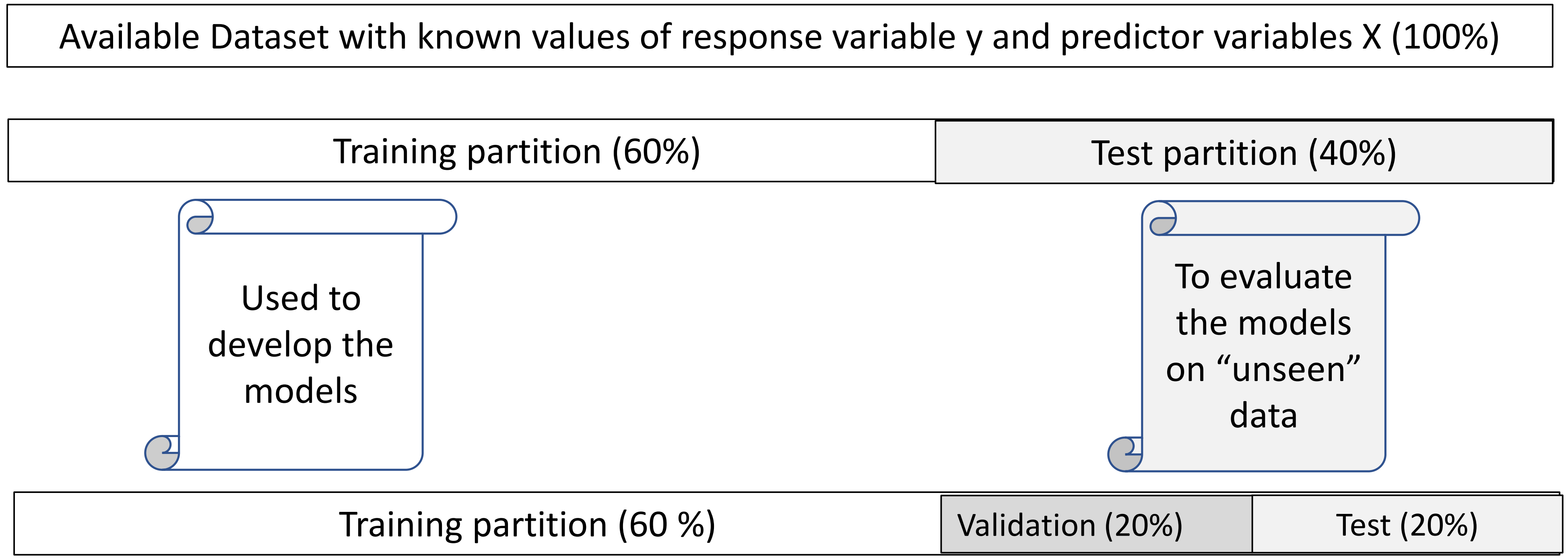
Model Evaluation – Partitioning

Overview

- Partitioning of data : Train – Validation – Test split
- Shortcomings of standard partitioning approach
- k-fold cross validation
- Variations in cross-validation approaches

Partitioning: Train-Test split

- How well will our model perform on fresh data, the model has not seen before and how to reduce the scope for overfitting?
    - ➢ Partition the available data into minimum 2 parts or ideally 3 parts



Available Dataset with known values of response variable y and predictor variables X (100%)
Training partition (60%)
Test partition (40%)
Used to develop the models
To evaluate the models on "unseen" data
Training partition (60 %)
Validation (20%)
Test (20%)

Need for a 3rd Partition

- When a model is developed using **training data**, it can overfit the training data and hence the need to assess the performance on 'unseen' data, a.k.a. validation partition

- Assessing <u>multiple models</u> on the same 'unseen' **data,** can again lead to overfitting on this data (i.e. the validation partition)

- Hence the <u>final selected model</u> is applied to the third **partition (Test partition)** to give an unbiased estimate of its performance on 'new' data. Test partition also referred to as 'Holdout' set

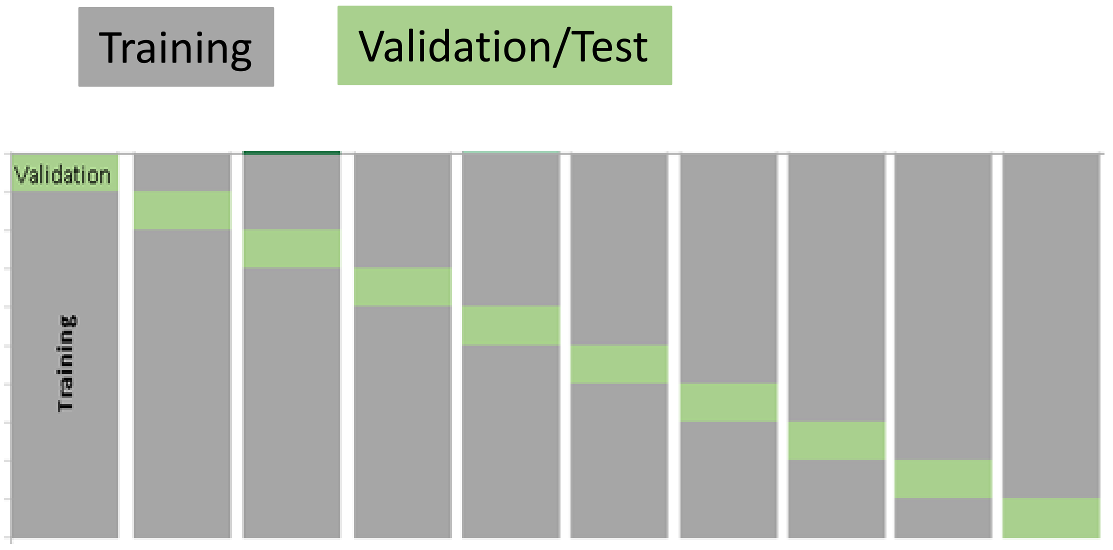
Note - ML literature typically refers to the 2 partition scenario as Training/Test, whereas some literature refer to them as Training/Validation.

Shortcomings of training, validation and test split

- Given the split into only one set of training, validation, test partitions, the performance of our models are highly dependent on the nature of data in those partitions
- Thus performance might vary if the split points were different and training and evaluation were on different subsets of data
- What if we could do this split into training and validation set multiple times, each time on different subsets of the same data, and then train and evaluate our models each time to look at the average performance of the models across multiple evaluations?

k-fold Cross Validation

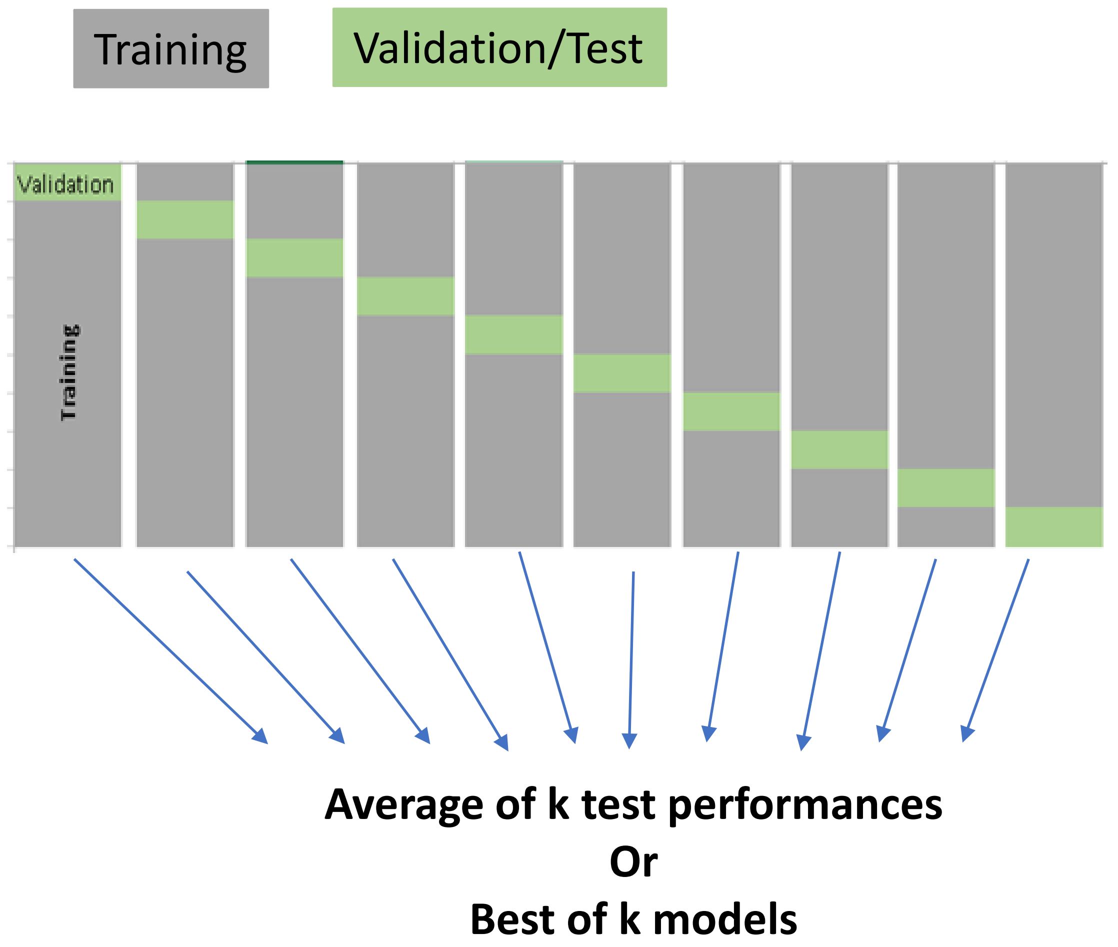
- An alternative to data partitioning, esp. when the number of records in the dataset is small
- Partitioning the data into "folds," or non-overlapping (disjoint) sub-samples.
- If we choose *k* = 10 folds, meaning that the data are randomly partitioned into 10 equal parts, where each fold has 10% of the observations.



Ref: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

k-fold Cross Validation

- Randomly split your entire dataset of n instances into k ‘folds’ (each of size n/k)
- For each k-fold in your dataset, build a model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold
- Record the error you see on each of the predictions
- Repeat this until each of the k-folds has served as the test set
- The average of your k recorded errors is called the cross validation error and will serve as your performance metric for the model



Ref: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

How to choose value of k in ‘k-fold cross validation’

- On small datasets, lower value of k could increase bias and hence not desirable (similar to the use of traditional partitions – train/validation/test)

- Higher values of k reduces bias, but increases scope for variability

- Special case of k = n, referred to as ‘Leave one out cross validation (LOOCV)’ or n-fold cross validation (where n = number of training samples)
  - Reserve only one data point from the available dataset, and train the model on the rest of the data. This process iterates for each data point and hence k=n.
  - Computationally expensive

Ref: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

k-fold Cross-Validation for Parameter Tuning

- Sometimes cross validation is built into ML algorithm implementation, with the results of the cross validation used for choosing the algorithm’s parameters

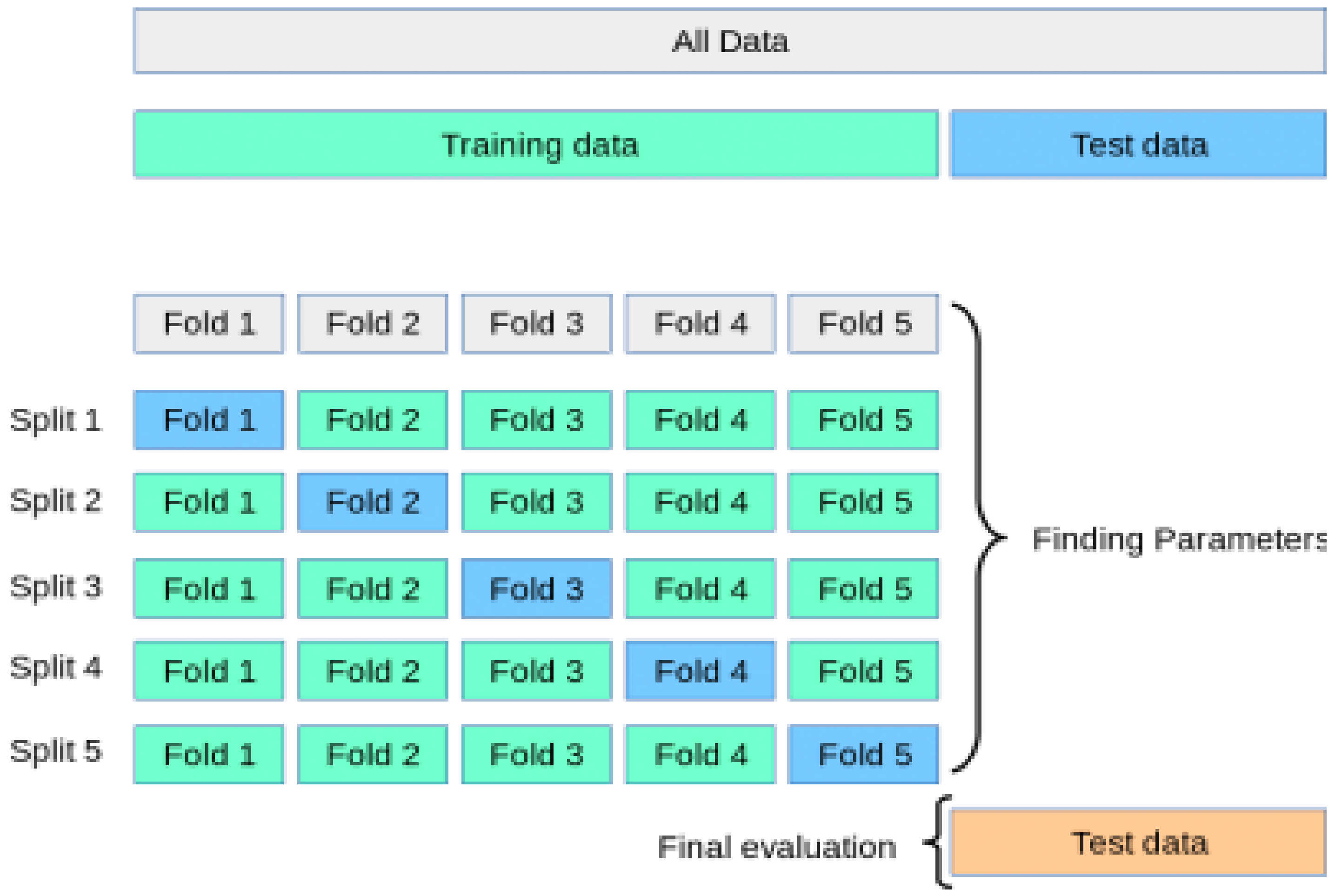


Image Source: https://towardsdatascience.com/cross-validation-and-hyperparameter-tuning-how-to-optimise-your-machine-learning-model-13f005af9d7d

Variations of Cross validation

- **Train/Test Split**: k=2 such that a single train/test split is created to evaluate the model.
- **LOOCV**: Leave-one-out cross-validation, k=n
- **Stratified**: Each fold has the same proportion of observations with a given categorical value (Genuine/Fraudulent), such as the class outcome value.
- **Repeated**: k-fold cross-validation procedure is repeated n times, the data sample is shuffled prior to each repetition, which results in a different splits
- **Nested**: k-fold cross-validation performed within each fold of cross-validation, often to perform hyperparameter tuning during model evaluation. Also called double cross-validation.

Ref: https://machinelearningmastery.com/k-fold-cross-validation/

Summary

- Partitioning available data into Training-Validation-Test sets, where multiple models are developed using the Training set, and evaluated against the Validation set, and the best model is re-evaluated against the test set before being applied on fresh data

- Performance might vary if the split points were different and training and evaluation were on different subsets of the same dataset. Instead, averaging over multiple iterations of such partitions is more robust than a single train-validation partition of data

- k-fold cross validation involves partitioning data into “folds,” or non-overlapping (disjoint) sub-samples so as to build the model on k – 1 folds of the dataset and then test the model on the k^{th} fold.

- k-fold cross validation is also used for parameter tuning to arrive at the optimal values of parameters of a model before applying it on test data

References

- https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6
- https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/
- https://machinelearningmastery.com/k-fold-cross-validation/
- https://towardsdatascience.com/cross-validation-and-hyperparameter-tuning-how-to-optimise-your-machine-learning-model-13f005af9d7d