Sick dataset analysis

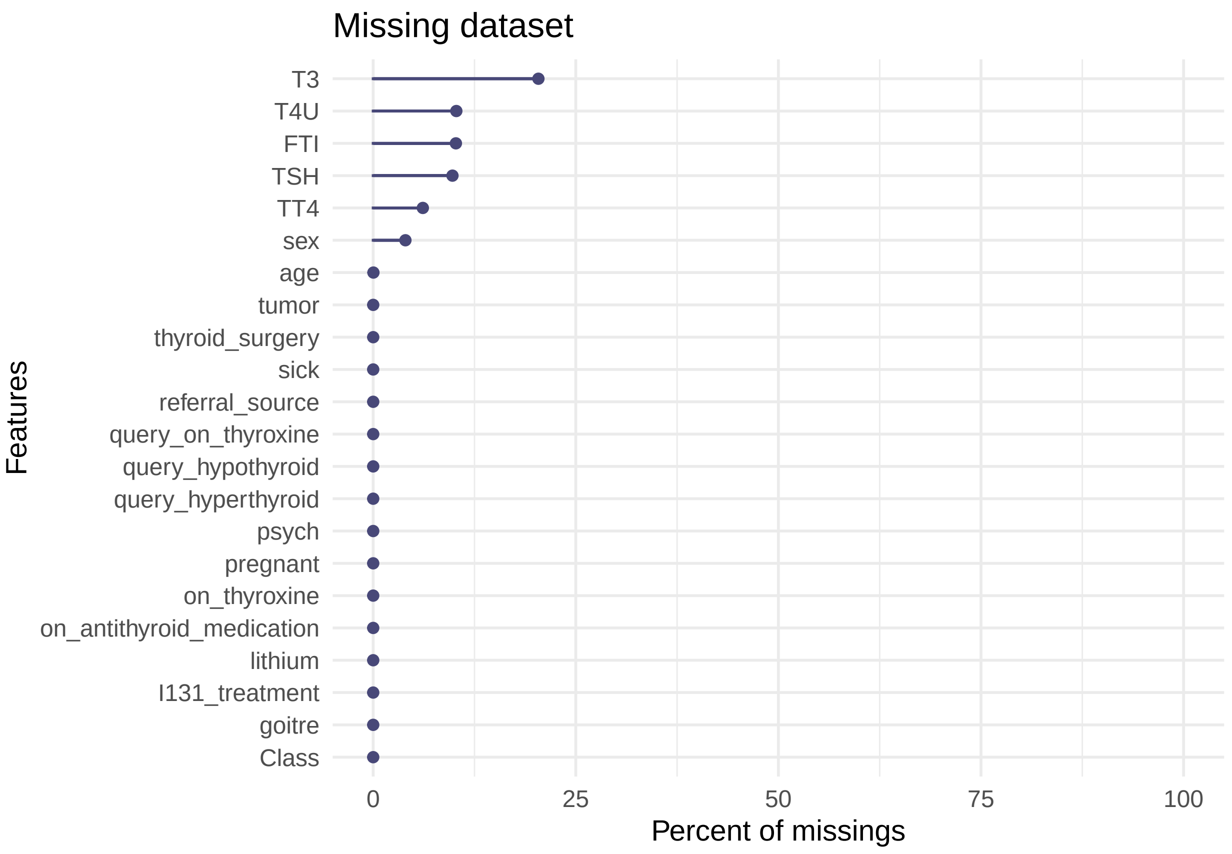
Malgorzata Wachulec

17 04 2020

Preprocessing

Some factor columns do not have sufficient number of observations in each of the levels. That means some sets will have observations with only one feature value. These variables will not be useful when training the model and might cause errors. For this reason I am deleting columns 'TBG', 'hypopituitary' and 'TBG_measured'.

After these columns are deleted, we can now look at the missing values. All the variables that end with '_measured' are flags indicating wheter a certain characteristic is missing or not. Since in every column there is less than 25% of missing data, I will imput it as opposed to deleting the entire column or observation for which the column is empty. This means I will no longer need these flags and I will delete them.



As I am planning use the mice package to imput missing data, which is predicting the missing values through model built on other observations, I will first divide data into training, validation and test sets. This will prevent for training set to have values inputed based on observations from test set and vice versa.

I have decided to use validation set as opposed to cross-validation, that was advised, because I am planning to use under- and oversampling, which does not work great with cross-validation. The results obtained in such manner would not reflect the measure levels expected on the test set.

```r
# dividing data into train and test
training_indicies <- read.table("indeksy_treningowe.txt")
training_indicies <- training_indicies$x
trainset_to_be_divided <- dataset[ training_indicies,]
testset <-  dataset[-training_indicies,]

# dividing training set into 75% training set and 25% valdation set
# each set has the same distribution of the target class
set.seed(3456)
trainIndex <- createDataPartition(trainset_to_be_divided$Class, p = .75,
                                  list = FALSE,
                                  times = 1)
trainset <- trainset_to_be_divided[ trainIndex,]
validset <- trainset_to_be_divided[-trainIndex,]
```

Imputation of missing values using mice() function from mice package. As to make sure, that the model for imputing missing values will not use target variable, it is taken away before imputing and added again to the training, validation and test set after the imputing process.

Building basic logistic regression

Here I am defining training, validation and test tasks, as well as the learner - logistic regression. Then I am checking it's performance on the validation set.

```r
# task and learner definition
trainTask <- TaskClassif$new(id = "train sick", backend = complete_trainset,
                             target = "Class", positive = "sick")
validTask <- TaskClassif$new(id = "valid sick", backend = complete_validset,
                             target = "Class", positive = "sick")
testTask <- TaskClassif$new(id = "test sick", backend = complete_testset,
                            target = "Class",positive = "sick")

learner <- lrn("classif.log_reg")
learner$predict_type <- "prob"

# model and prediction
learner$train(trainTask)
result <- learner$predict(validTask)
cat("Contigency table: \n")
```

```
## Contigency table:
```

```r
result$confusion
```

```
##           truth
## response   sick negative
##   sick       28        9
##   negative   18      699
```

27 cases from the validation set were classified incorrectly, but only 28 out of 56 sick patients were classified as such. Let's check recall and other measures of this prediction.

```
## Auc on validation set:  0.9340457
## Auprc on validation set:  0.5758775
```

```
## Recall on validation set:  0.6086957
```

```
## Specificity on validation set:  0.9872881
```

Recall is low, as expected, whereas specificity is high, also as expected.

Improving the model

Oversampling

Generating more observations with the the minorty target class using random over-sampling technique from ROSE package.

```
## [1] "Target variable class proportion"
```

```
##
## negative     sick
##     2124      138
```

```
## [1] "Target variable class proportion after oversampling"
```

```
##
## negative     sick
##     1140     1122
```

Defining new training task, including the new observations:

```
# new training task definition
trainTask <- TaskClassif$new(id = "train sick", backend = trainset.rose,
                             target = "Class", positive = "sick")

# model and prediction
learner$train(trainTask)
result <- learner$predict(validTask)
cat("Contigency table: \n")
```

```
## Contigency table:
```

```
result$confusion
```

```
##           truth
## response   sick negative
##   sick       38       83
##   negative    8      625
```

This time 91 cases were misclassified, but there were only 8 false negatives. Let's check the measures of the result.

```
## Auc on validation set:  0.9297777
```

```
## Auprc on validation set:  0.5076613
```

```
## Recall on validation set:  0.826087
```

```
## Specificity on validation set:  0.8827684
```

Auprc has dropped, same as specificity, and recall has risen as expected. Seems there has been too much oversampling.

Under- and oversampling

This trial combines oversampling with undersampling. Below we can see the contigency table of the result and measures of the prediction.

```
## [1] "Target variable class proportion after under- and oversampling:"
```

```
##
## negative     sick
##     2094      168
```

```
##
##
## Contigency table:
```

```
##           truth
## response   sick negative
##   sick       25        9
##   negative   21      699
```

```
##
##
## Auc on validation set:  0.9331245
```

```
## Auprc on validation set:  0.6145482
```

```
## Recall on validation set:  0.5434783
```

```
## Specificity on validation set:  0.9872881
```

Now the auprc measure grew higher, same as auc and recall.

Improving preprocessing

Here I go back to preprocessing and see whether I can improve the model this way. Let's apply logarithmic function to variables 'T4U' and 'TSH' and scale all of the numerical variables.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
##
## Contigency table:
```

```
##           truth
## response   sick negative
##   sick       26       10
##   negative   20      698
```

```
##
##
## Auc on validation set:  0.9451609
```

```
## Auprc on validation set:  0.5720376
```

```
## Recall on validation set:  0.5652174
```

```
## Specificity on validation set:  0.9858757
```

After additional preprocessing, the model performed worse on the validation set based on auprc. For this reason the final version is the one before based on the under- and oversampling method.

Final score on testset

These are the results obtained on the test set using the final version of the model.

```
## Contigency table:
```

```
##           truth
## response   sick negative
##   sick       22       14
##   negative   25      695
```

```
##
##
## Auc on test set:  0.9265372
```

```
## Auprc on test set:  0.4765698
```

```
## Recall on test set:  0.4680851
```

```
## Specificity on test set:  0.9802539
```