Exercises Head Start data

Marjolein Fokkema

Dataset: Predicting performance on cognitive tests

The data for this exercise are from a large study from the US, to evaluate long-term benefits of participation in the Head Start program. Head Start provides early childhood education, health, nutrition, and parent involvement services to children from low-income and families. The data comprise a birth cohort of children, who were enrolled around 1984 and tracked until 2004. The dataset contains both characteristics of the children and their families (mother, father, household).

The response variable of interest is `Test_Pct04`, a nationally normed percentile score (from 0-99) on a battery of cognitive tests in 2004. All predictors in the dataset were assessed between 1984 to 1996. Descriptions of the predictor variables are provided in a separate document in the repository.

Read in the data as follows and separate into 50% training and 50% test observations:

```
HSdata <- readRDS(file = "HeadStart.Rda")
set.seed(42)
train_ids <- sample(1:nrow(HSdata), size = nrow(HSdata)/2)
```

```
HSdata[train_ids, ] ## training data
HSdata[-train_ids, ] ## test data
```

Exercise 1: Fit a generalized additive model with smoothing splines

Fit a GAM using function `gam()` from package `mgcv` to predict the response variable `Test_Pct04` using smoothing splines of numeric predictors `AgeAFQT`, `PermInc`, and `Home_Pct88`, and the effects of categorical predictors `Race_Child` and `Father_HH86`, as shown in the code below. Consult the data documentation in the repository for the meaning of the predictors.

```
library("mgcv")
HSdata <- readRDS(file = "HeadStart.Rda")
set.seed(42)
gamod <- gam(Test_Pct04 ~ s(AgeAFQT) + s(PermInc) + Race_Child +
               s(HOME_Pct88) + Father_HH86,
             data = HSdata[train_ids,], distribution = "gaussian")
```

a) Apply function `summary` to the fitted model and interpret the results. Which predictors have a significant effect on the response? Which have the strongest effect?

b) Apply function `plot` to the fitted model and interpret the plots.

c) Use function `predict` to obtain predictions for the training and test observations. Compute test MSE and training MSE and compare.

Exercise 2: Fit a decision tree

a) Fit a conditional inference tree to the training data using function `ctree` (of package `partykit`). Inspect and interpret the result by printing and plotting.

b) Use function `predict` to obtain predictions for the training and test observations. Compute test MSE and training MSE and compare them. Also compare with the MSEs of the GAM fitted in the previous exercise.

Exercise 3: Fit a bagged tree ensemble and random forest

Make sure to `set.seed` before fitting the following ensembles, using function `randomForest` (from package with same name):

a) Fit a bagged ensemble by setting the `mtry` argument to p (i.e., `ncol(HSdata)-1L`). Specify `importance = TRUE` in the call to `randomForest` in order to obtain out-of-bag permutation importances later on.

b) Fit a random forest by setting `mtry` to $\sqrt(p)$. Again specify `importance = TRUE`.

c) Type `?randomForest` in R to find out: How many trees are generated by default? Is bootstrap or sub sampling employed by default? How can you change this? Which arguments are used to control tree depth and what are the defaults?

d) For both (bagged and RF) ensembles, use function `plot` to see how the OOB error decreases as a function of the number of trees. Would increasing the number of trees improve the accuracy?

e) For both ensembles, use functions `varImpPlot` and `importance` to inspect and interpret variable importances.

f) Use function `partialPlot` to inspect the effects of variables `AgeAFQT`, `PermInc`, `BirthOrder` and `Race_Child`. Note that you need to supply the (training) data to argument `pred.data` and the variable name to argument `x.var`. Describe the effects of these predictors.

g) Use function `predict` to obtain predictions for the training and test observations for both ensembles. Compute test MSE and training MSE and compare between the ensembles and with the single tree and GAM fitted earlier.

Exercise 4: Fit a gradient boosted tree ensemble

Fit a gradient-boosted ensemble, using function `gbm` (short for gradient boosting machine) from the package with the same name.

a) Load package `gbm` and type `?gbm` to find out: Which value is used by default for the learning rate (`shrinkage`)? How many trees are generated by default? Is bootstrap or sub sampling used by default? Which argument is used to control tree depth and what is the default?

In order to fit the boosted ensemble (i.e., stochastic gradient boosting is applied by default), random samples will be drawn, so make sure to set the random seed, first.

b) Fit a gradient-boosted ensemble on the training data using default settings. Specify `Test_Pct04` as the response in the model formula. Make sure to also specify `distribution = "gaussian"`, to indicate that this is a regression problem.

c) Apply function `gbm.perf` to the result to evaluate how error decreases (and perhaps increases again) as the number of trees increases.

d) Use function `summary` to obtain variable importances. What are the three most important predictors of performance on the cognitive test in 2024? Hints for plotting: Argument `cBars` controls the number of variables that importances are plotted for. To obtain readable variable labels in the plot, you can for example specify `las = 2` to have the labels printed horizontally instead of vertically; specify `cex.names = .5` to use smaller fonts for the labels; specify argument `cBars` to plot importances for a smaller number of variables.

e) Type `?summary.gbm` and check the method argument to see how the variable importances were computed.

f) Use function `plot` to obtain partial dependence plots of the three most important predictors. Use argument `i.var` to specify the names of the variables.

g) Use function `predict` to compute predictions for training and test observations. Use these predictions to compute training and test MSEs. Compare accuracy of the boosted ensemble with the GAM, single tree, bagged and random forest ensembles fitted earlier.

Exercise 5: Tune parameters of a gradient boosted tree ensemble

In this exercise, you tune the parameters of a gradient-boosted ensemble using cross validation (CV). To this end, we use function `train` from package `caret` (short for Classification And REgression Testing):

```
library("caret")
```

Each prediction method that can be tuned with function `train` has a pre-specified set of parameter values. These are rarely an exhaustive set of the parameters, but you can generally trust that these are the main parameters driving predictive accuracy. For each parameter, you need to specify a set of values. If you specify only a single value, the parameter will not be varied and tuned but kept fixed.

To inspect the pre-specified set of parameter values for the `gbm` method, type `modelLookup("gbm")`.

To specify the parameter values to test, provide a grid that has parameters on the columns and each row providing the set of values to try. Other approaches are also available (e.g., random search), but we will not use it in this course. Use the following grid (if you feel confident or adventurous, you may of course adjust this grid in any way you like):

```
grid <- expand.grid(shrinkage = c(.1, .01, .001),
                    n.trees = c(100, 1000, 2000, 2500),
                    interaction.depth = 1:4,
                    n.minobsinnode = 10)
```

a) Next, apply function `train`. Split the data into response and predictors, e.g.:

```
y <- HSdata$Test_Pct04
x <- HSdata[ , -which(names(HSdata) == "Test_Pct04")]
```

Make sure to set the random seed, first. Specify `x = x`, `y = y`, `tuneGrid = grid`, `method = "gbm"`, `trControl = trainControl(number = 10)`. Type `?trainControl` to see what this does.

Applying function `train` will take quite some time to run. Cross validating over the parameter grid involves fitting `nrow(grid)*10` models. You can additionally pass `verboseIter = TRUE` to function `trainControl` to have the fitting progress printed to the command line.

b) Once function `train` has completed, print the result. Use `plot` to visualize the result. For every parameter, make sure the results indicate that you have obtained the optimal value and you need not try a wider or finer grid.

c) Describe the main effects of the `shrinkage`, `n.trees` and `interaction.depth` parameters on the accuracy of the model. Also describe their possible interactions.

d) If you look at the effect of `interaction.depth`, what would you conclude about the possible presence of interactions?

e) Refit a gradient boosted ensemble on the training data using function `gbm`. This time, use the optimal set of parameter values returned by function `train` (you can extract this from the result of `train` as `$bestTune`). Use function `predict` to compute predictions for the training and test observations and compute training and test MSEs. Compare accuracy of this optmized boosted ensemble with that of all earlier models (GAM, single tree, bagged and random forest ensemble, boosted ensemble with default settings). Which method generalized best?