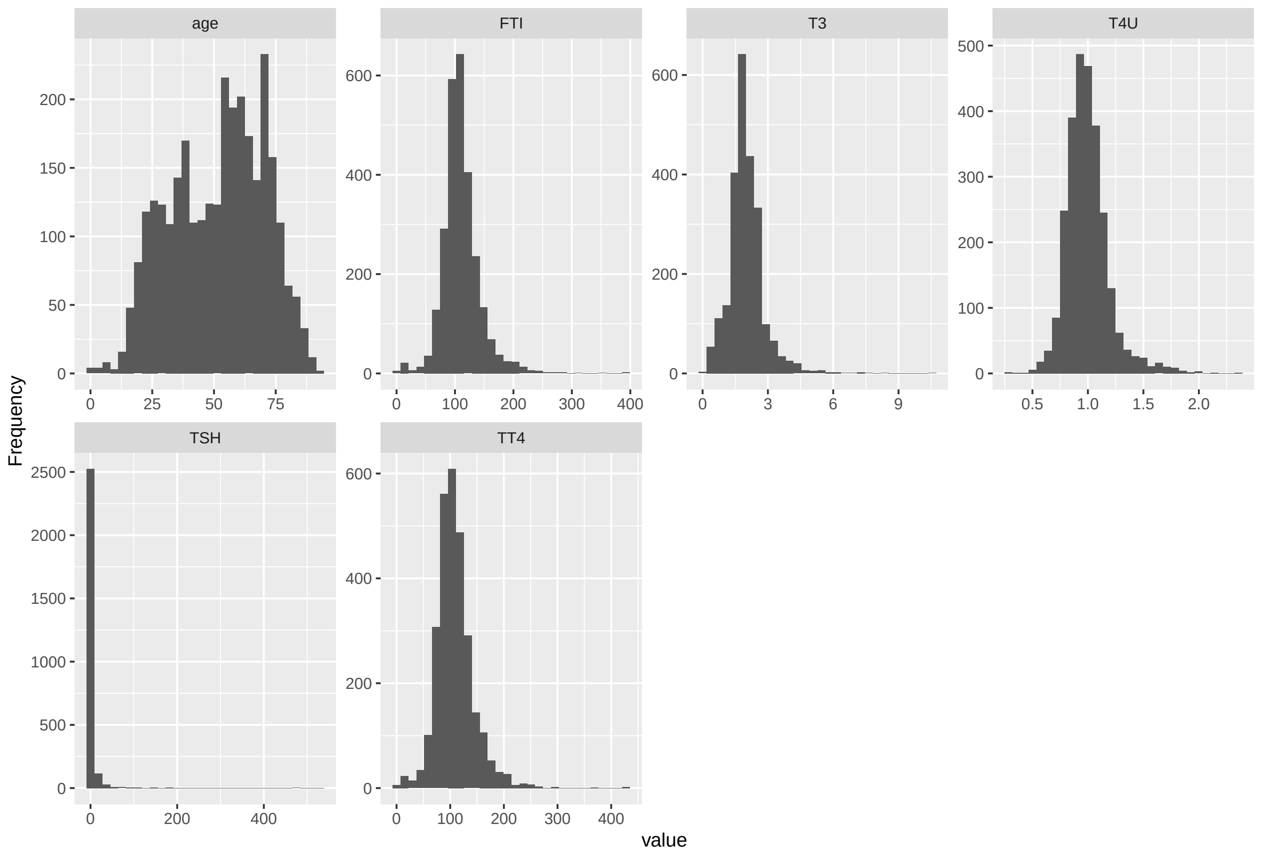
PD1

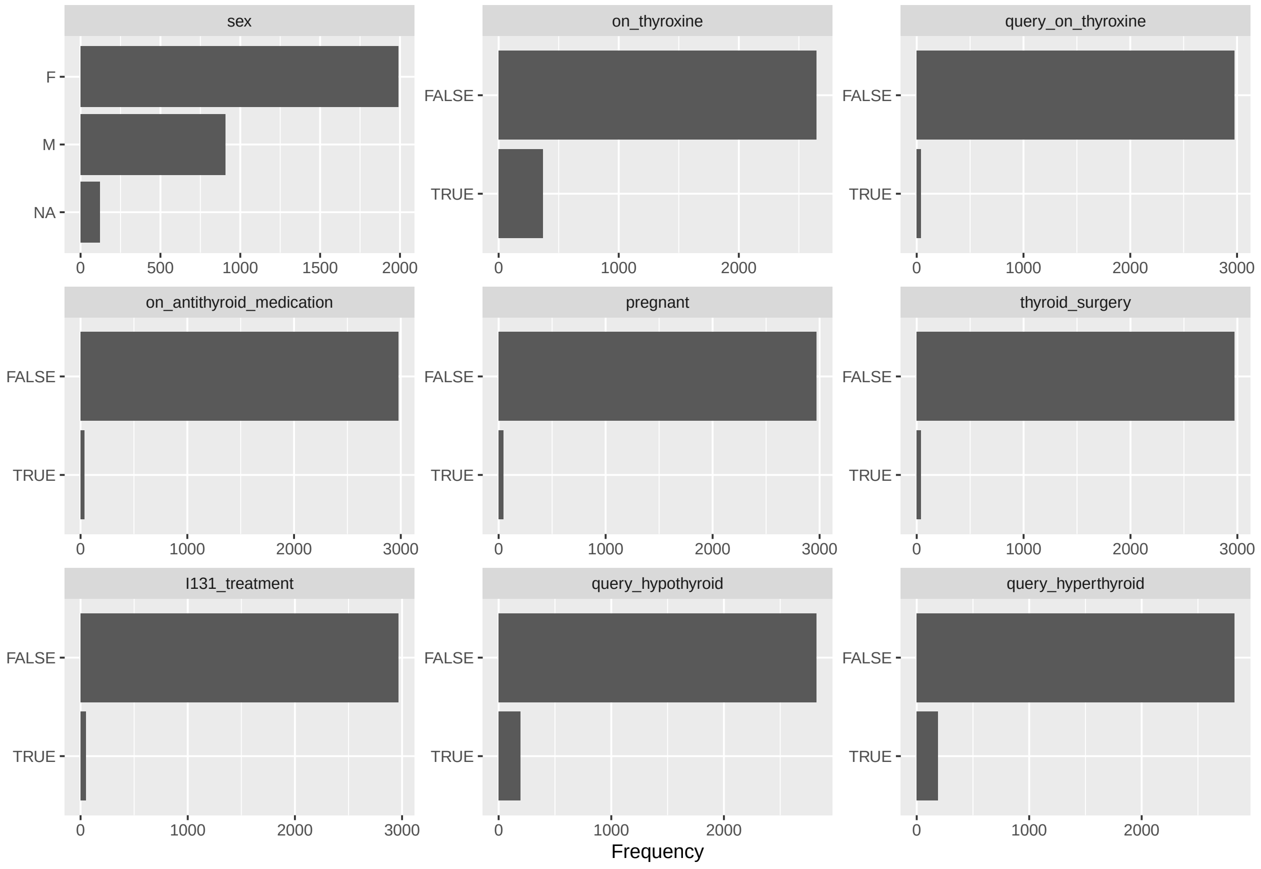
Dominik Rafacz

16.04.2020

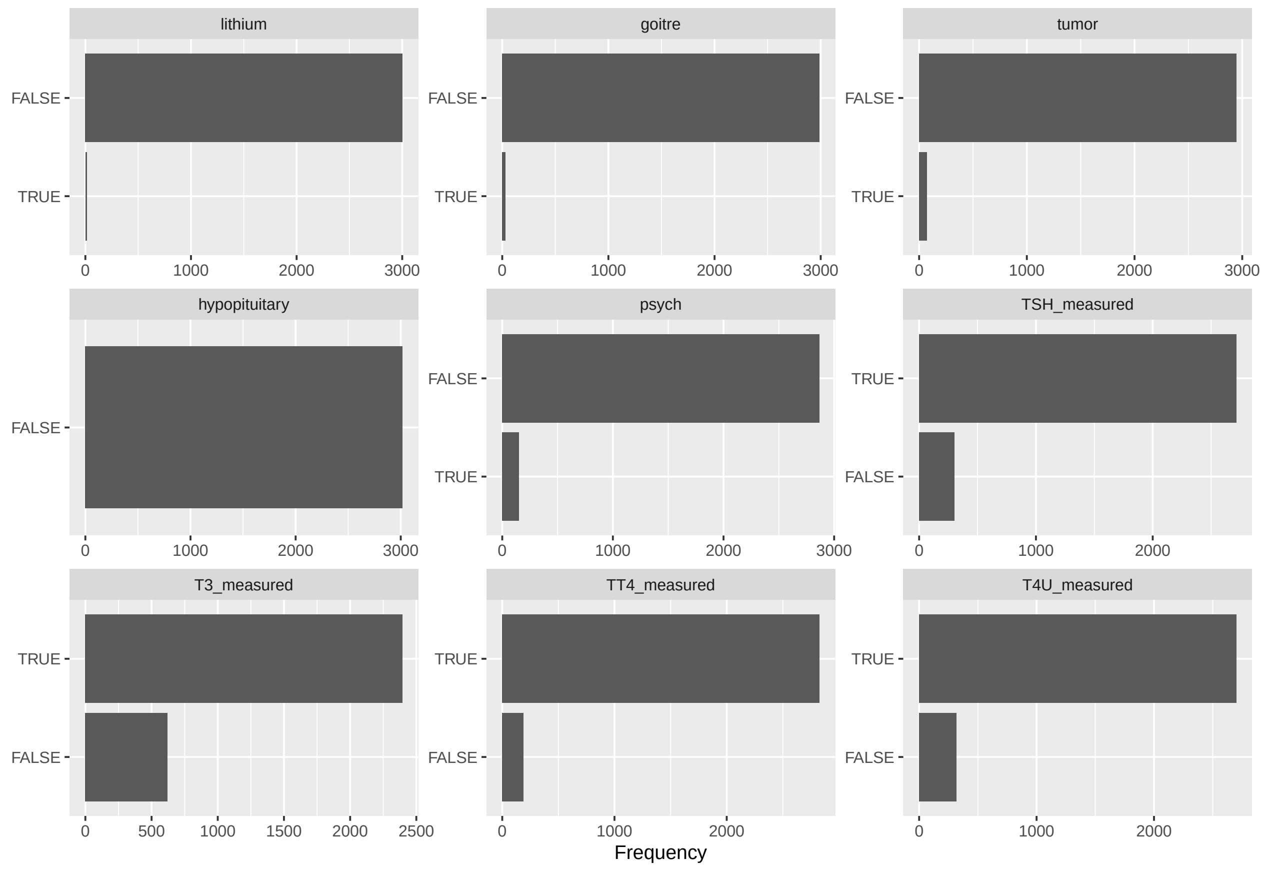
```r
plot_histogram(dat_raw[indices_train, ])
```



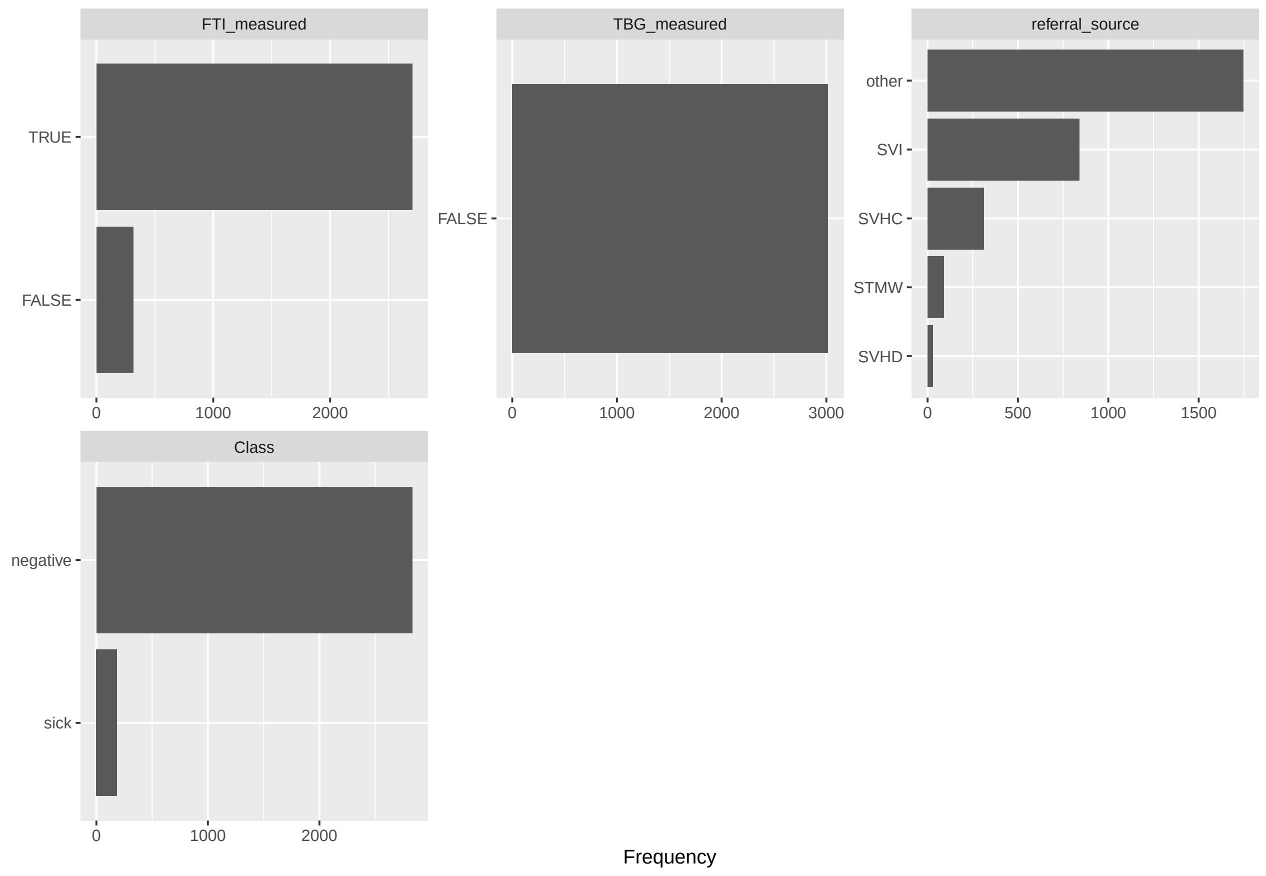
```r
plot_bar(dat_raw[indices_train, ])
```


sex
F
M
NA
0
500
1000
1500
2000
on_thyroxine
FALSE
TRUE
0
1000
2000
query_on_thyroxine
FALSE
TRUE
0
1000
2000
3000
on_antithyroid_medication
FALSE
TRUE
0
1000
2000
3000
pregnant
FALSE
TRUE
0
1000
2000
3000
thyroid_surgery
FALSE
TRUE
0
1000
2000
3000
I131_treatment
FALSE
TRUE
0
1000
2000
3000
query_hypothyroid
FALSE
TRUE
0
1000
2000
query_hyperthyroid
FALSE
TRUE
0
1000
2000
Frequency

Page 1

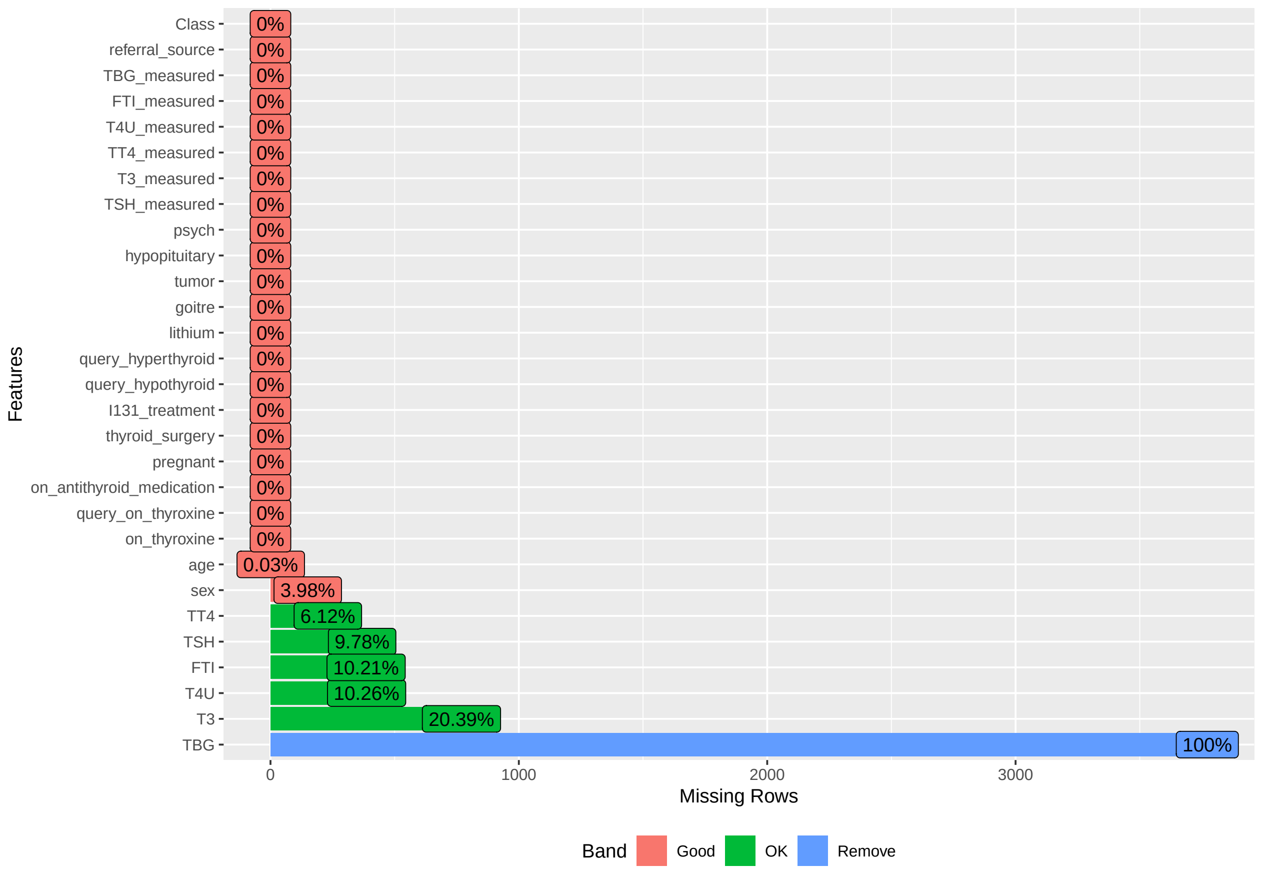

lithium
goitre
tumor
hypopituitary
psych
TSH_measured
T3_measured
TT4_measured
T4U_measured
FALSE
TRUE
0
500
1000
1500
2000
2500
3000
Frequency

Page 2



Page 3

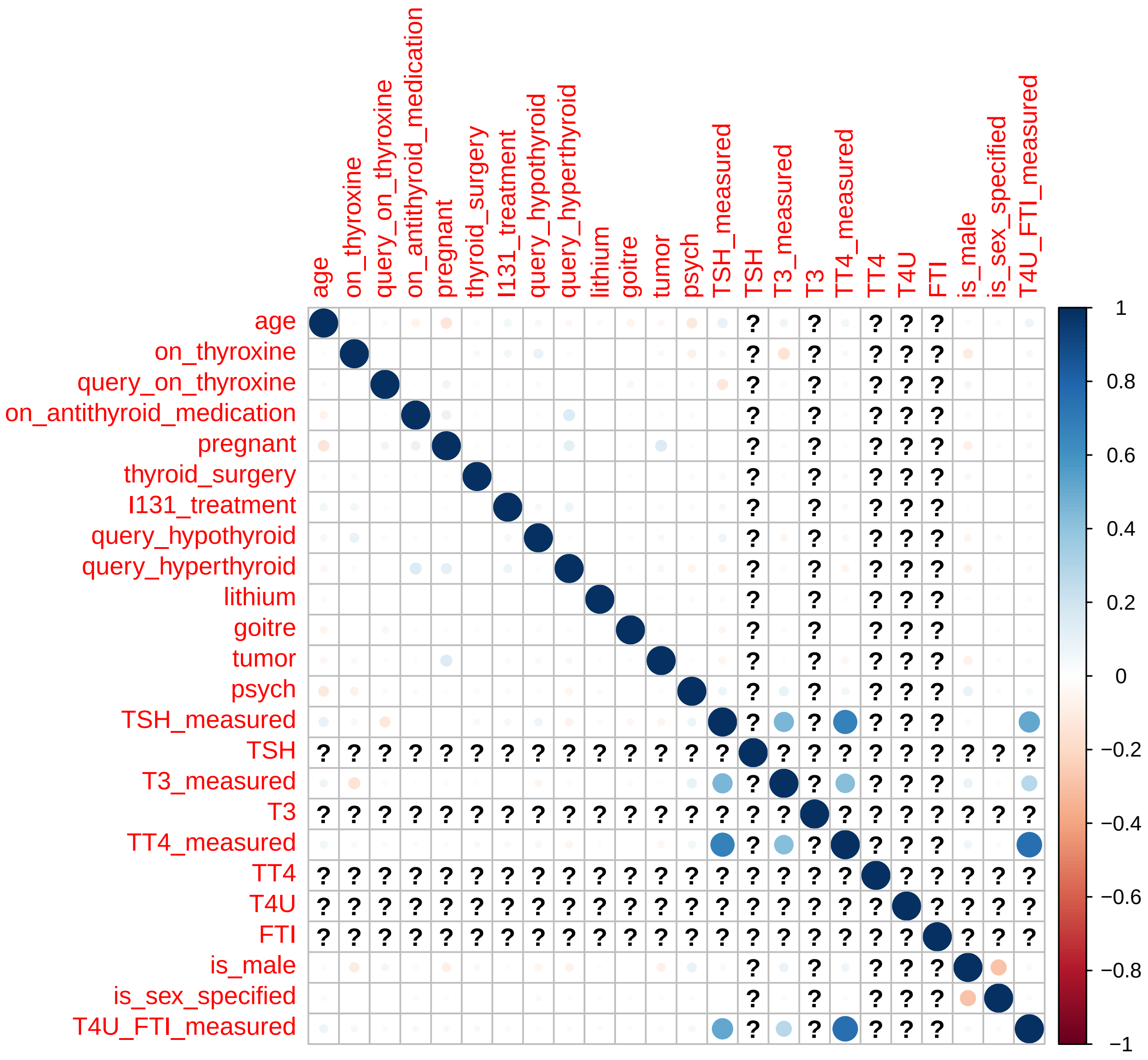
```r
plot_missing(dat_raw)
```



We can see that `TBG` is an empty column, `TBG_measured` has only one value, so does `hypopituitary`.

Preprocessing

```
corrplot(cor(dat[indices_train, -25]))
```



Now, let's prepare very simple rpart model with tuned parameters.

```
task <- TaskClassif$new("sick", dat[indices_train, ], "sick", "sick")
learner <- lrn("classif.rpart")
learner$predict_type = "prob"

set.seed(1998)
param_set <- ParamSet$new(params = list(
  ParamDbl$new("cp", lower = 0.001, upper = 0.1),
  ParamInt$new("minsplit", lower = 5, upper = 30)
))
terminator <- term("evals", n_evals = 10)
tuner <- tnr("random_search")
resampling_inner <- rsmp("holdout")
measures <- list(msr("classif.auc"), msr("classif.auprc"))

autotuner <- AutoTuner$new(learner, resampling_inner, measures = measures,
                           tune_ps = param_set, terminator = terminator, tuner = tuner)
```

```r
resampling_outer <- rsmp("cv", folds = 5)
```

```r
result <- resample(task = task, learner = autotuner, resampling = resampling_outer)
result$aggregate(measures)
```

```r
knitr::kable(cbind(classif.auc = 0.9540692, classif.auprc = 0.7859588)) %>%
  kable_styling(position = "center")
```

classif.auc	classif.auprc
0.9540692	0.7859588

Parameters chosen during tuning:

- cp = 0.005
- minsplit = 10

Operating on data

I have decided to combine information if some measurements were made into one column by summing them up.

```r
dat_t <- dat %>%
  mutate(measurements = T4U_FTI_measured + TT4_measured + T3_measured + TSH_measured) %>%
  select(-ends_with("measured"))

task <- TaskClassif$new("sick", dat_t[indices_train, ], "sick", "sick")

set.seed(1998)
learner <- lrn("classif.rpart", cp = 0.005, minsplit = 10)
learner$predict_type = "prob"
```

```r
result_t <- resample(task = task, learner = learner, resampling = resampling_outer)
result_t$aggregate(measures)
```

```r
knitr::kable(cbind(classif.auc = 0.9499272, classif.auprc = 0.8274248)) %>%
  kable_styling(position = "center")
```

classif.auc	classif.auprc
0.9499272	0.8274248

The result went up a little, so I decided to keep this change

Imputation

I've tried using imputation in order to use data properties to its limits. Firstly I tried using imputation with histogram.

```r
task <- TaskClassif$new("sick", dat_t[indices_train, ], "sick", "sick")

set.seed(1998)
learner <- lrn("classif.rpart", cp = 0.005, minsplit = 10)
learner$predict_type = "prob"
```

```r
po_imp_hist <- po("imputehist")
task <- po_imp_hist$train(list(task))[[1]]
```

```r
result_t <- resample(task = task, learner = learner, resampling = resampling_outer)
result_t$aggregate(measures)
```

```r
knitr::kable(cbind(classif.auc = 0.9621942, classif.auprc = 0.8384416)) %>%
  kable_styling(position = "center")
```

classif.auc	classif.auprc
0.9621942	0.8384416

Then I've tried using `MICE` algorithm.

```r
task <- TaskClassif$new("sick",
                        cbind(
                          complete(
                            mice(
                              dat_t[indices_train, -21])),
                          dat_t[indices_train, "sick"]),
                        "sick", "sick")

set.seed(1998)
learner <- lrn("classif.rpart", cp = 0.005, minsplit = 10)
learner$predict_type = "prob"
```

```r
result_t <- resample(task = task, learner = learner, resampling = resampling_outer)
result_t$aggregate(measures)
```

```r
knitr::kable(cbind(classif.auc = 0.9282098, classif.auprc = 0.8036027)) %>%
  kable_styling(position = "center")
```

classif.auc	classif.auprc
0.9282098	0.8036027

It turned out that in this case using imputation with histogram gives better performance and lifts AUC a little while not decreasing AUPRC too much, so I decided to keep this change.

Oversampling

At last, I've tried using generating artificial data in minority class using SMOTE algorithm.

```r
dat_m <- dat_t %>%
  mutate_all(as.numeric) %>%
  mutate(sick = dat_t$sick)

task <- TaskClassif$new("sick", dat_m[indices_train, ], "sick", "sick")

set.seed(1998)
learner <- lrn("classif.rpart", cp = 0.005, minsplit = 10)
learner$predict_type = "prob"

po_imp_hist <- po("imputehist")
```

```r
task <- po_imp_hist$train(list(task))[[1]]

po_smote <- po("smote", dup_size = 2)   # create twice as much positive class observations
task <- po_smote$train(list(task))[[1]]
```

```r
result_t <- resample(task = task, learner = learner, resampling = resampling_outer)
result_t$aggregate(measures)
```

```r
knitr::kable(cbind(classif.auc = 0.9870872, classif.auprc = 0.9244521)) %>%
  kable_styling(position = "center")
```

classif.auc	classif.auprc
0.9870872	0.9244521

And this solution resulted in significant increase in both measures. However, we need to remember that oversampling may result in similar observations in both training and validation set and in consequence overfitting. To find out if it is easy to generalise, we have to check result on the test set.

Final results

```r
task <- TaskClassif$new("sick", dat_m[indices_train, ], "sick", "sick")

set.seed(1998)
learner <- lrn("classif.rpart", cp = 0.001, minsplit = 10)
learner$predict_type = "prob"

po_imp_hist <- po("imputehist")
task <- po_imp_hist$train(list(task))[[1]]

po_smote <- po("smote", dup_size = 2)
task <- po_smote$train(list(task))[[1]]
enr_test_size <- nrow(task$data())

# append test set
task$rbind(dat_m[indices_test, ])

task <- po("imputehist")$train(list(task))[[1]]

learner$train(task, row_ids = 1:enr_test_size)
prediction <- learner$predict(task, row_ids = (enr_test_size + 1):nrow(task$data()))
```

```r
prediction$score(measures = list(msr("classif.auc"), msr("classif.auprc")))
```

And this is my final result:

```r
knitr::kable(cbind(classif.auc = 0.8727906, classif.auprc = 0.7687503)) %>%
  kable_styling(position = "center")
```

classif.auc	classif.auprc
0.8727906	0.7687503