An EDA and a white box classification of an OpenML sick dataset

Karol Pysiak

17.04.2020

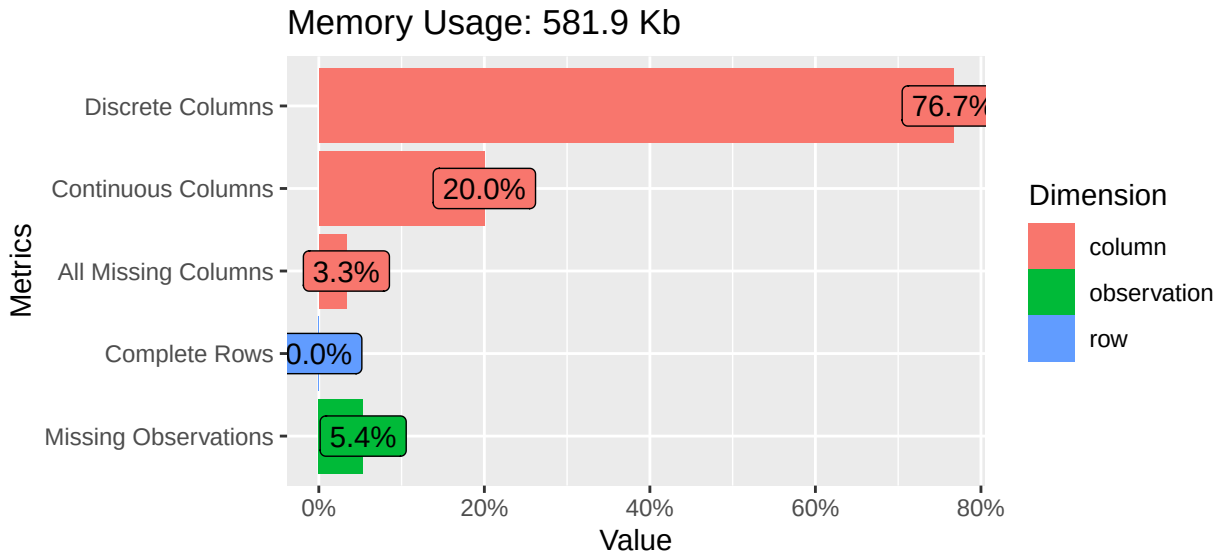
Introduction

In this exercise we will be working on a "sick" dataset from the OpenML database. What will be presented there is an Exploratory Data Analaysis and an attempt to predict some classes on a test data with an explainable model, which will be introduced later.

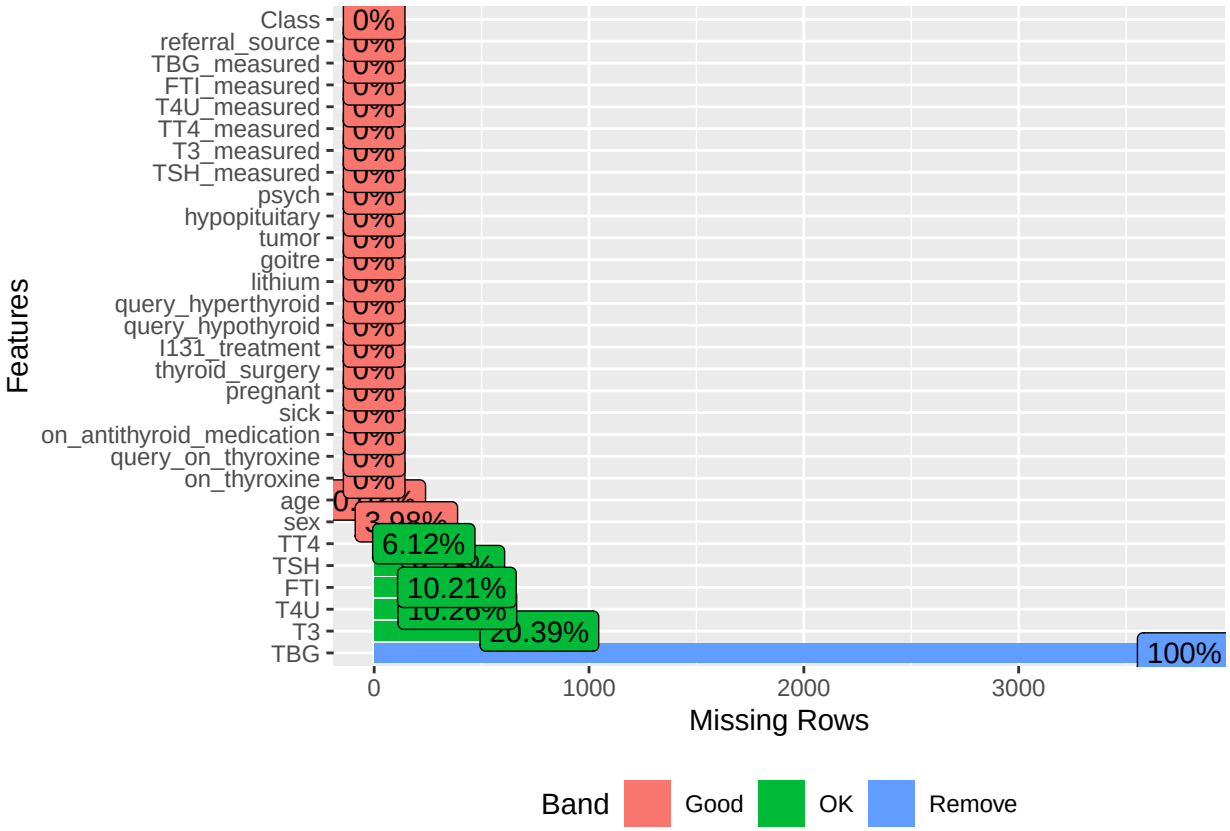
Exploratory Data Analysis

Insight into structure of data

Firstly, we want to check how the dataset is built. We will show if there are missing values, what types are the variables of and how variables are distributed.

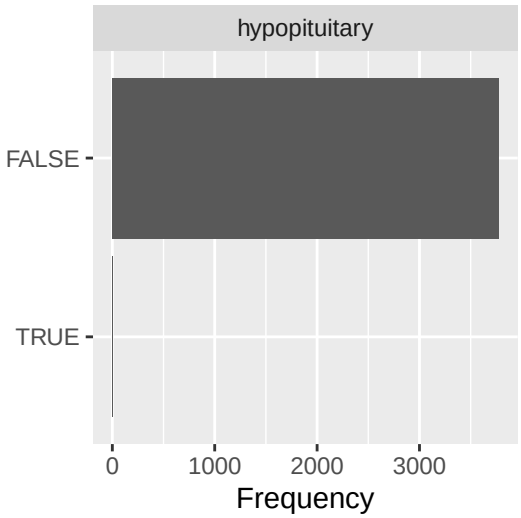


The one thing that can bother us when looking on this graph is that we have some fraction of variables completely missing. The percentage of missing values in total is not that bad. Let's check how this missing values look grouping by variables.



As we have seen above there is one variable completely missing and it is the `TBG`. We will delete it from dataset, because it is totally insignificant in this situation. The rest of variables have a quite small fraction of missing values, so for the sake of simplicity we will delete rows that contain missing values.

One more thing that can very much affect our classification is that the `hypopituitary` variable, which is a logical variable, has close to 0 occurances of a `TRUE` value, so it will bring no benefit to the model.



Some of variables are just flags if the other variable contains an `NA` value in that observation. This can give us very little information so we will omit these variables either.

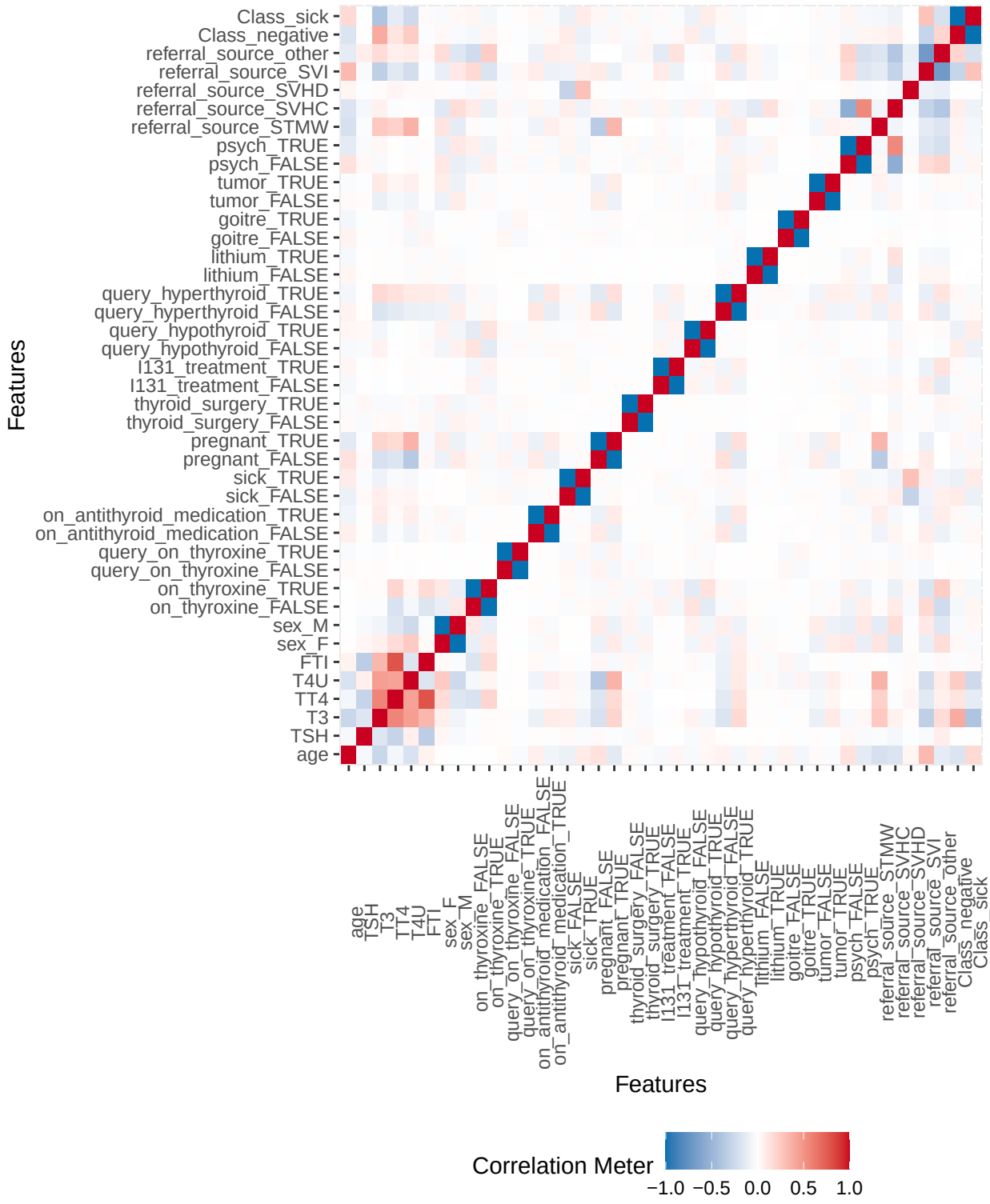
So after all these observations we have a dataset that look like this.

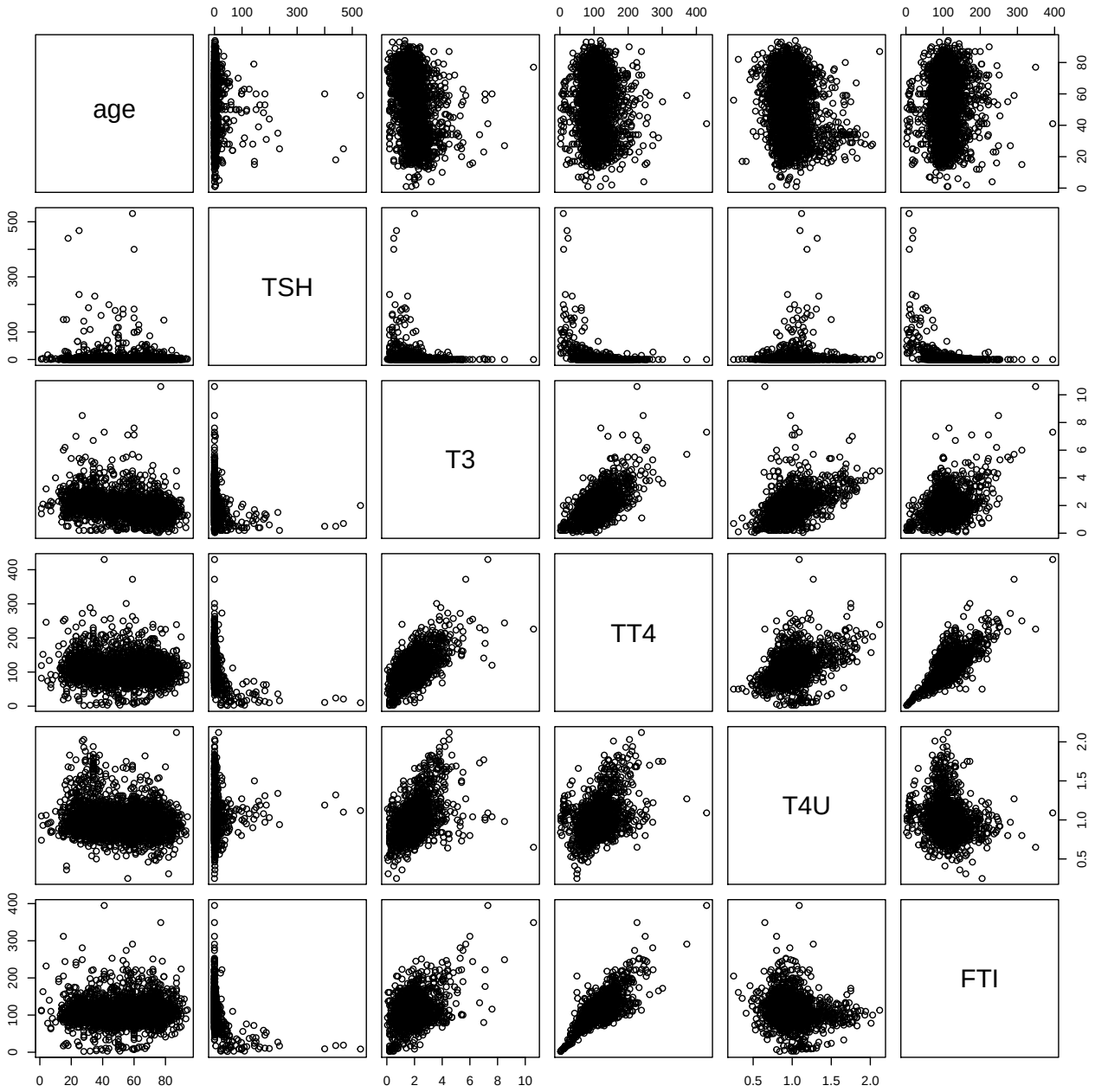
```
## # A tibble: 2,643 x 22
##      age sex   on_thyroxine query_on_thyrox~ on_antithyroid_~ sick  pregnant
##    <dbl> <chr> <lgl>        <lgl>            <lgl>            <lgl> <lgl>
##  1    41 F     FALSE        FALSE            FALSE            FALSE FALSE
##  2    70 F     FALSE        FALSE            FALSE            FALSE FALSE
##  3    80 F     FALSE        FALSE            FALSE            FALSE FALSE
##  4    66 F     FALSE        FALSE            FALSE            FALSE FALSE
##  5    68 M     FALSE        FALSE            FALSE            FALSE FALSE
##  6    84 F     FALSE        FALSE            FALSE            FALSE FALSE
##  7    71 F     FALSE        FALSE            FALSE            TRUE  FALSE
##  8    59 F     FALSE        FALSE            FALSE            FALSE FALSE
##  9    28 M     FALSE        FALSE            FALSE            FALSE FALSE
## 10    63 F     FALSE        FALSE            FALSE            FALSE FALSE
## # ... with 2,633 more rows, and 15 more variables: thyroid_surgery <lgl>,
## #   I131_treatment <lgl>, query_hypothyroid <lgl>, query_hyperthyroid <lgl>,
## #   lithium <lgl>, goitre <lgl>, tumor <lgl>, psych <lgl>, TSH <dbl>, T3 <dbl>,
## #   TT4 <dbl>, T4U <dbl>, FTI <dbl>, referral_source <chr>, Class <chr>
```

While processing the data it turned out that one of observations has invalid `age` of 455. It was 45 or 55 probably, but there is no need to choose it at random, because it is just a one observation, so we will just remove it from our dataset. Below we can see a part of the table of frequencies of ceratain ages of observations.

age	Freq
85	14
86	4
87	11
88	6
89	7
90	5
91	1
92	1
93	2
94	1
455	1

Now we will check the dependencies between variables.


Features
Class_sick
Class_negative
referral_source_other
referral_source_SVI
referral_source_SVHD
referral_source_SVHC
referral_source_STMW
psych_TRUE
psych_FALSE
tumor_TRUE
tumor_FALSE
goitre_TRUE
goitre_FALSE
lithium_TRUE
lithium_FALSE
query_hyperthyroid_TRUE
query_hyperthyroid_FALSE
query_hypothyroid_TRUE
query_hypothyroid_FALSE
I131_treatment_TRUE
I131_treatment_FALSE
thyroid_surgery_TRUE
thyroid_surgery_FALSE
pregnant_TRUE
pregnant_FALSE
sick_TRUE
sick_FALSE
on_antithyroid_medication_TRUE
on_antithyroid_medication_FALSE
query_on_thyroxine_TRUE
query_on_thyroxine_FALSE
on_thyroxine_TRUE
on_thyroxine_FALSE
sex_M
sex_F
FTI
T4U
TT4
T3
TSH
age
age
TSH
T3
TT4
T4U
FTI
sex_F
sex_M
on_thyroxine_FALSE
on_thyroxine_TRUE
query_on_thyroxine_FALSE
query_on_thyroxine_TRUE
on_antithyroid_medication_FALSE
on_antithyroid_medication_TRUE
sick_FALSE
sick_TRUE
pregnant_FALSE
pregnant_TRUE
thyroid_surgery_FALSE
thyroid_surgery_TRUE
I131_treatment_FALSE
I131_treatment_TRUE
query_hypothyroid_FALSE
query_hypothyroid_TRUE
query_hyperthyroid_FALSE
query_hyperthyroid_TRUE
lithium_FALSE
lithium_TRUE
goitre_FALSE
goitre_TRUE
tumor_FALSE
tumor_TRUE
psych_FALSE
psych_TRUE
referral_source_STMW
referral_source_SVHC
referral_source_SVHD
referral_source_SVI
referral_source_other
Class_negative
Class_sick
Features
Correlation Meter
−1.0 −0.5 0.0 0.5 1.0



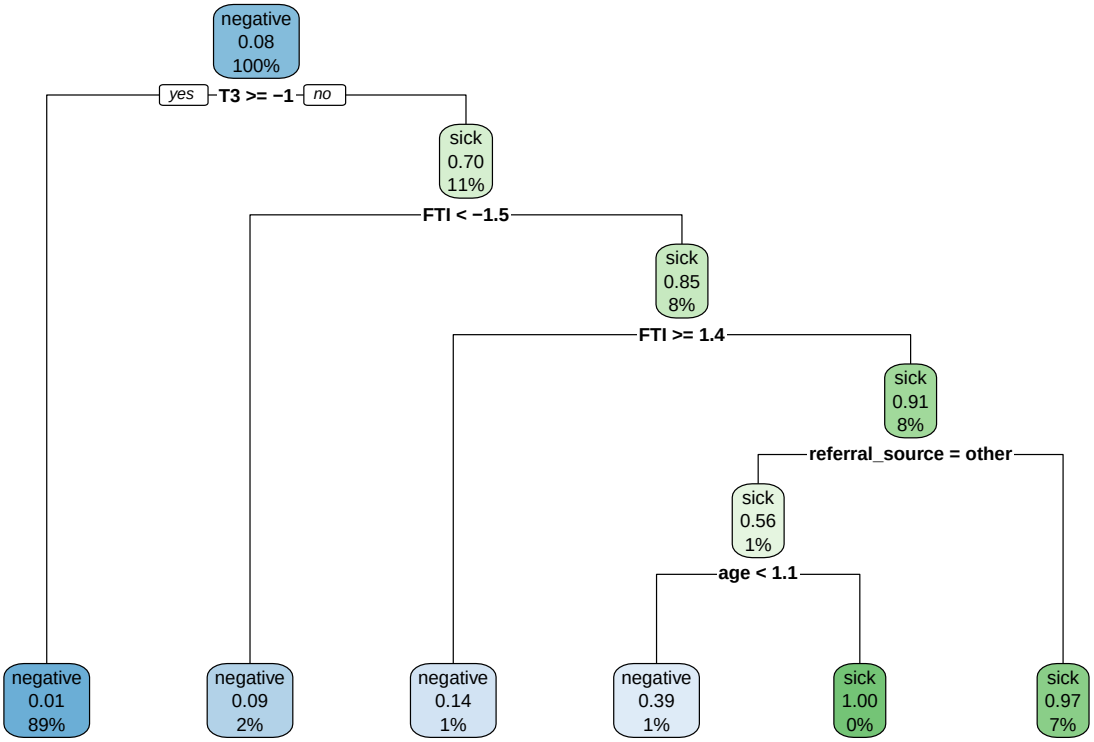
Variables `T3`, `TT4`, `T4U`, `FTI` looks somehow correlated, especially `TT4` and `FTI`, but not in a way that we should worry about.

White box models testing

Firstly we just standarize our data and split it into the training and testing data. In the training data, as it was mentioned before, we just remove rows that contain missing data. With the testing data we cannot do that because it could affect our end score, which we want to be as close to the reality as possible. We will replace the missing values in the testing data with a mean for numerical variables and a mode for categorical variables.

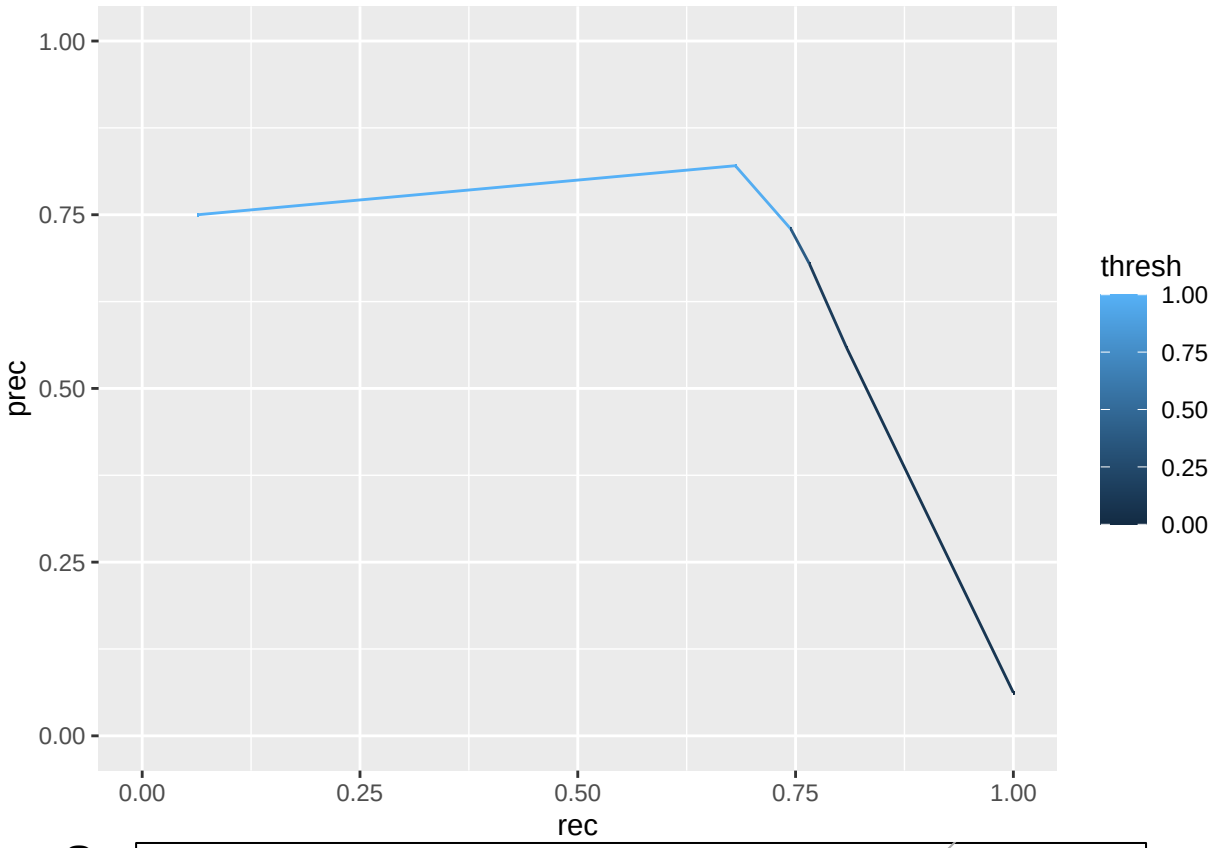
Decision Tree

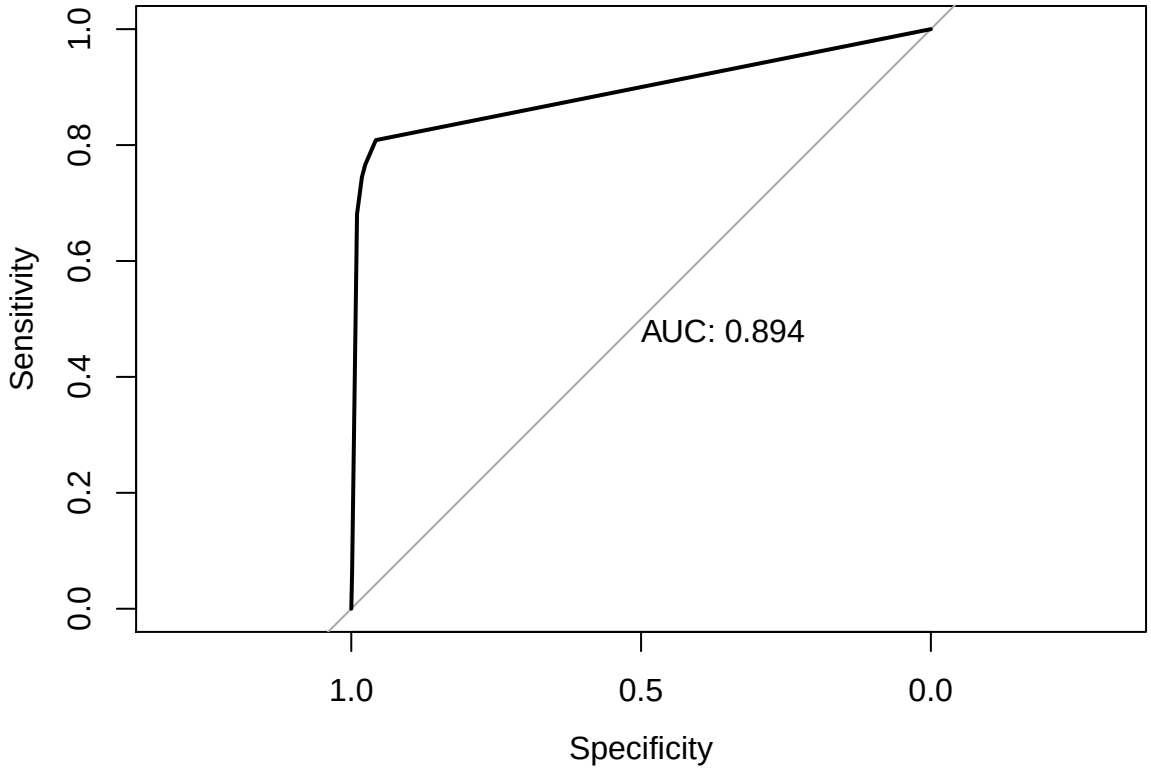
The first interpretable model that we will test here is Decision Tree. It is probably one of the easiest to describe and draw model. There we will use the `rpart` algorithm from the `mlr` package. After training it on out training data we have a tree that looks like this.



As we can see, model decides which way to go based on value of certain variables. Thresholds are tuned during the training process.

```
## [1] "AUPRC: 0.650825275451872"
```

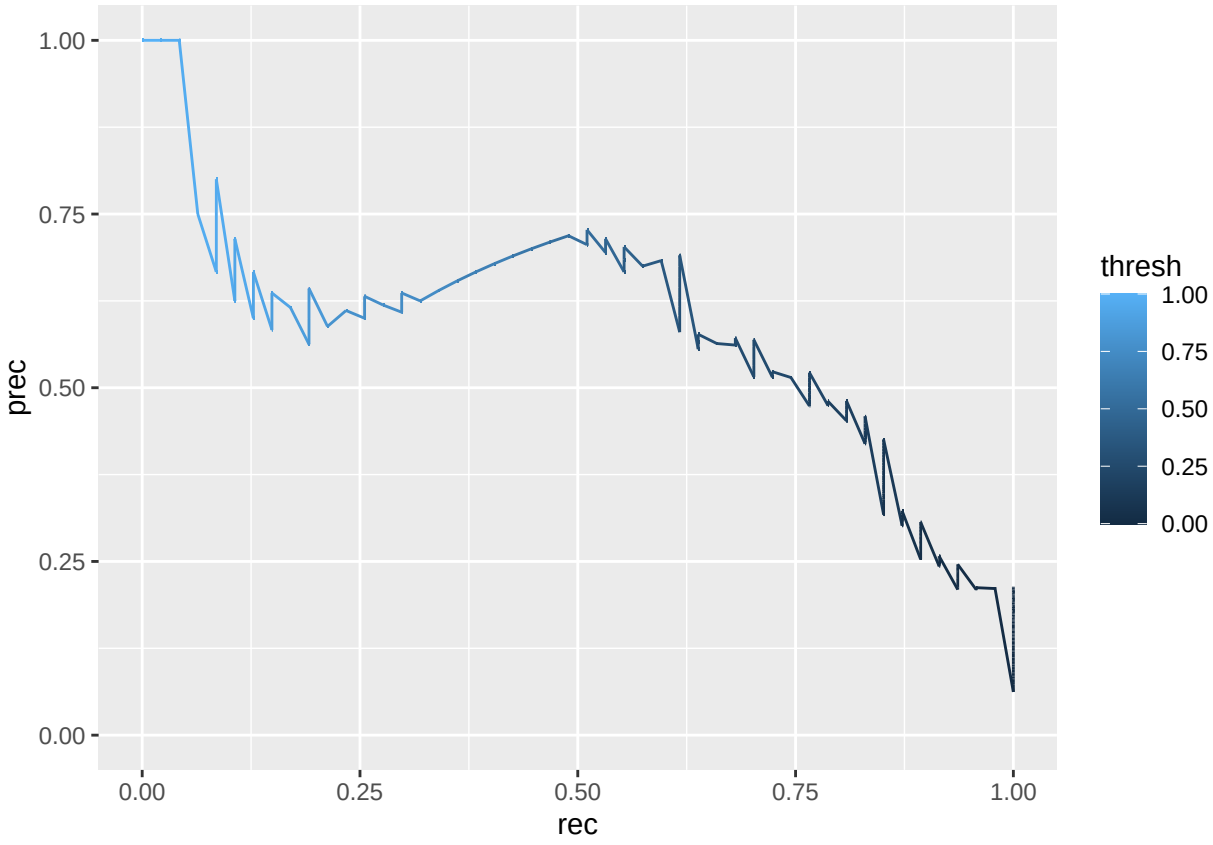

1.00
0.75
0.50
0.25
0.00
prec
0.00
0.25
0.50
0.75
1.00
rec
thresh
1.00
0.75
0.50
0.25
0.00

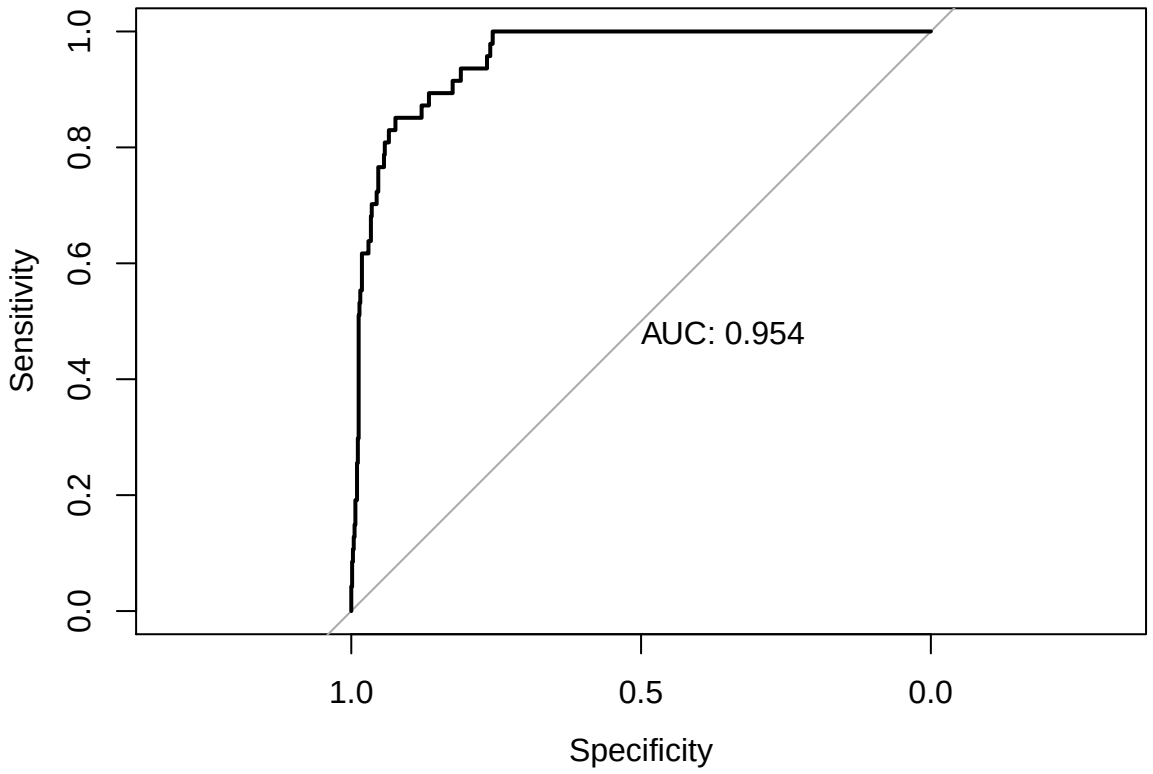

Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
AUC: 0.894
1.0
0.5
0.0
Specificity

Logistic Regression

Logistic Regression is easy to explain. It is like Linear Regression of classification.

```
## [1] "AUPRC: 0.561059652256059"
```



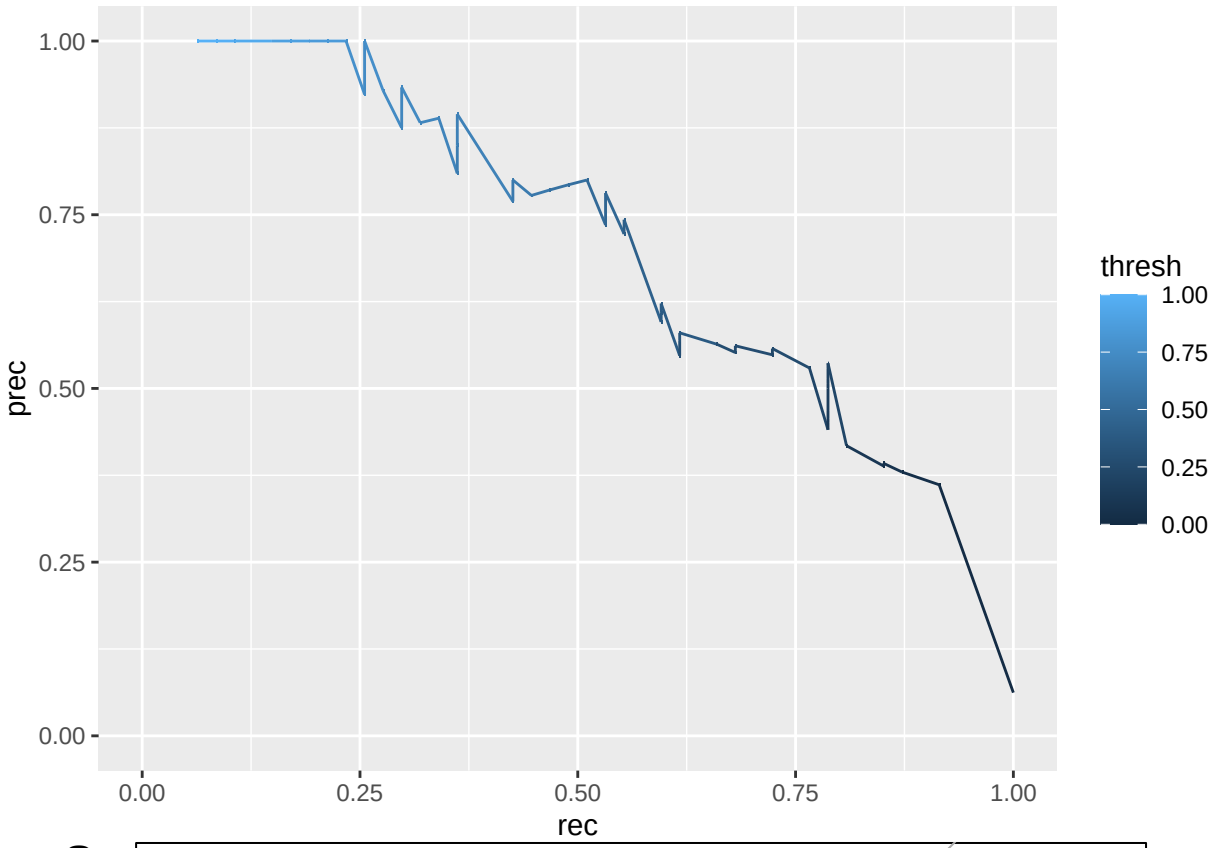


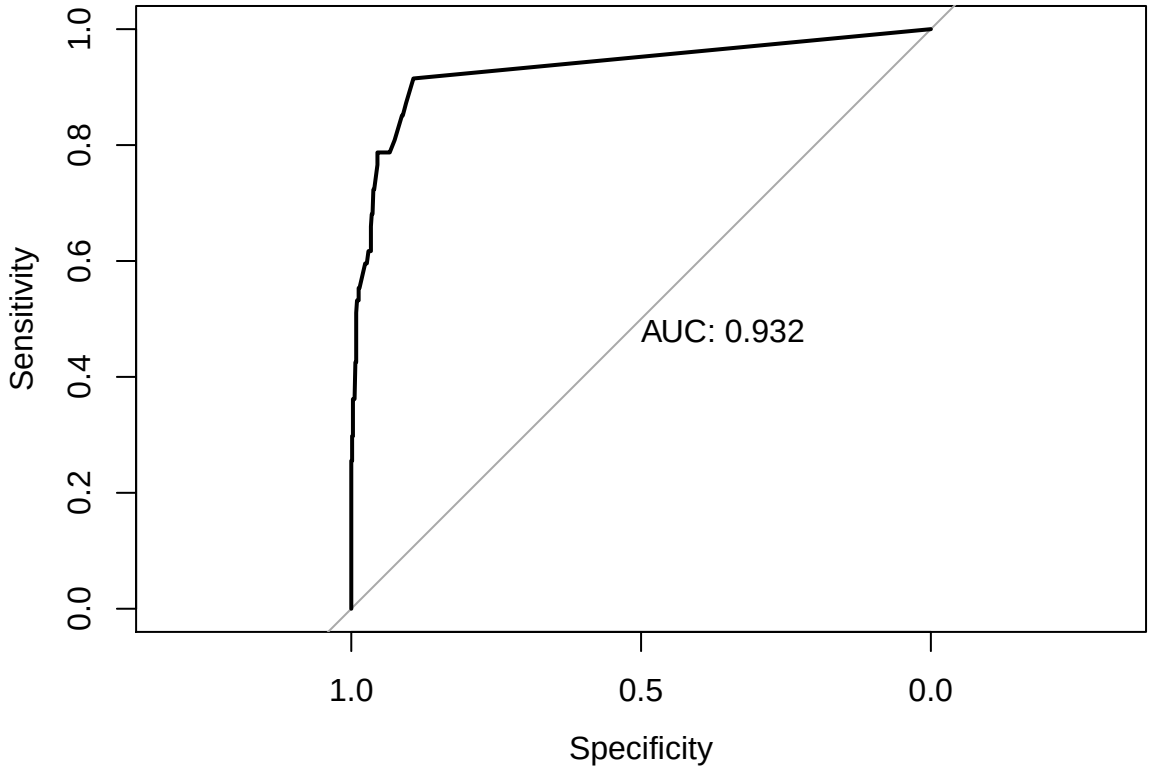
kNN

`kNN` model is based on how observations are distributed in space. We just look what number of k closest neighbors belong to each class.

Standard tests

```
## [1] "AUPRC: 0.676345223465442"
```

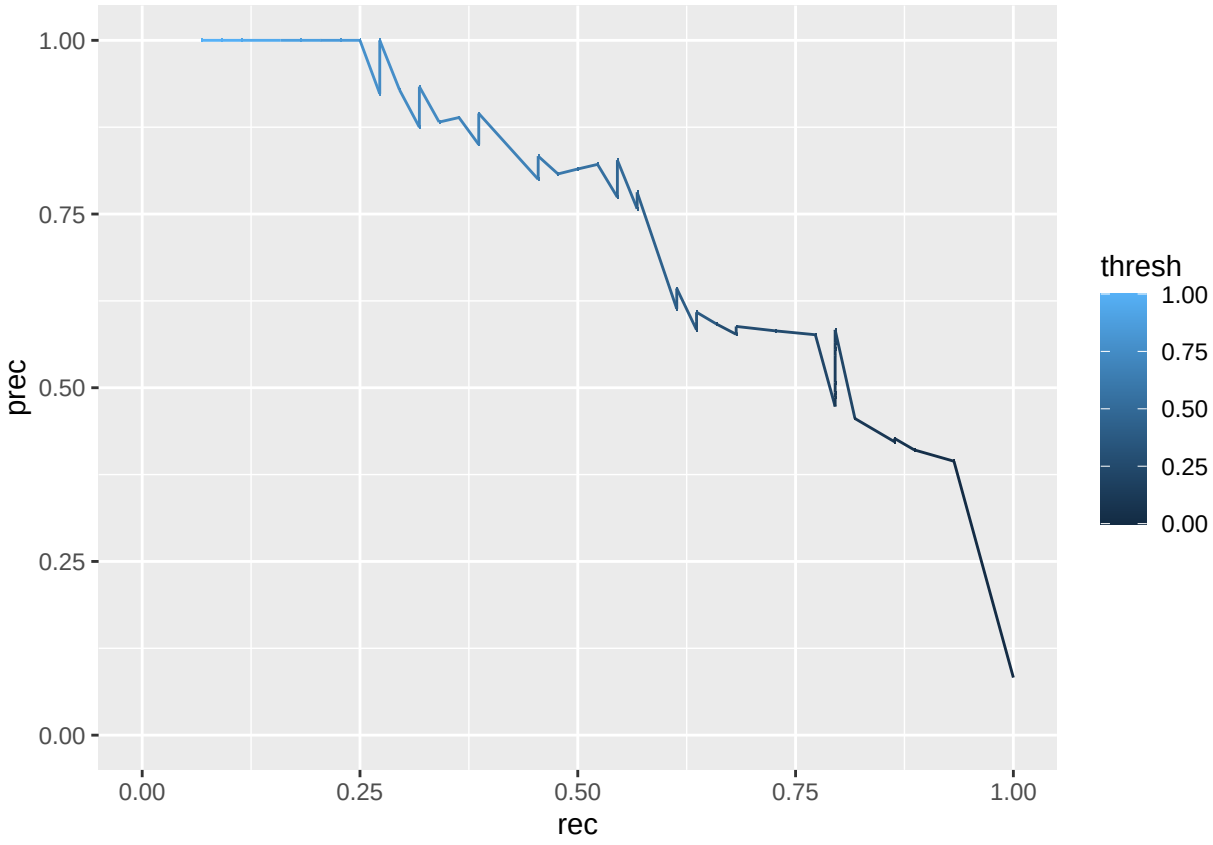

prec
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
rec
thresh
1.00
0.75
0.50
0.25
0.00

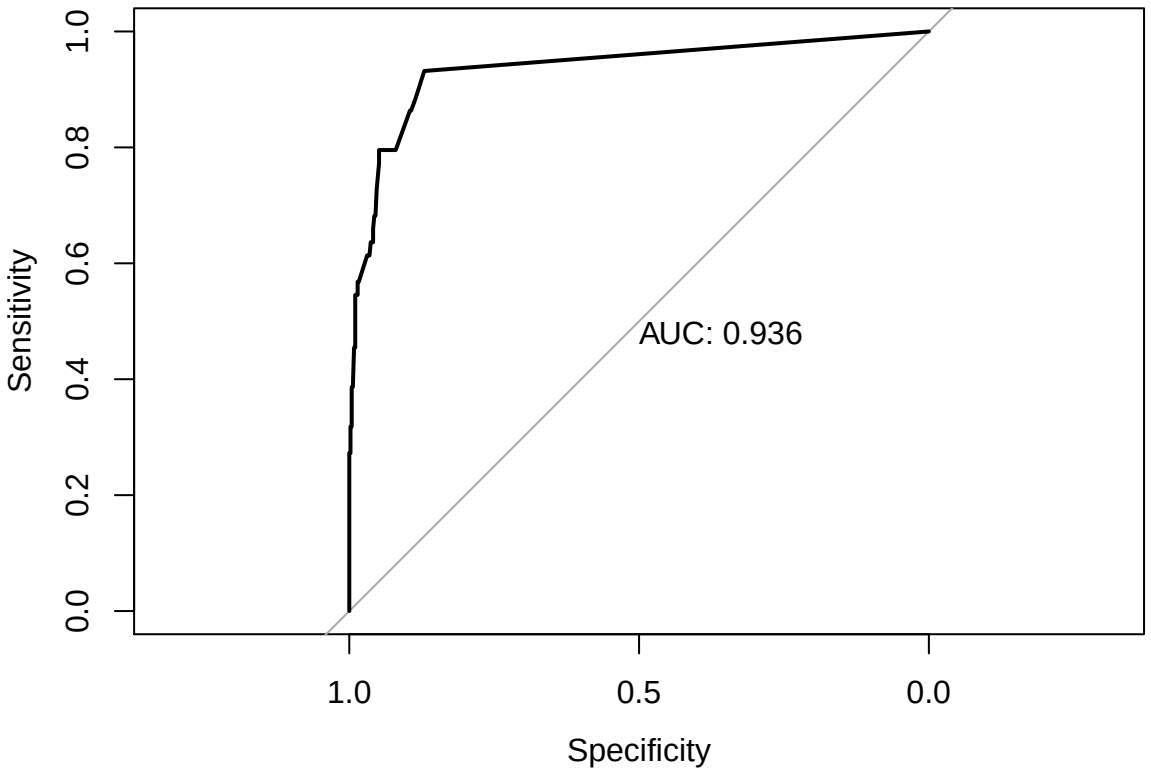

Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
AUC: 0.932
1.0
0.5
0.0
Specificity

Tests with removed rows with missing values

I just was curious how removing rows of testing data, instead of imputing them, will affect our scores.

```
## [1] "AUPRC: 0.70999610989033"
```





Although it was kind of predictable I find this experiment very informative, because sometimes we want to make something quickly and easily, but we end up with messed up scores and results. There we got better scores than we should and in a real life task it could have devastating effects on our project.

Summary

It is beyond doubt that some problems require really complicated solutions, but in many cases we can use much simpler and much more interpretable models. This case is an example that even with models like logistic regression or decision tree we can get pretty good results. What is more, when we know how exact model works we can tune it ourselves or augment our data a bit to boost the efficiency of model. One last thing is very interesting. When we take a look at `AUC` and `AUPRC` measures of models we can see that `AUC` is almost the same for all of the cases, when `AUPRC` differ significantlly.