Christine Pallon - Model 1

2022-12-01

Loading and pre-processing data

This first model uses kNN, an **ensemble classification model**, using the radiomics dataset. First, we need to load and pre-process the dataset.

```
## LOADING DATA ##

radiomics <- read.csv("radiomics_completedata.csv")


## CHECKING FOR NULL VALUES AND PREPARING DATA ##

is.null(radiomics)
```

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

```
radiomics <- radiomics %>% select(-Institution, -Failure)
radiomics$Failure.binary <- as.factor(radiomics$Failure.binary)
radiomics$Failure.binary
```

```
##   [1] 0 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1
##  [38] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0
##  [75] 0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0
## [112] 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 0 1 0 0 1 0 0
## [149] 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 1 0 0 0
## [186] 0 1 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

Then, we change the Failure.binary levels from 0 and 1 to No and Yes to prevent an error regarding class levels when running kNN.

```
levels(radiomics$Failure.binary)=c("No","Yes")

# Double-checking response variable after changing levels ##

head(radiomics$Failure.binary)
```

```
## [1] No  Yes No  Yes No  Yes
## Levels: No Yes
```

Correlation

Then, we get the correlation of the whole dataset except the categorical variables. The output of this has been hidden due to its length.

```
radiomics_non_categorical <- radiomics %>% select(-Failure.binary)
radiomics_correlation <- as.data.frame(cor(radiomics_non_categorical))
radiomics_correlation
```

Setting up the kNN model

Now, it's time to split the data into an 80/20 training/testing split, prepare blueprints, create a resampling method, and create a hyperparameter grid search.

```
radiomics <- radiomics %>% mutate_if(is.ordered, factor, ordered = FALSE)

set.seed(123)
radiomics_split <- initial_split(radiomics, prop = .8, strata = "Failure.binary")
radiomics_train <- training(radiomics_split)
radiomics_test  <- testing(radiomics_split)

blueprint_1 <- recipe(Failure.binary ~ ., data = radiomics_train) %>%
  step_nzv(all_nominal()) %>%
  step_integer(Entropy_cooc.W.ADC) %>%
  step_integer(GLNU_align.H.PET) %>%
  step_integer(Min_hist.PET) %>%
  step_dummy(all_nominal(), -all_outcomes(), one_hot = TRUE) %>%
  step_center(all_numeric(), -all_outcomes()) %>%
  step_scale(all_numeric(), -all_outcomes())

blueprint_2 <- recipe(Failure.binary ~ ., data = radiomics_test) %>%
  step_nzv(all_nominal()) %>%
  step_integer(Entropy_cooc.W.ADC) %>%
  step_integer(GLNU_align.H.PET) %>%
  step_integer(Min_hist.PET) %>%
  step_dummy(all_nominal(), -all_outcomes(), one_hot = TRUE) %>%
  step_center(all_numeric(), -all_outcomes()) %>%
  step_scale(all_numeric(), -all_outcomes())

cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)


hyper_grid <- expand.grid(
  k = floor(seq(1, nrow(radiomics_train)/3, length.out = 20))
)
```

Fitting the model using the training data

Now it's time to fit our kNN model in the training phase.

```
knn_grid_train <- train(
  blueprint_1,
  data = radiomics_train,
  method = "knn",
  trControl = cv,
  tuneGrid = hyper_grid,
  metric = "ROC"
)
```

Printing Top 20 important features during the training phase

Using varImp() to print the Top 20 most importan features during the training phase.

```
vi <- varImp(knn_grid_train)
print(vi)
```

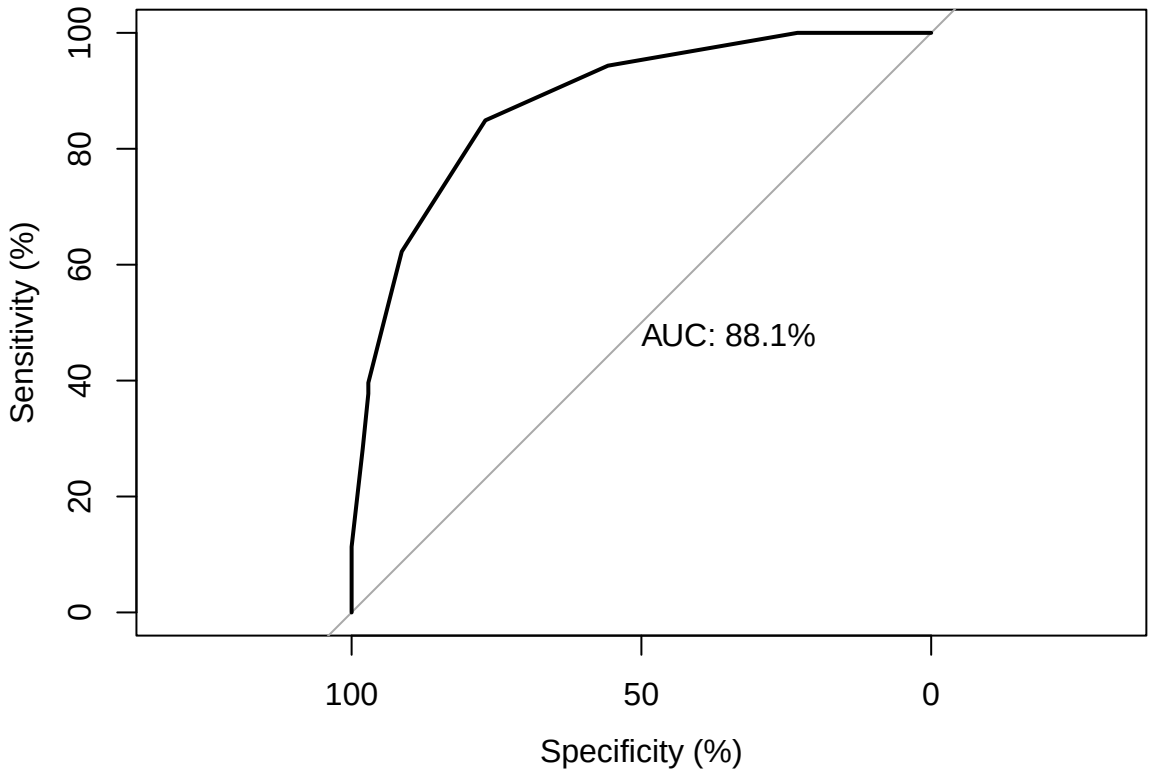
```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 428)
##
##                               Importance
## Entropy_cooc.W.ADC                100.00
## GLNU_align.H.PET                   55.01
## Min_hist.ADC                       53.25
## Complexity_vdif.W.ADC              48.82
## Autocorrelation_cooc.W.ADC         46.23
## DVAR_cooc.L.PET                    43.26
## Prominence_cooc.W.ADC              42.57
## SZHGE.W.ADC                        42.38
## HGSRE_align.W.ADC                  42.30
## Entropy_align.W.ADC                42.17
## HGLZE.W.ADC                        42.09
## HGRE_align.W.ADC                   41.88
## HGLRE_align.W.ADC                  41.25
## LZHGE.W.ADC                        40.25
## DVAR_cooc.W.ADC                    40.04
## Entropy_cooc.H.PET                 38.96
## Contrast_cooc.W.ADC                38.87
## SAVE_cooc.W.ADC                    37.83
## Average_cooc.W.ADC                 36.53
## Variance_cooc.W.ADC                34.78
```

Printing the AUC values during the training phase

Finally, before moving onto the testing phase, we print the AUC values.

```r
pred_knn_train <- predict(knn_grid_train, radiomics_train, type = "prob")

roc(radiomics_train$Failure.binary ~ pred_knn_train[,2], plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```




```
##
## Call:
## roc.formula(formula = radiomics_train$Failure.binary ~ pred_knn_train[,     2], plot = TRUE, legacy.a
##
## Data: pred_knn_train[, 2] in 104 controls (radiomics_train$Failure.binary No) < 53 cases (radiomics_t
## Area under the curve: 88.05%
```

Fitting the model using the testing data

Now, we fit our kNN model using the testing data.

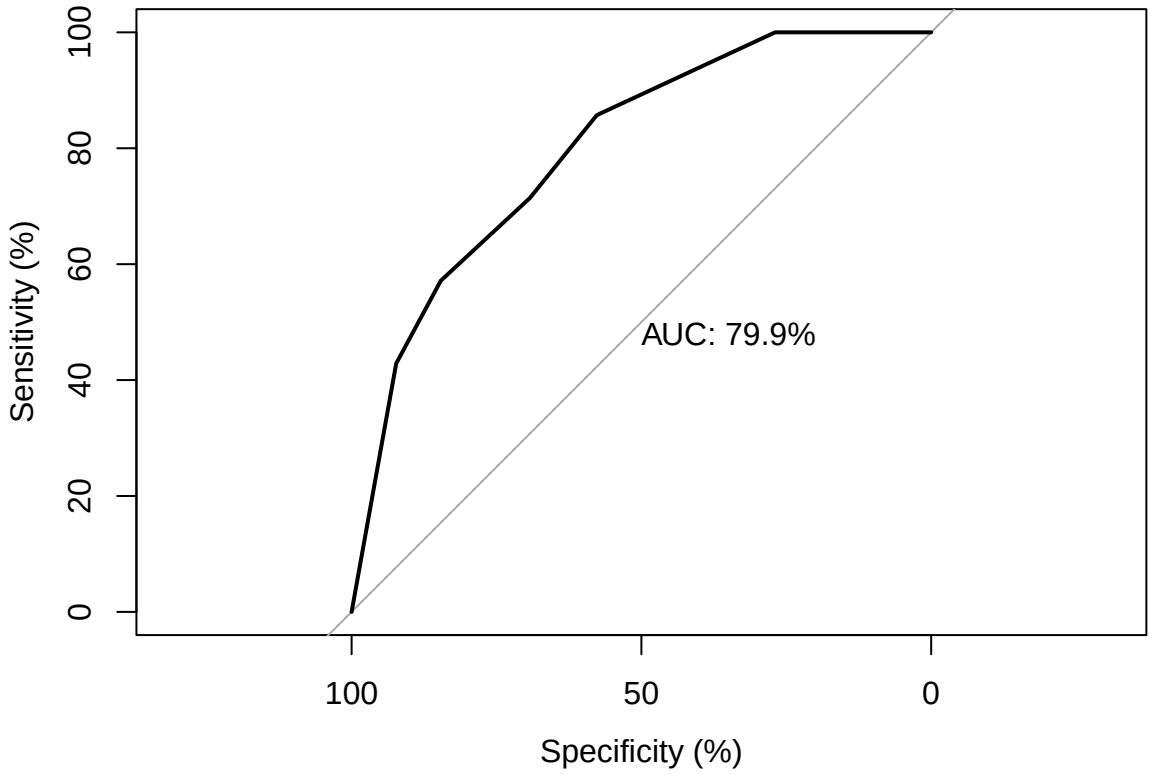
```r
knn_grid_test <- train(
  blueprint_2,
  data = radiomics_test,
  method = "knn",
  trControl = cv,
  tuneGrid = hyper_grid,
  metric = "ROC"
)
```

Printing the AUC values during the training data

Finally, we print the AUC values during the training phase.

```
pred_knn_test <- predict(knn_grid_test, radiomics_test, type = "prob")

roc(radiomics_test$Failure.binary ~ pred_knn_test[,2], plot=TRUE, legacy.axes=FALSE,
    percent=TRUE, col="black", lwd=2, print.auc=TRUE)
```




```
##
## Call:
## roc.formula(formula = radiomics_test$Failure.binary ~ pred_knn_test[,     2], plot = TRUE, legacy.ax
##
## Data: pred_knn_test[, 2] in 26 controls (radiomics_test$Failure.binary No) < 14 cases (radiomics_tes
## Area under the curve: 79.95%
```