Sick dataset analysis

Wojciech Bogucki

17/04/2020

Contents

Explanatory Data Analysis

First look at raw data

Firstly, let's have a look at selected training data from the original dataset *sick*:

Table 1: Features of dataset sick

rows	3016
columns	30
discrete_columns	23
continuous_columns	6
all_missing_columns	1
total_missing_values	4878
complete_rows	0
total_observations	90480

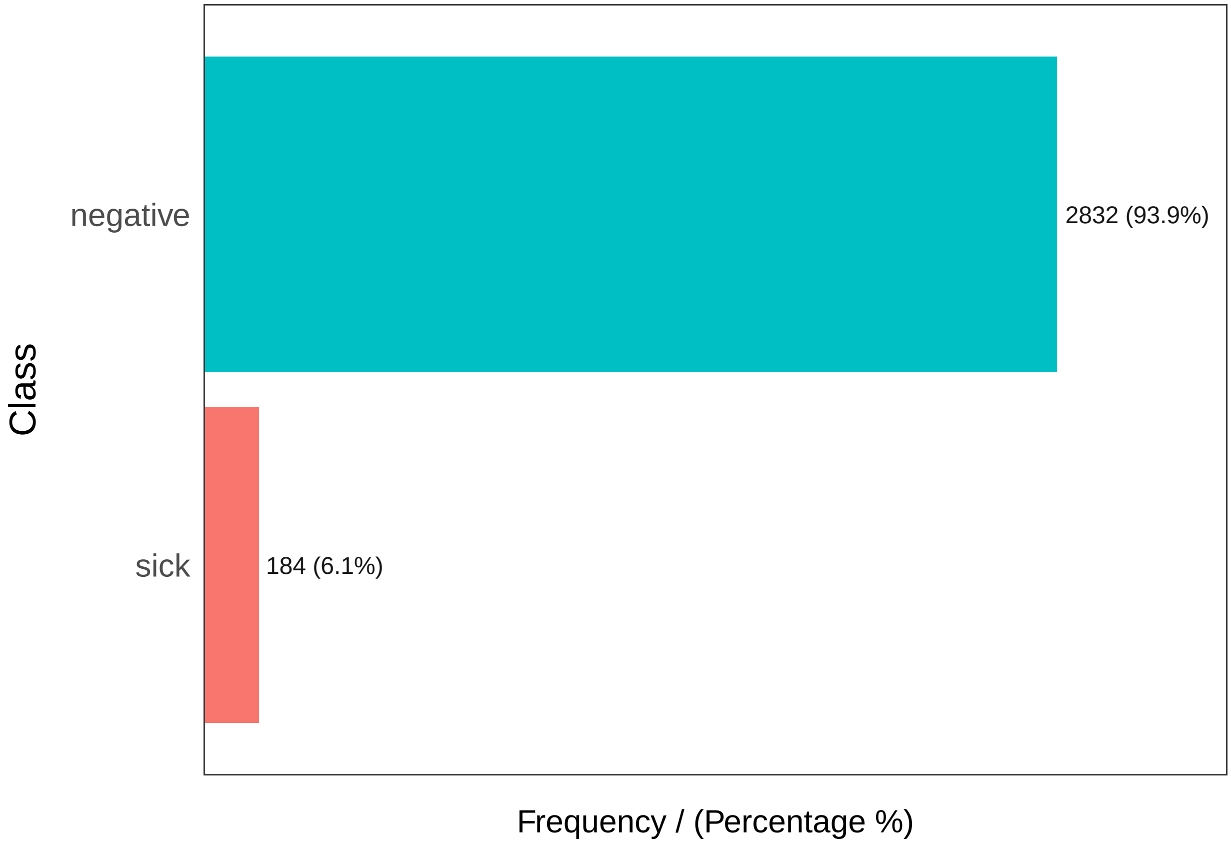
Table 2: Metrics for each variable of dataset sick

variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
age	0	0	0	0.00	0	0	numeric	88
sex	0	0	119	3.95	0	0	factor	2
on_thyroxine	0	0	0	0.00	0	0	factor	2
query_on_thyroxine	0	0	0	0.00	0	0	factor	2
on_antithyroid_medication	0	0	0	0.00	0	0	factor	2
sick	0	0	0	0.00	0	0	factor	2
pregnant	0	0	0	0.00	0	0	factor	2
thyroid_surgery	0	0	0	0.00	0	0	factor	2
I131_treatment	0	0	0	0.00	0	0	factor	2
query_hypothyroid	0	0	0	0.00	0	0	factor	2
query_hyperthyroid	0	0	0	0.00	0	0	factor	2
lithium	0	0	0	0.00	0	0	factor	2
goitre	0	0	0	0.00	0	0	factor	2
tumor	0	0	0	0.00	0	0	factor	2
hypopituitary	0	0	0	0.00	0	0	factor	1
psych	0	0	0	0.00	0	0	factor	2
TSH_measured	0	0	0	0.00	0	0	factor	2
TSH	0	0	301	9.98	0	0	numeric	272
T3_measured	0	0	0	0.00	0	0	factor	2
T3	0	0	621	20.59	0	0	numeric	67
TT4_measured	0	0	0	0.00	0	0	factor	2
TT4	0	0	191	6.33	0	0	numeric	222
T4U_measured	0	0	0	0.00	0	0	factor	2
T4U	0	0	316	10.48	0	0	numeric	140
FTI_measured	0	0	0	0.00	0	0	factor	2
FTI	0	0	314	10.41	0	0	numeric	217
TBG_measured	0	0	0	0.00	0	0	factor	1
TBG	0	0	3016	100.00	0	0	numeric	0
referral_source	0	0	0	0.00	0	0	factor	5
Class	0	0	0	0.00	0	0	factor	2

As we can see in the table, variable `TBG` has 100% missing values and variables `TBG_measured` and `hypopituitary` have only one unique value. Therefore, I removed this three variables from training and test datasets.

Our target class ‘Class’ is very imbalanced.

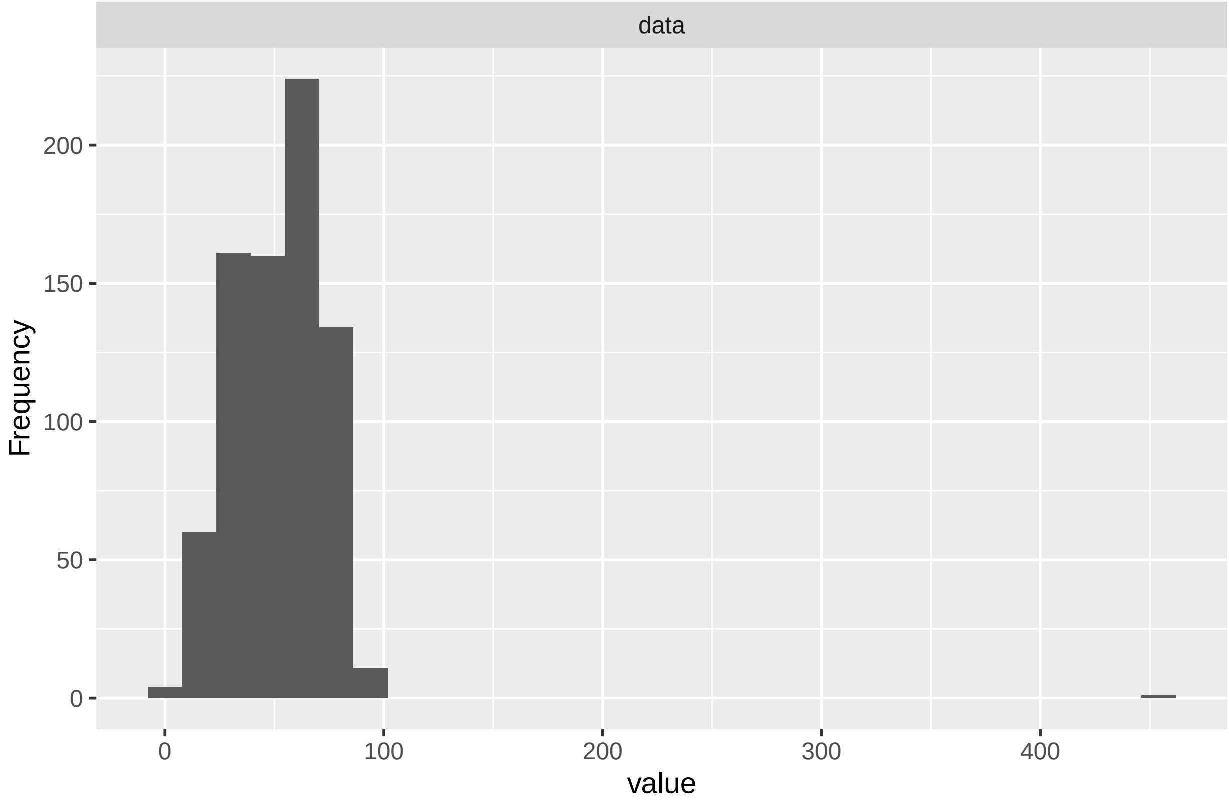
```
freq(sick_train, input=c('Class'))
```



```
##      Class frequency percentage cumulative_perc
## 1 negative      2832       93.9            93.9
## 2     sick       184        6.1           100.0
```

Mistakes in data

When we look closely at the continuous variables in test dataset, we can observe that age variable has some big values.



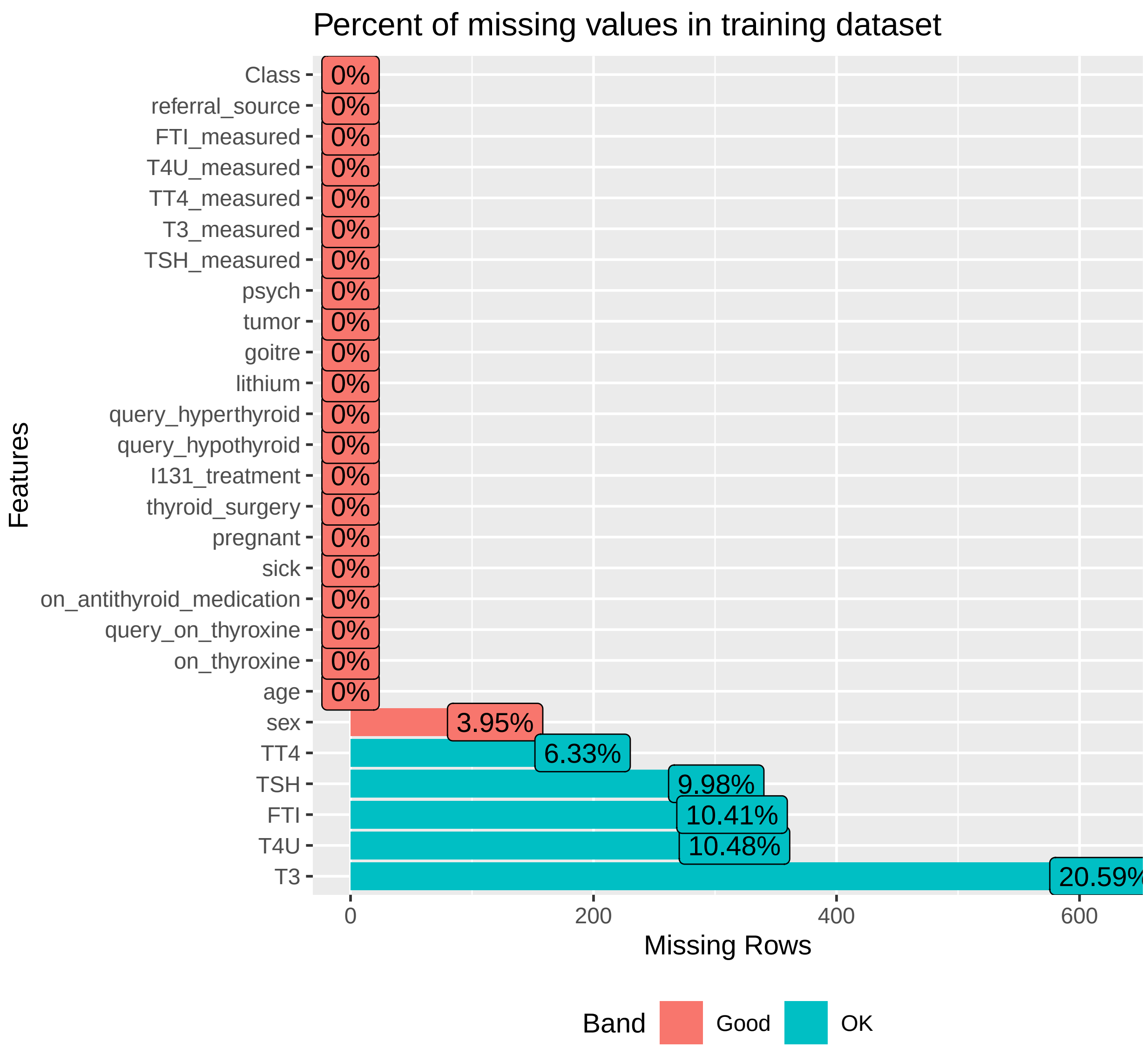
```
## sick[, "age"]
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3771        1       93        1    51.74    21.98       20       25
##      .25      .50      .75      .90      .95
##       36       54       67       75       79
##
## lowest :   1   2   4   5   6, highest:  91  92  93  94 455
```

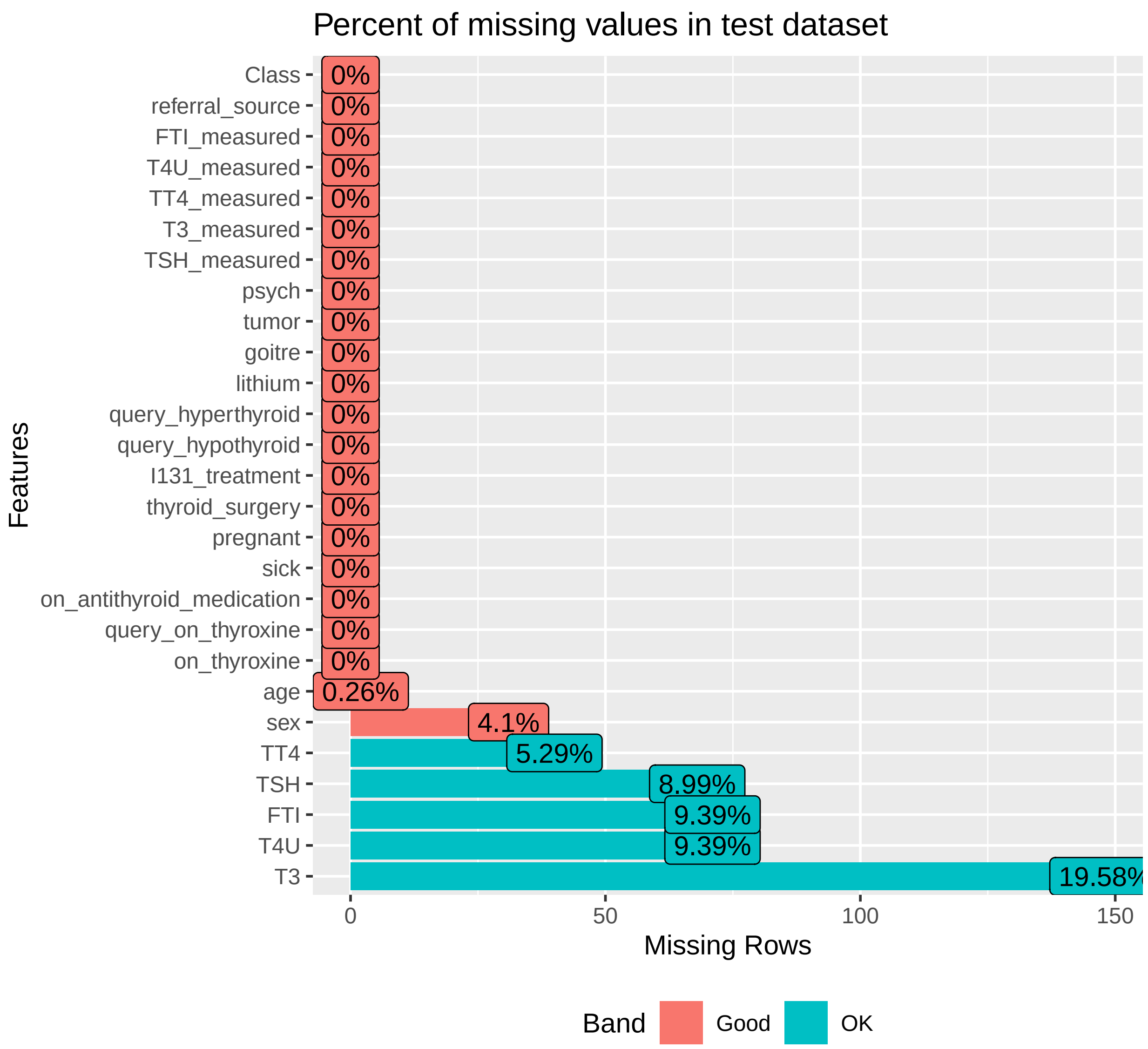
Age for one observation is 455, which is obviously a mistake. To avoid such mistakes data will be filtrated. Impossible age values will be changed to missing values.

```
sick_train <- sick_train %>% mutate(age=replace(age, age>130 | age<0, NA))
sick_test <- sick_test %>% mutate(age=replace(age, age>130 | age<0, NA))
```

Missing data

Dataset contains some missing values.


Percent of missing values in training dataset
Features
Class
referral_source
FTI_measured
T4U_measured
TT4_measured
T3_measured
TSH_measured
psych
tumor
goitre
lithium
query_hyperthyroid
query_hypothyroid
I131_treatment
thyroid_surgery
pregnant
sick
on_antithyroid_medication
query_on_thyroxine
on_thyroxine
age
sex
TT4
TSH
FTI
T4U
T3
0%
3.95%
6.33%
9.98%
10.41%
10.48%
20.59%
0
200
400
600
Missing Rows
Band
Good
OK

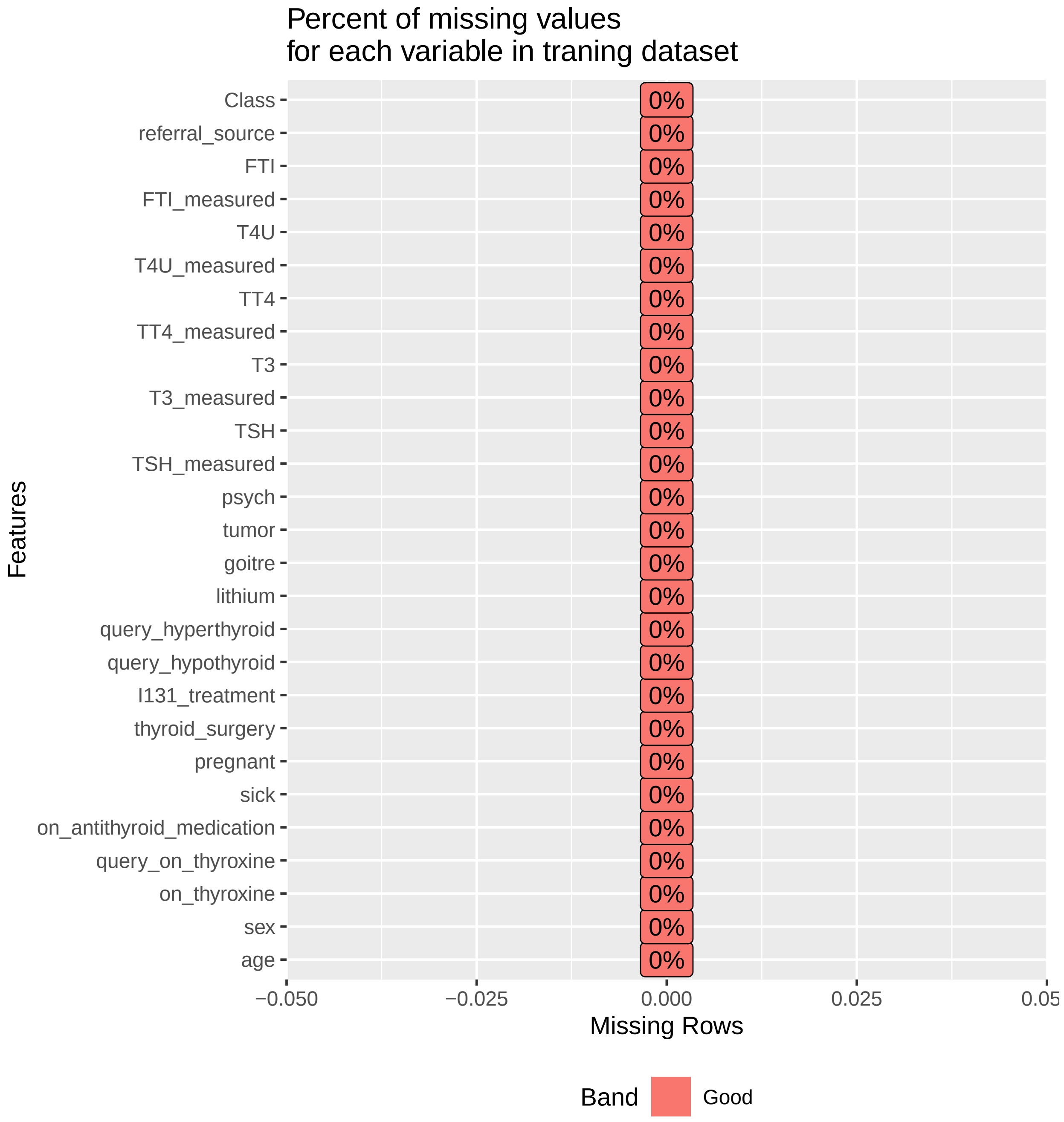


Imputation

To eliminate missing data in training set I used package `mice`.

Table 3: Impuation method for each variable

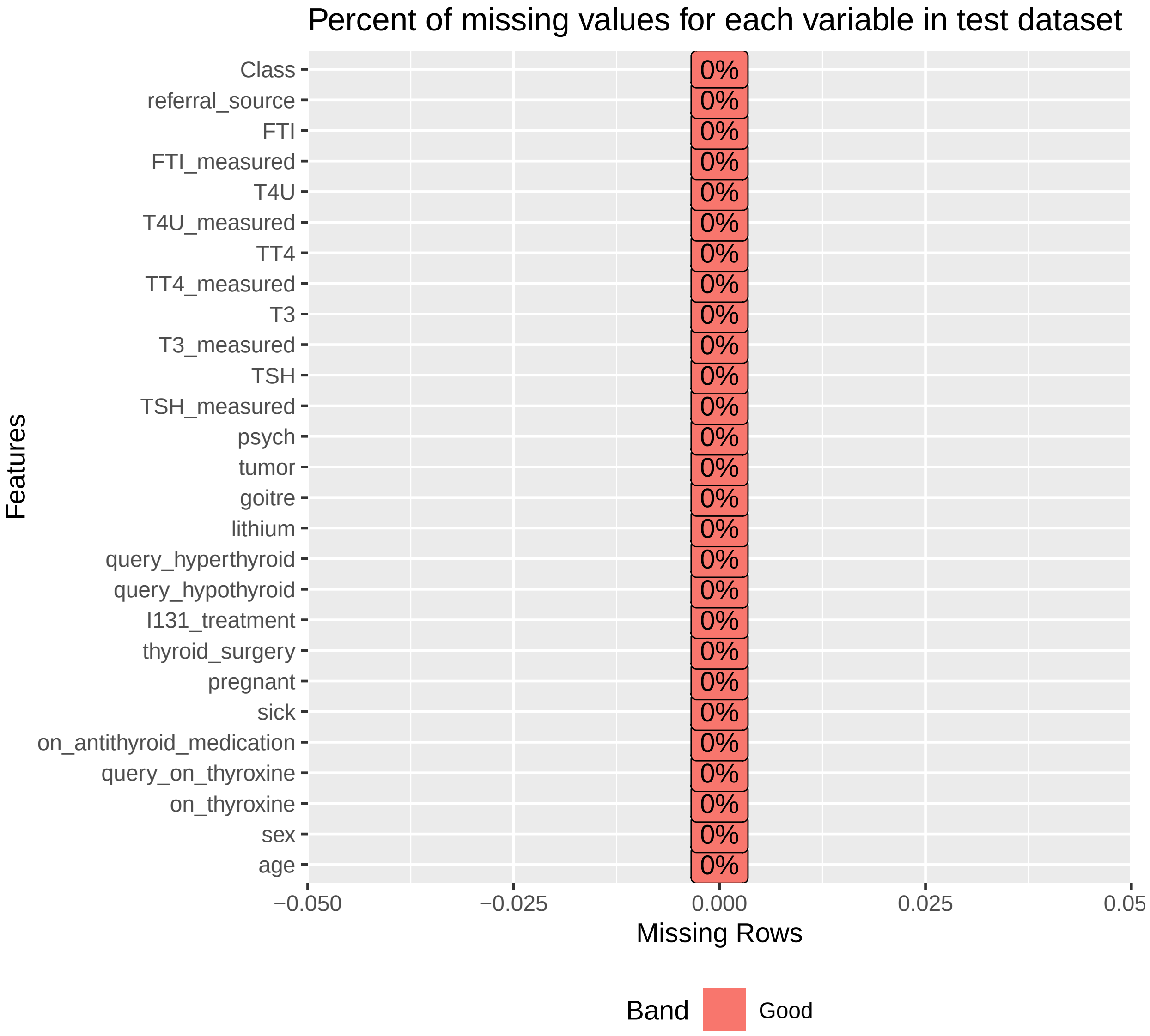
variable	imputaton method
sex	Logistic regression
TSH	Predictive mean matching
T3	Predictive mean matching
TT4	Predictive mean matching
T4U	Predictive mean matching
FTI	Predictive mean matching



Then I performed imputation on dataset containing training data after imputation and test data.

Table 4: Impuation method for each variable

variable	imputaton method
age	Predictive mean matching
sex	Logistic regression
TSH	Predictive mean matching
T3	Predictive mean matching
TT4	Predictive mean matching
T4U	Predictive mean matching
FTI	Predictive mean matching



Creating models

For prediction I chose three interpretable models:

- logistic regression
- decision tree
- naive Bayes

Also decision tree and naive Bayes will be trained on dataset with missing values. ## Training

During training I performed hyperparameters tuning on decision tree. These hyperparameteres were used:

- Decision tree:

```
## $minsplit
## [1] 44
##
## $minbucket
## [1] 6
##
## $cp
## [1] 0.01295749
```

- Decision tree with missing values:

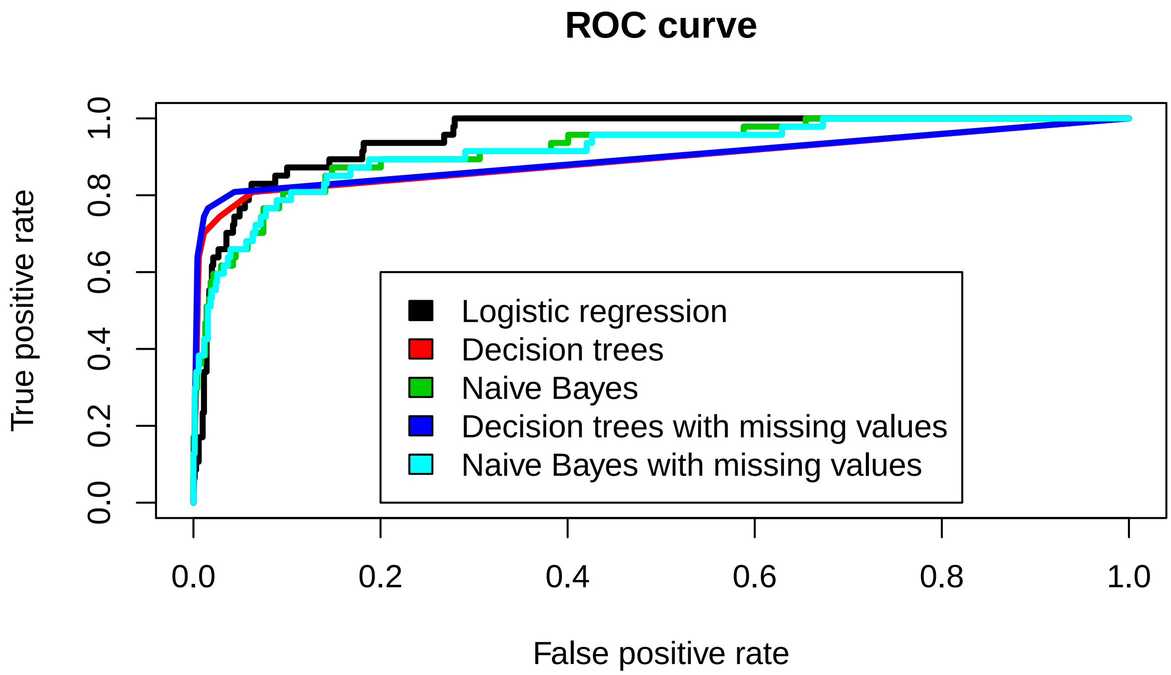
```
## $minsplit
## [1] 36
##
## $minbucket
## [1] 16
##
## $cp
## [1] 0.008562876
```

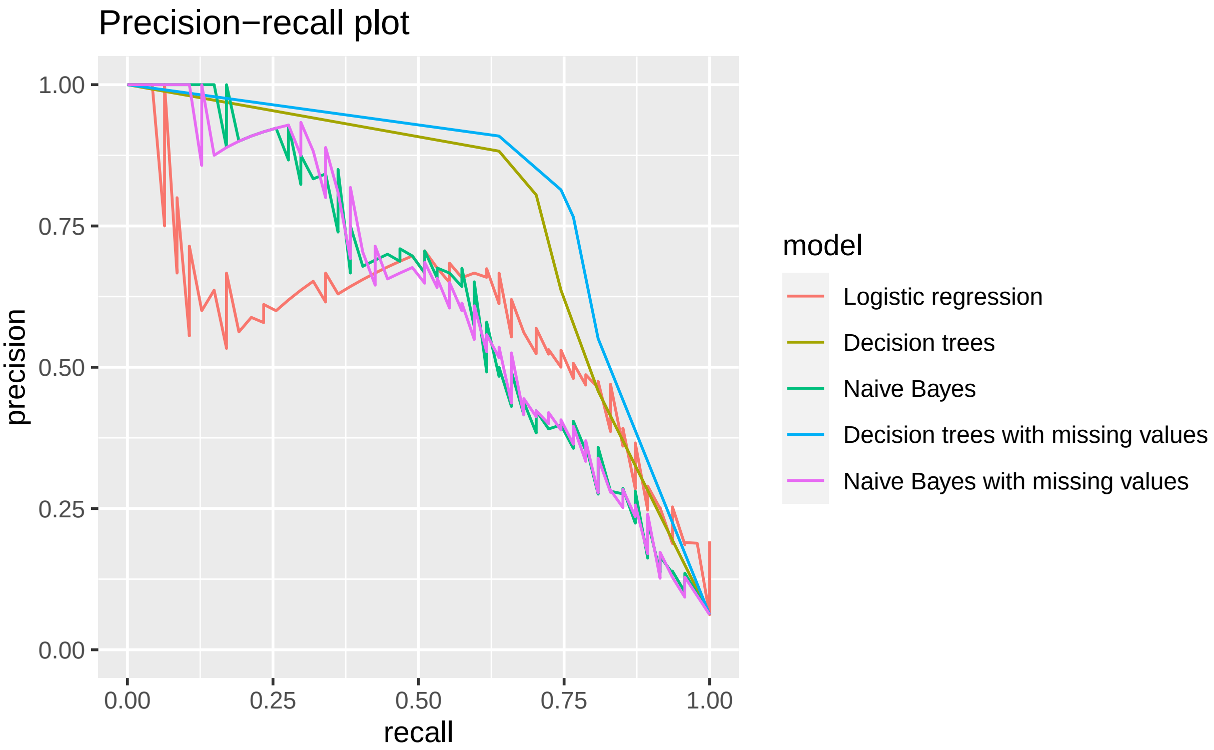
Measures

Next, I calculated measures of goodness of predicton: agggregated AUC from 5-fold crossvalidation on training set, AUC on test set and AUPRC on test set. I also created a comparison of ROC curve and precision-recall plot for each model.

Table 5: Measures of goodness of prediction for each model

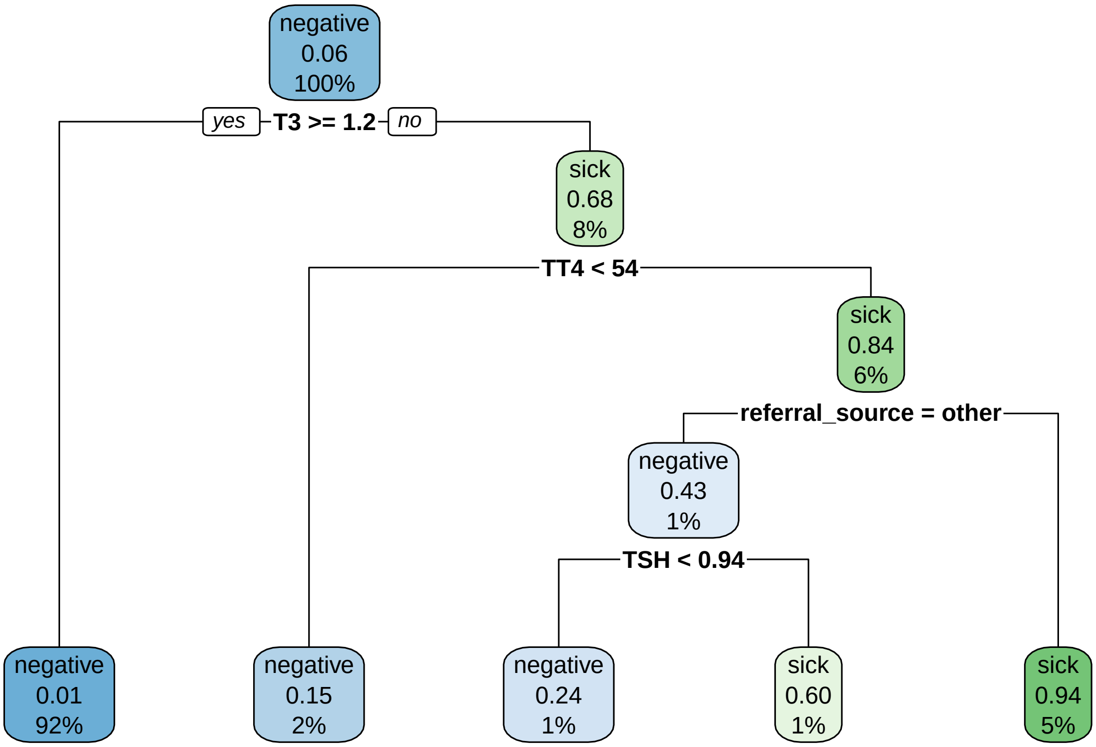
model	auc.5.crossvalidation	auc	auprc
Logistic regression	0.9315380	0.9518351	0.5546012
Decision trees	0.9501499	0.8920715	0.6827865
Naive Bayes	0.9247676	0.9192150	0.6039720
Decision trees with missing values	0.9533666	0.8963479	0.7184984
Naive Bayes with missing values	0.9250840	0.9167242	0.5985010


ROC curve
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
■ Logistic regression
■ Decision trees
■ Naive Bayes
■ Decision trees with missing values
■ Naive Bayes with missing values


Precision−recall plot
precision
1.00
0.75
0.50
0.25
0.00
recall
0.00
0.25
0.50
0.75
1.00
model
Logistic regression
Decision trees
Naive Bayes
Decision trees with missing values
Naive Bayes with missing values

Conclusion

Decision tree with and without missing data has better aggregated AUC from 5-fold crossvalidation on training data than other models. Surprisingly they have the worst AUC on test data, but also best AUPRC on test data and this measure is more important due to imbalanced target class. ALos in this case is better to leave missing values for model to handle than doing imputation. Decion tree from package `rpart` is an example of an interpretable model. We can see the decision tree.


negative
0.06
100%
yes
T3 >= 1.2
no
sick
0.68
8%
TT4 < 54
sick
0.84
6%
referral_source = other
negative
0.43
1%
TSH < 0.94
negative
0.01
92%
negative
0.15
2%
negative
0.24
1%
sick
0.60
1%
sick
0.94
5%