Ensemble Learning and Random Forests

What is Ensemble Learning?

- **Definition:** Combine multiple models to produce a better predictive performance than any single model.
- **Analogy:** Wisdom of the crowd.
- **Types:**
  - Voting / Averaging
  - Bagging & Pasting
  - Boosting
  - Stacking

Voting Classifiers

- **Idea:** Aggregate predictions from multiple classifiers and predict the class that gets the most votes.
- **Hard Voting:** Majority vote
- **Soft Voting:** Average of predicted probabilities
- **Equation:**

$$\hat{y} = mode\{C_1(x), C_2(x), \ldots, C_n(x)\}$$

- Ensemble methods work best when the predictors are independent.

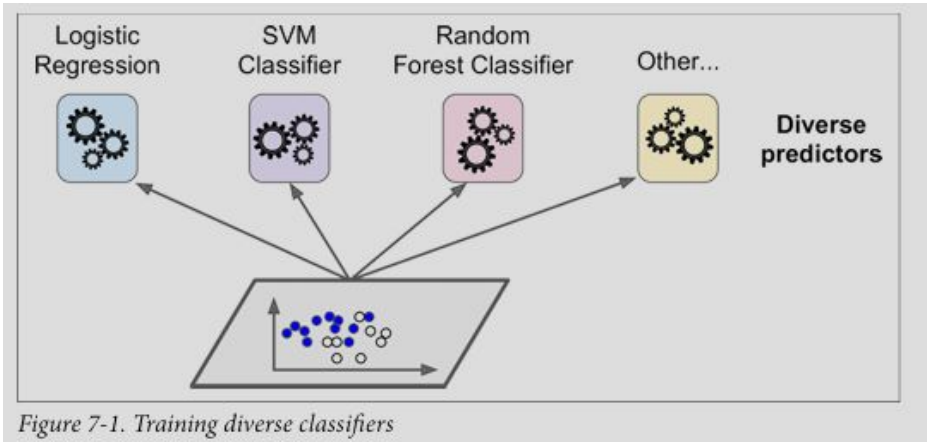


Figure 7-1. Training diverse classifiers

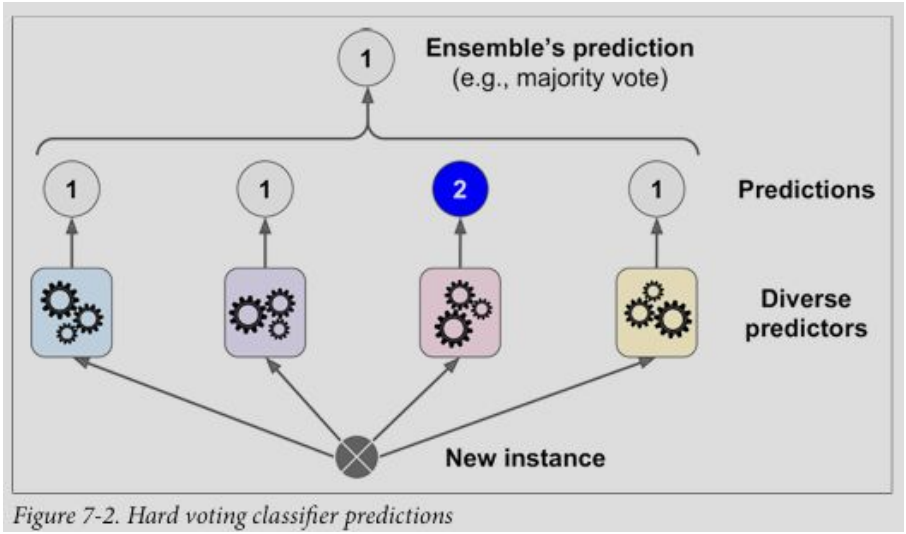


Figure 7-2. Hard voting classifier predictions

Hard vs. Soft Voting

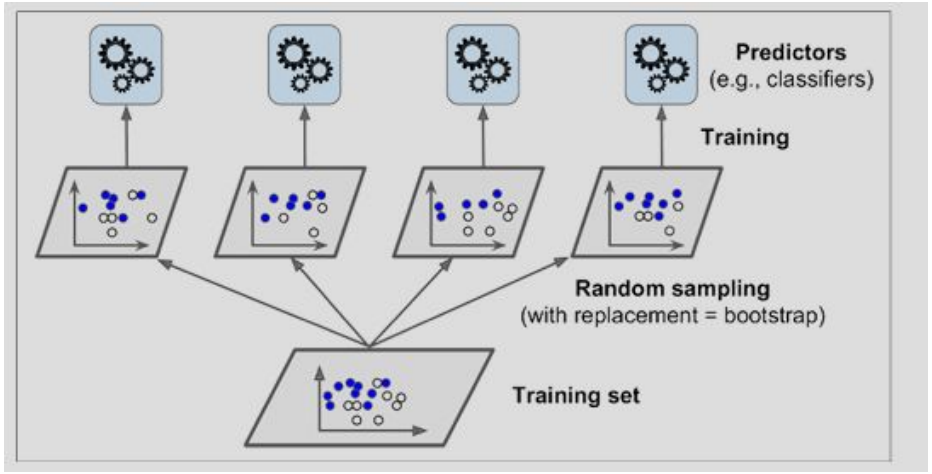
- **Hard Voting:** Each classifier gets one vote.
- **Soft Voting:** Weights votes by confidence (probability).

Example:

- Classifier 1: [0.9, 0.1] → Class 0
- Classifier 2: [0.6, 0.4] → Class 0
- Classifier 3: [0.4, 0.6] → Class 1
- **Hard Vote:** Class 0 wins (2 vs 1)
- **Soft Vote:** Avg prob = [0.63, 0.37] → Class 0

Bagging and Pasting

- **Bagging (Bootstrap Aggregating):** Train multiple models on random subsets of the training data *with replacement*.
- **Pasting:** Sampling *without replacement*.
- Reduces variance and helps avoid overfitting.



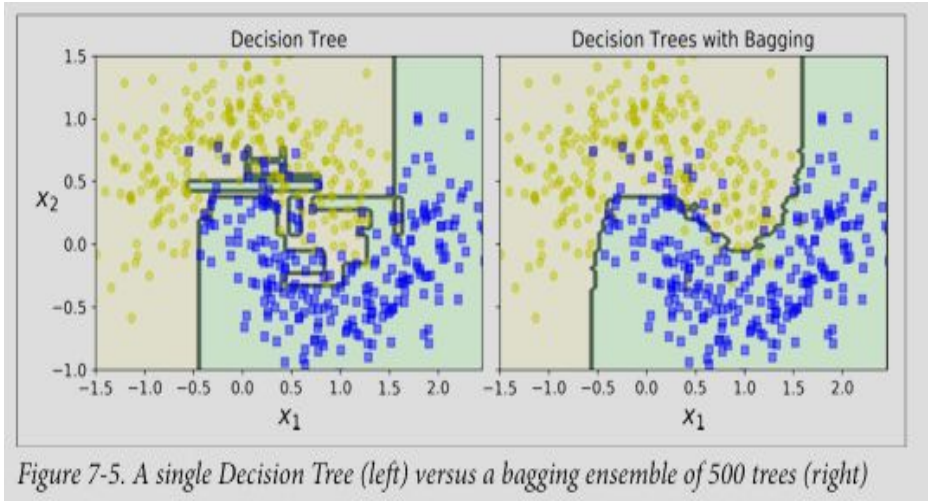


Figure 7-5. A single Decision Tree (left) versus a bagging ensemble of 500 trees (right)

Out-of-Bag Evaluation

In bagging, some instances are never sampled (~37% left out)

These can be used for validation without a separate validation set

Random Patches and Random Subspaces

Random Patches: Sample both instances and features.

Random Subspaces: Sample only features (keep all instances).

Useful for high-dimensional data.

Random Forests

- Ensemble of Decision Trees trained via bagging.
- Introduces extra randomness when splitting nodes.
  - Searches for the best feature among a random subset of features.
- More diverse trees → better generalization.

Extra-Trees (Extremely Randomized Trees)

Random Forest variant

Uses random thresholds for splits instead of best possible thresholds.

Faster to train, similar performance.

Feature Importance

- Random Forests can measure feature importance.
- How much a feature reduces impurity across all trees.

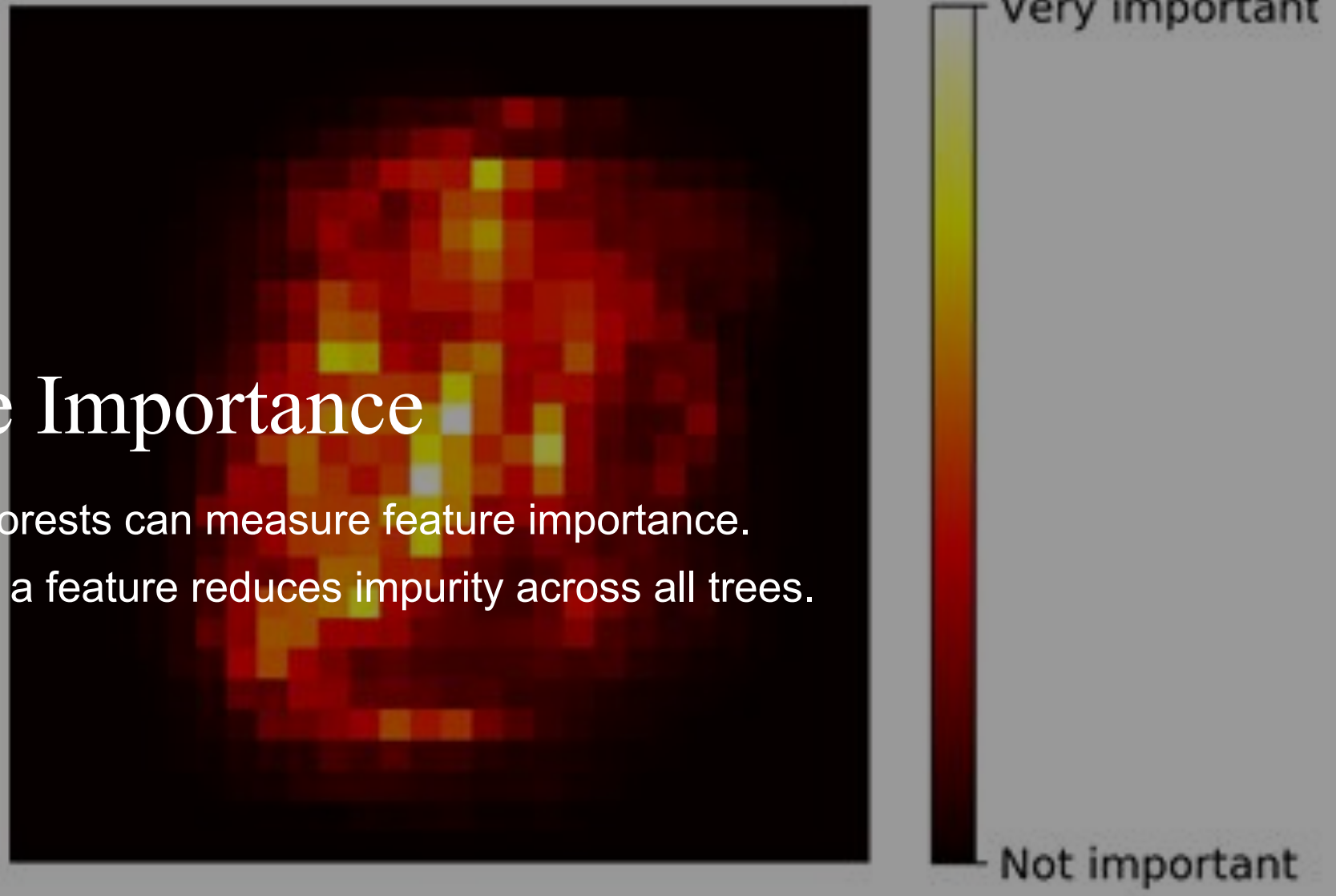
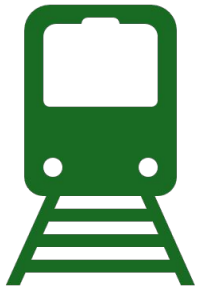


Figure 7-6. MNIST pixel importance (according to a Random Forest classifier)

Boosting

Idea: Combine multiple weak learners into a strong learner.



Train models sequentially, each trying to correct its predecessor.

Types: AdaBoost, Gradient Boosting.

AdaBoost (Adaptive Boosting)

- Weights misclassified instances higher in each iteration.
- Algorithm:
  - Train a base classifier.
  - Increase weight of misclassified instances.
  - Train next classifier with updated weights.
  - Combine all classifiers.

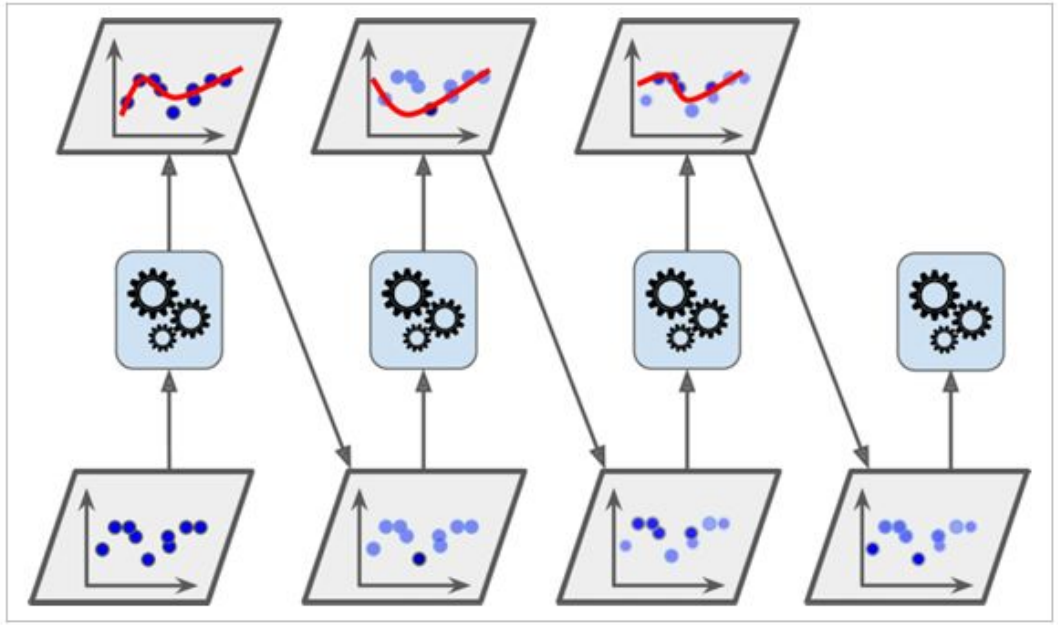


Figure 7-7. AdaBoost sequential training with instance weight updates

AdaBoost Mathematically

- Initial weight w^i of each instance = $\frac{1}{m}$.
- Weighted error rate r_1 of a first trained predictor is computed on the training set.
- **Weighted error rate of the *j*th predictor**

$$r_j = \frac{\left(\sum_{\substack{i=1 \\ \hat{y}_j^i \neq y^i}}^{m} w^i\right)}{\left(\sum_{i=1}^{m} w^i\right)}$$

Where $\hat{y}_j^i$ is the jth predictor's prediction for the ith instance.

- **Predictor weight (α_j)**

$$\alpha_j = \eta \times log \frac{1 - r_j}{r_j}$$

- **Weight update rule** boosts the weights of the misclassified instances.

$$for\ i = 1,2, \dots, m$$

$$w^i = \begin{cases} w^i, & if\ \hat{y}_j^i = y^i \\ w^i \exp(\alpha_j), & if\ \hat{y}_j^i \neq y^i \end{cases}$$

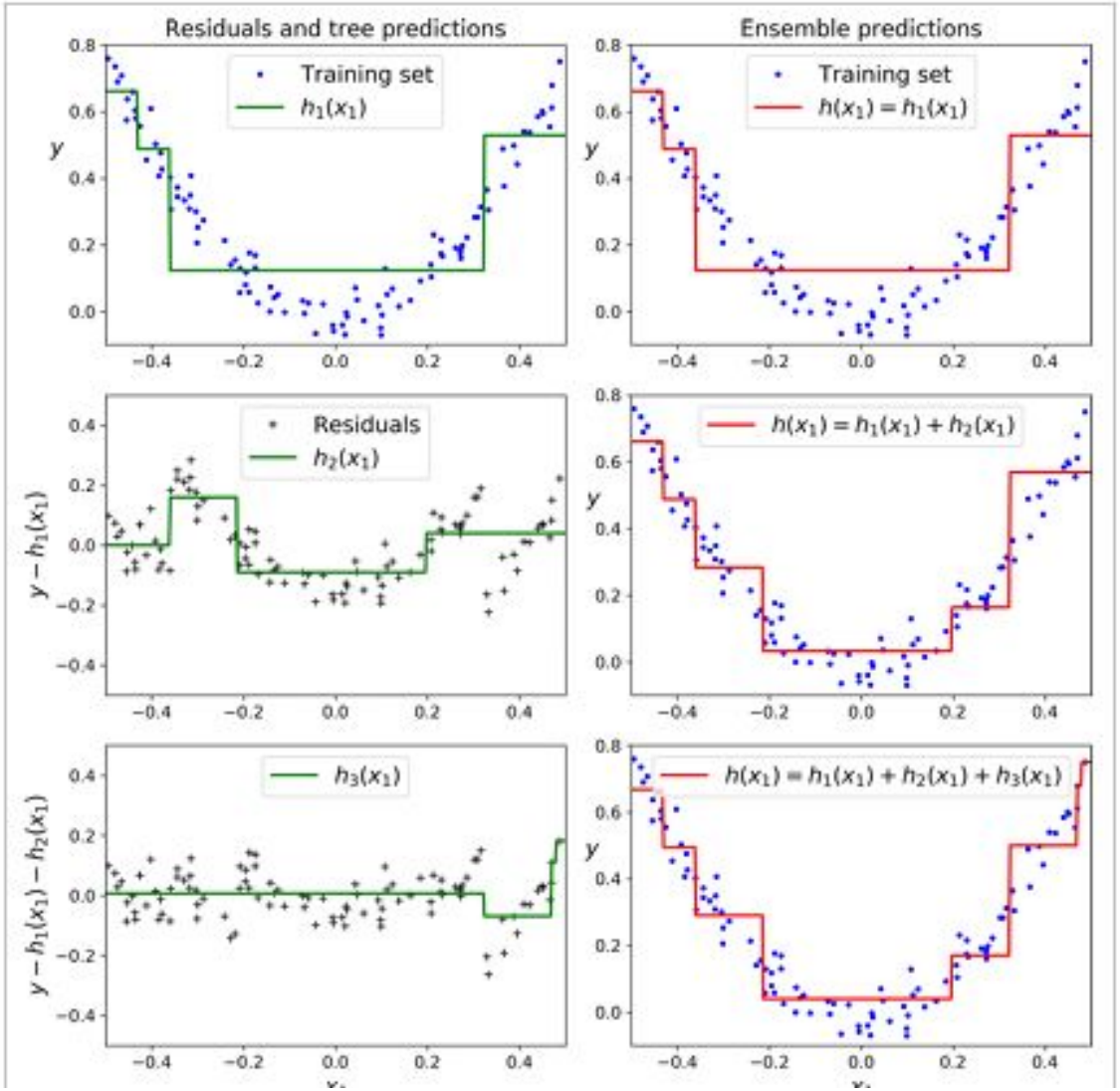
- **AdaBoost predictions**

$$\hat{y}_x = \underset{k}{argmax} \sum_{\substack{j=1 \\ \hat{y}_j(x)=k}}^{N} \alpha_j$$

where N is the number of predictors.

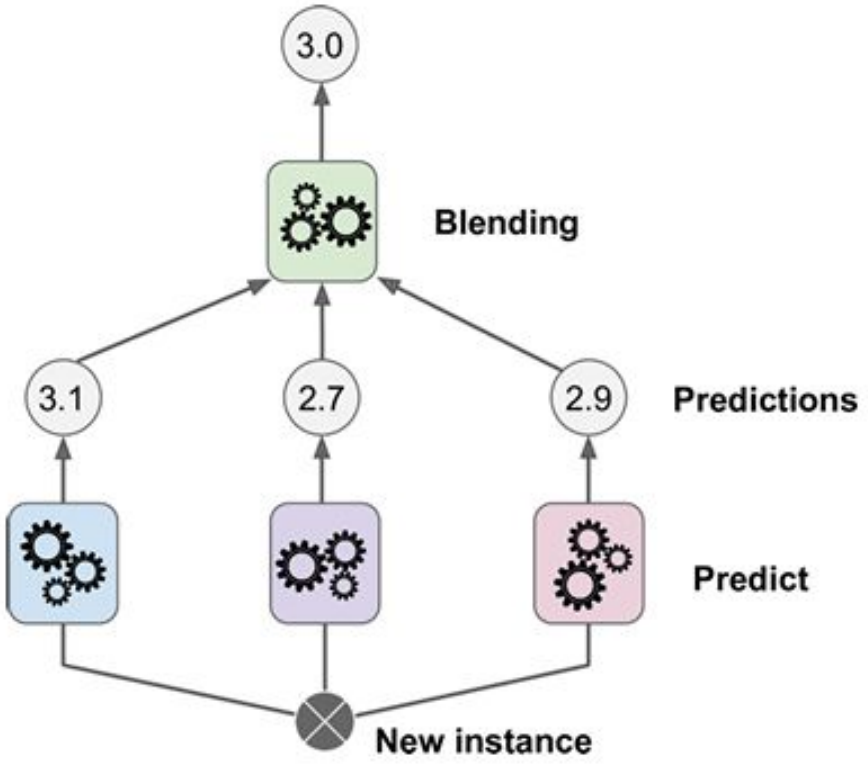
Gradient Boosting

- Sequentially adds predictors to correct the ***errors*** of the previous one.
- Fits new model to the residual errors.
- **Example:**
  - Gradient Tree Boosting, or Gradient Boosted Regression Trees (GBRT).
  - Train tree 1 → Compute residuals → Train tree 2 on residuals → Repeat



Stacking (Stacked Generalization)

- Train a model to combine the predictions of several base models.
- **Blender / Meta-learner:** Final model that takes predictions as input.



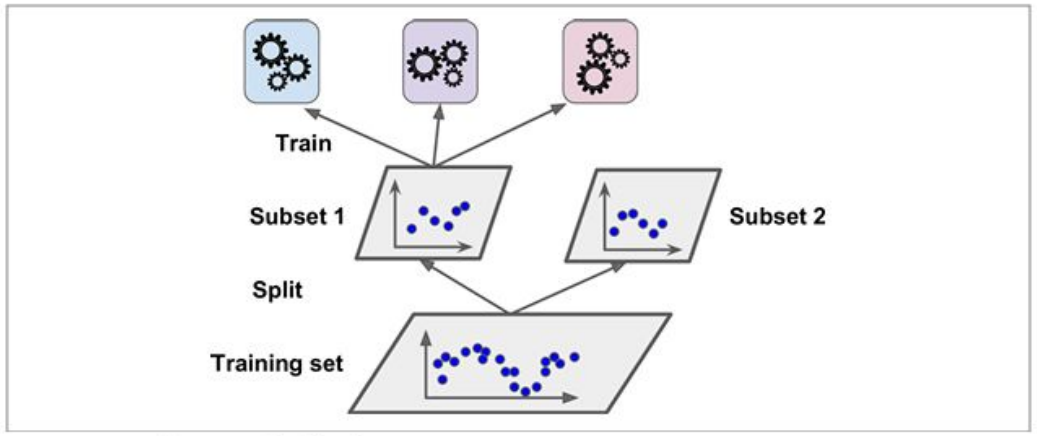


Figure 7-13. Training the first layer

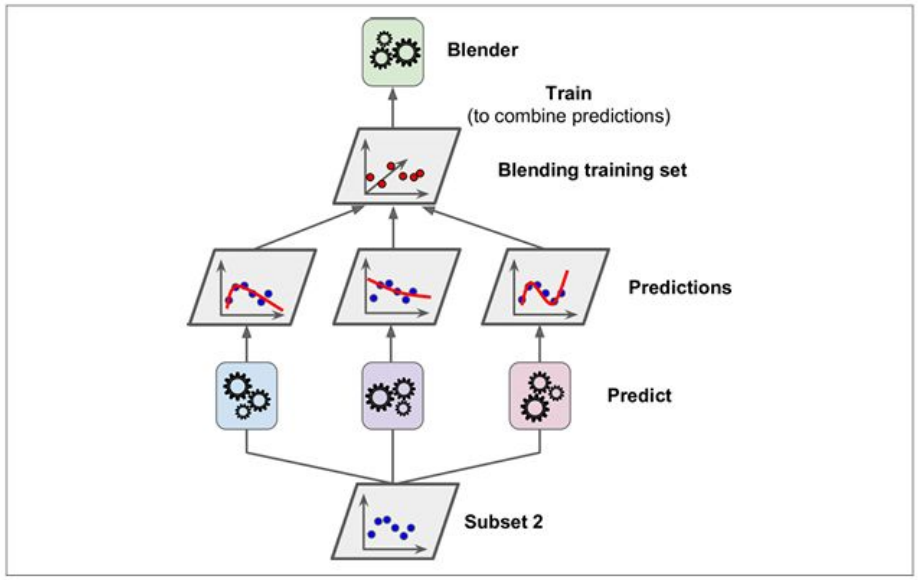


Figure 7-14. Training the blender

Training Blender