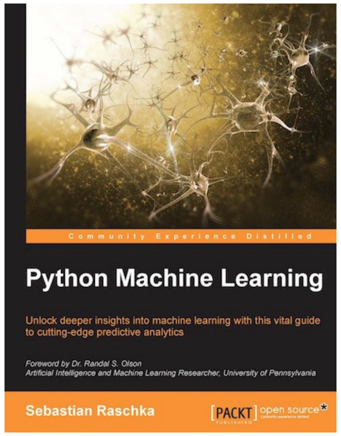
Lecture 8: Evaluating Machine Learning Models

Disclaimer

The following slides are partially based on:


Community Experience Distilled

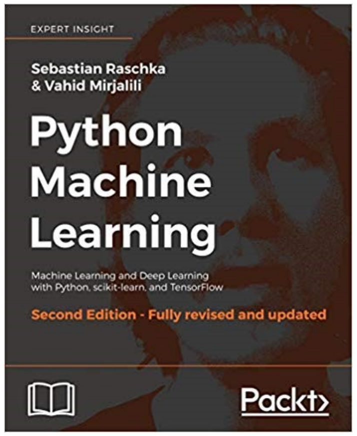
Python Machine Learning

Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics

Foreword by Dr. Randal S. Olson
Artificial Intelligence and Machine Learning Researcher, University of Pennsylvania

Sebastian Raschka

[PACKT] open source


EXPERT INSIGHT

Sebastian Raschka
& Vahid Mirjalili

Python Machine Learning

Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow

Second Edition - Fully revised and updated

Packt>

Important Steps

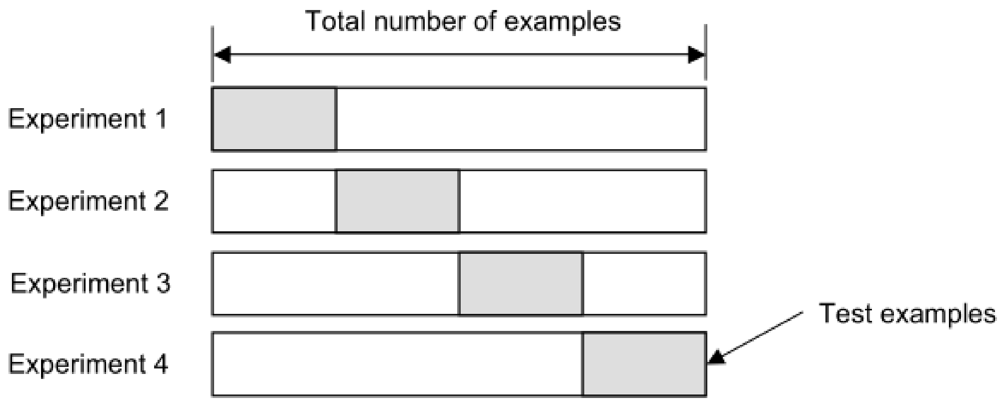
1. Determine relevant features (expert knowledge)
2. Collect data
3. **Split labeled data into training and test datasets**
4. Use training data to train machine learning algorithm.
5. Predict labels of examples in test data,
6. Evaluate algorithm.

Test Data

- Remember: if we reuse the same test dataset over and over again during model selection,
  - Test data will become part of our training data and thus the model will be more likely to overfit.
  - The reported performance will not be correct
- Despite this issue, many people still use the test set for model selection, which is NOT a good machine learning practice.

How to Split Data?

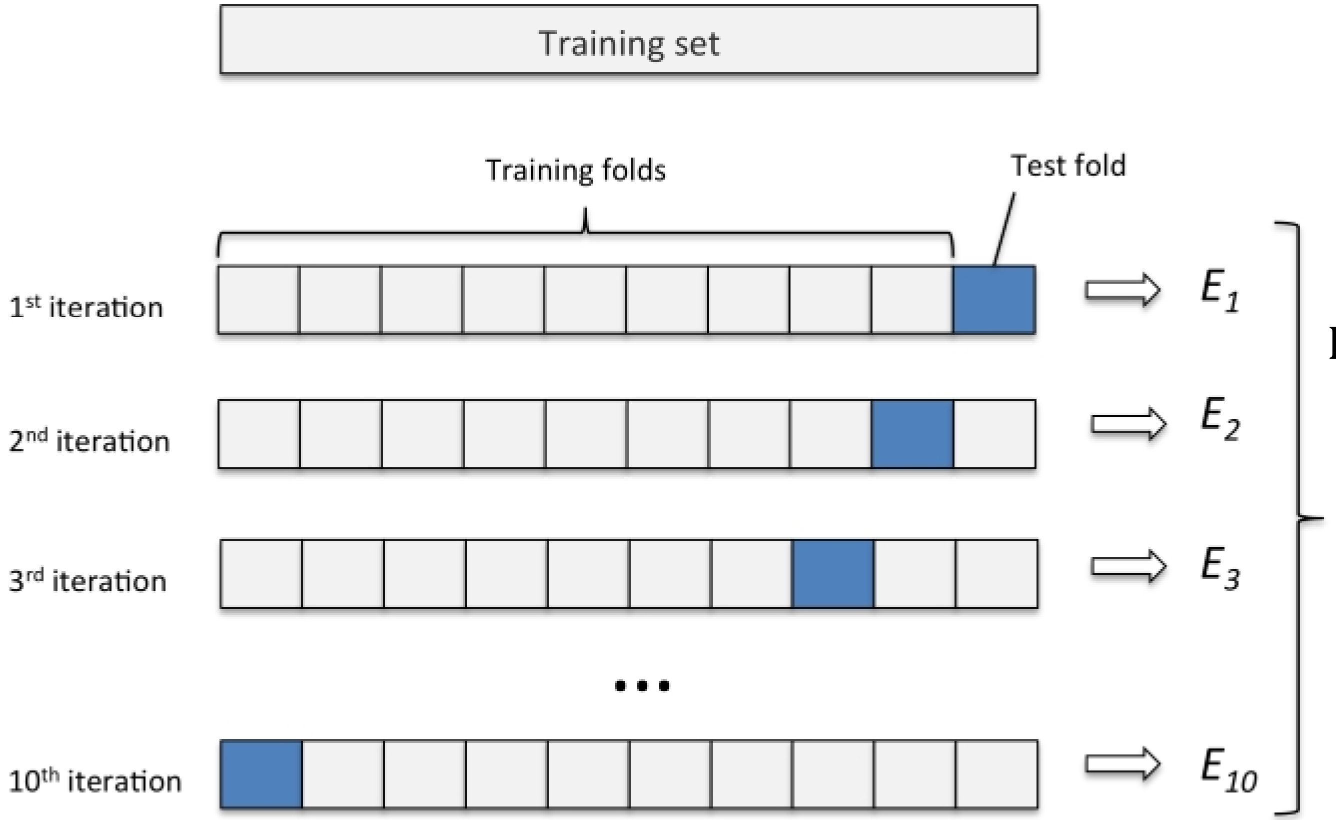
- Holdout
  - Training set
  - (validation set)
  - Test set
- K-fold Cross-validation
  - E.g. 10 folds cross-validation



Methods of Sampling

- Holdout
  - E.g. Reserve 2/3 for training and 1/3 for testing
- Random subsampling
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Stratified sampling
- Bootstrap
  - Sampling with replacement

Cross-Validation



E= Average Error

$$E = \frac{1}{10}\sum_{i=1}^{10} E_i$$

E_i=Individual Error

K-fold

- K =10 is typical
- If working with very small datasets, use leave-one-out (LOO)
  - If data from subjects, leave-one-subject-out
- cross_val_score() in scikit-learn
  - n_jobs can be set to distribute the evaluation across multiple CPUs
  - See Notebook

Stratified k-fold

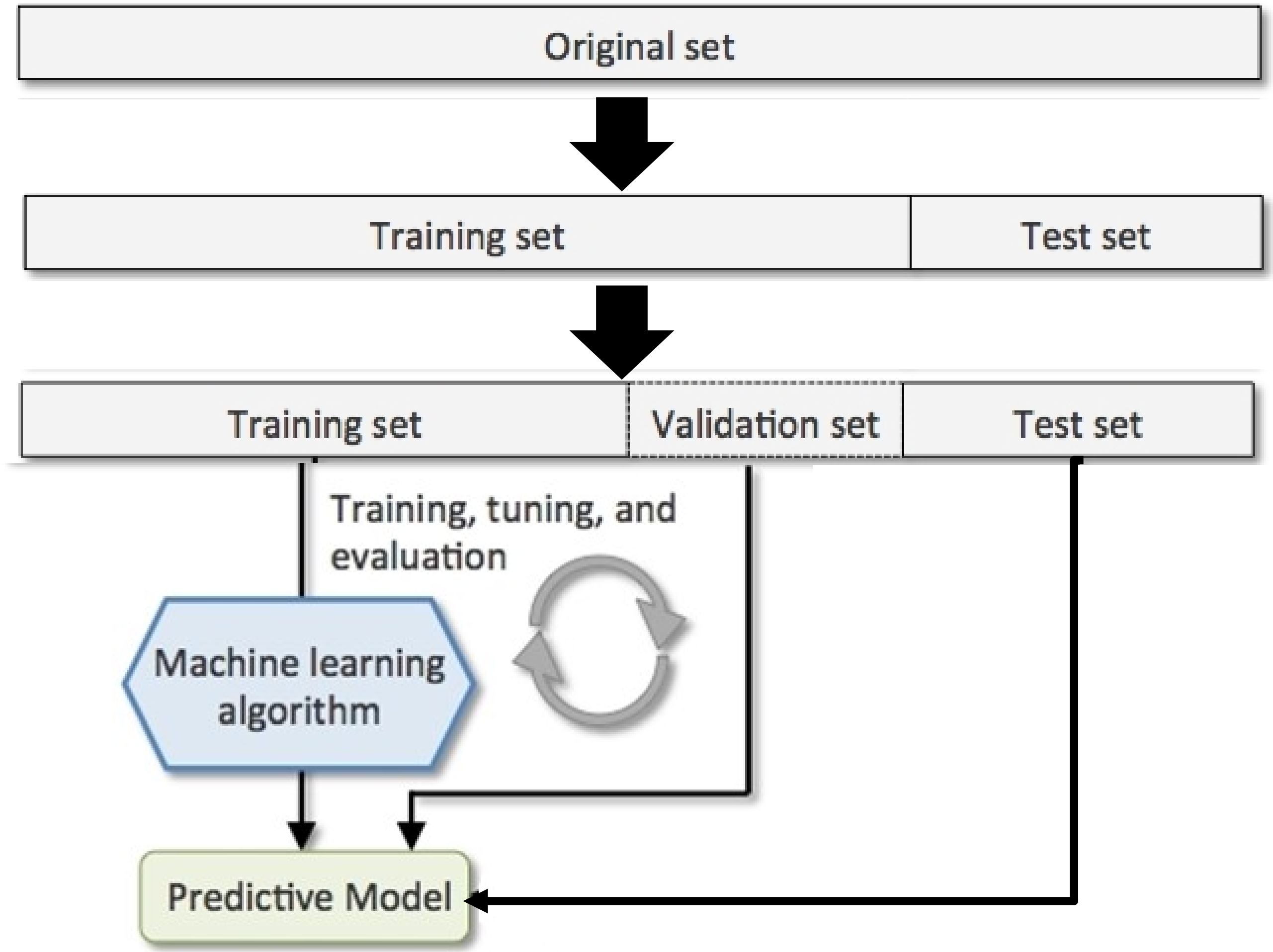
- The class proportions are preserved in each fold to ensure each fold is representative

Recommended Approach

- Separate your data into three parts
  - training, validation, testing
- Use either
  - Holdout + k-folds cross-validation
  - Nested cross-validation

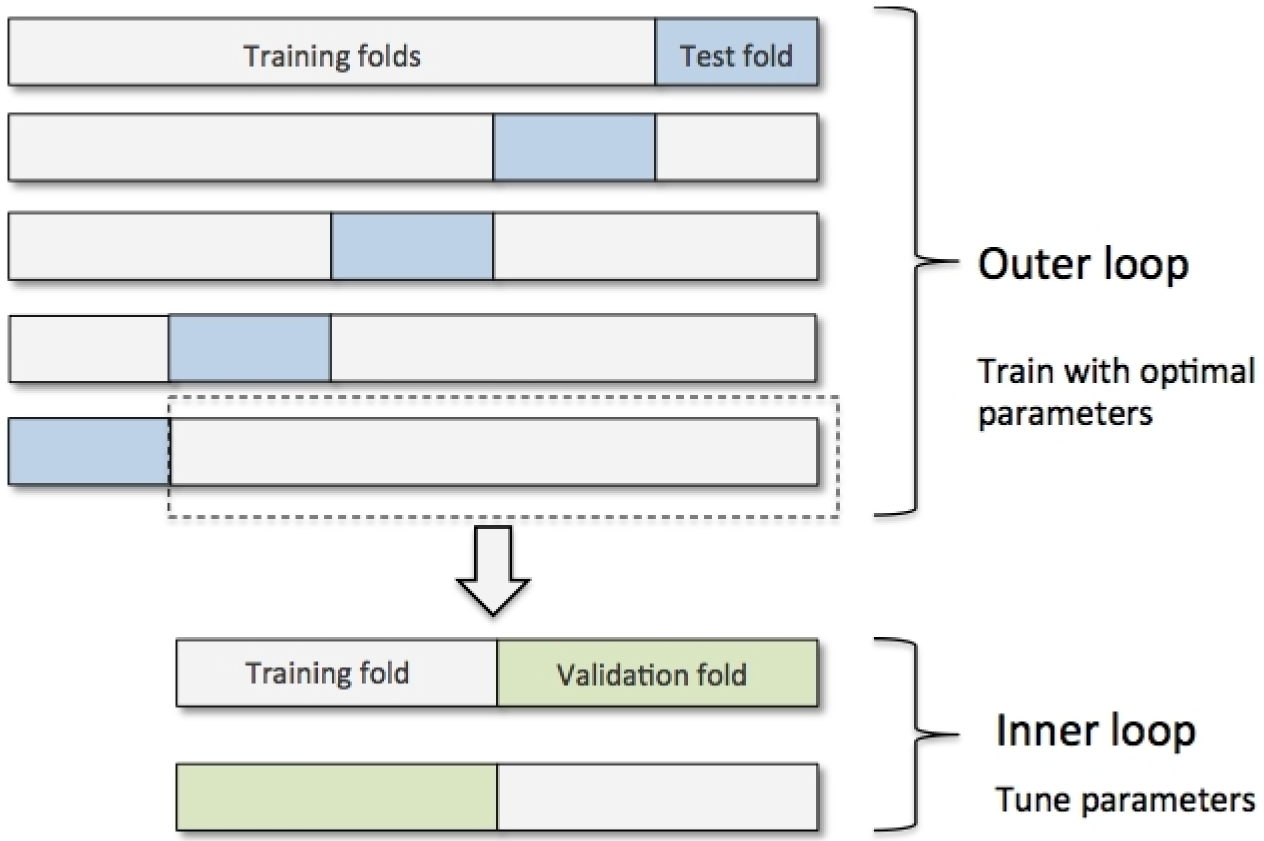
- Note: once you find satisfactory hyper-parameter values, you can retrain the model on the complete training set and obtain a final performance estimate using the independent test set.

Hold-out + Cross-Validation



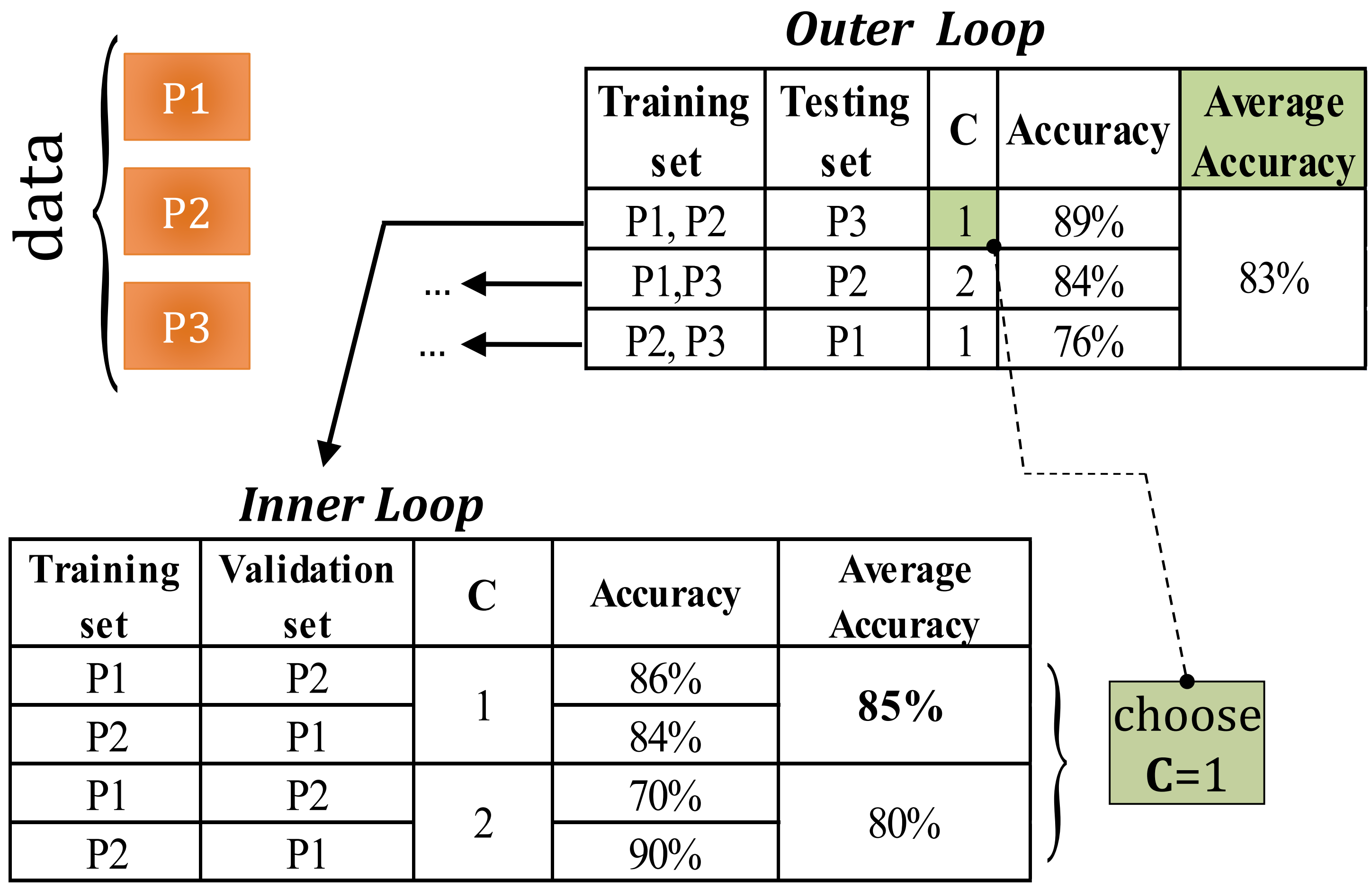
5×2 Cross-validation

- A form of nested variation



Example of 3*2 nested cross-validation

Consider that we use 3-fold cross-validation and we want to optimize parameter C that takes values “1” and “2”.

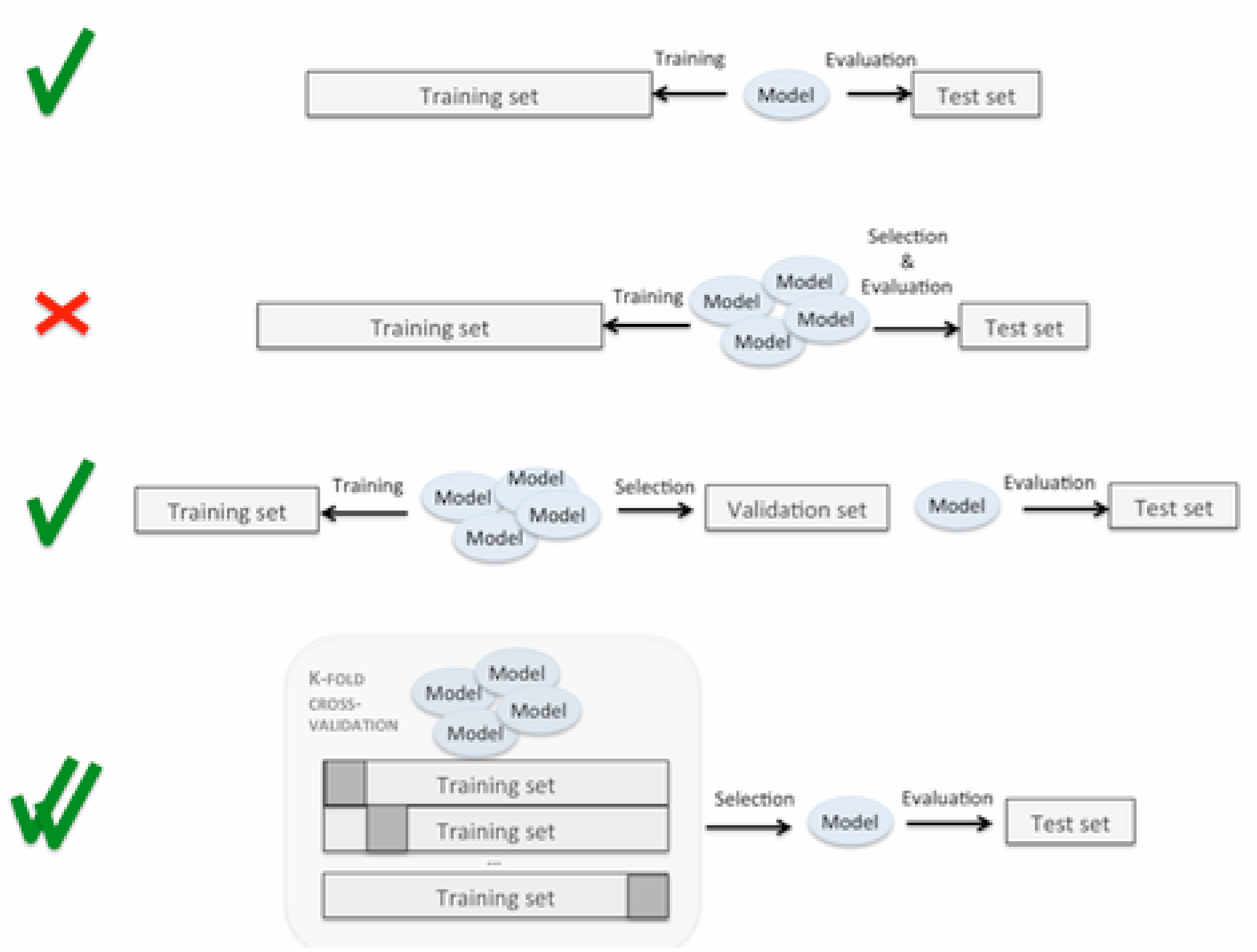


Outer Loop

Training set	Testing set	C	Accuracy	Average Accuracy
P1, P2	P3	1	89%	83%
P1,P3	P2	2	84%	
P2, P3	P1	1	76%	

Inner Loop

Training set	Validation set	C	Accuracy	Average Accuracy
P1	P2	1	86%	**85%**
P2	P1		84%	
P1	P2	2	70%	80%
P2	P1		90%	



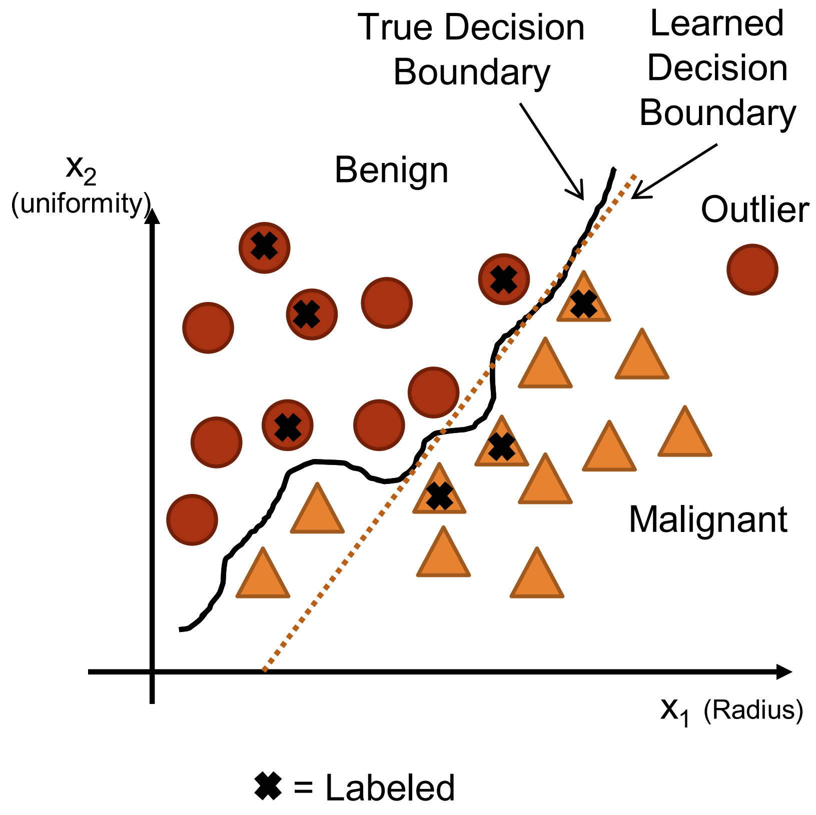
tp://sebastianraschka.com/faq/docs/evaluate-a-model.html

Important Steps

1. Determine relevant features (expert knowledge)
2. Collect data
3. Split labeled data into training and test datasets
4. **Use training data to train machine learning algorithm.**
5. **Predict labels of examples in test data**
6. Evaluate algorithm.

Decision Boundary

- We seek to find this boundary

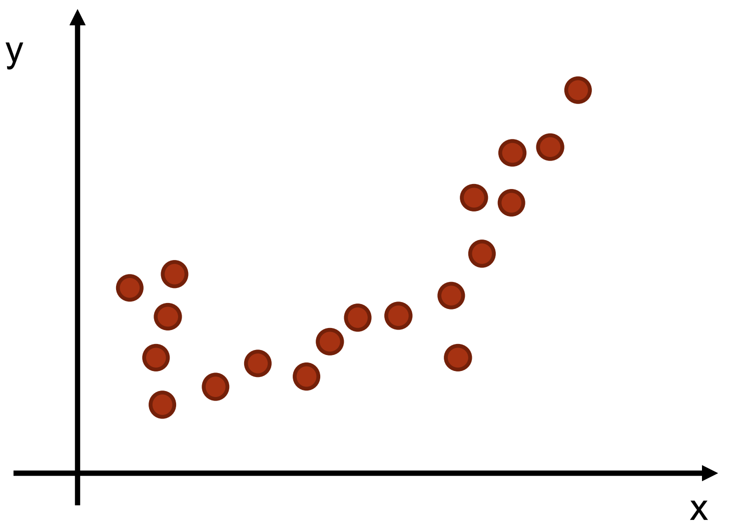


Why Noise?

- Noise might be due to different reasons
  - Imprecision in recording the input data
  - Errors in labeling data
  - We might not have considered additional features (*latent*, or *hidden* features)
- When there is noise, the decision boundary becomes more complex

Overfitting

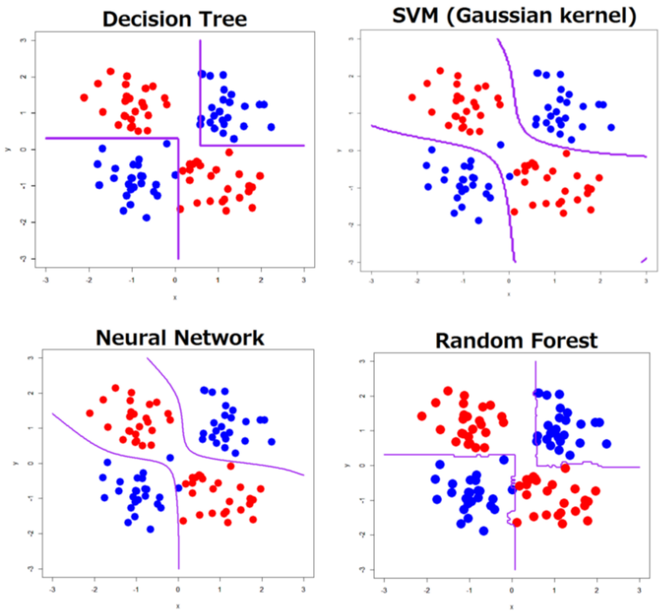
- Data are well described by our model, but the predictions do not generalize to new data.
  - A very rich hypothesis space
  - Training set too small

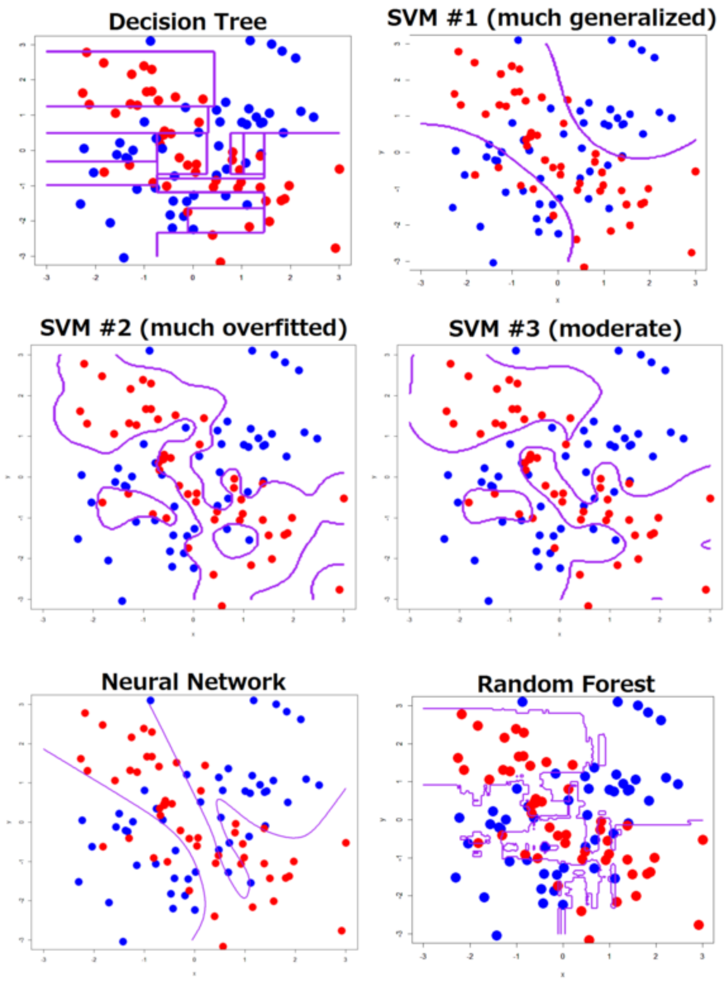


Overfitting and Underfitting

- Underfitting
  - If your hypothesis is less complex than the actual function
    - Using a linear equation to model data generated by a third order polynomial
- Overfitting
  - If your hypothesis is more complex than the actual function
    - Using a fifth order polynomial to model data generated by a second order polynomial

Over-fitted Decision Boundaries





http://www.kdnuggets.com/2015/06/decision-oundaries-deep-learning-machine-learning-ssifiers.html

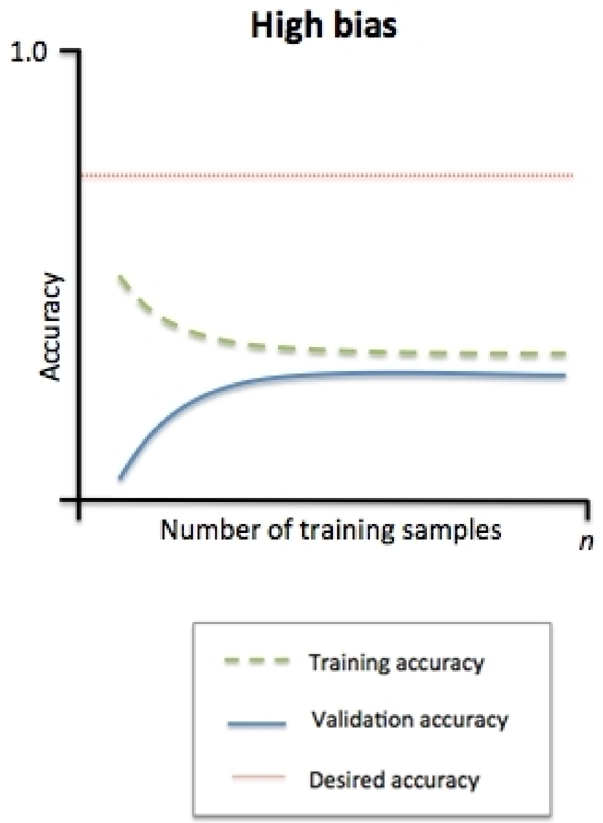
Bias-Variance

- Variance = consistency of the model for a particular sample if we retrain the model multiple times, e.g. on different subsets of data
  - i.e. we can say the model is sensitive to the randomness in data
- Bias = how far off are the predictions from the correct values (systematic error not due to randomness in data)

- Simple linear model => high bias
- Complex model => high variance

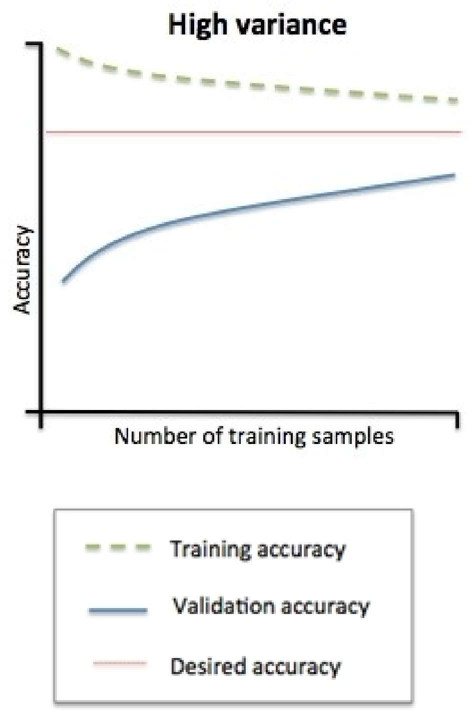
Model Diagnosis Using Learning Curves

- This model has both low training and cross-validation accuracy, which indicates that it underfits the training data.
- We can
  - Increase the number of parameters of the model
    - More features
    - Decreasing regularization



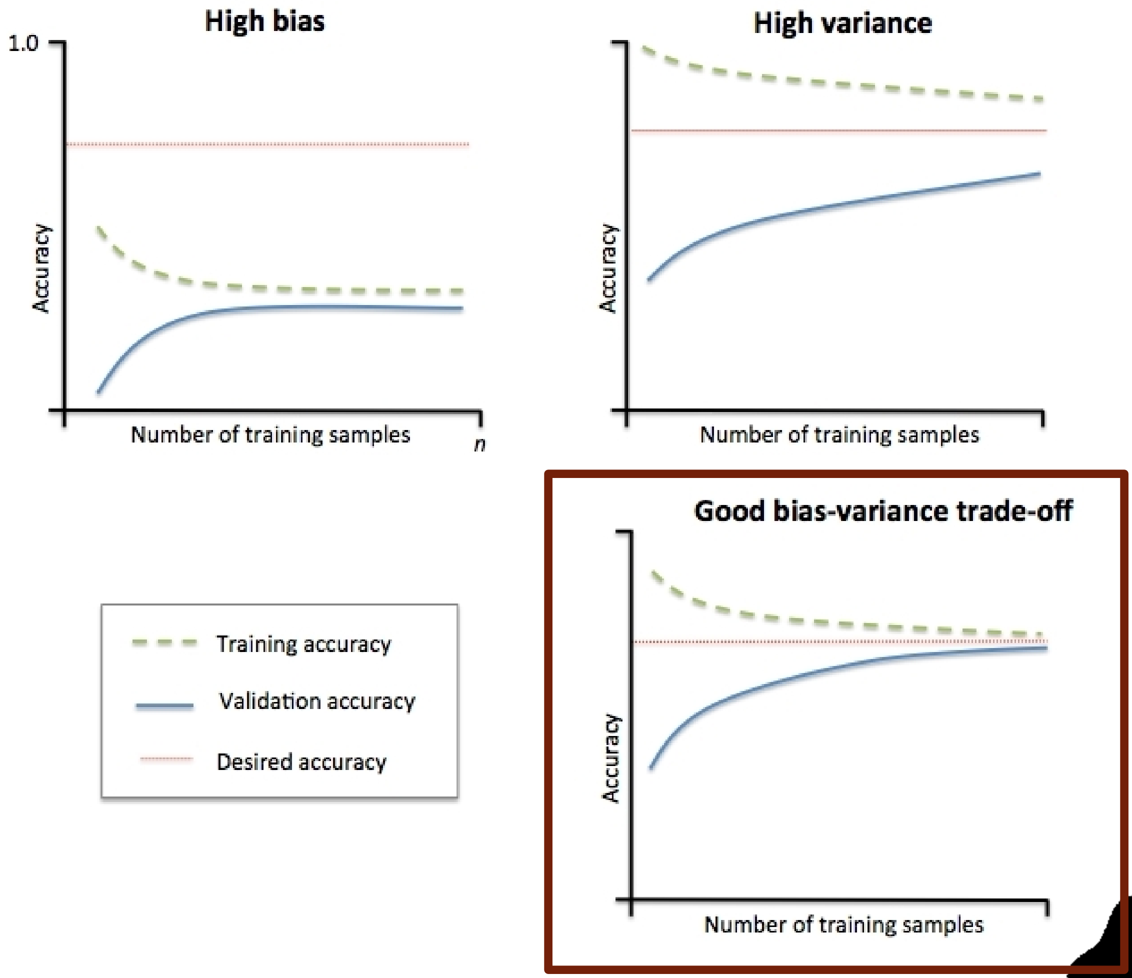
Model Diagnosis Using Learning Curves

- A model that suffers from high variance, which is indicated by the large gap between the training and cross-validation accuracy.
- To address this problem
  - we can collect more training data
  - or reduce the complexity of the model, e.g. more regularization parameter
  - or decrease the number of features via feature selection



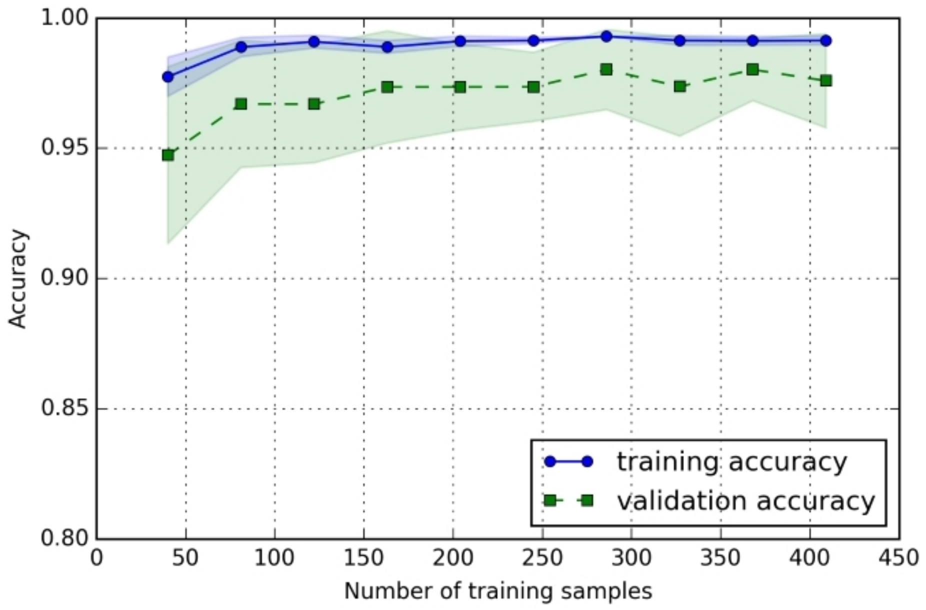
Model Diagnosis Using Learning Curves

- A good balance



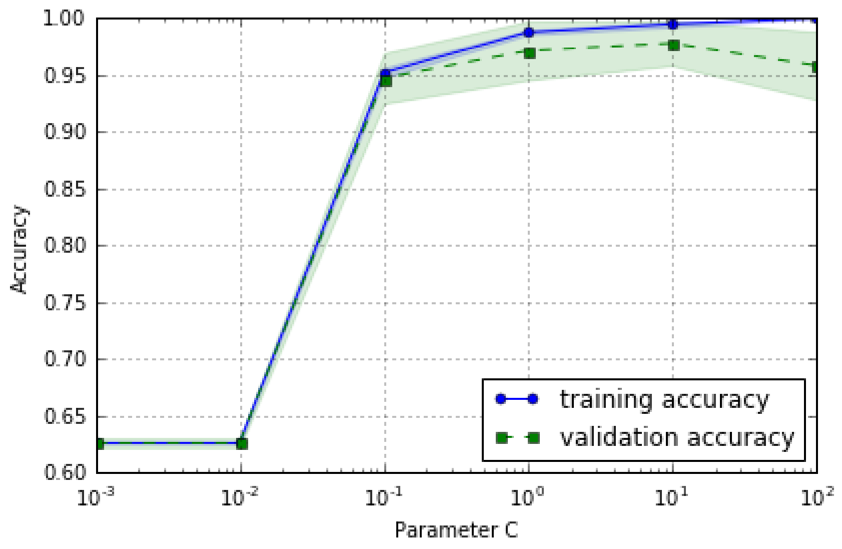
Learning Curve in scikit-learn

- Use the learning_curve() function in sklearn
- See notebook



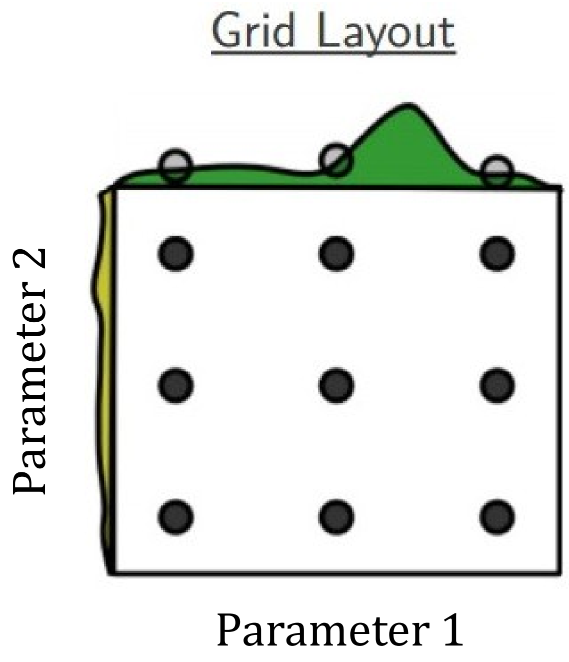
Validation Curve

- Similar to learning curves
  - Instead of plotting against the sample size, we vary the values of the model parameters, e.g. C in SVM
  - Use validation_curve()



Fine-tune via Grid Search

- You can use Grid Search to automate the process of finding the best parameter values



Important Steps

1. Determine relevant features (expert knowledge)
2. Collect data
3. Split labeled data into training and test datasets
4. Use training data to train machine learning algorithm.
5. Predict labels of examples in test data
6. **Evaluate algorithm.**

Model Evaluation

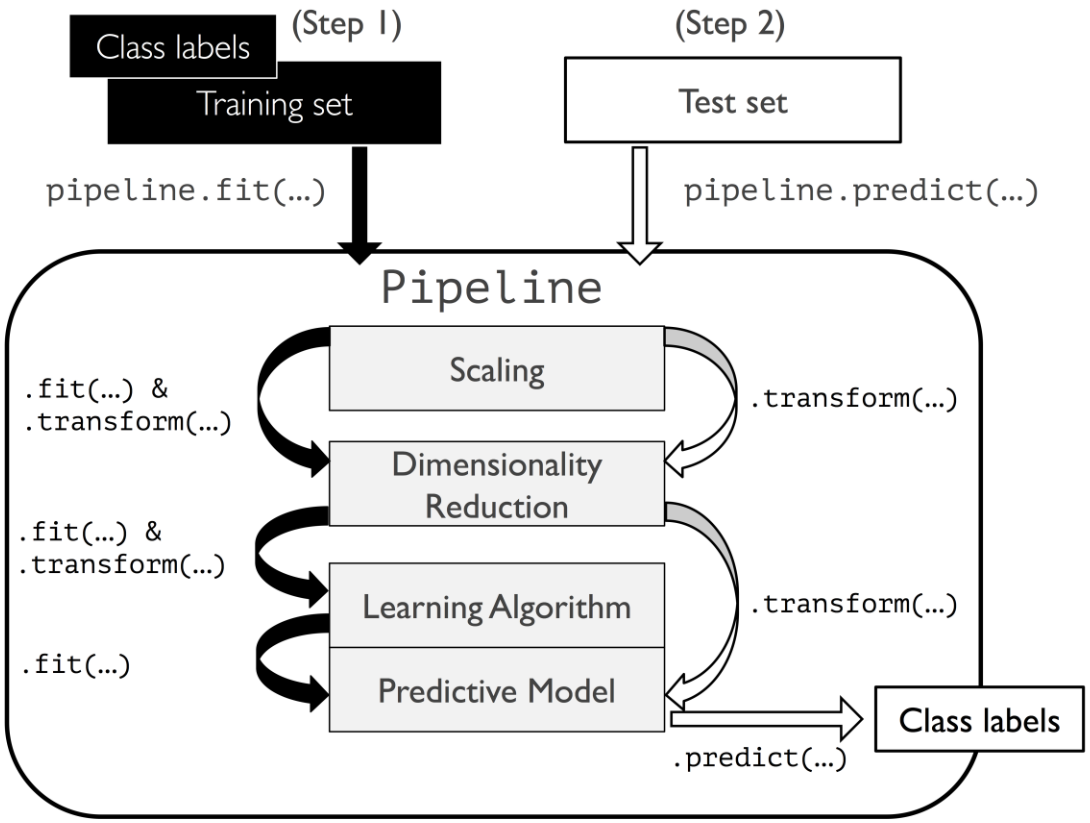
- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Triple Tradeoff

- Complexity of the hypothesis space: *C*
- Amount of training data: *N*
- Generalization error on new data: *E*

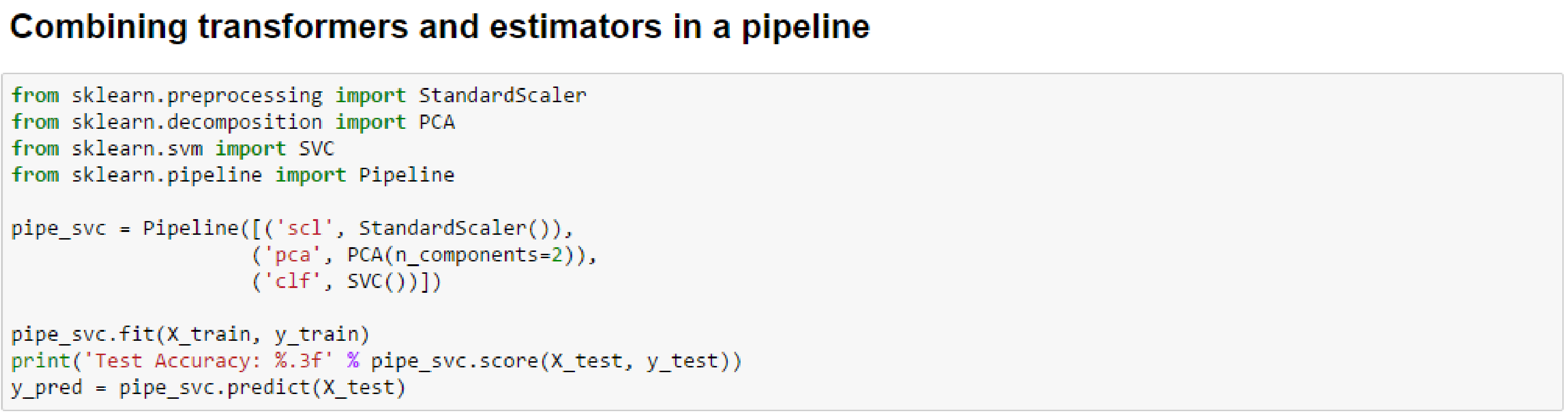
- $N \uparrow \;\rightarrow\; E\downarrow$
- $C \uparrow \;\rightarrow\;$ *first* $E\downarrow$, *then* $E\uparrow$

Development Pipeline



Scikit-Learn: Pipeline

- The Pipeline object takes a list of tuples as input
  - first value = an arbitrary identifier string
  - second value = a scikit-learn transformer or estimator.



Combining transformers and estimators in a pipeline

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline

pipe_svc = Pipeline([('scl', StandardScaler()),
                    ('pca', PCA(n_components=2)),
                    ('clf', SVC())])

pipe_svc.fit(X_train, y_train)
print('Test Accuracy: %.3f' % pipe_svc.score(X_test, y_test))
y_pred = pipe_svc.predict(X_test)
```