Analysis of 'sick' dataset

Mateusz Bakala

2020, 15 of April

Data

Analysed data was taken from OpenML website. Dataset name is "sick" and is about diagnosed (or not) cases with thyroid disease. Training and test indices were given in advance.

Preprocessing

Removing

"Every data needs some cleaning", as the Blues Brothers sang. This one is no exception. We can use some prior knowledge to quickly localize and neutralize potential sources of pain and suffering (of the model).

```
summary(data$TBG_measured)
```

```
##    f
## 3772
```

```
summary(data$TBG)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##      NA      NA      NA     NaN      NA      NA    3772
```

Take column `TBG_measured`, for example. It has only one level, `f`, equal to "false". It means that no patient was measured TBG level and, subsequently, `TBG` column has only missing data. We can safely drop these columns.

```
summary(data$hypopituitary)
```

```
##    f    t
## 3771    1
```

Similarly, `hypopituitary` column has only one `t`, so it's too imbalanced to has an impact on the model. It can be safely dropped as well.

```
data %>% select(age) %>% filter(age > 99)
```

```
##   age
## 1 455
```

One may also notice that `age` has a maximum of `455`. Not even turtles live for that long, so there is probably error in data, most likely duplicated `5`. We can either set it as `NA` or replace with `45`. For medium-sized dataset, let's not waste data and go for the second solution.

```
summary(data$referral_source)
```

```
##  SVHC other   SVI  STMW  SVHD
##   386  2201  1034   112    39
```

There is also multiple-class factor in the column `referral_source`. Our best bet might be one-hot encoding or leaving it as-is. For now, the second option shall suffice. Now we can actually apply presented actions.

```
data <- data %>%
  select(-c(TBG_measured, TBG, hypopituitary)) %>%
  mutate(age = ifelse(age == 455, 45, age))
```

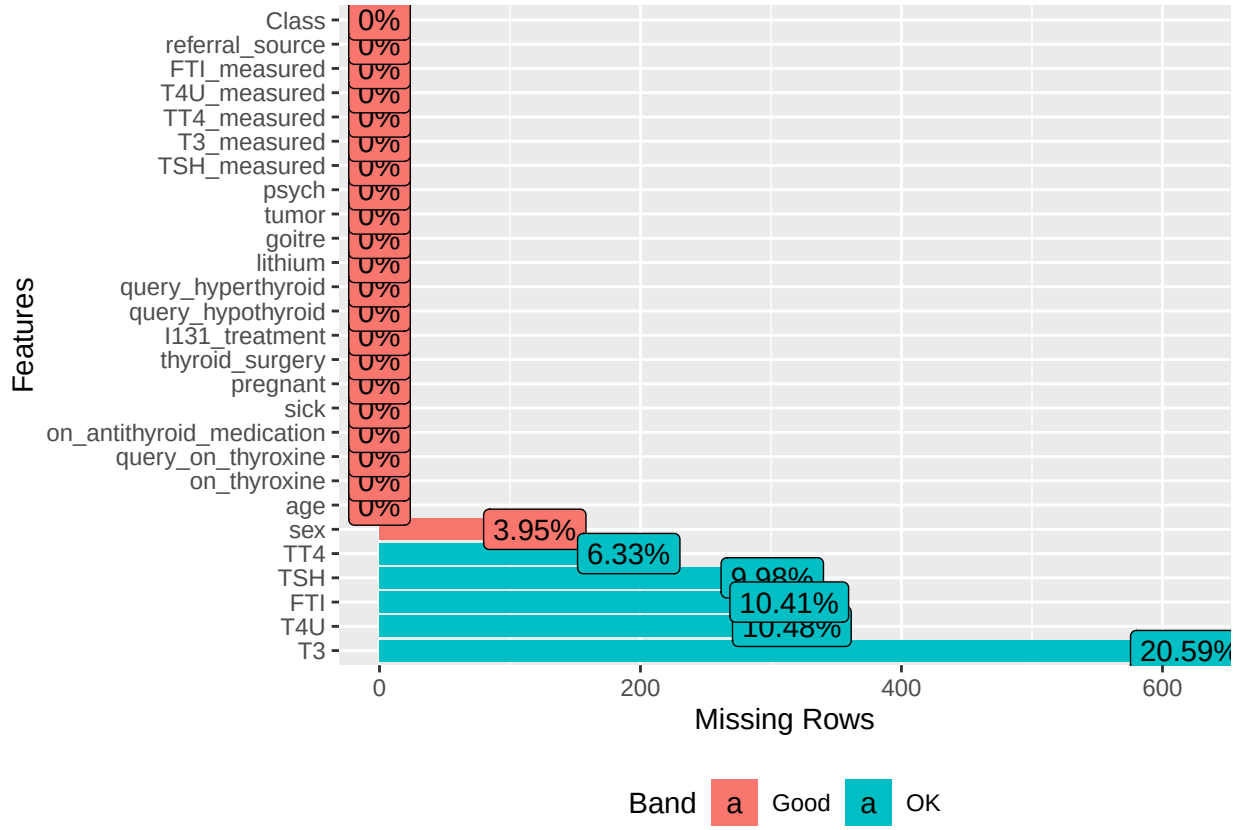
Dataset splitting

We are supposed to split dataset into training and testing subsets, so we can do it now.

```
data_train <- slice(data, indices$x)
data_test <- slice(data, -indices$x)
```

Imputation

Let's have a look at missing data statistics.



There aren't many columns with missing values and even those that have such aren't lacking many, mostly fitting around ten percent mark. That means we can try imputation with kNN and high `k` parameter setting. Obviously, both datasubsets (does such word exist? Hopefully it does) should be treated separately, but luckily `knnImputation` has parameter `distData` which can be utilized to pass distribution data of training dataset while imputing testing one. (I mean, really luckily, I didn't check that in advance.)

```
data_train <- knnImputation(data_train, k = 15)
data_test <- knnImputation(data_test, k = 15, distData = data_train)
```

Tuning and training

For heavily imbalanced two-class classification a safe choice would be forest-esque model. We have no way of getting more data, so we have to work with what we have. Rpart was chosen for its availability in `mlr3` package.

We shall tune two of three worthy parameters of our model, as we should allow node size equal to `1` due to heavy imbalance. We will use random search because GenSA algorythm demands non-discrete values (as an optimalization problem) and grid search is awful. Also, five-fold cross-validation, as demanded. AUPRC as measure registered thanks to stellar Dominik Rafacz's job with his `auprc` package.

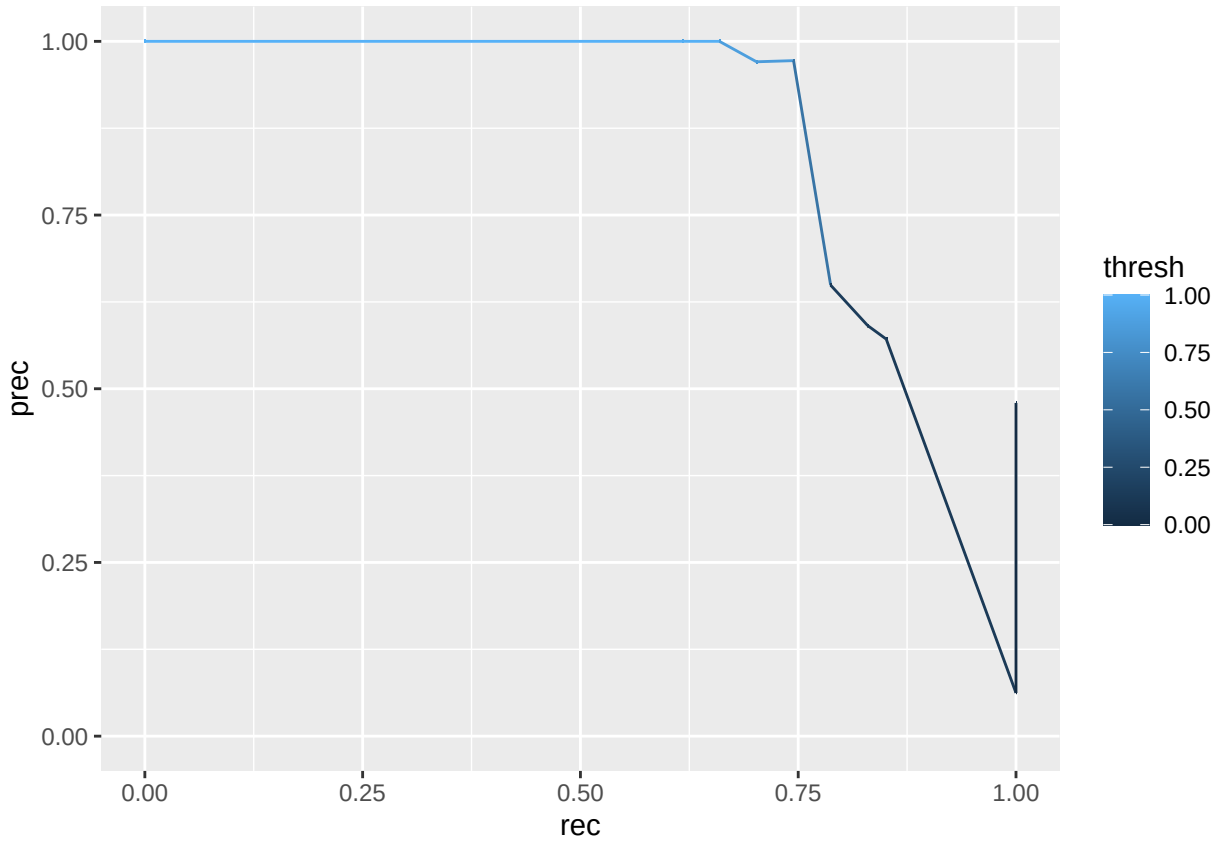
```
learner <- lrn("classif.rpart", predict_type = "prob")
task <- TaskClassif$new(id = "data",
                        backend = rbind(data_train, data_test),
                        target = target,
                        positive = "sick")
resampling <- rsmp("cv", folds = 5)
measure <- msr("classif.auc")
tune_ps <- ParamSet$new(list(
  ParamDbl$new("cp", lower = 0.001, upper = 0.05),
  ParamInt$new("maxdepth", lower = 3, upper = 10)
))
terminator = term("evals", n_evals = 100)

instance <- TuningInstance$new(
  task = task,
  learner = learner,
  resampling = resampling,
  measures = measure,
  param_set = tune_ps,
  terminator = terminator
)
tuner <- tnr("random_search")
tuner$tune(instance)

learner$param_set$values <- instance$result$params
prediction1 <- learner$train(
  task, row_ids = 1:train_length)$predict(
    task, row_ids = train_length + 1:test_length)
```

Prediction

We tried to use non-tuned rpart on non-modified data for comparison, but then realized that with missing data it's impossible... so no comparison then. What's the score though?

```
auprc(prediction1$prob[, "sick"], data_test$Class, "sick")
```

```
## [1] 0.8164319
```



Is it any good? We shall see once we can compare with others. But don't hold your hopes high, it's unlikely.