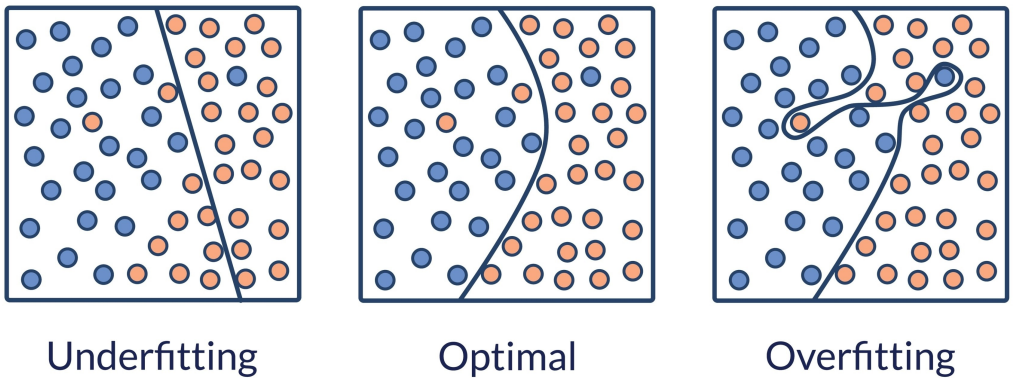
The Machine Learning Landscape Part 4

Testing and Validating

- The only reliable way to know if a model generalizes well is by evaluating it on unseen data.
- Evaluating only on the training data can give a false impression of good performance.
- Train/Test Split
  - Training set: used to train the model.
  - Test set: used to evaluate performance.
- Performance on the test set provides an estimate of the generalization error (also called out-of-sample error).
- If the training error is low but the test error is high, the model is likely overfitting.
- Common Split Ratio: A typical configuration is 80% training / 20% testing, although it may vary depending on dataset size.

Hyperparameter Tuning and Model Selection

- The problem



  -
  - Relying solely on the test set for hyperparameter tuning is dangerous.
  - If you repeatedly tweak the model to minimize test set error, you are adapting it to that specific data.
  - Result: The model "overfits" the test set, yielding an optimistic error rate but poor performance on truly new data (production).

- Haldout validation:
  - Hold out a portion of the training set to create a validation set (or dev set).
  - Process:
    - Train multiple candidate models (with different hyperparameters) on the reduced training set.
    - Select the best model based on performance on the validation set.
    - Refit the best model on the full training set (reduced training + validation).
    - Final Step: Evaluate this final model on the test set to estimate generalization error.

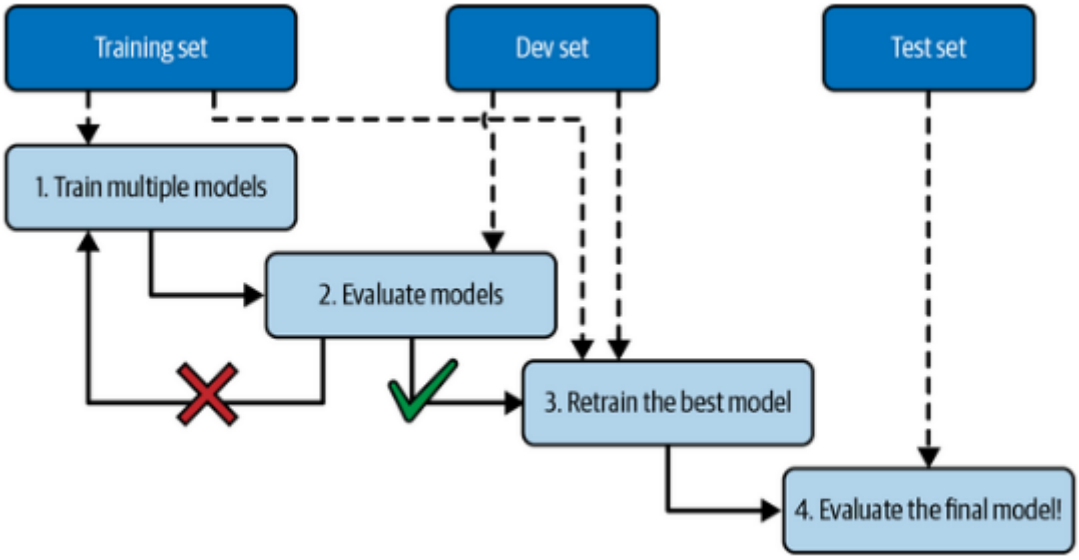


Figure 1-25. Model selection using holdout validation

- Trade-offs in Validation Set Size:
  - Too small: Evaluations become imprecise, risking the selection of a suboptimal model.
  - Too large: The remaining training set becomes too small. Since the final model is trained on the full set, comparing candidates trained on a tiny fraction gives a distorted view of their potential (like judging a marathon runner based on a sprint).
- Repeated Cross-Validation (The Solution):
  - Why use it: It solves the dilemma of validation set size.
  - Method: Use many small validation sets. Each model is evaluated once per set after training on the rest of the data.
  - Benefit: Averaging all evaluations provides a much more accurate measure of performance.
  - Drawback: Training time is multiplied by the number of validation sets (computationally expensive).

Here is the enhanced English version for your notes. I have structured it to clearly distinguish between the two types of errors, capturing the logic we discussed.

Data Mismatch

- The Context:
  - Often, you have a large amount of training data from one source (e.g., high-quality Web images) but your target application uses data from a different source (e.g., blurry Mobile images).
  - Golden Rule: To measure real-world performance, your Validation (Dev) and Test sets must come exclusively from the target distribution (Mobile images).
- The Diagnostic Problem:
  - If you train on Web images and evaluate on Mobile images (Dev set), and the error is high, you are blind to the cause. You cannot tell if:
    1. The model is overfitted (it can't generalize to anynew data).

2. The model is suffering from Data Mismatch (it doesn't understand the specific styleof Mobile images).

- The Solution: The "Train-Dev" Set
    - Create a new subset called the Train-Dev set.
    - Source: It is carved out of the original Training data (Web images).
    - Rule: The model is nottrained on it. It serves as a control group to check generalization within the same data distribution.
- The Diagnosis Workflow (Gap Analysis):
    - Compare Training Error vs. Train-Dev Error (Variance Check):
        - Scenario: The model performs well on Training data but poorly on Train-Dev data.
        - Observation: Both sets come from the same source (Web), yet the model fails on the unseen one.
        - Conclusion: Overfitting (High Variance). The model is memorizing, not learning.
        - Fix: Regularization, more data, simpler model.
    - Compare Train-Dev Error vs. Dev Error (Mismatch Check):
        - Scenario: The model performs well on Train-Dev (Web) but poorly on Dev (Mobile).
        - Observation: The model generalizes well withinthe Web distribution, but fails when the distribution changesto Mobile.
        - Conclusion: Data Mismatch. The model hasn't learned the features specific to the production environment.
        - Fix: Make training data look more like production data (Artificial Data Synthesis), e.g., adding noise, blur, or simulating bad lighting.