Chapter 2 - the modelling process

Prerequisites

Note that `h2O.init()` gives a warning about the version being old, but can be ignored.

```r
# Helper packages
library(dplyr)     # for data manipulation
library(ggplot2)   # for awesome graphics

# Modeling process packages
library(rsample)   # for resampling procedures
library(caret)     # for resampling and model training
suppressWarnings(suppressMessages(library( h2o )))
```

```r
# just print
# h2o set-up
h2o.no_progress()  # turn off h2o progress bars
h2o.init()        # launch h2o
```

Load the data and convert it into an h2o object. Why? Well apparently h2o objects are what ML peeps use instead of data.frames. What is h2o? Apparently an open source platform for ML (or AI as they call it), that also of course interfaces with R, python etc. We will use it later on. But I'm still not clear on what the objects asre

```r
# load ames housing data
ames <- AmesHousing::make_ames()
ames.h2o <- as.h2o(ames)

# Job attrition data
# load and change ordered factors into regular factors
churn <- rsample::attrition %>%
  mutate_if(is.ordered, .funs = factor, ordered = FALSE)

churn.h2o <- as.h2o(churn)
```

Data splitting

Goal: find $f(X)$ that best predicts $\hat{Y}$ based on some X. You want the alg to generalise to X values not seen already. So we split our dataset into

- **Training data** – used to develop feature sets, train the alg, tune the hyperparameters
- **Testing data** – used to estimate the unbiased estimate of our model's performance - the *generalization error*.

Too much data in the training set (>80%) leads to overfitting and we don't get a good assesment of predictive performace.

Too much on testing (>40%) can also mean we don't get good estimates of model parameters.

Also in very large data sets increasing the training set may lead to only marginal gains, but take a lot longer, so unnecessary.

We split the data usually using either *simple random sampling* or *stratified sampling*

Simple random sampling

Simple random sampling ignores the distribution of features, including the outcome variable.

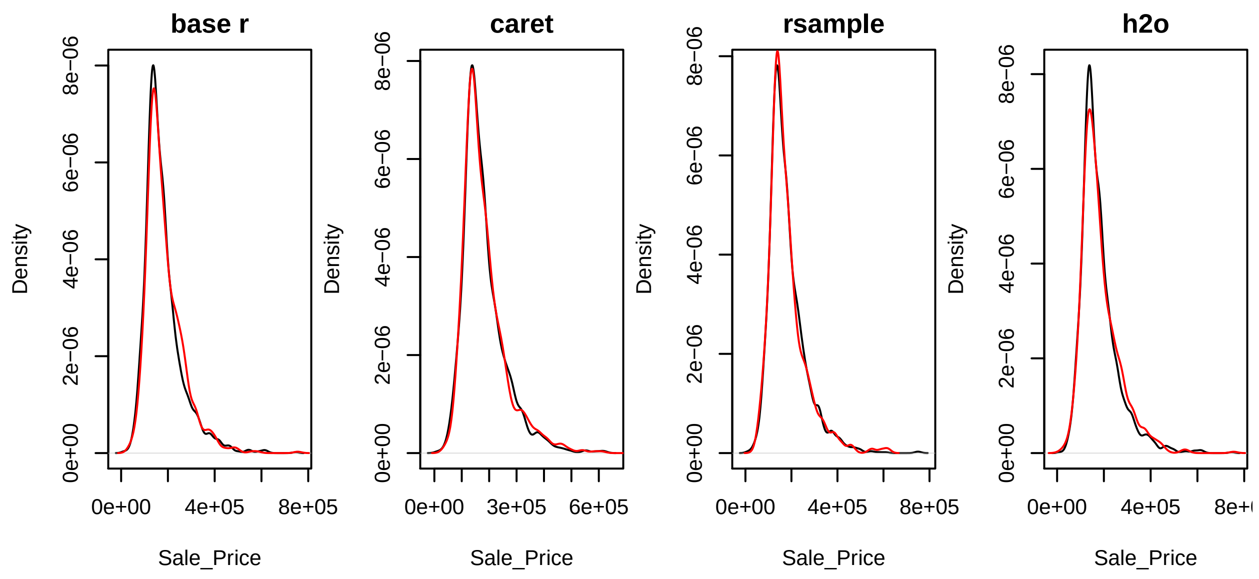

Figure 1: Distribution of target variable in training (black) and testing (red) sets

```
set.seed(123)
# Using base R
index_1 <- sample(1:nrow(ames), round(nrow(ames) * 0.7))
train_1 <- ames[index_1, ]
test_1  <- ames[-index_1, ]

# Using caret package
set.seed(123)
index_2 <- createDataPartition(ames$Sale_Price, p = 0.7,
                               list = FALSE)
train_2 <- ames[index_2, ]
test_2  <- ames[-index_2, ]

# Using rsample package
set.seed(123)  # for reproducibility
split_1  <- initial_split(ames, prop = 0.7)
train_3  <- training(split_1)
test_3   <- testing(split_1)

# Using h2o package
split_2 <- h2o.splitFrame(ames.h2o, ratios = 0.7,
                          seed = 123)
train_4 <- split_2[[1]]
test_4  <- split_2[[2]]
```

If the dataset is large enough it is usually sufficient to get a similar distribution of the target variable Y in the training set and the test set.

Stratified sampling