Supervised Modeling Process (MSBA 7027)

Zhengli Wang

Faculty of Business and Economics
The University of Hong Kong
2023

The ML Process

- ML process is very iterative and heuristic-based.
- With minimal knowledge of the problem or data at hand, it is difficult to know which ML method will perform the best.
- It is common for many ML approaches to be applied, evaluated, and modified before a final, optimal model can be determined.

Approach ML modeling correctly

- Spending our data wisely on learning and validation procedures.
- Properly pre-processing the feature
- Minimizing data leakage
- Tuning hyperparameters
- Assessing model performance.

- Modeling process is NOT a short sprint
- BUT a **marathon**: many iterations of these steps repeated before eventually finding the final optimal model.

The ML Process: A Systematic Approach

But also keep in mind: in real life, every data set is different
NOT always 100% correct, exception to the rule happens frequently

The ML Process

- Prerequisite
- Data Splitting: Stratified Sampling
- Direct vs Meta Engine
- Resampling Methods: K-fold CV & Bootstrap
- Hyperparameter Tuning
- Model Performance Metric

The ML Process: Prerequisite

```
# 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
library(h2o)       # for resampling and model training

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

The ML Process: Prerequisite

Ames Housing Data

```
# Ames housing data

ames <- AmesHousing::make_ames()
```

Rich dataset, used in Kaggle Competition
(More details see HMLR Chapter 3.2)

The ML Process: Prerequisite

Ames Housing Data

```
> summary(ames)
                              MS_SubClass                                  MS_Zoning     Lot_Frontage       Lot_Area
One_Story_1946_and_Newer_All_Styles :1079   Floating_Village_Residential: 139   Min.   :  0.00   Min.   :  1300
Two_Story_1946_and_Newer            : 575   Residential_High_Density    :  27   1st Qu.: 43.00   1st Qu.:  7440
One_and_Half_Story_Finished_All_Ages: 287   Residential_Low_Density     :2273   Median : 63.00   Median :  9436
One_Story_PUD_1946_and_Newer        : 192   Residential_Medium_Density  : 462   Mean   : 57.65   Mean   : 10148
One_Story_1945_and_Older            : 139   A_agr                       :   2   3rd Qu.: 78.00   3rd Qu.: 11555
Two_Story_PUD_1946_and_Newer        : 129   C_all                       :  25   Max.   :313.00   Max.   :215245
(Other)                             : 529   I_all                       :   2
 Street                 Alley                     Lot_Shape    Land_Contour  Utilities      Lot_Config   Land_Slope
Grvl:  12   Gravel         : 120   Regular             :1859   Bnk: 117     AllPub:2927   Corner : 511   Gtl:2789
Pave:2918   No_Alley_Access:2732   Slightly_Irregular  : 979   HLS: 120     NoSeWa:   1   CulDSac: 180   Mod: 125
            Paved          :  78   Moderately_Irregular:  76   Low:  60     NoSewr:   2   FR2    :  85   Sev:  16
                                   Irregular           :  16   Lvl:2633                   FR3    :  14
                                                                                         Inside :2140
```

…. (Many features omitted)

```
  Pool_Area           Pool_QC                    Fence      Misc_Feature    Misc_Val           Mo_Sold
Min.   :  0.000   Excellent:   4   Good_Privacy     : 118   Elev:   1    Min.   :    0.00   Min.   : 1.000
1st Qu.:  0.000   Fair     :   2   Good_Wood        : 112   Gar2:   5    1st Qu.:    0.00   1st Qu.: 4.000
Median :  0.000   Good     :   4   Minimum_Privacy  : 330   None:2824    Median :    0.00   Median : 6.000
Mean   :  2.243   No_Pool  :2917   Minimum_Wood_Wire:  12   Othr:   4    Mean   :   50.63   Mean   : 6.216
3rd Qu.:  0.000   Typical  :   3   No_Fence         :2358   Shed:  95    3rd Qu.:    0.00   3rd Qu.: 8.000
Max.   :800.000                                            TenC:   1    Max.   :17000.00   Max.   :12.000

  Year_Sold       Sale_Type    Sale_Condition   Sale_Price      Longitude        Latitude
Min.   :2006   WD     :2536   Abnorml: 190   Min.   : 12789   Min.   :-93.69   Min.   :41.99
1st Qu.:2007   New    : 239   AdjLand:  12   1st Qu.:129500   1st Qu.:-93.66   1st Qu.:42.02
Median :2008   COD    :  87   Alloca :  24   Median :160000   Median :-93.64   Median :42.03
Mean   :2008   ConLD  :  26   Family :  46   Mean   :180796   Mean   :-93.64   Mean   :42.03
3rd Qu.:2009   CWD    :  12   Normal :2413   3rd Qu.:213500   3rd Qu.:-93.62   3rd Qu.:42.05
Max.   :2010   ConLI  :   9   Partial: 245   Max.   :755000   Max.   :-93.58   Max.   :42.06
               (Other):  21
```

```
> ncol(ames)
[1] 81
```

The ML Process: Data Splitting

Want an algorithm that:

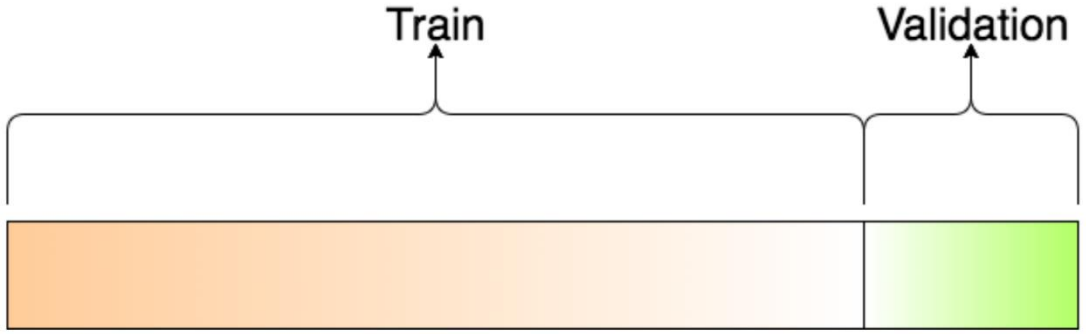
- Not only fits well to our past data
- But also predicts a future outcome accurately

Training set: develop feature sets & train our algorithms, tune hyperparameters, compare models and to choose a final model

Validation set: estimate an unbiased assessment of our final model's performance

Note: validation set should not be used prior to selecting the final model.

The ML Process: Data Splitting

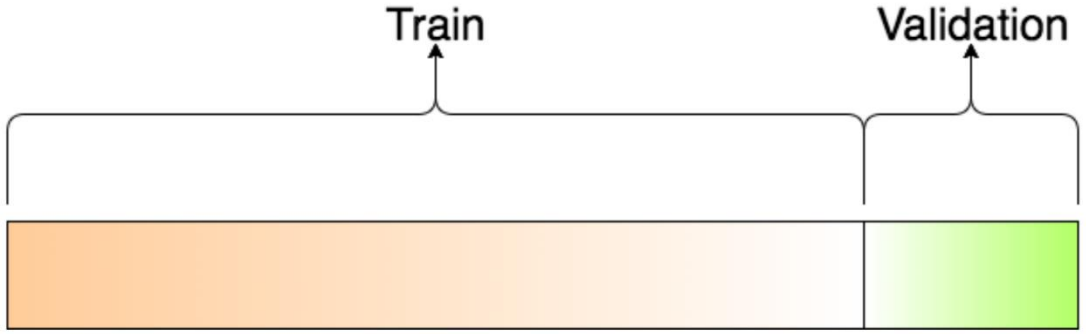


Typical recommendations for splitting your data into training-validation splits:

- 60%–40%;
- 70%–30%; or
- 80%–20%.

- Too much in training
  - Too little in validation - assessment of model performance not accurate.
  - May find a model that overfits
- Too little in training
  - Model may not be good (underfitting)

The ML Process: Data Splitting



Typical guidelines

For very large data sets: smaller % for training

- Larger % for training only marginal gains
- ↑ computation speed

For very small data sets: larger % for training

- Larger % often needed to train a good model

The ML Process: Data Splitting

- Splitting data: Stratified sampling
- Control the sampling: training & validation sets have similar *Y* distributions
- Can be used in both regression (e.g., segment into quantiles and randomly sample from each) and classification (e.g., Same percentage of "Yes" and "No") problems

The ML Process: Data Splitting

Splitting data: Stratified sampling

Perform stratified sampling: use **rsample package** (have seen this in KNN)

```
# orginal response distribution
table(churn$Attrition) %>% prop.table()
##
##        No       Yes
## 0.8387755 0.1612245


# stratified sampling with the rsample package
set.seed(123)
split_strat  <- initial_split(churn, prop = 0.7,
                              strata = "Attrition")
train_strat  <- training(split_strat)
test_strat   <- testing(split_strat)
```

```
# consistent response ratio between train & test
table(train_strat$Attrition) %>% prop.table()
##
##        No       Yes
## 0.838835 0.161165
table(test_strat$Attrition) %>% prop.table()
##
##        No       Yes
## 0.8386364 0.1613636
```

Similar percentage

The ML Process: Direct Engine vs Meta Engine

Direct Engines (e.g. lm, glm, knn, svm)

```
lm_lm    <- lm(Sale_Price ~ ., data = ames)
lm_glm   <- glm(Sale_Price ~ ., data = ames, family = gaussian)
```

Meta Engines (e.g. caret::train)

```
lm_caret <- caret::train(Sale_Price ~ ., data = ames, method = "lm")
```

- Meta engines: can apply any direct engine with [method = "<method-name>"]
- Direct engines: need to deal with syntax differences of each method

Table 1: Syntax for computing predicted class probabilities with direct engines.

Algorithm	Package	Code
Linear discriminant analysis	**MASS**	`predict(obj)`
Generalized linear model	**stats**	`predict(obj, type = "response")`
Mixture discriminant analysis	**mda**	`predict(obj, type = "posterior")`
Decision tree	**rpart**	`predict(obj, type = "prob")`
Random Forest	**ranger**	`predict(obj)$predictions`
Gradient boosting machine	**gbm**	`predict(obj, type = "response", n.trees)`

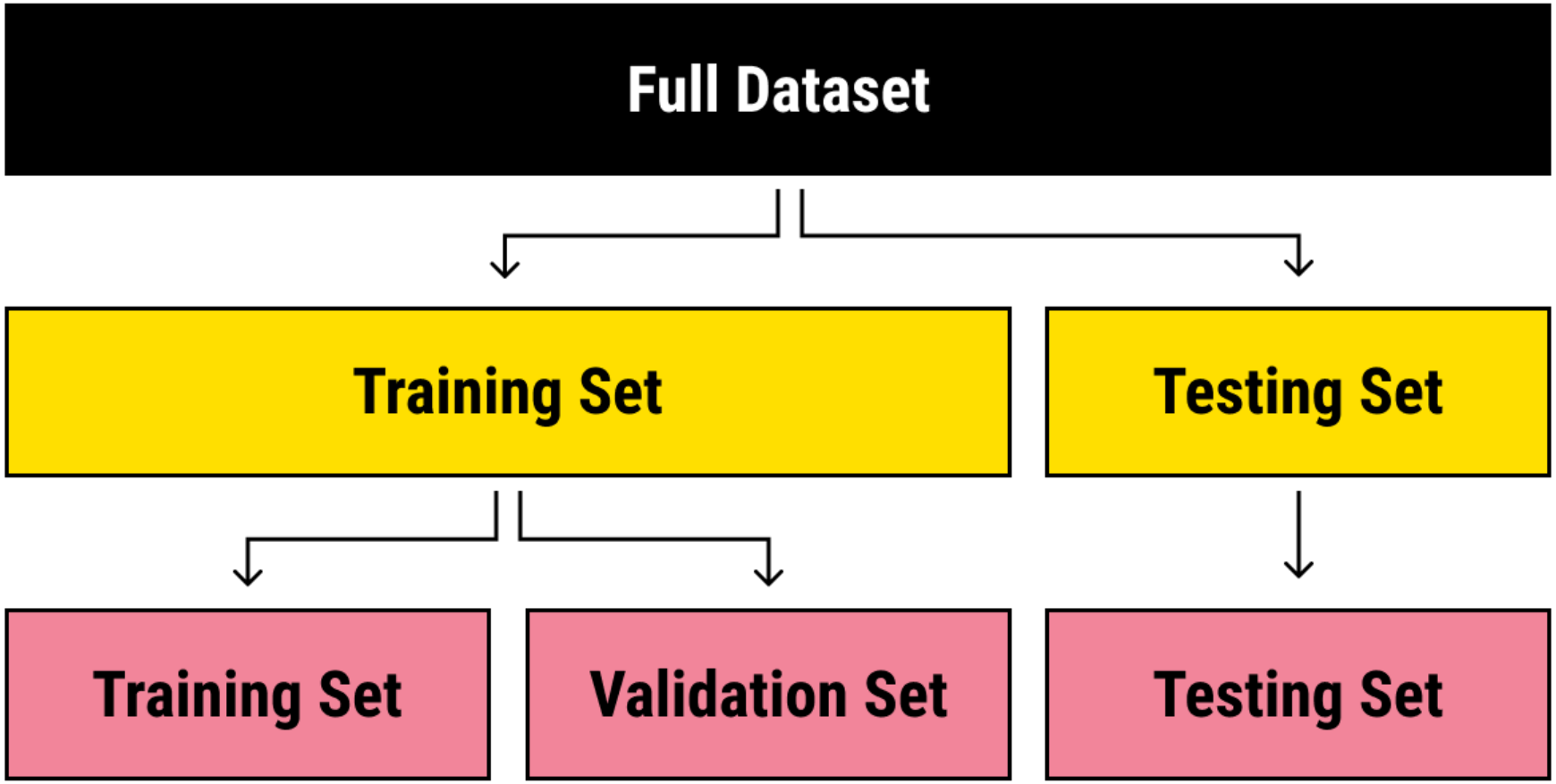
The ML Process: Resampling Methods

Definition of CV

- Any procedure that produces an error estimate of a model without using the training data

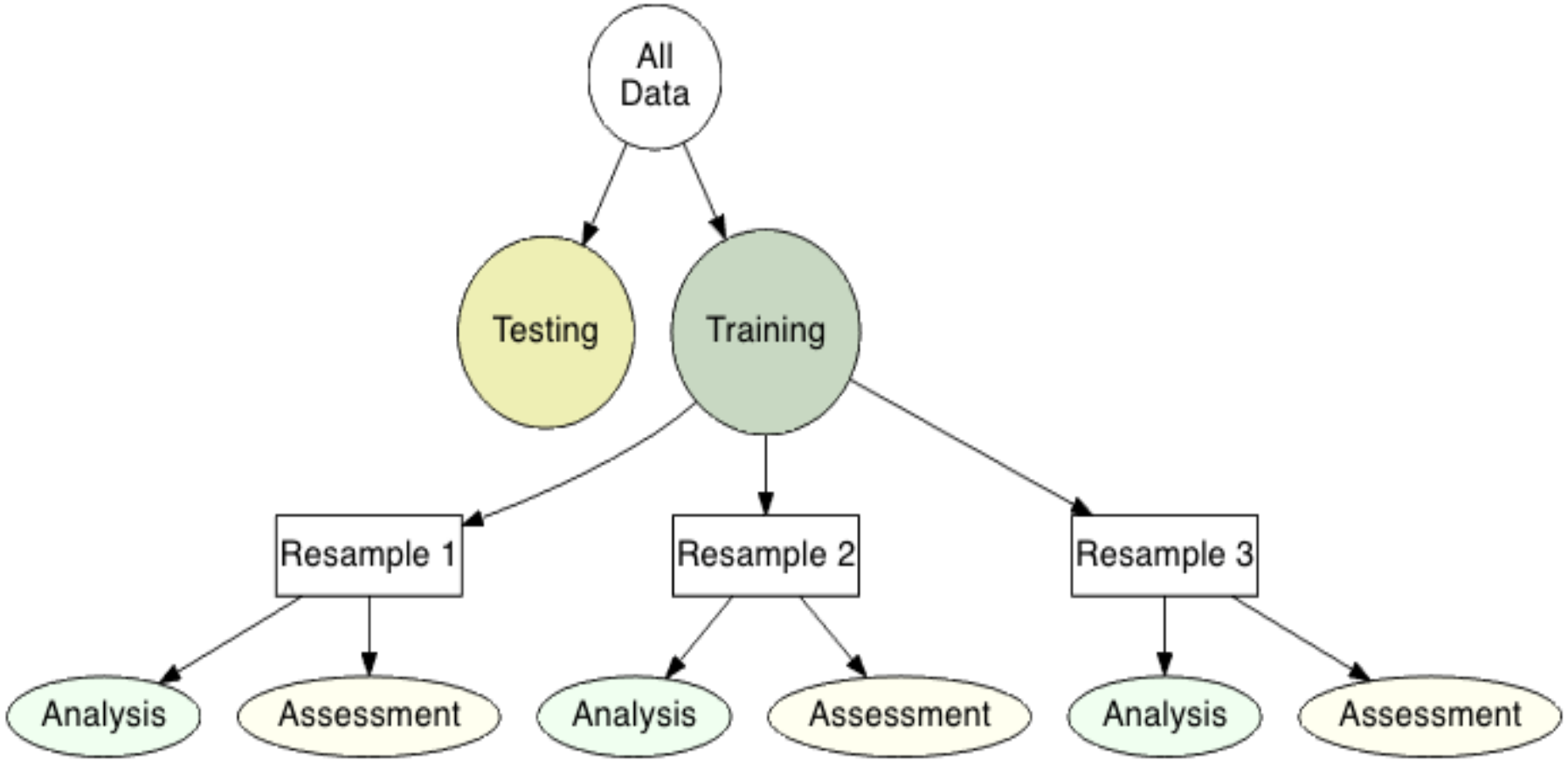
- We have seen K-fold CV, but there are other kinds, e.g. Bootstrapping

The ML Process: Resampling Methods



Source: https://labelyourdata.com/articles/machine-learning-and-training-data

The ML Process: Resampling Methods



The ML Process: Resampling Methods

Resampling methods: Repeatedly do the following

- Fit a model to parts of training data
- Assess performance on other parts.

Two common resampling methods:

- K-fold CV
- Bootstrapping

The ML Process: Resampling Methods

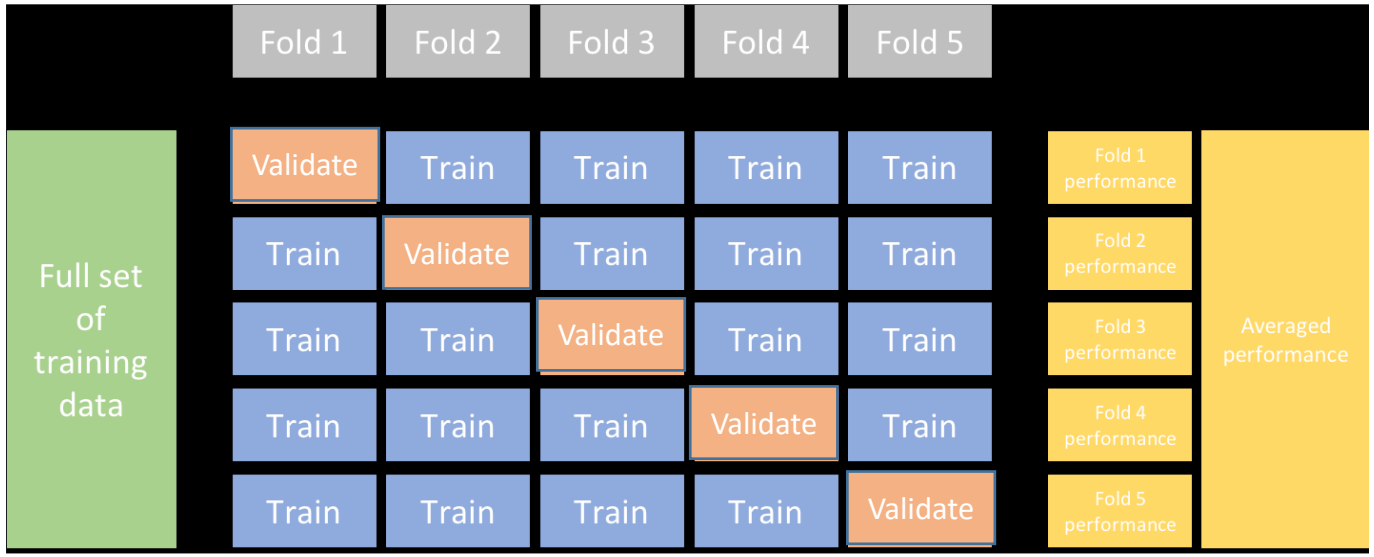
K-fold CV

- Randomly divides training data into K groups (folds) of approx. equal size
- Fit the model on $K-1$ folds
- Assess model performance on remaining fold
- Procedure repeated K times: obtain K estimates of error

- K-fold CV error = average of the K errors.
  - Typically uses $K = 5$ or $K = 10$
  - $K \uparrow$: Estimated error more accurate, but computationally more demanding.
  - Holdout method / LOOCV

The ML Process: Resampling Methods

K-fold CV




```
# example of 10 fold CV in caret
caret_cv <- train(
   Sale_Price ~ .,
   data = ames_train,
   method = "lm",
   trControl = trainControl(method = "cv", number = 10)
)
```

The ML Process: Resampling Methods

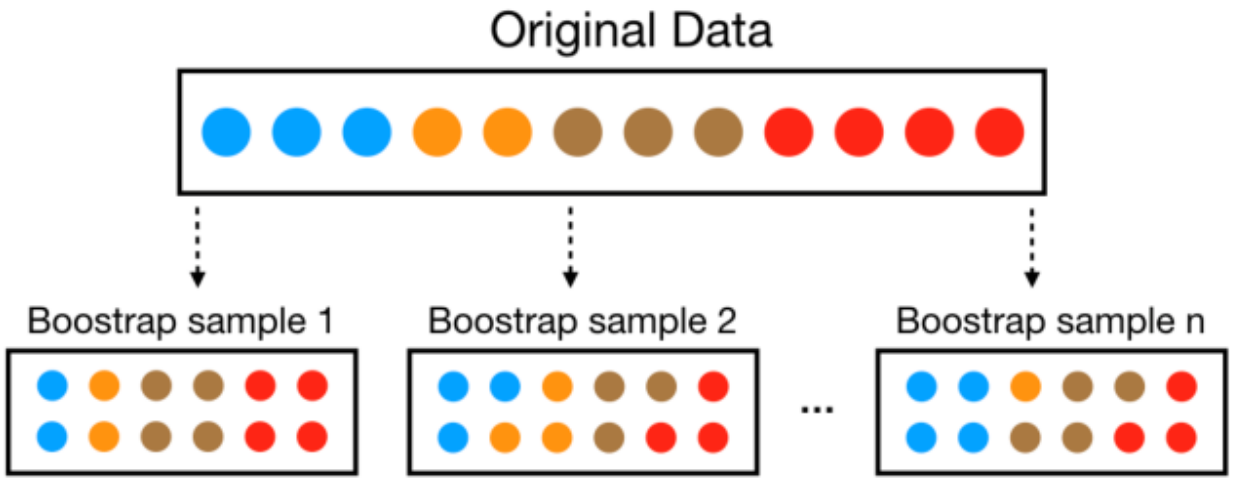
Bootstrapping

- A bootstrap sample
  - a random sample of the data taken *with replacement*
  - typically has the same size as the original data set
  - contain approx. the same distribution of values as the original data set.

- For large sample size, a data point has a 63% probability of appearing in a bootstrap sample
- Observations NOT contained in a bootstrap sample: out-of-bag (OOB)
- When bootstrapping, a model is validated on the OOB samples

The ML Process: Resampling Methods

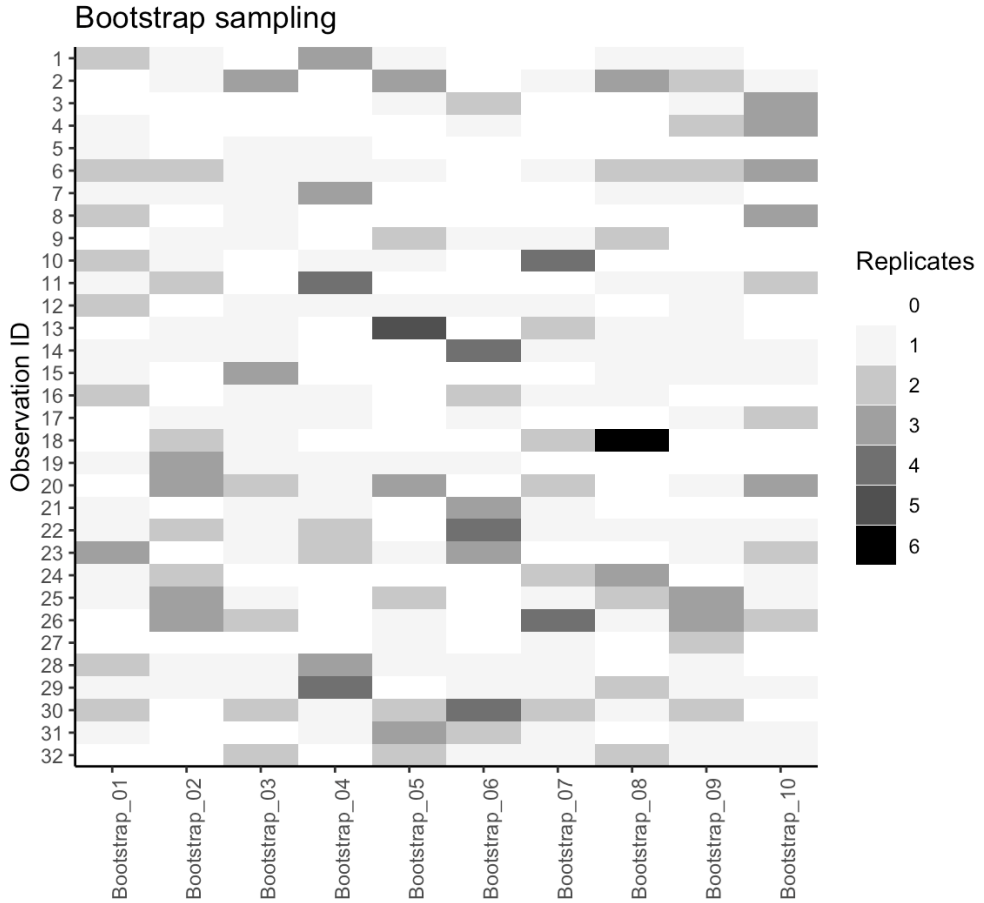
Bootstrapping



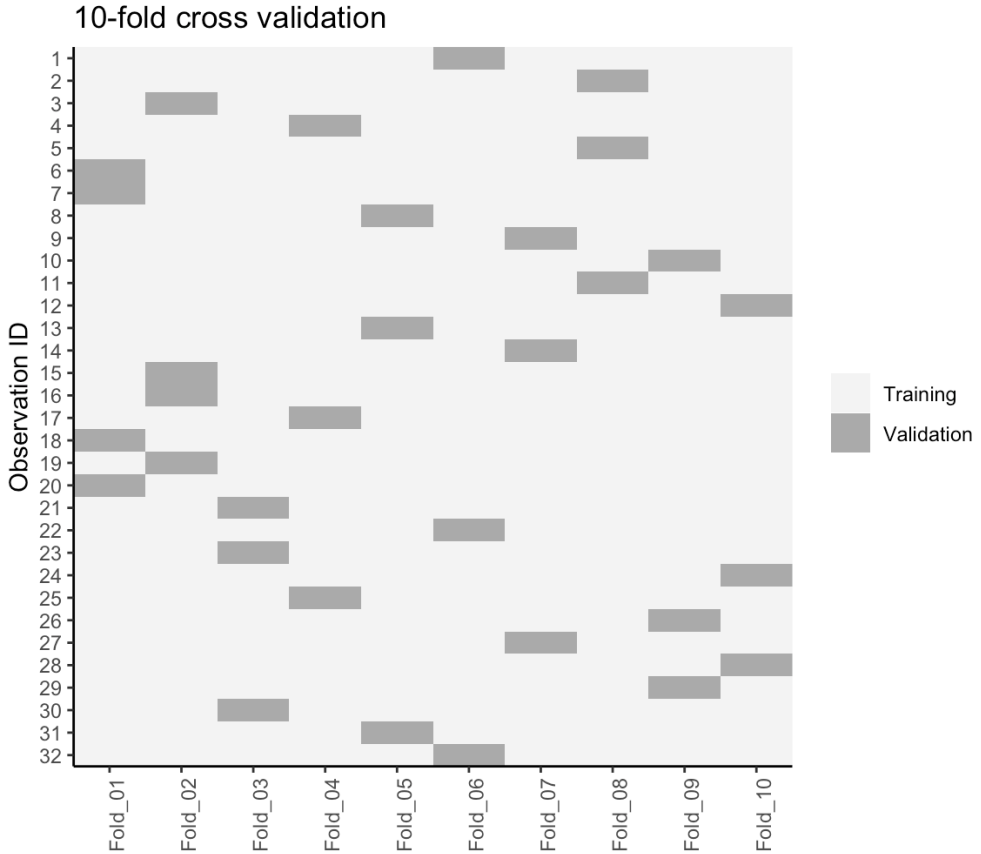

```
# example of 10 bootstrap samples in caret
caret_boot <- train(
    Sale_Price ~ .,
    data = ames_train,
    method = "lm",
    trControl = trainControl(method = "boot", number = 10)
)
```

The ML Process: Resampling Methods

Bootstrapping vs K-fold CV: Visualization



Bootstrapping



10-Fold CV

Darker color: more #replicates of the observation

The ML Process: Resampling Methods

Bootstrapping compared to K-fold CV

By [Efron, 1983], bootstrapping typically has

- Less variability in the error measure
- Higher bias of error estimate

The ML Process: Hyperparameter Tuning

Hyperparameter vs Parameter

The ML Process: Hyperparameter Tuning

Grid Search: expand.grid func

```
hyper_grid <- expand.grid(
  mtry = c(4,12,20),
  min.node.size = 3,
  replace = c(TRUE, FALSE),
  sample.fraction = c(.5, .63, .8)
)
```

```
> hyper_grid
   mtry min.node.size replace sample.fraction
1     4             3    TRUE             0.50
2    12             3    TRUE             0.50
3    20             3    TRUE             0.50
4     4             3   FALSE             0.50
5    12             3   FALSE             0.50
6    20             3   FALSE             0.50
7     4             3    TRUE             0.63
8    12             3    TRUE             0.63
9    20             3    TRUE             0.63
10    4             3   FALSE             0.63
11   12             3   FALSE             0.63
12   20             3   FALSE             0.63
13    4             3    TRUE             0.80
14   12             3    TRUE             0.80
15   20             3    TRUE             0.80
16    4             3   FALSE             0.80
17   12             3   FALSE             0.80
18   20             3   FALSE             0.80
```

The ML Process: Model Evaluation Metric

Regression

- Mean Square Error (MSE) / Root Mean Square Error (RMSE)

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

- Mean Absolute Error (MAE)
- Mean Absolute Percent Error (MAPE)
- Root Mean Squared Logarithmic Error (RMSLE)

The ML Process: Model Evaluation Metric

Classification

- Classification Accuracy

- Precision = TP/(TP+FP)
- Sensitivity (Recall, TPR) = TP/(TP+FN)
- Specificity (1 – FPR) = TN/(TN+FP)

	Predicted events	Predicted non-events
Actual events	True Positive	False Negative
Actual non-events	False Positive	True Negative

The ML Process: Model Evaluation Metric

Classification

- Classification Accuracy = (TP+TN)/Total = (100+50)/165 ≅ 0.91

- Precision = TP/(TP+FP) = 100/(100+10) ≅ 0.91
- Sensitivity (Recall, TPR) = TP/(TP+FN) = 100/(100+5) ≅ 0.95
- Specificity (1 – FPR) = TN/(TN+FP) = 50/(50+10) ≅ 0.83

	Predicted events	Predicted non-events
Actual events	True Positive	False Negative
Actual non-events	False Positive	True Negative

	Predicted events	Predicted non-events
Actual events	100	5
Actual non-events	10	50

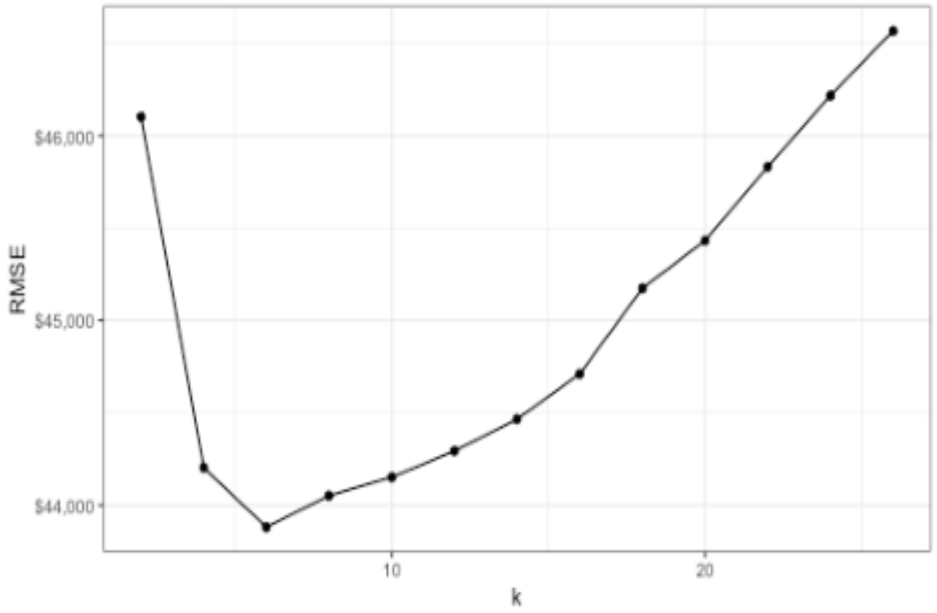
Together

(Illustrated with KNN method)

1. Split into training vs testing data
2. Specify a resampling procedure
3. Create our hyperparameter grid
4. Execute grid search
5. Evaluate performance

```
# 1. stratified sampling with the rsample package
set.seed(123)
split <- initial_split(ames, prop = 0.7, strata = "Sale_Price")
ames_train <- training(split)
ames_test <- testing(split)
# 2. create a resampling method
cv <- trainControl(
     method = "repeatedcv",
     number = 10,
     repeats = 5
)
# 3. create a hyperparameter grid search
hyper_grid <- expand.grid(k = seq(2, 26, by = 2))
# 4. execute grid search with knn model
# use RMSE as preferred metric
knn_fit <- train(
     Sale_Price ~ .,
     data = ames_train,
     method = "knn",
     trControl = cv,
     tuneGrid = hyper_grid,
     metric = "RMSE"
     )
```

Together




```
# 5. evaluate results
# print model results
knn_fit
## k-Nearest Neighbors
##
## 2054 samples
## 80 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1848, 1850, 1848, 1848, 1848, 1848, ...
## Resampling results across tuning parameters:
##
## k RMSE Rsquared MAE
## 2 46100.84 0.6618945 30205.06
## 4 44203.37 0.6875495 28696.28
## 6 43881.20 0.6960556 28476.22
## 8 44051.92 0.6975777 28510.18
## The final value used for the model was k = 6.
```

RMSE: ~44,000, i.e. on avg, our model mispredicts the expected home price by $44,000