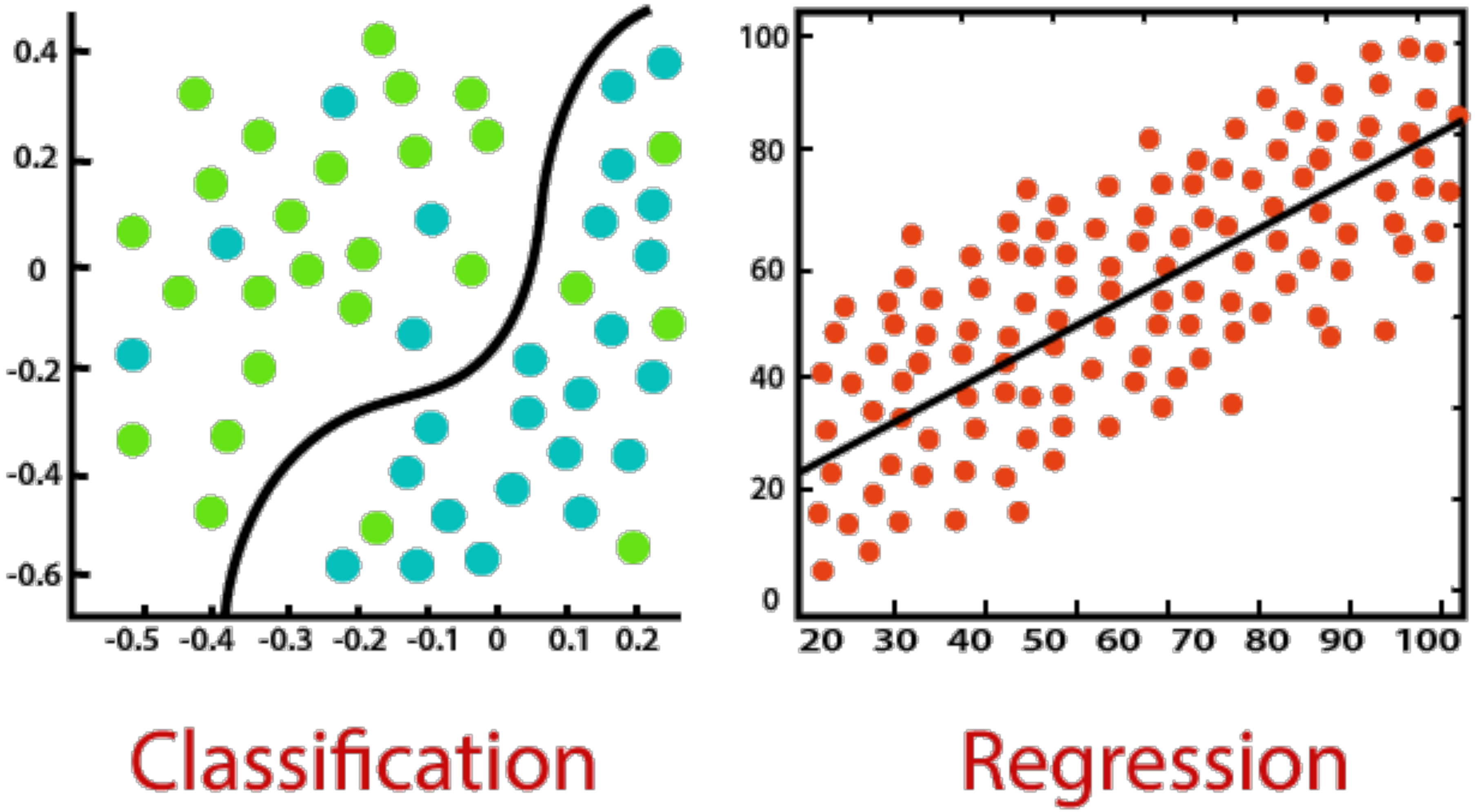
Topic 7
Classification: Part 1

One more thing on Topic Modeling

- Is it a type of clustering?
  - Sometimes LDA is referred to as a type of fuzzy clustering
  - But clustering doesn’t come up much when discussing topic modeling
  - Soft clustering can be used to try and generate topics, but it isn’t quite the same approach as something like LDA, LSA, PLSA, etc.

Classification



https://www.javatpoint.com/regression-vs-classification-in-machine-learning

Pipeline

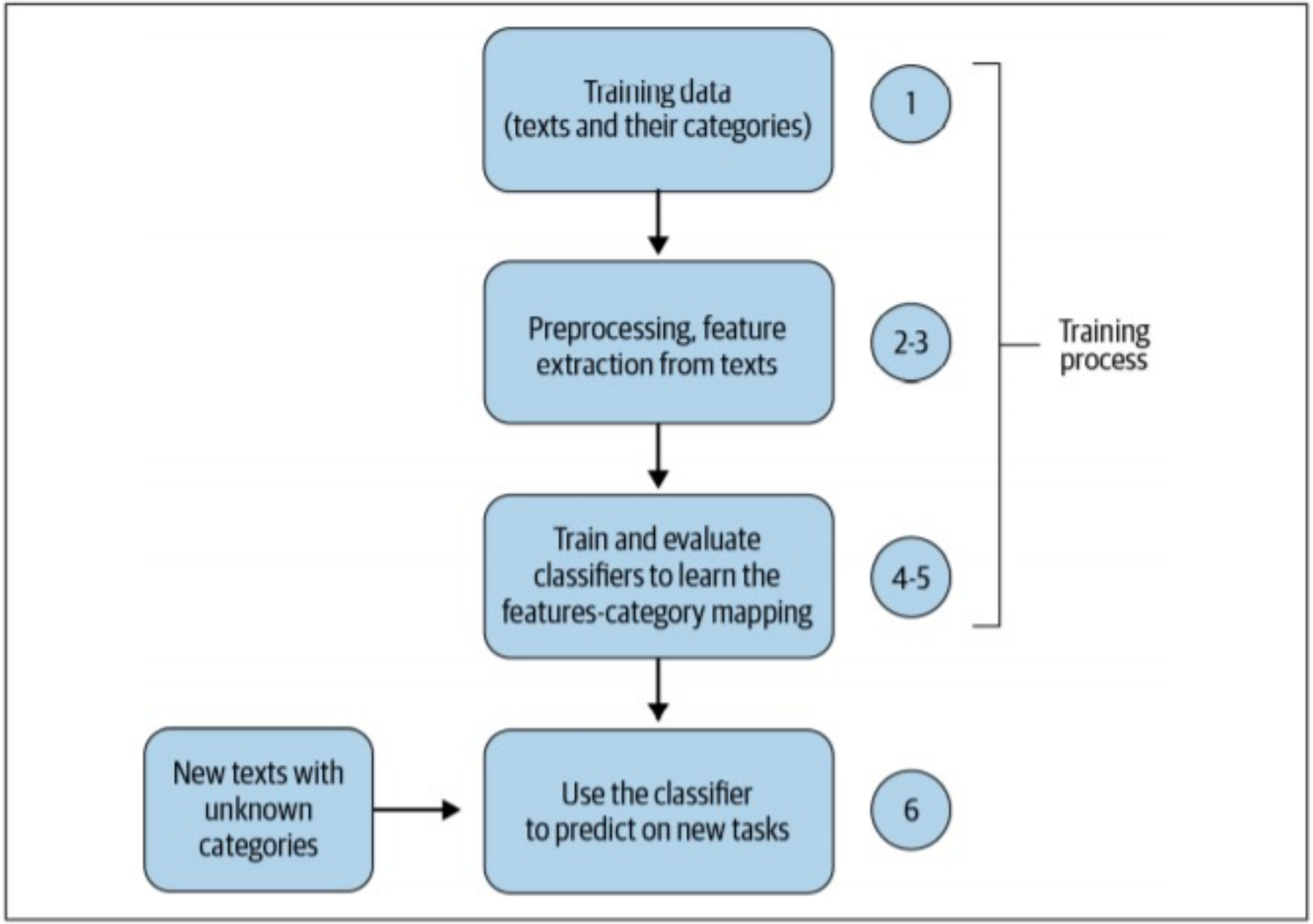
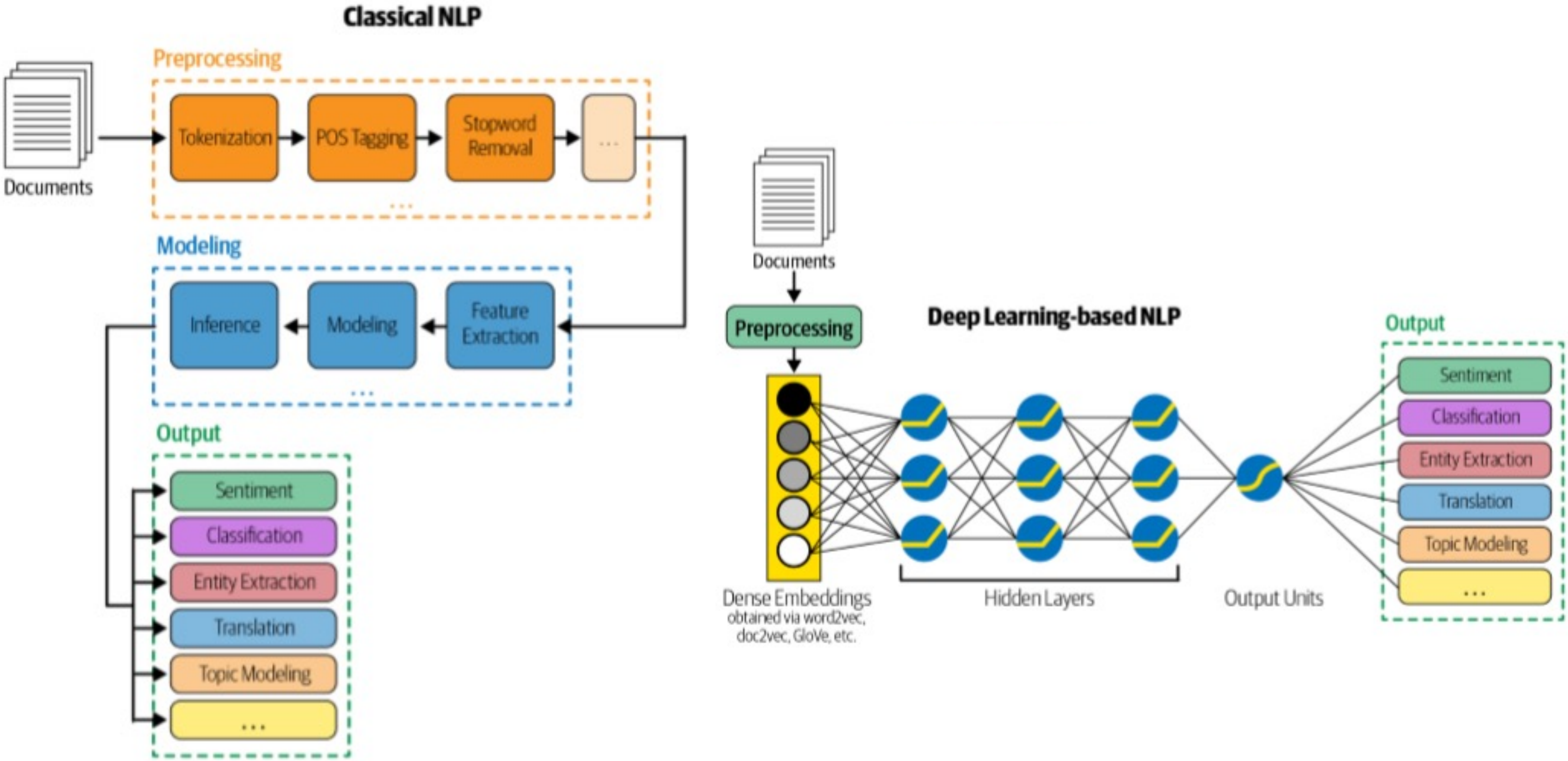


Figure 4-3. Flowchart of a text classification pipeline

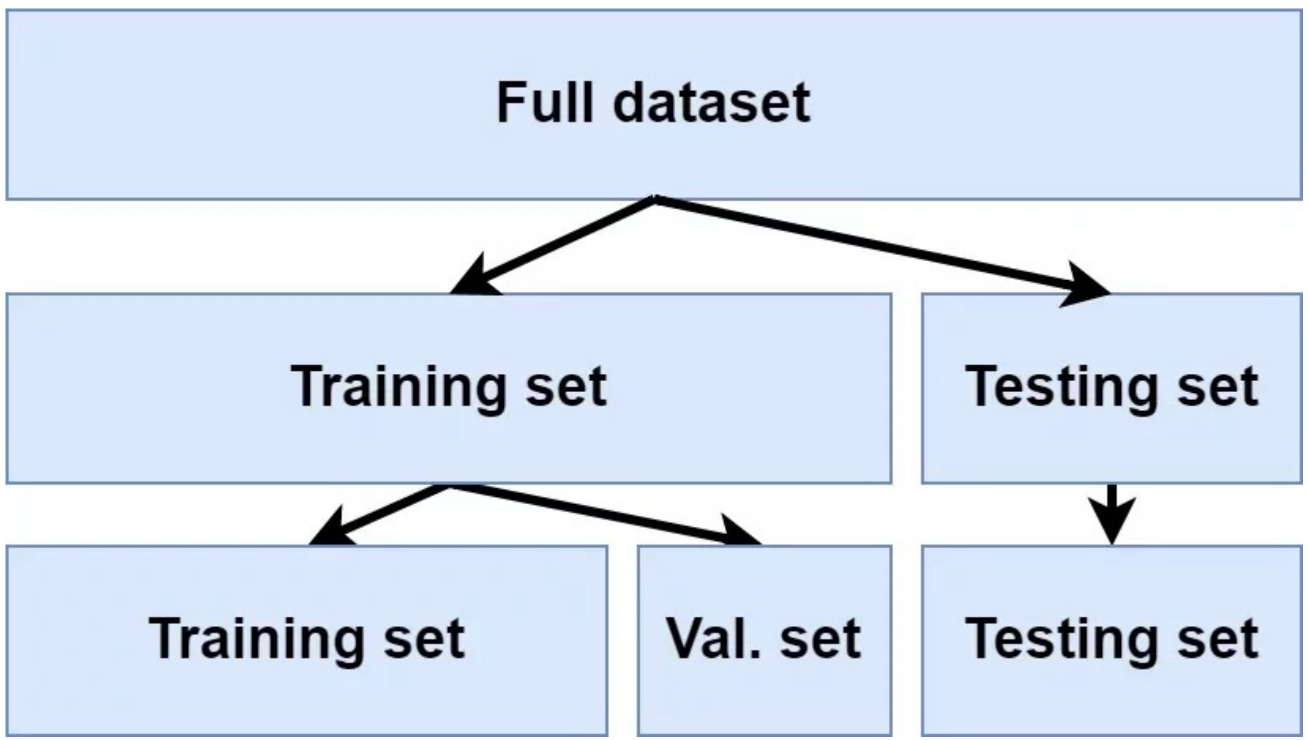


Classical NLP
Preprocessing
Documents
Tokenization
POS Tagging
Stopword Removal
...
...
Modeling
Inference
Modeling
Feature Extraction
...
Output
Sentiment
Classification
Entity Extraction
Translation
Topic Modeling
...
Documents
Preprocessing
Deep Learning-based NLP
Output
Sentiment
Classification
Entity Extraction
Translation
Topic Modeling
...
Dense Embeddings
obtained via word2vec, doc2vec, GloVe, etc.
Hidden Layers
Output Units

Real-world Examples

- Content classification/organization
- Customer service and Ecommerce
- Author attribution
- Language identification
- Medical research

Data splitting



Generalization error and test error

Generalization error:

- The error made by the model when applied to unseen data

Test set & test error:

- The error made by the model when applied to **a set** of unseen data
- **An unbiased estimate** of generalization error

Training error

- The error made by a model when the model is applied to the data in the training set
- **An over-optimistic estimate** of generalization error

Validation set

- A surrogate for test set during model development

Validation error

- The error made by the model when applied to validation set (implicitly seen by the model)
- A biased estimate of generalization error when there are hyperparameters to tune

Irreducible error

- Irreducible error is the lowest achievable prediction error
- It is a characteristic of the dataset/task under study
- It is independent of the model being used
- It often cannot be calculated analytically
- Often we used human error as an estimate (upper bound) for it

Underfitting

How to identify?
Training error is much higher than the **irreducible error**

How to deal with?
Increasing model complexity

Read more:
[Le, William Trung, et al. "Overview of machine learning: part 2: deep learning for medical image analysis." Neuroimaging Clinics 30.4 (2020): 417-431.]

Overfitting

How to identify?
Test error is much higher than the **training error**

How to deal with?

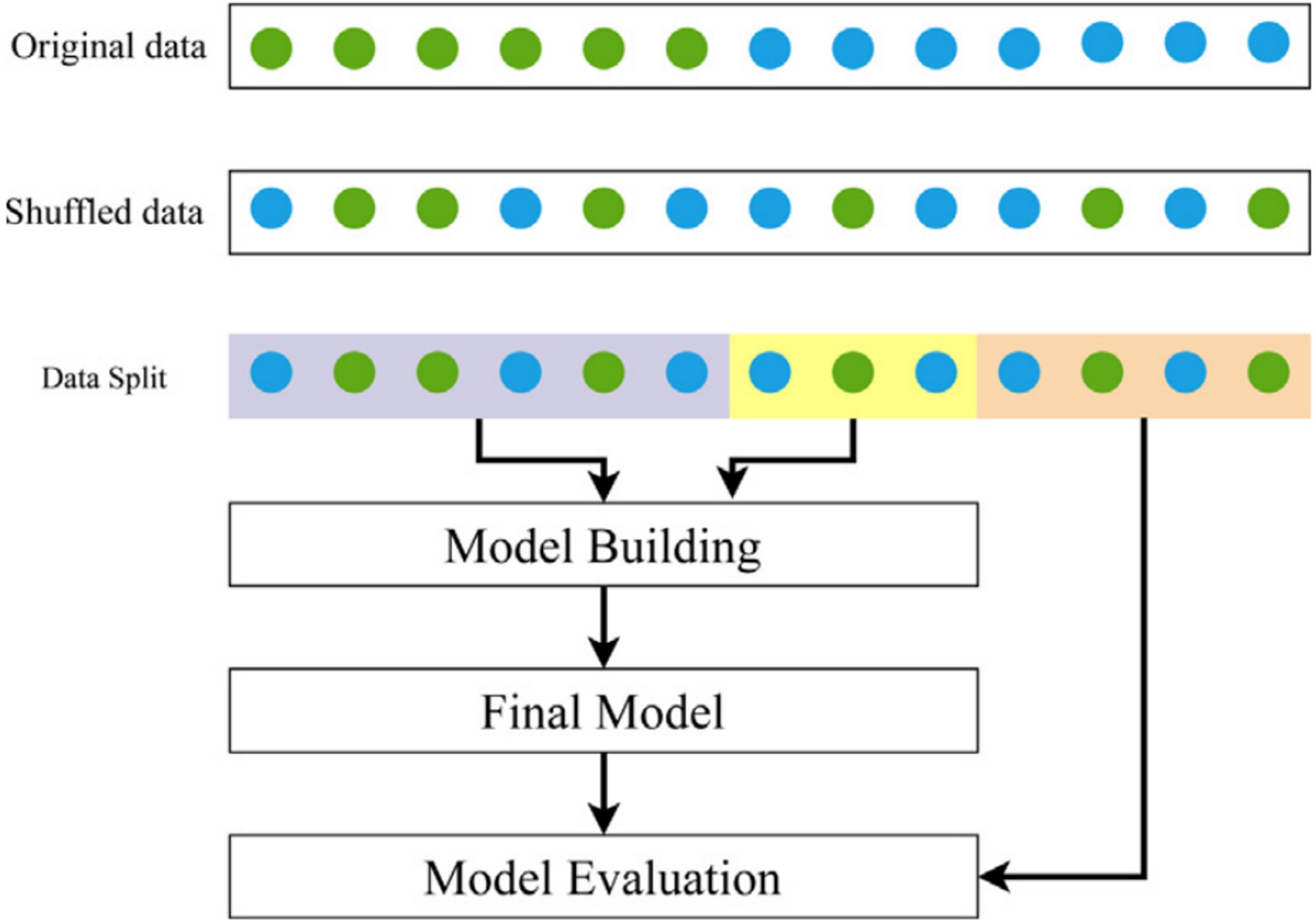
- Decrease model complexity
- Collect more data
- Use data augmentation

Read more:
[Le, William Trung, et al. "Overview of machine learning: part 2: deep learning for medical image analysis." Neuroimaging Clinics 30.4 (2020): 417-431.]

Can overfitting happen with Word2Vec?

Calculating training, validation, and test errors

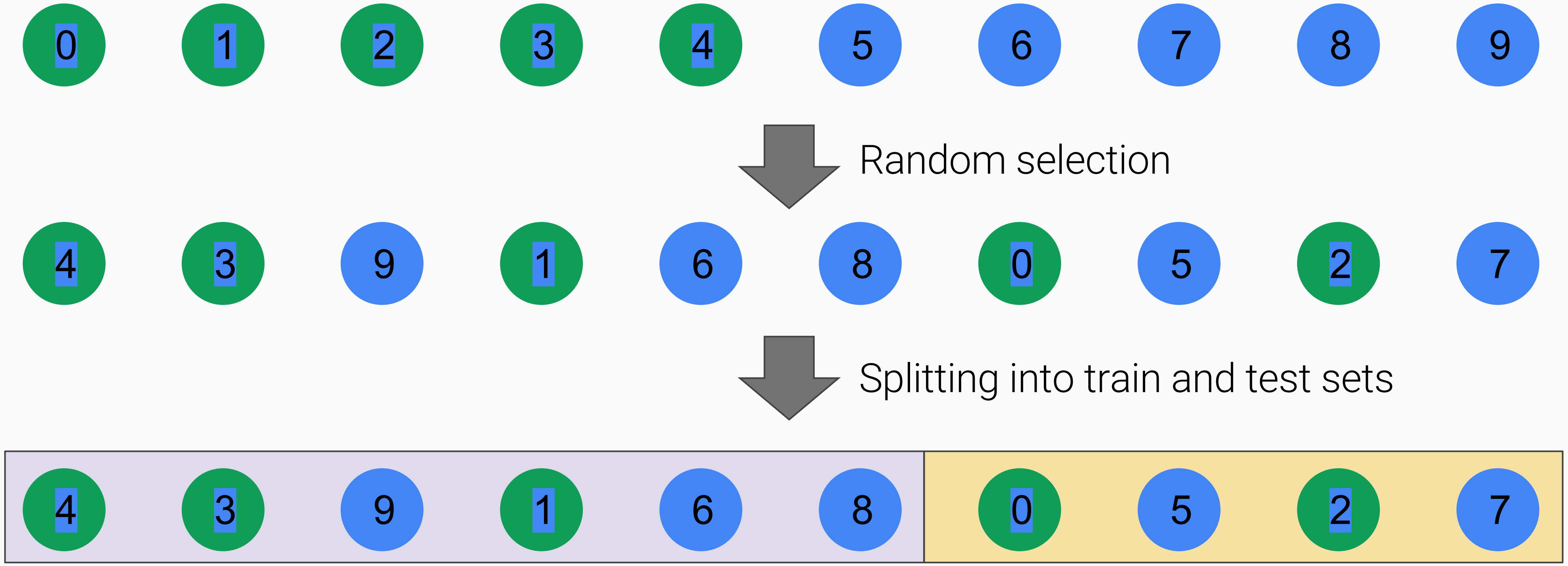
Holdout validation



Why shuffle?

Where is the danger?

Splitting data: Random split



Random split: Code

```
from sklearn.model_selection import train_test_split
train_test_split(*arrays,
                 test_size=None, train_size=None,
                 random_state=None,
                 shuffle=True, stratify=None)
```

This returns sample values.

Read more

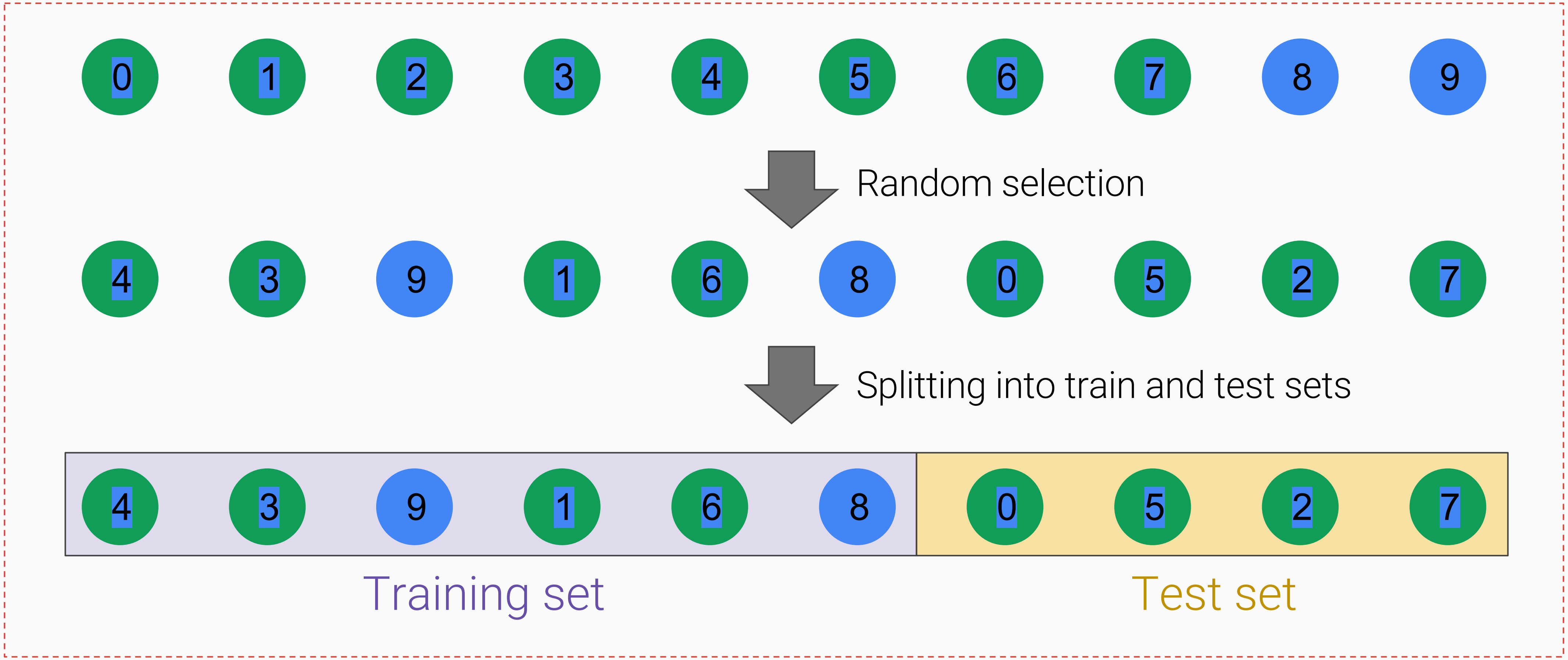
```
from sklearn.model_selection import ShuffleSplit
ShuffleSplit(n_splits=10,
             test_size=None, train_size=None,
             random_state=None)
```

Using the split method, this returns sample indices.

Read more

What is the main shortcoming of random split?

Class imbalance



Does the test set represent the data distribution?

How to address this issue? **Stratified Shuffle Split**

```
from sklearn.model_selection import train_test_split
train_test_split(*arrays,
                 test_size=None, train_size=None,
                 random_state=None,
                 shuffle=True,
                 stratify=None)
```

This returns sample values.

Read more

```
from sklearn.model_selection import StratifiedShuffleSplit
StratifiedShuffleSplit(n_splits=10,
                       test_size=None, train_size=None,
                       random_state=None)
```

Using the split method, this returns sample indices.

Read more

Group Shuffle Split

Safeguarding the independence of training and test samples

```
from sklearn.model_selection import GroupShuffleSplit
GroupShuffleSplit(n_splits=5,
                  test_size=None, train_size=None,
                  random_state=None)
```

Using the split method, this returns sample indices.

Read more

Scenarios

Be aware of the following mistakes:

☠️ **Oversampling before data split**

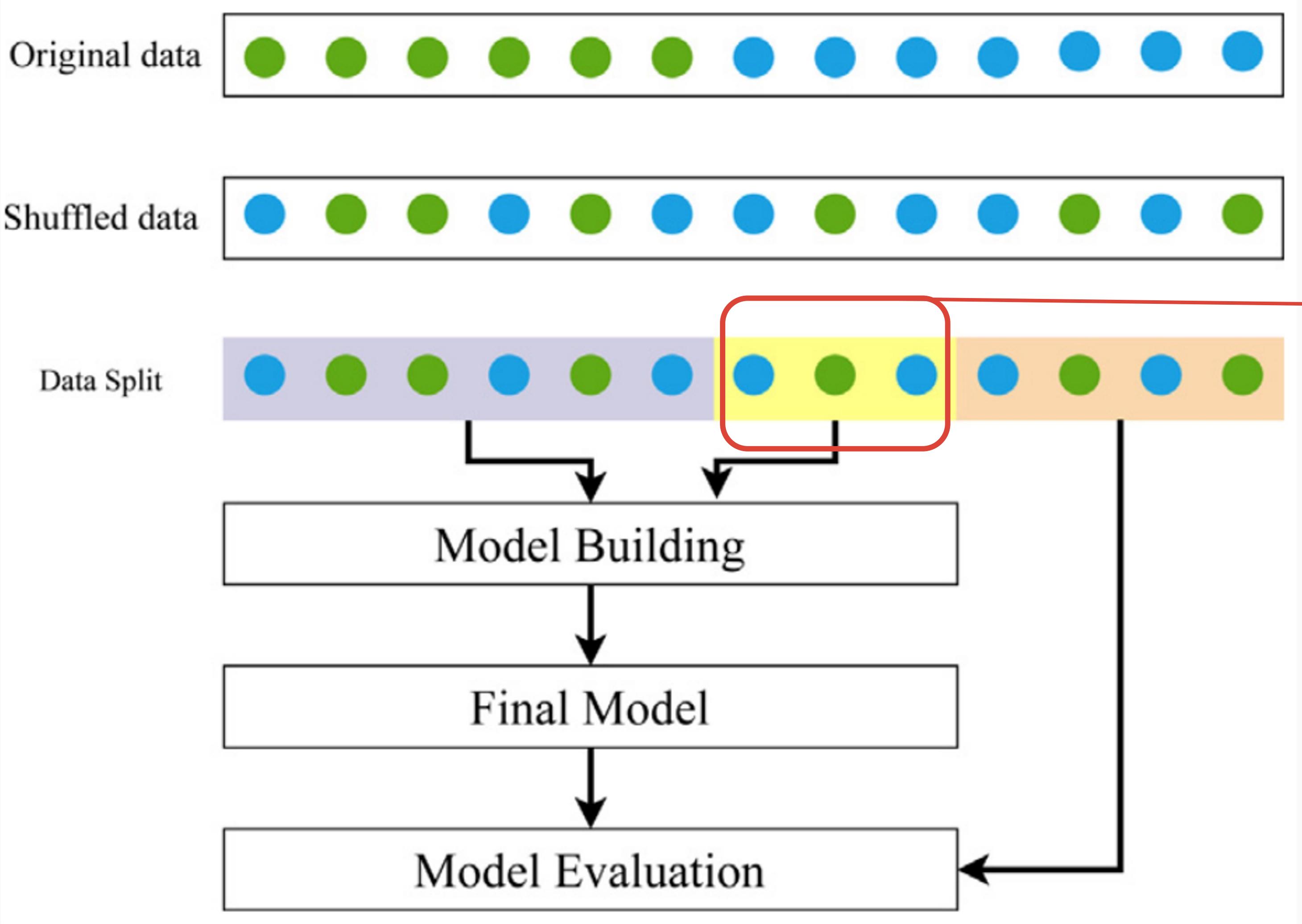
☠️ **Data augmentation before data split**

☠️ **Sample data points across data splits**

Balancing Datasets

Cross-Validation

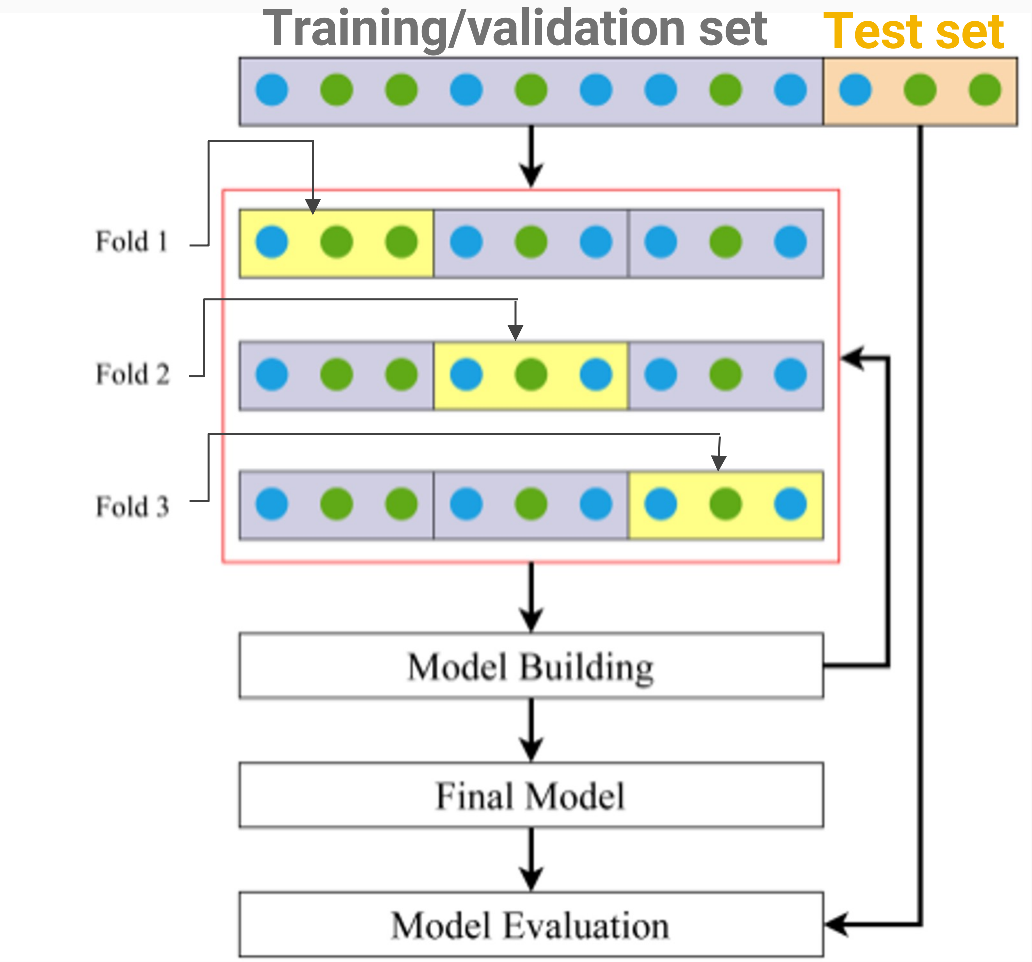
Why cross-validation?



Validation error may significantly vary depending on the composition of validation set

Repeating with different training and validation sets

K-fold cross-validation




```
from sklearn.model_selection import KFold
KFold(n_splits=10,
      shuffle=False,
      random_state=None)
```

Using the split method, this returns sample indices. Read more here!

```
from sklearn.model_selection import StratifiedKFold
StratifiedKFold(n_splits=10,
                shuffle=False,
                random_state=None)
```

Using the split method, this returns sample indices. Read more here!

```
from sklearn.model_selection import GroupKFold
GroupKFold(n_splits=10,
           random_state=None)
```

Using the split method, this returns sample indices. Read more here!

Further readings

Leave-one-out cross-validation

- A special case of K-fold cross-validation

Leave-p-out cross-validation

- An extended form of Leave-one-out cross-validation
- An exhaustive approach and computationally expensive
- Rarely used for p > 2

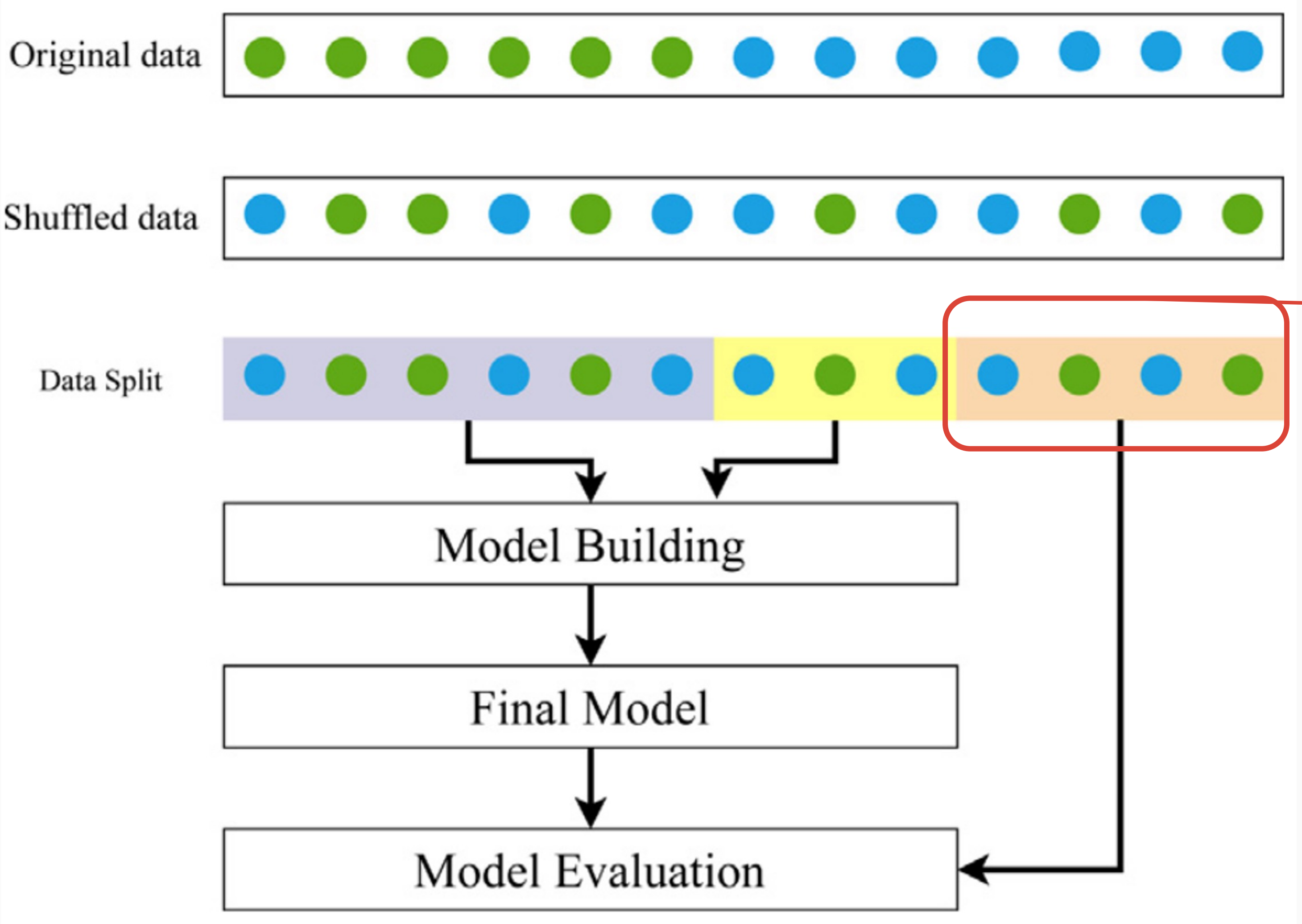
Leave-one-group-out cross-validation

- Similar to one Leave-one-out cross-validation, but uses one group instead of one sample

Nested cross-validation

- Applies two cross-validation in a nested manner
- used to train a model in which hyperparameters also need to be optimized

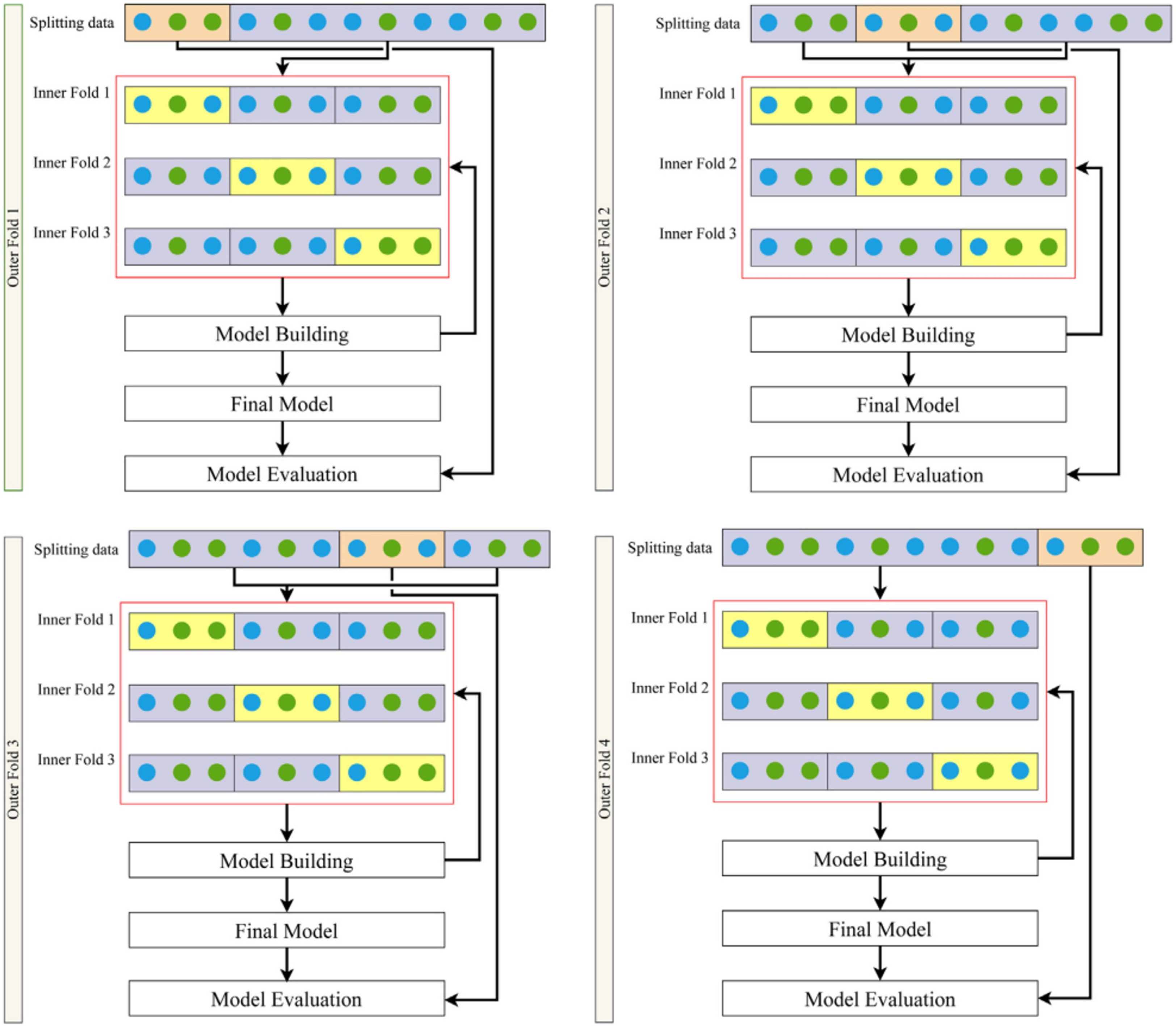
Why nested cross-validation?



Test error may significantly vary depending on the composition of test set

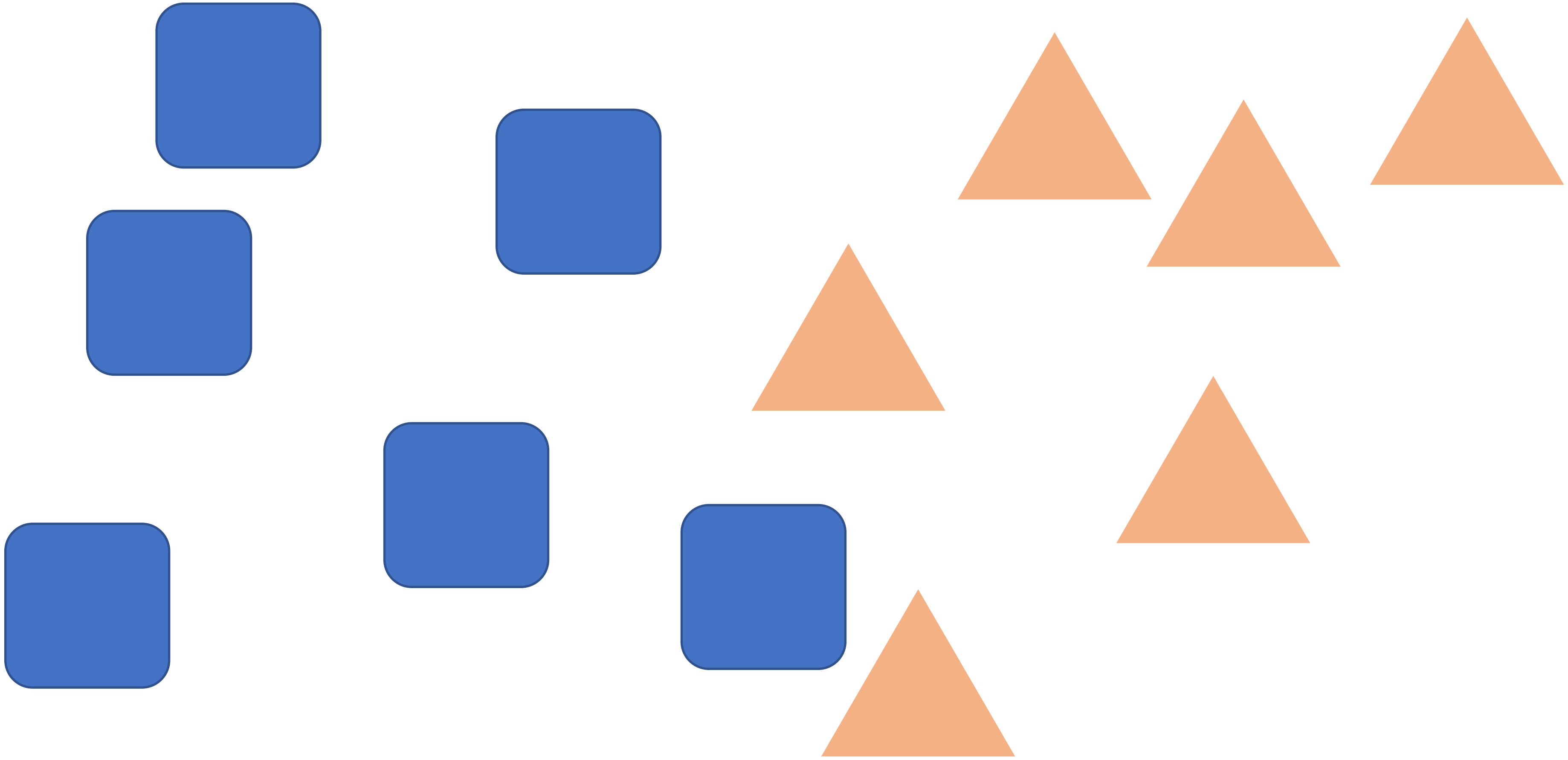
Repeating with different test sets

A solution

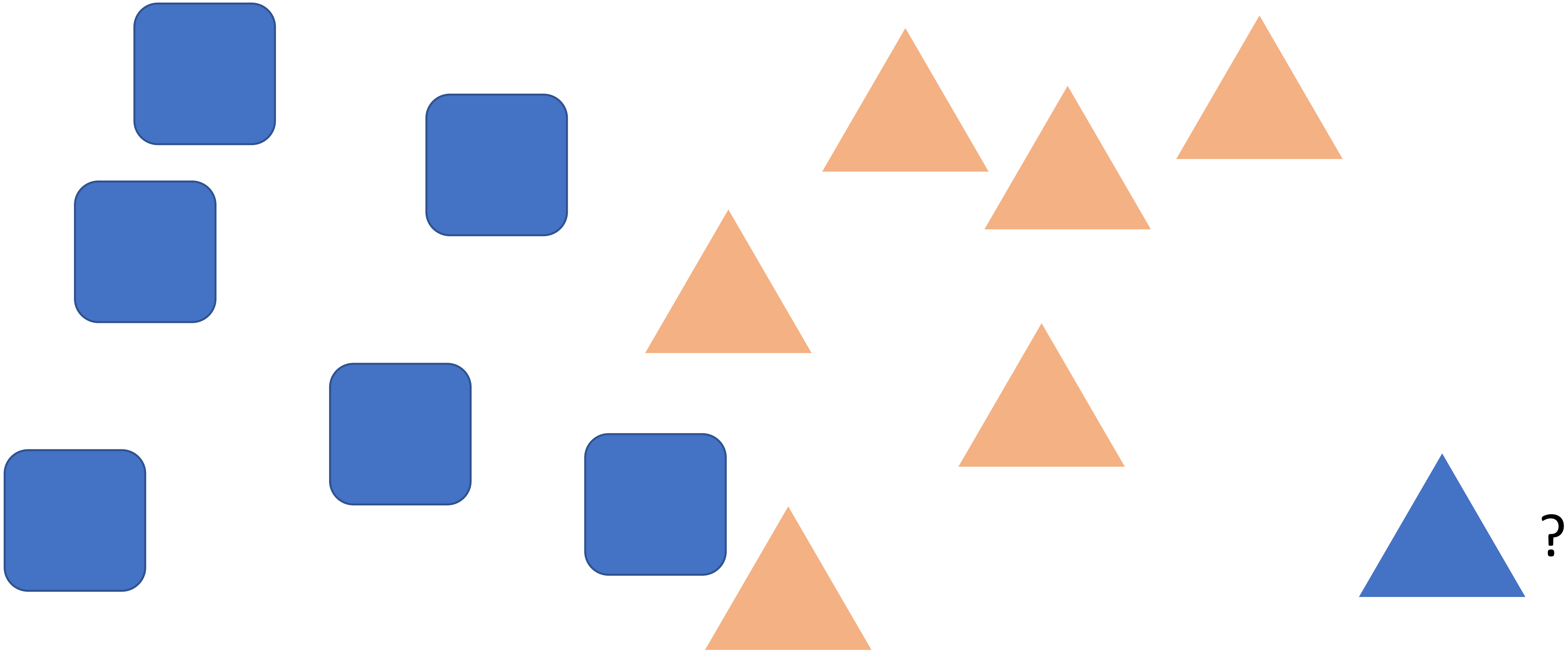


[Maleki, Farhad, et al. "Machine Learning Algorithm Validation: From Essentials to Advanced Applications and Implications for Regulatory Certification and Deployment." Neuroimaging Clinics 30.4 (2020): 433-445.]

What is a batch effect?



What is a batch effect?



In the context of NLP

- Combining datasets, also referred to as Frankenstein datasets to increase dataset size
  - Maybe the writing style is different depending on platform
  - Age of users or other characteristics

Evaluation Metrics

1. Accuracy Score – no. of correctly classified instances/total no. of instances
2. Precision Score – the ratio of correctly predicted instances over total positive instances
3. Recall Score – the ratio of correctly predicted instances over total instances in that class
4. Roc Curve – a plot of true positive rate against false positive rate
5. Classification Report – report of precision, recall and f1 score
6. Confusion Matrix – a table used to describe the classification models

What does the data look like for classification in NLP?

- Need to perform feature extraction for ML solutions
  - BOW
  - TF-IDF
  - Distributed representation vectors
  - Etc.

Naïve Bayes Classification

- Bayes theorem

$$P(y|X) = \frac{P(X|y) * P(y)}{P(X)}$$

We consider every word in our dataset as **independent**

$$P(X|y) = P(x_1, x_2, \ldots., x_n|y)$$

$$= P(x_1|x_2, \ldots., x_n, y) * P(x_2|x_3, \ldots., x_n, y) \ldots. P(x_n|y)$$

$$P(X|y) = P(x_1|y) * P(x_2|y) \ldots\ldots P(x_n|y)$$

$$P(y|X) = \frac{P(x_1|y) * P(x_2|y) \ldots\ldots P(x_n|y) * P(y)}{P(x_1) * P(x_2) \ldots. P(x_n)}$$

Example

15 Not Spam emails and ***10 Spam*** emails

- P(Dear|Not Spam) = 8/34
- P(Visit|Not Spam) = 2/34
- P(Dear|Spam) = 3/47
- P(Visit|Spam) = 6/47
- Etc.

	Not Spam	Spam
Dear	8	3
Visit	2	6
Invitation	5	2
Link	2	7
Friend	6	1
Hello	5	4
Discount	0	8
Money	1	7
Click	2	9
Dinner	3	0
Total Words	34	47

$$P(Hello\ Friend|Not\ Spam) = P(Hello|Not\ Spam) * P(Friend|Not\ Spam)$$

$$P(Not\ Spam|Hello\ Friend) = P(Hello|Not\ Spam) * P(Friend|Not\ Spam) * P(Not\ Spam)$$

$$P(Not\ Spam|Hello\ Friend) = \frac{5}{34} * \frac{6}{34} * \frac{15}{25} = 0.0155$$

$$P(Spam|Hello\ Friend) = \frac{4}{47} * \frac{1}{47} * \frac{10}{25} = 0.00072$$

Laplacian Smoothing

$$P(Not\ Spam|dear\ visit\ dinner\ money\ money\ money)$$

$$= P(dear\ visit\ dinner\ money\ money\ money|Not\ Spam) * P(Not\ Spam)$$

$$P(Spam|dear\ visit\ dinner\ money\ money\ money)$$

$$= P(dear\ visit\ dinner\ money\ money\ money|Spam) * P(Spam)$$

$$\hat{\theta} = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d)$$

https://medium.com/analytics-vidhya/naïve-bayes-algorithm-5bf31e9032a2

How can we improve?

- Maybe we are using too many features
- Few samples of relevant articles (class imbalance)
- Better algorithm
- Preprocessing and feature extraction
- Hyperparameter tuning

Many other options for classification

- SVM
- Logistic Regression
- KNN
- Decision tree
- Random Forest
- XGBoost
- etc.

Next time

- Pre-trained models
- Using these pre-trained models for classification