Cross-Validation and Hyper-parameter Search

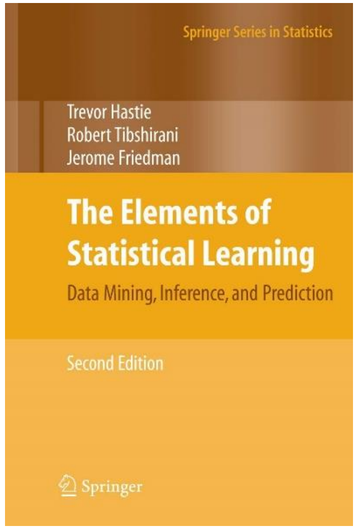
Assoc. Prof. Karl Ezra Pilario, Ph.D.
Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

Outline

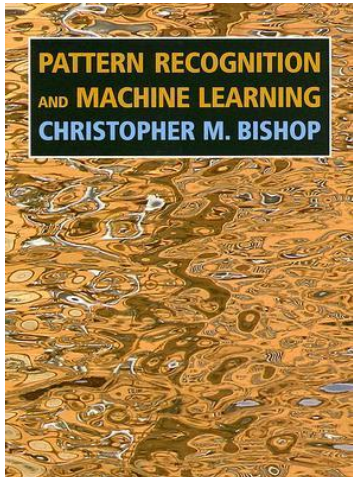
- How to Validate Models?
    - Holdout Validation
    - K-Fold Cross-Validation
    - Other Variants

- Hyper-parameter Search Methods
    - Grid Search
    - Random Search
    - Optuna

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2^{nd} Ed. Springer.



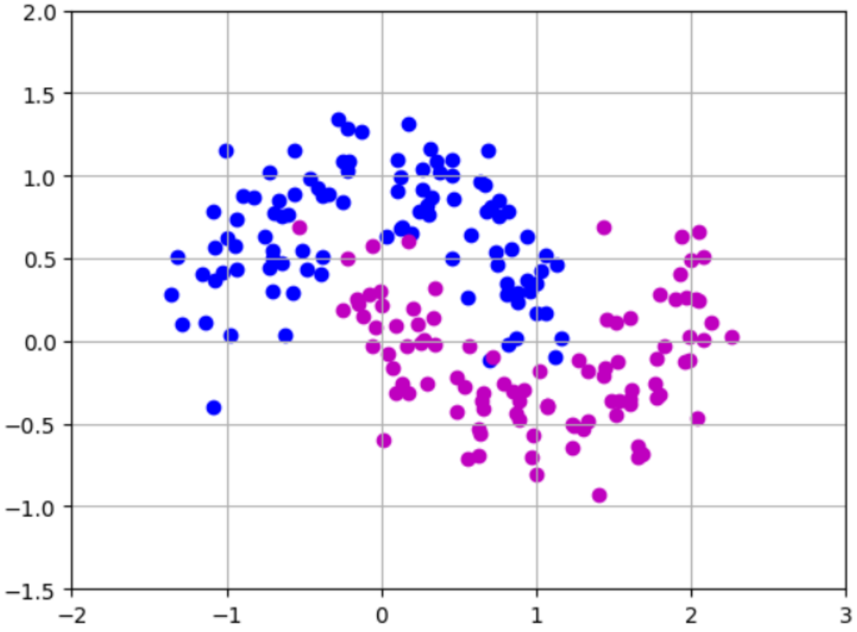
Bishop (2006)
Pattern Recognition and Machine Learning. Springer.



Why validate our models?

Example 1: Two Moons Data Set

Given the following binary classification data set, fit an SVM classifier with $C = 20$ and default kernel in Scikit-learn.



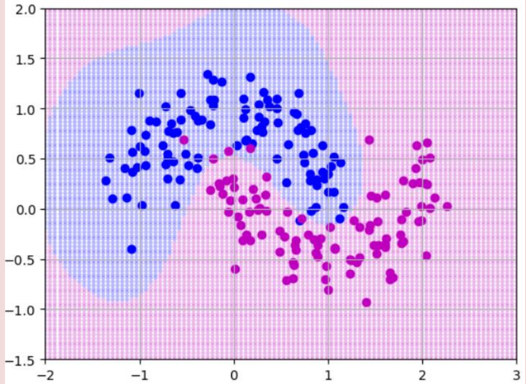
SVM without Validation

- All data points were used to train the SVM. No data points are left to test if the SVM would work well in practice!

97.0%
Accuracy

99	1
5	95

Confusion matrix



SVM with Validation

- Prior to analysis, the data was split:
  - **70% Training, 30% Testing**

- Reported accuracies:

97.8%
Training accuracy

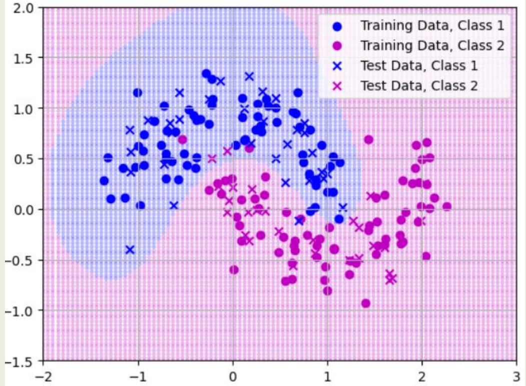
93.3%
Test accuracy

67	1
2	70

Training confusion matrix

30	2
2	26

Test confusion matrix



Model Validation

What we did here is called Holdout Validation.

SVM with Validation

- Prior to analysis, the data points were split:
  - **70% Training, 30% Testing**
- Reported accuracies:

97.8%
Training accuracy

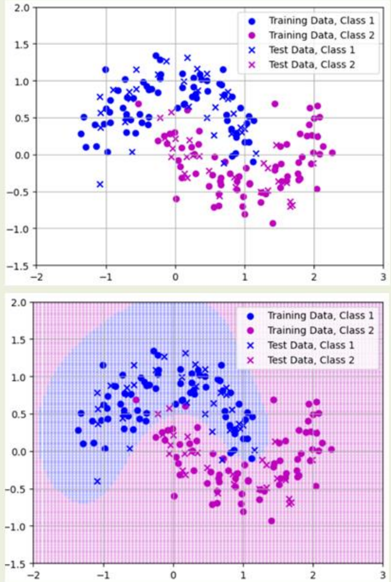
93.3%
Test accuracy

67	1
2	70

Training confusion matrix

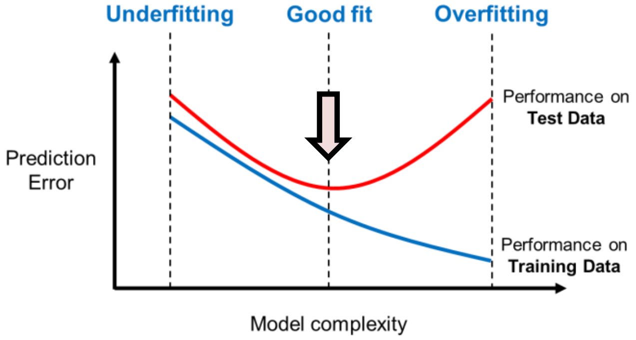
30	2
2	26

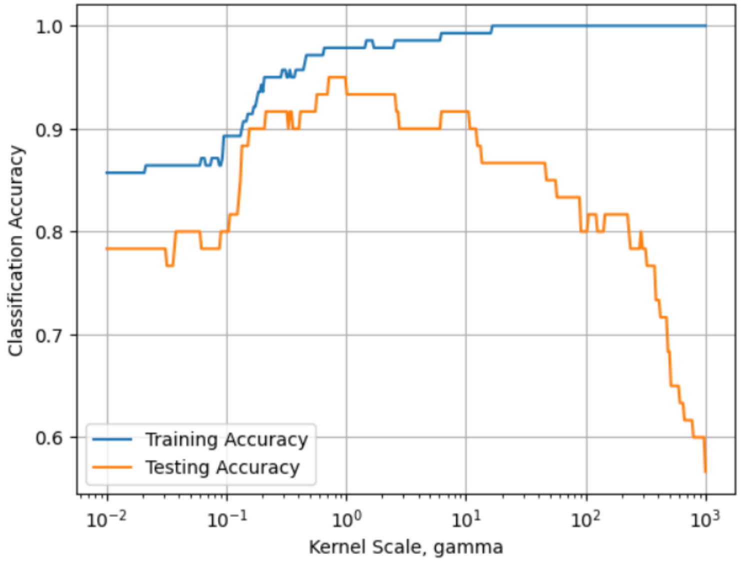
Test confusion matrix



Holdout Validation

- Split data set into Training and Testing.
- Split can be 70%-30%, 60%-40%, or 85%-15%.
- Test data are "held out" from the training phase.
- Avoid data leakage! Test data should be independent from training data.
- We can investigate the testing accuracy while varying a hyper-parameter in the model:
  - In our example, here are the SVM training and testing accuracies at varying values of the *RBF kernel scale*, all others being constant.
  - We can see the regions where SVM underfits and overfits.





Model Validation

Holdout Validation

- Train the model 1 time, validate it 1 time.
- Validation score is the score for the test data.

Train (70%)	Test (30%)

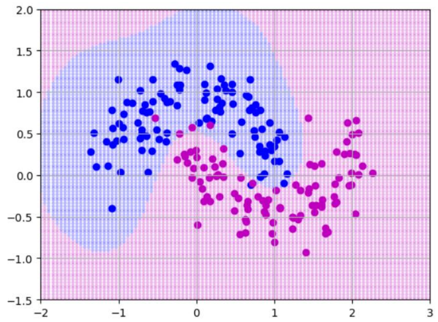
K-fold Cross-Validation

- Train the model K times, validate it K times.
- Overall cross-validation score is the average of K results.
- Data set is used more wisely than holdout.

				Test	5-fold
			Test		
		Test			
	Test				
Test					

Example 1: Two Moons Data Set

- No need to explicitly split the data in the code.
- We only need to specify the no. of splits, K.

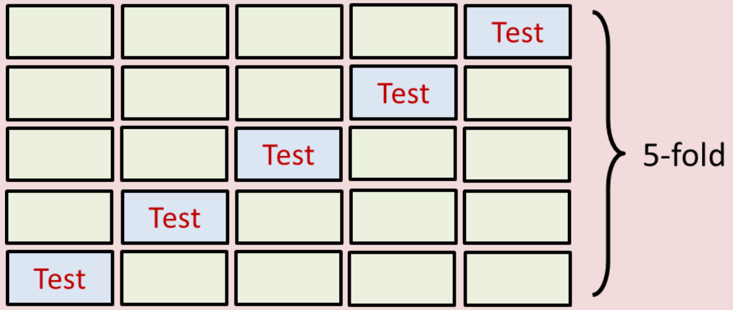


5-fold Test Accuracies	5-fold Cross-validation Score	10-fold Cross-validation Score
100%		
95%		
95%	**96%** ± 0.029	**96%** ± 0.032
92.5%		
97.5%		

Model Validation

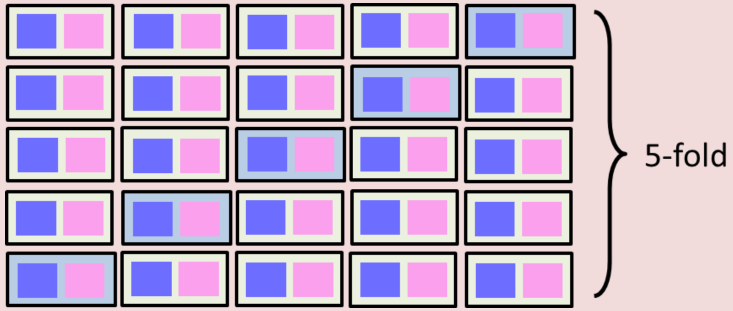
K-fold Cross-Validation

- Train the model K times, validate it K times.
- Overall cross-validation score is the average of K results.
- Data set is used more wisely than Holdout.

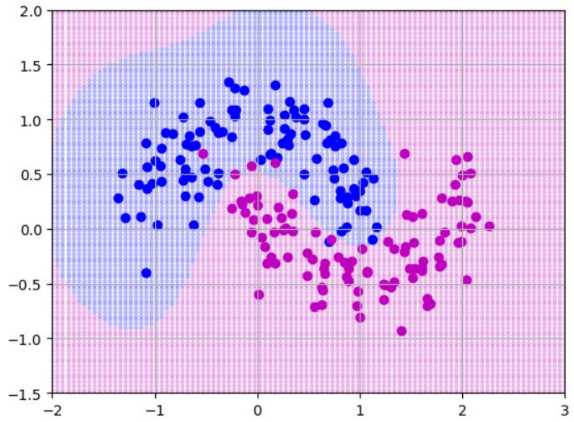


Stratified K-fold Cross-Validation

- Same as K-fold cross-validation but the percentage of samples from each class is *preserved*.



Example 1: Two Moons Data Set



Results for an SVM classifier with $C = 20$ and default kernel in Scikit-learn:

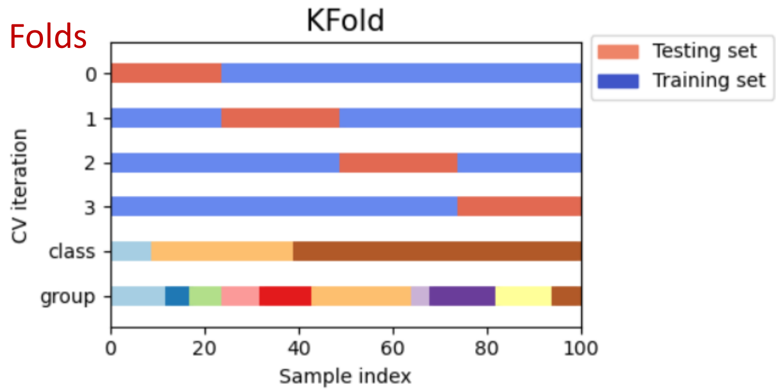
5-fold Cross-validation Score	10-fold Cross-validation Score	5-fold Stratified Cross-validation Score
96%	**96%**	**96.5%**
± 0.029	± 0.032	± 0.029

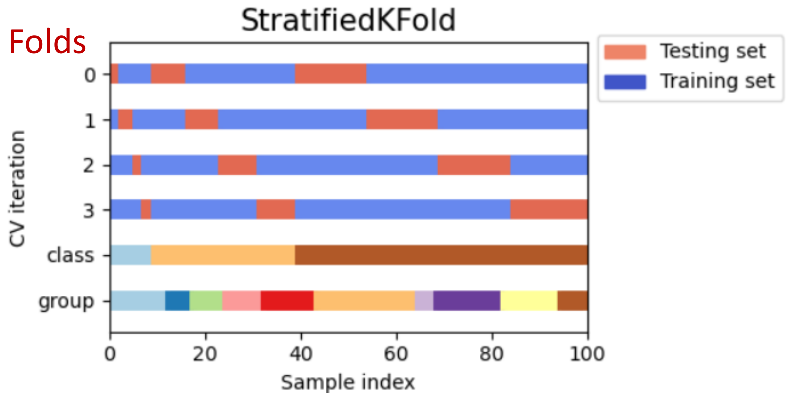
Model Validation

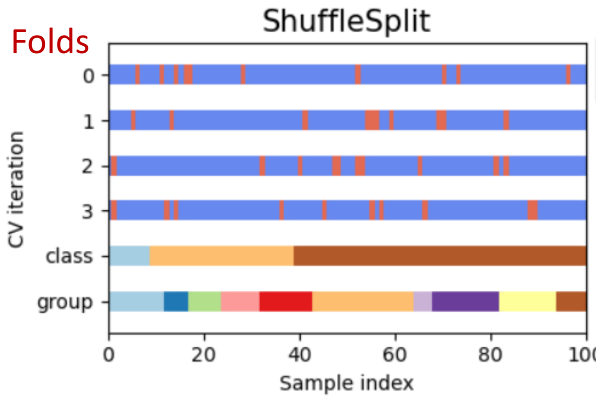
An example of “grouped” data is when medical samples are taken multiple times from one patient: Samples from one patient is one group.

Other Kinds of Validation Schemes

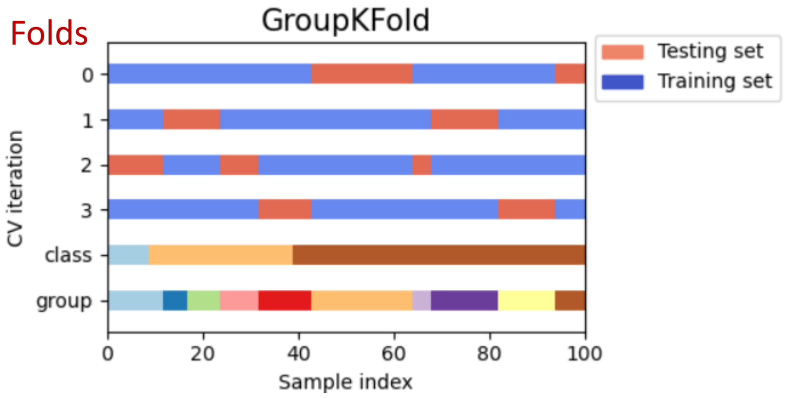
Say we have 100 randomly generated input datapoints, with 3 classes split unevenly across datapoints, and 10 “groups” split unevenly across datapoints. Different validation schemes utilize group and class info differently.

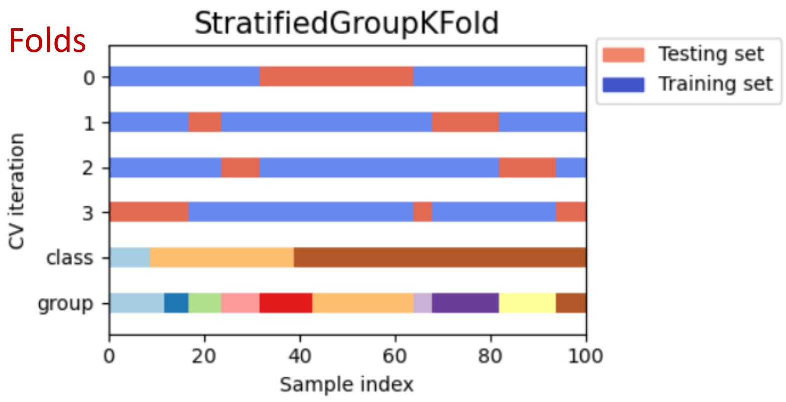


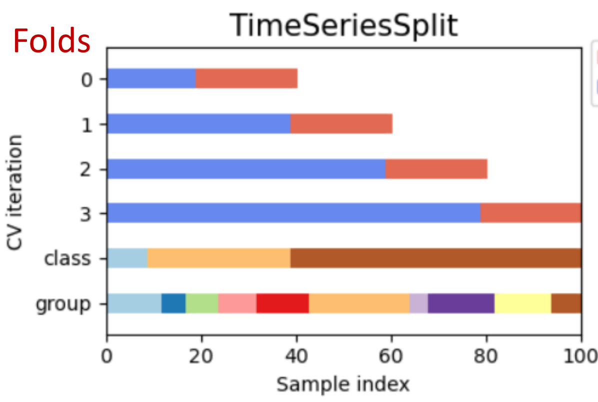




(Repeated Random Subsampling Validation or Monte Carlo Validation)

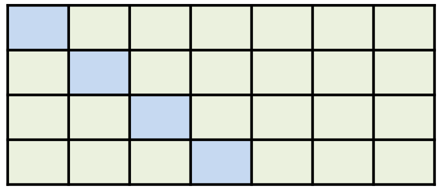






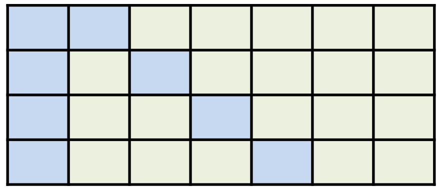
(Forward Chaining Validation)

Model Validation: Other Variants



Leave-one-out Cross-Validation (LOOCV)

- Same as **N**-fold cross-validation, where **N** is the number of samples.
- In each fold, **test data size is always 1**.
- **Total number of folds is always N**.



Leave-P-out Cross-Validation (LPOCV)

- Also similar with K-fold cross-validation.
- In each fold, **test data size is always P**.
- **Total number of folds is always Combination(N, P)**, since all possible ways to take P samples from N are exhausted.
- Difference between LPOCV and K-fold CV with K=N-P:
  LPOCV may create *overlapping* test data sets, but K-fold CV ensures that test data in each fold do not overlap.

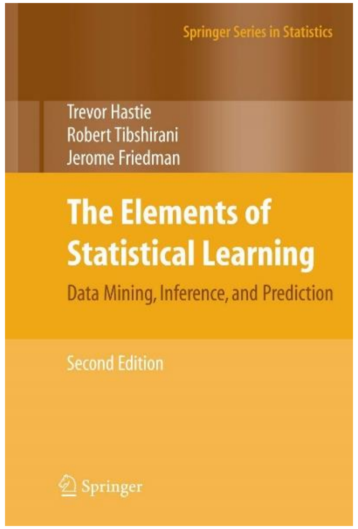
Results for an SVM classifier with $C = 20$ and default kernel in Scikit-learn:

5-fold Cross-validation Score	10-fold Cross-validation Score	5-fold Stratified Cross-validation Score	LOOCV Score	L-2-OCV Score	Monte Carlo Validation Score
96%	**96%**	**96.5%**	**96%**	**96%**	**96%**
± 0.029	± 0.032	± 0.029	± 0.196	± 0.139	± 0.029

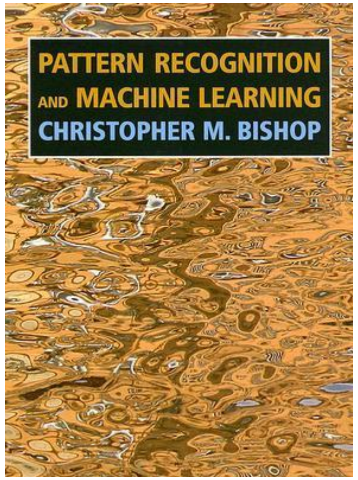
Outline

- How to Validate Models?
    - Holdout Validation
    - K-Fold Cross-Validation
    - Other Variants
- Hyper-parameter Search Methods
    - Grid Search
    - Random Search
    - Optuna

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



Bishop (2006)
Pattern Recognition and Machine Learning. Springer.



Hyper-parameter Tuning

Model Parameters

The knobs in the model that are tuned upon exposure to training data. (e.g. $\boldsymbol{w}, \boldsymbol{b}$)

Hyper-parameters

The knobs in the model that are tuned **prior** to exposure to training data. (e.g. λ, l)

- Often, hyper-parameters are just tuned manually based on user experience (e.g. heuristics).
- Different hyper-parameter settings can give different models.
- Their values control the training behavior itself.

Examples

	Model Parameters	Hyper-Parameters
Linear Regression	Weights, w	Regularization parameter, λ Type of regularization
Logistic Regression	Weights, w	Regularization parameter, λ Type of regularization Solver
Locally Weighted Regression	Weights, w	Weighting function, ω Bandwidth, τ
Support Vector Classifier	Dual variables, α Bias, b	Kernel type Kernel scale Box constraint Multi-class strategy

Hyper-parameter Tuning

Model Parameters

The knobs in the model that are tuned upon exposure to training data. (e.g. $\boldsymbol{w}, \boldsymbol{b}$)

Hyper-parameters

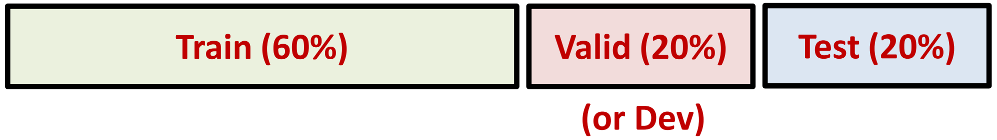
The knobs in the model that are tuned **prior** to exposure to training data. (e.g. λ, l)

To perform hyper-parameter tuning, we now need a **validation data set** aside from the training and test data sets.

Holdout Validation: *Revisited*

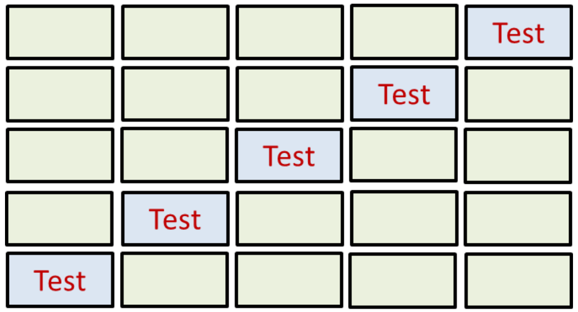
If we wish to **validate** a single model:

If we wish to **tune** a model's hyper-parameters:

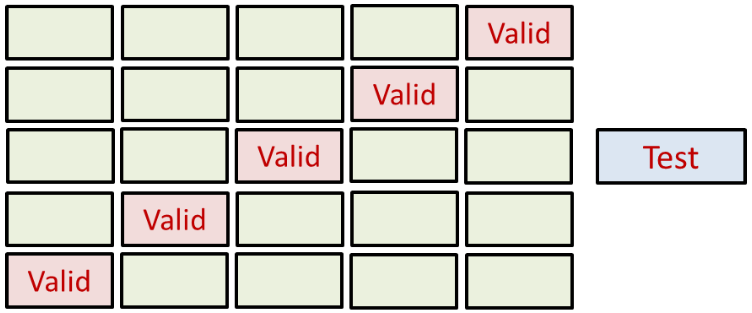


K-fold Cross-Validation: *Revisited*

If we wish to **validate** a single model:

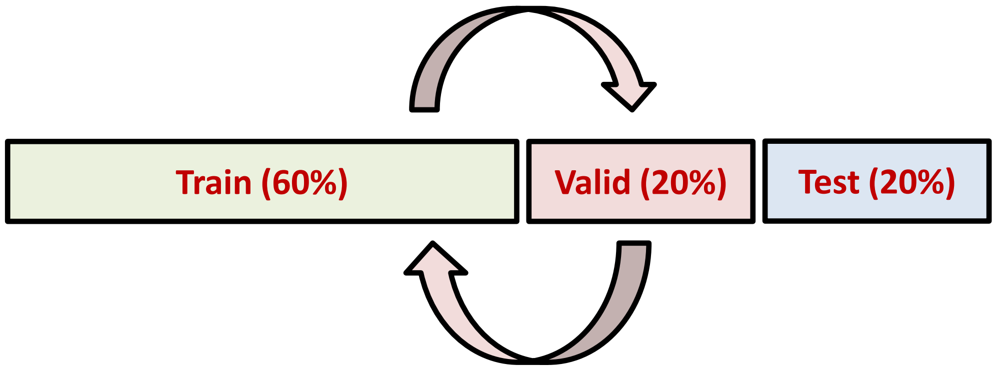


If we wish to **tune** a model's hyper-parameters:

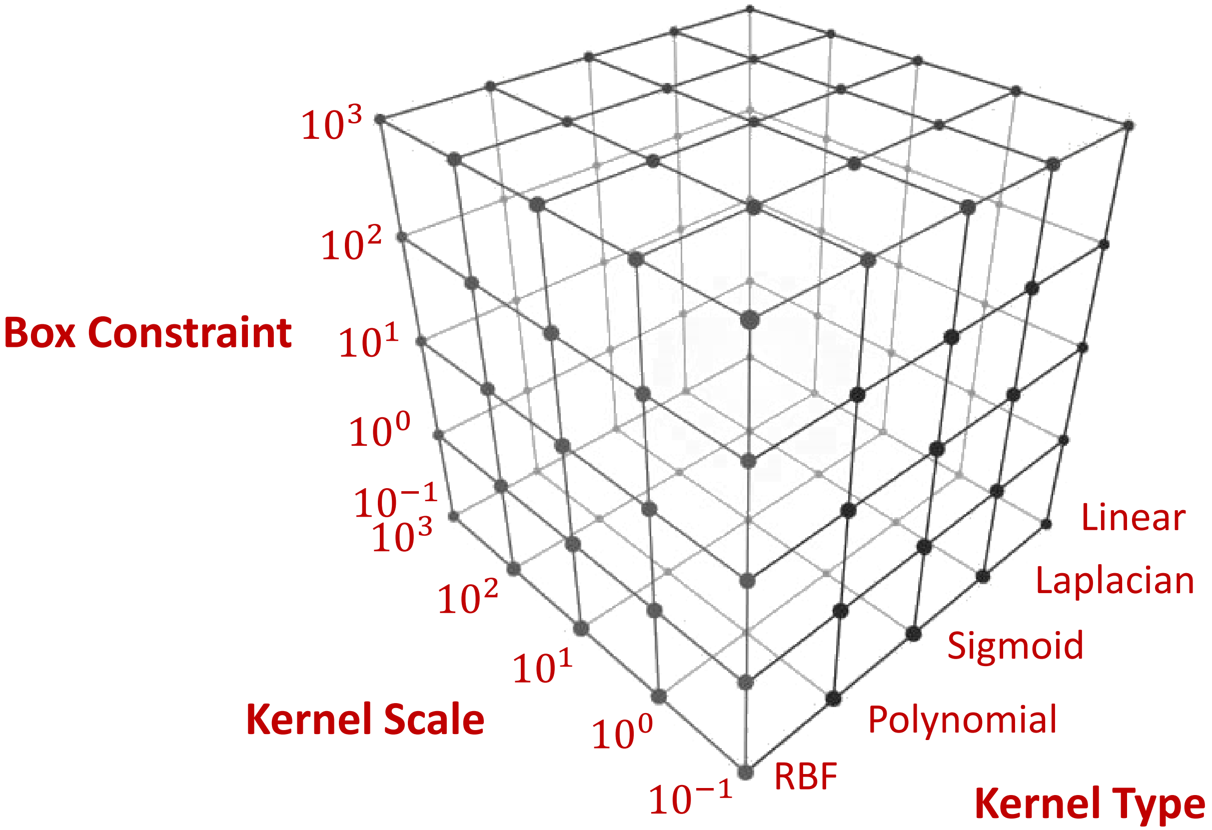


Hyper-parameter Tuning: Grid Search

1. Define a hyper-parameter grid.
2. For each hyper-parameter combination,
   a. Train a candidate model using training data.
   b. Compute its performance on the validation data.
3. Report the hyper-parameter combination with the *highest score* on validation data.
4. Make one final validation on test data.

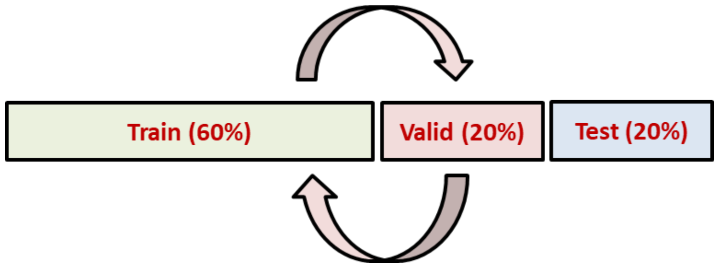


For example, in the Support Vector Classifier:



Hyper-parameter Tuning: Random Search

- Random Search acknowledges that some hyper-parameters are unimportant to investigate.
- Instead of checking the entire grid of candidate parameters, searching randomly inside the search space is more efficient (Bergstra and Bengio, 2012).
- Search range can be a distribution rather than just a list of values as in Grid Search.
- Only need to specify how many times (n_iter) we sample the search space.



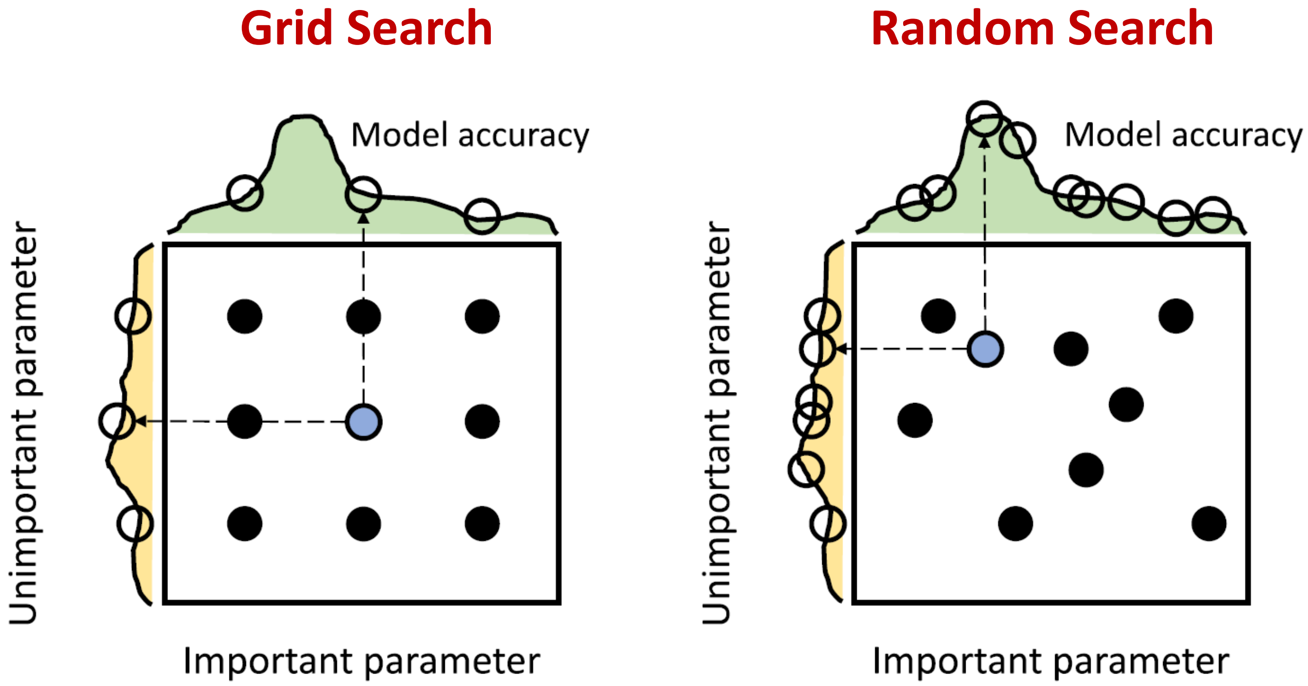


Figure from Pilario et al., (2021). A Kernel Design Approach to Improve Kernel Subspace Identification. IEEE Trans. On Industrial Electronics. 68(7): 6171-6180.

Hyper-parameter Tuning

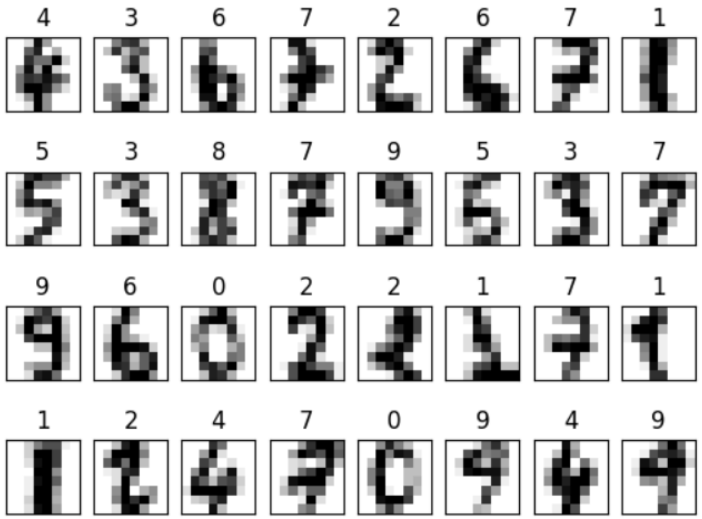
Example 2: 8x8 Handwritten Digit Recognition

Source: https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

- Each datapoint is an 8x8 image of a digit.
- Available in scikit-learn datasets as "load_digits."
- Other details:
  - Classes: 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  - Samples per class: ~180
  - Total no. of samples: 1797
  - Dimensionality: 64
  - Feature values: 0 to 16

Tune an SVM with the following hyper-parameter value grid candidates:

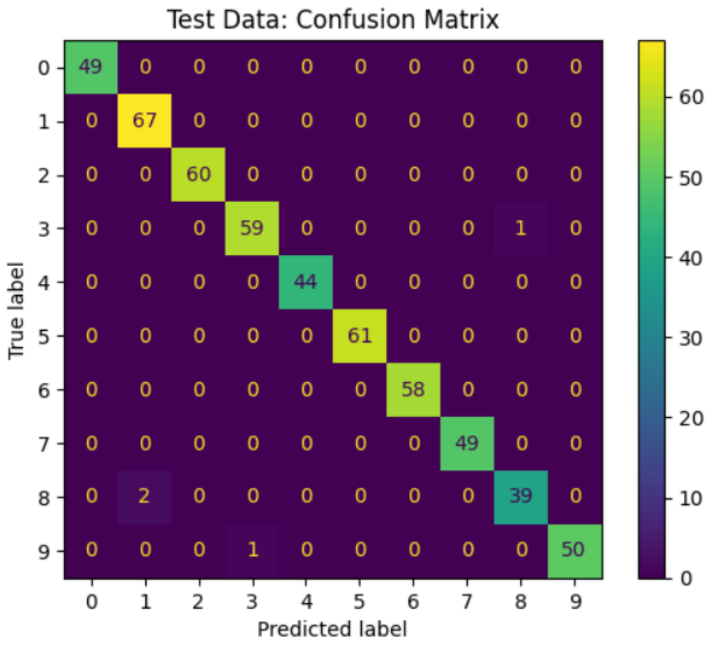
C: $10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}$
Gamma: $10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}$
Kernel: RBF, Poly, Linear, Sigmoid



Grid Search Results

Best SVM:
C = 1
Gamma = 0.001
Kernel = RBF

Accuracy: 0.99
No. of Trials: 100
Runtime: 60.83 sec



Note: GridSearchCV uses the following defaults:

- Score is based on Stratified 5-fold cross-validation.
- For classification, the performance metric being optimized is the accuracy score.
- The multi-class strategy is one-vs-rest.

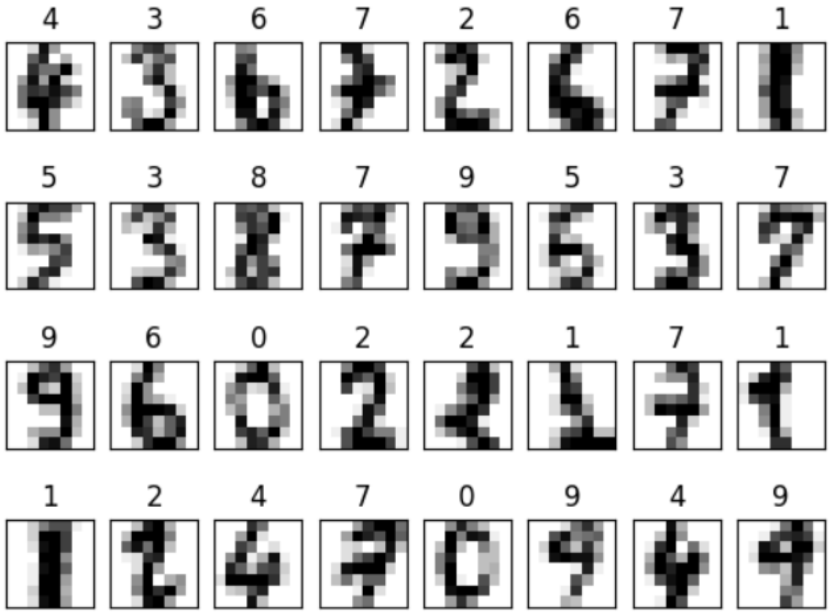
Hyper-parameter Tuning

Example 2: 8x8 Handwritten Digit Recognition

Source: https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

Tune an SVM with the following hyper-parameter value grid candidates:

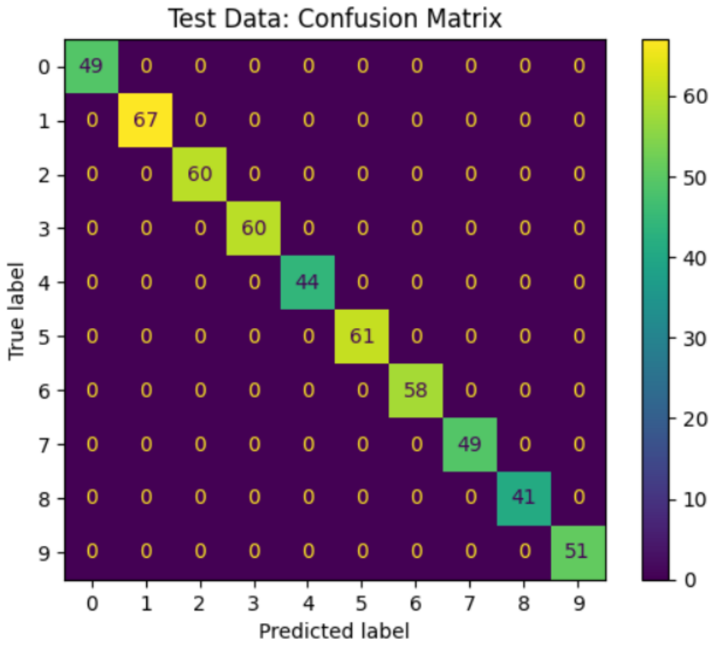
C: Exponential Distribution $(\lambda = 100)$
Gamma: Exponential Distribution $(\lambda = 0.1)$
Kernel: RBF, Poly, Linear, Sigmoid



Random Search Results

Best SVM:
C = 105.31
Gamma = 0.00068
Kernel = RBF

Accuracy: 1.00
No. of Trials: 20
Runtime: 23.90 sec



Random Search proves to be more efficient than Grid Search in this example.

Hyper-parameter Tuning: Optuna

Optuna: A hyperparameter optimization framework

Akiba et al., (2019) Optuna: A Next-generation Hyperparameter Optimization Framework. https://arxiv.org/pdf/1907.10902.pdf

Optuna has modern functionalities as follows:

Lightweight, versatile, and platform agnostic architecture

- Handle a wide variety of tasks with a simple installation that has few requirements.

Pythonic search spaces

- Define search spaces using familiar Python syntax including conditionals and loops.

Efficient optimization algorithms

- Adopt state-of-the-art algorithms for sampling hyperparameters and efficiently pruning unpromising trials.

Easy parallelization

- Scale studies to tens or hundreds of workers with little or no changes to the code.

Quick visualization

- Inspect optimization histories from a variety of plotting functions.

- Grid Search implemented in `GridSampler`
- Random Search implemented in `RandomSampler`
- Tree-structured Parzen Estimator algorithm implemented in `TPESampler`
- CMA-ES based algorithm implemented in `CmaEsSampler`
- Algorithm to enable partial fixed parameters implemented in `PartialFixedSampler`
- Nondominated Sorting Genetic Algorithm II implemented in `NSGAIISampler`
- A Quasi Monte Carlo sampling algorithm implemented in `QMCSampler`

The default sampler is `TPESampler`.

Main algorithm:

TPE (Tree-structured Parzen Estimator)

- A variant of Bayesian Optimization

Hyper-parameter Tuning: Optuna

Main algorithm:

TPE (Tree-structured Parzen Estimator)

- A variant of Bayesian Optimization

Advantages:

- <u>BO is sample-efficient.</u> It is suited for black-box objective functions that are *expensive* to evaluate.
- <u>BO is gradient-free.</u> It does not need to calculate gradients of the objective function.
- <u>BO can easily control the trade-off between exploration and exploitation</u>.
- In particular, the **Tree-structured Parzen Estimator** can optimize <u>both categorical and continuous hyper-parameters</u>, where as GP-EI can only optimize continuous ones.

What is Bayesian Optimization (BO)?

A global optimization method that uses Bayes Theorem to sequentially direct the search for the optimum.

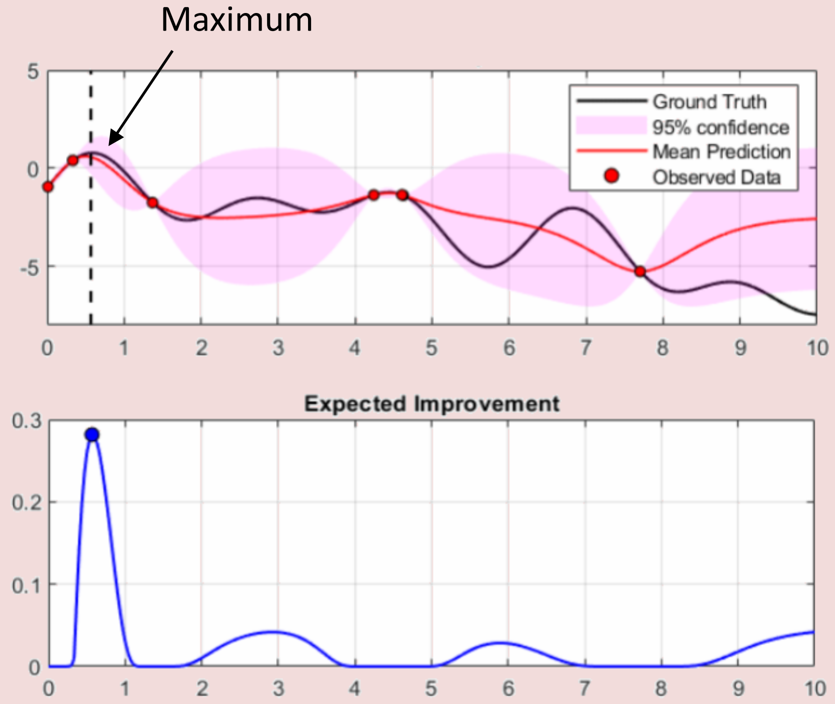
One variant of BO is the GP-EI (Gaussian process, Expected Improvement) scheme.

We'll learn about this in a future lecture.

Example:

In this example, the goal is to find the max value of the **black** curve.

- So far, we tested 6 samples from the black curve.
- A surrogate model (**red** curve) was fitted to these samples.
- The uncertainty of the surrogate model is given as the pink shaded area.
- A **blue** curve was generated from the surrogate's outputs, and it tells us where to optimally sample next.



Hyper-parameter Tuning

Example 2: 8x8 Handwritten Digit Recognition

Source: https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

Create a handwritten digit recognizer that optimizes between the best Logistic regression and best SVM model with the following hyper-parameters:

SVM

C: 10^{-1} to 10^{3} (log-scale)

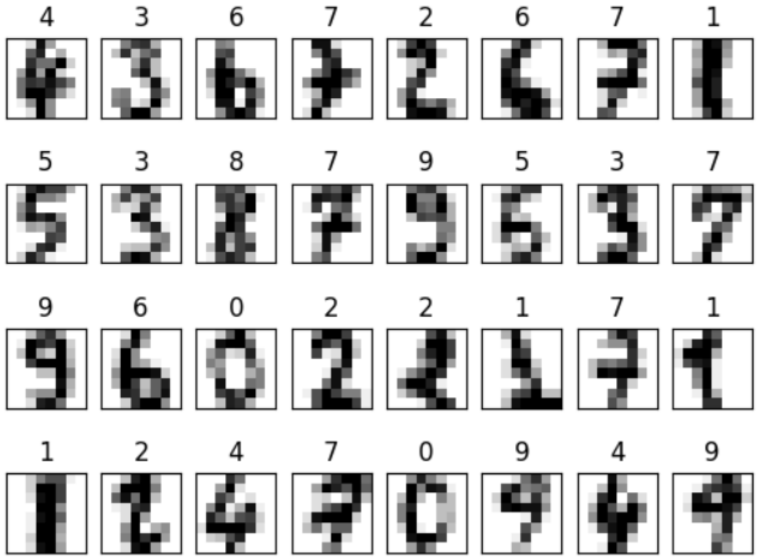
Gamma: 10^{-4} to 10^{1} (log-scale)

Kernel: RBF, Poly, Linear, Sigmoid

Logistic Regression

Penalty: L1, L2

Regularization: 10^{-2} to 10^{2} (log-scale)



Optuna Results

Best Model:

SVM

C = 100.7

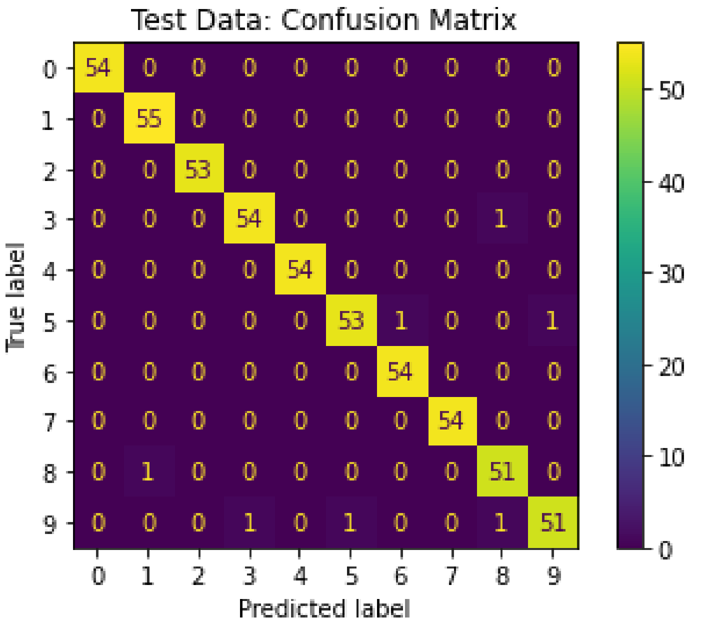
Gamma = 0.0002

Kernel = RBF

Accuracy: 0.987

No. of Trials: 100

Runtime: 41.52 sec



	0	1	2	3	4	5	6	7	8	9
0	54	0	0	0	0	0	0	0	0	0
1	0	55	0	0	0	0	0	0	0	0
2	0	0	53	0	0	0	0	0	0	0
3	0	0	0	54	0	0	0	0	1	0
4	0	0	0	0	54	0	0	0	0	0
5	0	0	0	0	0	53	1	0	0	1
6	0	0	0	0	0	0	54	0	0	0
7	0	0	0	0	0	0	0	54	0	0
8	0	1	0	0	0	0	0	0	51	0
9	0	0	0	1	0	1	0	0	1	51

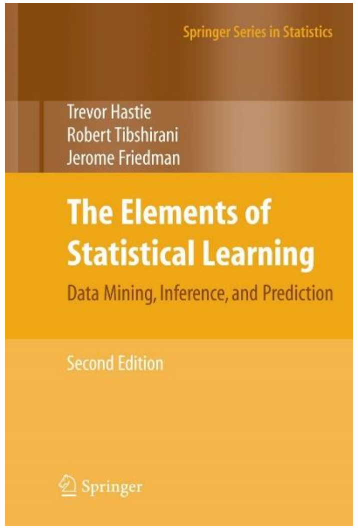
You can add more models and let Optuna find which of them is best, while each of them are also tuned!

Outline

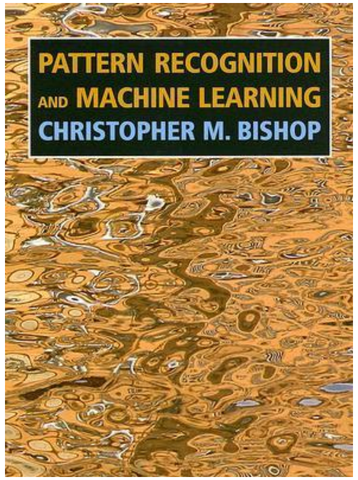
- How to Validate Models?
    - Holdout Validation
    - K-Fold Cross-Validation
    - Other Variants

- Hyper-parameter Search Methods
    - Grid Search
    - Random Search
    - Optuna

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



Bishop (2006)
Pattern Recognition and Machine Learning. Springer.



Further Reading

- https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html
- https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
- https://www.mygreatlearning.com/blog/gridsearchcv
- https://www.kaggle.com/code/vitalflux/k-fold-cross-validation-example/notebook
- R. Bharat Rao, G. Fung, R. Rosales, On the Dangers of Cross-Validation. An Experimental Evaluation, SIAM 2008.
- Optuna paper: https://arxiv.org/pdf/1907.10902.pdf
- https://neptune.ai/blog/optuna-guide-how-to-monitor-hyper-parameter-optimization-runs
- Hyperopt paper: https://pdfs.semanticscholar.org/d4f4/9717c9adb46137f49606ebbdf17e3598b5a5.pdf
- Frazier, 2018. A Tutorial on Bayesian Optimization: https://arxiv.org/abs/1807.02811
- Hyper-parameter Optimization: https://medium.com/criteo-engineering/hyper-parameter-optimization-algorithms-2fe447525903
- Bergstra, James S., Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. “Algorithms for hyper-parameter optimization.” In *Advances in neural information processing systems*, pp. 2546–2554. 2011.
- Bergstra, James, and Yoshua Bengio. “Random search for hyper-parameter optimization.” *Journal of Machine Learning Research* 13, no. Feb (2012): 281–305.
- Yang, L. and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316. https://www.sciencedirect.com/science/article/pii/S0925231220311693