Exercise:
Ensemble Methods

Introduction to Machine Learning with R

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:

```r
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
```

```r
# 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`)

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?

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()`)

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.

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).

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?

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`?

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.

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).

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.

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.

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.