Introduction to Machine Learning

Code demo for Kaggle Challenge

Working Group “Computational Statistics” – Bernd Bischl et al.

Code demo for Kaggle Challenge

In this code demo we

- use CART to compete in a kaggle challenge,
- learn how to make a submission for the challenge,
- improve the model by using feature engineering.

Introductory kaggle challenge

We will compete in our first kaggle challenge on the prediction of titanic survivors.

Preprocessing and Data check

```r
### Data preprocess

# load and check the data
all_train <- read.csv(file = "data/train_titanic.csv")
str(all_train)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: ..
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2..
##  $ Age        : num  22 38 26 35 35 NA 54 2 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 ..
##  $ Fare       : num  7.25 71.28 7.92 53.1 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 ..
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 ..
```

```r
# no target column "survived" on test dataset
all_test <- read.csv(file = "data/test_titanic.csv")

# transform target to factor variable for mlr3
all_train$Survived <- as.factor(all_train$Survived)

# can we use all features?
# Nope: delete those with too many levels as this would inflate the model
# also kill the ID

train <- all_train[, -c(
  which(colnames(all_train) == "Cabin"),
  which(colnames(all_train) == "Name"),
  which(colnames(all_train) == "Ticket"),
  which(colnames(all_train) == "PassengerId")
)]

test <- all_test[, -c(
  which(colnames(all_test) == "Cabin"),
  which(colnames(all_test) == "Name"),
  which(colnames(all_test) == "Ticket"),
  which(colnames(all_test) == "PassengerId")
)]
```

Build a first simple model with `mlr3` and check the performance via CV

```r
### model corner
library(mlr3)
library(mlr3learners)
library(mlr3tuning)
library(mlr3filters)
library(paradox)
library(mlr3tuning)
requireNamespace("lgr")

# show only warning messages
logger = lgr::get_logger("mlr3")
logger$set_threshold("warn")
logger = lgr::get_logger("bbotk")
logger$set_threshold("warn")

# choose specific model and parameters
task <- TaskClassif$new(
  id = "titanic_train", backend = train,
  target = "Survived"
)

# check choosable parameters and set accordingly
lrn("classif.rpart")$param_set
```

```
## <ParamSet>
##                 id    class lower upper      levels
##  1:       minsplit ParamInt     1   Inf
##  2:      minbucket ParamInt     1   Inf
##  3:             cp ParamDbl     0     1
##  4:     maxcompete ParamInt     0   Inf
##  5:   maxsurrogate ParamInt     0   Inf
##  6:       maxdepth ParamInt     1    30
##  7:   usesurrogate ParamInt     0     2
##  8: surrogatestyle ParamInt     0     1
##  9:           xval ParamInt     0   Inf
## 10:     keep_model ParamLgl    NA    NA  TRUE,FALSE
##            default value
##  1:             20
##  2: <NoDefault[3]>
##  3:           0.01
##  4:              4
##  5:              5
##  6:             30
##  7:              2
##  8:              0
##  9:             10     0
## 10:          FALSE
```

```r
# check available settings here:
# https://www.rdocumentation.org/packages/rpart/versions/4.1-12/topics/rpart.control
learner <- lrn(
  "classif.rpart",
  predict_type = "prob",
  minsplit = 10,
  cp = 0.05
)

# train the model
learner$train(task)

### performance estimate via CV
resampling <- rsmp("cv", folds = 10)
cv <- resample(learner = learner, task = task, resampling = resampling)

# use mlr3::mlr_measures to get list of possible measures
# important: always check on which measure they evaluate you!
cv$aggregate(measures = msrs(c("classif.ce", "classif.acc")))
```

```
##  classif.ce classif.acc
##       0.213       0.787
```

Store and submit your predictions

```r
# predict for submission
pred <- learner$predict_newdata(newdata = test)
submission <- as.data.frame(pred$response)

submission$PassengerId <- all_test$PassengerId

colnames(submission) <- c("Survived", "PassengerId")

write.csv(submission, file = "data/submissionTitanic_1.csv", row.names = FALSE)
```

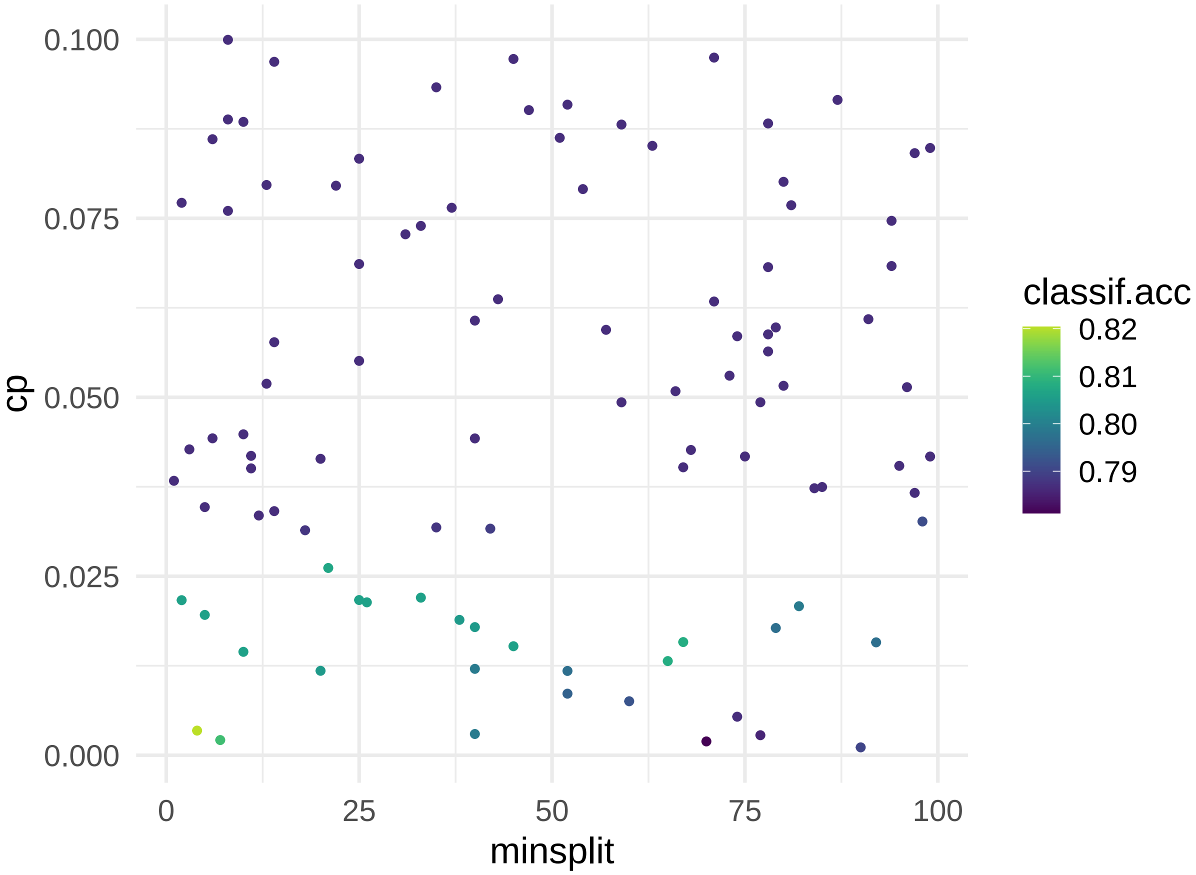
Tune the Hyperparameters of the algorithm

```r
### Tune the model
# we chose two numeric parameters above and now search for optimal values
# check available parameters
set.seed(1337)
learner <- lrn("classif.rpart", predict_type = "prob")
resampling <- rsmp("cv", folds = 10)
measure <- msr("classif.acc")
# make parameter set
tune_ps <- ParamSet$new(list(
  ParamDbl$new("cp", lower = 0.001, upper = 0.1),
  ParamInt$new("minsplit", lower = 1, upper = 100)
))
terminator <- trm("evals", n_evals = 100)

# choose random search - why not grid search?
tuner <- tnr("random_search")
```

```
at <- AutoTuner$new(
  learner = learner,
  resampling = resampling,
  measure = measure,
  search_space = tune_ps,
  terminator = terminator,
  tuner = tuner
)
at$train(task)
```

Visualize the random search over both parameters:

```
library(ggplot2)
vis_hyper <- at$tuning_instance$archive$data()[
  ,
  c(
    "cp",
    "minsplit",
    "classif.acc"
  )
]
ggplot(vis_hyper, aes(x = minsplit, y = cp, color = classif.acc)) +
  geom_point()
```



```
# tuning result
at$tuning_result
```

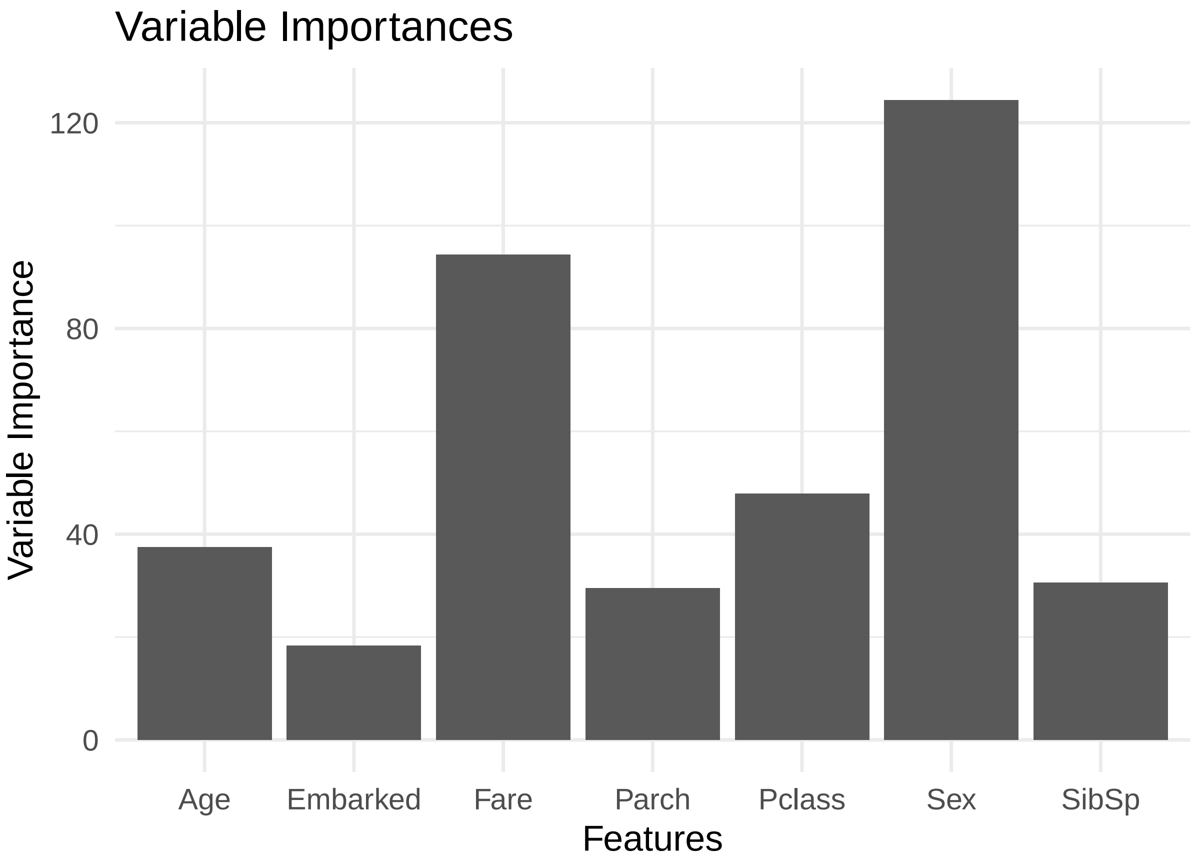
```
##         cp minsplit learner_param_vals  x_domain classif.acc
## 1: 0.00344        4          <list[3]> <list[2]>        0.82
```

Store and submit those results to kaggle

```
# use those param settings for the CART
learner <- lrn(
  "classif.rpart",
  predict_type = "prob"
) # inspect the learner
# learner
learner$param_set$values <-  as.list(at$tuning_result[, c("cp", "minsplit")])
learner$train(task)

# predict for submission
pred <- learner$predict_newdata(newdata = test)
submission <- as.data.frame(pred$response)

submission$PassengerId <- all_test$PassengerId

colnames(submission) <- c("Survived", "PassengerId")

write.csv(submission, file = "data/submissionTitanic_2.csv", row.names = FALSE)
```

```

filter <- flt("importance", learner = learner)
filter$calculate(task)

var <- as.data.table(filter)
ggplot(data = var, aes(x = feature, y = score)) +
  geom_bar(stat = "identity") +
  ggtitle(label = "Variable Importances") +
  labs(x = "Features", y = "Variable Importance")
```

Check variable importances



Feature engineering

Can we further condense the information from the multi-level factors and use it for our model?

We take a closer look at the names of the guests.

```
### feature engineering
library(dplyr)

# indicator for train or test set
all_train$train <- 1
all_test$train <- 0
all_test$Survived <- NA

# compute once for all data and split again for training with ID
all_data <- rbind(all_train, all_test)
eng_data <- all_data

head(all_data$Name)
```

```
## [1] Braund, Mr. Owen Harris
## [2] Cumings, Mrs. John Bradley (Florence Briggs Thayer)
## [3] Heikkinen, Miss. Laina
## [4] Futrelle, Mrs. Jacques Heath (Lily May Peel)
## [5] Allen, Mr. William Henry
## [6] Moran, Mr. James
## 1307 Levels: Abbing, Mr. Anthony ... van Billiard, Master. Walter John
```

We can see, that there is information on the title of the people in their names. We use that information as a new feature!

```
# use regular expressions via strplit to extract the title of the people
# temporary storage
temp <- sapply(
  strsplit(as.character(all_data$Name), split = ","),
  function(x) x[2]
)
title <- strsplit(temp, split = " ")
eng_data$title <- sapply(title, function(x) x[2])
# unfortunately still too many titles with too few observations
table(eng_data$title)
```

```
##
##     Capt.      Col.      Don.      Dona.       Dr. Jonkheer.
##         1         4         1         1         8         1
##     Lady.    Major.   Master.     Miss.     Mlle.       Mme.
##         1         2        61       260         2         1
##       Mr.      Mrs.       Ms.      Rev.      Sir.        the
##       757       197         2         8         1         1
```

Btw.: we found the Captain:

```
# btw.: we found the captain:
all_data[which(eng_data$title == "Capt."), "Name"]
```

```
## [1] Crosby, Capt. Edward Gifford
## 1307 Levels: Abbing, Mr. Anthony ... van Billiard, Master. Walter John
```

condense those with obs < 5 to class “other”

```
freqs <- as.data.frame(table(eng_data$title))
other_titles <- freqs[which(freqs$Freq < 5), "Var1"]
eng_data[which(eng_data$title %in% other_titles), "title"] <- "other"
eng_data$title <- as.factor(eng_data$title)
# looks better now
table(eng_data$title)
```

```
##
##     Dr. Master.   Miss.     Mr.    Mrs.     Rev.   other
##       8      61     260     757     197        8      18
```

Build updated model

```
### model corner 2 with engineered feature
train <- eng_data %>%
  filter(train == 1) %>%
  select(-c(PassengerId, Name, Ticket, train, Cabin))

# transform target to factor variable for mlr3
train$Survived <- as.factor(all_train$Survived)

test <- eng_data %>%
  filter(train == 0) %>%
  select(-c(PassengerId, Name, Ticket, train, Cabin, Survived))
```

```r
# choose specific model and parameters
task <- TaskClassif$new(
  id = "titanic_train", backend = train,
  target = "Survived"
)

learner <- lrn("classif.rpart", predict_type = "prob")
resampling <- rsmp("cv", folds = 10)
measure <- msr(c("classif.acc"))
# make parameter set
tune_ps <- ParamSet$new(list(
  ParamDbl$new("cp", lower = 0.001, upper = 0.1),
  ParamInt$new("minsplit", lower = 1, upper = 100)
))
terminator <- trm("evals", n_evals = 100)

# choose random search - why not grid search?
tuner <- tnr("random_search")

at <- AutoTuner$new(
  learner = learner,
  resampling = resampling,
  measure = measure,
  search_space = tune_ps,
  terminator = terminator,
  tuner = tuner
)
at$train(task)
```

Check tuning result

```r
at$tuning_result
```

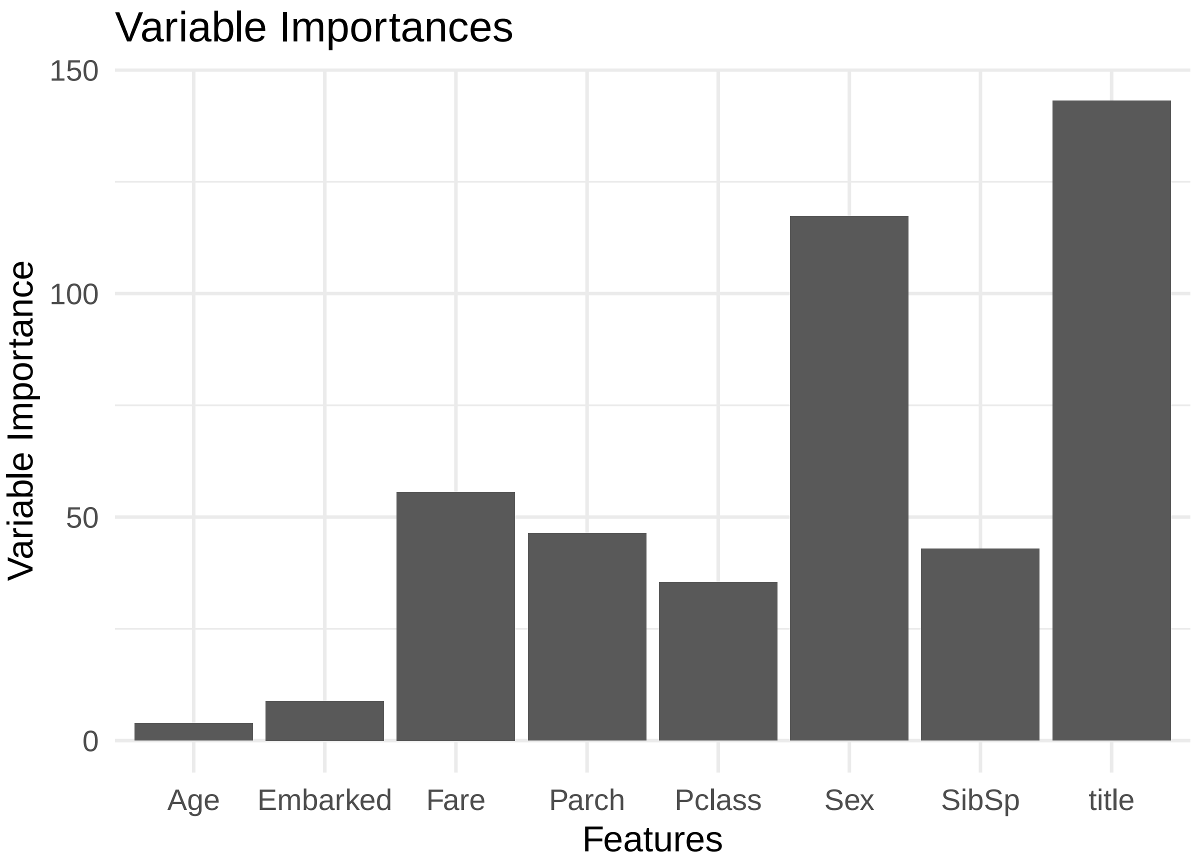
```
##        cp minsplit learner_param_vals  x_domain classif.acc
## 1: 0.0221        7          <list[3]> <list[2]>       0.825
```

Write and store the submission

```r
# use those param settings for the CART
learner <- lrn(
  "classif.rpart",
  predict_type = "prob"
) # inspect the learner
# learner
learner$param_set$values <- as.list(at$tuning_result[, c("cp", "minsplit")])
learner$train(task)

# predict for submission
pred <- learner$predict_newdata(newdata = test)
submission <- as.data.frame(pred$response)

submission$PassengerId <- all_test$PassengerId

colnames(submission) <- c("Survived", "PassengerId")

write.csv(submission, file = "data/submissionTitanic_3.csv", row.names = FALSE)
```

```r
filter <- flt("importance", learner = learner)
filter$calculate(task)

var <- as.data.table(filter)
ggplot(data = var, aes(x = feature, y = score)) +
  geom_bar(stat = "identity") +
  ggtitle(label = "Variable Importances") +
  labs(x = "Features", y = "Variable Importance")
```

Check Variable Importances



What can we see? How could we criticize that result? Is there a way to detect the problem?