5 Hands On: Linear Regression and Tree-Based Models

Load the packages `tidyverse`, `tidymodels`, `rpart.plot` and `vip`

5.1 Multiple Linear Regression

1. Load the `Boston` data set from `MASS` package and inspect what it represents. Use this data set to answer the following questions.

```
data(Boston, package = "MASS")
```

(a) Separate the data set into 70% / 30% for training and test set, respectively.

(b) Perform a simple linear regression by the least-squares method between the variables `rm` and `medv`. Inspect the obtained results. Use the engine `lm`.

```
model_lm <- linear_reg(engine = "lm")
lm_fit1 <- model_lm %>%
    fit(medv ~ rm, data = boston_train)

tidy(lm_fit1)

ggplot(boston_train, aes(x = rm, y = medv)) + geom_point()

lm_preds1 <- boston_test %>%
    dplyr::select(medv) %>%
    bind_cols(predict(lm_fit1, boston_test))

lm_preds1 %>%
    metrics(truth = medv, estimate = .pred)
```

(c) Add the variable `crim` to the previous linear regression. Did it improve the previous results?

```
lm_fit2 <- model_lm %>%
    fit(medv ~ rm + crim, data = boston_train)
```

(d) Use all the predictor variables to build a Multiple Linear Regression model on the training data.

(e) Use the previous model to make predictions on the test set and assess its performance.

(f) Plot the output predictions

```
lm_preds %>%
    ggplot(aes(x = medv, y = .pred)) + geom_point() + geom_smooth(method = "lm",
    formula = "y ~ x") + ggtitle("Boston: Linear Regression Predictions") + xlab("True Values") +
    ylab("Predicted Values") + xlim(0, 50) + ylim(0, 50)
```

(g) Build a Ridge Regression model on the training data. Use the engine `glmnet` with `mixture=0`. Try different values for `penalty` (e.g. 100, 0.01). Inspect the obtained coefficients.

```
model_glm_ridge <- linear_reg(engine = "glmnet", penalty = 0.01, mixture = 0)
```

(h) Assess the performance of the previous model in the test set, considering different `penalty` values.

(i) Build a Lasso Regression model on the training data. Use the engine `glmnet` with `mixture=1`. Try different values for `penalty` (e.g. 100, 0.01). Inspect the obtained coefficients.

(j) Assess the performance of the previous model in the test set, considering different `penalty` values.

5.2 CART Trees

2. Build the CART trees for the following data sets using the `rpart` engine. Plot them using the function `rpart.plot()`. Use the package `vip` to inspect the variable importance. With `vip()` you get one consistent interface to computing variable importance for many types of supervised learning models across a number of packages.

(a) The `Boston` data set.

```
model_rt <- decision_tree(mode = "regression", engine = "rpart")
rt_fit <- model_rt %>%
    fit(medv ~ ., data = Boston)

library(rpart.plot)
# to extract it from the parsnip
rt_fit %>%
    extract_fit_engine() %>%
    rpart.plot(roundint = FALSE)
```

```
library(vip)
vip(rt_fit)

# bostonTree <- rt_fit %>% extract_fit_engine() bostonTree$variable.importance
```

(b) The `iris` data set.

(c) The golf.csv data set. What happened? Try again with the parameters `min_n = 2`.

3. Assess the performance of CART for `Boston` data set under the experimental setup stated in question 1.