Exercise:
Ensemble Methods

Machine Learning and Prediction Modelling

SOLUTION

Exercise 1: Diabetes in Pima Indian women (again)

In the last exercise we fitted a decision tree to the `Pima.tr` data available in the `MASS` package with the goal of predicting the diabetes status (variable `type`: Yes/No). We evaluated the predictive performance of the model using the corresponding `Pima.te` data as a test set. Here are again the code and results from the last exercise:

```
library(MASS)
suppressMessages(library(partykit))   # Dont show messages when loading package
set.seed(3487)
tr <- ctree(type ~ ., data = Pima.tr)
pred_tre <- predict(tr, newdata=Pima.te, type='response')
# Confusion matrix:
confT_tre <- table(pred_tre, Pima.te$type)
confT_tre
```

```
##
## pred_tre  No Yes
##      No  184  51
##      Yes  39  58
```

```
# Test error:
missMat <- confT_tre
diag(missMat) <- 0
test_err_tre <- sum(missMat)/sum(confT_tre)
test_err_tre
```

```
## [1] 0.2710843
```

As we can see, the decision tree achieved a misclassification rate of 27.1% on the test data. In this exercise we want to see how a random forest performs on the same task.

a) Fit a random forest consisting of 500 trees to the `Pima.tr` data using the `partykit` package. The target variable is again the `type` variable and all other variables are used as predictors. We will leave the `mtry` parameter at its default value. Try to find out with the help page what the default value of `mtry` is in the `cforest` function. (**Hint**: `?cforest`)

```
set.seed(9428)
rf_type <- cforest(type ~ ., data = Pima.tr, ntree=500)
```

Using the help page, we can see that mtry is set to the default value mtry=ceiling(sqrt(nvar)). The associated

description in the Details section tells us that nvar is the number of predictor variables. In our case there are 7 predictor variables. Therefore, mtry was automatically set to 3.

b) Evaluate the created random forest on the `Pima.te` data by calculating the associated confusion matrix and misclassification error. How does it perform compared to the decision tree?

```
pred_rf <- predict(rf_type, newdata=Pima.te, type='response')
# Confusion matrix:
confT_rf <- table(Pima.te$type, pred_rf)
# Test error:
missMat <- confT_rf
diag(missMat) <- 0
test_err_rf <- sum(missMat)/sum(confT_rf)
### Compare with decision tree:
list("Tree_mat"=confT_tre, "Forest_mat"=confT_rf)   # Confusion matrices
```

```
## $Tree_mat
##
## pred_tre  No Yes
##      No  184  51
##      Yes  39  58
##
## $Forest_mat
##      pred_rf
##        No Yes
##   No  198  25
##   Yes  45  64
```

```
### Misclassification rates
c("Tree_miscl"=test_err_tre, "Forest_miscl"=test_err_rf)
```

```
##   Tree_miscl Forest_miscl
##    0.2710843    0.2108434
```

The random forest seems to perform quite a bit better than a single decision tree.

c) Random forests come with their own evaluation tool (OOB error). Instead of splitting the data as above, fit a random forest to the complete data (`Pima.tr` and `Pima.te` combined) and calculate the OOB error to get an estimate of the forest's predictive performance. (**Hint**: `rbind()`)

```
### Combine data:
Pima.comb <- rbind(Pima.tr, Pima.te)
rf_type_comb <- cforest(type ~ ., data = Pima.comb, ntree=500)
pred_oob <- predict(rf_type_comb, OOB=TRUE)
# Confusion matrix:
confT_oob <- table(Pima.comb$type, pred_oob)
# Test error:
missMat <- confT_oob
diag(missMat) <- 0
test_err_oob <- sum(missMat)/sum(confT_oob)
test_err_oob
```

```
## [1] 0.212406
```

The OOB error returns the slightly different estimate of the random forest's performance of 21.2%.

Exercise 2: Corona stress data

The data set for this exercise is a sample from the COVIDiSTRESS Global Survey – Round II data (published on the platform of the Open Science Framework: https://osf.io/36tsd/). The COVIDiSTRESS study used a questionnaire to assess stress perception and many other variables during the pandemic. We have preprocessed the data for this workshop by creating sumscores of various subscales from the questionnaire and drawing a random subsample of size 2000 to keep computation times for the exercises manageable.

The variables included in the data set are:

- age
- education (multilevel factor)
- relationship_status (multilevel factor)
- stress (how stressed people felt during the pandemic)
- isolation (how isolated people felt during the pandemic)
- fearCorona (how afraid people are of catching corona -> themselves or close ones)
- support (how socially supported people felt during the pandemic)
- compliance (how compliant people are regarding state regulations surrounding the pandemic)
- vaccWill (how willing people are to get vaccinated, binary factor)
- vacAtt (general attitude towards vaccines, the higher the more positive)
- trustInstitut (how much trust people have in institutions)
- resilience (how resilient people are)
- uncertainty_susc (how susceptible to uncertainty people are, the higher the more stressed by uncertainty)
- information_acquis (how much people have informed themselves about the pandemic)
- mispercept (how misinformed people are about the pandemic)
- conspiracy (how much people tend towards conspiracy thinking)
- antiExpert (how much people distrust experts)

a) Read in the corona stress data by loading the `Covid_stress_data_subs.rda` file. The name of the contained data frame is `dat_cvid`. Try to get a first impression of the data.

```
load('Covid_stress_data_subs.rda')
head(dat_cvid)
```

```
##       age               education relationship_status stress isolation
## 513    30       University_degree              Dating     31        11
## 14146  19 University_withoutDegree              Single     18         3
## 5541   58             Upto12years             Married     23         4
## 12531  21       University_degree             Married     21         7
## 11662  19 University_withoutDegree              Dating     20         5
## 9203   37                     PhD             Married     20         2
##       fearCorona support compliance vaccWill vaccAtt trustInstitut resilience
## 513            8       9         56  willing      25            48         12
## 14146          5      21         43 hesitant      21            27         19
## 5541           8      18         54  willing      37            46         30
## 12531          7      12         44  willing      23             0         25
## 11662          4      10         30  willing      38            43         29
```

```
## 9203           4      16         51  willing      37            44         35
##       uncertainty_susc information_acquis mispercept conspiracy antiExpert
## 513                 17                35          9         18          4
## 14146               14                38          6         18         13
## 5541                13                44          3         11          4
## 12531               15                 6         10         24         14
## 11662               16                28          3         15          9
## 9203                17                46          3         13          4
```

```
str(dat_cvid)
```

```
## 'data.frame':    2000 obs. of  17 variables:
##  $ age                : int  30 19 58 21 19 37 19 20 35 22 ...
##  $ education          : Factor w/ 7 levels "None","Upto6years",..: 6 5 4 6 5 7 5 5 4 5 ...
##  $ relationship_status: Factor w/ 7 levels "Cohabitating",..: 2 6 3 3 2 3 6 5 1 6 ...
##  $ stress             : num  31 18 23 21 20 20 11 11 25 16 ...
##  $ isolation          : int  11 3 4 7 5 2 3 0 12 4 ...
##  $ fearCorona         : int  8 5 8 7 4 4 1 0 6 4 ...
##  $ support            : int  9 21 18 12 10 16 18 21 7 13 ...
##  $ compliance         : num  56 43 54 44 30 51 39 30 36 52 ...
##  $ vaccWill           : Factor w/ 2 levels "hesitant","willing": 2 1 2 2 2 2 2 1 1 2 ...
##  $ vaccAtt            : num  25 21 37 23 38 37 35 27 21 29 ...
##  $ trustInstitut      : int  48 27 46 0 43 44 57 27 30 22 ...
##  $ resilience         : num  12 19 30 25 29 35 32 34 20 25 ...
##  $ uncertainty_susc   : int  17 14 13 15 16 17 14 15 16 15 ...
##  $ information_acquis : int  35 38 44 6 28 46 39 21 18 29 ...
##  $ mispercept         : num  9 6 3 10 3 3 10 3 10 8 ...
##  $ conspiracy         : int  18 18 11 24 15 13 12 26 17 6 ...
##  $ antiExpert         : int  4 13 4 14 9 4 8 10 13 11 ...
##  - attr(*, "na.action")= 'omit' Named int [1:2447] 25 27 29 38 58 85 95 96 103 107 ...
##   ..- attr(*, "names")= chr [1:2447] "25" "27" "29" "38" ...
```

The data contains a range of numerical and categorical variables. A pairs plot (not shown) informs us about a couple of interesting dependencies in the data (e.g. relation between education and trust in institutions).

b) We want to predict the willingness of people to get vaccinated (`vaccWill`, hesitant/willing). Fit a random forest to the data using `vaccWill` as the target variable and all other variables as predictors. We want our random forest to include 500 trees and to sample 3 variables in each node of a tree to consider as potential splitting variables (this might take around 1 minute of computation time).

```
set.seed(4298)
rf_vac <- cforest(vaccWill ~ ., data = dat_cvid, ntree = 500, mtry = 3)
```

c) We once again want to evaluate the accuracy of our model. Generate a confusion matrix comparing the true values of `vaccWill` with the OOB predictions of our random forest. What can you see?

```
preds_rf <- predict(rf_vac, OOB = TRUE)
confT <- table(dat_cvid$vaccWill, preds_rf)
confT
```

```
##           preds_rf
##            hesitant willing
##   hesitant      377      97
##   willing        45    1481
```

```
### OOB misclassification rate:
confT_mis <- confT
diag(confT_mis) <- 0
errRF <- sum(confT_mis)/sum(confT)
errRF
```

```
## [1] 0.071
```

The willingness to get vaccinated seems to be predicted quite well. Apparently, the accuracy for the prediction of the label hesitant (1st row: 377 correct out of 474) is lower compared to the label willing(2nd row: 1481 correct out of 1526).

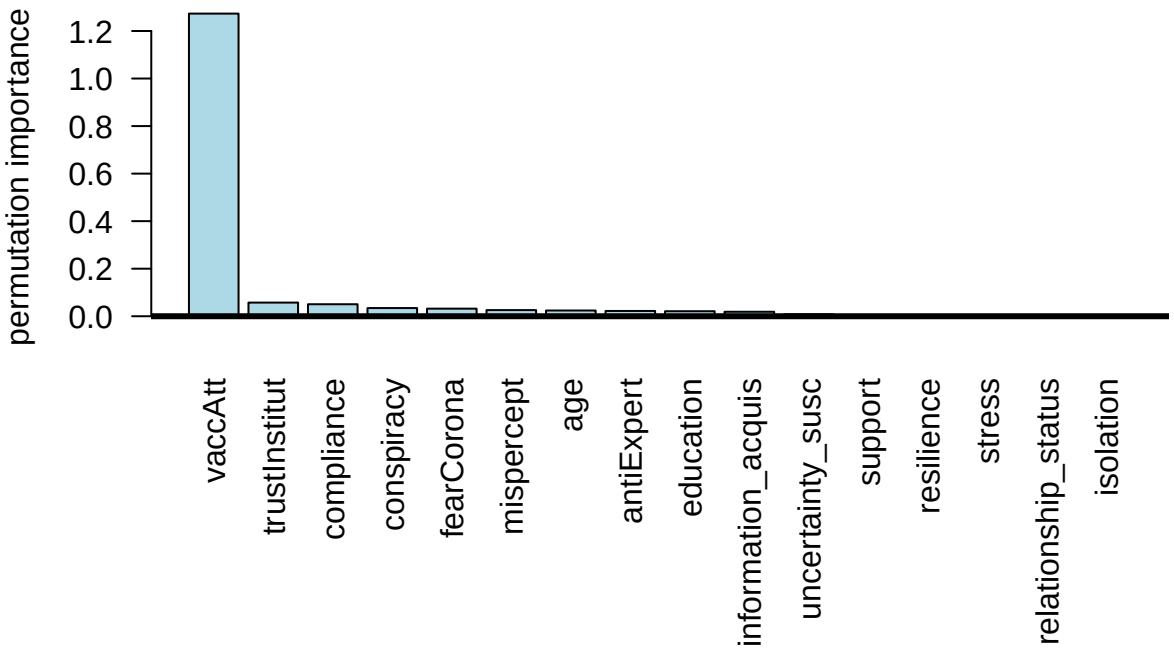
The OOB misclassification rate is 7.1%.

d) Calculate the permutation importance scores of all predictor variables and visualize the importance scores in a plot. Which predictor seems to be most important for the prediction of `vaccWill`?

```
imp_vac <- varimp(rf_vac)
imp_vac
```

```
##                 age           education relationship_status              stress
##        2.411510e-02        2.111419e-02       -6.342481e-04       -9.919002e-05
##           isolation          fearCorona             support          compliance
##       -1.752590e-03        3.192317e-02        3.485661e-03        5.050375e-02
##             vaccAtt       trustInstitut          resilience    uncertainty_susc
##        1.273238e+00        5.741439e-02        3.449886e-03        8.209642e-03
##  information_acquis          mispercept          conspiracy          antiExpert
##        1.887551e-02        2.584531e-02        3.443081e-02        2.225939e-02
```

```
### Plot importances:
par(mar=c(10, 4,4,2))
barplot(sort(imp_vac, decreasing = TRUE), col='lightblue',
        ylab = 'permutation importance', ylim = c(-0.1, max(imp_vac)), las=2)
abline(h = 0, col=1, lty=1, lwd=3)
```

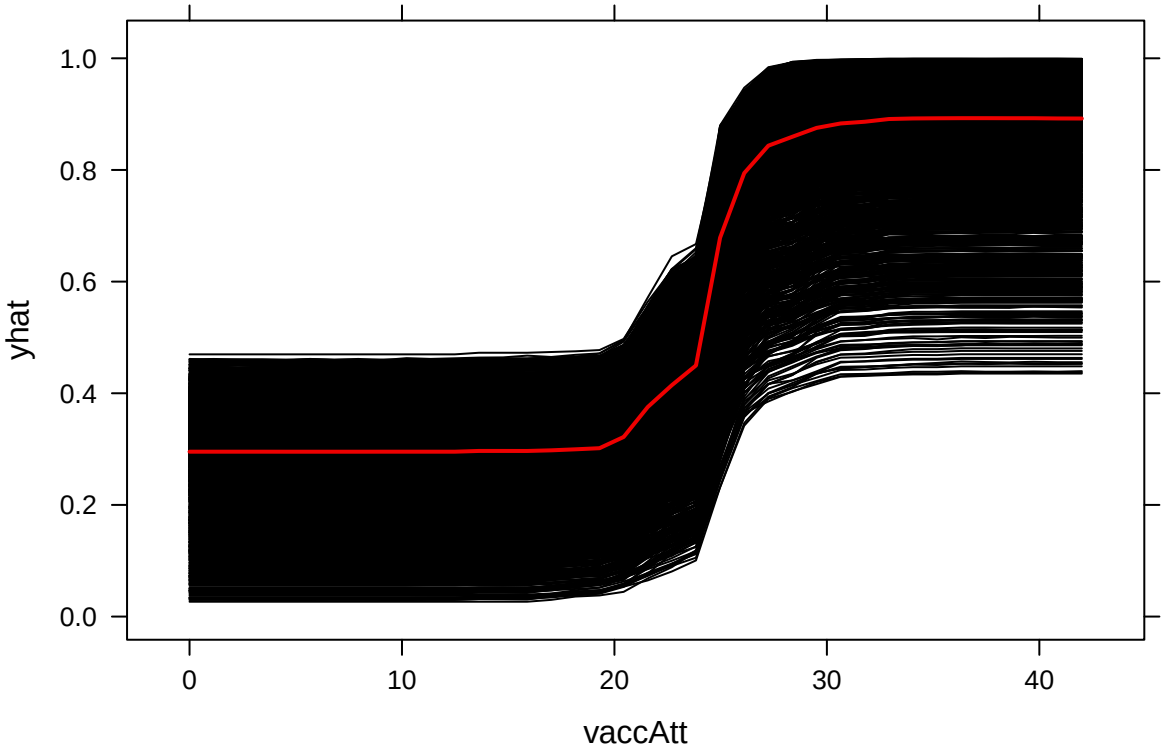


The variable vaccAtt shows the largest importance score and, therefore, seems to be the most influencial predictor of vaccWill.

Note: The varimp function calculates the importance score of a predictor as the difference in the OOB Error before vs. after permuting the predictor. However, as a default for classification not the misclassification rate is used but the log-likelihood to express the goodness of predictions. This is why it is possible to have a score larger than 1 as seen in our results.

e) To further investigate the influence of `vaccAtt` on `vaccWill`, draw the corresponding partial dependence plot (showing ICE lines as well). How do you interpret the plot? Note: The computation of the partial dependence will take approx. 5 minutes.

```
library(pdp)
partial(rf_vac, pred.var = "vaccAtt", prob = TRUE,
        which.class = 2, ice = TRUE, plot = TRUE)
```



The predicted probability of vaccWill=willing grows with a higher value of vaccAtt.

Exercise 3: Corona stress data (regression)

We now want to predict the numeric variable `stress` in the `dat_cvid` data, using all other variables as predictors.

a) Accordingly, fit a random forest (using the default values for `ntree` and `mtry`) to the corona data and calculate the mean squared error (MSE) of its OOB predictions (see last exercise for the formula of the MSE).

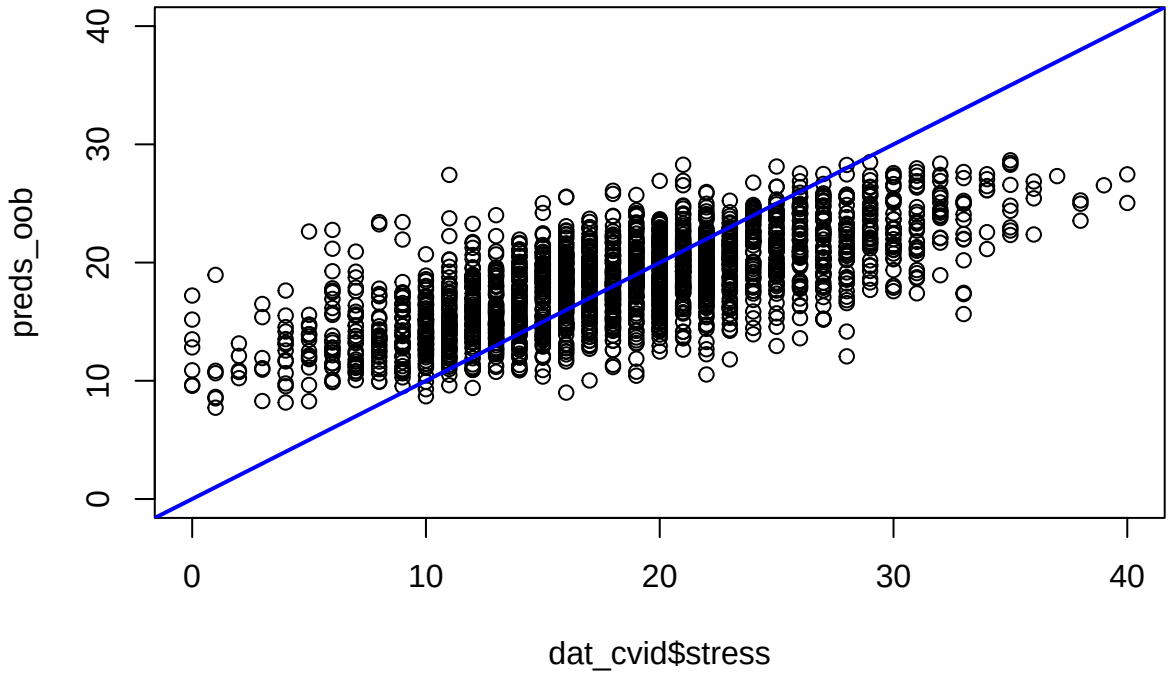
```
set.seed(8334)
rf_str <- cforest(stress ~., data = dat_cvid)
### Predictions for test data:
```

```
preds_oob <- predict(rf_str, OOB=TRUE)
### Calculate MSE:
rf_mse <- mean((dat_cvid$stress - preds_oob)^2)
rf_mse
```

```
## [1] 25.70772
```

b) It is difficult to interpret the MSE since it is just a number. Another way to visually express the accuracy of our predictions would be to plot the predicted `stress` values vs the true `stress` values in a scatter plot. Create such a plot and think about how the points would be positioned in such a plot if our predictions were perfect.

```
plot(x = dat_cvid$stress, y = preds_oob,
     xlim = c(min(dat_cvid$stress) ,max(dat_cvid$stress)),
     ylim = c(min(dat_cvid$stress) ,max(dat_cvid$stress)))
abline(a=0, b=1, col='blue', lwd=2)
```



We can see that our model seems to slightly overestimate small values of stress and underestimate higher values of stress. With perfect prediction the points would lie on the identity line (indicated in blue).

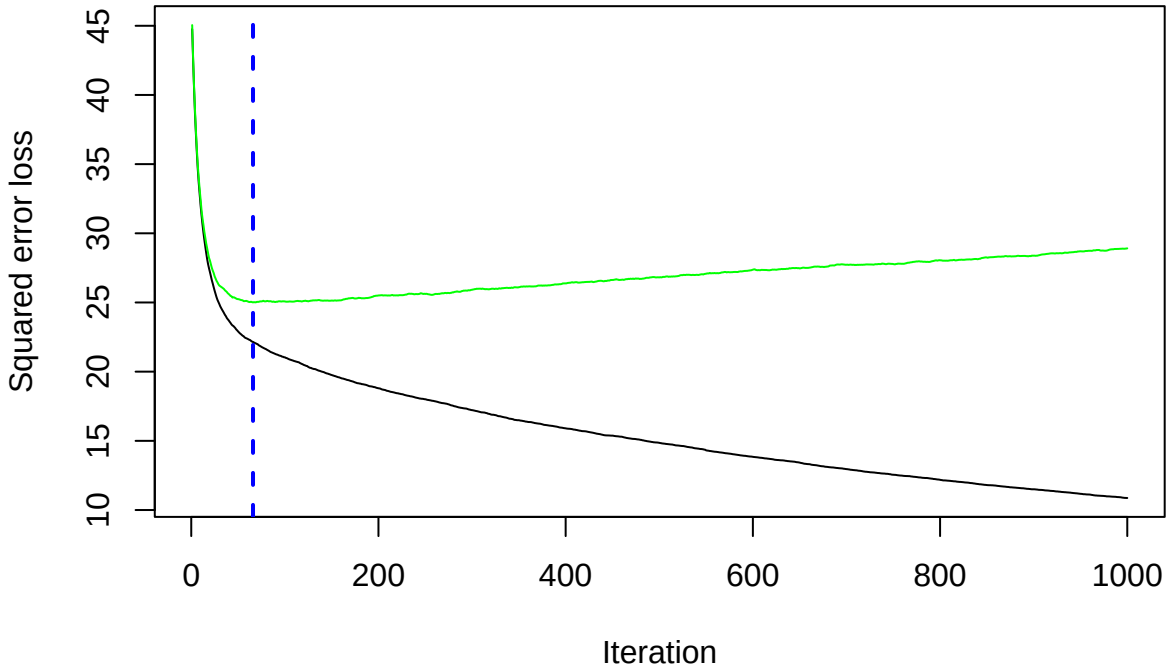
c) We now want to predict the `stress` variable with a gradient boosting ensemble (**Hint**: `gbm` package). Because we are doing regression, the `distribution` argument has to be set to `"gaussian"`. First, choose values for the hyperparameters `n.trees`, `shrinkage` and `interaction.depth` yourself and plot the training and test error (using 10-fold crossvalidation) vs. the number of trees (**Hint**: `gbm.perf()`). What number of trees showed the best performance? Try out different values of hyperparameters and check how the progression of errors changes.

```
library("gbm")
```

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```r
set.seed(2984)
boost_cvid <- gbm(stress ~., data = dat_cvid,
                  distribution = 'gaussian',
                  n.trees = 1000,
                  shrinkage = 0.1,
                  interaction.depth = 3,
                  cv.folds = 10)
### Find ideal number of trees:
idtr <- gbm.perf(boost_cvid, method = 'cv')
```




```r
idtr   # Number of trees with best performance
```

```
## [1] 66
```

One can see how the test error starts high (under-fitting), then decreases and finally starts increasing again (over-fitting). Given our selection of hyperparameters, the number of trees which showed the best performance was 66 (indicated by dashed line in plot). Your results may differ slightly from the presented results even when choosing the same seed.

d) We can extract the recorded MSEs (based on cross-validation) from the generated object using the `$` sign (`gbmobject$cv.error`). This vector contains the collected MSEs for all iterations. Extract the MSE which corresponds to the best tree number (determined in the previous exercise) and compare this MSE with the MSE of the random forest.

```r
c("Boosting_MSE"=boost_cvid$cv.error[idtr], "RandomForest_MSE"=rf_mse)
```

```
##     Boosting_MSE RandomForest_MSE
##          25.00812          25.70772
```