DSE6211 Module 05 Lab 05

Joseph Annand

2024-02-17

## Load Libraries

library(dplyr)  
library(caret)  
library(reticulate)  
library(tensorflow)  
library(keras)  
library(MESS)  
library(reticulate)  
library(tensorflow)  
library(keras)  
library(AppliedPredictiveModeling)

## Process and

data <- read.csv("lab\_5\_data/lab\_5\_data.csv")  
training\_ind <- createDataPartition(data$lodgepole\_pine,  
 p = 0.75,  
 list = F,  
 times = 1)

training\_set <- data[training\_ind, ]  
test\_set <- data[-training\_ind, ]  
  
top\_20\_soil\_types <- training\_set %>%  
 group\_by(soil\_type) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count)) %>%  
 select(soil\_type) %>%  
 top\_n(20)  
  
training\_set$soil\_type <- ifelse(training\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 training\_set$soil\_type,  
 "other")  
  
training\_set$wilderness\_area <- factor(training\_set$wilderness\_area)  
training\_set$soil\_type <- factor(training\_set$soil\_type)  
  
  
onehot\_encoder <- dummyVars(~ wilderness\_area + soil\_type,  
 training\_set[, c("wilderness\_area", "soil\_type")],  
 levelsOnly = T,  
 fullRank = T)  
  
onehot\_enc\_training <- predict(onehot\_encoder,  
 training\_set[, c("wilderness\_area", "soil\_type")])  
  
training\_set <- cbind(training\_set, onehot\_enc\_training)  
  
  
test\_set$soil\_type <- ifelse(test\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 test\_set$soil\_type,  
 "other")  
  
test\_set$wilderness\_area <- factor(test\_set$wilderness\_area)  
test\_set$soil\_type <- factor(test\_set$soil\_type)  
  
onehot\_enc\_test <- predict(onehot\_encoder, test\_set[, c("wilderness\_area", "soil\_type")])  
test\_set <- cbind(test\_set, onehot\_enc\_test)  
  
  
test\_set[, -c(11:13)] <- scale(test\_set[, -c(11:13)],  
 center = apply(training\_set[, -c(11:13)], 2, mean),  
 scale = apply(training\_set[, -c(11:13)], 2, sd))  
training\_set[, -c(11:13)] <- scale(training\_set[, -c(11:13)])  
  
  
training\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(training\_set), 33))  
training\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(training\_set)))  
  
test\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(test\_set), 33))  
test\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(test\_set)))

use\_virtualenv("my\_tf\_workspace")  
  
  
model <- keras\_model\_sequential(list(  
 layer\_dense(units = 50, activation = "relu"),  
 layer\_dense(units = 25, activation = "relu"),  
 layer\_dense(units = 1, activation = "sigmoid")))  
  
compile(model,  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = "accuracy")  
  
history <- fit(model, training\_features, training\_labels,  
 epochs = 40, batch\_size = 512, validation\_split = 0.33)

## Evaluate classification model

predictions <- predict(model, test\_features)

## 69/69 - 0s - 105ms/epoch - 2ms/step

test\_set$p\_prob <- predictions[, 1]  
head(predictions, 10)

## [,1]  
## [1,] 0.8619266  
## [2,] 0.1766049  
## [3,] 0.1203053  
## [4,] 0.2772984  
## [5,] 0.7628230  
## [6,] 0.6399689  
## [7,] 0.7847828  
## [8,] 0.9888736  
## [9,] 0.6266261  
## [10,] 0.8773760

over\_threshold <- test\_set[test\_set$p\_prob >= 0.5, ]  
  
# Percentage of incorrect predictions for observations of the negative class  
fpr <- sum(over\_threshold$lodgepole\_pine==0)/sum(test\_set$lodgepole\_pine==0)  
fpr

## [1] 0.5315068

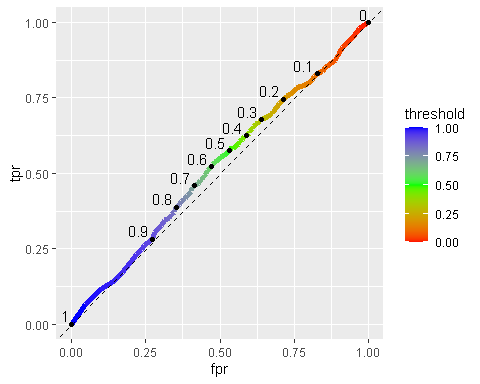
# Percentage of correct predictions for observations of the positive class  
tpr <- sum(over\_threshold$lodgepole\_pine==1)/sum(test\_set$lodgepole\_pine==1)  
tpr

## [1] 0.5747232

### ROC Curve

# Data processing for ROC curve  
roc\_data <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)  
for (i in roc\_data$threshold) {  
   
 over\_threshold <- test\_set[test\_set$p\_prob >= i, ]  
   
 fpr <- sum(over\_threshold$lodgepole\_pine==0)/sum(test\_set$lodgepole\_pine==0)  
 roc\_data[roc\_data$threshold==i, "fpr"] <- fpr  
   
 tpr <- sum(over\_threshold$lodgepole\_pine==1)/sum(test\_set$lodgepole\_pine==1)  
 roc\_data[roc\_data$threshold==i, "tpr"] <- tpr  
   
}  
  
# ROC curve  
ggplot() +  
 geom\_line(data = roc\_data, aes(x=fpr, y=tpr, color = threshold), size = 2) +  
 scale\_color\_gradientn(colors = rainbow(3)) +  
 geom\_abline(intercept = 0, slope = 1, lty = 2) +  
 geom\_point(data = roc\_data[seq(1, 101, 10), ], aes(x = fpr, y =tpr)) +  
 geom\_text(data = roc\_data[seq(1, 101, 10), ],  
 aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2))

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



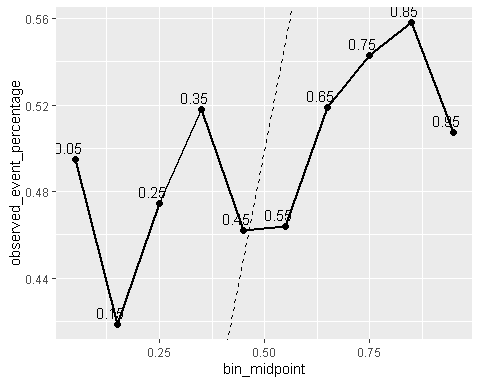
### AUC

auc <- auc(x = roc\_data$fpr, y = roc\_data$tpr, type = "spline")  
auc

## [1] 0.5202309

### Calibration Curve

calibration\_data <- data.frame(bin\_midpoint=seq(0.05, 0.95, 0.1),  
 observed\_event\_percentage=0)  
for (i in seq(0.05,0.95,0.1)) {  
   
 in\_interval <- test\_set[test\_set$p\_prob >= (i-0.05) & test\_set$p\_prob <= (i+0.05), ]  
 oep <- nrow(in\_interval[in\_interval$lodgepole\_pine==1, ])/nrow(in\_interval)  
 calibration\_data[calibration\_data$bin\_midpoint==i, "observed\_event\_percentage"] <- oep  
   
}  
  
# Calibration curve  
  
ggplot(data = calibration\_data, aes(x = bin\_midpoint, y = observed\_event\_percentage)) +  
 geom\_line(size = 1) +  
 geom\_abline(intercept = 0, slope = 1, lty = 2) +  
 geom\_point(size = 2) +  
 geom\_text(aes(label = bin\_midpoint), hjust = 0.75, vjust = -0.5)



## Exercises

### Exercise 1

x <- roc\_data$threshold[71]  
roc\_data[roc\_data$threshold==x, ]

## threshold fpr tpr  
## 71 0.3 0.6383562 0.6780443

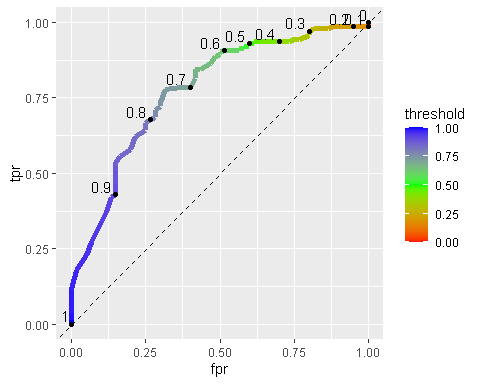
### Exercise 2

Under confident because the value lies above the diagonal dashed line.

### Exercise 3

data("logisticCreditPredictions")  
lcp <- logisticCreditPredictions

# ROC curve  
  
roc\_data <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)  
 for (i in roc\_data$threshold) {  
 over\_threshold <- lcp[lcp$Good >= i, ]  
 fpr <- sum(over\_threshold$obs=="Bad")/sum(lcp$obs=="Bad")  
 roc\_data[roc\_data$threshold==i, "fpr"] <- fpr  
 tpr <- sum(over\_threshold$obs=="Good")/sum(lcp$obs=="Good")  
 roc\_data[roc\_data$threshold==i, "tpr"] <- tpr  
 }  
  
ggplot() +  
 geom\_line(data = roc\_data, aes(x = fpr, y = tpr, color = threshold), size = 2) +  
 scale\_color\_gradientn(colors = rainbow(3)) +  
 geom\_abline(intercept = 0, slope = 1, lty = 2) +  
 geom\_point(data = roc\_data[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +  
 geom\_text(data = roc\_data[seq(1, 101, 10), ],  
 aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2))



calibration\_data <- data.frame(bin\_midpoint=seq(0.05,0.95,0.1),  
 observed\_event\_percentage=0)  
for (i in seq(0.05,0.95,0.1)) {  
 in\_interval <- lcp[lcp$Good >= (i-0.05) & lcp$Good <= (i+0.05), ]  
 temp <- nrow(in\_interval[in\_interval$obs=="Good", ])/nrow(in\_interval)  
 calibration\_data[calibration\_data$bin\_midpoint==i, "observed\_event\_percentage"] <- temp  
}  
  
ggplot(data = calibration\_data, aes(x = bin\_midpoint, y = observed\_event\_percentage)) +  
 geom\_line(size = 1) +  
 geom\_abline(intercept = 0, slope = 1, lty = 2) +  
 geom\_point(size = 2) +  
 geom\_text(aes(label = bin\_midpoint), hjust = 0.75, vjust = -0.5)

