Notes on Ch5: SpendingOurData

Norman LIm

2025-06-28

- some of the steps in creating a useful model:

1. parameter estimation
2. model selection
3. model tuning
4. performance assessment

- data budget is the pool of data available for all the tasks needed to complete the steps in model creation
- when data are *reused* for multiple tasks (instead of getting 'spent' from the data budget), some risk factors are increased
- examples of these risks include bias accentuation and compounding the effects of methodological errors
- it is considered a smart strategy to allocate specific subsets of data for different tasks
- if the initial data pool is not large, there will be some overlap between when and how data is allocated.

Example of smart data allocation: - when data and predictors are abundant, it is wise to first spend a specific subset of data to determine which predictors are informative

Common methods for splitting data

- one of the most common way of "spending" the data budget is by splitting data into training and testing sets

Example: splitting data using the `initial_split()` function from the **rsample** package

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.3.0 --
```

```
## v broom        1.0.8     v recipes      1.3.1
## v dials        1.4.0     v rsample      1.3.0
## v dplyr        1.1.4     v tibble       3.3.0
## v ggplot2      3.5.2     v tidyr        1.3.1
## v infer        1.0.8     v tune         1.3.0
## v modeldata    1.4.0     v workflows    1.2.0
## v parsnip      1.3.2     v workflowsets 1.1.1
## v purrr        1.0.4     v yardstick    1.3.2
```

```
## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
```

```
tidymodels_prefer()

# set random seed for reproducibility
set.seed(501)

# Save the split information for an 80/20 split of data
ames_split <- initial_split(ames, prop = 0.8)
ames_split
```

```
## <Training/Testing/Total>
## <2344/586/2930>
```

- the output denotes the number of data points for each "split" – 2344 (80%), 586 (20%), out of 2930 (100%)
- this result is an `rsplit` object that we can pass to the `training()` and `testing()` functions to produce the split data:

```
ames_train <- training(ames_split)
ames_test <- testing(ames_split)

dim(ames_train)
```

```
## [1] 2344   74
```

```
dim(ames_test)
```

```
## [1] 586  74
```

- when there is a dramatic *class imbalance*, this random splitting is not OK, since the majority class will swamp out the minority class in the split data
- one way to remedy this is by using *stratified sampling* instead, to preserve the proportion of classes of the original data when we split them (into training and testing sets)
- we can use only 1 column for stratification

```
set.seed(502)
ames_split <- initial_split(ames, prop = 0.8, strata = Sale_Price)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)

dim(ames_train)
```

```
## [1] 2342   74
```

```
dim(ames_test)
```

```
## [1] 588  74
```

- when data is time-series, random sampling is usually not advisable
- instead we split the data by the proportion of the first part of the data to the last part of the data

What about validation sets?

- validation sets are small part of data that we hold back and use to evaluate the training of ML models.
- after training, the test set can be used to evaluate the performance of the "best" model

Example of splitting method with a validations set:

```
# 60% training, 20% validation, 20% testing
ames_val_split <- initial_validation_split(ames, prop = c(0.6, 0.2))
ames_val_split
```

```
## <Training/Validation/Testing/Total>
## <1758/586/586/2930>
```

- extracting the data train, test, and validation sets:

```
ames_train <- training(ames_val_split)
ames_test <- testing(ames_val_split)
ames_val <- validation(ames_val_split)
```

Multilevel Data

- in some data set, in information in each row is considered to be independent of each other (1 row per experimental unit).
- in the Ames housing data set, for example, property is considered to be the *independent experimental unit*
- for other applications, this is not always the case

Examples of data having multiple rows per experimental unit):
- longitudinal studies where the same experimental unit can be measured over multiple time points (ex. human subject in a medical trial) - repeated measures design where replicate data points from a batch are collected at multiple times

Considerations when dealing with multilevel data:
- simple resampling is a no-no since this will lead to some data within one experimental unit to be in the training set and test set - data splitting should occur at the experimental unit level of data

Other considerations for Budgeting Data

- quarantine the test set from any model building activities to prevent "data leakage" or "information leakage"
- information leakage occurs when data **outside** of the training set are used in the modeling process
- it is critical that the test set mirrors what the model would encounter in the wild (test sets should always resemble new data that will be fed to the model)