, Pclass, Survived, MF)
head(Titanic_Sub_filter)
model <- naiveBayes(Titanic_Sub_filter[((Titanic_Sub_filter$Age) &
(Titanic_Sub_filter$Pclass) & (Titanic_Sub_filter$MF)), c("Age", "Pclass", "MF")],
Titanic_Sub_filter$Survived, laplace = 1)
Titanic_clean = Titanic %>% filter(!is.na(Age) & !is.na(Pclass) & !is.na(MF))
set.seed(4)
trainIndices = sample(seq(1:length(Titanic_clean$MF)),round(.7*length(Titanic_clean$MF)))
trainTitanic = Titanic_clean[trainIndices,]
testTitanic = Titanic_clean[-trainIndices,]
head(trainTitanic)
dim(trainTitanic)
head(testTitanic)
dim(testTitanic)
model <- naiveBayes(trainTitanic,as.factor(trainTitanic$Survived) , laplace = 1)
````
````

```
summary(model)
dim(model)
#head(model)
df <- data.frame(testTitanic)
nrow(df)
x <- round(predict(model, df, type = "raw"), digits = 0)
y <- x[,2]
y
dim(df$Survived)
nrow(df$Survived)
table(y, df$Survived)
confusionMatrix(table(y, df$Survived))
```

Question 4 - ***Now repeat the above with a new seed and compare the accuracy, sensitivity and specificity. Do this 3 or 4 times to observe the variance in the statistics. (At least one slide.)***

###By changing the seed, the metrics in the confusion matrix changed. Likely because there were more or less NA's in each instance, but the accuracy never wavered much away from 100%. I must say this is surprising as it just doesn't seem likely to have a model that is 100% accurate. I checked and rechecked my model and my data.frame and made sure that the model was fed by training data via the model and testing data via the other variable in the table/confusionMatrix. 100% sure made me think I was comparing train to train or test to test. I still have a nagging feeling that I've missed something somewhere.

Question 5 - ***Write a loop to repeat the above for 100 different values of the seed. Find the average of the accuracy, sensitivity and specificity to get a stable (smaller variance) statistic to evaluate the model. (At least one slide.)***

```{r}
Titanic_clean = Titanic %>% filter(!is.na(Age) & !is.na(Pclass) & !is.na(MF))
iterations = 100
master_sens <- 0
master_spec <- 0
master_acc <- 0
master_sens <- data.frame(master_sens)
master_spec <- data.frame(master_spec)
master_acc <- data.frame(master_acc)

for(i in 1:iterations) {
set.seed(i)
trainIndices = sample(seq(1:length(Titanic_clean$MF)),round(.7*length(Titanic_clean$MF)))
trainTitanic = Titanic_clean[trainIndices,]
testTitanic = Titanic_clean[-trainIndices,]

model <- naiveBayes(trainTitanic,as.factor(trainTitanic$Survived) , laplace = 1)
df <- data.frame(testTitanic)
x <- round(predict(model, df, type = "raw"), digits = 0)
y <- x[,2]
y
master_sens[,i] = sensitivity(factor(y), factor(df$Survived))
master_spec[,i] = specificity(factor(y), factor(df$Survived))
z <- table(factor(y), factor(df$Survived))
CM <- confusionMatrix(z, k = i)
master_acc[,i] = CM$overall[1]

}
mean_sens = colMeans(master_sens)
mean_spec = colMeans(master_spec)
mean_acc = colMeans(master_acc)

which.max(mean_sens)
max(mean_sens)
```

```
which.max(mean_spec)
max(mean_spec)
which.max(mean_acc)
max(mean_acc)


plot(mean_sens,xlab = "Iterations", ylab = "Mean Sensitivity", main = "Seed iterations
versus Mean Sensitivity", type = "b")
plot(mean_spec,xlab = "Iterations", ylab = "Mean Specificity", main = "Seed iterations
versus Mean Specificity",type = "b")
plot(mean_acc, xlab = "Iterations", ylab = "Mean Accuracy", main = "Seed iterations versus
Mean Accuracy",type = "b")
```


```{r}
View(iris)
iris_clean = iris %>% filter(!is.na(Sepal.Length) & !is.na(Sepal.Width))
iterations = 100

for(i in 1:iterations) {
set.seed(i)
trainiris =
sample(seq(1:length(iris_clean$Sepal.Length)),round(.7*length(iris_clean$Sepal.Length)))
trainIris = iris_clean[trainIndices,]
testIris = iris_clean[-trainIndices,]

model <- naiveBayes(trainIris,as.factor(trainIris$Sepal.Length & trainIris$Sepal.Width) ,
laplace = 1)
df <- data.frame(testIris)
x <- round(predict(model, df, type = "raw"), digits = 1)
y <- x[,2]
master_sens[,i] = sensitivity(factor(y), factor(df$Survived))
master_spec[,i] = specificity(factor(y), factor(df$Survived))
CM <- confusionMatrix(factor(y), factor(df$Survived), k = i)
master_acc[,i] = CM$overall
}

mean_sens = colMeans(master_sens)
mean_spec = colMeans(master_spec)
mean_acc = colMeans(master_acc)

which.max(mean_sens)
max(mean_sens)
which.max(mean_spec)
max(mean_spec)
which.max(mean_acc)
max(mean_acc)
```

```
set.seed(1)
iterations = 10
numks = 20

masterAcc = matrix(nrow = iterations, ncol = numks)

for(j in 1:iterations)
{

  for(i in 1:numks)
  {
    CM = confusionMatrix(table(iris[,5],knn.cv(iris[,c(1,2)],iris[,5],k = i)))
    masterAcc[j,i] = CM$overall[1]

  }
```

```
}

MeanAcc = colMeans(masterAcc)

plot(seq(1,numks,1),MeanAcc, type = "l")

which.max(MeanAcc)
max(MeanAcc)
```