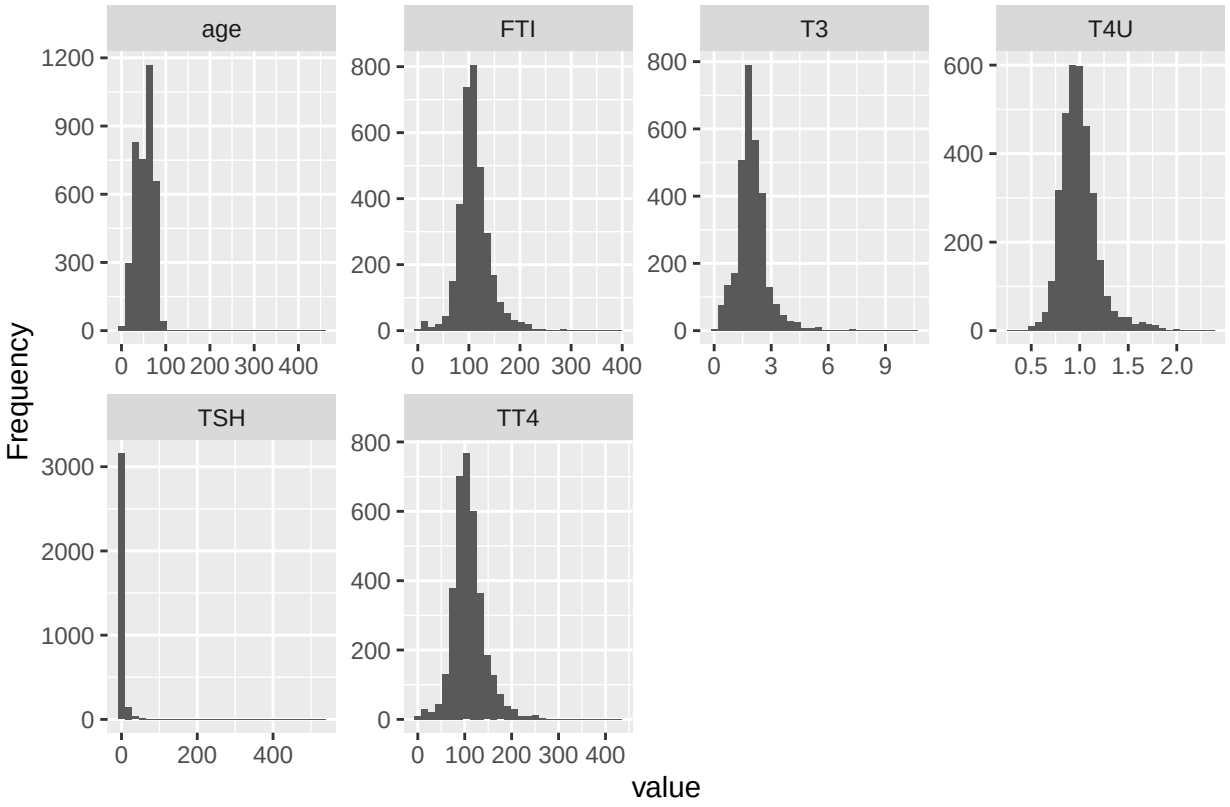
Sick dataset analysis

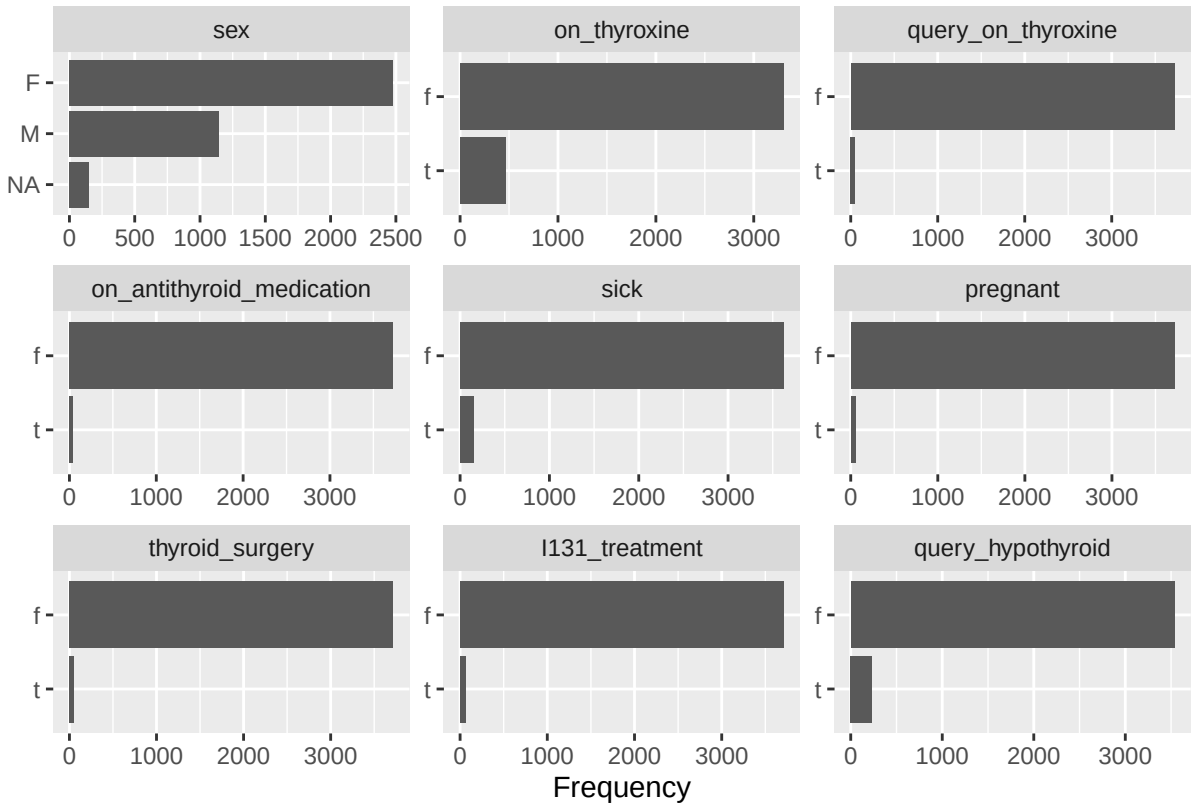
Olaf Werner

17 04 2020

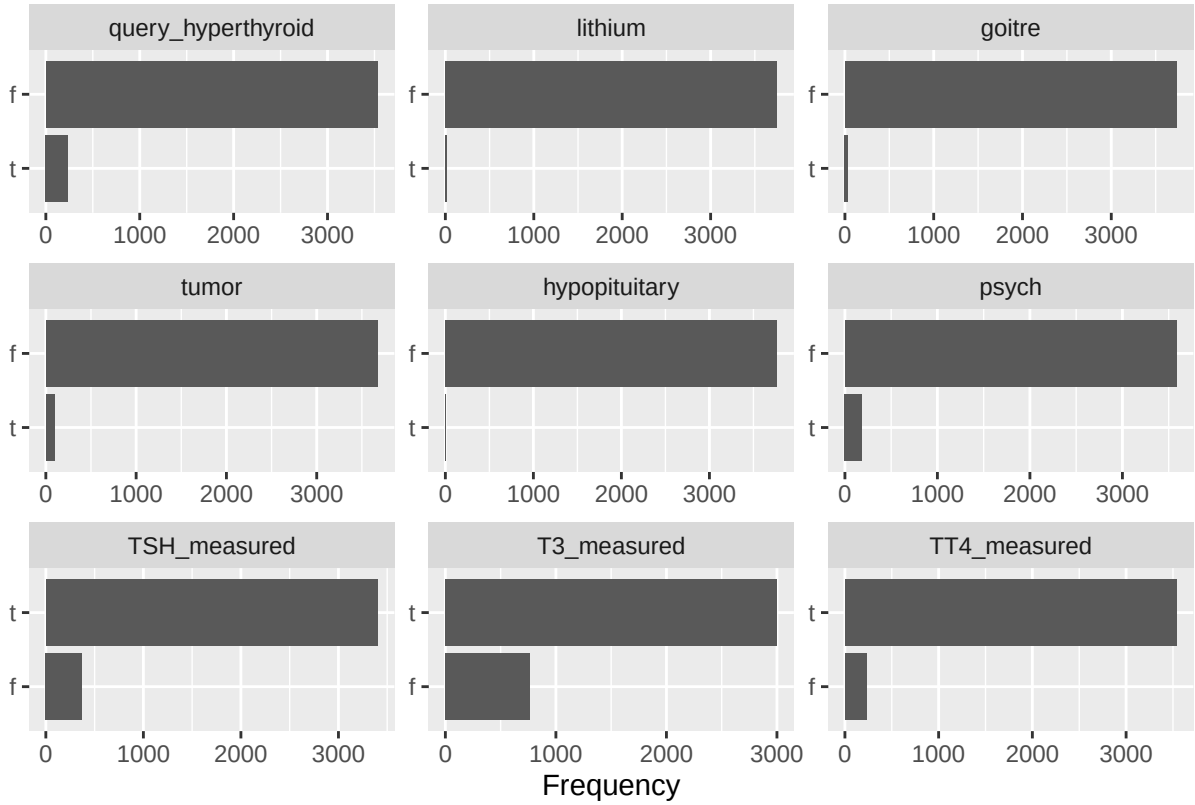
```
DataExplorer::plot_histogram(dataset_raw)
```



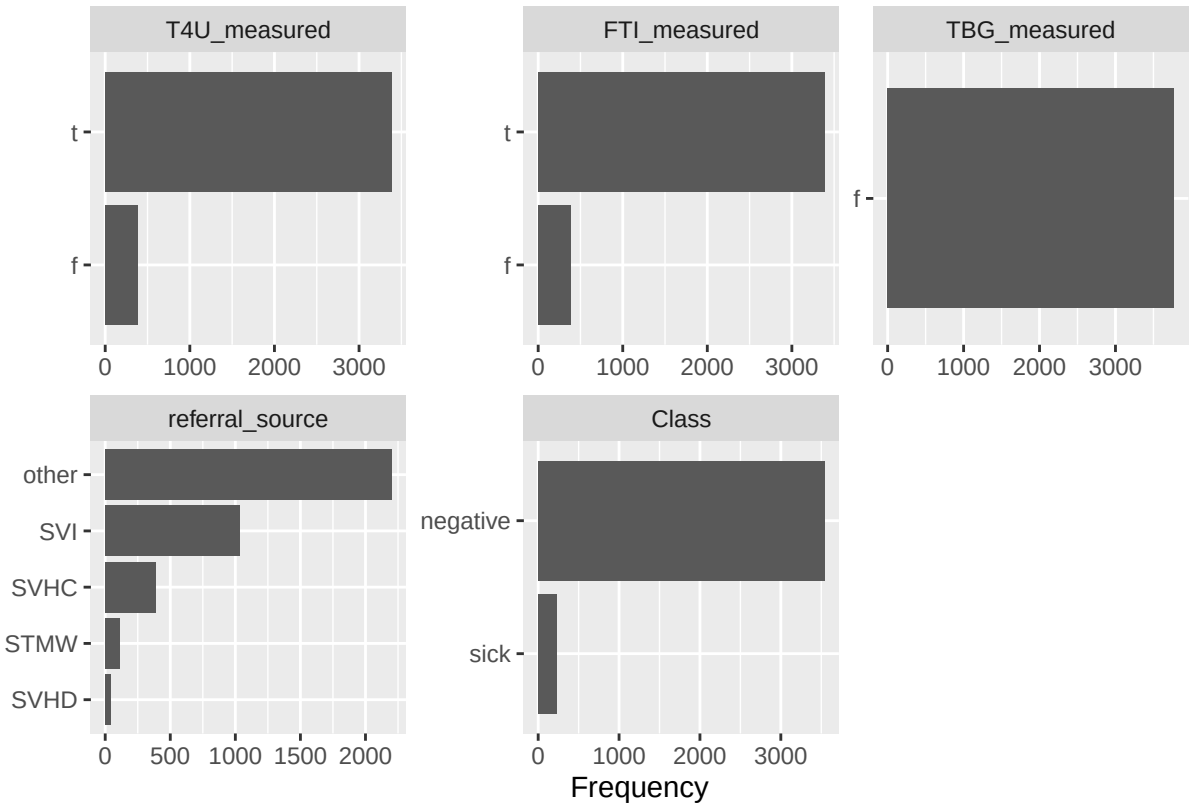

```
DataExplorer::plot_bar(dataset_raw)
```


sex
F
M
NA
0 500 1000 1500 2000 2500
on_thyroxine
f
t
0 1000 2000 3000
query_on_thyroxine
f
t
0 1000 2000 3000
on_antithyroid_medication
f
t
0 1000 2000 3000
sick
f
t
0 1000 2000 3000
pregnant
f
t
0 1000 2000 3000
thyroid_surgery
f
t
0 1000 2000 3000
I131_treatment
f
t
0 1000 2000 3000
query_hypothyroid
f
t
0 1000 2000 3000
Frequency

Page 1

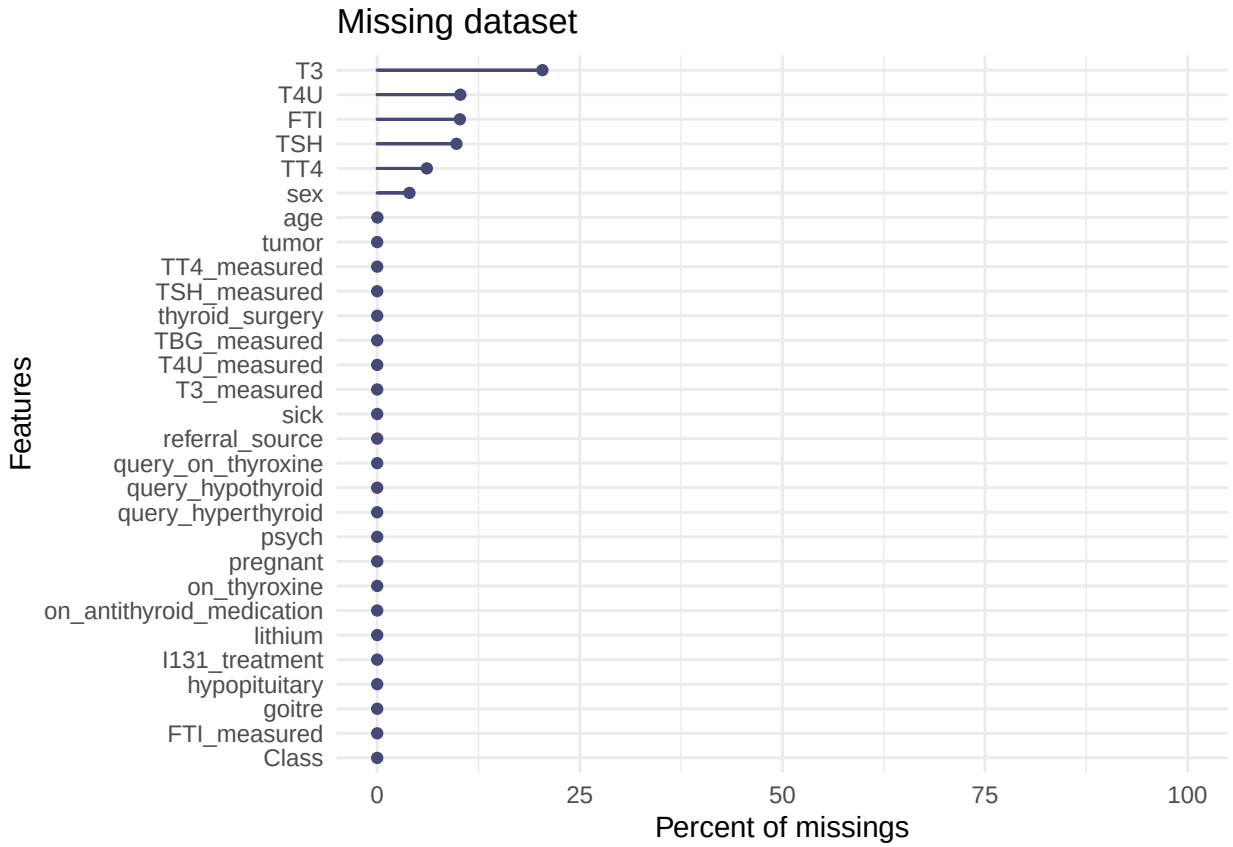

query_hyperthyroid
f
t
0 1000 2000 3000
lithium
f
t
0 1000 2000 3000
goitre
f
t
0 1000 2000 3000
tumor
f
t
0 1000 2000 3000
hypopituitary
f
t
0 1000 2000 3000
psych
f
t
0 1000 2000 3000
TSH_measured
t
f
0 1000 2000 3000
T3_measured
t
f
0 1000 2000 3000
TT4_measured
t
f
0 1000 2000 3000
Frequency

Page 2



Preprocessing

```
gg_miss_var(dataset,
            show_pct = TRUE) +
  ylim(0, 100) +
  labs(title = "Missing dataset",
       x = "Features",
       y = "Percent of missings")
```




```
# 80% train i 20% test data

# cross-validation
```

As we can see our dataset is very imbalanced on 3772 people only 231 are sick. So we are going to use AUC and AUPRC measures. Let us split our dataset to training and testing parts according to the file.

```
sick <- suppressMessages(read_csv("dataset_38_sick.csv"))
sick<-sick %>%
  mutate_if(is.character, list(~na_if(., "?")))
sick<-suppressMessages(type_convert(sick))
sick<-DataExplorer::drop_columns(sick,c("TBG","hypopituitary"))
sick<-sick %>% mutate_if(is.character,as.factor)
sick$Class<-sick$Class=="sick"
sick<-sick %>% mutate_if(is.logical,as.factor)

train_index<-suppressMessages(read_table2("indeksy_treningowe.txt"))
train_index<-train_index$y
train_sick<-sick[train_index,]
test_sick<-sick[-train_index,]
```

Because in mlr there is no AUPRC we create custom one using PRROC library.

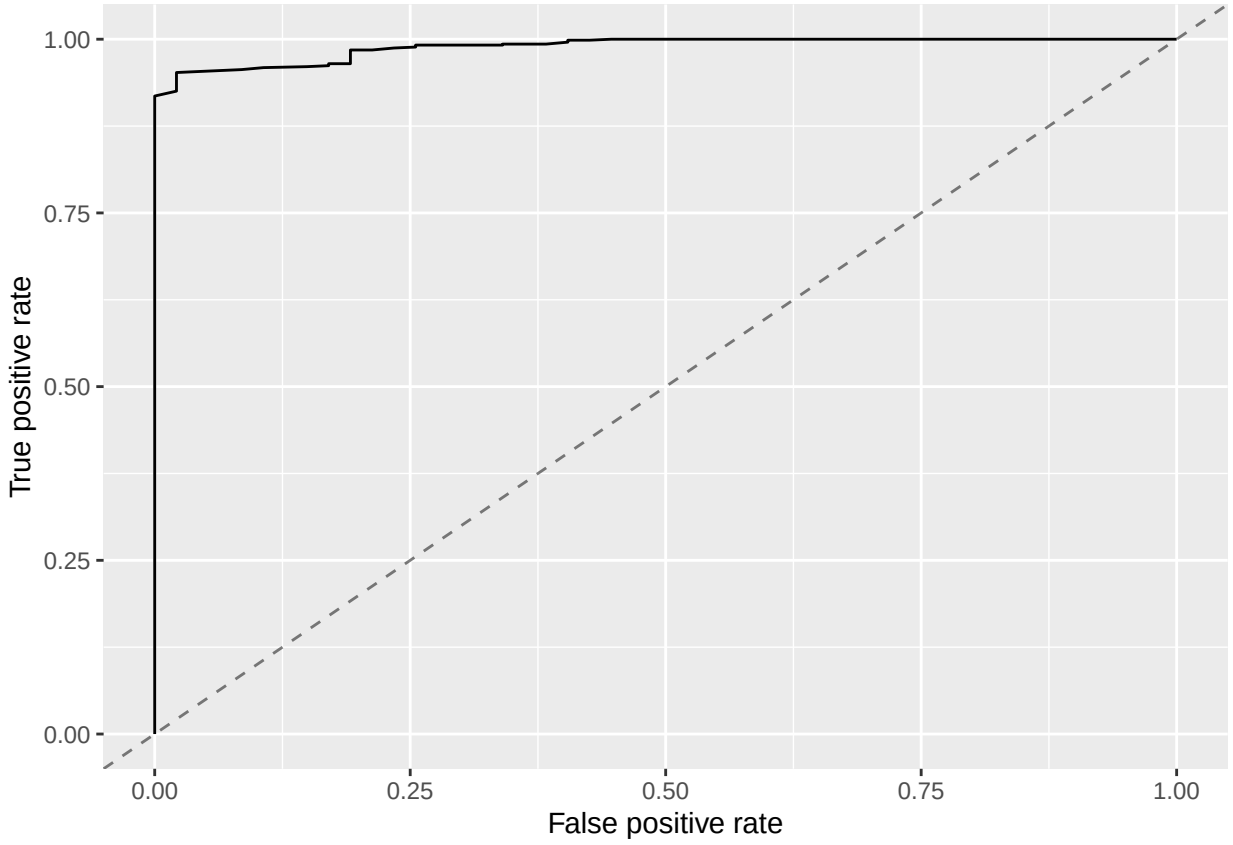
```
require(PRROC)
AUPRC2<-function(task, model, pred, feats, extra.args){
  fg<-pred$data[pred$data$truth=="TRUE",]$prob.TRUE
  bg<-pred$data[pred$data$truth=="FALSE",]$prob.TRUE
  pr <- pr.curve(scores.class0 = fg, scores.class1 = bg, curve = T)
```

```
  pr$auc.integral
}
AUPRC<-makeMeasure(id = "auprc", minimize = FALSE,
                   properties = c("classif", "req.pred", "req.truth"),
                   best = 1, worst = 0, fun = AUPRC2)
```

In our dataset not only Class column is imbalanced, so we must consider dropping some columns out. Except hypopituitary because there is only 1 observation diffrent from the rest so we drop it right now.

cforest without further cleaning

```
task<-makeClassifTask(data = train_sick, target = "Class")
rf_learner<-makeLearner("classif.cforest", predict.type = "prob")
rdesc = makeResampleDesc("CV", iters = 5)
r_rf = resample(rf_learner, task, rdesc, measures = list(mlr::auc, AUPRC),
                show.info = FALSE)

rf_model <- train(rf_learner, task)
prediction <- predict(rf_model, newdata = test_sick)

df = generateThreshVsPerfData(prediction, measures = list(fpr, tpr, mmce))
plotROCCurves(df)
```



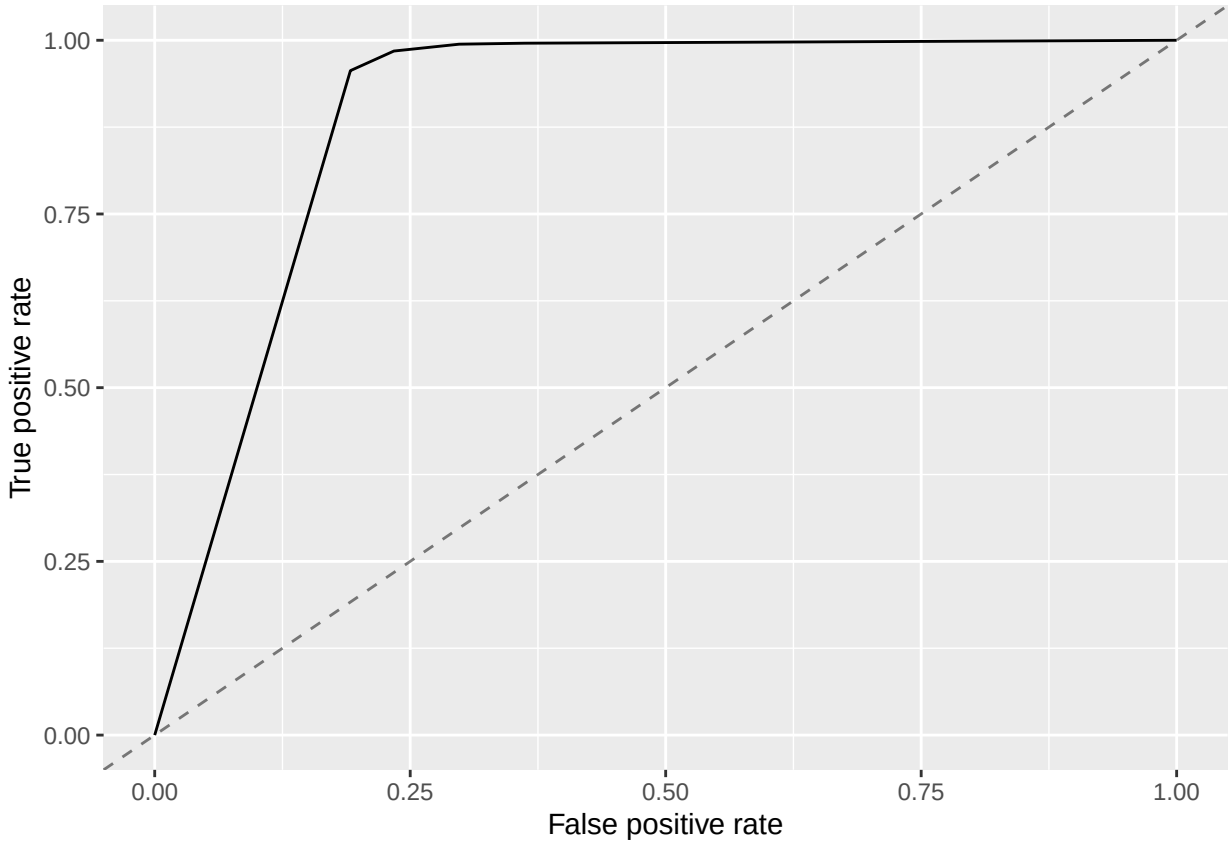

```
res_rf=mlr::performance(prediction,list(mlr::auc,AUPRC))
```

	score
auc.cross	0.990
auprc.cross	0.930
auc.test	0.989
auprc.test	0.882

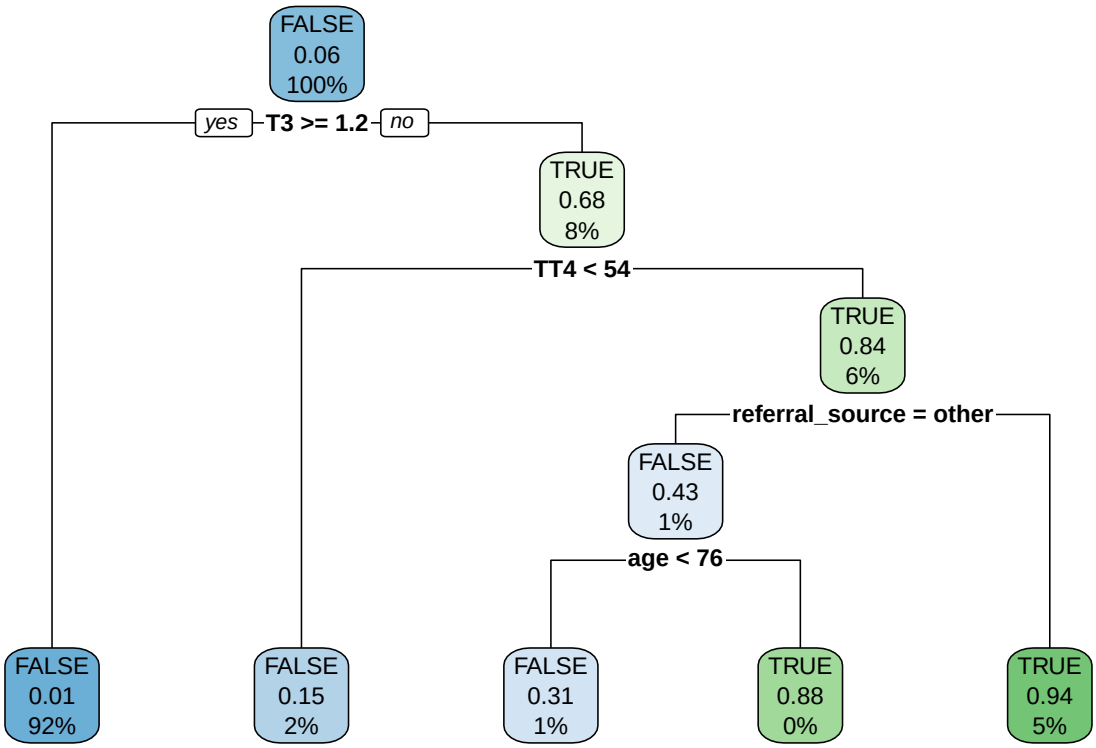
Cforest is very complex and slow model but gives us very good results. Our future models should try to be better than Cforest

Rpart with 80% relevent columns

```
task<-makeClassifTask(data = train_sick, target = "Class")
rpart_learner<-makeLearner("classif.rpart", predict.type = "prob")
fv = generateFilterValuesData(task, method ="FSelectorRcpp_information.gain")
rdesc = makeResampleDesc("CV", iters = 5)
clean_task = filterFeatures(task, fval = fv, perc = 0.8)
r_rpart = resample(rpart_learner, clean_task, rdesc, measures = list(mlr::auc, AUPRC),
                   show.info = FALSE)
rpart_model <- train(rpart_learner, clean_task)
prediction <- predict(rpart_model, newdata = test_sick)
df = generateThreshVsPerfData(prediction, measures = list(fpr, tpr, mmce))
plotROCCurves(df)
```




```
res_rpart=mlr::performance(prediction,list(mlr::auc,AUPRC))

rpart.plot::rpart.plot(rpart_model$learner.model,roundint = FALSE)
```



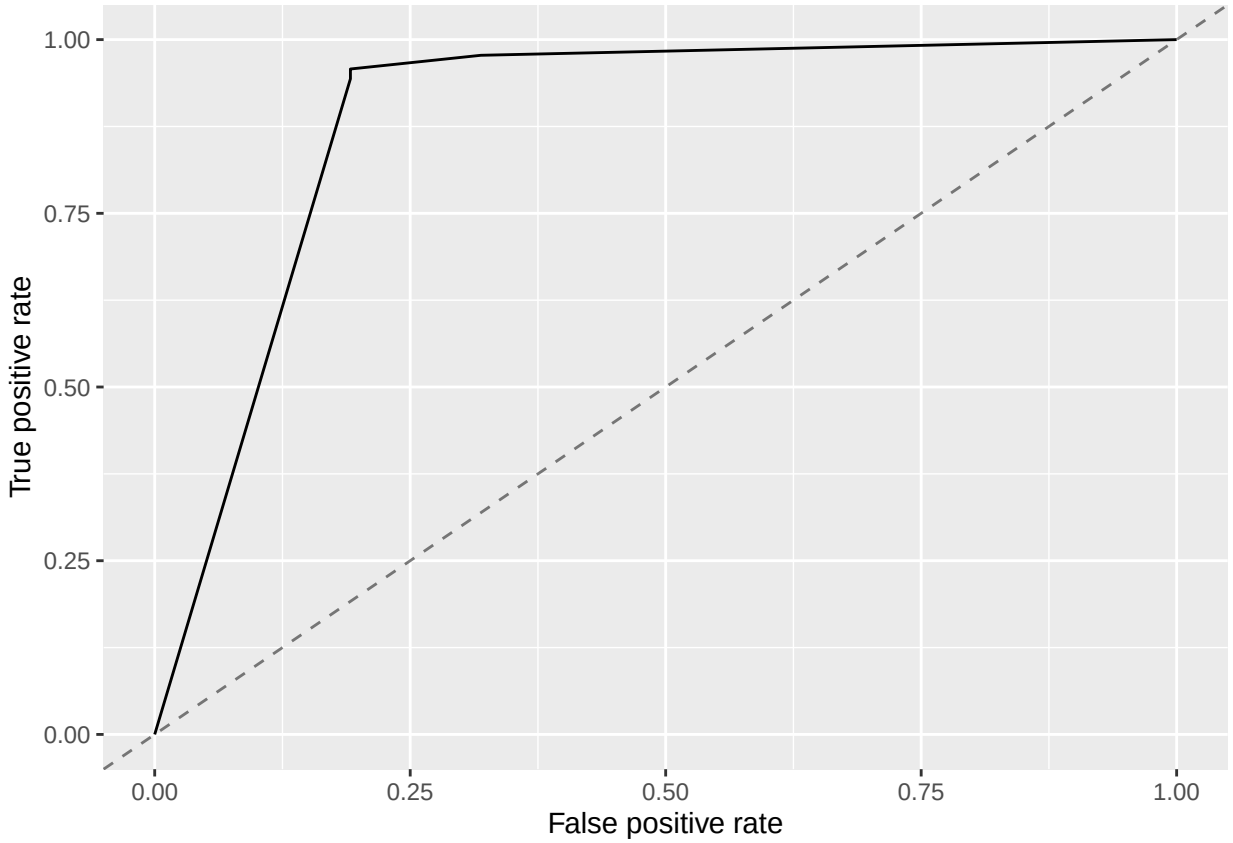
	score
auc.cross	0.957
auprc.cross	0.858
auc.test	0.896
auprc.test	0.745

Results are not as good as Cforest but model iseft is very clear and simple.

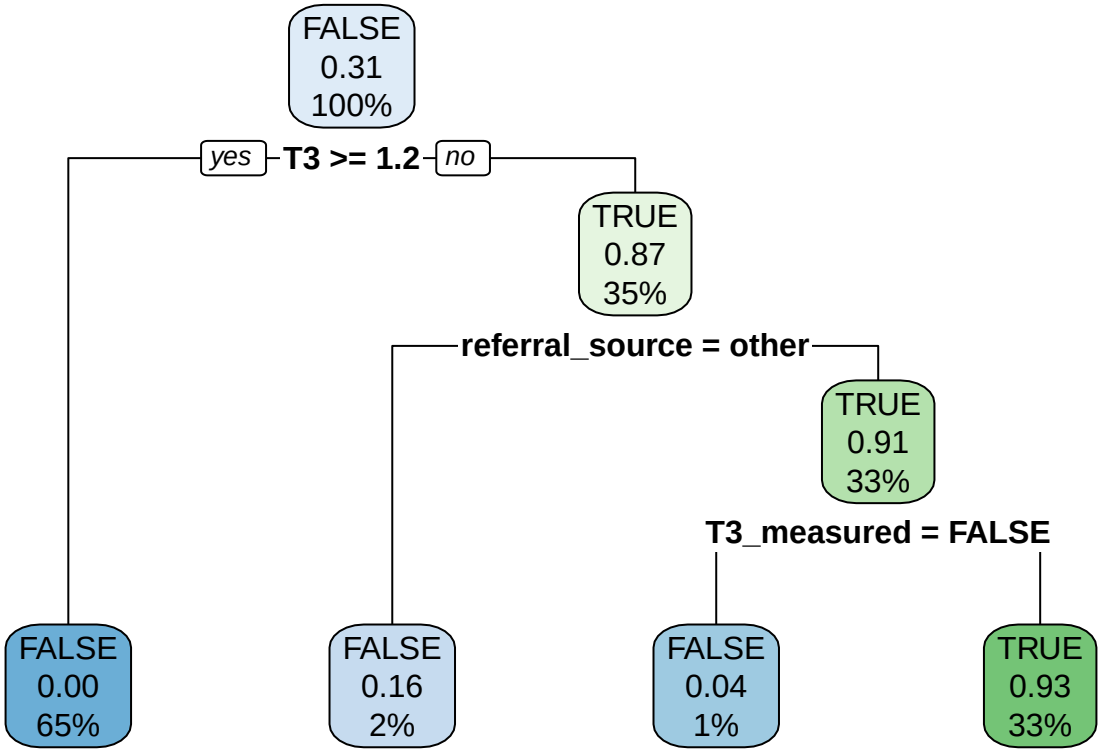
Rpart with generated dataset

```
train_sick_t<-train_sick[train_sick$Class=="TRUE",]
train_sick_f<-train_sick[train_sick$Class=="FALSE",]

random_t<-sapply(1:27,FUN = function(x){train_sick_t[sample(1:184,4000,replace = TRUE),x]})

random_f<-sapply(1:27,FUN = function(x){train_sick_f[sample(1:2832,1000,replace = TRUE),x]})

random_t<-data.frame(random_t)

random_f<-data.frame(random_f)

prediction <- predict(rf_model,newdata =  random_t)
response_t<-prediction$data$response
certainty<-abs(prediction$data$prob.TRUE-0.5)*2
sure_t<-certainty>0.3

prediction <- predict(rf_model,newdata =  random_f)
```

```
response_f<-prediction$data$response
certainty<-abs(prediction$data$prob.TRUE-0.5)*2
sure_f<-certainty>0.3

random_t$Class<-response_t

random_f$Class<-response_f

random<-rbind(random_t[sure_t,],random_f[sure_f,])

random<-rbind(random,train_sick)

task<-makeClassifTask(data = random, target = "Class")
rpart_learner<-makeLearner("classif.rpart", predict.type = "prob")
rdesc = makeResampleDesc("CV", iters = 5)
r_rpart_random = resample(rpart_learner, task, rdesc, measures = list(mlr::auc, AUPRC),show.info = FALSE

rpart_model <- train(rpart_learner, task)
prediction <- predict(rpart_model, newdata = test_sick)

df = generateThreshVsPerfData(prediction, measures = list(fpr, tpr, mmce))
plotROCCurves(df)
```




```
res_rpart_random=mlr::performance(prediction,list(mlr::auc,AUPRC))

rpart.plot::rpart.plot(rpart_model$learner.model,roundint = FALSE)
```



	score
auc.cross	0.980
auprc.cross	0.933
auc.test	0.887
auprc.test	0.558

We generated additional data by splitting training dataset into parts with TRUE and FALSE classes then we generated random data in each part by resampling rows that means: for each column we were independently chosing random row and from such combination we got new observation. Classes of such obervations were determined by cforest we trained earlier. We also chose only such rows which cforest was certain of (probablity greater than 65%). Unfortunately we got worse model.

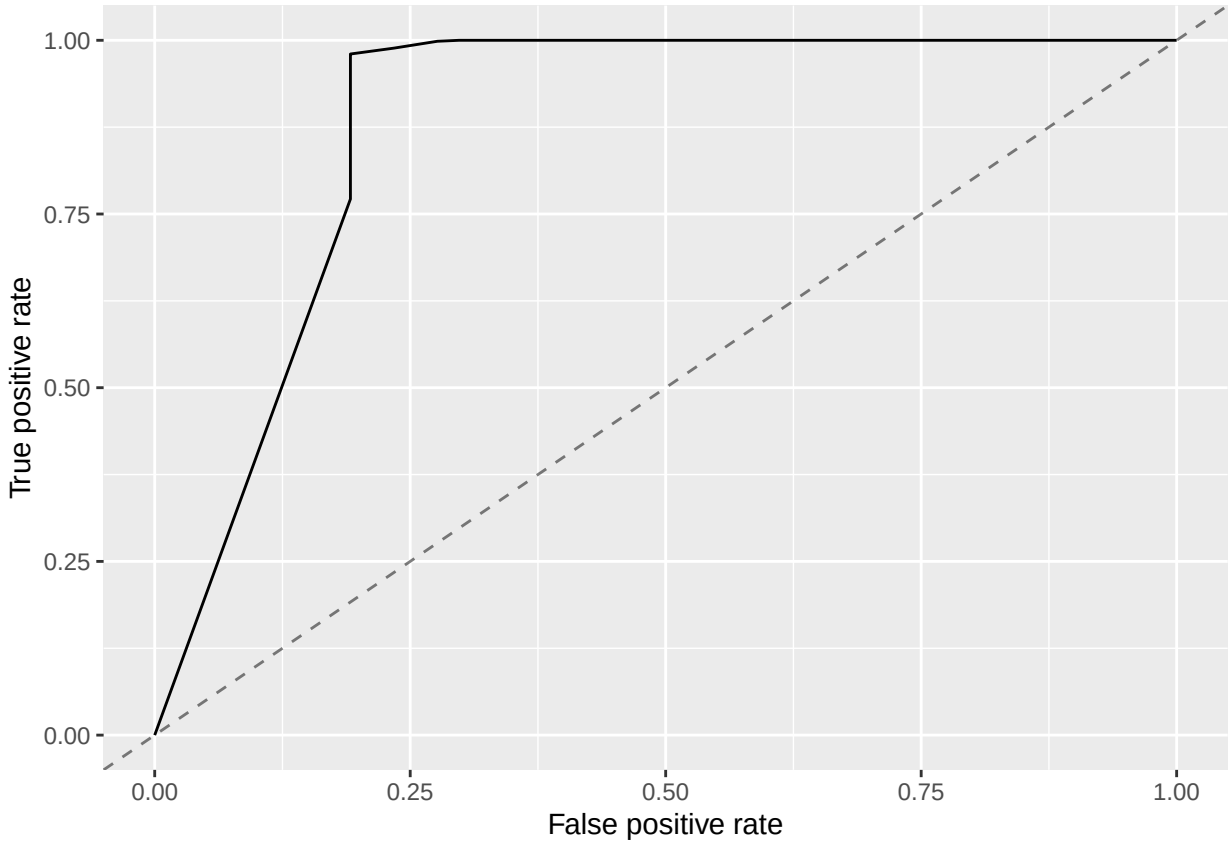
Rpart tuned

```
learner<-makeLearner("classif.rpart", predict.type = "prob")
task<-makeClassifTask(data = train_sick, target = "Class")

dt_param <- makeParamSet(
  makeDiscreteParam("minsplit", values=seq(5,10,1)),
  makeDiscreteParam("minbucket", values=seq(round(5/3,0), round(10/3,0), 1)),
  makeNumericParam("cp", lower = 0.01, upper = 0.05),
  makeDiscreteParam("maxcompete", values=6),
  makeDiscreteParam("usesurrogate", values=0),
  makeDiscreteParam("maxdepth", values=5) )

ctrl = makeTuneControlGrid()
rdesc = makeResampleDesc("CV", iters = 5L, stratify=TRUE)
(dt_tuneparam <- tuneParams(learner=learner,
```

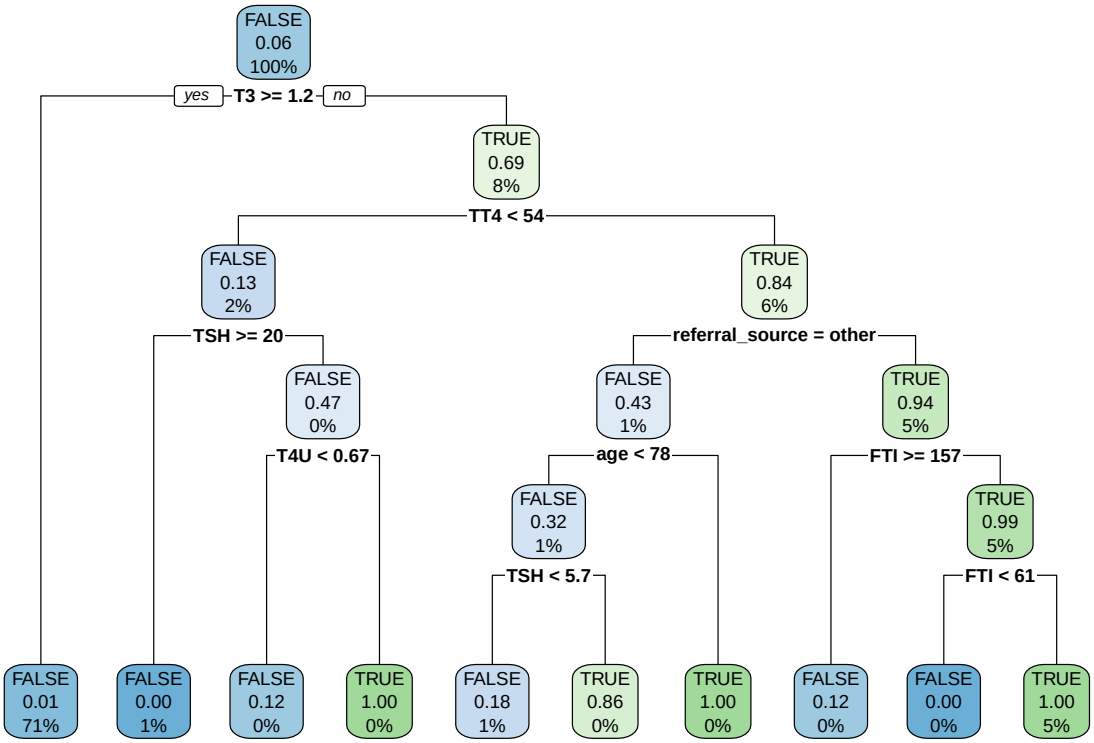
```
                              resampling=rdesc,
                              measures=list(AUPRC,mlr::acc),
                              par.set=dt_param,
                              control=ctrl,
                              task=task,
                              show.info = FALSE) )
```

```
## Tune result:
## Op. pars: minsplit=9; minbucket=2; cp=0.01; maxcompete=6; usesurrogate=0; maxdepth=5
## auprc.test.mean=0.8856749,acc.test.mean=0.9887302
```

```
dtree <- setHyperPars(learner, par.vals = dt_tuneparam$x)

rdesc = makeResampleDesc("CV", iters = 5)
r_rpart_tuned = resample(dtree, task, rdesc, measures = list(mlr::auc, AUPRC),show.info = FALSE)

dtree_train <- train(learner=dtree, task=task)
prediction <- predict(dtree_train, newdata = test_sick)

df = generateThreshVsPerfData(prediction, measures = list(fpr, tpr, mmce))
plotROCCurves(df)
```




```
res_rpart_tuned=mlr::performance(prediction,list(mlr::auc,AUPRC))

rpart.plot::rpart.plot(dtree_train$learner.model,roundint = FALSE)
```

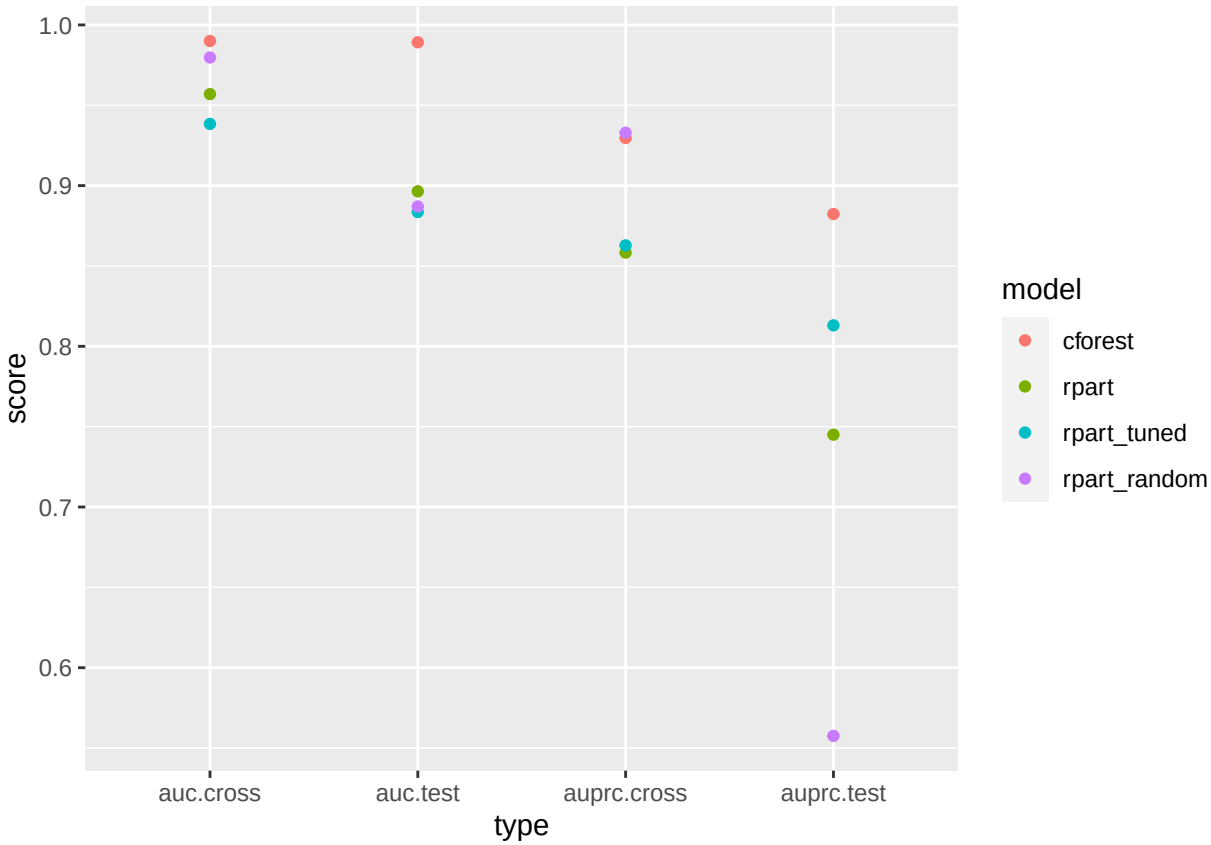


	score
auc.cross	0.938
auprc.cross	0.863
auc.test	0.884
auprc.test	0.813

Model after tuning performs a bit better than earlier, but is also more complicated.

Summary

```
ggplot(all_score,aes(x=type,y=score,color=model))+geom_point()
```



Analizing this plot we can observe many interesting things. First of all our aproach with randomly generated data did not work. During cross validation on training data set it was getting similiar results to the cforest, but on test dataset it was getting worse results. This aproach may require more experiments and theoretical knownledge. Our tuned rpart was always bit worse during cross validation than default rpart, but on test dataset it was getting much better results in AUPRC and sligthy better AUC.