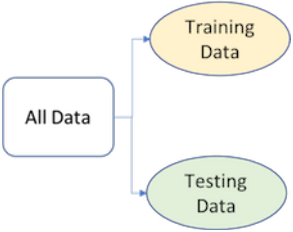
Lecture 5:
Cross-Validation for Regression
STAT 452, Spring 2021

Cross-Validation

- A natural question to ask is how well does a regression model perform at predicting new, future values of the response variable.
- One way to directly address this question is through a data-splitting technique called cross-validation.

Here is one simple approach to cross-validation, called the **validation set approach** or **hold-out method**:

- First the data set is randomly divided into two parts: a **training set** and a **validation** or **test set**.
- The regression model is fit (estimated) on data in training set.
- Then the fitted regression model is used to predict the responses for the observations in the test set.
- The predicted response values are then compared to the actual response values on the test set.



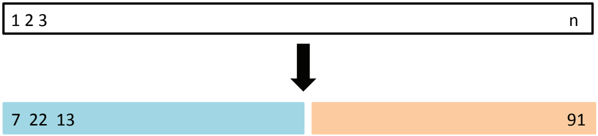


Figure: A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical model is fit on the training set, and its performance is evaluated on the validation set.

- Typical recommendations for splitting your data into training-test splits include 60% (training)–40% (testing), 70%–30%, or 80%–20%
  - Spending too much in training (e.g.,$>$ 80%) won't allow us to get a good assessment of predictive performance.
  - Spending too much in testing (e.g., $>$ 40%) won't allow us to get a good assessment of model parameters.
- There are a number of ways to split our data in R.

Splitting data using base R:

```
set.seed(123)
train_index <- sample(1:nrow(ames), round(nrow(ames) * 0.7))
ames_train <- ames[train_index, ]
ames_test <- ames[-train_index, ]
```

Note we use `set.seed()` to make the results of the random splitting reproducible.

Splitting data using caret package:

```
library(caret)

set.seed(123)
train_index <- createDataPartition(ames$Sale_Price,
                                   p = 0.7, list = FALSE)
ames_train <- ames[train_index, ]
ames_test <- ames[-train_index, ]
```

Assessing Model Performance

The **mean-square error (MSE)** is the most commonly used measure for the performance of a regression model on withheld test data.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- n is the number of observations in the test set
- y_i is the i^{th} observation in the test set
- $\hat{y}_i$ is the i^{th} prediction in the test set

The **root mean squared error (RMSE)** is defined as

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

which is in the same units as the data (e.g., if the data are in units `cm` then the MSE is in $\texttt{cm}^2$ and the RMSE is in `cm`).

- Note that the MSE and RMSE are performance measures for regression tasks, that is, predicting a quantitative response variables (e.g., predicting salary, weight).
- Different types of performance measures are used for classification tasks, that is, predicting a categorical response variables (e.g., classifying emails as `spam` or `not spam`)

These other performance measures are also sometimes reported:

- **Mean absolute error (MAE)**:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- **Test $\mathbf{R}^2$**: proportion of the variance in the response variable that is predictable from the explanatory variables.
  - Can be computed as the squared correlation between the predicted and actual values of the response on the test set (can use `cor()` function).
  - Not to be confused with the **training $\mathbf{R}^2$**, which is what is reported in the regression output from the `summary()` function.