Data Analysis and Visualization Exercise 12

Daniela Klaproth-Andrade, Felix Brechtmann, Julien Gagneur

Quizzes (from the lecture)

The following quizzes will be solved orally by the students and the professor during the lecture.

1. We fit three classification models a dataset containing samples belonging to two classes (displayed as blue and red dots below). The model fits are illustrated with black lines. Assign one of the following scenarios to each of the three fitted models:

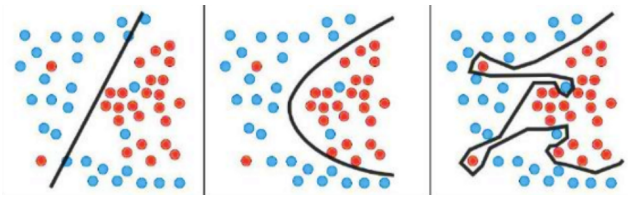

Figure 1: from [1]

a) The model overfits the data.
b) The model underfits the data.
c) The model fits the data appropriately.

2. We train a binary classification model that predicts for a given position (x, y) if a hole made by a fork in cake dough exists. Why does using cross-validation fail here?

[1] https://ichi.pro/de/techniken-zum-umgang-mit-underfitting-und-overfitting-im-machine-learning-57783049486521

a) Cross-validation only works for one-dimensional data points.
b) The position of each hole depends on the position of other holes.
c) There are not enough data points for using cross validation.
d) Cake dough (data) can only be eaten, not cross-validated.

3. A random forest consists of many decision trees that are trained on the same data set. How is randomness used to prevent over fitting? Select all that apply.

a) Randomness is introduced by using different subsets of the original data set for training each decision tree.
b) Randomness is introduced by CPU concurrency - parallel threads allow for tiny differences in feature importance.
c) Randomness is introduced by using different feature sets of the original data set for training each decision tree.

Tutorial

The following exercises will be solved during the tutorial sessions.

Section 00 - Getting ready

1. Make sure you have already installed and loaded the following libraries:

```
library(ggplot2)
library(data.table)
library(magrittr)
library(tidyr)
library(ggrepel)

library(plotROC)
library(caret)
library(rpart)
library(randomForest)
```

Section 01 - Supervised Learning

Are the following statements correct or incorrect? Please justify your answer.

a) A supervised learning algorithm does not require data labeled with an outcome.
b) When performing supervised training on two classes, the train set should only contain samples from one of these two classes. The test set should then only contain the class not used in the train set.
c) Models trained using cross validation do not over-fit.
d) A regression model that fits the train set perfectly, i.e. the train error is practically 0, can be used without further caution to collect predictions samples from any other dataset.

Section 02 - Tree-based models on Diabetes Dataset

As in the previous exercise sheet, in this section we are considering the dataset `pima-indians-diabetes.csv` which is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. A more detailed description of the data can be obtained from Kaggle: https://www.kaggle.com/uciml/pima-indians-diabetes-database.

Load the dataset with the following lines of code:

```
diabetes_dt <- fread("../../extdata/pima-indians-diabetes.csv")
diabetes_dt[, Outcome := as.factor(Outcome)]

# Store feature variables that we will need for later
feature_vars <- colnames(diabetes_dt[,-c("Outcome")])

diabetes_dt
```

1. Build a decision tree using the `rpart` function from the library `rpart` for predicting the `Outcome` given all feature variables. Use the following command for this:

```
full_formula <- as.formula(paste(c("Outcome ~ ",
                                   paste(feature_vars, collapse = " + ")),
                                 collapse = ""))
# We do not use this command full_formula <- Outcome~.
# Since we are adding some columns later and we do not want the formula to be altered

dt_classifier <- rpart(full_formula,
                       data =diabetes_dt,
                       control = rpart.control(minsplit = 3, cp = 0.001))
```

Note that `cp` determines when the splitting up of the decision tree stops and `minsplit` determines the minimum amount of observations in a leaf of the tree.

You can inspect the summary of your built decision tree with the command `summary(dt_classifier)`. Optionally, you can try to visualize your decision tree with the following commands:

```
#install.packages("rpart.plot")
library(rpart.plot)
rpart.plot(dt_classifier)
```

2. Plot a ROC curve for the decision tree using the function `geom_roc` from the library `plotROC`. What do you conclude about the performance of the decision tree based on this plot?

Hint: Use the function `predict()` to obtain the class probabilities predicted by the model for the specified dataset and translate them into a single score for each data point.

3. Build a second decision tree model this time using a train-test split strategy. This means that you will use 70% of the data for training and 30% of the data for testing. Plot the ROC curves for the performance on the training and on the test dataset. What do you conclude from this? *Hint*: To make your results reproducible, run the command `set.seed(13)` before splitting your data into train and test sets.

4. In the lecture we learned that random forests are more robust to overfitting. Build a random forest using the `randomForest` function from the library `randomForest` for predicting the Outcome given all feature variables using the same train-test split strategy from before. Set the following values for the following hyper-parameters:

- `ntree = 200` for the number of trees in the forest,
- `nodesize = 20` for the maximum amount of leaf nodes,
- `maxnodes = 7` for the minimum size of leaf nodes and
- `mtry = 5` for the number of variables randomly sampled as candidates at each split.

Plot the ROC curves for test and train set for the build random forest. Is the performance on the test set better than the one given from the previously built decision tree? How about the performance on the train set?

5. [OPTIONAL] Train a logistic regression model, as done last week, to predict the `Outcome` variable but this time only on the same train set as defined in the previous questions. Evaluate the performance on test set for the trained model. Discuss whether the logistic regression model performs better than the random forest.

Homework

Please solve the exercises below at home. The solutions will be discussed in the central exercise.

Section 03 - Cross Validation on the Diabetes Dataset

1. Implement a 5-fold cross-validation on the diabetes dataset for building a logistic regression model using all feature variables. Obtain 5-fold cross-validated sensitivity, specificity and AUC using the `caret` package. *Hint:* Use the functions `trainControl()` and `train()`.

2. What is the fold with the highest AUC?

3. Create a box plot displaying the specificity and sensitivity of the logistic regression model over all folds. Add the individual points to the box plot.

4. [OPTIONAL] Change the value k for the number of folds of the cross validation (e.g. $k = 2, 20, 100$). Does the average performance on the validation fold differ much from the initial result with 5 folds?

Section 04 - Searching for optimal Hyper-parameters

1. [OPTIONAL] Try changing the hyper-parameters of the random forest selected in Section 01 with the aim of achieving a better performance on both train and test sets evaluated with the same ROC curve as before. Two possible approaches for searching for optimal hyper-parameters are random and grid search. A short description of these approaches can be found here:
https://web.archive.org/web/20160701182750/http://blog.dato.com/how-to-evaluate-machine-learning-models-part-4-hyperparameter-tuning

Section 05 - Feature importances of feature variables

1. [OPTIONAL] The aim of this section is to investigate the importance of each feature variable on the Diabetes dataset to predict the `Outcome` variable. In this scenario, we define the feature importance of a variable as the difference in the computed area under the ROC curve of a model trained without this variable. Compute the feature importance of every feature variable and visualize these computed quantities with a suitable plot.