Homework 1

Antonio Padalino

2022-03-28

Contents

Homework 1

```
library('tidyverse')
library('tidymodels')
library('gridExtra')
library('ggrepel')
library('e1071')
library('kknn')
library('MASS')
```

The dataset we are about to discuss comes from a study conducted at a UK hospital, investigating the possible factors affecting the decision of pregnant women to breastfeed their babies, in order to target breastfeeding promotions towards women with a lower probability of choosing it.

Thus, the aim of the following examination will be to design and fit a statistical model able to predict, given a number of factors, whether or not a woman is likely to breastfeed her baby.

1. Exploratory analysis

First, we load the `breastfeed` dataset and take a look at some descriptive summary statistics.

For the study, responses were classified into 2 categories, referred to by the variable `breast` in the dataset.

The possible factors available are:

- `pregnancy`: the advancement of the pregnancy [2-level factors];
- `howfed`: how the mothers were fed as babies [2-level factors];

- `howfedfrhow`: the mother's friend fed their babies [2-level factors];
- `partner`: if they have a partner [2-level factors];
- `age` their age [numeric];
- `educat`: the age at which they left full-time education [numeric];
- `ethnic`: their ethnic group [2-level factors];
- `smokebf`: and if they have ever smoked [2-level factors];
- `smokenow`: or if they have stopped smoking [2-level factors]

```
load('breastfeed.Rdata')
df <- breastfeed

head(df)
```

breast	pregnancy	howfed	howfedfr	partner	smokenow	smokebf	age	educat	ethnic
Breast	Beginning	Breast	Breast	Partner	No	No	24	19	Non-white
Breast	Beginning	Bottle	Breast	Partner	No	No	27	18	White
Bottle	Beginning	Breast	Breast	Partner	No	No	39	16	White
Bottle	Beginning	Breast	Breast	Partner	Yes	Yes	29	16	White
Breast	Beginning	Breast	Breast	Partner	No	No	21	21	White
Bottle	Beginning	Breast	Bottle	Partner	No	No	NA	28	White

```
summary(df)
```

```
##     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
```

There are some issues we can already point out:

- the target `breast` is quite unbalanced (36 Bottle vs. 99 Breast)
- we have 2 NA values in the `educat` column and 2 in the `age` columns

Also, we might want to check for the presence of outliers in the numeric variables `age` and `educat` .

Such issues will be addressed in the upcoming section.

2. Data preparation

2.1 Dealing with NA values

First, we inspect NA values in our dataset.

```
df[rowSums(is.na(df)) > 0, ]
```

	breast	pregnancy	howfed	howfedfr	partner	smokenow	smokebf	age	educat	ethnic
6	Bottle	Beginning	Breast	Bottle	Partner	No	No	NA	28	White
22	Bottle	End	Breast	Bottle	Partner	No	No	31	NA	Non-white
46	Bottle	End	Bottle	Bottle	Partner	No	No	38	NA	White
125	Breast	End	Breast	Bottle	Partner	No	No	NA	16	White

We have at least 2 possibilities:

1. we could simply drop the rows containing NA values
2. we could replace the NA values with some statistics, such as mean or median

3 out of 4 NA rows are of `Bottle`-class, the factor that is least present in the `breast` column. Having noticed that, we might want to preserve our NA rows so that our models will be fitted with the maximum number of observations available.

We opt for the second way then, and we replace each NA value with the mean of its columns, filtered by its `breast` class. We do that in order to recreate a more representative scenario, in which missing values are represented by the average of the specific class of the observation.

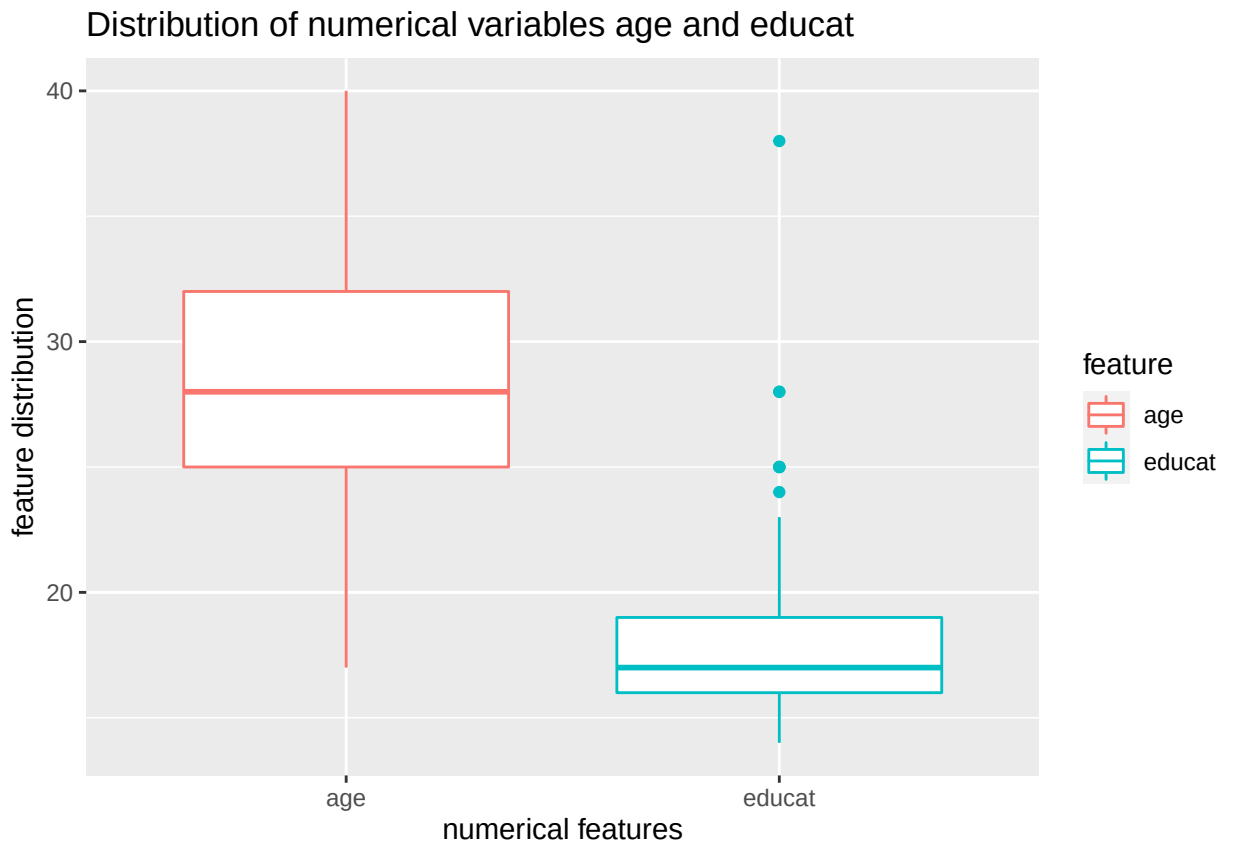
```
df <- df %>%
  group_by(breast) %>%
  mutate(age = ifelse(is.na(age), mean(age, na.rm = TRUE), age)) %>%
  mutate(educat = ifelse(is.na(educat), mean(educat, na.rm = TRUE), educat))

df <- ungroup(df)
```

2.2 Dealing with outliers

Another useful step might be to check for the presence of outliers. let's inspect the distribution of the 2 numeric variables of the dataset - `age` and `educat` - with boxplots.

```
to_plot <- data.frame(
  value = c(df$age,df$educat),
  feature = c(
    rep('age',length(df$age)),
    rep('educat',length(df$educat))
  )
)

ggplot(to_plot, aes(x=feature, y=value, col=feature)) +
  geom_boxplot() +
```

```
  labs(title = 'Distribution of numerical variables age and educat',
       x = 'numerical features',
       y = 'feature distribution')
```



As we can see, the `educat` variable has clearly at least one outlier.

To compute such outliers, we use the Inter-Quantile Range approach.

We compute the 1st quantile, the 3rd quantile, and the IQR of variable `educat` through functions quantile() and IQR(), and then we define the normal data range with limits:

$$[\ Q_1 - \delta * IQR,\ Q_3 + \delta * IQR\ ]$$

With δ being 2 - instead of the classic 1.5 - since we want to throw away as less data as we can.

Any data point outside such range is to be considered outlier and should be removed for further analysis.

```
Q1 <- quantile(df$educat,.25)
Q3 <- quantile(df$educat,.75)
IQR <- IQR(df$educat)

outliers <- df$educat[df$educat > Q3+2*IQR]

cat('Outliers for educat variable are:',outliers)
```

```
## Outliers for educat variable are: 28 38 28
```

```
df <- filter(df, !educat %in% outliers)
```

2.2 Data splitting

Before going any further performing random operations, we set a seed to make our results reproducible.

```
seed = 999
set.seed(seed)
```

For the sake of this examination, we are going to adopt the following approach:

1. split dataset into a **train set** and a **test set**;
2. perform **model selection** for each model using **train set alone**;
3. lastly, **evaluate each model** performance on the **test set**

Notice how the test subset is left aside from the beginning and it is not used in model selection operations, so to have a common (and untouched!) common ground to fairly evaluate the three models in the end.

We need to split our dataset into train and test subset before fitting any model, yet we shouldn't overlook the imbalance of the `breast` class. We might want either to:

1. split the dataset manually, to preserve the same imbalance in the train-test subsets;
2. use a dedicated function to do so, such as `rsample::initial_split()`

We go for the second option, as follows:

```
split <- initial_split(df, strata=breast, prop=0.7)
df_train <- training(split)
df_test <- testing(split)
```

```
table(df$breast)/nrow(df)
```

```
##
##    Bottle    Breast
## 0.2720588 0.7279412
```

```
table(df_train$breast)/nrow(df_train)
```

```
##
##    Bottle    Breast
## 0.2659574 0.7340426
```

```
table(df_test$breast)/nrow(df_test)
```

```
##
##    Bottle    Breast
## 0.2857143 0.7142857
```

We can quickly notice the imbalance between classes remained almost the same.

3. Models fitting

We're now about to fit 3 different models to our dataset:

- Logistic Regression
- k-Nearest Neighbours
- Naive Bayes

Before diving further, it might be useful to design a a simple function to 'evaluate' our models.

The `eval_model` function compute accuracy, error rate, sensitivity and specificity of our model and return everything in a dataframe for a tidy visualization.

```
eval_model <- function(model, df_test, nb=F) {

  #' Given the fact that a Naive Bayes model object come from a different package
  #' - not tidymodels - slightly different steps are required to make and store
  #' predictions, hence the necessity for a 'nb' flag in the function.

  if (nb == T) {
    augmented <- cbind(
      df_test,
      .pred_class = predict(model, newdata = df_test, type = 'class'),
      .pred_Bottle = predict(model, newdata = df_test, type = 'raw')[,1],
      .pred_Breast = predict(model, newdata = df_test, type = 'raw')[,2])
  }

  if (nb == F) {
    augmented <- cbind(
      df_test,
      predict(model, new_data = df_test, type = 'class'),
      predict(model, new_data = df_test, type = 'prob'))
  }

  acc <- augmented %>%
    accuracy(truth = breast, estimate = .pred_class) %>%
    pull(.estimate)

  sen <- augmented %>%
    sens(truth = breast, estimate = .pred_class) %>%
    pull(.estimate)

  spe <- augmented %>%
    spec(truth = breast, estimate = .pred_class) %>%
    pull(.estimate)

  results <- data_frame(
    accuracy = acc,
    error = 1 - acc,
    sensitivity = sen,
    specificity = spe
  )
```

```
  return(results)

}
```

3.1 Logistic regression

Let's start with a Logistic Regression model, a classification model belonging to the class of generalized linear models. The most immediate model we could fit is:

$$logit\big(E(breast)\big) = \beta_0 + \beta_1\ pregnancy + \beta_2\ howfed + \beta_3\ howfedfr +$$
$$+\beta_4\ partner + \beta_5\ age + \beta_6\ educat + \beta_7\ ethnic + \beta_8\ smokenow + \beta_9\ smokebf$$

However, using all predictors does not always guarantee to fit the best model.

A. Model Selection

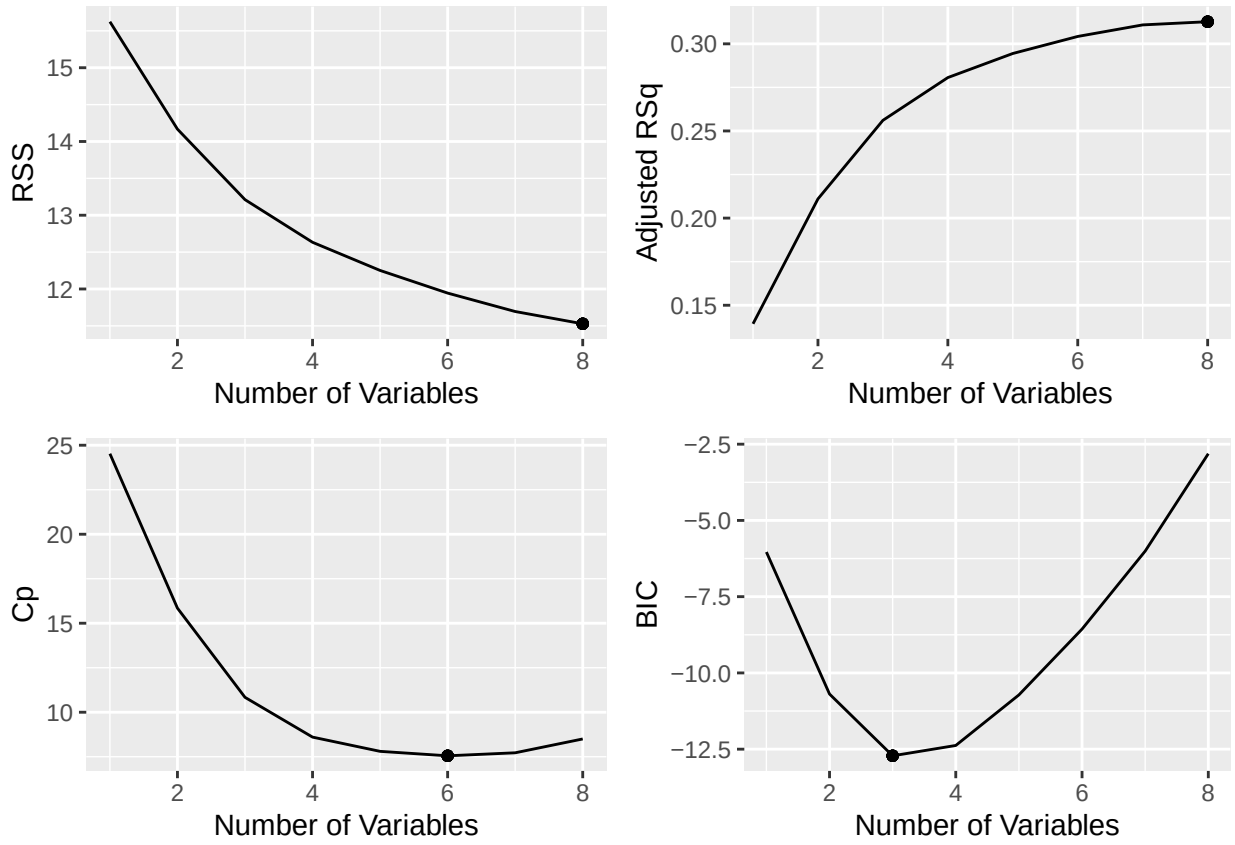
We might want to check which is the best subset of predictors to use to fit our Logistic Regression model with. In order to do this, we take advantage of the `regsubsets()` function, and then we plot some useful statistics such as R^2, RSS, adjusted R^2, Mallow's C_p, BIC over the number of variables.

```
library('leaps')

regfit.full <- regsubsets(breast ~ ., data=df_train)
mod.summary <- summary(regfit.full)

mod.summary
```

```
## Subset selection object
## Call: regsubsets.formula(breast ~ ., data = df_train)
## 9 Variables  (and intercept)
##                     Forced in Forced out
## pregnancyBeginning     FALSE      FALSE
## howfedBreast           FALSE      FALSE
## howfedfrBreast         FALSE      FALSE
## partnerPartner         FALSE      FALSE
## smokenowYes            FALSE      FALSE
## smokebfYes             FALSE      FALSE
## age                    FALSE      FALSE
## educat                 FALSE      FALSE
## ethnicNon-white        FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          pregnancyBeginning howfedBreast howfedfrBreast partnerPartner
## 1  ( 1 ) " "                " "          " "             " "
## 2  ( 1 ) " "                " "          " "             " "
## 3  ( 1 ) " "                " "          "*"             " "
## 4  ( 1 ) " "                " "          "*"             "*"
## 5  ( 1 ) " "                " "          "*"             "*"
## 6  ( 1 ) "*"                " "          "*"             "*"
## 7  ( 1 ) "*"                " "          "*"             "*"
## 8  ( 1 ) "*"                " "          "*"             "*"
##          smokenowYes smokebfYes age educat ethnicNon-white
```

```
## 1  ( 1 ) "*"         " "         " " " " "    " "
## 2  ( 1 ) "*"         " "         " " " " "    "*"
## 3  ( 1 ) "*"         " "         " " " " "    "*"
## 4  ( 1 ) "*"         " "         " " " " "    "*"
## 5  ( 1 ) "*"         "*"         " " " " "    "*"
## 6  ( 1 ) "*"         "*"         " " " " "    "*"
## 7  ( 1 ) "*"         "*"         " " "*" "    "*"
## 8  ( 1 ) "*"         "*"         "*" "*" "    "*"
```

```
to_plot <- data.frame(
  nvariables = seq(1:8),
  rss = mod.summary$rss,
  adjr2 = mod.summary$adjr2,
  bic = mod.summary$bic,
  cp = mod.summary$cp
)

nv1 <- which.min(mod.summary$rss)

g1 <- ggplot(to_plot, aes(x=nvariables, y=rss)) +
  geom_line() +
  geom_point(aes(x=nv1,y=mod.summary$rss[nv1])) +
  labs(y="RSS", x="Number of Variables")

nv2 <- which.max(mod.summary$adjr2)

g2 <- ggplot(to_plot, aes(x=nvariables, y=adjr2)) +
  geom_line() +
  geom_point(aes(x=nv2,y=mod.summary$adjr2[nv2])) +
  labs(y="Adjusted RSq", x="Number of Variables")

nv3 <- which.min(mod.summary$cp)

g3 <- ggplot(to_plot, aes(x=nvariables, y=cp)) +
  geom_line() +
  geom_point(aes(x=nv3,y=mod.summary$cp[nv3])) +
  labs(y="Cp", x="Number of Variables")

nv4 <- which.min(mod.summary$bic)

g4 <- ggplot(to_plot, aes(x=nvariables, y=bic)) +
  geom_line() +
  geom_point(aes(x=nv4,y=mod.summary$bic[nv4])) +
  labs(y="BIC", x="Number of Variables")

gridExtra::grid.arrange(g1, g2, g3, g4, nrow=2)
```



Inspecting the plots, we notice that the best values for the different indicators correspond to quite different numbers of variables used. Let's make some observations:

- the RSS **decreases monotonically** with the number of included variables;
- the R^2 **increases monotonically** with the number of included variables

This means we cannot rely on RSS or R^2 alone for selecting the optimal model.

A much more reliable indicator is the *Bayes Information Criterion*, which shows a quite clear minimum in 3.

Going back to the summary of `regsubsets()`, we observe that the best LR model with 3 variables is:

$$logit\big(E(breast)\big) = \beta_0 + \beta_1\ pregnancy + \beta_2\ howfedfr + \beta_3 smokenow$$

B. Model Evaluation

We can now fit the Logistic Regression model, with the set of predictors we have chosen.

```
log_reg <- logistic_reg() %>%
    set_engine("glm") %>%
    set_mode("classification")

lr_mod <- log_reg %>%
    fit(breast ~ pregnancy + howfed + smokenow, data = df_train)
```

```
lr_mod_all <- log_reg %>%
    fit(breast ~ ., data = df_train)

tidy(lr_mod)
```

term	estimate	std.error	statistic	p.value
(Intercept)	1.1800923	0.4303765	2.742000	0.0061066
pregnancyBeginning	-0.5521063	0.5440050	-1.014892	0.3101573
howfedBreast	1.4355438	0.5569081	2.577703	0.0099459
smokenowYes	-1.8811492	0.5638709	-3.336135	0.0008495

We evaluate it with the `eval_model()` function we declared before:

```
lr_results <- eval_model(lr_mod, df_test)

lr_results
```

accuracy	error	sensitivity	specificity
0.7857143	0.2142857	0.25	1

3.2 k-Nearest Neighbours

We will now fit a k-NN model on our dataset. The number of 'neighbours' to consider fo this process, which goes by the parameter k, is to be select carefully before training our final model.

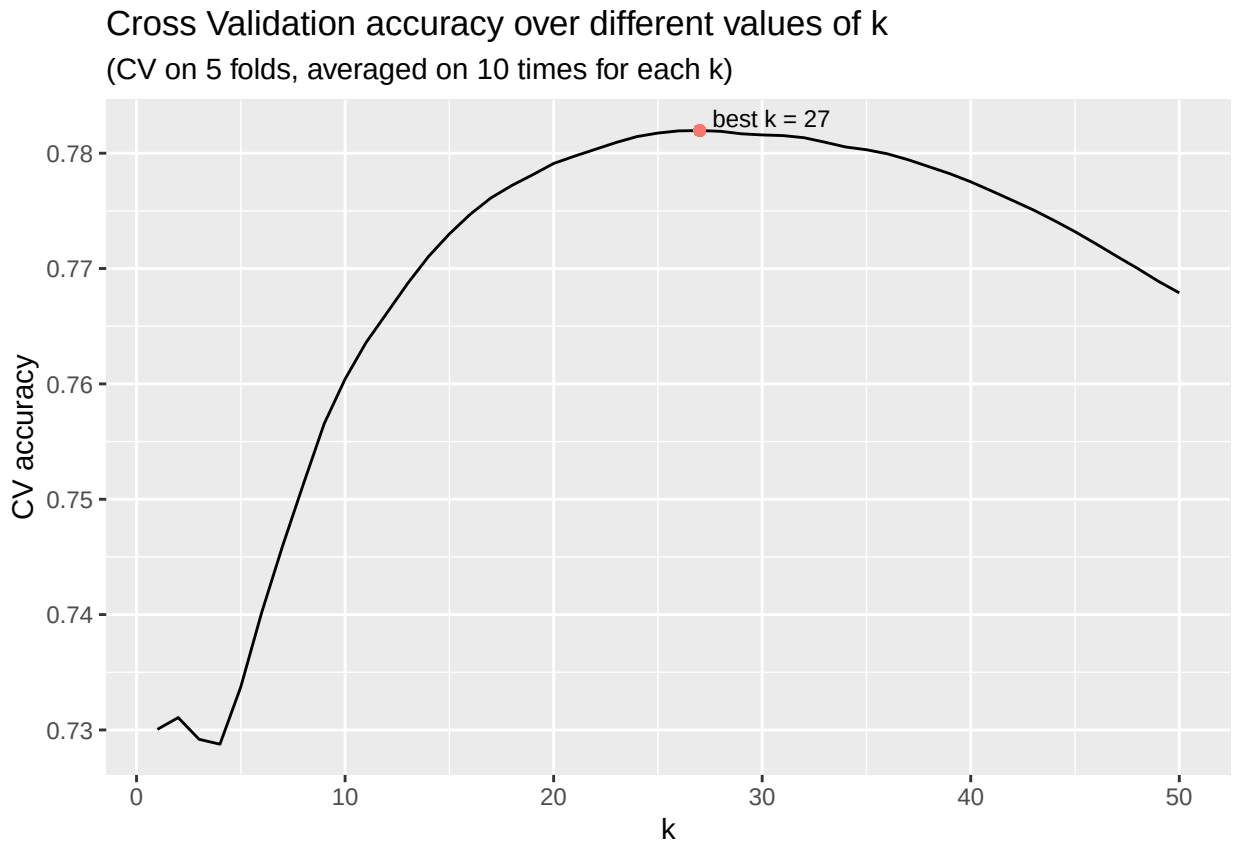
A. Model selection

We will perform cross validation on $k = 5$ folds (on the train dataset alone) for different values of k to see which one is the optimal one. And later, with that k, we will fit our model and compute its accuracy on the test dataset for the evaluation part. We are also going to plot the CV accuracies for all tested values of k, to show clearly where the curve reach its maximum.

The accuracies we will get in Cross Validation will of course depend on the random split performed by R. For this reason (and also because of the quite small size of the dataset) we expect this curve to be kind of ragged, meaning that bad decisions on the best k could have been made. To prevent this, we are going to perform the whole CV process 10 times for each value of k.

```
kmax <- 50      # max k to test
kf <- 5         # number of folds for CV
n_times <- 10   # times CV is performed

best_acc <- 0   # to store the best accuracy
best_k <- 0     # to store the best k

all_cv_acc <- vector()
all_cv_accs <- vector()

for (k in 1:kmax) {  # cycle through different k

  rep_cv_acc <- vector()
```

```
  for (t in 1:n_times) {  # perfrom n_times the whole CV process

    folds_ids <- caret::createFolds(df_train$breast, k = kf, list = TRUE, returnTrain = TRUE)

    for (f in names(folds_ids)) {  # fit, and compute acc. for each fold

      df_train_subtrain <- df[folds_ids[[f]],]
      df_train_subtest <- df[-folds_ids[[f]],]

      knn <- nearest_neighbor(neighbors = k) %>%
        set_mode("classification") %>%
        set_engine("kknn")

      knn_mod <- knn %>%
        fit(breast ~ ., data = df_train_subtrain)

      acc <- eval_model(knn_mod, df_train_subtest)$accuracy
      all_cv_acc <- c(all_cv_acc,acc)

    }

    cv_acc <- mean(all_cv_acc)  # average the accuracies of all folds
    rep_cv_acc <- c(rep_cv_acc,cv_acc)

  }

  true_cv_acc <- mean(rep_cv_acc)  # average the mean accuracy of CV for a certain k
  all_cv_accs <- c(all_cv_accs,true_cv_acc)

  if (true_cv_acc >= best_acc) {
    best_acc <- true_cv_acc  # store best CV accuracy
    best_k <- k  # store best k so far

  }

}

### Plot CV accuracies over different k

to_plot <- data.frame(k = seq(1:kmax), cv_acc = all_cv_accs)

ggplot(to_plot, aes(x=k, y=cv_acc)) +
  geom_line() +
  geom_point(aes(best_k, best_acc, col = 'red')) +
  geom_text_repel(data=filter(to_plot, k==best_k),
                  aes(label=paste('best k =',best_k)),
                  size=3,
                  segment.size = .25) +
  scale_color_discrete(guide=FALSE) +
  labs(title = "Cross Validation accuracy over different values of k",
       subtitle = "(CV on 5 folds, averaged on 10 times for each k)",
       y = 'CV accuracy')
```



The best parameter is clearly shown in the figure above. Using such k we can proceed to the evaluation.

B. Model Evaluation

Knowing the optimal value for k, we can fit our model on the whole train dataset.

```
knn <- nearest_neighbor(neighbors = best_k) %>%
    set_mode("classification") %>%
    set_engine("kknn")

knn_mod <- knn %>%
    fit(breast ~ ., data = df_train)

knn_acc <- mean(
  predict(knn_mod, df_test)$.pred_class == df_test$breast
)

knn_mod
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = breast ~ ., data = data, ks = min_rows(27L,     data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.2021277
```

```
## Best kernel: optimal
## Best k: 27
```

```
knn_results <- eval_model(knn_mod, df_test)

knn_results
```

accuracy	error	sensitivity	specificity
0.8333333	0.1666667	0.4166667	1

3.3 Naive Bayes

Lastly, we want to fit a Naive Bayes model.

A Naive Bayes Classificator is a SL algorithm based on the Bayes Theorem, funded upon the assumption that the predictors in the model are independent from each other.

For such reason, we cannot make any model selection in this case.

```
nb_mod <- naiveBayes(breast ~ ., data = df_train)

nb_mod
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##    Bottle    Breast
## 0.2659574 0.7340426
##
## Conditional probabilities:
##         pregnancy
## Y              End Beginning
##   Bottle 0.5600000 0.4400000
##   Breast 0.6811594 0.3188406
##
##         howfed
## Y           Bottle    Breast
##   Bottle 0.7200000 0.2800000
##   Breast 0.3913043 0.6086957
##
##         howfedfr
## Y           Bottle    Breast
##   Bottle 0.6800000 0.3200000
##   Breast 0.3188406 0.6811594
##
##         partner
## Y           Single   Partner
```

```
##   Bottle 0.3200000 0.6800000
##   Breast 0.1594203 0.8405797
##
##         smokenow
## Y              No       Yes
##   Bottle 0.4800000 0.5200000
##   Breast 0.8550725 0.1449275
##
##         smokebf
## Y              No       Yes
##   Bottle 0.4400000 0.5600000
##   Breast 0.6956522 0.3043478
##
##         age
## Y           [,1]     [,2]
##   Bottle 28.96000 6.160898
##   Breast 27.90514 5.459188
##
##         educat
## Y           [,1]     [,2]
##   Bottle 16.59568 1.935619
##   Breast 18.43478 2.648408
##
##         ethnic
## Y           White Non-white
##   Bottle 0.9200000 0.0800000
##   Breast 0.5217391 0.4782609
```

We can evaluate our model with the usual function.

```
nb_results <- eval_model(nb_mod, df_test, nb=T)

nb_results
```

accuracy	error	sensitivity	specificity
0.9047619	0.0952381	0.75	0.9666667

4. Conclusions and wrap up

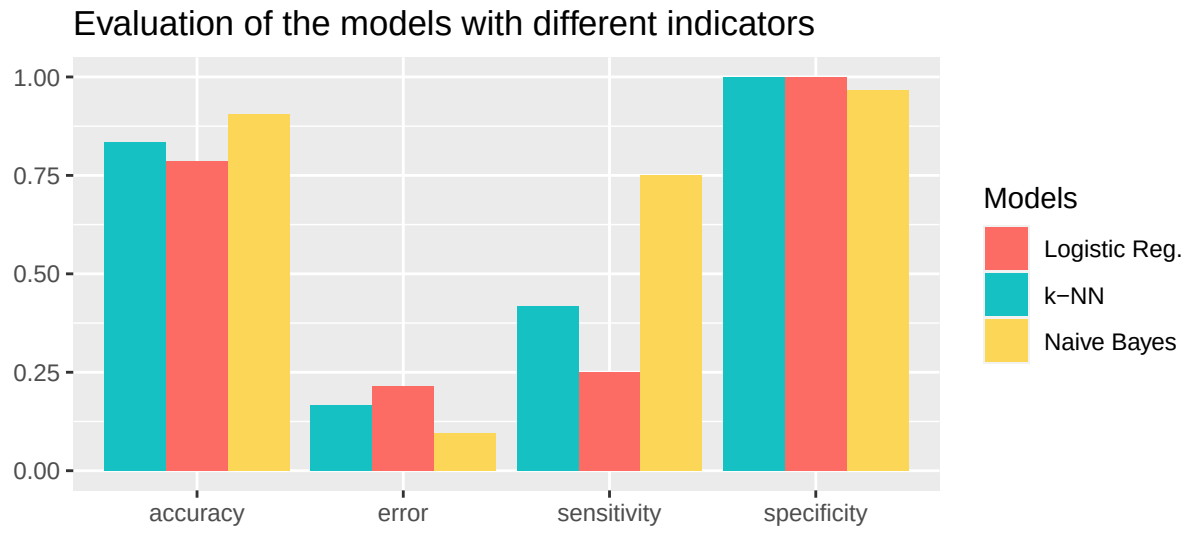
It might be useful to wrap all the results on previous models in a single dataframe, for a tidier visualization.

```
all_res <- rbind(lr_results,
                 knn_results,
                 nb_results)

model = c('Logistic Reg.', 'k-NN', 'Naive Bayes')

all_res <- add_column(all_res, model, .before = 1)

all_res %>% arrange(-accuracy)
```

model	accuracy	error	sensitivity	specificity
Naive Bayes	0.9047619	0.0952381	0.7500000	0.9666667
k-NN	0.8333333	0.1666667	0.4166667	1.0000000
Logistic Reg.	0.7857143	0.2142857	0.2500000	1.0000000

To make it more informative (and visually appealing!), we can plot a simple bar chart to show our results.

```
to_plot <-  data.frame(
  value = c(all_res$accuracy, all_res$error, all_res$sensitivity, all_res$specificity),
  model = rep(c('Logistic Reg.','k-NN','Naive Bayes'),4),
  indic = c(rep('accuracy',3),rep('error',3),rep('sensitivity',3),rep('specificity',3))
)

ggplot(to_plot, aes(fill=model, y=value, x=indic)) +
  geom_bar(position="dodge", stat="identity") + theme(aspect.ratio = 0.5) +
  labs(title='Evaluation of the models with different indicators', x = '', y = '') +
  scale_fill_manual("Models", values = c('Logistic Reg.' = '#FC6C64',
                                         'k-NN' = '#15C1C2',
                                         'Naive Bayes' = '#FCD757'))
```



As we can see, the model which scored the best accuracy on the test dataset is the Naive Bayes model.

Notice that the outcome of such comparison may significantly vary over different re-runs of the code with different seeds (totally normal), bringing to a different choice of a best model (less normal).

This is probably due to:

- the small size of the dataset;
- the fact that overall the 3 models analyzed, overall, performs similarly.

Beside accuracy and error (which is 1-accuracy) we also evaluated sensitivity and specificity for each model.

Recall that:

- **Sensitivity** is the proportion of true positives that are correctly identified by a test, thus it shows how good the test is at detecting a certain class
- **Specificity** is the proportion of the true negatives that are correctly identified by a test, thus it suggests how good the test is at identifying a negative condition

Beyond Naive Bayes, which displays both a high specificity and a quite high sensitivity, Logistic Regression and k-NN shows high sensitivity and low specificity. In other words, they are good for catching actual `breast` classes but they also come with a fairly high rate of false positives.

To the extent of our goal (targeting breastfeeding promotions towards women with a lower probability of choosing it) LR and k-NN might not prove to be the best models on the business, since they come with a higher probability of 1) catching `breast` classes and 2) misclassifying `bottle` classes (our actual target).

Relying purely on the result of this examination,
we may suggest that a Naive Bayes model is the best overall choice.