Machine Learning I (DATS 6202)

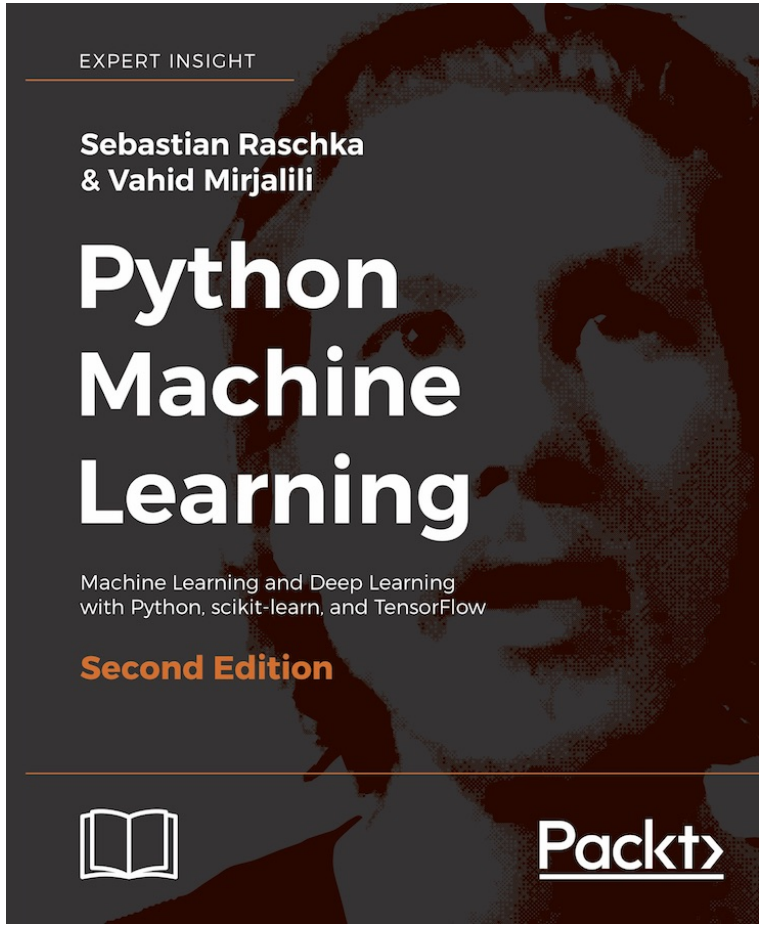
Model Evaluation and Hyperparameter Tuning

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University
yuxiaohuang@gwu.edu

November 5, 2018

Reference



Picture courtesy of the website of the book code repository and info resource

Reference

- This set of slices is an excerpt of the book by Raschka and Mirjalili, with some trivial changes by the creator of the slides
- Please find the reference to and website of the book below:
  - *Raschka S. and Mirjalili V. (2017). Python Machine Learning. 2nd Edition.*
  - `https://sebastianraschka.com/books.html`
- Please find the website of the book code repository and info resource below:
  - `https://github.com/rasbt/python-machine-learning-book-2nd-edition`

Overview

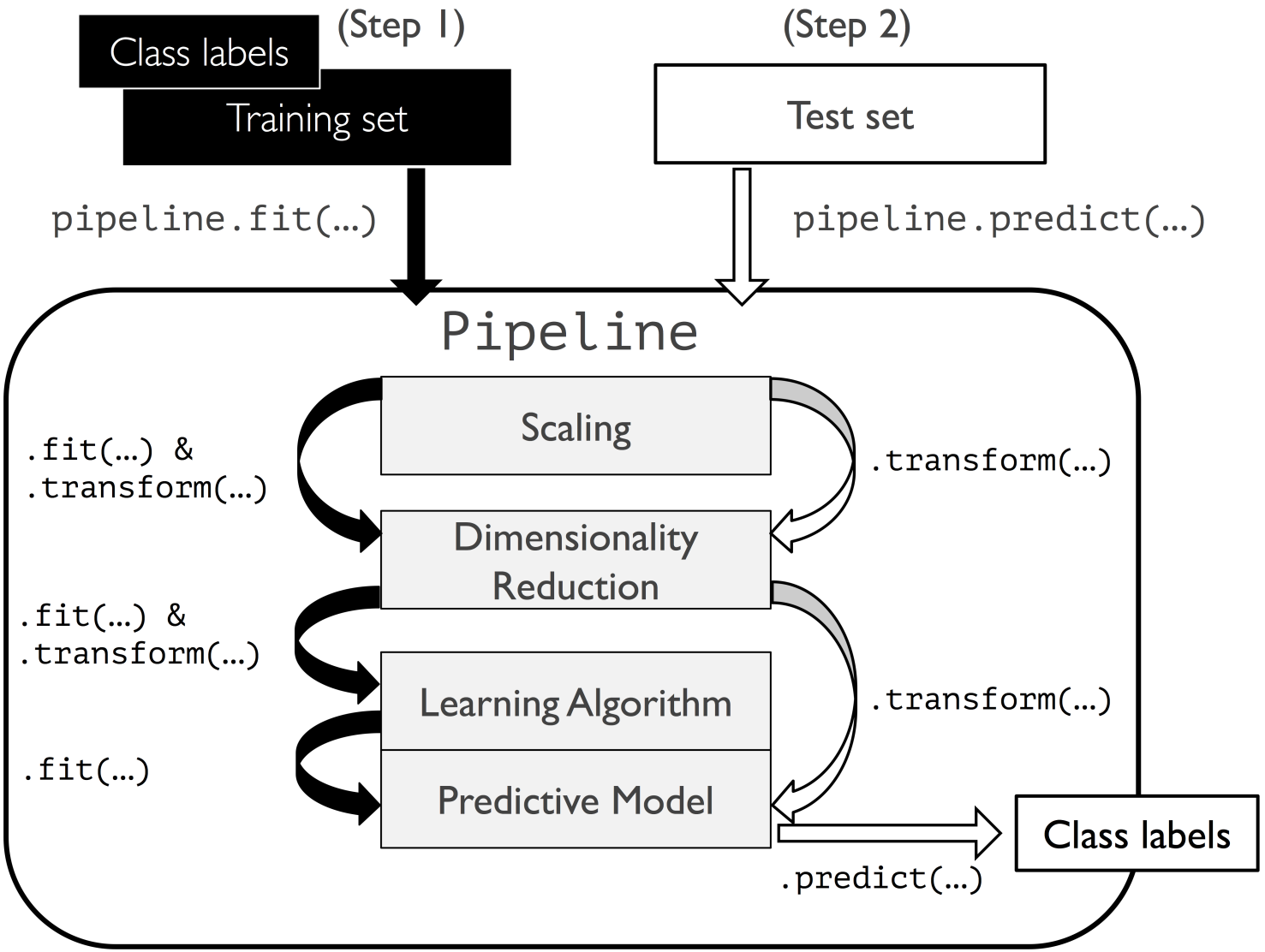
1. Streamlining workflows with pipelines

2. Using k-fold cross-validation to assess model performance

The `sklearn.pipeline` module

- After dividing the data into training and testing, we can chain the transformation and estimation steps in a pipeline using the `sklearn.pipeline` module
  - standardization
  - dimensionality reducition
  - ...
  - training (fit) and testing (predict)
- The `make_pipeline` function in the pipeline module takes as input an arbitrary number of scikit-learn transformers, followed by a scikit-learn estimator:
  - the transformers must support the `fit` and `transform` functions
  - the estimator must support the `fit` and `predict` functions

Figure 1


(Step 1)
Class labels
Training set
pipeline.fit(...)
(Step 2)
Test set
pipeline.predict(...)
Pipeline
Scaling
Dimensionality Reduction
Learning Algorithm
Predictive Model
.fit(...) & .transform(...)
.fit(...) & .transform(...)
.fit(...)
.transform(...)
.transform(...)
.predict(...)
Class labels

Hyperparameter tuning and model selection

- So far we have been discussing training the model on training data, and testing it on testing data
- **Q:** What shall we do if the result on the testing data is bad?

Hyperparameter tuning and model selection

- So far we have been discussing training the model on training data, and testing it on testing data
- **Q:** What shall we do if the result on the testing data is bad?
- **A:** We can:
  1. tune the settings of the hyperparameters to have a different model. This process is called hypermarameter tuning.
  2. compare the performance of different models. This process is called model selection.
- **Q:** Can we use the testing data for model selection?

Hyperparameter tuning and model selection

- So far we have been discussing training the model on training data, and testing it on testing data
- **Q:** What shall we do if the result on the testing data is bad?
- **A:** We can:
  1. tune the settings of the hyperparameters to have a different model. This process is called hypermarameter tuning.
  2. compare the performance of different models. This process is called model selection.
- **Q:** Can we use the testing data for model selection?
- **A:** No! Otherwise the testing data will become part of the training data, and this is cheating!

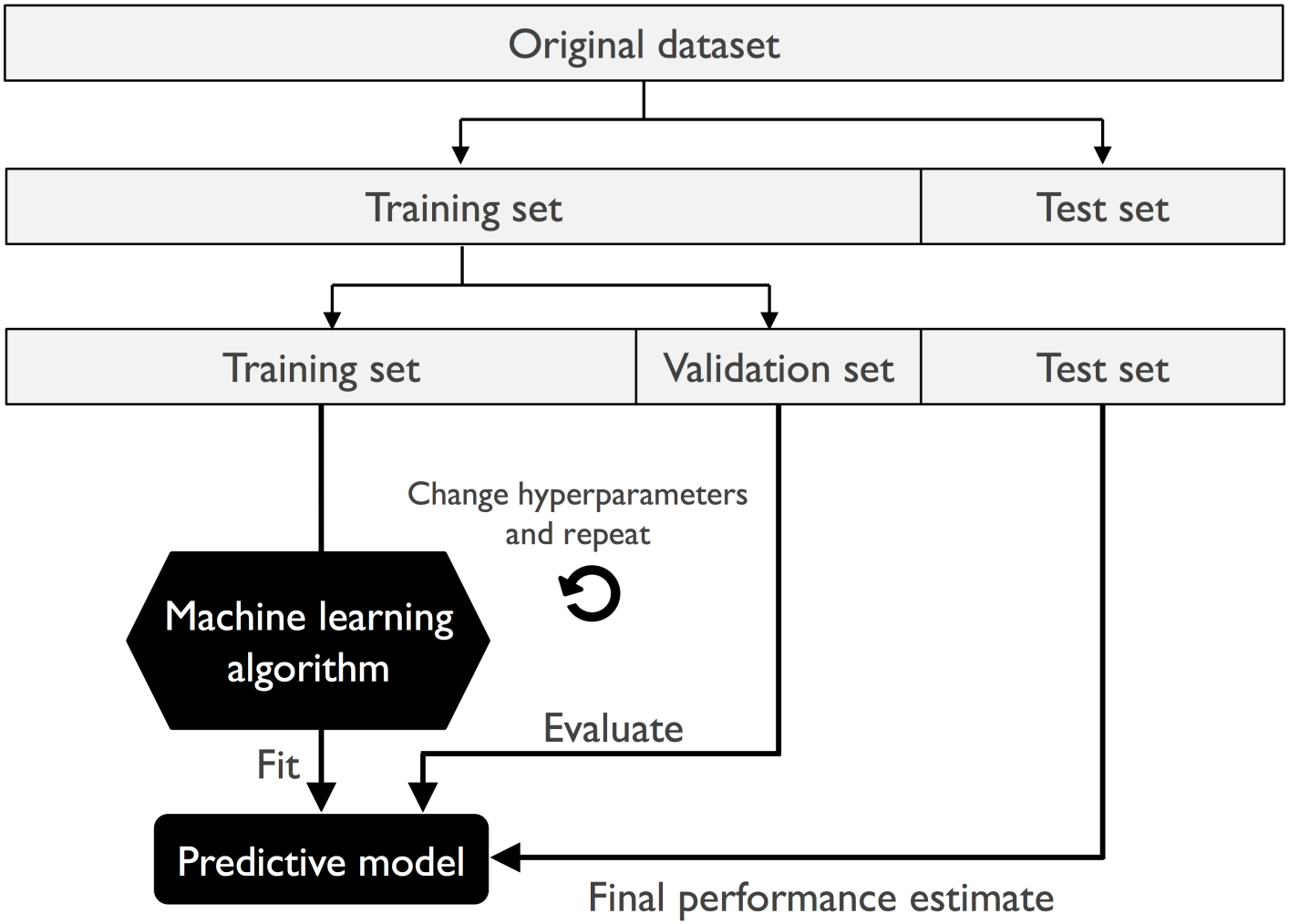
Cross validation

- Instead of using the testing data for model selection, we further divide the training data into two subsets
  - one subset is for training, we call it the training set
  - the other is for model selection, we call it the validation set
- The process of further dividing the training data into training and validation sets is called cross validation
- Two kinds of cross validation
  - holdout cross-validation
  - k-fold cross-validation

The holdout cross-validation

- The idea of holdout cross-validation is as follows
  - further divide the training data into training and validation sets
  - using the training set for hyperparameter tuning
  - using the validation set for model selection
- Figure 2 (see next page) shows the flowchart of the holdout method

Figure 2


Original dataset
Training set
Test set
Training set
Validation set
Test set
Change hyperparameters and repeat
Machine learning algorithm
Evaluate
Fit
Predictive model
Final performance estimate

The limitation and solution

- **Q:** Any problem with the holdout method?

The limitation and solution

- **Q:** Any problem with the holdout method?
- **A:** Since the new training set is smaller than the old one, hyperparameter tuning and model selection could be more sensitive to the distribution of the new training and validation sets
- **Q:** Can you think of a way to address this problem?

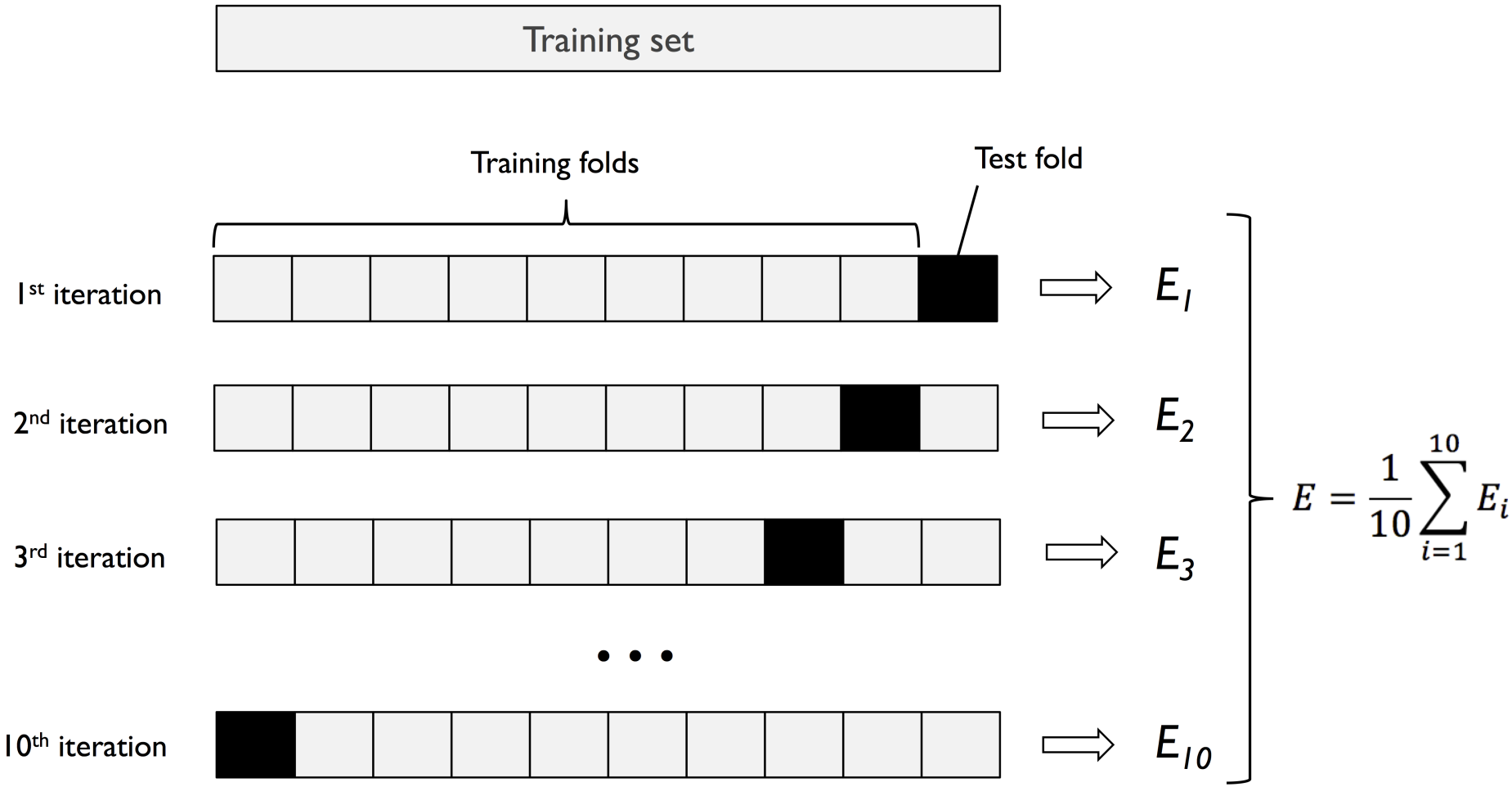
The limitation and solution

- **Q:** Any problem with the holdout method?
- **A:** Since the new training set is smaller than the old one, hyperparameter tuning and model selection could be more sensitive to the distribution of the new training and validation sets
- **Q:** Can you think of a way to address this problem?
- **A:** Yes! k-fold cross validation!

K-fold cross validation

- The idea of k-fold cross validation is as follows:
  1. split the training set into k folds (without replacement)
  2. for i from 1 to k, use:
     - the ith fold for validation
     - the remaining folds for training
- A good standard value for k is 10
- Figure 3 (see next page) shows the flowchart of the 10-fold cross validation method

Figure 3


Training set
Training folds
Test fold
1st iteration
2nd iteration
3rd iteration
10th iteration
E_1
E_2
E_3
E_{10}
$E = \frac{1}{10}\sum_{i=1}^{10} E_i$

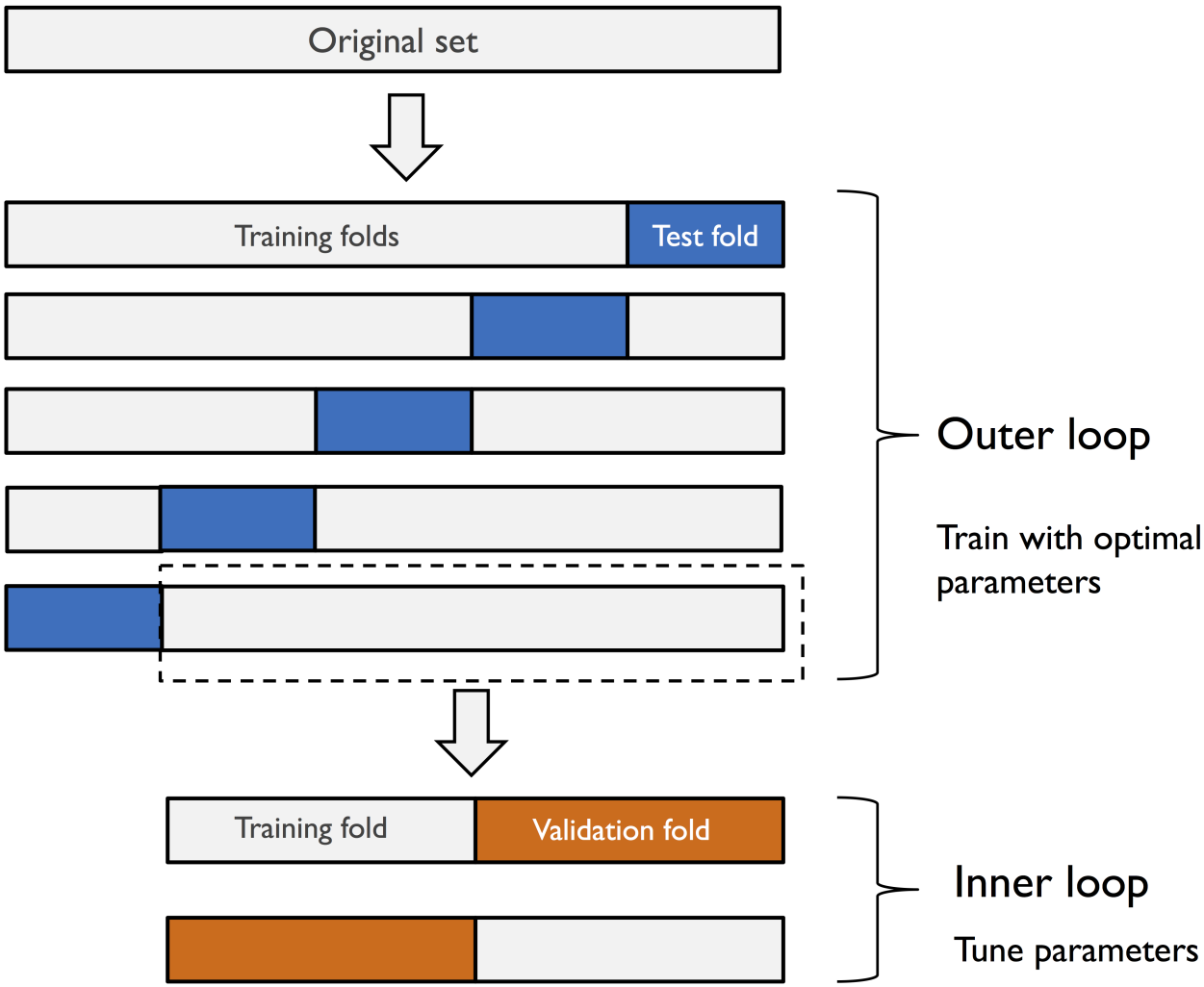
Stratified k-fold cross validation

- When splitting the data into training and testing, we used the stratify method (by setting `stratify=y`) to preserve the class proportions in the training and testing data
- The goal is to create training and testing data that represent the same (or similar) class proportions as those in the raw data
- Similarly, we can use the stratify method for k-fold cross validation, so that each fold represents the same (or similar) class proportions as those in the raw data
- We call this the Stratified k-fold cross validation

Nested k-fold cross validation

- The idea of nested k-fold cross validation is having an outer and an inner k-fold cross validation loop
- The idea of the inner loop is further dividing the training set into training and validation sets
- Figure 4 (see next page) illustrates the flowchart of nested k-fold cross validation

Figure 4


Original set
Training folds
Test fold
Outer loop
Train with optimal parameters
Training fold
Validation fold
Inner loop
Tune parameters

Nested k-fold cross validation

- The idea of nested k-fold cross validation is having an outer and an inner k-fold cross validation loop
- The idea of the inner loop is further dividing the training set into training and validation sets
- Figure 4 (see next page) illustrates the flowchart of nested k-fold cross validation
- **Q:** Why the inner loop is necessary?

Nested k-fold cross validation

- The idea of nested k-fold cross validation is having an outer and an inner k-fold cross validation loop
- The idea of the inner loop is further dividing the training set into training and validation sets
- Figure 4 (see next page) illustrates the flowchart of nested k-fold cross validation
- **Q:** Why the inner loop is necessary?
- **A:** If we use the validation set for model selection, then it gradually becomes part of the training set, the same problem as we discussed previously

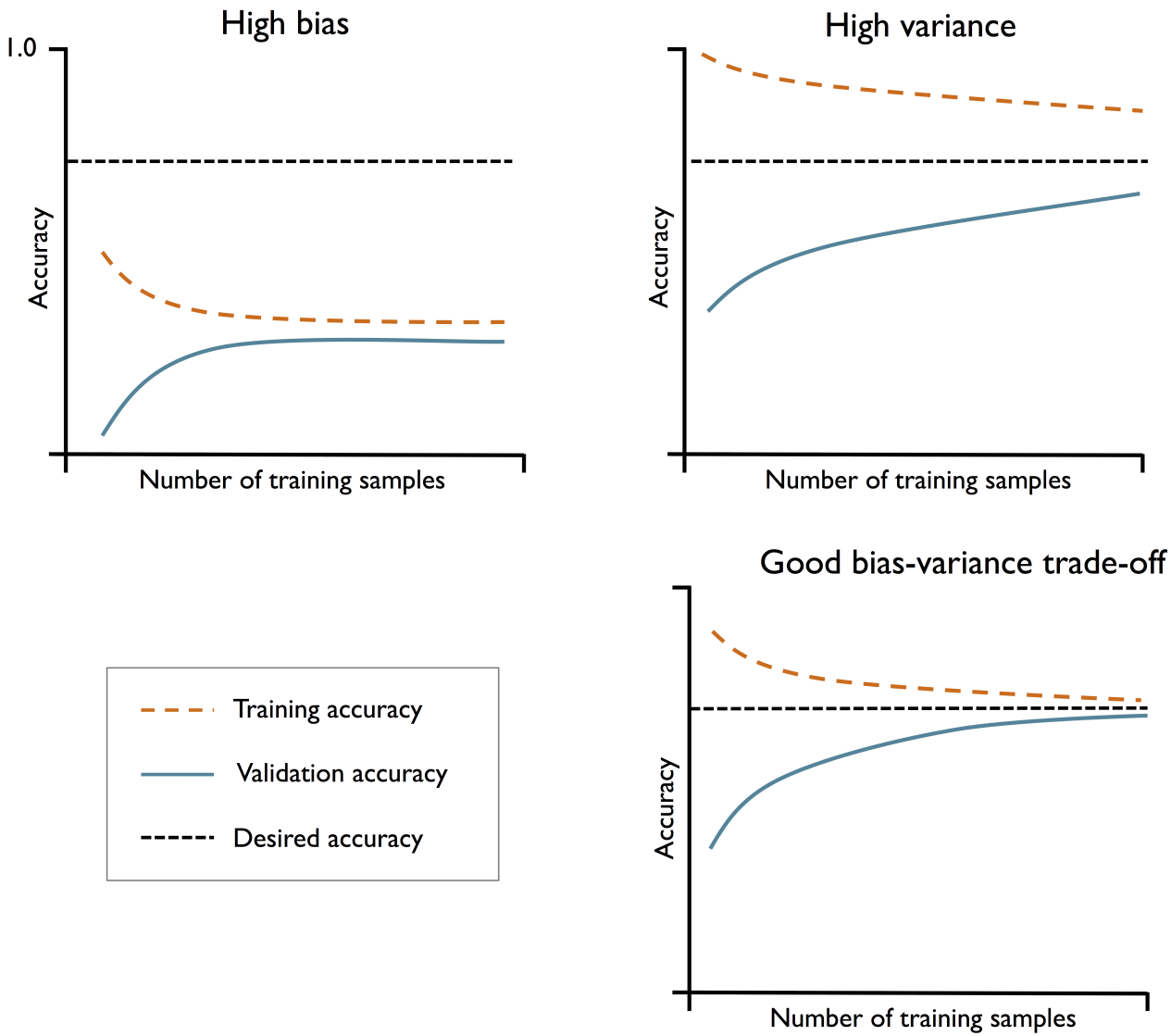
Leave-one-out cross-validation (LOOCV)

- For very small datasets with n samples, we can let $k = n$
- Each time only one sample is used for validation, and the remaining $n - 1$ samples are used for training
- This method is called Leave-one-out cross-validation (LOOCV)

The bias-variance tradeoff

- As discussed previously, high bias is the sign of underfitting whereas high variance is the sign of overfitting
- The tradeoff between bias and variance in essence is the tradeoff between underfitting and overfitting
- This is shown in Figure 5 (see next page)

Figure 5


High bias
1.0
Accuracy
Number of training samples
High variance
Accuracy
Number of training samples
Good bias-variance trade-off
Accuracy
Number of training samples
Training accuracy
Validation accuracy
Desired accuracy

The learning curve VS validation curve

- We can address the bias-variance tradeoff by plotting the learning and validation curves
  - the learning curve is the accuracy on the training and validation sets as a function of sample size
  - the validation curve is the accuracy on the training and validation sets as a function of hyperparameter values
- Figures 6 and 7 (see next two pages) illustrate the learning and validation curves

Figure 6

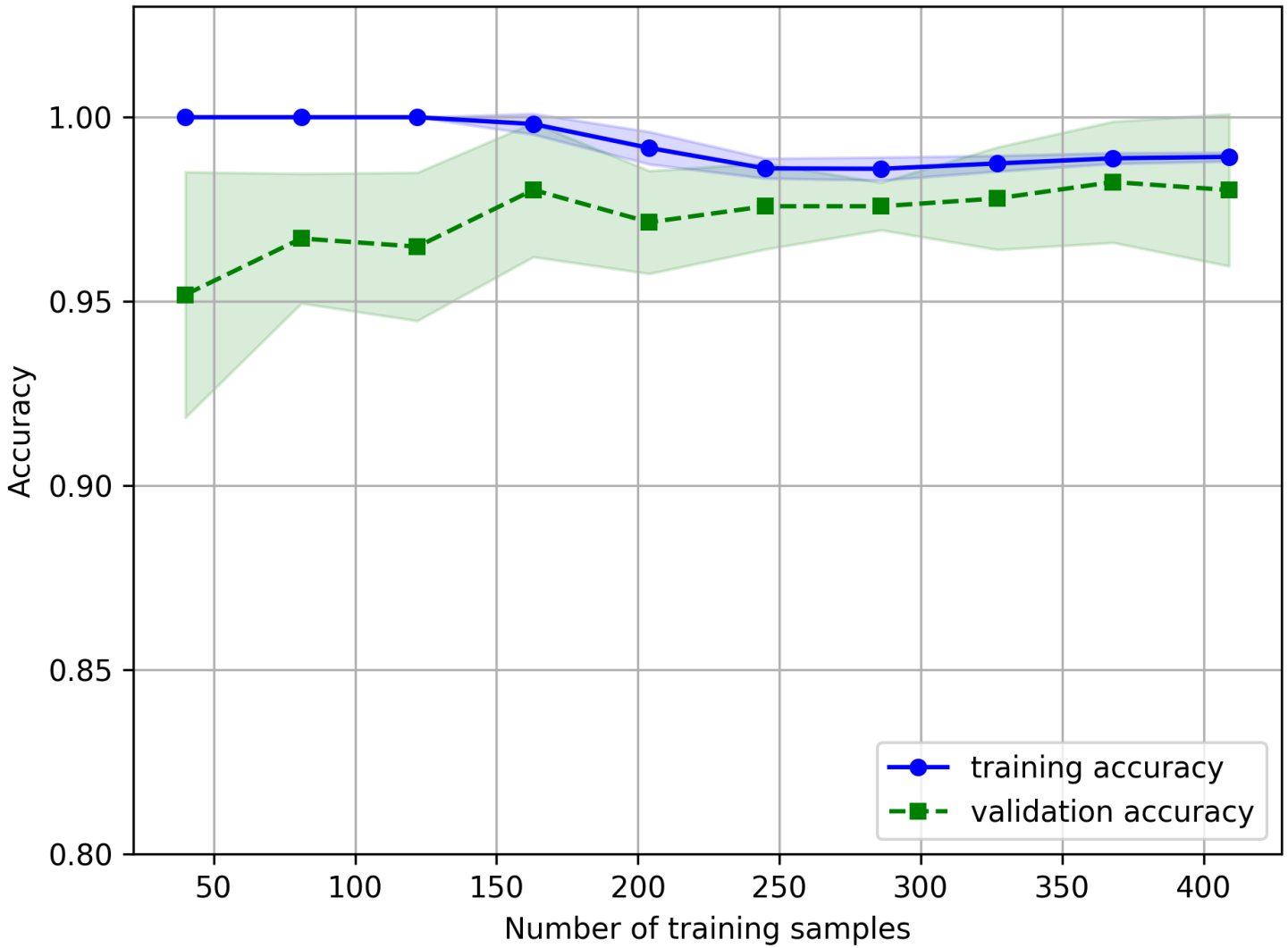
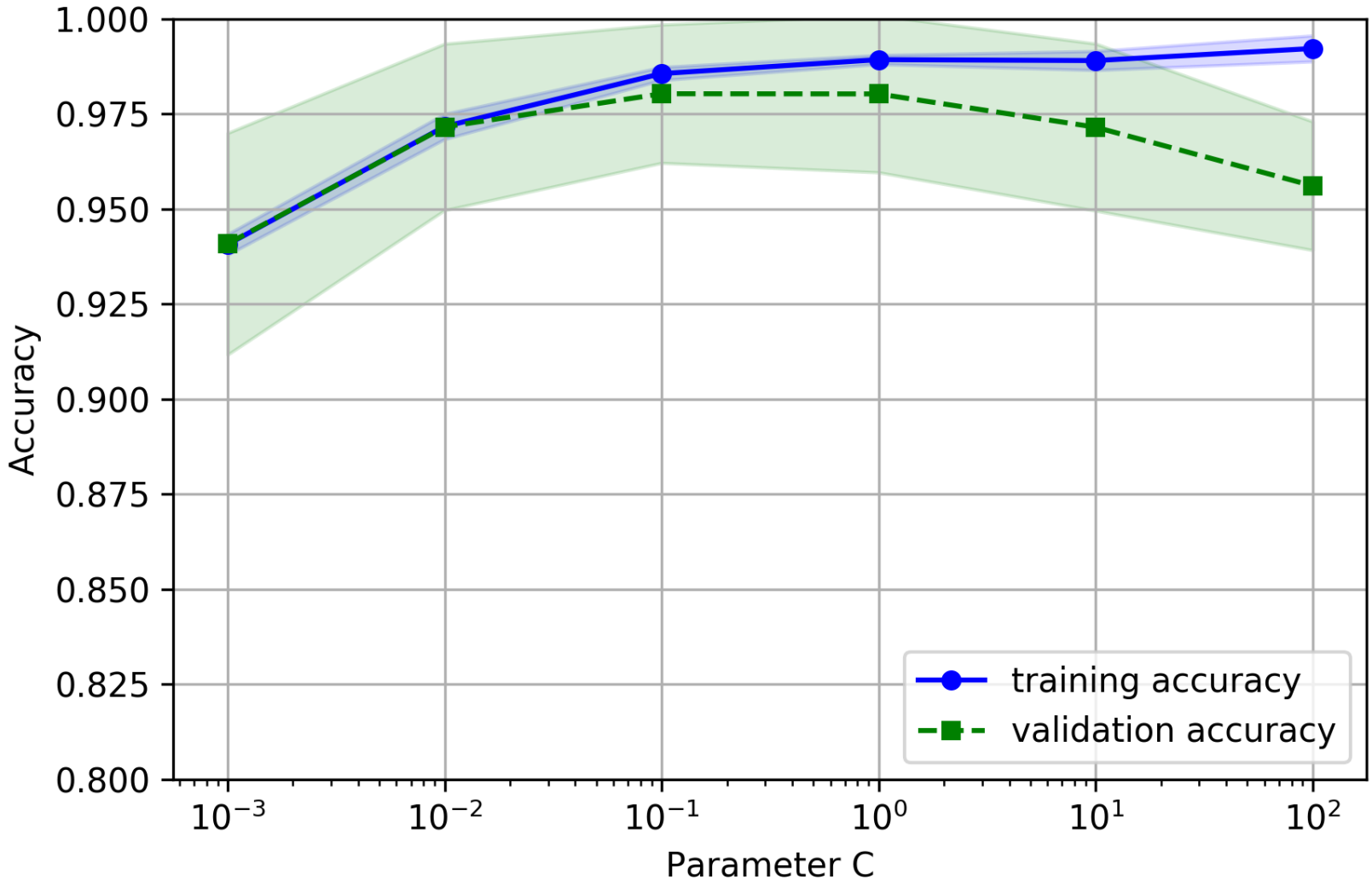

1.00
0.95
0.90
0.85
0.80
Accuracy
50
100
150
200
250
300
350
400
Number of training samples
training accuracy
validation accuracy

Figure 7


1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracy
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Parameter C
training accuracy
validation accuracy

Grid search and randomized search

- The idea of grid search is to select the best model across all the combinations of hyperparameter values
- Since it is a brute-force exhaustive search, it has exponential time complexity (2^n where n is the number of possible combinations), and infeasible for moderately large number of combinations
- In such cases, we can use randomized search
- Instead of visiting all the provided values of each parameter, the method only visits the most probable values generated by the provided distribution of the parameter

The metrics for evaluation

- There are several metrics that can be used for evaluating the models
  - accuracy
  - confusion matrix
  - precision, recall, and F1-score
  - The Receiver Operating Characteristic (ROC) and ROC area Under the Curve (ROC AUC)
- Figures 8 and 9 (see next two pages) illustrate the confusion matrix and ROC

Figure 8

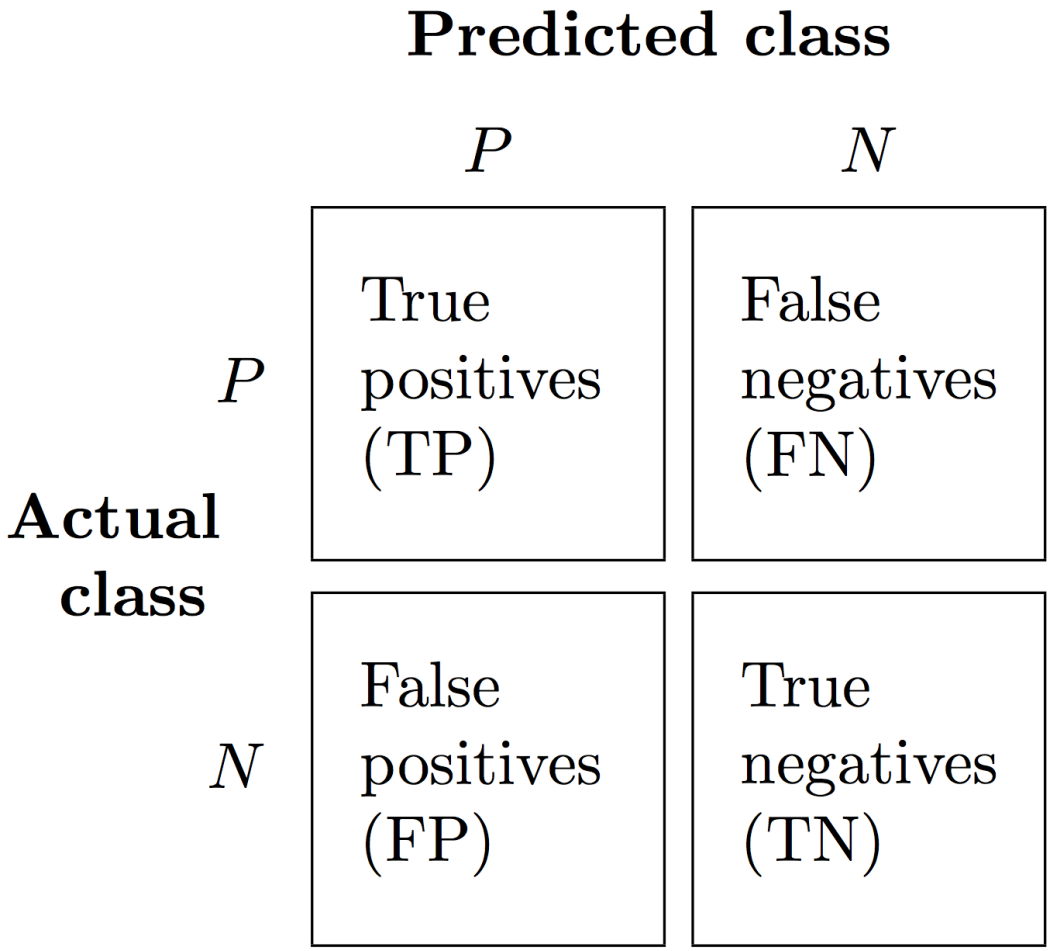
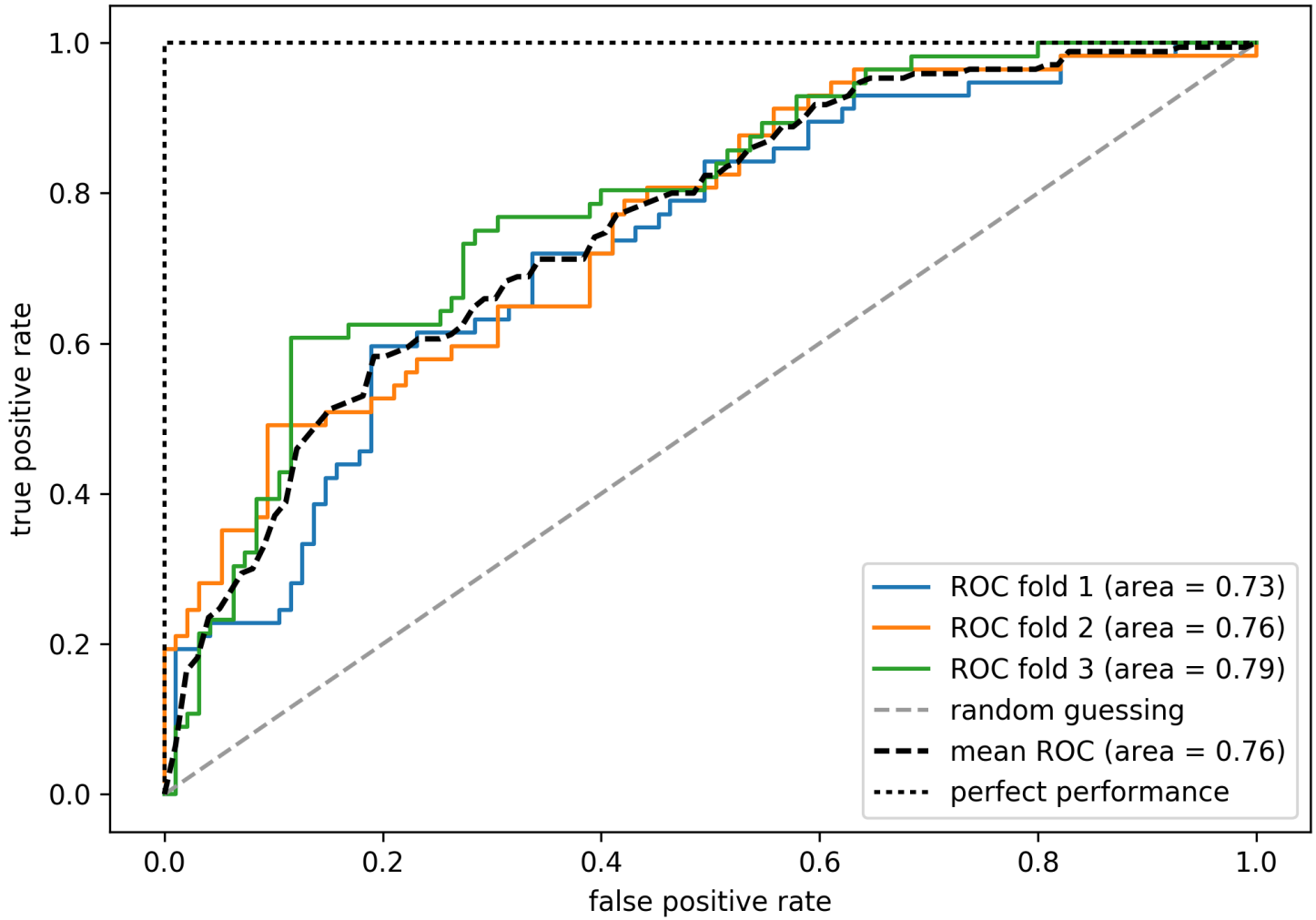

Predicted class
P
N
Actual class
P
True positives (TP)
False negatives (FN)
N
False positives (FP)
True negatives (TN)

Figure 9


1.0
0.8
0.6
0.4
0.2
0.0
true positive rate
0.0
0.2
0.4
0.6
0.8
1.0
false positive rate
ROC fold 1 (area = 0.73)
ROC fold 2 (area = 0.76)
ROC fold 3 (area = 0.79)
random guessing
mean ROC (area = 0.76)
perfect performance

Dealing with class imbalance

- In reality class imbalance is quite common
- **Q:** Why this is a problem for machine learning?

Dealing with class imbalance

- In reality class imbalance is quite common
- **Q:** Why this is a problem for machine learning?
- **A:** Because the cost due to wrongly classifying the minority classes will be overwhelmed by the award due to correctly classifying the majority classes
- There are several ways to handle class imbalance:
  - balancing the classes by duplicating minority ones
  - giving higher weights when misclassifying the minority classes
- There is no universally best solution