Sick dataset analysis part 2

Wojciech Bogucki

27/04/2020

Contents

```
sick_train_mice <- mice(sick_train, printFlag = FALSE)
```

```
## Warning: Number of logged events: 125
```

```
sick_train_imp <- complete(sick_train_mice)
n <- nrow(sick_test)
sick_all <- rbind(sick_test, sick_train_imp)
sick_all_mice <- mice(sick_test[,-27], printFlag = FALSE)
```

```
## Warning: Number of logged events: 126
```

```
sick_all_imp <- complete(sick_all_mice)
sick_test_imp <- cbind(sick_all_imp[1:n,], Class=sick_test$Class)
```

```
# decision trees with missing values
task_rpart_mis<- makeClassifTask("task_rpart", data=sick_train, target = "Class")
learner_rpart_mis <- makeLearner("classif.rpart", predict.type = 'prob')
cv_rpart_mis <- crossval(learner_rpart_mis, task_rpart_mis,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation

## Measures:             auc

## [Resample] iter 1:    0.9578379

## [Resample] iter 2:    0.9485165

## [Resample] iter 3:    0.9337785

## [Resample] iter 4:    0.9335828

## [Resample] iter 5:    0.9277931

##

## Aggregated Result: auc.test.mean=0.9403017

##
```

```
model_rpart_mis <- train(learner_rpart_mis, task_rpart_mis)
pred_rpart_mis <- predict(model_rpart_mis, newdata = sick_test)

# decision trees with missing values with tune
learner_rpart_mis_tune <-  setHyperPars(learner_rpart_mis, minsplit=21, minbucket=7, cp=0.000367)
cv_rpart_mis_tune <- crossval(learner_rpart_mis_tune, task_rpart_mis,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation

## Measures:             auc

## [Resample] iter 1:    0.9583088

## [Resample] iter 2:    0.9734318

## [Resample] iter 3:    0.9577077

## [Resample] iter 4:    0.9474048

## [Resample] iter 5:    0.9690434

##

## Aggregated Result: auc.test.mean=0.9611793

##
```

```
model_rpart_mis_tune <- train(learner_rpart_mis_tune, task_rpart_mis)
pred_rpart_mis_tune <- predict(model_rpart_mis_tune, newdata = sick_test)
```

```
# ranger
task_ranger<- makeClassifTask("task_ranger", data=sick_train_imp, target = "Class")
learner_ranger <- makeLearner("classif.ranger", predict.type = 'prob')
cv_ranger <- crossval(learner_ranger, task_ranger,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation

## Measures:             auc

## [Resample] iter 1:    0.9870841

## [Resample] iter 2:    0.9948937

## [Resample] iter 3:    0.9972565

## [Resample] iter 4:    0.9967127

## [Resample] iter 5:    0.9967193
```

```
##
```

```
## Aggregated Result: auc.test.mean=0.9945332
```

```
##
```

```
set.seed(10, "L'Ecuyer")
model_ranger <- train(learner_ranger, task_ranger)
pred_ranger <- predict(model_ranger, newdata = sick_test_imp)

# ranger with tune
learner_ranger_tune <-  setHyperPars(learner_ranger,
                                     mtry=7,
                                     min.node.size=3,
                                     splitrule='gini',
                                     replace=FALSE)
cv_ranger_tune <- crossval(learner_ranger_tune, task_ranger,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation
```

```
## Measures:             auc
```

```
## [Resample] iter 1:    0.9977159
```

```
## [Resample] iter 2:    0.9922345
```

```
## [Resample] iter 3:    0.9994121
```

```
## [Resample] iter 4:    0.9978138
```

```
## [Resample] iter 5:    0.9888545
```

```
##
```

```
## Aggregated Result: auc.test.mean=0.9952062
```

```
##
```

```
set.seed(10, "L'Ecuyer")
model_ranger_tune <- train(learner_ranger_tune, task_ranger)
pred_ranger_tune <- predict(model_ranger_tune, newdata = sick_test_imp)
```

```
task_gbm <- makeClassifTask("task_gbm", data=sick_train, target = "Class")
learner_gbm <- makeLearner("classif.gbm", predict.type = 'prob')
cv_gbm <- crossval(learner_gbm, task_gbm,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation
```

```
## Measures:             auc
```

```
## Distribution not specified, assuming bernoulli ...

## [Resample] iter 1:    0.9928156

## Distribution not specified, assuming bernoulli ...

## [Resample] iter 2:    0.9526956

## Distribution not specified, assuming bernoulli ...

## [Resample] iter 3:    0.9874339

## Distribution not specified, assuming bernoulli ...

## [Resample] iter 4:    0.9936986

## Distribution not specified, assuming bernoulli ...

## [Resample] iter 5:    0.8341847

##

## Aggregated Result: auc.test.mean=0.9521657

##
```

```
set.seed(10, "L'Ecuyer")
model_gbm <- train(learner_gbm, task_gbm)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
pred_gbm <- predict(model_gbm, newdata = sick_test)


learner_gbm_tune <-  setHyperPars(learner_gbm, n.trees=169,
                             interaction.depth=3,
                             n.minobsinnode=4,
                             distribution='gaussian',
                             shrinkage=0.0932)
cv_gbm_tune <- crossval(learner_gbm_tune, task_gbm,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation

## Measures:             auc

## [Resample] iter 1:    0.9973183

## [Resample] iter 2:    0.9977256
```

```
## [Resample] iter 3:    0.9495567

## [Resample] iter 4:    0.9987585

## [Resample] iter 5:    0.9984376

##

## Aggregated Result: auc.test.mean=0.9883594

##
```

```
set.seed(10, "L'Ecuyer")
model_gbm_tune <- train(learner_gbm_tune, task_gbm)
pred_gbm_tune <- predict(model_gbm_tune, newdata = sick_test)
```

```
indx <- sapply(sick_train[,-27], is.factor)
sick_train_num <- sick_train
sick_train_num[indx] <- lapply(sick_train[indx], function(x) as.numeric(x)-1)
sick_test_num <- sick_test
sick_test_num[indx] <- lapply(sick_test[indx], function(x) as.numeric(x)-1)

# xgboost
task_xgb <- makeClassifTask("task_xgb", data=sick_train_num, target = "Class")
learner_xgb <- makeLearner("classif.xgboost", predict.type = 'prob')
cv_xgb <- crossval(learner_xgb, task_xgb,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation

## Measures:             auc

## [Resample] iter 1:    0.9562125

## [Resample] iter 2:    0.9482268

## [Resample] iter 3:    0.9427019

## [Resample] iter 4:    0.9970768

## [Resample] iter 5:    0.9837176

##

## Aggregated Result: auc.test.mean=0.9655871

##
```

```r
model_xgb <- train(learner_xgb, task_xgb)
pred_xgb <- predict(model_xgb, newdata = sick_test_num)

# xgboost with tune
learner_xgb_tune <-  setHyperPars(learner_xgb,
                            min_child_weight=4.97,
                            max_depth=4,
                            gamma=3.86,
                            eta=0.374)
cv_xgb_tune <- crossval(learner_xgb_tune, task_xgb,iters = 5,measures = list(auc))
```

```
## Resampling: cross-validation

## Measures:             auc

## [Resample] iter 1:    0.9785175

## [Resample] iter 2:    0.9600133

## [Resample] iter 3:    0.9656371

## [Resample] iter 4:    0.9441827

## [Resample] iter 5:    0.9199600

##

## Aggregated Result: auc.test.mean=0.9536621

##
```

```r
model_xgb_tune <- train(learner_xgb_tune, task_xgb)
pred_xgb_tune <- predict(model_xgb_tune, newdata = sick_test_num)
```

```r
preds <- list(pred_rpart_mis,pred_rpart_mis_tune,pred_ranger,pred_ranger_tune,pred_gbm,pred_gbm_tune,pr
mods <- c( "Decision trees","Decision trees with tune", "Ranger","Ranger with tune","Gradient Boosting
n_mods <- length(mods)
perf_auc <- list()
perf_auprc <- list()
perf_rocr <- list()
for (i in 1:n_mods){
  perf_auc[i] <- performance(preds[[i]],list(auc))
  perf_auprc[i] <- auprc(preds[[i]]$data$prob.sick, sick_test_imp$Class, "sick")
  pred2 <- ROCR::prediction(as.vector(preds[[i]]$data$prob.sick), as.vector(preds[[i]]$data$truth))
  perf_rocr[i] <- ROCR::performance(pred2,"tpr","fpr")
}
```

```
## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated

## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated

## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated

## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated

## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated

## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated

## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated

## Warning in `[<-`(`*tmp*`, i, value = ROCR::performance(pred2, "tpr", "fpr")):
## implicit list embedding of S4 objects is deprecated
```

```
kable(data.frame(model=mods,'auc 5-crossvalidation'=c(cv_rpart_mis$aggr,cv_rpart_mis_tune$aggr, cv_range
  kable_styling(latex_options = "hold_position")
```

Table 1: Measures of goodness of prediction for each model

model	auc.5.crossvalidation	auc	auprc
Decision trees	0.9403017	0.8966330	0.7701941
Decision trees with tune	0.9611793	0.9733968	0.8927406
Ranger	0.9945332	0.9928578	0.8944267
Ranger with tune	0.9952062	0.9941182	0.9090941
Gradient Boosting Machine	0.9521657	0.9644840	0.7456489
Gradient Boosting Machine with tune	0.9883594	0.9957687	0.9223405
Xgboost	0.9655871	0.9954386	0.9105864
Xgboost with tune	0.9536621	0.9924977	0.8675983