CMSC/LING/STAT 208: Machine Learning

Abhishek Chakraborty [Much of the content in these slides have been adapted from *An Introduction to Statistical Learning: with Applications in R*, James et al.]

Classification Problems

- Response Y is qualitative (categorical).
- The objective is to build a classifier $\hat{Y} = \hat{C}(\mathbf{X})$ that assigns a class label to a future unlabeled (unseen) observation and understand the relationship between the predictors and response.
- There can be two types of predictions based on the research problem.
  - Class probabilities
  - Class labels

Classification Problems: Example

Default dataset

```
library(ISLR2)   # load library
data("Default")   # load dataset
```

```
head(Default)   # print first six observations
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

```
table(Default$default)   # class frequencies
```

```
##
##   No  Yes
## 9667  333
```

We will consider **`default`** as the response variable.

K-Nearest Neighbors Classifier

Given a value for K and a test data point x_0,

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} I(y_i = j)$$

where $\mathcal{N}_0$ is known as the **neighborhood** of x_0.

For classification problems, the predictions are obtained in terms of **majority vote** (unlike in regression where predictions are obtained by averaging).

K-Nearest Neighbors Classifier: Build Model

Default dataset

response (Y): `default` and predictor (X): `balance`

```
library(caret)   # load package 'caret'

knnfit <- knn3(default ~ balance, data = Default, k = 10)   # fit 10-nn model
```

K-Nearest Neighbors Classifier: Predictions

Default dataset

- One can directly obtain the class label predictions as below.

```
knn_class_preds_1 <- predict(knnfit, newdata = Default, type = "class")   # obtain class label predictions directly

head(knn_class_preds_1)   # predicted class labels for first six observations
```

```
## [1] No No No No No No
## Levels: No Yes
```

- Otherwise, one can first obtain predictions in terms of probabilities and then convert them into class label predictions based on a threshold.

```
knn_prob_preds <- predict(knnfit, newdata = Default, type = "prob")   # obtain predictions as probabilities

head(knn_prob_preds)   # predicted probabilities for first six observations
```

```
##      No Yes
## [1,]  1   0
## [2,]  1   0
## [3,]  1   0
## [4,]  1   0
## [5,]  1   0
## [6,]  1   0
```

```
threshold <- 0.5   # set threshold

knn_class_preds_2 <- factor(ifelse(knn_prob_preds[,2] > threshold, "Yes", "No"))   # obtain predictions as class labels
```

K-Nearest Neighbors Classifier: Performance

Default dataset

```
# use the following code only when all predictions are from the same class
# levels(knn_class_preds_1) = c("No", "Yes")

confusionMatrix(data = knn_class_preds_1, reference = Default$default, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  9620  213
##        Yes   47  120
##
##                Accuracy : 0.974
##                  95% CI : (0.9707, 0.977)
##     No Information Rate : 0.9667
##     P-Value [Acc > NIR] : 1.406e-05
##
##                   Kappa : 0.4682
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3604
##             Specificity : 0.9951
##          Pos Pred Value : 0.7186
##          Neg Pred Value : 0.9783
##              Prevalence : 0.0333
##          Detection Rate : 0.0120
##    Detection Prevalence : 0.0167
##       Balanced Accuracy : 0.6777
##
##        'Positive' Class : Yes
##
```

Confusion Matrix Terms

	Actual Positive/Event	Actual Negative/Non-event
Predicted Positive/Event		
Predicted Negative/non-event		

ROC Curve and AUC

The **ROC (Receiver Operating Characteristics)** curve is a popular graphic for comparing different classifiers across all possible thresholds. The ROC curve plots the Specificity (1-false positive rate) along the x-axis and the Sensitivity (true positive rate) along the y-axis.

Another popular metric for comparing classifiers is the **AUC (Area Under the ROC Curve)**. An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier.

ROC Curve and AUC

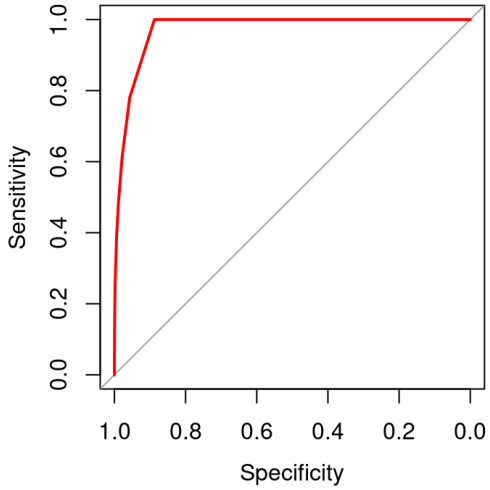
Default dataset

```
library(pROC)   # load library

# create object for ROC curve for KNN fit

roc_object_knn <- roc(response = Default$default, predictor = knn_prob_preds[,2])

plot(roc_object_knn, col = "red")   # plot ROC curve
```




```
auc(roc_object_knn)   # obtain AUC
```

```
## Area under the curve: 0.9739
```

Classification Problems

For some algorithms, we might need to convert the categorical response to numeric (0/1) values.

Default dataset

```
Default$default_id <- ifelse(Default$default == "Yes", 1, 0)   # create 0/1 variable

head(Default, 10)   # print first ten observations
```

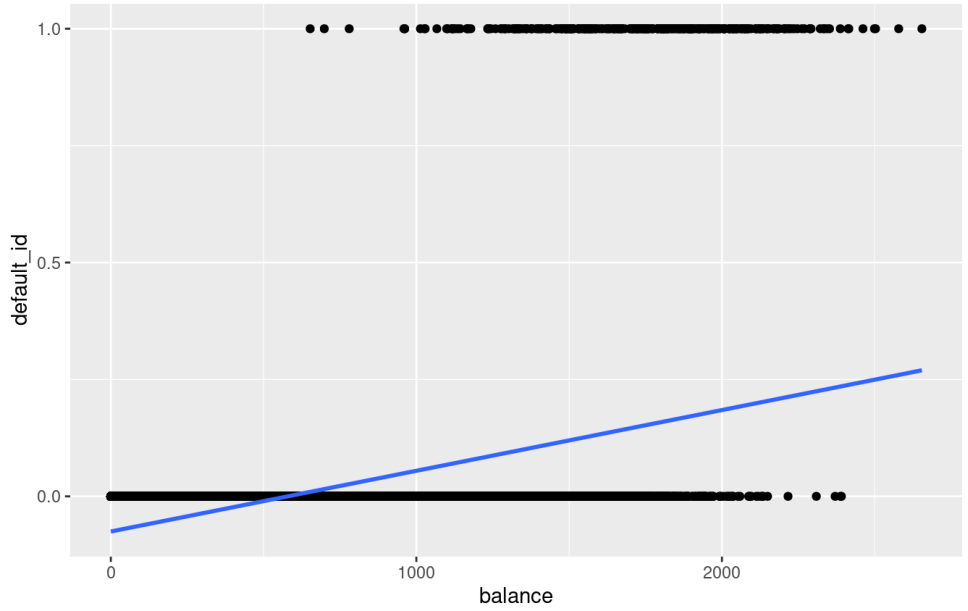
```
##    default student   balance    income default_id
## 1       No      No  729.5265 44361.625          0
## 2       No     Yes  817.1804 12106.135          0
## 3       No      No 1073.5492 31767.139          0
## 4       No      No  529.2506 35704.494          0
## 5       No      No  785.6559 38463.496          0
## 6       No     Yes  919.5885  7491.559          0
## 7       No      No  825.5133 24905.227          0
## 8       No     Yes  808.6675 17600.451          0
## 9       No      No 1161.0579 37468.529          0
## 10      No      No    0.0000 29275.268          0
```

Why Not Linear Regression?

Default dataset

```
lrfit <- lm(default_id ~ balance, data = Default)   # fit SLR
summary(lrfit$fitted.values)   # summary of y_hats
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.07519 -0.01263  0.03178  0.03330  0.07628  0.26953
```



Linear regression does not model probabilities well. Linear regression might produce probabilities less than zero or bigger than one.

Why Not Linear Regression?

Suppose we have a response Y,

$$Y = \begin{cases} 1; & \text{if stroke} \\ 2; & \text{if drug overdose} \\ 3; & \text{if epileptic seizure} \end{cases}$$

Linear regression suggests an ordering, and in fact implies that the difference between stroke and drug overdose is the same as between drug overdose and epileptic seizure.

Logistic Regression

Consider a one-dimensional two-class problem.

- Transform the linear model $\beta_0 + \beta_1\ X$ so that the output is a probability.
- Use **logistic** or **sigmoid** function

$$g(t) = \frac{e^t}{1 + e^t} \quad \text{for } t \in \mathcal{R}$$

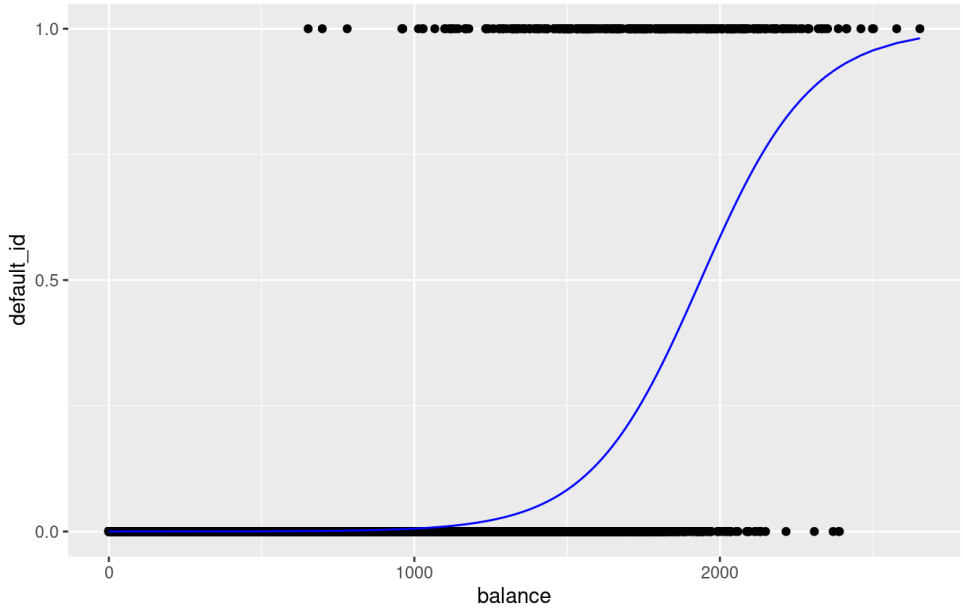
- Suppose $p(X) = P(Y = 1|X)$. Then,

$$p(X) = g\left(\beta_0 + \beta_1\ X\right) = \frac{e^{\beta_0 + \beta_1\ X}}{1 + e^{\beta_0 + \beta_1\ X}}$$

- $e \approx 2.71828$ is a mathematical constant (Euler's number).
- $\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1\ X$

Logistic Regression

Default dataset



Logistic Regression Implementation (Estimating Parameters)

Default dataset

A logistic regression model with `default` as the response and `balance` as the predictor.

```
logregfit <- glm(default ~ balance, data = Default, family = binomial)   # fit logistic regression model
summary(logregfit)   # obtain results
```

```
##
## Call:
## glm(formula = default ~ balance, family = binomial, data = Default)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
## balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

Logistic Regression Implementation (Predictions)

Default dataset

For `balance`=$700,

$$\hat{p}(X) = \frac{e^{b_0+b_1X}}{1+e^{b_0+b_1X}} = \frac{e^{-10.65+(0.005499\times 700)}}{1+e^{-10.65+(0.005499\times 700)}} = 0.0011$$

```
predict(logregfit, newdata = data.frame(balance = 700))   # obtain log-odds predictions
```

```
##        1
## -6.802089
```

```
predict(logregfit, newdata = data.frame(balance = 700), type = "response")   # obtain probability predictions
```

```
##          1
## 0.001110217
```

Interpretations: A positive $\hat{\beta}_1$ indicates that a higher balance is associated with a larger log odds of default, in other words, a higher probability of default. However, the change in the probability of default due to a one-unit change in balance depends on the current balance value.

Logistic Regression Implementation (Predictions)

Default dataset

To predict **probabilities** for all observations, we use

```
logreg_prob_preds <- predict(logregfit, newdata = Default, type = "response")   # obtain probability predictions

head(logreg_prob_preds)   # predicted probabilities for first six observations
```

```
##           1           2           3           4           5           6
## 0.0013056797 0.0021125949 0.0085947405 0.0004344368 0.0017769574 0.0037041528
```

Set a threshold to obtain predicted **class labels**. The following uses a threshold of 0.5.

```
threshold <- 0.5   # set threshold

logreg_class_preds <- factor(ifelse(logreg_prob_preds > threshold, "Yes", "No"))   # obtain class predictions

head(logreg_class_preds)   # predicted class labels for first six observations
```

```
##  1  2  3  4  5  6
## No No No No No No
## Levels: No Yes
```

Logistic Regression Implementation (Assessing Performance)

Default dataset

```
# use the following code only when all predictions are from the same class
# levels(logreg_class_preds) = c("No", "Yes")

confusionMatrix(data = logreg_class_preds, reference = Default$default, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  9625  233
##        Yes   42  100
##
##                Accuracy : 0.9725
##                  95% CI : (0.9691, 0.9756)
##     No Information Rate : 0.9667
##     P-Value [Acc > NIR] : 0.0004973
##
##                   Kappa : 0.4093
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3003
##             Specificity : 0.9957
##          Pos Pred Value : 0.7042
##          Neg Pred Value : 0.9764
##              Prevalence : 0.0333
##          Detection Rate : 0.0100
##    Detection Prevalence : 0.0142
##       Balanced Accuracy : 0.6480
##
##        'Positive' Class : Yes
##
```

ROC Curve and AUC

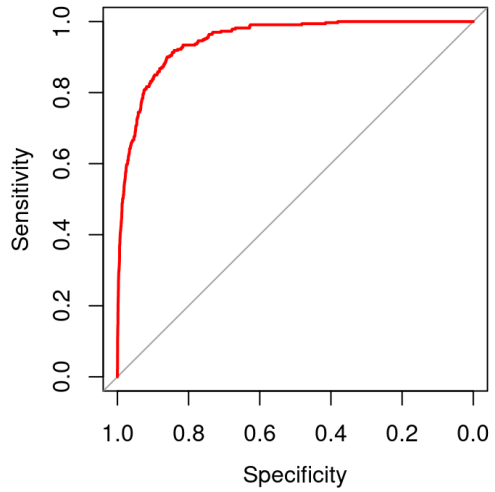
Default dataset

```
library(pROC)   # load library

# create object for ROC curve

roc_object <- roc(response = Default$default, predictor = logreg_prob_preds)

plot(roc_object, col = "red")   # plot ROC curve
```




```
auc(roc_object)   # obtain AUC
```

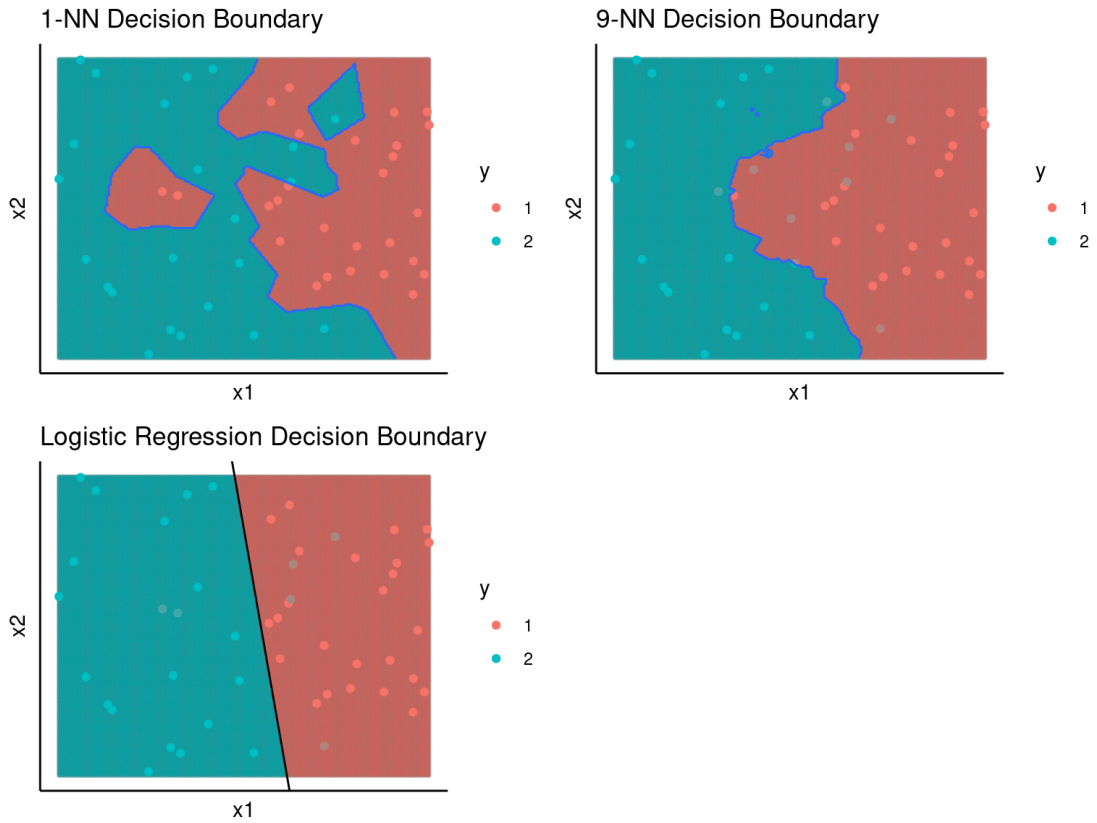
```
## Area under the curve: 0.948
```

Logistic Regression vs K-Nearest Neighbors

- Logistic regression is a parametric approach (with restrictive assumptions), KNN is non-parametric.
- Logistic regression works only for classification problems (Y categorical), KNN can be used for both regression and classification.
- Logistic regression is interpretable, KNN is not.
- Logistic regression can accommodate qualitative predictors and can be extended to include interaction terms as well. Using Euclidean distance with KNN does not allow for qualitative predictors.
- In terms of prediction, KNN can be pretty good for small p, that is, $p \leq 4$ and large n. Performance of KNN deteriorates as p increases - curse of dimensionality.

Logistic Regression vs K-Nearest Neighbors

Simulated 50 observations



The Machine Learning Process

For the next 2-3 class periods, we are going to discuss the overall ML process. Note that, the procedures we will discuss are not ML models, rather the models go through this process to obtain the best (optimal) predictive model.

Ames Housing dataset

```
ames <- readRDS("AmesHousing.rds")   # load dataset
```

```
# we won't use the entire dataset now (that's for later)
# select the variables to work with for this class (04/18)

ames <- ames %>% select(Sale_Price, Garage_Area, Year_Built)
```

Exploratory Data Analysis

Ames Housing dataset

```
summary(ames)   # summary of the variables
```

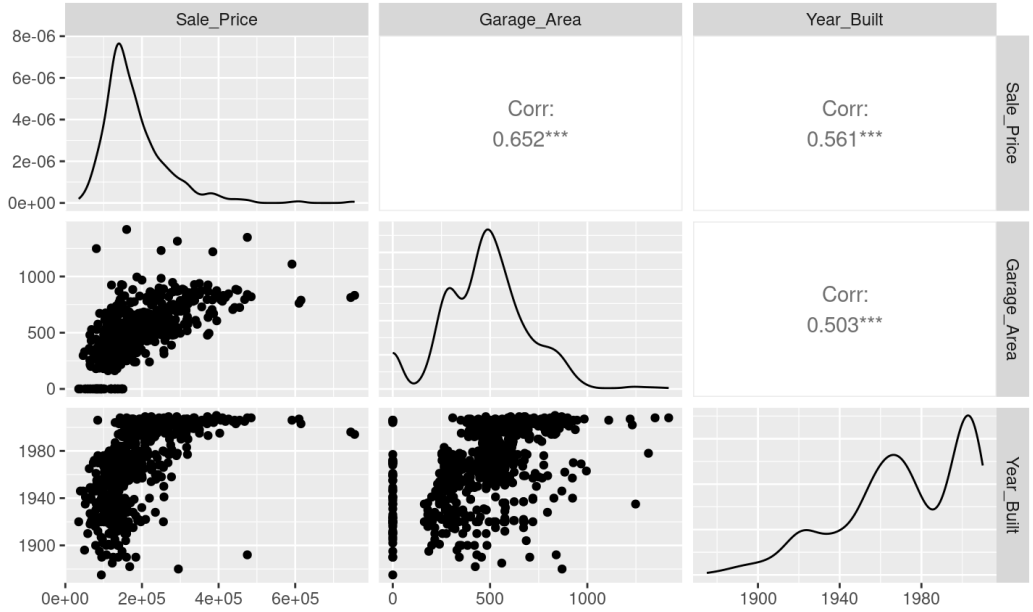
```
##    Sale_Price      Garage_Area       Year_Built
##  Min.   : 34900   Min.   :   0.0   Min.   :1875
##  1st Qu.:129500   1st Qu.: 324.0   1st Qu.:1954
##  Median :160000   Median : 480.0   Median :1972
##  Mean   :181115   Mean   : 476.5   Mean   :1971
##  3rd Qu.:213500   3rd Qu.: 592.0   3rd Qu.:2000
##  Max.   :755000   Max.   :1418.0   Max.   :2010
```

Exploratory Data Analysis

Ames Housing dataset

```
library(GGally)

ggpairs(ames)   # correlation plot
```



Data Splitting

Available data split into **training** and **test** datasets.

- **Training set:** these data are used to develop feature sets, train our algorithms, tune hyperparameters, compare models, and all of the other activities required to choose an optimal model.
- **Test set:** having chosen a final optimal model, these data are used to obtain an unbiased estimate of the model's performance.

It is critical that the test set not be used prior to selecting your final model. Assessing results on the test set prior to final model selection biases the model selection process since the testing data will have become part of the model development process.

Data Splitting

```
# split data

set.seed(041824)  # fix the random number generator for reproducibility

library(caret)  # load library

# split available data into 80% training and 20% test datasets
train_index <- createDataPartition(y = ames$Sale_Price, p = 0.8, list = FALSE)

ames_train <- ames[train_index,]   # training data

ames_test <- ames[-train_index,]   # test data
```

Resampling Methods

- **Idea:** Repeatedly draw samples from the training data and refit a model on each sample, and evaluate its performance on the other parts.
- **Objective:** To obtain additional information about the fitted model.
- **Cross-Validation (CV)** is probably the most widely used resampling method. It is a general approach that can be applied to almost any statistical learning method.

Cross-Validation (CV)

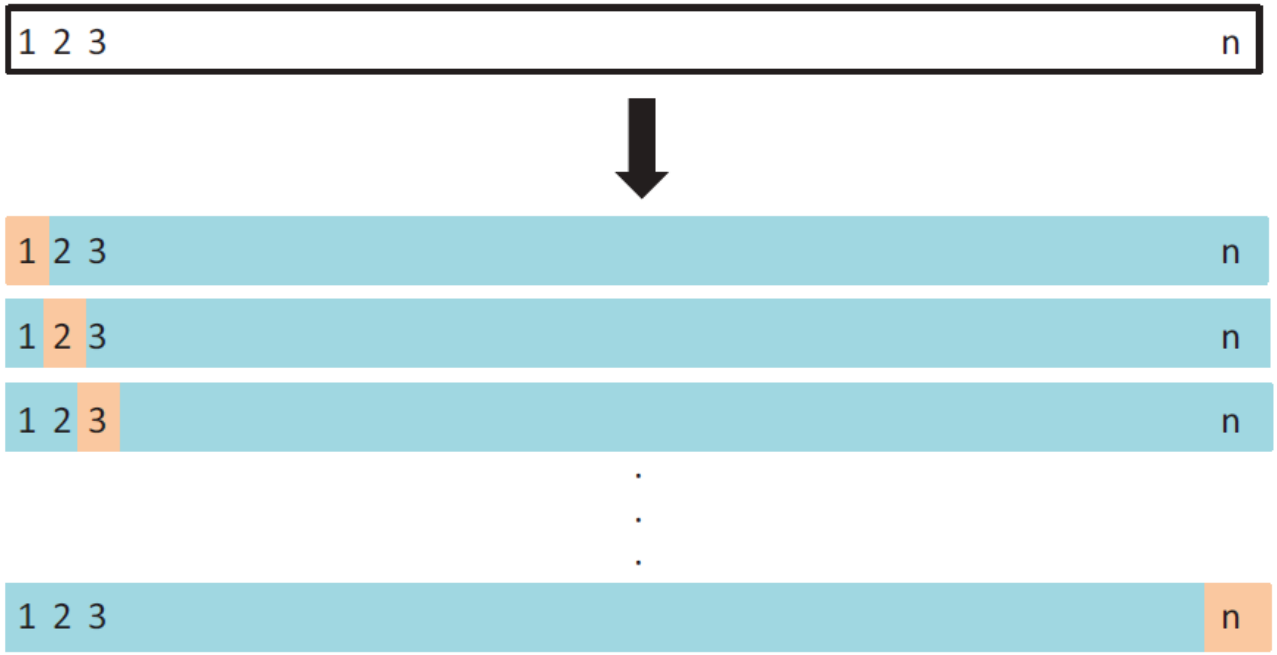
Used for

- **model selection**: select the optimum level of flexibility (tune hyperparameters) or compare different models to choose the best one
- **model assessment**: evaluate the performance of a model (estimate the error and variability)

We will talk about

- Leave-One-Out Cross-Validation (LOOCV)
- k-Fold Cross-Validation

Leave-One-Out Cross-Validation (LOOCV)



Your Turn!!!

Suppose you implement LOOCV on a dataset with $n = 100$ observations.

1. What is the size (number of observations) of each training set?
2. What is the size of each validation set?
3. How many steps/iterations are required to complete the overall LOOCV process?

Leave-One-Out Cross-Validation (LOOCV)

Advantages

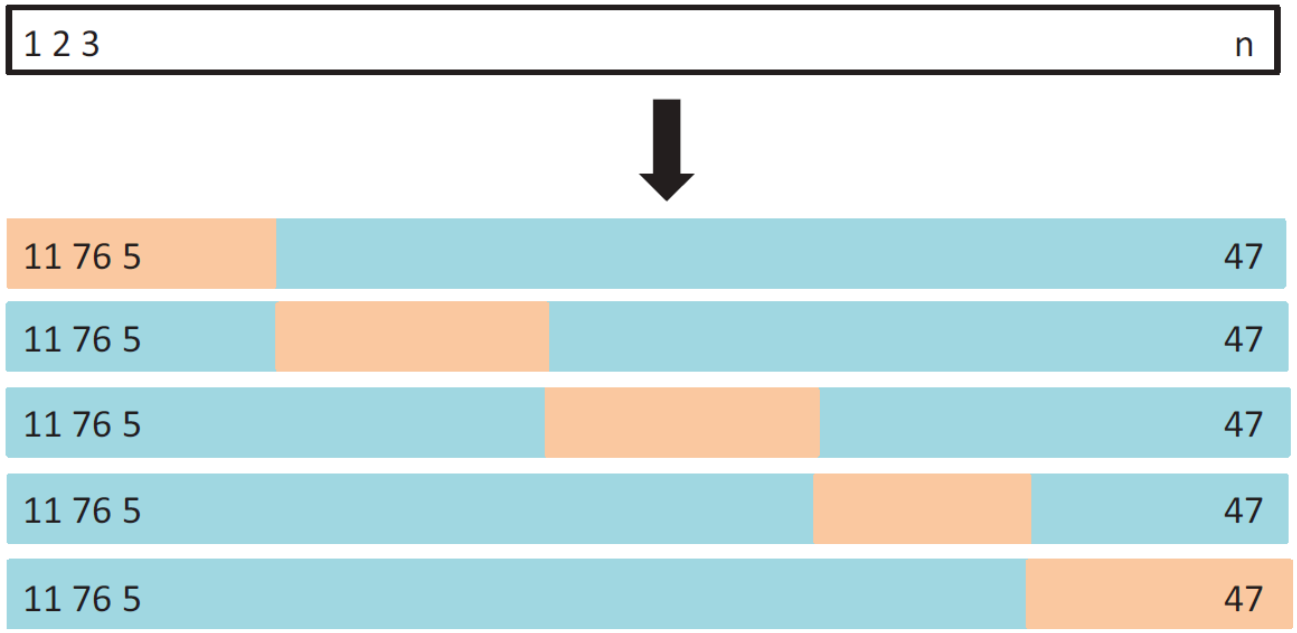
- LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations, which is almost as many as the number of observations in the full training dataset.
- Performing LOOCV multiple times will always yield the same results.

Disadvantages

- Can be potentially expensive to implement, specially for large n.
- LOOCV error estimate can have high variance.

k-Fold Cross-Validation

- Randomly divide the training data into k groups or **folds** (approximately equal size).
- Consider one of these folds as the validation set. Fit the model on the remaining $k-1$ folds combined, and obtain predictions for the k^{th} fold. Repeat for all k folds.



Your Turn!!!

Suppose you implement 10-fold CV on a dataset with $n = 100$ observations.

1. What is the size (number of observations) of each training set?
2. What is the size of each validation set?
3. How many steps/iterations are required to complete the overall CV process?

k-Fold Cross-Validation: Implementation

Ames Housing dataset

Consider `Sale_Price` as the response variable. We will compare the following three linear regression models:

- with `Garage_Area` as the only predictor;
- with `Year_Built` as the only predictor;
- with `Garage_Area` and `Year_Built` as predictors.

k-Fold Cross-Validation: Implementation

Ames Housing dataset

Define CV specifications.

```
cv_specs <- trainControl(method = "repeatedcv",   # CV method
                         number = 10,    # number of folds
                         repeats = 5)     # number of repeats
```

k-Fold Cross-Validation: Implementation

Ames Housing dataset - Implement 10-fold CV with the first model.

```
set.seed(041824)

m1 <- train(form = Sale_Price ~ Garage_Area,    # specify model
            data = ames_train,   # specify dataset
            method = "lm",       # specify type of model
            trControl = cv_specs, # CV specifications
            metric = "RMSE")   # metric to evaluate model

m1   # summary of CV
```

```
## Linear Regression
##
## 706 samples
##   1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 635, 635, 636, 635, 636, 634, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   58987.5  0.4596152  41932.08
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
m1$results  # estimate and variability of metrics
```

```
##   intercept    RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1      TRUE 58987.5 0.4596152 41932.08 9600.181 0.09307209 4549.962
```

k-Fold Cross-Validation: Implementation

Ames Housing dataset - Implement 10-fold CV with the second model.

```
set.seed(041824)

m2 <- train(form = Sale_Price ~ Year_Built,
            data = ames_train,
            method = "lm",
            trControl = cv_specs,
            metric = "RMSE")

m2
```

```
## Linear Regression
##
## 706 samples
##   1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 635, 635, 636, 635, 636, 634, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   64091.45  0.3558058  46702.28
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
m2$results
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1      TRUE 64091.45 0.3558058 46702.28 9941.842 0.07605311 4360.542
```

k-Fold Cross-Validation: Implementation

Ames Housing dataset - Implement 10-fold CV with the third model.

```
set.seed(041824)

m3 <- train(form = Sale_Price ~ Garage_Area + Year_Built,
            data = ames_train,
            method = "lm",
            trControl = cv_specs,
            metric = "RMSE")

m3
```

```
## Linear Regression
##
## 706 samples
##   2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 635, 635, 636, 635, 636, 634, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   54166.42  0.5452268  39063.29
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
m3$results
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1      TRUE 54166.42 0.5452268 39063.29 9229.475 0.07930526 4022.956
```

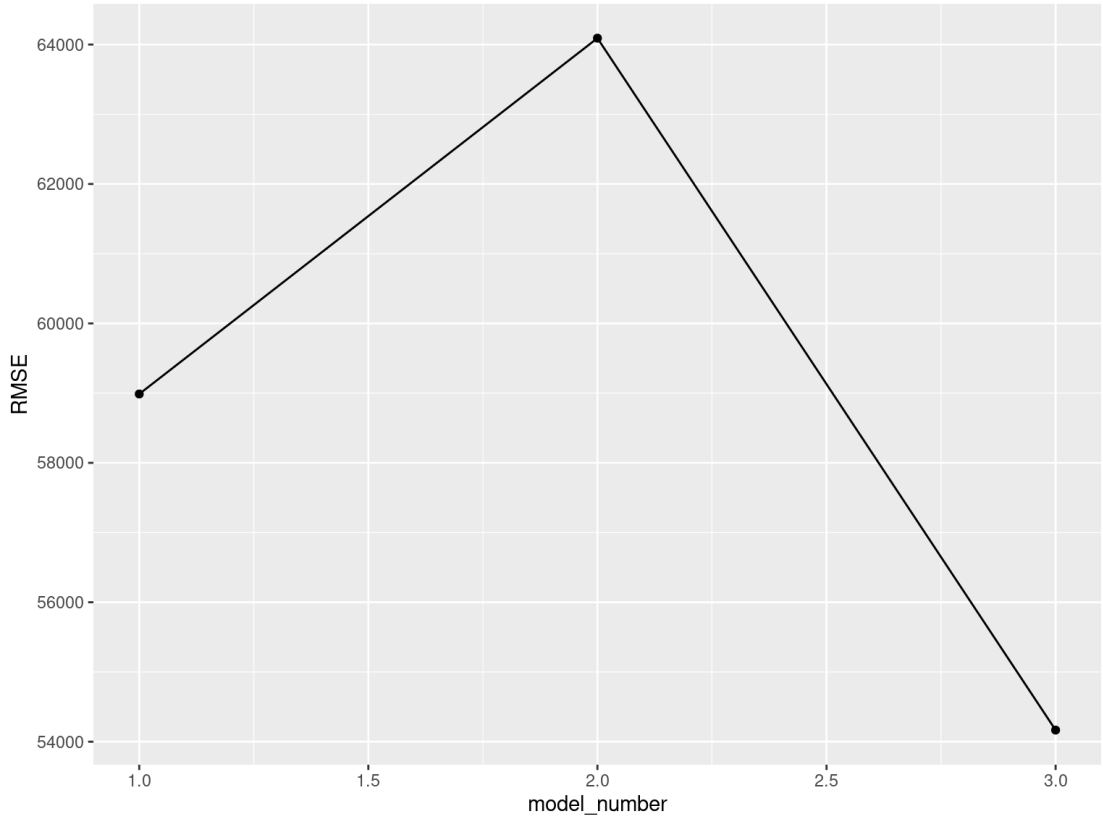
k-Fold Cross-Validation: Results

Ames Housing dataset

Compare CV results for different models.

```
# create data frame to plot results
df <- data.frame(model_number = 1:3, RMSE = c(m1$results$RMSE,
                                              m2$results$RMSE,
                                              m3$results$RMSE))

# plot results from CV
ggplot(data = df, aes(x = model_number, y =  RMSE)) +
  geom_point() + geom_line()
```



Final Model and Prediction Error Estimate

Ames Housing dataset

```
# after choosing final (optimal) model, refit final model using ALL training data

m3$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
## (Intercept)  Garage_Area   Year_Built
##  -1662799.8        177.8        891.8
```

```
# obtain estimate of prediction error from test data

final_model_preds <- predict(m3, newdata = ames_test)   # obtain predictions on test data

pred_error_est <- sqrt(mean((ames_test$Sale_Price - final_model_preds)^2))    # calculate RMSE (estimate of prediction error) from test data

pred_error_est   # test set RMSE
```

```
## [1] 75546.72
```

Bias-Variance Trade-off for LOOCV and k-fold CV

- LOOCV has very less bias. Using $k = 5$ or 10 yields more bias than LOOCV.
- For LOOCV, the error estimates for each fold are highly (positively) correlated. k-fold CV error estimates are somewhat less correlated. LOOCV error estimate has higher variance than k-fold CV error estimate.
- Typically, $k = 5$ or 10 is chosen.