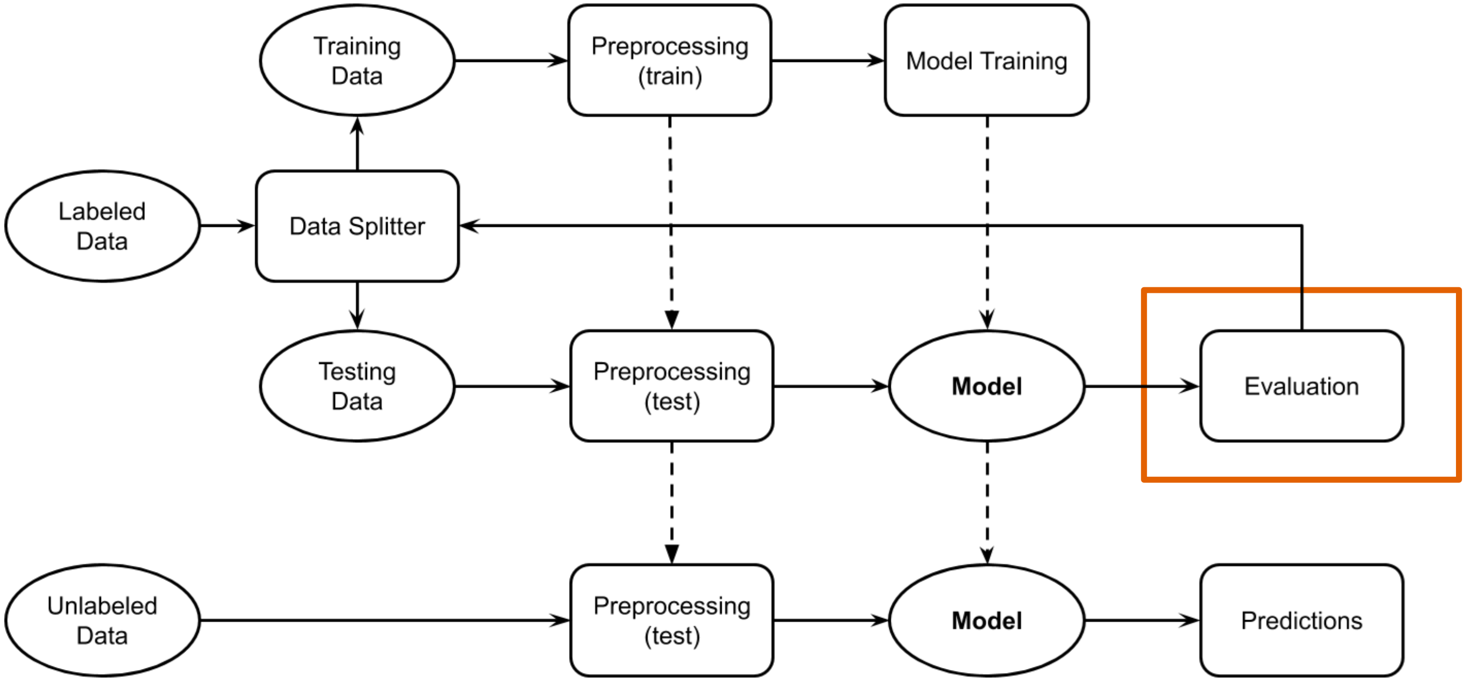
Foundations of Data Science & Analytics: Evaluation and Testing

Ezgi Siir Kibris

Introduction to Data Mining, 2nd Edition
by
Tan, Steinbach, Karpatne, Kumar

Evaluation


Training
Data
Preprocessing
(train)
Model Training
Labeled
Data
Data Splitter
Testing
Data
Preprocessing
(test)
Model
Evaluation
Unlabeled
Data
Preprocessing
(test)
Model
Predictions

Types of Classification Problems

- Binary Classification
- Multi-Class Classification

Binary Classification

Confusion Matrix		Predicted	
		Yes	**No**
Actual	**Yes**	TP	FN
	No	FP	TN

Binary Classification

Confusion Matrix		Predicted	
		Yes	**No**
Actual	**Yes**	5	2
	No	40	1000

Binary Classification

Confusion Matrix		Predicted	
		Yes	**No**
Actual	**Yes**	TP	FN
	No	FP	TN

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Error\ Rate = \frac{FP+FN}{TP+TN+FP+FN}$$

$$= 1 - Accuracy$$

- Pros:
  - Can apply to multi-class
  - A general evaluation
- Cons:
  - Tends to ignore minority classes
  - Need to decide a threshold

Binary Classification

Confusion Matrix		Predicted	
		Yes	**No**
Actual	**Yes**	TP	FN
	No	FP	TN

$$Recall = \frac{TP}{TP+FN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

- Pros:
  - Good indicator for imbalanced classes
- Cons:
  - **Must be used together (precision + recall)**
  - Cannot directly apply to multi-class

Binary Classification

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Confusion Matrix		Predicted	
		Yes	**No**
Actual	**Yes**	5	2
	No	40	1000

Accuracy = 1005/1047 = 0.960

Precision = 5/45 = 0.111

Recall = 5/7 = 0.714

F1 score = 0.192

Multi-Class Classification

$$Accuracy = \frac{correct\ predictions}{all\ data}$$

$$Error\ Rate = \frac{incorrect\ predictions}{all\ data}$$
$$= 1 - Accuracy$$

- Pros:
  - A general evaluation
- Cons:
  - Tend to ignore minority classes
  - Need to decide a threshold

Multi-Class Classification

<table>
<tr><td colspan="2" rowspan="2">One vs. Rest</td><td colspan="2">Predicted</td></tr>
<tr><td>Yes (Ci)</td><td>No (Not Ci)</td></tr>
<tr><td rowspan="2">Actual</td><td>Yes (Ci)</td><td>TP(Ci)</td><td>FN(Ci)</td></tr>
<tr><td>No (Not Ci)</td><td>FP(Ci)</td><td>TN(Ci)</td></tr>
</table>

Multi-Class Classification

$$Precision(C_i) = \frac{TP(C_i)}{TP(C_i)+FP(C_i)}$$

$$Recall(C_i) = \frac{TP(C_i)}{TP(C_i)+FN(C_i)}$$

$$F_1(Ci) = 2 \times \frac{precision(Ci) \times recall(Ci)}{precision(Ci)+recall(Ci)}$$

Multi-Class Classification

		True/Actual		
		Cat (🐱)	Fish (🐠)	Hen (🐔)
Predicted	Cat (🐱)	4	6	3
	Fish (🐠)	1	2	0
	Hen (🐔)	1	2	6

Multi-Class Classification

Class	Precision	Recall	F1-score
Cat	30.8%	66.7%	42.1%
Fish	66.7%	20.0%	30.8%
Hen	66.7%	66.7%	66.7%

Macro-precision = (31% + 67% + 67%) / 3 = 54.7%

Macro-recall = (67% + 20% + 67%) / 3 = 51.1%

Macro-F1 = (42.1% + 30.8% + 66.7%) / 3 = 46.5%

Minority Classes are represented equally in Macro-metrics

Multi-Class Classification

		True/Actual		
		Cat (🐱)	Fish (🐠)	Hen (🐔)
Predicted	Cat (🐱)	4	6	3
	Fish (🐠)	1	2	0
	Hen (🐔)	1	2	6

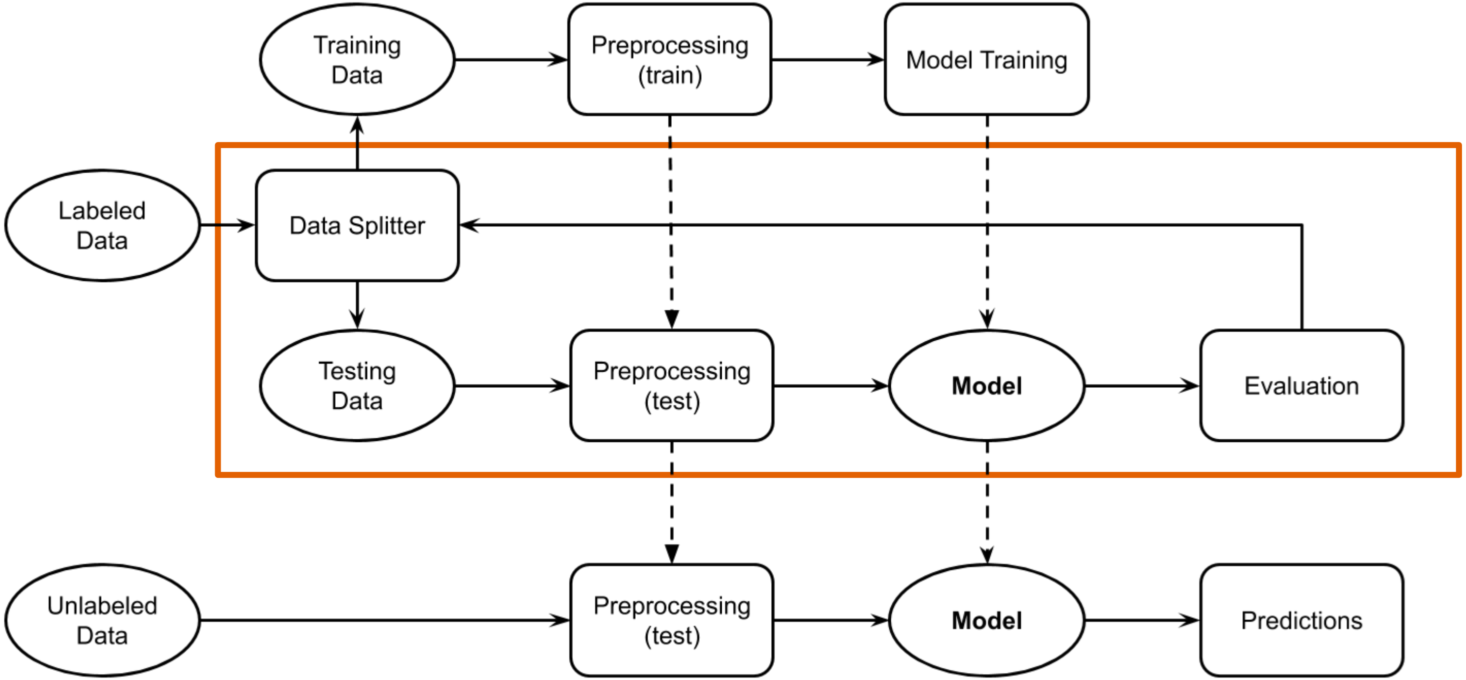
Weighted-F1 = (6 × 42.1% + 10 × 30.8% + 9 × 66.7%) / 25 = 46.4%

Multi-Class Classification

		True/Actual		
		Cat	Fish	Hen
Predicted	Cat	4	6	3
	Fish	1	2	0
	Hen	1	2	6

Micro-F1 = Micro-Precision = Micro-Recall = Accuracy = Green/(Green+Red)=12/25

Testing


Training
Data
Preprocessing
(train)
Model Training
Labeled
Data
Data Splitter
Testing
Data
Preprocessing
(test)
Model
Evaluation
Unlabeled
Data
Preprocessing
(test)
Model
Predictions

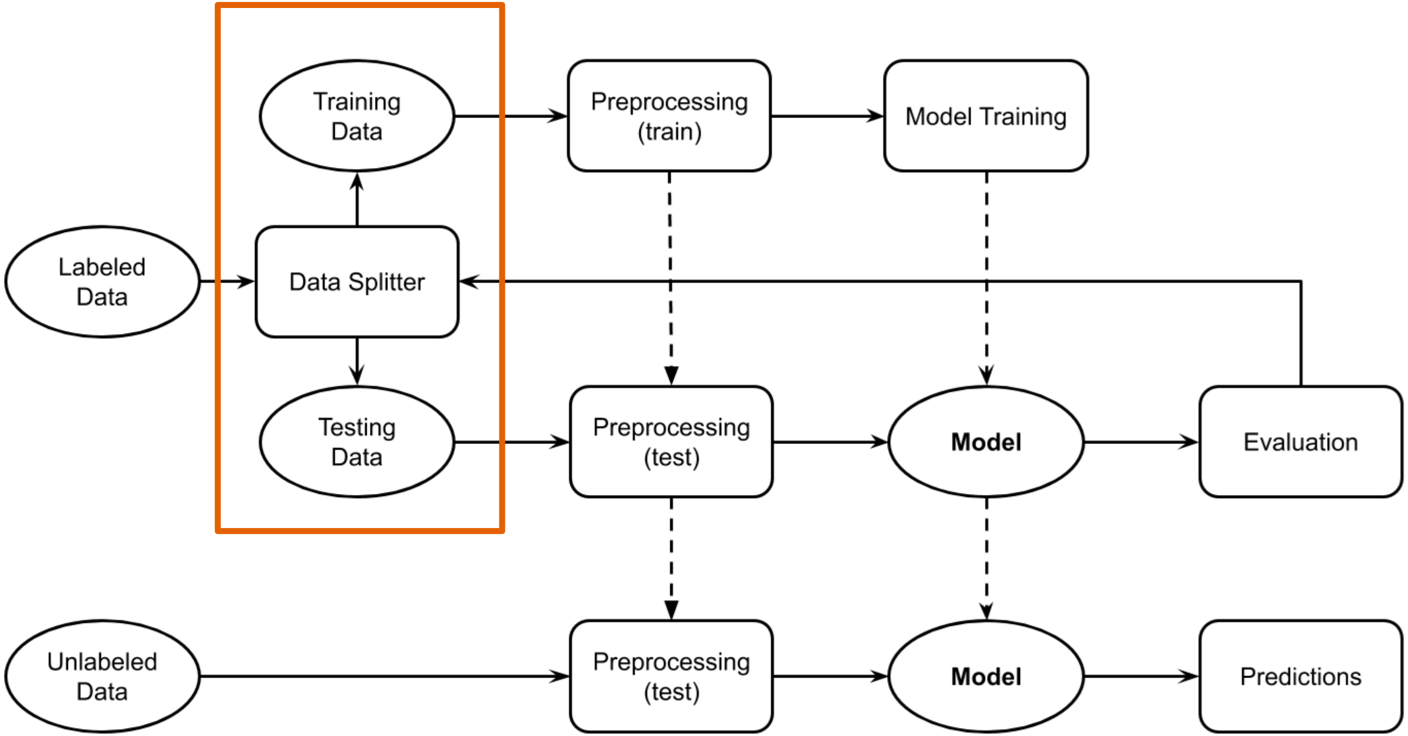
Why is Testing Needed?

- Compare the performance of different models
- Check if performance of a certain model is "good enough" on future (unseen) data

How?

- Reserve a subset of the full data set as the test set
- Train model on rest of the data, then evaluate model predictions on the test set.

How?

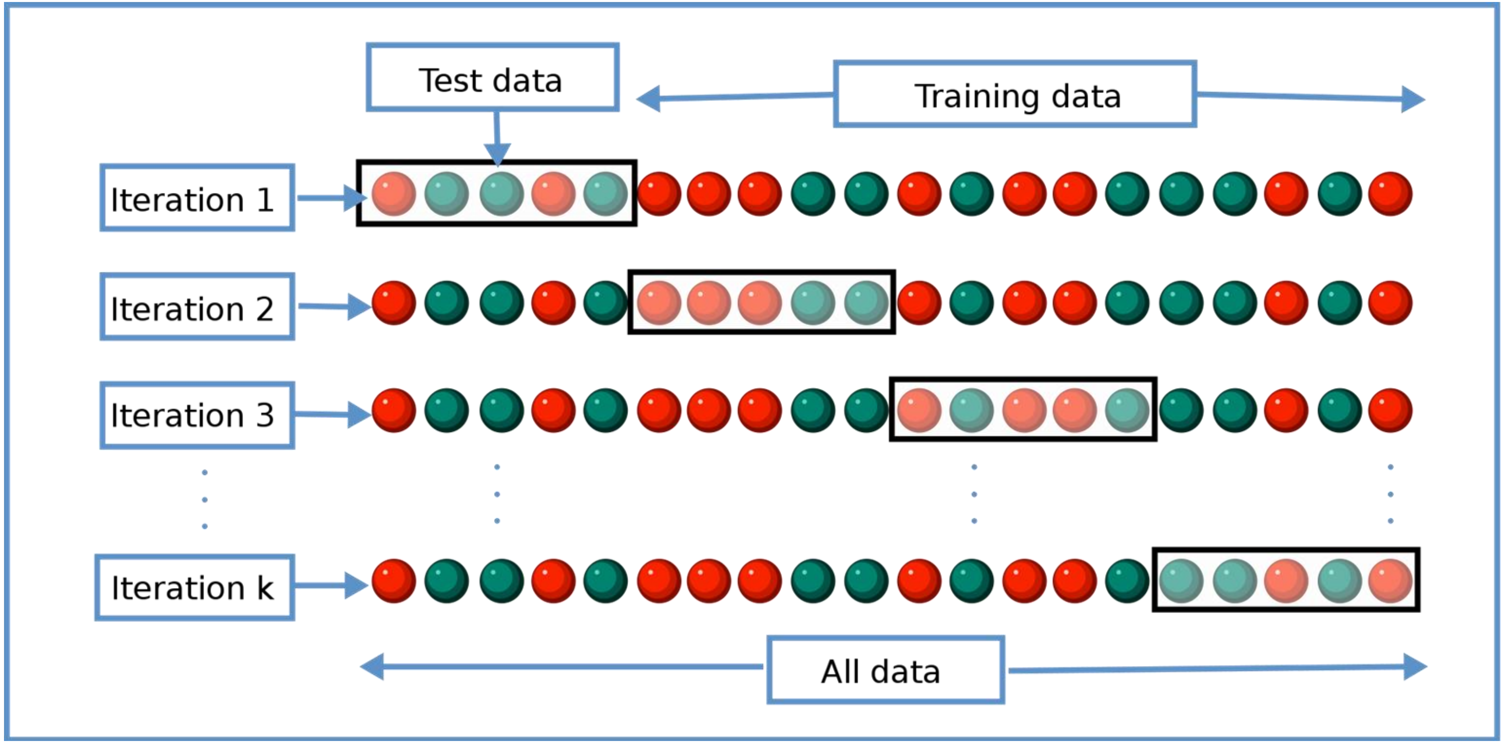

Training Data
Preprocessing (train)
Model Training
Labeled Data
Data Splitter
Testing Data
Preprocessing (test)
Model
Evaluation
Unlabeled Data
Preprocessing (test)
Model
Predictions

Holdout

- The most basic way to create a training and test set is to use “Holdout”
- Split the data using X% for training and Y% for testing
  - Typical splits tend to be 75/25, 80/20, 90/10
  - Splits should use stratified sampling, based upon the class distribution of the full data set
- This process should be repeated multiple times to mitigate the effects of “luck"

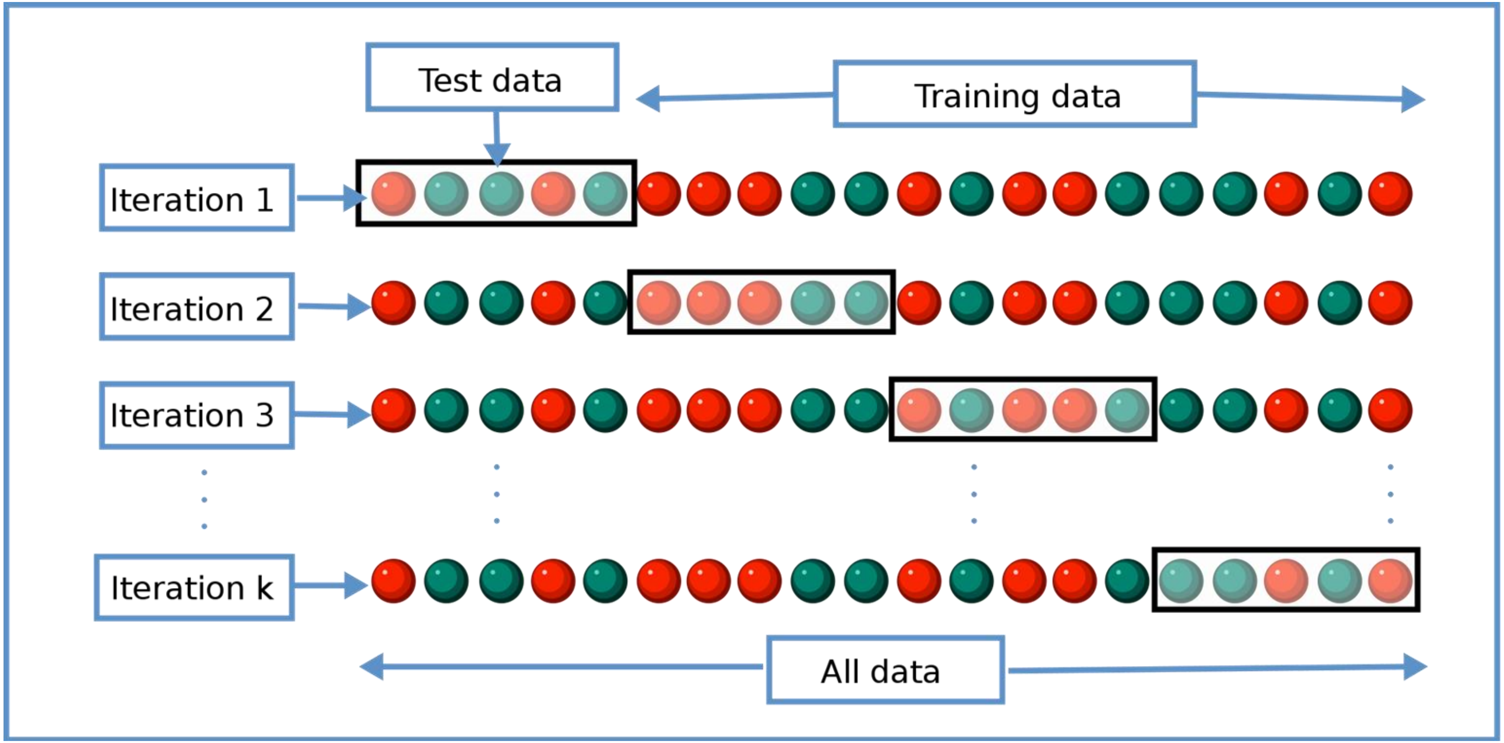
Cross-Validation

Use all data for testing



Cross-Validation

- **K-folds:**
  - split data into K subsets
  - each subset will be used as validation set once
  - performances are averaged across all K subsets



F1 = 0.6

F1 = 0.7

F1 = 0.6

F1 = 0.4

F1_cross = (0.6 + 0.7 + 0.6 + 0.4) / 4 = 0.575

Cross-Validation

- **L × K-fold:**
  - Shuffle data **L** times randomly
    - Each time perform a K-fold cross-validation
    - Collect performance
  - End up with **L** performance metrics

Avoid **lucky** wins

- **Compare the L metrics of each treatment to decide which is the best**