

# Introduction to Machine Learning

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

# Code demo for resampling

## Self made Cross Validation

We want to assess the performance of our model, i.e., try to estimate its generalization error. Why is it a good idea to use cross-validation (CV)?

Let's write our very own CV function for a  $k$ -NN learner to experiment with...

```
library(mlr3)
library(mlr3learners)
library(mlbench)

set.seed(13)
spiral <- as.data.frame(mlbench.spirals(n = 500, sd = 0.1))

# Cross validation for kNN
# inputs:
#   data a data set to use
#   target name of the column in <data> to classify
#   folds number of CV folds
#   k neighborhood size for kNN
# returns vector of test set errors for the different folds
knn_cv <- function(data, target, folds, k) {
  cv_errors <- as.numeric(folds)

  indices <- c(
    sample(x = seq(1, nrow(data), by = 1), size = nrow(data), replace = FALSE),
    rep(NA, (folds - nrow(data) %% folds) %% folds)
  )

  # index matrix for folds
  index_mat <- matrix(data = indices, byrow = FALSE, nrow = folds)

  for (i in 1:folds) {
    # data
    test_data <- data[na.omit(index_mat[i, ]), ]
    train_data <- data[-na.omit(index_mat[i, ]), ]
    task <- TaskClassif$new(
      id = "spirals_train",
      backend = train_data, target = target
    )
    # model
    learner <- lrn("classif.kknn", k = k)
    # train on training set
    learner$train(task = task)
    # evaluate on test data
    cv_errors[i] <- learner$predict_newdata(test_data)$score()
  }

  cv_errors
}

result <- knn_cv(data = spiral, target = "classes", folds = 11, k = 4)
```

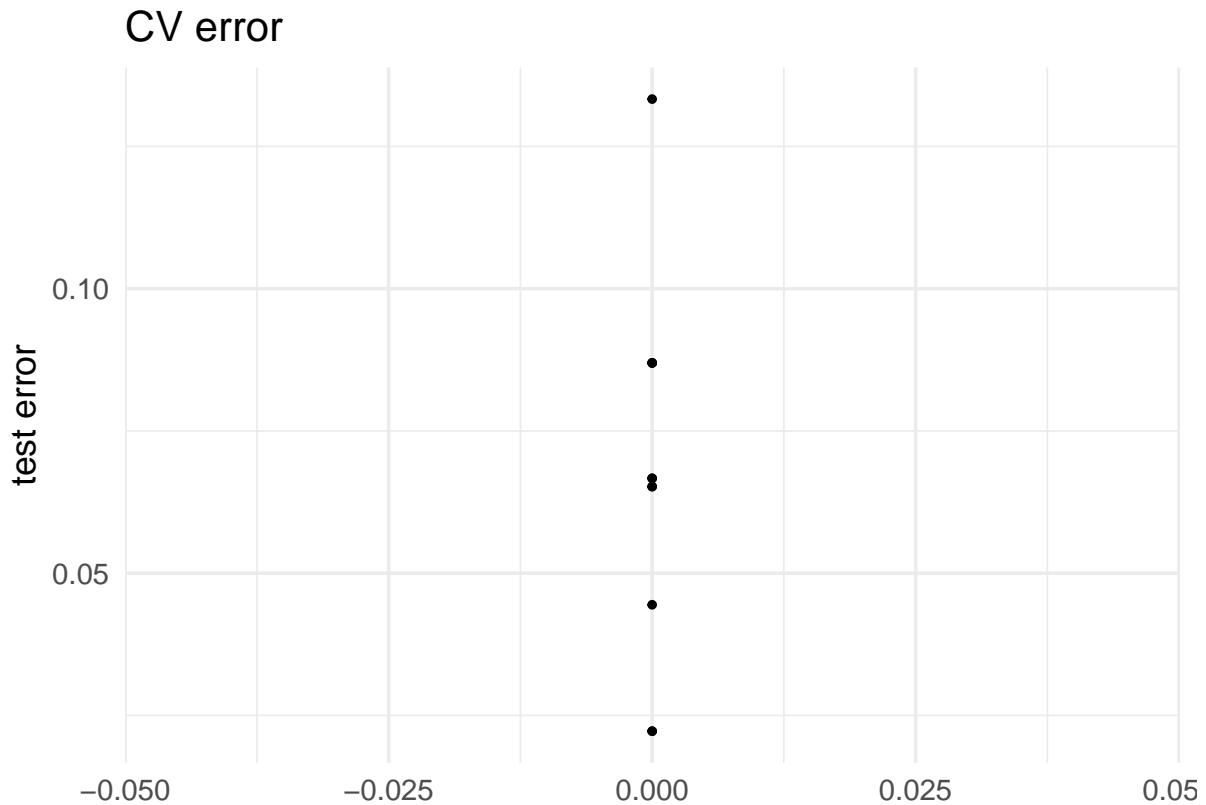
```
result
```

```
## [1] 0.0652 0.0870 0.0652 0.0870 0.0870 0.0667 0.1333 0.0444  
## [9] 0.0667 0.0222 0.0222
```

```
mean(result)
```

```
## [1] 0.0679
```

```
p <- ggplot(data = as.data.frame(result), aes(y = result)) +  
  geom_point(x = 0) +  
  ggtitle(label = "CV error") +  
  xlab("") + ylab("test error") + xlim(c(0, 0))  
p
```



So what happens if we increase the number of folds?

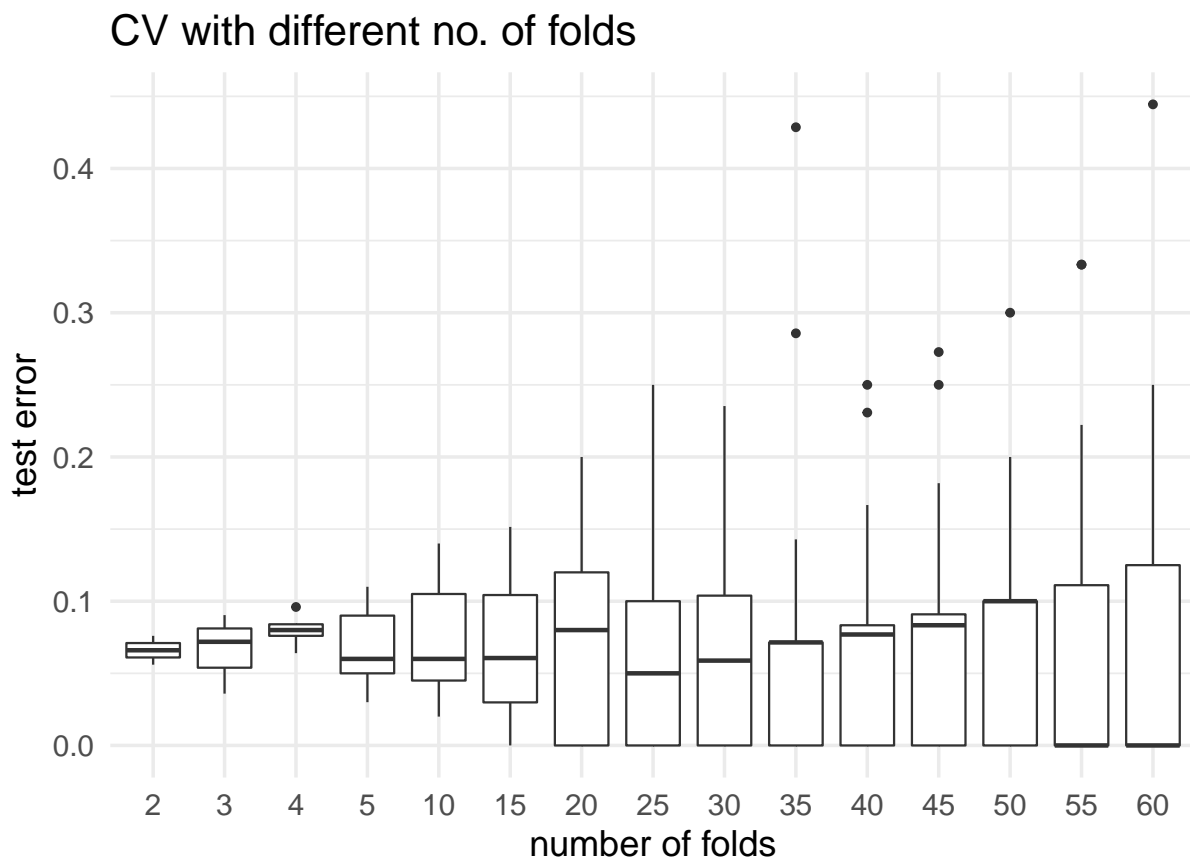
```
# run CV with 2, 3, 4, 5, 10, 15, ..., 60 folds and record the test set errors.  
cv_results <- lapply(  
  X = c(2, 3, 4, 5 * (1:12)),  
  FUN = function(folds) {  
    data.frame(  
      folds = as.character(folds),  
      cv_errors = knn_cv(  
        data = spiral, target = "classes",
```

```

    folds = folds, k = 4
  )
}
)
cv_data <- do.call(rbind, cv_results)

cv_plot <-
  ggplot(cv_data, aes(x = folds, y = cv_errors)) + geom_boxplot() +
  ggtitle(label = "CV with different no. of folds") +
  xlab("number of folds") + ylab("test error")
cv_plot

```



The more we increase the number of folds, the larger each training set becomes. Hence the *pessimistic bias* for the estimated model performance becomes smaller.

But since the test sets also become smaller, the *variance* of the resulting performance estimate increases. In addition, with a higher number of folds, the computation time increases. (Think about that: by how much? is the increase linear in the number of folds? why or why not?)

Can we get better results with a smaller amount of computation? Let's see what happens if we do repeated CV and collect only their means:

```

# do reps = 10 repetitions each of 5, 10, 15, 20-fold CV
rep_cv_results <- lapply(
  X = c(5, 10, 15, 20),

```

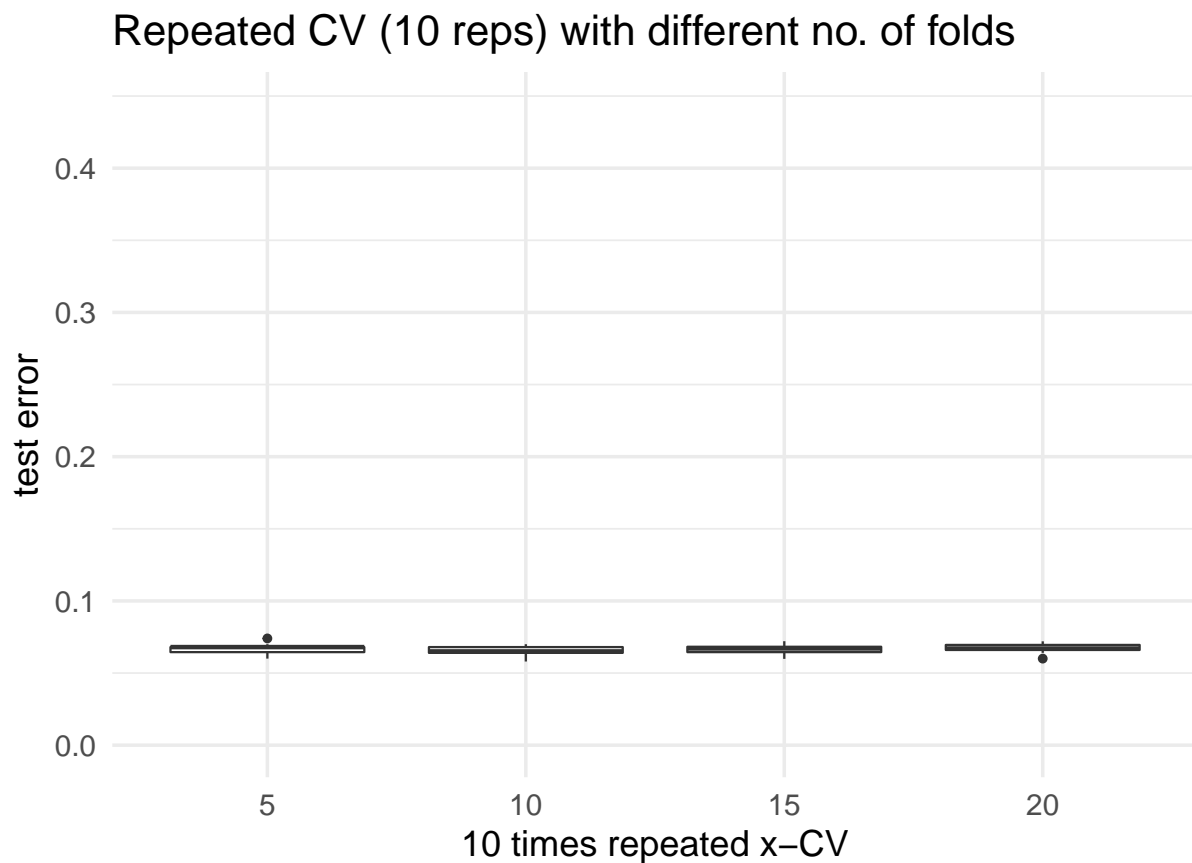
```

FUN = function(folds, reps = 10) {
  mean_cv_errors <- replicate(
    reps,
    mean(knn_cv(
      data = spiral, target = "classes",
      folds = folds, k = 4
    ))
  )
  data.frame(
    folds = as.character(folds),
    mean_cv_errors = mean_cv_errors
  )
}

rep_cv_data <- do.call(rbind, rep_cv_results)

ggplot(rep_cv_data, aes(x = folds, y = mean_cv_errors)) +
  geom_boxplot() +
  xlab("10 times repeated x-CV") + ylab("test error") +
  ggtitle(label = "Repeated CV (10 reps) with different no. of folds") +
  ylim(range(cv_data$cv_errors))

```



We see that our estimation results stabilize.

## mlr3's CV implementation

```
set.seed(1337)
task <- TaskClassif$new(
  id = "spirals_task",
  backend = spiral, target = "classes"
)
rdesc_cv <- rsmp("repeated_cv", folds = 10, repeats = 10)

mlr_cv <- resample(
  resampling = rdesc_cv, learner = lrn("classif.kknn", k = 4),
  task = task
)
```

```
## INFO [10:24:51.024] Applying learner 'classif.kknn' on task 'spirals_task' (iter 1/100)
## INFO [10:24:51.095] Applying learner 'classif.kknn' on task 'spirals_task' (iter 2/100)
## INFO [10:24:51.148] Applying learner 'classif.kknn' on task 'spirals_task' (iter 3/100)
## INFO [10:24:51.173] Applying learner 'classif.kknn' on task 'spirals_task' (iter 4/100)
## INFO [10:24:51.199] Applying learner 'classif.kknn' on task 'spirals_task' (iter 5/100)
## INFO [10:24:51.224] Applying learner 'classif.kknn' on task 'spirals_task' (iter 6/100)
## INFO [10:24:51.250] Applying learner 'classif.kknn' on task 'spirals_task' (iter 7/100)
## INFO [10:24:51.275] Applying learner 'classif.kknn' on task 'spirals_task' (iter 8/100)
## INFO [10:24:51.301] Applying learner 'classif.kknn' on task 'spirals_task' (iter 9/100)
## INFO [10:24:51.326] Applying learner 'classif.kknn' on task 'spirals_task' (iter 10/100)
## INFO [10:24:51.366] Applying learner 'classif.kknn' on task 'spirals_task' (iter 11/100)
## INFO [10:24:51.393] Applying learner 'classif.kknn' on task 'spirals_task' (iter 12/100)
## INFO [10:24:51.417] Applying learner 'classif.kknn' on task 'spirals_task' (iter 13/100)
## INFO [10:24:51.440] Applying learner 'classif.kknn' on task 'spirals_task' (iter 14/100)
## INFO [10:24:51.462] Applying learner 'classif.kknn' on task 'spirals_task' (iter 15/100)
## INFO [10:24:51.485] Applying learner 'classif.kknn' on task 'spirals_task' (iter 16/100)
## INFO [10:24:51.507] Applying learner 'classif.kknn' on task 'spirals_task' (iter 17/100)
## INFO [10:24:51.529] Applying learner 'classif.kknn' on task 'spirals_task' (iter 18/100)
## INFO [10:24:51.552] Applying learner 'classif.kknn' on task 'spirals_task' (iter 19/100)
## INFO [10:24:51.575] Applying learner 'classif.kknn' on task 'spirals_task' (iter 20/100)
## INFO [10:24:51.597] Applying learner 'classif.kknn' on task 'spirals_task' (iter 21/100)
## INFO [10:24:51.620] Applying learner 'classif.kknn' on task 'spirals_task' (iter 22/100)
## INFO [10:24:51.643] Applying learner 'classif.kknn' on task 'spirals_task' (iter 23/100)
## INFO [10:24:51.666] Applying learner 'classif.kknn' on task 'spirals_task' (iter 24/100)
## INFO [10:24:51.688] Applying learner 'classif.kknn' on task 'spirals_task' (iter 25/100)
## INFO [10:24:51.711] Applying learner 'classif.kknn' on task 'spirals_task' (iter 26/100)
## INFO [10:24:51.734] Applying learner 'classif.kknn' on task 'spirals_task' (iter 27/100)
## INFO [10:24:51.757] Applying learner 'classif.kknn' on task 'spirals_task' (iter 28/100)
## INFO [10:24:51.780] Applying learner 'classif.kknn' on task 'spirals_task' (iter 29/100)
## INFO [10:24:51.802] Applying learner 'classif.kknn' on task 'spirals_task' (iter 30/100)
## INFO [10:24:51.824] Applying learner 'classif.kknn' on task 'spirals_task' (iter 31/100)
## INFO [10:24:51.859] Applying learner 'classif.kknn' on task 'spirals_task' (iter 32/100)
## INFO [10:24:51.890] Applying learner 'classif.kknn' on task 'spirals_task' (iter 33/100)
## INFO [10:24:51.918] Applying learner 'classif.kknn' on task 'spirals_task' (iter 34/100)
## INFO [10:24:51.945] Applying learner 'classif.kknn' on task 'spirals_task' (iter 35/100)
## INFO [10:24:51.972] Applying learner 'classif.kknn' on task 'spirals_task' (iter 36/100)
## INFO [10:24:51.998] Applying learner 'classif.kknn' on task 'spirals_task' (iter 37/100)
## INFO [10:24:52.020] Applying learner 'classif.kknn' on task 'spirals_task' (iter 38/100)
## INFO [10:24:52.043] Applying learner 'classif.kknn' on task 'spirals_task' (iter 39/100)
```

[illegible]

```
## INFO [10:24:53.395] Applying learner 'classif.kknn' on task 'spirals_task' (iter 94/100)
## INFO [10:24:53.427] Applying learner 'classif.kknn' on task 'spirals_task' (iter 95/100)
## INFO [10:24:53.455] Applying learner 'classif.kknn' on task 'spirals_task' (iter 96/100)
## INFO [10:24:53.478] Applying learner 'classif.kknn' on task 'spirals_task' (iter 97/100)
## INFO [10:24:53.502] Applying learner 'classif.kknn' on task 'spirals_task' (iter 98/100)
## INFO [10:24:53.525] Applying learner 'classif.kknn' on task 'spirals_task' (iter 99/100)
## INFO [10:24:53.548] Applying learner 'classif.kknn' on task 'spirals_task' (iter 100/100)
```

```
mlr_cv$score()[, c("iteration", "classif.ce")]
```

```
##      iteration classif.ce
## 1:           1      0.10
## 2:           2      0.10
## 3:           3      0.02
## 4:           4      0.08
## 5:           5      0.08
## 6:           6      0.06
## 7:           7      0.04
## 8:           8      0.02
## 9:           9      0.10
## 10:          10      0.10
## 11:          11      0.14
## 12:          12      0.04
## 13:          13      0.10
## 14:          14      0.04
## 15:          15      0.08
## 16:          16      0.04
## 17:          17      0.04
## 18:          18      0.08
## 19:          19      0.04
## 20:          20      0.08
## 21:          21      0.10
## 22:          22      0.04
## 23:          23      0.02
## 24:          24      0.10
## 25:          25      0.06
## 26:          26      0.06
## 27:          27      0.02
## 28:          28      0.10
## 29:          29      0.12
## 30:          30      0.10
## 31:          31      0.10
## 32:          32      0.02
## 33:          33      0.08
## 34:          34      0.10
## 35:          35      0.08
## 36:          36      0.06
## 37:          37      0.06
## 38:          38      0.08
## 39:          39      0.06
## 40:          40      0.06
## 41:          41      0.08
## 42:          42      0.04
## 43:          43      0.08
```



##	44:	44	0.10
##	45:	45	0.04
##	46:	46	0.04
##	47:	47	0.04
##	48:	48	0.12
##	49:	49	0.08
##	50:	50	0.04
##	51:	51	0.06
##	52:	52	0.08
##	53:	53	0.04
##	54:	54	0.08
##	55:	55	0.08
##	56:	56	0.04
##	57:	57	0.12
##	58:	58	0.00
##	59:	59	0.08
##	60:	60	0.06
##	61:	61	0.08
##	62:	62	0.06
##	63:	63	0.04
##	64:	64	0.08
##	65:	65	0.04
##	66:	66	0.02
##	67:	67	0.06
##	68:	68	0.06
##	69:	69	0.12
##	70:	70	0.08
##	71:	71	0.08
##	72:	72	0.04
##	73:	73	0.02
##	74:	74	0.08
##	75:	75	0.14
##	76:	76	0.06
##	77:	77	0.04
##	78:	78	0.02
##	79:	79	0.14
##	80:	80	0.06
##	81:	81	0.12
##	82:	82	0.02
##	83:	83	0.02
##	84:	84	0.04
##	85:	85	0.08
##	86:	86	0.04
##	87:	87	0.06
##	88:	88	0.14
##	89:	89	0.10
##	90:	90	0.06
##	91:	91	0.12
##	92:	92	0.12
##	93:	93	0.04
##	94:	94	0.08
##	95:	95	0.08
##	96:	96	0.06
##	97:	97	0.02

```
## 98:      98      0.06
## 99:      99      0.02
## 100:     100      0.08
##      iteration classif.ce
```

```
mlr_cv$aggregate()
```

```
## classif.ce
##      0.0678
```

```
library(ggplot2)
```

```
ggplot(data = mlr_cv$score()[, "classif.ce"], aes(y = classif.ce, x = 1)) +
  geom_boxplot() +
  ggtitle(label = "Repeated CV (10-10) with mlr") +
  xlab("") + ylab("test error") + xlim(c(0, 2))
```

