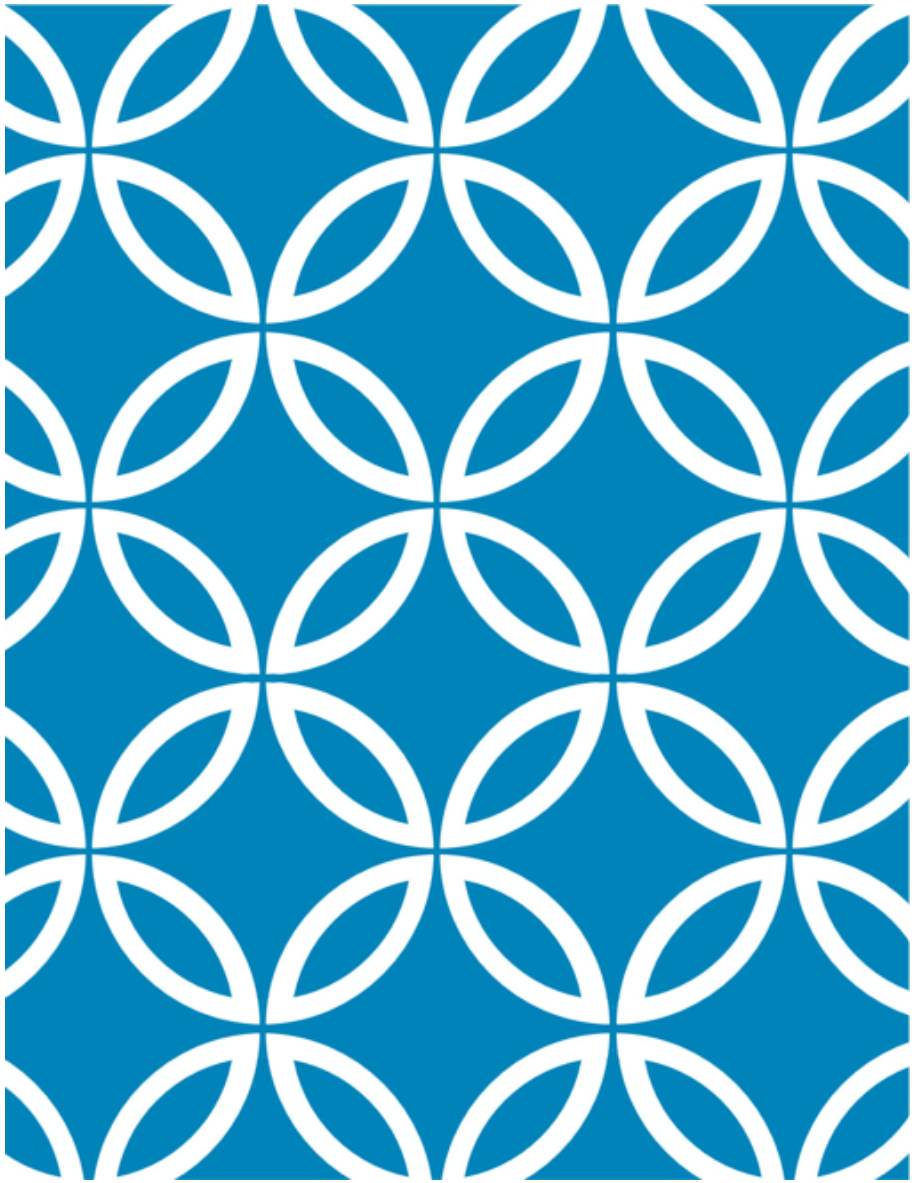

MODEL EVALUATION, MODEL SELECTION, AND ALGORITHM SELECTION IN MACHINE LEARNING

Research Review by William Steimel

SOURCE

Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

- Sebastian Raschka- November 2018 – University of Wisconsin

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION: ESSENTIAL MODEL EVALUATION TERMS AND TECHNIQUES

Machine Learning is becoming important to everyday life and model evaluation is a step in the machine learning pipeline for selecting a good model.

If we cannot evaluate models then we cannot understand how well the model performs in the real world.

1.1 PERFORMANCE ESTIMATION: GENERALIZATION PERFORMANCE VS MODEL SELECTION

Making predictions of future unseen data is often the main problem we want to solve in machine learning.

Performance Estimates: To select the best performing model we need a method to rank them.

- Predictive performance
- Computational performance

There are 3 main reasons why we evaluate predictive performance:

- We want to see how the model performs on unseen data
- To increase predictive performance with model tweaking and selecting the best one
- We want to compare different algorithms and select the best one suited to the problem

1.2 ASSUMPTIONS/TERMINOLOGY

i.i.d – Independent and identically distributed (All data from the same probability distribution)

Supervised Learning/Classification – This paper focuses on classification (prediction of categorical value) area of supervised learning

Prediction Accuracy - $ACC = 1 - ERR$

Error – $Err_S = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i)$

0-1 Loss – $L(\hat{y}_i, y_i) = \begin{cases} 0 \; if \;\; \hat{y}_i = y_i \\ 1 \; if \; \hat{y}_i \neq y_i \end{cases}$

Bias- Refers to statistical bias (difference between expected value $E[\hat{\beta}]$ and true value of β)

- $Bias = E[\hat{\beta}] - \beta$
- If this value is 0 it means we have an unbiased estimator.

Variance- Statistical Variance of estimator β, Variance is a measure of a models variability to make predictions. The more sensitive toward fluctuate the higher the variance.

- $Variance = E[\,(\hat{\beta} - E[\hat{\beta}])^2]$

Target Function – The function we are interested in modeling – the target function $f(x) = y$ is the true function $f(\cdot)$ that we want to model.

Hypothesis – A function we believe to be true to the target function $f(\cdot)$

Model – Materialization of this guess to test the hypothesis.

Learning Algorithm – Set of instructions that tries to model the target function using training set.

Hyperparameters – tuning parameters of Machine Learning algorithm (L2 penalty in LR)

1.3 RE-SUBSTITUTION VALIDATION AND HOLDOUT METHOD

The holdout method is the simplest model evaluation technique

Steps:

- 1. Take label set and split into two parts
  - Training/Test Set
- 2. Fit a model to the training data and predict labels of test set.
- 3. Models predictive accuracy can be determined by comparing the predicted labels to the ground truth labels of test set.

Assumptions:

- We do not want to train and evaluate on the same dataset as it introduces optimistic bias.
- The method of splitting a train and test set is a simple process of random subsampling where it is assumed all data points come from the same distributions

1.4 STRATIFICATION

Lets randomly divide the Iris dataset (50 setosa, 50 versicolor, 50 virginica)

- 2/3 Training Set
- 1/3 Test set

Whats the problem?

- When we randomly divide we violate the assumption of statistical independence.
- If we split at random we may end up with
  - Training- 38 setosa, 28 versicolor, 34 virginica (38 %/28%/34%)
  - Test- 12 setosa, 22 versicolor, 16 virginica (24 %/44%/32%)
- We just created two imbalanced datasets

Imbalanced Data is not ideal unless our algorithm can handle class imbalance and made even worse when our dataset is extremely unbalanced.

- It is recommended to split the dataset in a stratified manner.
  - All classes equally represented in the training and test sets

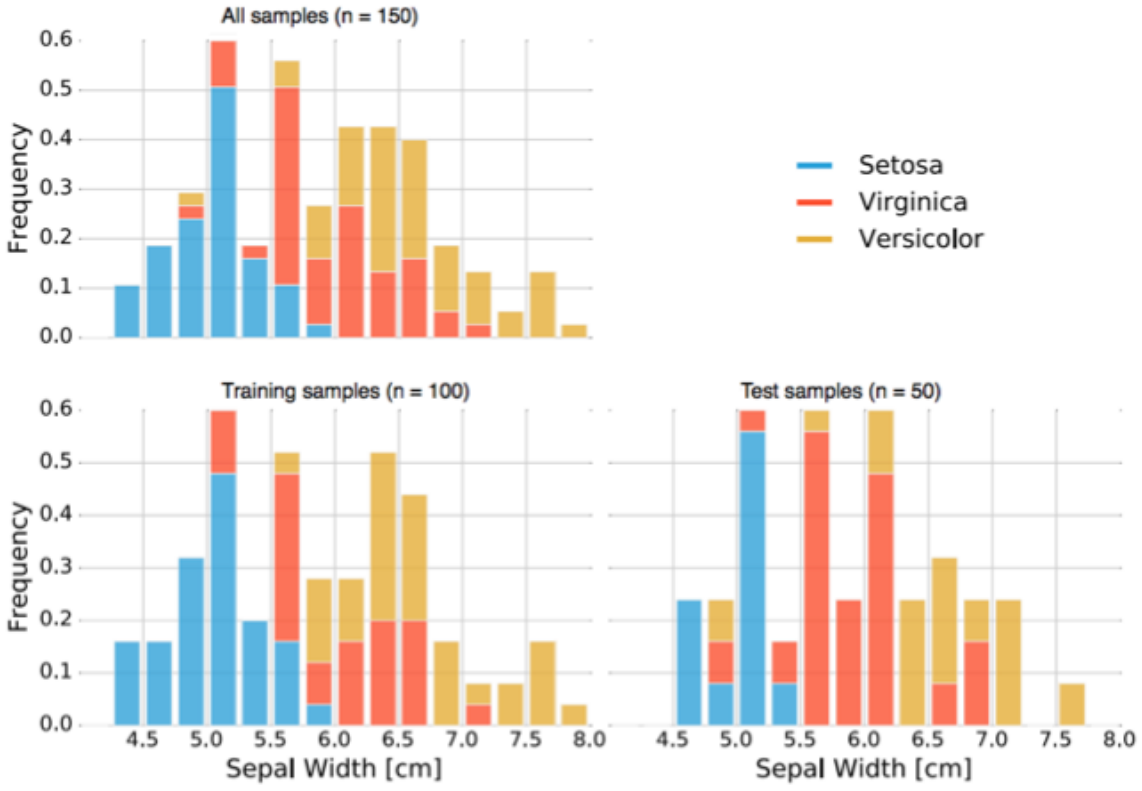

Figure 1: Distribution of *Iris* flower classes upon random subsampling into training and test sets.

1.5 HOLDOUT VALIDATION

Step1: Divide dataset into two subsets training set/test set (Usually 70/30, 80/20)

Step 2: Pick appropriate learning algorithm for problem

Step 3: Test set is used for evaluation as it represents new unseen data. We then compare predicted class labels against actual labels to estimate generalization accuracy/error.

Step 4: We have obtained a measure of how well the model performs on unseen data. We can now use all of the data to train the algorithm as in theory it should be perform better with more data.

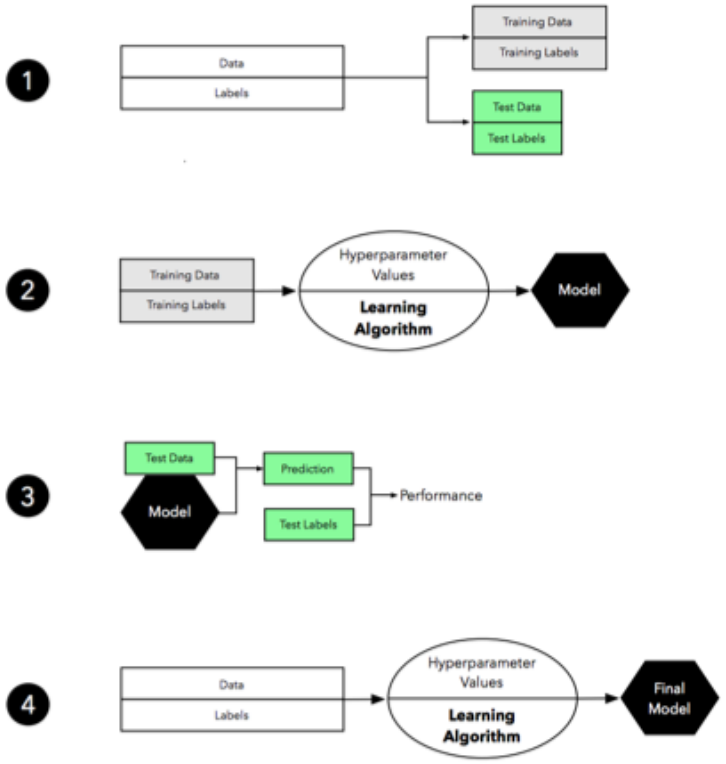

Figure 2: Visual summary of the holdout validation method.

1.6 PESSIMISTIC BIAS

Re-substitution Validation and the holdout method illustrate two problems that occur when splitting into separate train and test sets.

- Violation of independence and changing of class proportions due to subsampling
- Capacity of Model – a model that has not reached capacity would have an performance improvement with more data but since all the data has been used we cannot accurately evaluate it.
  - "We should be aware that our estimate of generalization performance may be pessimistically biased if only a portion of the dataset (training) is used for model fitting."

1.7 CONFIDENCE INTERVALS VIA NORMAL APPROXIMATION

In section 1.5 the generalization performance was calculated with the holdout method but the confidence intervals around the estimate would be useful as they can be more informative.

- A simple way to calculate confidence intervals – Normal Approximation

We compute the prediction accuracy on Dataset S (here: test set) of size n

$$ACC_S = \frac{1}{n}\sum_{i=1}^{n}\delta(L(\hat{y}_i, y_i)),$$

We could now consider each prediction as a Bernoulli trial, and the number of correct predictions X is following a binomial distribution $X \sim (n, p)$

- n = test examples
- k = trials
- p = probability of success
- Where $n \in \mathbb{N}\ and\ p \in [0,1]$

$$f(k; n, p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k},$$

1.7 CONFIDENCE INTERVALS VIA NORMAL APPROXIMATION

for $k = 0, 1, 2, ..., n$, where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

Expected number of successes is computed as $u = np$, if the model has 50 % success rate 20/40 estimate has a variance and standard deviation of:

$$\sigma^2 = np(1-p) = 10$$

$$\sigma = \sqrt{np(1-p)} = 3.16.$$

1.7 CONFIDENCE INTERVALS VIA NORMAL APPROXIMATION

However, we are interested in average number of success, not its absolute value so we can compute the variance and standard deviation of the accuracy estimate as:

$$\sigma^2 = \frac{1}{n} ACC_S(1 - ACC_S), \qquad \sigma = \sqrt{\frac{1}{n} ACC_S(1 - ACC_S)}.$$

Under normal approximation we compute the confidence interval as:

$$ACC_S \pm z\sqrt{\frac{1}{n} ACC_S\,(1 - ACC_S)},$$

- α = error quantile
- $z = 1 - \frac{\alpha}{2}$ of standard normal distribution (for a typical confidence interval 95 %, z = 1.96)

Key note: having fewer samples in the set increases variance and widens the confidence interval.

In practice it is recommended to repeat training-test split multiple times to compute the confidence interval on the mean estimate (averaging the runs).

2. BOOTSTRAPPING AND UNCERTAINTIES

2.1 OVERVIEW

The previous section discussed model evaluation terms and techniques including

- Holdout Method
- Normal Approximation

This section introduces more advanced techniques for estimating uncertainty of model performance as well as models variance/stability

2.2 RESAMPLING

Performance Estimates may suffer from bias/variance and we are looking for a good trade off.

Re-substitution evaluation – Optimistically biased (Model thinks it is stronger than it really is)

- (Using training set for evaluation)

Withholding a large portion of dataset – Pessimistic Bias (Model is assumed weaker than it really is)

This section introduces alternative resampling methods for finding a good balance between bias and variance for model evaluation and selection.

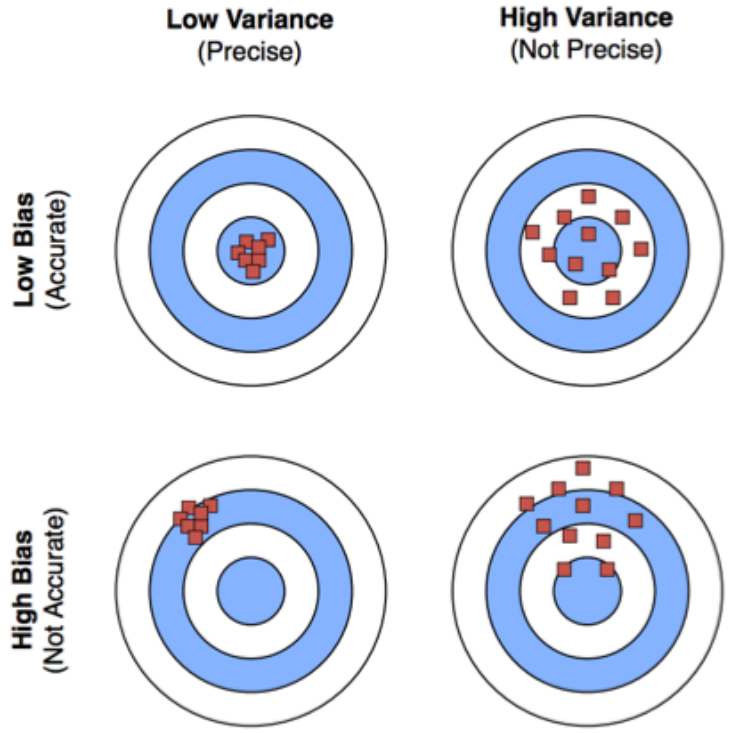

Figure 3: Illustration of bias and variance.

2.2 RESAMPLING

Why do proportionately larger test sets increase pessimistic bias?

- The model may have not yet reached full capacity
- If the model saw more data it could have created a more generalized hypothesis

The graph on the right illustrates the learning curves of classifiers fit to MNIST at various Training Set Sizes.

We can conclude from the figure:

- 1. Re-substitution accuracy (training set) declines as number of samples grow
- 2. Improved generalization accuracy (test set) with more training samples

Now that we understand that pessimistic biases occur with disproportionately larger test sets how does reduction of the test size impact evaluation?

- May lead to substantial variance of models performance estimate as it depends on which samples end up in training/test set

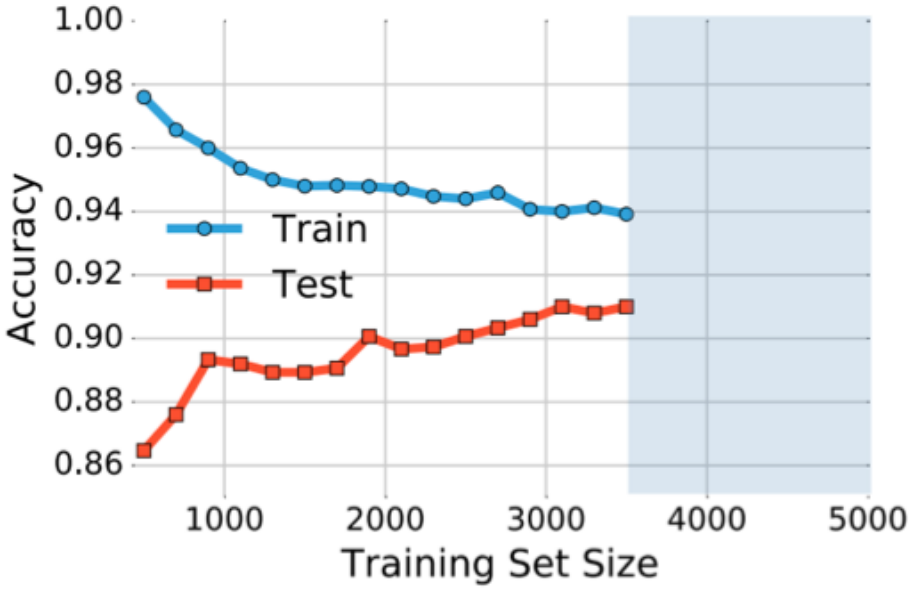

Figure 4: Learning curves of softmax classifiers fit to MNIST.

2.2 RESAMPLING

Supervised Learning algorithms typically operate under the assumption that the training data is representative of the test set.

- Resampling alters the statistics/distribution of the sample
- Stratification can help but the problem becomes more obvious with smaller datasets as show by figure 5.

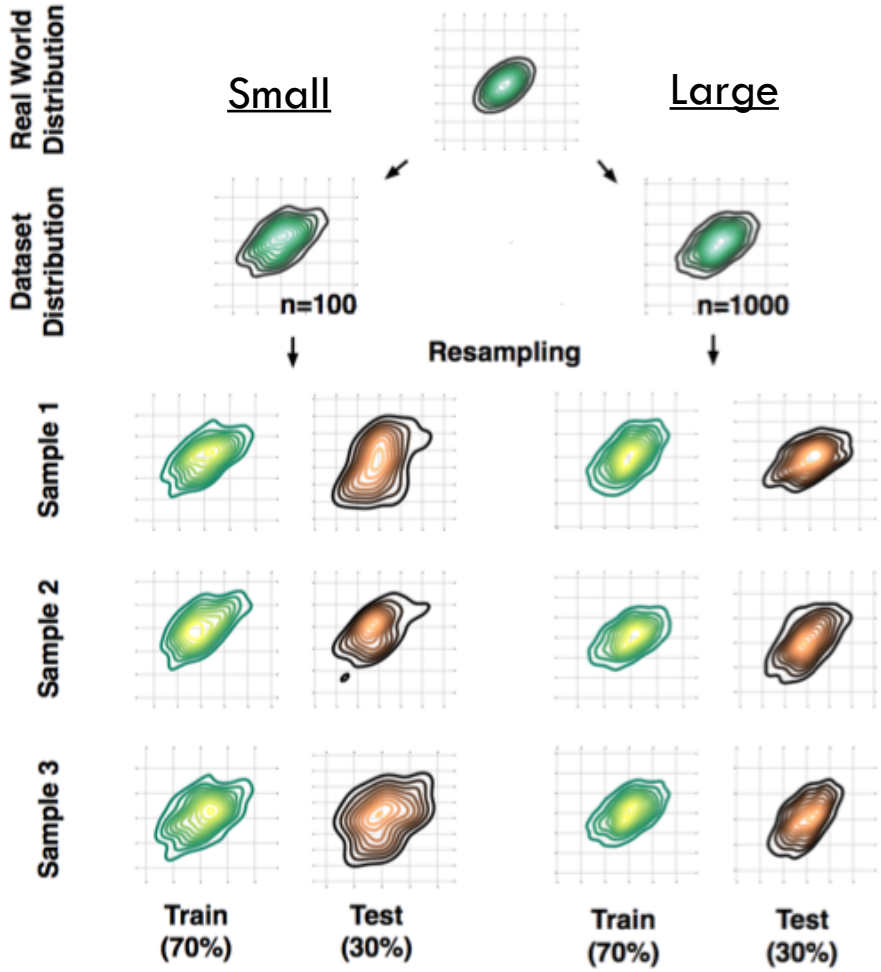

Figure 5: Repeated subsampling from a two-dimensional Gaussian distribution.

2.3 REPEATED HOLDOUT VALIDATION

One performance estimate that is less variant to how we split data into training/test sets is called Repeated Holdout Validation:

Steps: Repeat holdout method k times with different random seeds and compute average performance over k repetitions:

$$\text{ACC}_{avg} = \frac{1}{k}\sum_{j=1}^{k} ACC_j,$$

Where ACC_j is the accuracy estimate of the jth test set of size m

$$\text{ACC}_j = 1 - \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}_i, y_i).$$

This provides a better estimate compared to standard holdout validation:

Insight into models stability (How repeated holdout validation changes with different splits.)

We can observe two trends based on Figure 6:

- Variance of estimate increases as size of test set decreases.
- Small decrease in pessimistic bias when training size decreased.

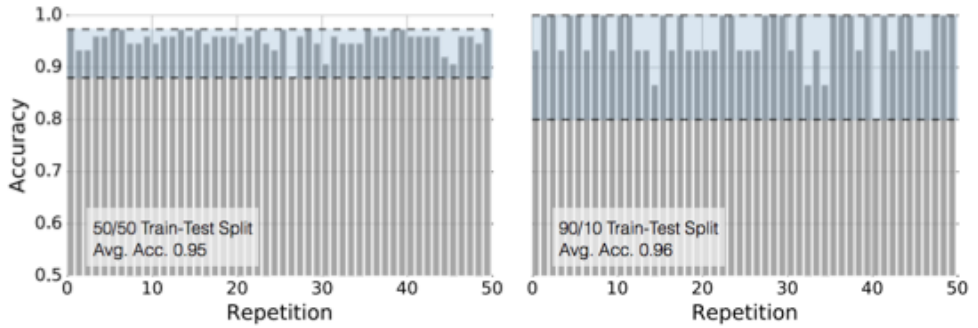

Figure 6: Repeated holdout validation with 3-nearest neighbor classifiers fit to the Iris dataset.

2.4 THE BOOTSTRAP METHOD AND EMPIRICAL CONFIDENCE INTERVALS

Lets assume we would like to compute a confidence interval around a performance estimate to judge its certainty. A method called bootstrap could be used.

The main idea behind bootstrapping is generating new samples by sampling from an empirical distribution.

The bootstrap method is a resampling technique for estimating sampling distribution (in this case understanding the certainty of a performance estimate)

- Generate new data from population by repeated sampling with replacement

2.4 THE BOOTSTRAP METHOD AND EMPIRICAL CONFIDENCE INTERVALS

Bootstrap Steps:

1. We are given dataset size n

2. For b bootstrap rounds, we draw a single instance from the dataset and assign it to the jth bootstrap sample until n - size of the original dataset.

3. We fit each model to b bootstrap samples and calculate re-substitution accuracy

4. Compute model accuracy over average of b estimates.

$$\mathrm{ACC}_{boot} = \frac{1}{b}\sum_{j=1}^{b}\frac{1}{n}\sum_{i=1}^{n}\Big(1 - L\big(\hat{y}_i, y_i\big)\Big)$$

In order to leverage this for predictive models (LOOB) Leave-one-out Bootstrap technique is recommended.

- "out-of-bag" samples are used for test set evaluation as seen in figure 7.

Also discussed in this section are other variants of the Bootstrap method which will not be discussed in this presentation.

- Percentile Method

Figure 7: Illustration of training and test data splits in the Leave-One-Out Bootstrap (LOOB).

3 CROSS-VALIDATION AND HYPERPARAMETER OPTIMIZATION

3.1 OVERVIEW

This section focuses on different methods for cross-validation for model evaluation and selection.

Methods for ranking models' ability to generalize with several hyperparameter configurations are covered.

3.2 ABOUT HYPERPARAMETERS AND MODEL SELECTION

What are hyperparameters?

Lets consider k-nearest neighbors algorithm:

- K-value (Hyperparameter) – number of neighbors
- Depending on how we set this number, the performance of the algorithm changes.

In the case of Logistic Regression:

- A sample hyperparameter could be the number of iterations or passes over the training set in gradient-based optimization (epochs)
- Lambda Term- L2 Regularized Logistic Regression

- Changing the hyperparameter values when running a learning algorithm may result in very different models.
- The process of finding the best-performing model from a set of models that were produced by different hyperparameter settings is called model selection.

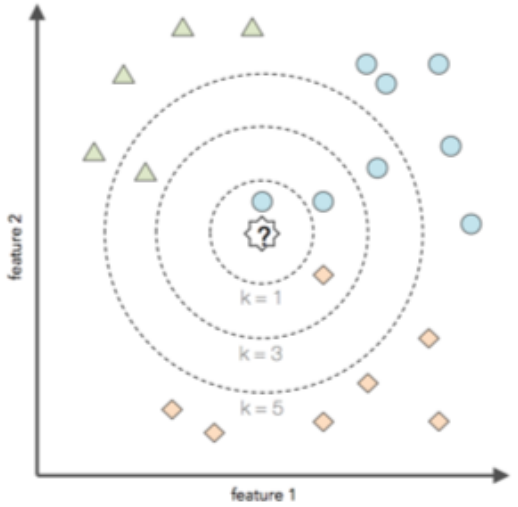

Figure 10: Illustration of the k-nearest neighbors algorithm with different choices for k.

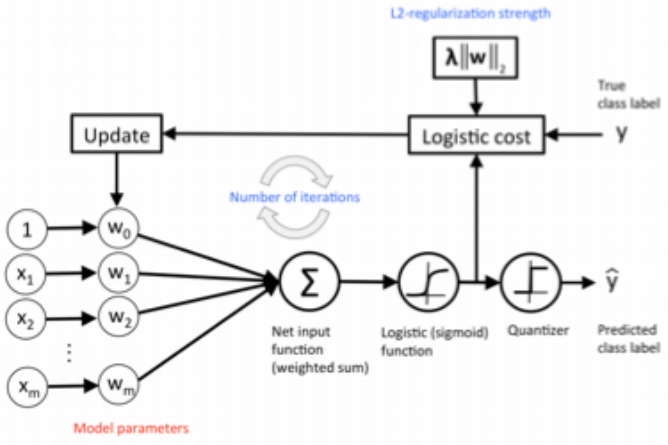

Figure 11: Conceptual overview of logistic regression.

3.3 THE THREE WAY HOLDOUT METHOD FOR HYPERPARAMETER TUNING

A slight modification to the holdout method splits the dataset into three parts for hyperparameter tuning. (training, validation, and test)

Steps:

- Step 1: We start by splitting the dataset into three parts (training, validation, and test)
- Step 2: Hyperparameter tuning stage - We use the learning algorithm with different hyperparameter settings (this example uses 3) to fit models to the training data.
- Step 3: Evaluate performance of models on validation set. We choose the hyperparameters associated with the best performance.
- Step 4: The performance estimates may suffer from pessimistic bias if the training set is too small. The training and validation set are merged for with the best hyperparameter settings are used for training.
- Step 5: The model is evaluated on the independent test set as the dataset has not been used yet.
- Step 6: Finally, make use of all data and fit the model for re-world use. (Optional)

In theory, using all data should improve performance in the last optional step. In real world applications, often having the best performing model is desired.

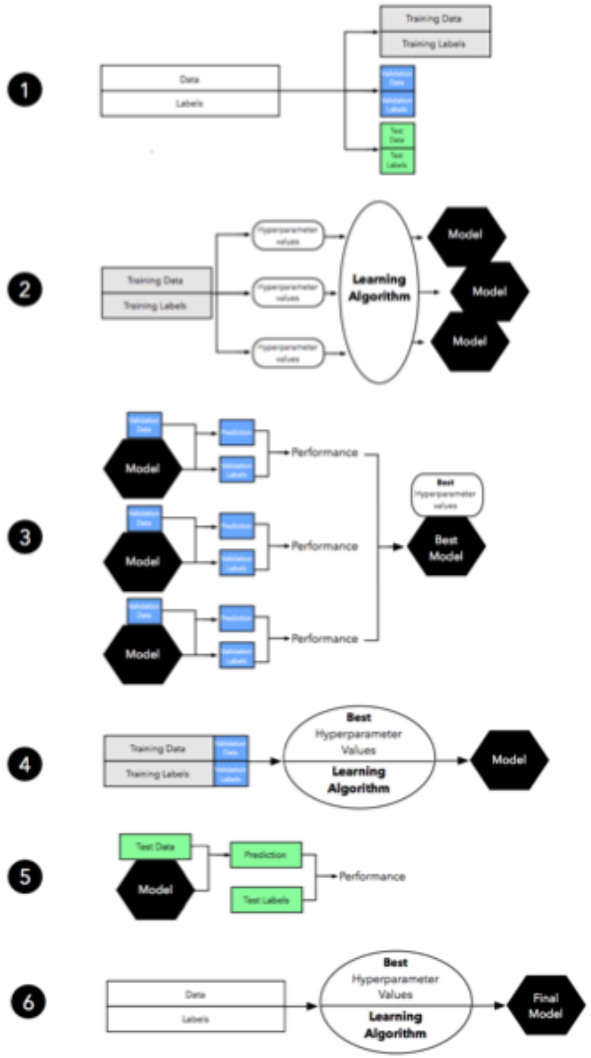

Figure 12: Illustration of the three-way holdout method for hyperparameter tuning.

3.4 INTRODUCTION TO K-FOLD CROSS-VALIDATION

K-Fold cross-validation - The most common technique for model evaluation and model selection in machine learning practice.

The main idea behind cross validation is that each sample in the dataset has an opportunity of being tested.

K-fold cross-validation is a special case where we iterate over the dataset k times.

In each round, we split the data into k parts. One part is used for validation and the remaining k-1 parts are merged for training subset

5-Fold cross-validation is illustrated in Figure 13.

- 5-fold cross-validation results in 5 different fitted models in which the cross-validation performance is computed by calculating the arithmetic mean over the performance estimates.

K-fold cross validation is typically used for model selection or algorithm selection.

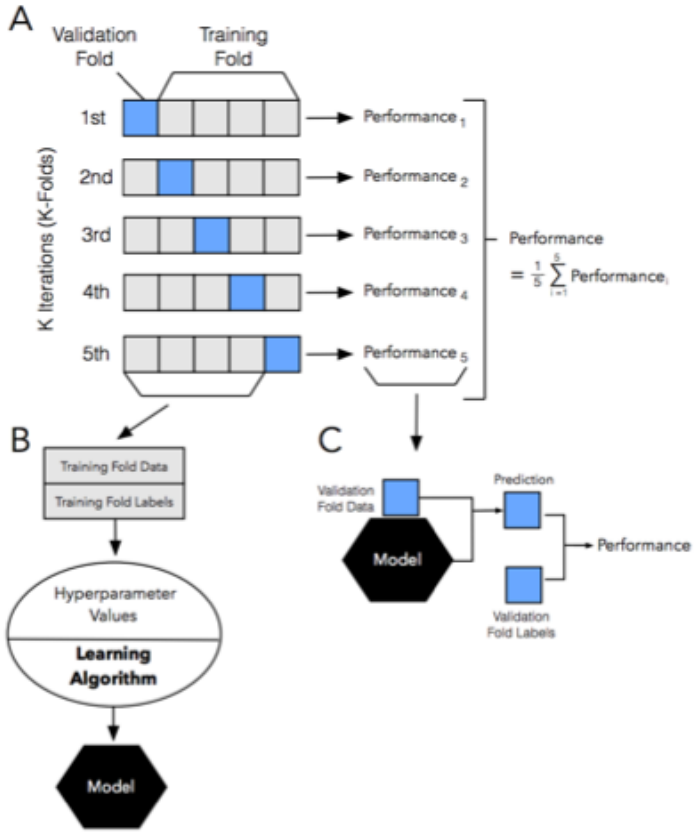

Figure 13: Illustration of the *k*-fold cross-validation procedure.

3.5 SPECIAL CASES: 2-FOLD AND LEAVE-ONE-OUT CROSS-VALIDATION

K = 5 is a common choice for K-fold Cross validation as illustrated in the previous slide.

- (10 is also common)

However, there are two special cases of k in cross-validation

- k=2
- k=n

K=2 is often seen as being equal to the holdout method but is not exactly the same.

2- Fold Cross-Validation

This would only be the case if we split data 50/50 in holdout validation and took the arithmetic mean of both model's performance

K=n (number of folds equal to number of training instances)

- We refer to the process as Leave-One-Out Cross-Validation (LOOCV)
- This process is computationally expensive but useful on small datasets.

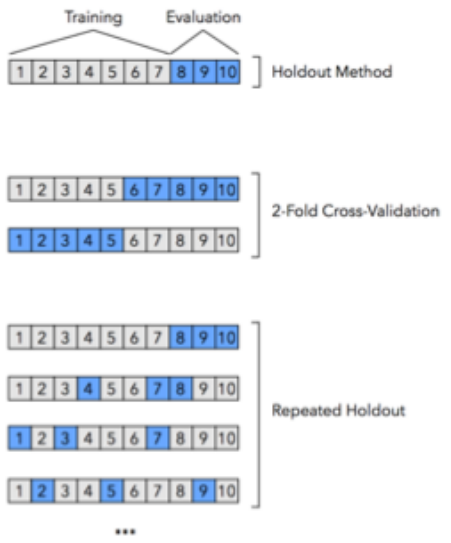

Figure 14: Comparison of the holdout method, 2-fold cross-validation, and the repeated holdout method.

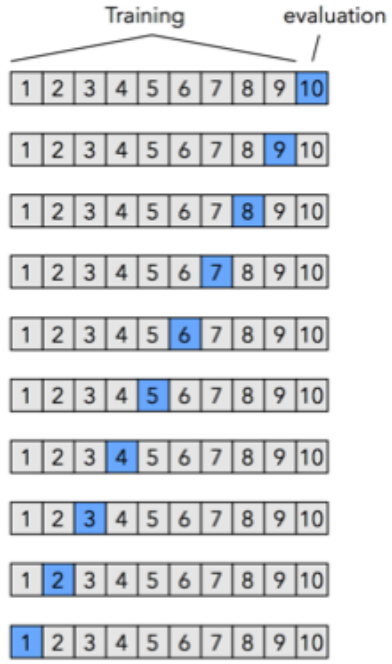

Figure 15: Illustration of leave-one-out cross-validation.

3.6 K-FOLD CROSS-VALIDATION AND THE BIAS-VARIANCE TRADE-OFF

The Bias-Variance Trade-Off associated with K-Fold Cross-Validation can be summarized below:

General trends when increasing number of folds (k):

- The bias of performance estimator decreases (more accurate)
- The variance of performance estimators increases (more variability)
- The Computational Cost increases (More model fitting)

Exception: Decreasing to two or three folds will lead to increased variance on small datasets due to random sampling effects.

3.7 MODEL SELECTION VIA K-FOLD CROSS-VALIDATION

Previous sections illustrated K-fold cross-validation for model evaluation but this section refers to K-Fold cross validation in the context of model selection.

Steps:

- Step 1: Split the dataset into two parts (train/test set)
- Step 2: Experiment with various hyperparameter settings (We could use Bayesian optimization, randomized search, grid search. Etc.) For each hyperparameter configuration k-fold cross-validation is applied resulting in multiple performance estimates.
- Step 3: We take the hyperparameter results that produced the best result in k-fold cv - we can now use the complete training set for model fitting with these settings.
- Step 4: Test set is used to evaluate the model
- Step 5: We can optionally fit the model to the entire dataset (Deployment Model)

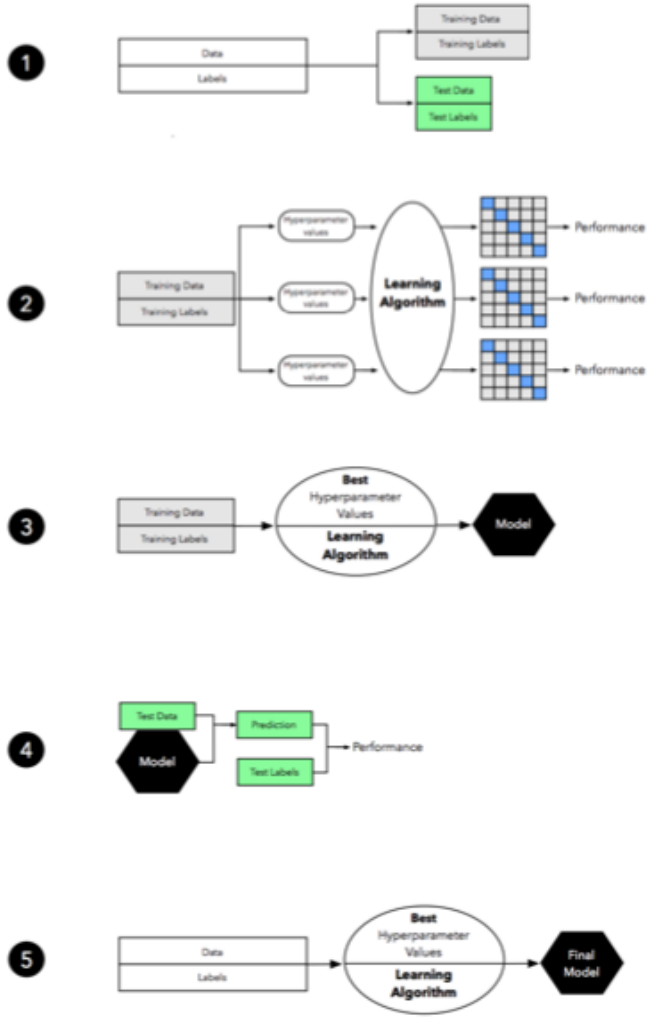

Figure 16: Illustration of *k*-fold cross-validation for model selection.

3.8 A NOTE ABOUT MODEL SELECTION AND LARGE DATASETS

3-way holdout is commonly used for model evaluation in deep learning literature

- Possibly due to it being computationally cheaper
- Also when using deep learning models we typically have relatively large sample sizes where we do not have to worry as much about high variance or the sensitivity of our data splits.

It is alright to use the holdout method with training, validation, and test split over k-fold cross validation when datasets are relatively large.

- I actually see this method used often in Kaggle Competitions

3.9 A NOTE ABOUT FEATURE SELECTION DURING MODEL SELECTION

Feature Selection is typically done inside the cross-validation loop instead of before splitting the data into folds.

- Feature selection inside the cross validation reduces bias (avoids looking at test data during the training stage)
- May lead to an overly pessimistic estimate (Less data available)

3.10 LAW OF PARSIMONY

Two Conflicting views

- **Occam's Razor (Law of Parsimony) –** "Among Competing Hypothesis, the one with the fewest assumptions should be selected."
- Occams Razor can be applied with the one-standard error method

  Consider the numerically optimal estimate and standard error

  Select the model with performance that is within one standard error of step 1
- **Pedro Domingos –** "Sometimes the simplest hypothesis consistent with the data is less accurate for prediction than a more complicated one. Some of the most powerful learning algorithms output models that seem gratuitously elaborate? – sometimes even continuing to add to them after they've perfectly fit the data – but that's how they beat the less powerful ones."

There are many reasons for preferring a simpler model although it may not be the most accurate:

- Computationally more efficient
- Easier to implement
- Easier to understand/reason

Lets consider another example dataset to illustrate Occam's Razor and the one-standard error method:

- 300 Samples
- Concentric circles
- Uniform class distribution (class 1 (150), class 2 (150))

The data is then split (70 % training set, 30 % test set)

- 210 samples from the training samples are shown in figure 17

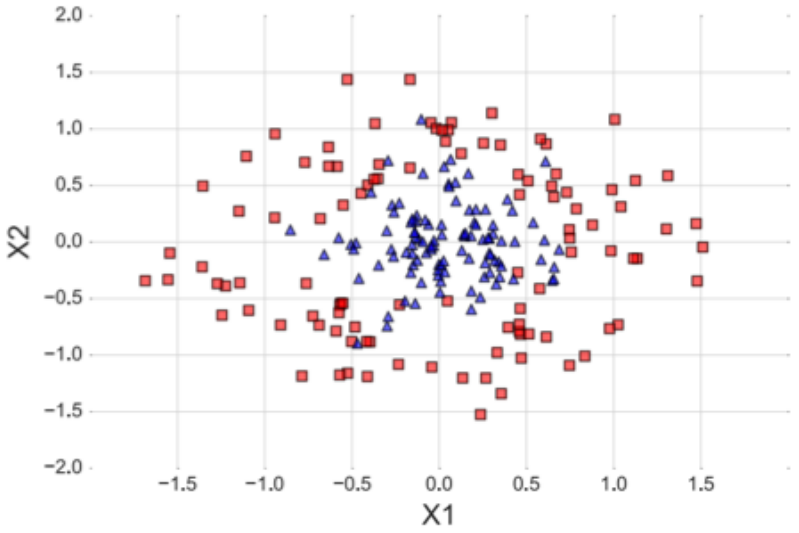

Figure 17: Concentric circles dataset with 210 training examples and uniform class distribution.

3.10 LAW OF PARSIMONY

Lets assume we our goal is to optimize gamma γ hyperparameter of Support Vector Machine (SVM) with a non-linear Radial Basis Function-kernel (RBF-kernel) on the previous dataset.

Gamma can be thought of a hyperparameter that controls influence of single training samples influence on the decision boundary.

As can be seen in figure 18-

- γ Gamma 0.1- 100 : 80 % accuracy or more
- γ Gamma 10.0 : Complex Decision Boundary
- γ Gamma .001 : Simple Decision boundary (cannot separate the classes)

γ Gamma 0.1 Seems like a good trade-off in this case between complexity and accuracy.

- Performance of the corresponding model falls within one standard error of the best performing model with γ = 0 or γ = 10.

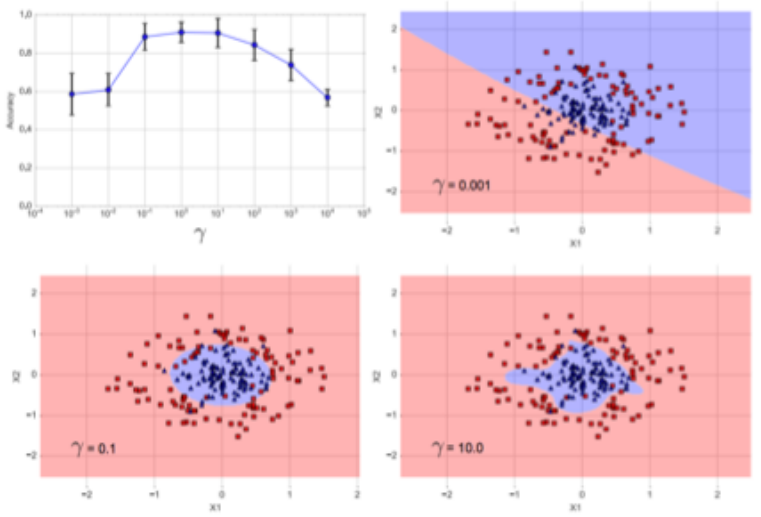

Figure 18: Performance estimates and decision regions of an RBF-kernel SVM with stratified 10-fold cross-validation for different γ values. Error bars represent the standard error of the cross-validation estimates.

CONCLUSION

- Figure 23 summarizes the author's personal recommendations based on the literature that was reviewed.
- In an ideal world we would have unlimited data but since in the real world the size of data is limited we can rely on the statistical tests in this article as an aid to decision making.
- Gael Varoquaux – "Cross-Validation is not a silver bullet. However, it is the best tool available, because it is the only non-parametric method to test for model generalization.

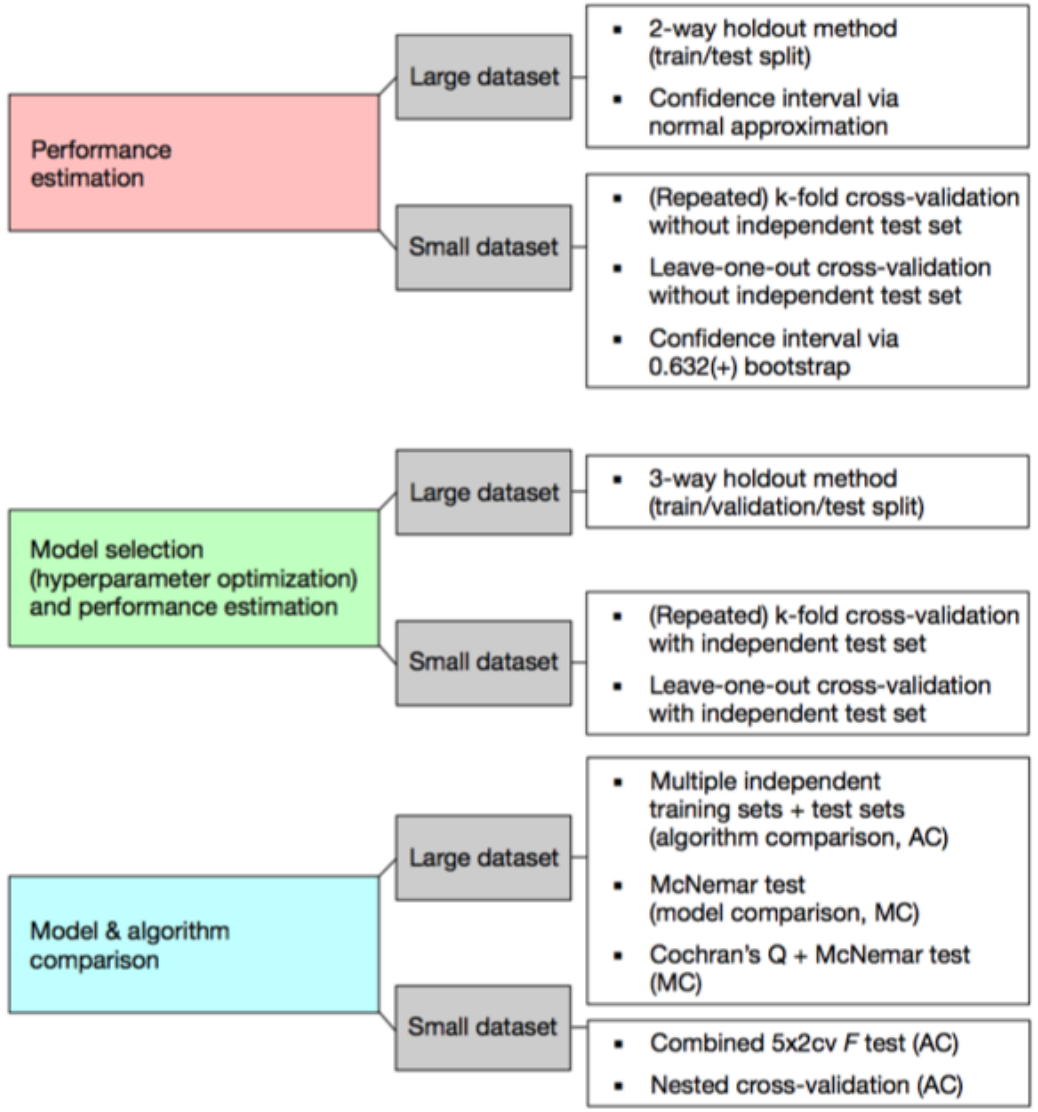

Figure 23: A recommended subset of techniques to be used to address different aspects of model evaluation in the context of small and large datasets. The abbreviation "MC" stands for "Model Comparison," and "AC" stands for "Algorithm Comparison," to distinguish these two tasks.