Homework 1

Lorenzo Bocchi

2022-04-11

The aim of the following analysis is to understand which factors mostly affect the decision of pregnant women to breastfeed their babies. The data comes from a study conducted in the UK. In total, 139 mothers were asked what kind of feeding method they would choose for their incoming baby. Throughout the analysis we will see different prediction models and compare their results. We start by loading the data and exploring it:

```
load("./breastfeed.Rdata")
```

```
bf <- breastfeed
summary(bf)
```

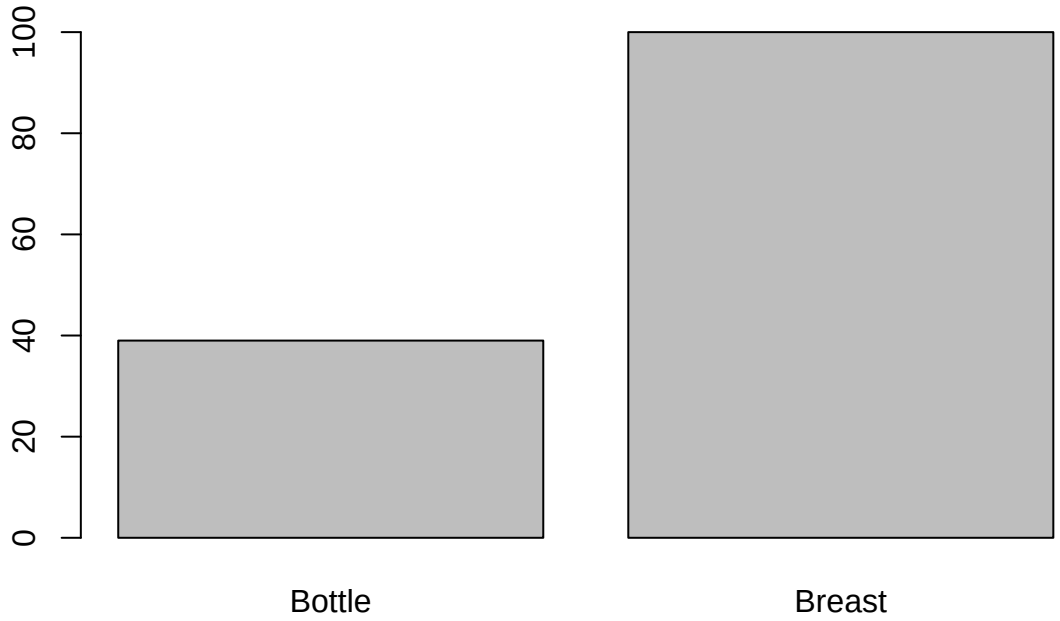
```
##     breast        pregnancy     howfed     howfedfr     partner    smokenow
##  Bottle: 39   End      :84   Bottle:59   Bottle:54   Single : 21   No :107
##  Breast:100   Beginning:55   Breast:80   Breast:85   Partner:118   Yes: 32
##
##
##
##
##
##  smokebf       age            educat            ethnic
##  No :88   Min.   :17.00   Min.   :14.00   White    :80
##  Yes:51   1st Qu.:25.00   1st Qu.:16.00   Non-white:59
##           Median :28.00   Median :17.00
##           Mean   :28.26   Mean   :18.15
##           3rd Qu.:32.00   3rd Qu.:19.00
##           Max.   :40.00   Max.   :38.00
##           NA's   :2       NA's   :2
```

The factors that could influence the decision are multiple:

- the advancement of the pregnancy (pregnancy)
- how the mothers were fed as babies (howfed)
- how the mother's friend fed their babies (howfedfr)
- if they have a partner (partner)
- their age (age)
- the age at which they left full-time education (educat)
- their ethnicity (ethnic)
- if they have ever smoked (smokebf)
- if they have stopped smoking (smokenow)

We can visualize the comparison of bottle vs breastfeeding:

```
plot(bf$breast)
```



As we can see, in general the majority of women prefer breastfeeding (100) over bottlefeeding (39). We now check for any missing values:

```
any(is.na(bf))
```

```
## [1] TRUE
```

Having NA values might be a problem for the outcome, but removing them could distort it. Therefore, we try to substitute all NA values with the mean of the column instead:

```
for(i in 1:ncol(bf)) {
  bf[ , i][is.na(bf[ , i])] <- mean(bf[ , i], na.rm = TRUE)
}
any(is.na(bf))
```

```
## [1] FALSE
```

```
summary(bf)
```

```
##     breast        pregnancy     howfed     howfedfr     partner    smokenow
##  Bottle: 39   End      :84   Bottle:59   Bottle:54   Single : 21   No :107
##  Breast:100   Beginning:55   Breast:80   Breast:85   Partner:118   Yes: 32
##
##
##
##
```

```
##  smokebf       age            educat            ethnic
##  No :88   Min.   :17.00   Min.   :14.00   White    :80
##  Yes:51   1st Qu.:25.00   1st Qu.:16.00   Non-white:59
##           Median :28.26   Median :17.00
##           Mean   :28.26   Mean   :18.15
##           3rd Qu.:32.00   3rd Qu.:19.00
##           Max.   :40.00   Max.   :38.00
```

The values remained unchanged, but now we don't have any NA value in the dataset.

We now split the data in the reproducible training and testing sets. The training data will contain 80% of the samples. To create the sample, we use caret's createDataPartition, which allows us to specify the percentage of the training data and automatically generates a random list of numbers, corresponding to the indexes of the data to use in the training set. We also set a cross validation control for our future computations:

```
set.seed(1)
sample <- caret::createDataPartition(bf$breast, p=0.8)
train_data <- bf[sample$Resample1, ]
test_data <- bf[-sample$Resample1, ]
ctrl <- trainControl(method="cv")
```

The first model to be fit is the GLM model

$$logit(E(breast)) = \beta 0 + \beta 1 pregnancy + \beta 2 howfed + \beta 3 howfedfr$$
$$+\beta 4 partner + \beta 5 age + \beta 6 educat + \beta 7 ethnic + \beta 8 smokenow + \beta 9 smokebf$$

To do this, we use caret's train function, which takes in input not only the function and the data, but also the method for the model (in this case "glm") and the training control. This latter parameter allows us to use the previously set cross validation:

```
glmFit <- train(breast ~ .,
                data = train_data,
                method = "glm",
                trControl = ctrl)
summary(glmFit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0412  -0.5951   0.2918   0.5232   2.6168
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.95225    2.20779  -1.337  0.18116
## pregnancyBeginning -0.99079    0.58930  -1.681  0.09270 .
## howfedBreast       -0.45042    0.64511  -0.698  0.48505
## howfedfrBreast      1.91433    0.61711   3.102  0.00192 **
## partnerPartner      1.01765    0.75844   1.342  0.17967
## smokenowYes        -2.57285    1.02263  -2.516  0.01187 *
## smokebfYes          1.27231    1.02106   1.246  0.21274
## age                 0.01964    0.05351   0.367  0.71356
## educat              0.09247    0.09884   0.936  0.34948
## `ethnicNon-white`   1.87420    0.73557   2.548  0.01084 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 134.012  on 111  degrees of freedom
## Residual deviance:  87.844  on 102  degrees of freedom
## AIC: 107.84
##
## Number of Fisher Scoring iterations: 5
```

By looking at the coefficients, we can see that the most significant one seems to be "howfedfr" for the value "Breast". This means that the way that a mother's friend feeds their baby seems to have the most influence on how she will feed hers. The other two coefficients that seem to have some significance, although lower, are "smokenow" for the value "Yes" and "ethnic" for the value "Non-white".

We can make the predictions for the test data and visualize both the confusion matrix and the accuracy of the model:

```
glm.probs <- predict(glmFit, test_data, type="prob")

glm.pred <- rep("Bottle", nrow(test_data))
glm.pred[glm.probs[,2] > 0.5] <- "Breast"

table(glm.pred, test_data$breast)
```

```
##
## glm.pred Bottle Breast
##   Bottle      5      4
##   Breast      2     16
```

```
(accGlm <- mean(glm.pred == test_data$breast))
```

```
## [1] 0.7777778
```

The accuracy for this model is ~77.8%.

Now we try to fit a k-nn classifier. For this, we will do a nested cross validation instead of a regular cross validation. We start by dividing our training data in multiple partitions to test. Since we don't have lots of observations available, we will use 5 folds for the outer layer:

```
set.seed(1)
trainK <- train_data
folds <- list()
nfolds <- 5
sampleLen <- round((nrow(trainK)/nfolds)) #here we round to the nearest lower integer
for (n in c(1:nfolds)) {
  sample <- sample.int(nrow(trainK), sampleLen, replace = FALSE)
  folds[paste0("fold", n)] <- list(trainK[sample, ])
  trainK <- trainK[-sample,]
}
#in this case we leave out one observation to keep the folds balanced
```

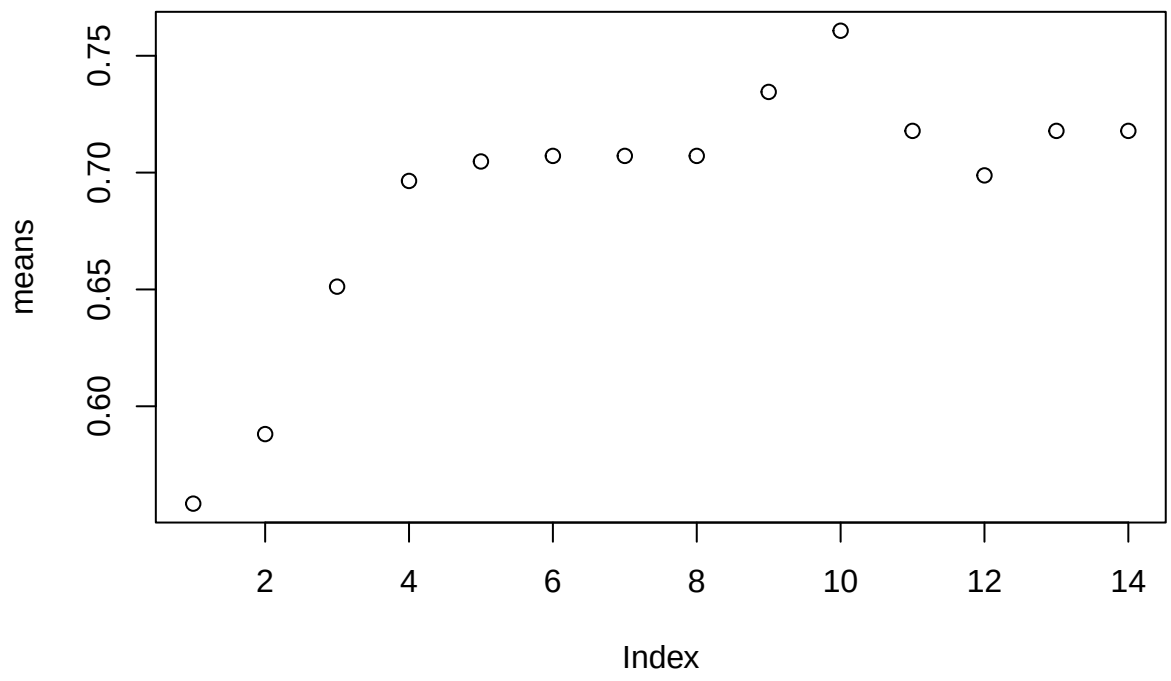
Then, we proceed following these steps:

- train a model for each fold with a k that varies between 1 and 14 (maximum limit of the train function)
- find out the mean accuracy of each k between each model
- choose the k based on this

```r
set.seed(1)
foldFits <- list()
i <- 1
kLim <- 14 #setting the maximum k
#training a model on each fold for all the ks
for (fold in folds){
  tmp <- train(breast ~ .,
               data = fold,
               method = "knn",
               trControl = trainControl(method="cv", number = 3),
               tuneGrid = expand.grid(k = 1:kLim))
  foldFits[paste0("fold", i)] <- list(tmp)
  i <- i + 1
}

accuracies <- list()
i <- 1
#saving the accuracies
for(fold in foldFits){
  accuracies[[i]] <- fold$results$Accuracy
  i <- i + 1
}

means <- c()
tmp <- 0
#calculating the mean for the accuracies
for(i in 1:kLim){
  for (acc in accuracies){
    tmp <- tmp + acc[i]
  }
  tmp <- tmp / nfolds
  means <- c(means, tmp)
  tmp <- 0
}
highest <- c(0,1)
#finding the best k
for(n in 1:kLim){
  if (means[n] > highest[1]){
    highest[1] = means[n]
    highest[2] = n
  }
}
plot(means)
```




```
print(paste("The best k is: ", highest[2]))
```

```
## [1] "The best k is:  10"
```

We can also visualize the confusion matrix for the predictions. In this case, we fit the model (with the best k) using tidyverse and tidymodels:

```
knn_spec <- nearest_neighbor(neighbors=highest[2]) %>%
    set_mode("classification") %>%
    set_engine("kknn")

knn_fit <- knn_spec %>%
    fit(breast ~ ., data = train_data)

augmented_ts <- augment(knn_fit, new_data=test_data)

augmented_ts %>%
    conf_mat(truth=breast, estimate=.pred_class)
```

```
##           Truth
## Prediction Bottle Breast
##     Bottle      3      2
##     Breast      4     18
```

And the accuracy:

```
(accKnn <- augmented_ts %>%
    accuracy(truth=breast, estimate=.pred_class))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.778
```

The accuracy here is ~77.8%, like the GLM model. Now we fit our last model, the Naïve Bayes classifier, once again using caret's train function with cross validation. Then, we visualize the confusion matrix and the accuracy:

```
fitNaive <- train(breast ~ .,
                  data = train_data,
                  method = "naive_bayes",
                  trControl = ctrl)
predNaive <- predict(fitNaive, test_data, type = "prob")
predFinal <- ifelse(predNaive[,2] >= 0.5, "Breast", "Bottle")

table(predFinal, test_data$breast)
```

```
##
## predFinal Bottle Breast
##    Bottle      4      3
##    Breast      3     17
```

```
(accNaive <- mean(predFinal == test_data$breast))
```

```
## [1] 0.7777778
```

The Naïve Bayes classifier has an accuracy of ~77.8%, same as the previous models. We now compare the precision/specificity of all the models:

```
specGlm <- specificity(table(glm.pred, test_data$breast))
specKnn <- specificity(table(augmented_ts$.pred_class, test_data$breast))
specNaive <- specificity(table(predFinal,test_data$breast))

(dfSpec <- data.frame(Model = c("GLM", "KNN", "NAIVE"),
                Specificity = c(specGlm, specKnn, specNaive)))
```

```
##   Model Specificity
## 1   GLM        0.80
## 2   KNN        0.90
## 3 NAIVE        0.85
```

Here, the GLM model scores the lowest, while the k-nn has the highest score. Let's visualize all the scores, together with the sensitivity:

```
sensGlm <- sensitivity(table(glm.pred, test_data$breast))
sensKnn <- sensitivity(table(augmented_ts$.pred_class, test_data$breast))
sensNaive <- sensitivity(table(predFinal,test_data$breast))

dfFinal <- data.frame(GLM = c(accGlm, specGlm, sensGlm),  KNN = c(accKnn$.estimate, specKnn, sensKnn), NAIVE
rownames(dfFinal) <- c("Accuracy","Specificity", "Sensitivity")
dfFinal
```

```
##                   GLM       KNN     NAIVE
## Accuracy    0.7777778 0.7777778 0.7777778
## Specificity 0.8000000 0.9000000 0.8500000
## Sensitivity 0.7142857 0.4285714 0.5714286
```

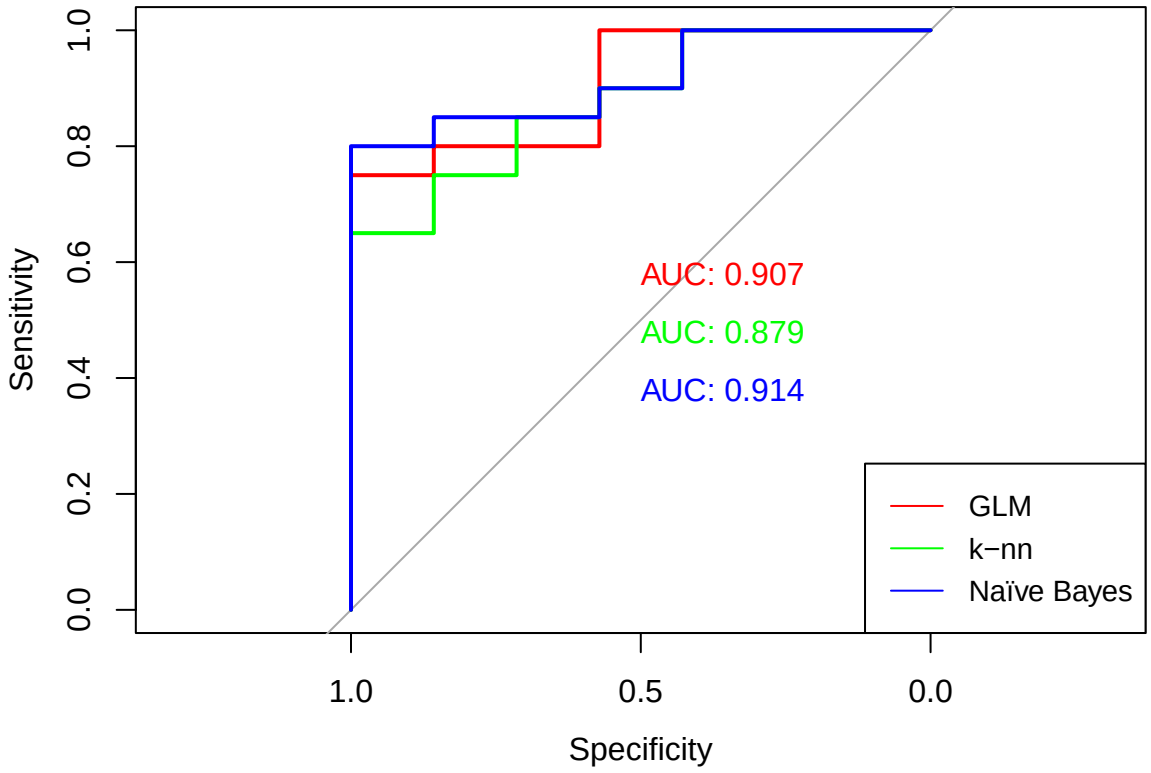
Finally, we plot the ROC curve for all the models and compare them visually. For this, we will use pROC's "roc" function. In the plot, we also visualize the AUC score. The AUC score, which stands for "Area Under Curve", is yet another measure to help us determine how well a model classifies data. Therefore, for the scope of this analysis, we can use it to decide which model is the best:

```
glmRoc = roc(test_data$breast ~ glm.probs[,2])
plot.roc(glmRoc, print.auc = TRUE, col = "red", print.auc.y = 0.6)

knnRoc = roc(test_data$breast ~ augmented_ts$.pred_Breast)
plot.roc(knnRoc, add = TRUE, print.auc = TRUE, col="green")

bayRoc = roc(test_data$breast ~ predNaive[,2])
plot.roc(bayRoc, add = TRUE, print.auc = TRUE, col="blue", print.auc.y = 0.4)

par(cex = 0.9)
legend("bottomright", legend=c("GLM", "k-nn", "Naïve Bayes"), col=c("red", "green", "blue"),
       lty=c(1, 1, 1))
```



From the ROC curves, we can see that the Naïve Bayes is the one with the better AUC score, the GLM is right behind it with only 0.07 points of difference and the k-nn model performs the worst but, once again, not by much. Overall, all the models seem very close.

To recap:

- the accuracy is the same for all models
- the k-nn model has the best specificity
- the GLM model has the best sensitivity
- the Naïve Bayes model has the better AUC score in the ROCR curve analysis, but the GLM seems very close

In the end, the GLM seems the better one, especially since the AUC is very close to the Naïve Bayes but the sensitivity is much better.