Performance Evaluation

IT461: Practical Machine Learning

Outline

- Hyperparameter Tuning
- Model Evaluation and Selection
- Model Genralization
- Holdout for hyperparamter tuning.
- Model selection via K-fold Cross validation
- Model Fit

```
[ ]  print(f'Best parameters are: {grid.best_params_}')
     mean_score = grid.cv_results_['mean_test_score']
     std_score = grid.cv_results_['std_test_score']
     params = grid.cv_results_['params']
     for mean,std,params in zip(mean_score,std_score,params):
           print(f'{round(mean,2)} + or -{round(std,2)} for the {params}')
```

```
Best parameters are: {'svc__C': 1, 'svc__degree': 1, 'svc__gamma': 0.005}
0.82 + or -0.01 for the {'svc__C': 1, 'svc__degree': 1, 'svc__gamma': 0.005}
```

```
[ ]  model = grid.best_estimator_
     ypred = model.predict(Xtest)

     ##calculate and print the accuracy and f1 scores for SVM with best poly kernel:
     bestpoly_accuracy = accuracy_score(ytest, ypred)
     bestpoly_f1 = f1_score(ytest, ypred, average='weighted')
     print('Accuracy (Best poly Kernel): ', "%.2f" % (bestpoly_accuracy*100))
     print('F1 (Best poly Kernel): ', "%.2f" % (bestpoly_f1*100))
```

```
Accuracy (Best poly Kernel):  87.54
F1 (Best poly Kernel):  87.70
```

```
from sklearn.metrics import classification_report
print(classification_report(ytest, ypred,
                            target_names=faces.target_names))
```

```
                   precision    recall  f1-score   support

     Ariel Sharon       0.92      0.80      0.86        15
     Colin Powell       0.74      0.94      0.83        68
  Donald Rumsfeld       0.83      0.81      0.82        31
    George W Bush       0.96      0.87      0.92       126
Gerhard Schroeder       0.83      0.83      0.83        23
      Hugo Chavez       1.00      0.75      0.86        20
Junichiro Koizumi       1.00      0.92      0.96        12
       Tony Blair       0.87      0.93      0.90        42

         accuracy                           0.88       337
        macro avg       0.89      0.86      0.87       337
     weighted avg       0.89      0.88      0.88       337
```

Hyperparameter tuning

Model Selection

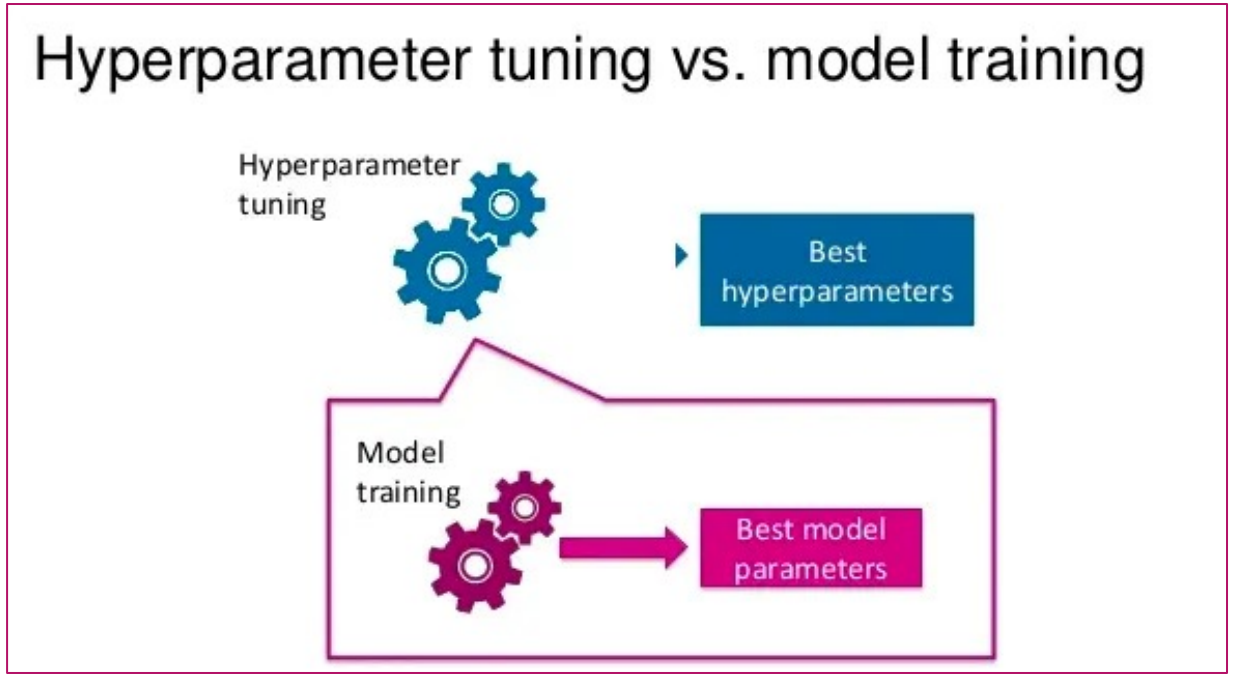
Grid and random Search

Hyperparameter tuning

- **Hyperparameter tuning** (or hyperparameter optimization) focuses on finding the optimal set of hyperparameters for a given model that maximizes the model performance.
- Their correct setting can significantly influence the model's performance.
- Hyperparameters, includes:
  - the learning rate, which determines how quickly a model updates its parameters in response to the training data.
  - the regularization term, which helps prevent overfitting.
- Two main approaches for Hyperparameter tuning: Grid search, Random Search

Hyperparameter tuning

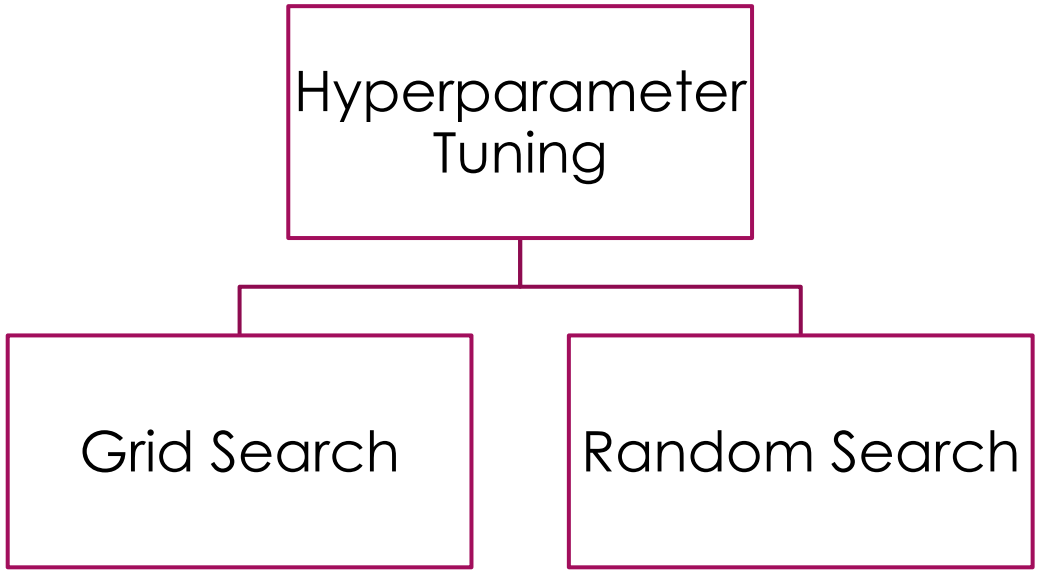
- **Model parameters** are the model's aspects learned from the data during training, such as weights of linear regression model.
- **Hyperparameters:** are the parameters of the learning algorithm.
- Unlike model parameters, these hyperparameters are not learned during training but are set before the training begins.



Model Parameters vs. Hyperparmeters

Model Parameters	Hyperparameters
They are required for making predictions.	They are required for estimating the model parameters.
They are estimated by optimization algorithms(i.e. Gradient Descent).	They are estimated by hyperparameter tuning.
They are not set manually.	They are set manually.
The final parameters found after training will decide how the model will perform on unseen data.	The choice of hyperparameters decide how efficient the training is.

Hyperparameter Tuning



Grid Search

- Grid Search involves exhaustively trying out every possible combination of hyperparameters in a predefined search space.
- For instance, if you're fine-tuning a model and considering two hyperparameters, a grid search would test all combinations of the values you specify for these hyperparameters.

Example: Grid Search

- Let's consider classifying images of handwritten digits (a classic problem known as the MNIST classification). Here, the images are 28x28 pixels, and the goal is to classify them into one of the ten classes (0 through 9).
- For an SVM applied to this problem, two critical hyperparameters are:
  - The type and parameters of the kernel: For instance, if using the Radial Basis Function (RBF) kernel, we need to determine the gamma value.
  - The regularization parameter (C) determines the trade-off between maximizing the margin and minimizing classification error.
- Using grid search, we can systematically explore combinations of:
  - Different kernels: linear, polynomial, RBF, etc.
  - Various values of gamma: e.g., [0.1, 1]
  - Different values of C: e.g., [0.1, 1, 10]
- By training the SVM with each combination and validating its performance on a separate dataset, grid search allows us to pinpoint the combination that yields the best classification accuracy.

Grid Search

- Advantages:
  - Comprehensive: Since it tests all possible combinations, there's a high chance of finding the optimal set.
  - Simple to implement: It doesn't require complex algorithms or techniques.
- Disadvantages:
  - Computationally expensive: As the number of hyperparameters or their potential values increases, the number of combinations to test grows exponentially.
  - Time-consuming: Due to its exhaustive nature, it can be slow, especially with large datasets or complex models.

Random Search

- Random Search, as the name suggests, involves randomly selecting and evaluating combinations of hyperparameters.
- Unlike Grid Search, which exhaustively tries every possible combination, Random Search samples a predefined number of combinations from a specified distribution for each hyperparameter.

Random Search

- Advantages:
  - Efficiency: Random Search can be more efficient than Grid Search, especially when the number of hyperparameters is large. It doesn't need to try every combination, which can save time.
  - Flexibility: It allows for a more flexible specification of hyperparameters, as they can be drawn from any distribution, not just a grid.
  - Surprising Results: Sometimes, Random Search can stumble upon hyperparameter combinations that might be overlooked in a more structured search approach.
- Disadvantages:
  - No Guarantee: There's no guarantee that Random Search will find the optimal combination of hyperparameters, especially if the number of iterations is too low.
  - Dependence on Iterations: The effectiveness of Random Search is highly dependent on the number of iterations. Too few iterations might miss the optimal settings, while too many can be computationally expensive.

Model Evaluation and Selection

- Model evaluation is the process that uses some metrics which help us to analyze the performance of the model. .
- Model selection is the process of choosing between different models, different model types, tuning parameters, and features.
- Model selection involves tuning and comparing different parameter settings to select the optimal values of parameters that would improve the performance for making predictions on unseen data.

Model generalization

- Model generalization: assessing predictive accuracy for new data.
- The primary goal of supervised machine learning is accurate prediction.
- ML model should be as accurate as possible when predicting on new data (for which the target variable is unknown).
- The model, which has been built from training data, must generalize well to new data. So, when the model is deployed, we can be assured that the predictions generated are of high quality.
- Therefore, when we evaluate the performance of a model, we want to determine how well that model will perform on new data.
- How?

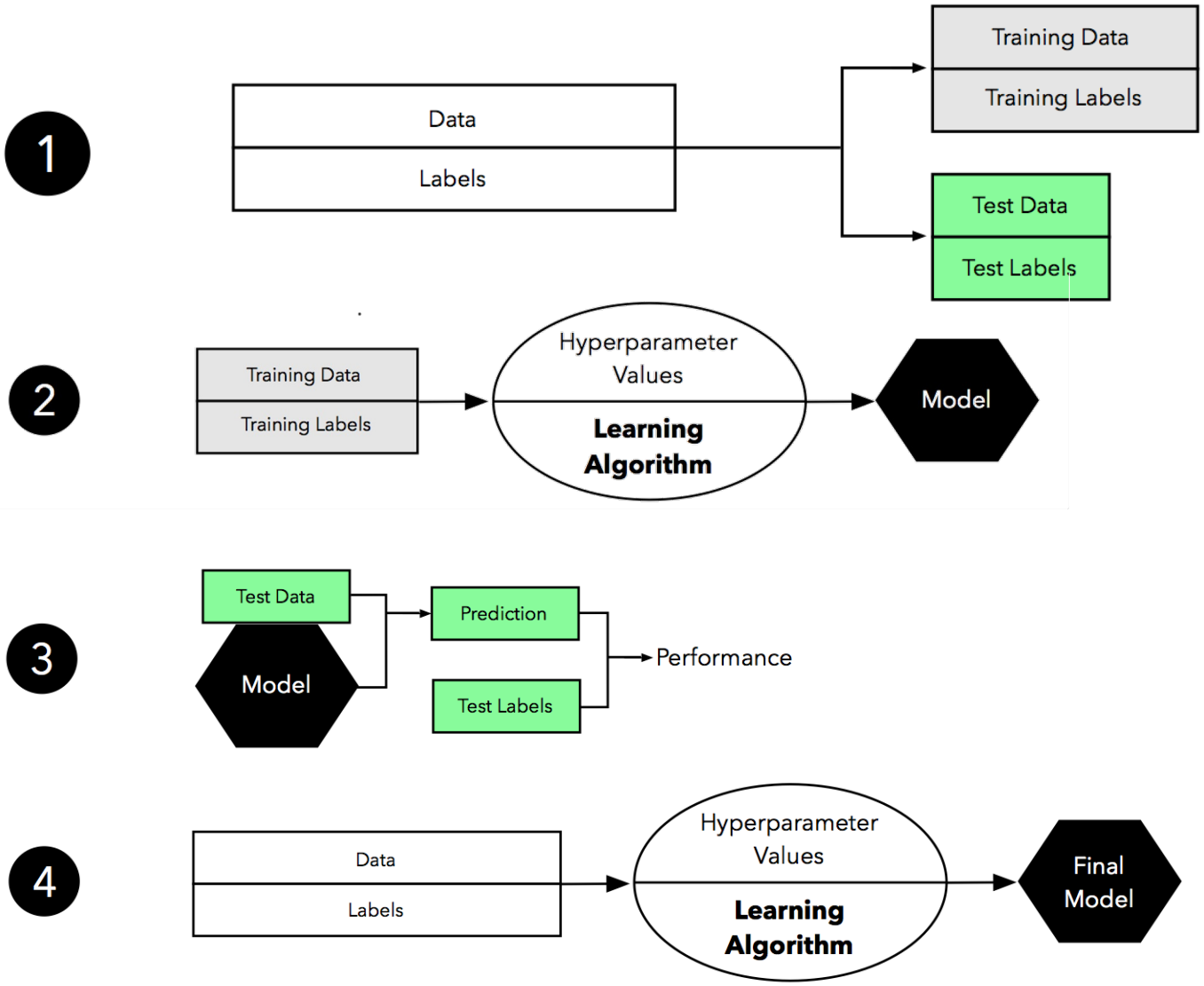
Holdout : Train/test split

- Using the holdout method: split initial dataset into separate two subsets.
  - Training datasets - used for model training.
  - Test datasets - used to estimate the generalization performance of machine learning models to check how well a model learns and generalizes to the new data.
- If the same test dataset is used over and over again during model selection, it will become part of the training data and thus the model will be more likely to overfit.

Basic Idea:

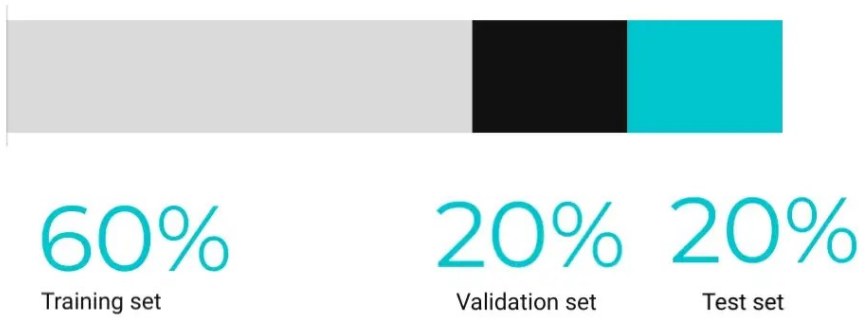
Randomly split data into train and test set
Train a model on training set and predict on the test set and check accuracy.

Holdout : Train/test split



Holdout: Train/validation/test split

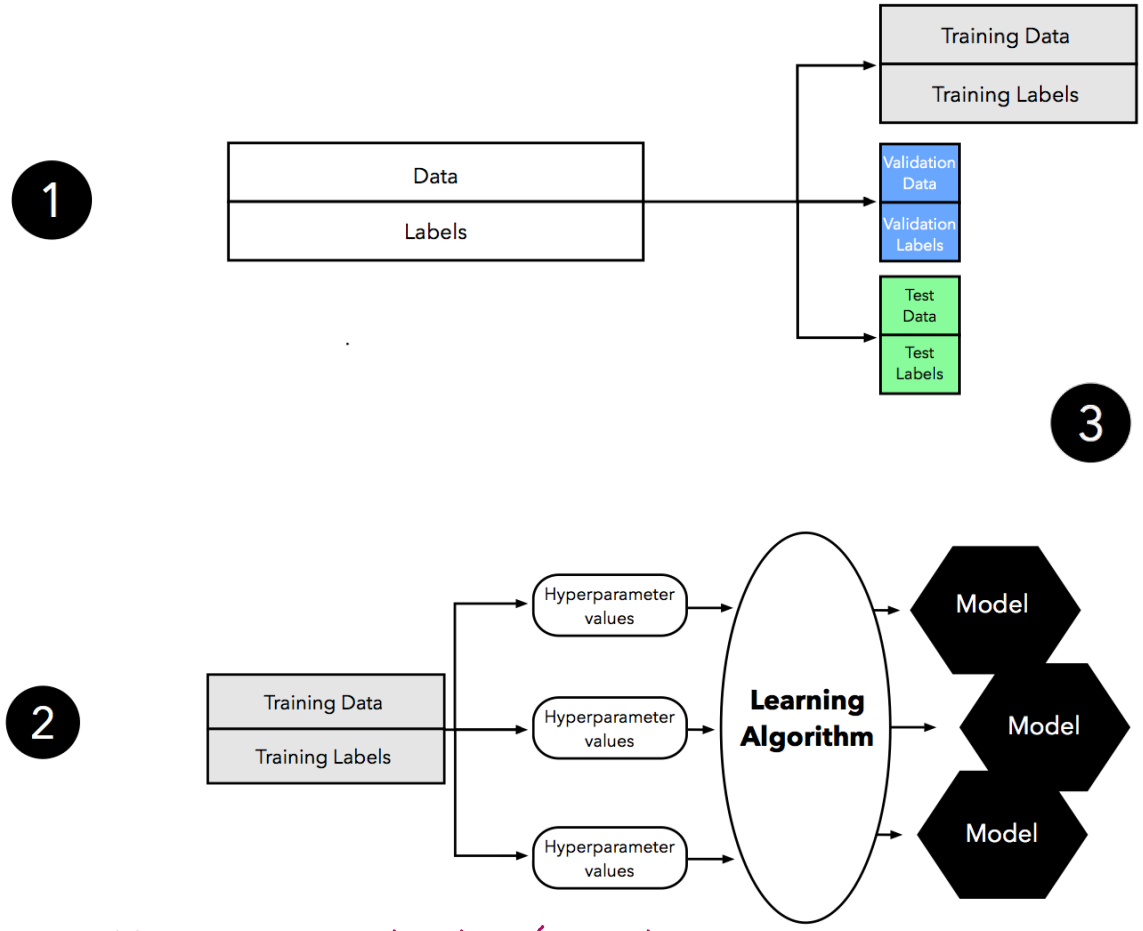
- A better way of using the holdout method for model selection is to separate the data into three parts: a training dataset, a validation dataset, and a test dataset.
  - the training dataset is used to fit the different models.
  - the performance on the validation dataset is then used for the model selection.
- Having a training-validation pair for hyperparameter tuning and model selections allows us to keep the test set "independent" for model evaluation.
- The advantage of having a test dataset that the model hasn't seen before during the training and model selection steps is that we can obtain a less biased estimate of its ability to generalize to new data.



Holdout: Train/validation/test split for Hyperparamter Tuning

- Step 1: split the dataset into three parts, a training set for model fitting, a validation set for model selection, and a test set for the final evaluation of the selected model.
- Step 2: use the learning algorithm with different hyperparameter settings to fit models to the training data.
- Step 3: evaluate the performance of the models on the validation set: comparing the performance estimates and choose the hyperparameters settings associated with the best performance.
- Step 4: merge the training and validation set after model selection and use the best hyperparameter settings from the previous step to fit a model to this larger dataset.
- Step 5: use the independent test set to estimate the generalization performance of the best model.
- Step 6: use all data -merging training and test set — and fit a model for real-world use. Using all data (that is, training and test data) to fit the model should only improve its performance.

Holdout: Train/validation/test split



Hyperparameter tuning stage:

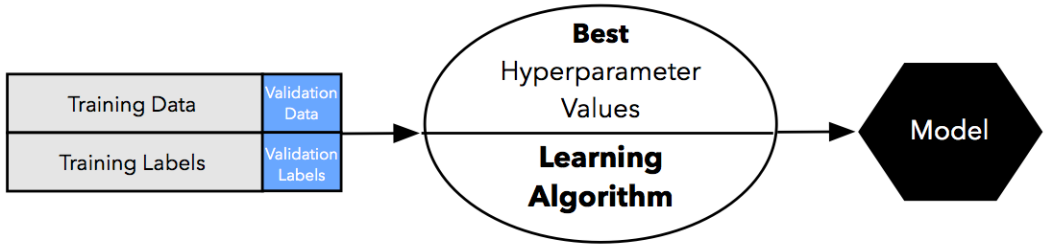
Train many models on Training data

Model selection stage:

Pick the best hyperparameters using Validation data

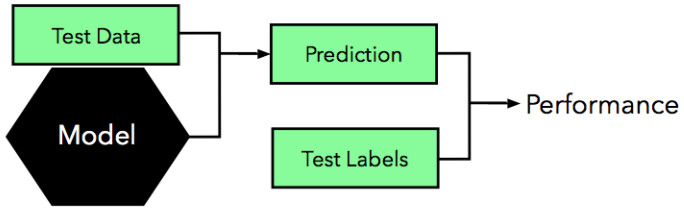
Holdout: Train/validation/test split

4



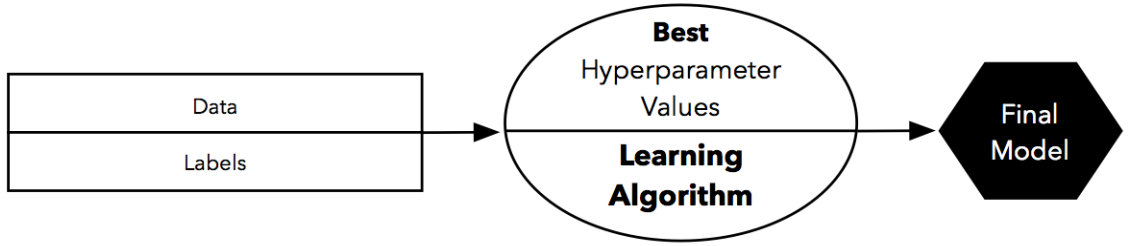
Train a model with best hyperparameters on Training + Validation

5



Compute accuracy on test data ← estimated performance

6



Train the final model on all data ← final model (deployed)

Cross-Validation

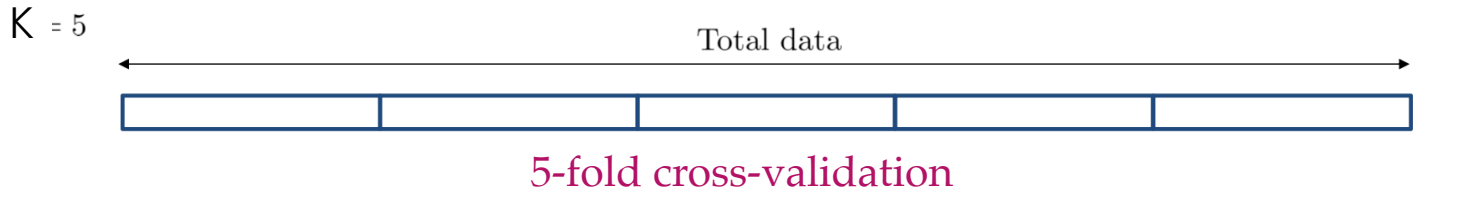
- The main idea behind cross-validation is that each sample in the dataset has the opportunity of being tested.
- K-fold cross-validation is a special case of cross-validation where we iterate over a dataset set k times.
- In each round, we split the dataset into k parts: one part is used for validation, and the remaining k-1 parts are merged into a training subset for model training.

Cross-Validation

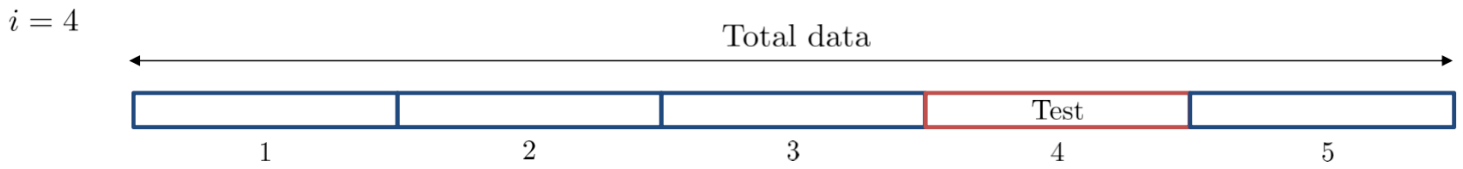
- Partition data into K parts

Repeat K times
- (step i) Test = partition i; Training = the rest
- (step i) Compute accuracy of step i

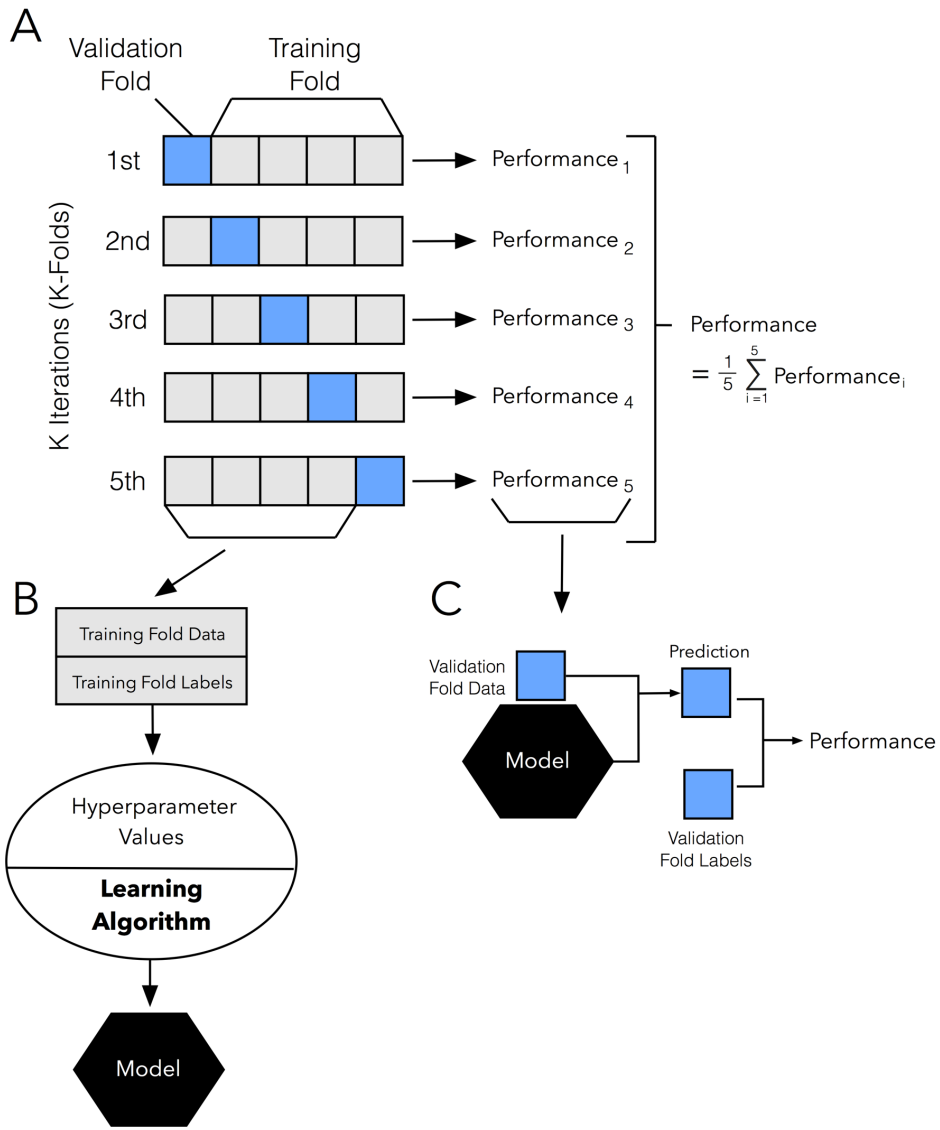
Average K accuracy measures at the end



5-fold cross-validation



K-fold Cross-Validation

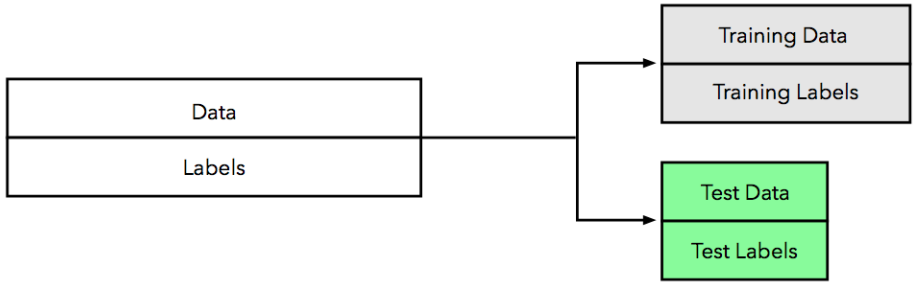


Model Selection via K-fold Cross-validation

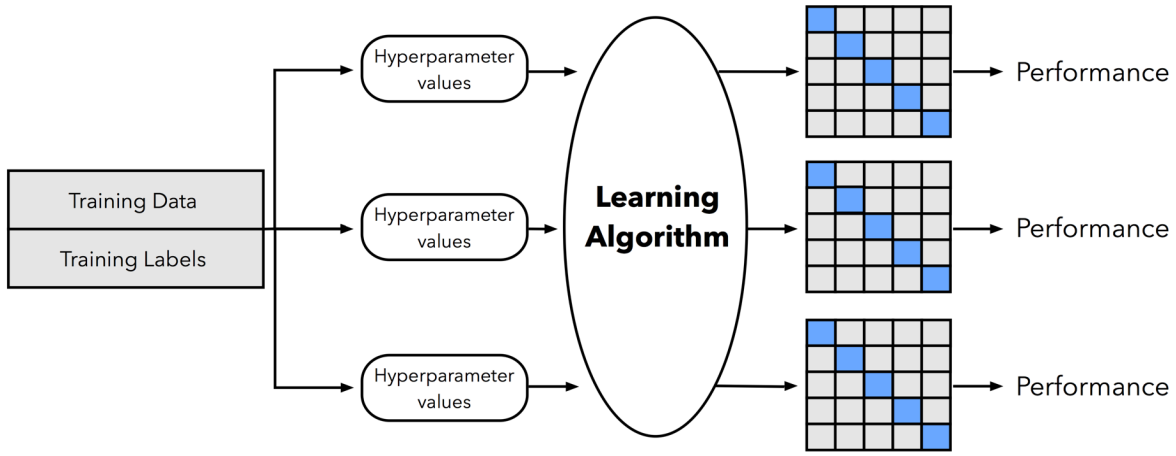
- **Step 1:** split dataset into two parts, a training and an independent test set.
- **Step 2:** For each hyperparameter configuration, apply the k-fold cross-validation on the training set, resulting in multiple models and performance estimates. (use Randomized Search, or Grid Search with various hyperparameter settings)
- **Step 3:** fit a model using the complete training set with the hyperparameter settings that correspond to the best-performing model.
- **Step 4:** use the test set to evaluate the model that we obtained from step 3.
- **Step 5:** fit a model to all data, which could be the model for (the so-called) deployment.

Model Selection via K-fold Cross-validation

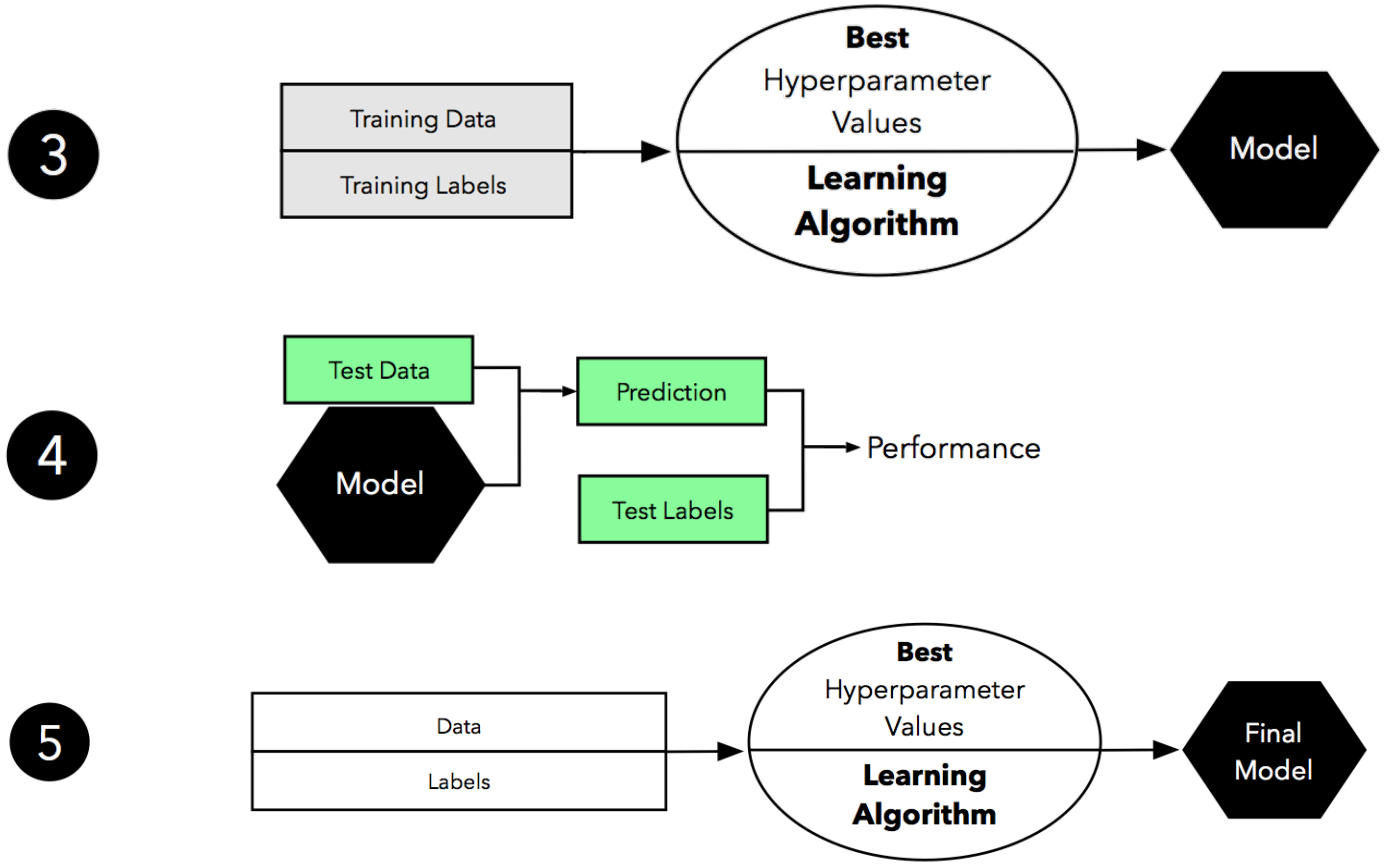
1



2



Model Selection via K-fold Cross-validation



Final model: retrain the best performing model on the entire data.

Model Fit

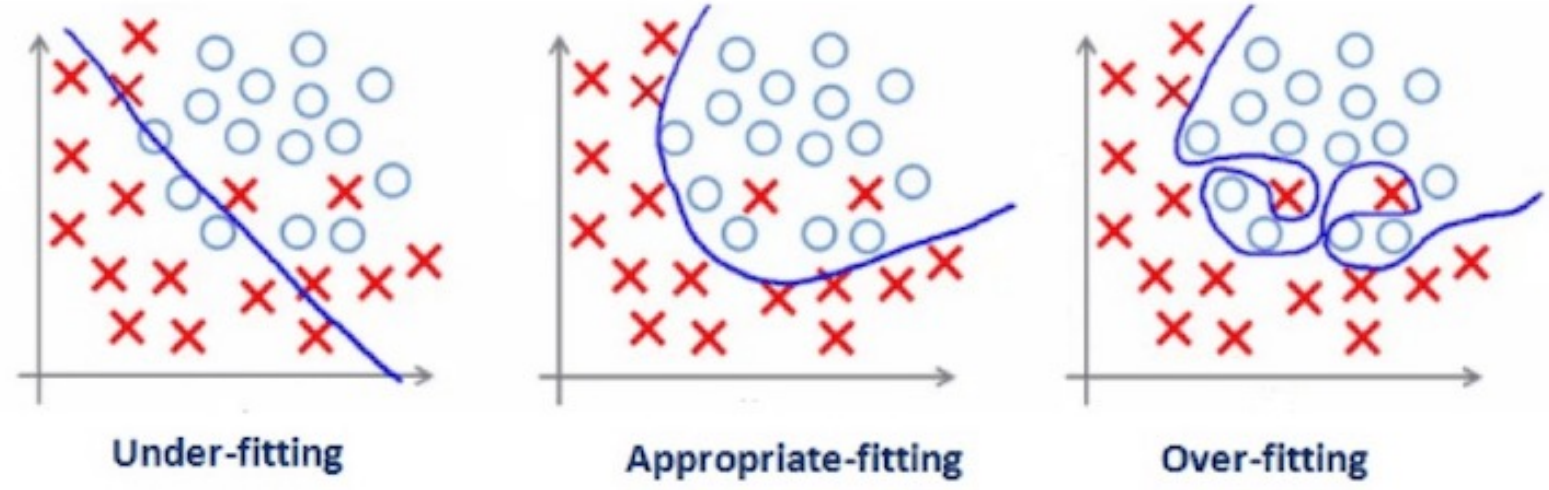
Underfit/ overfit

Bias and Variance

Learning curves

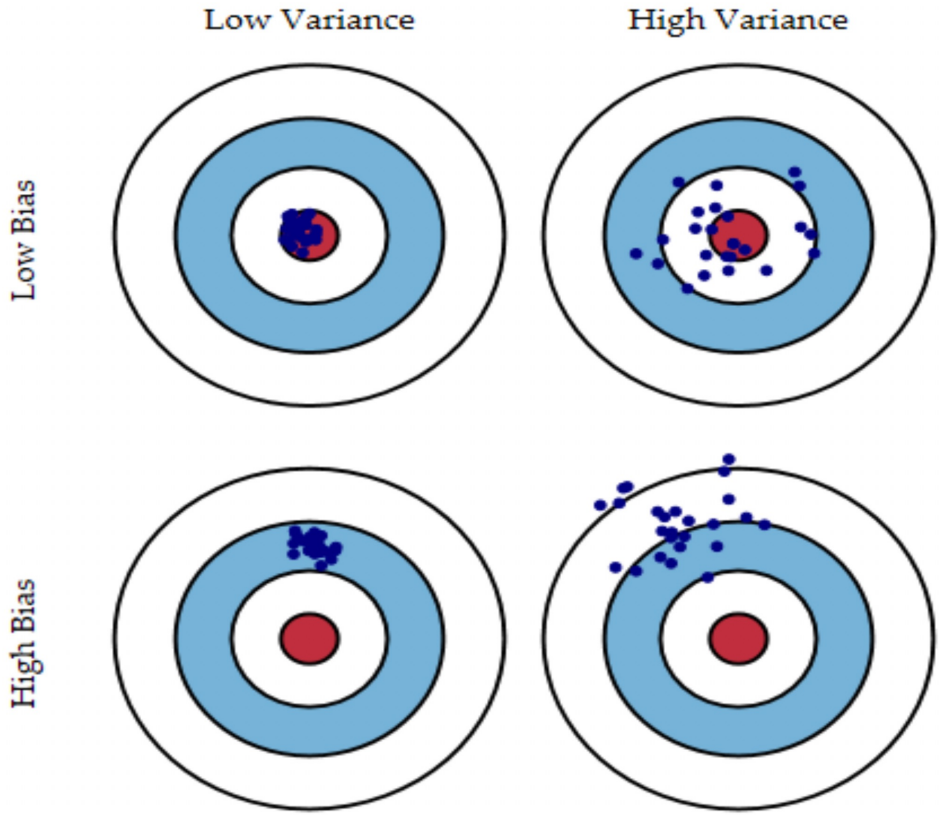
Model Fit: Underfitting / Overfitting

- **Overfitting** means that the model works well on the training set but is unable to perform better on the test sets. It is known as the problem of **high variance**.
- **Underfitting** means the model works so poorly that it is unable to fit even training set well. It is known as the problem of **high bias**.



Bias – Variance

- Bias measures how far off, in general, the predictions are from the actuals.
- The variance is how much the prediction varies.



overfitting

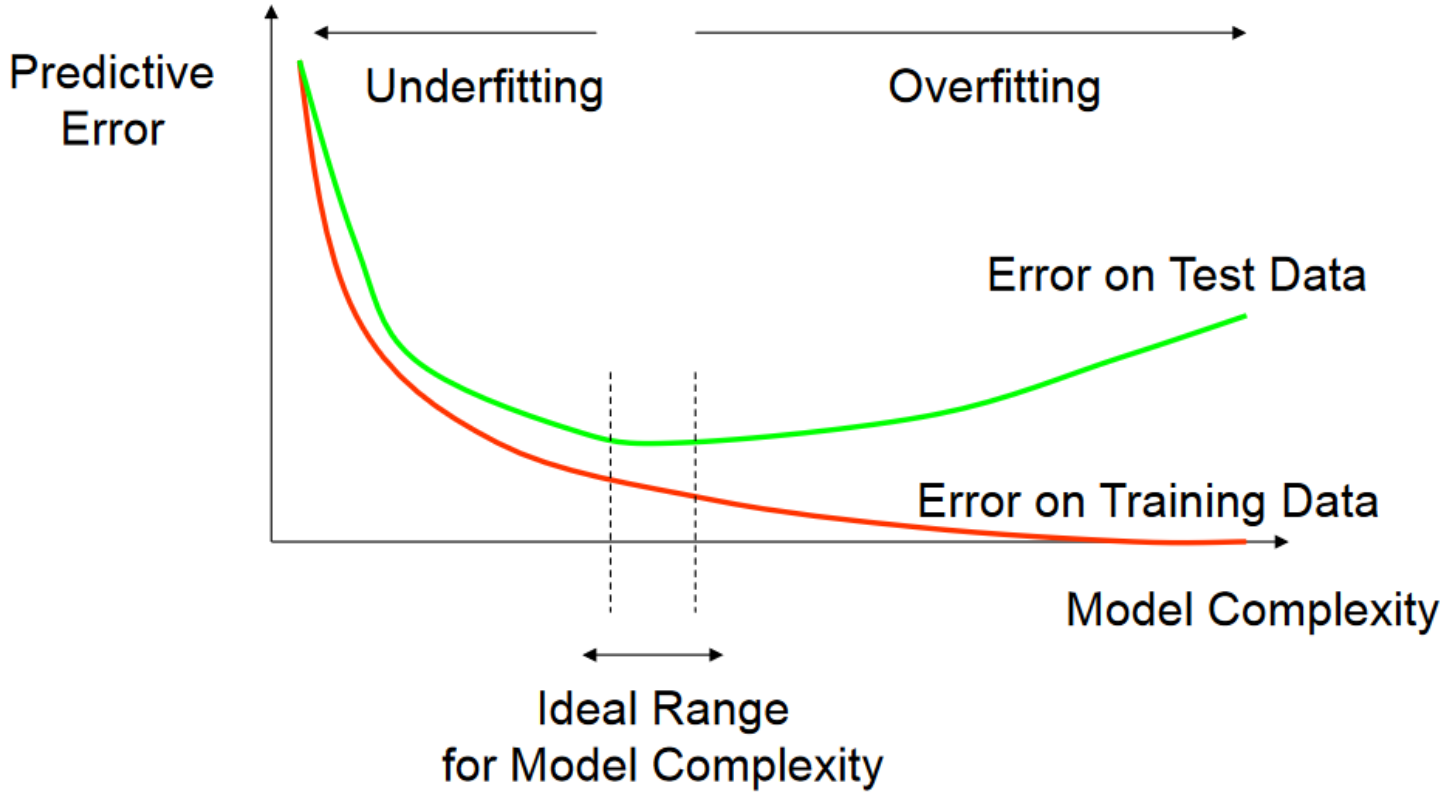
underfitting

***Bias** : the average of distance between the prediction and the target.*

***Variance**: Variability in the predictions.*

Bias – Variance Trade-Off

- Bias is reduced and variance is increased in relation to model complexity.

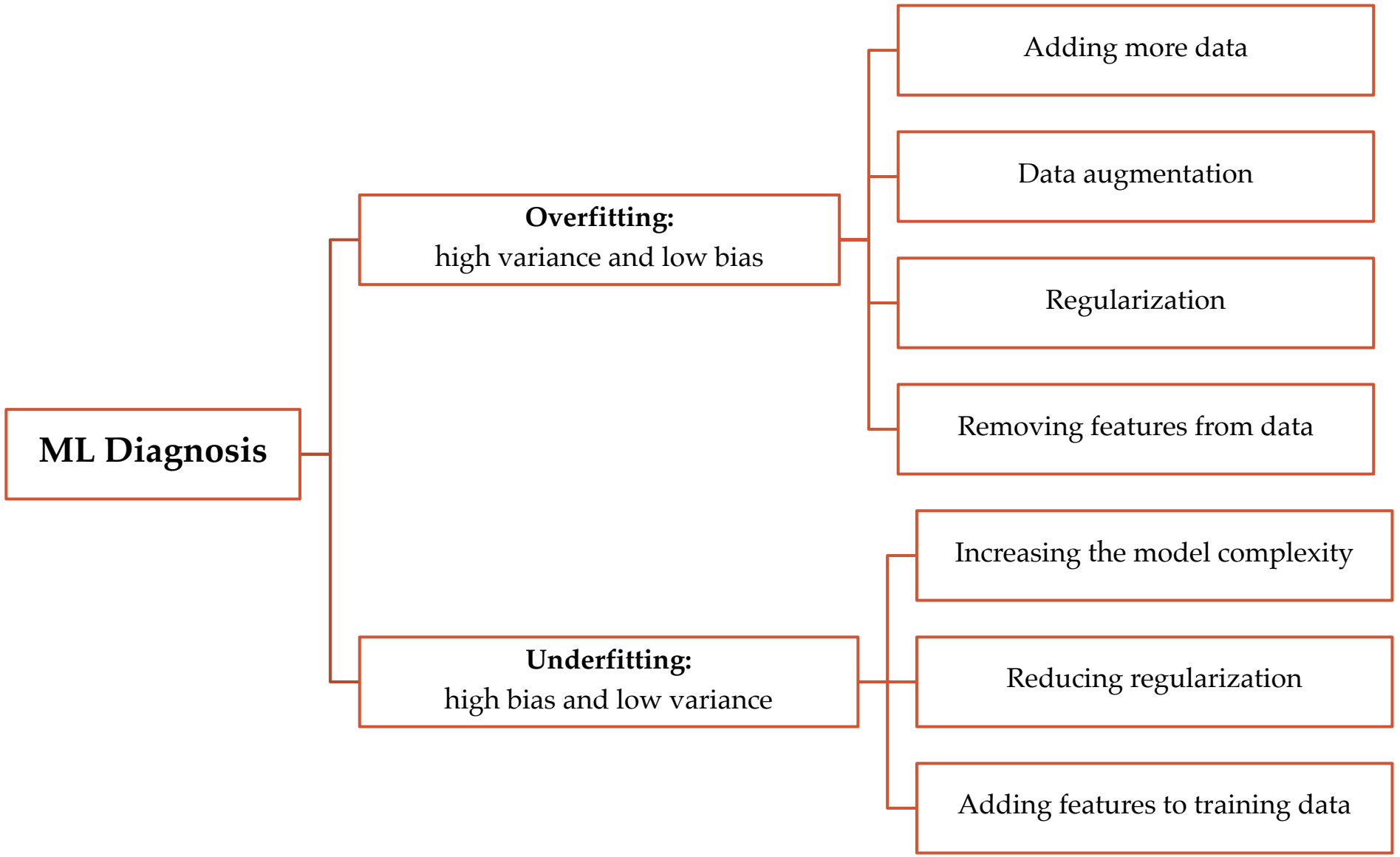


How to resolve overfitting?

- Adding more data:
  - The model is overfitting when it fails to generalize to new data. That means the data it was trained on is not representative of the data. duction. So, retraining your algorithm on a bigger, richer and more diverse data set should improve its performance.
- Data augmentation:
  - If getting more data can prove to be very difficult; either because collecting it is very expensive or because very few samples are regularly generated.
  - Data augmentation is a set of techniques used to artificially increase the size of a dataset by applying transformations to the existing data.
- Regularization:
  - Regularization are very powerful to avoid overfitting.
- Removing features from data:
  - The model may fail to generalize because the data it was trained on was too complex and the model missed the patterns it should have detected.
  - Removing some features and making the data simpler can help to reduce overfitting.

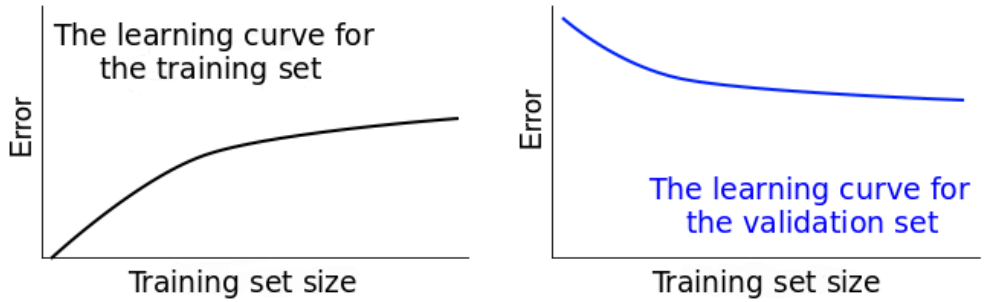
How to resolve underfitting?

- Increasing the model complexity
  - The model may be underfitting simply because it is not complex enough to capture patterns in the data. Using a more complex model will very often help solve underfitting.
  - For instance, by switching from a linear to a non-linear model or by adding hidden layers to a neural network.
- Reducing regularization
  - The used algorithms include by default regularization parameters meant to prevent overfitting. Sometimes, they prevent the algorithm from learning.
- Adding features to training data
  - In contrast to overfitting, the model may be underfitting because the training data is too simple. It may lack the features that will make the model detect the relevant patterns to make accurate predictions. Adding features and complexity to the data can help overcome underfitting.


ML Diagnosis
Overfitting:
high variance and low bias
Adding more data
Data augmentation
Regularization
Removing features from data
Underfitting:
high bias and low variance
Increasing the model complexity
Reducing regularization
Adding features to training data

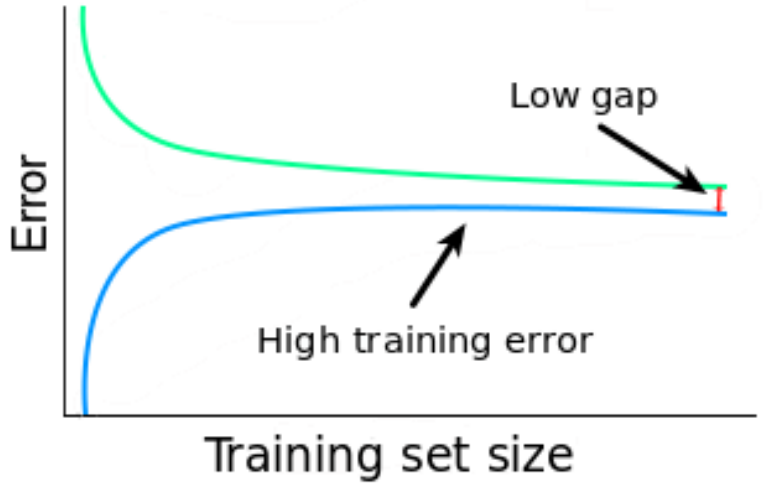
Learning Curve

- Learning curves show the relationship between training set size and the chosen evaluation metric (e.g., accuracy, etc.) on training and validation sets.
  - to diagnose the model performance, whether the model is suffering from bias or variance.
- The graph will display two curves:
  - Training curve: The curve calculated from the training data; used to inform how well a model is learning.
  - Validation curve: The curve calculated from the validation data; used to inform of how well the model is generalizing to unseen instances.
- These curves show us how well the model is performing as the data grows.



Learning curves – underfitting

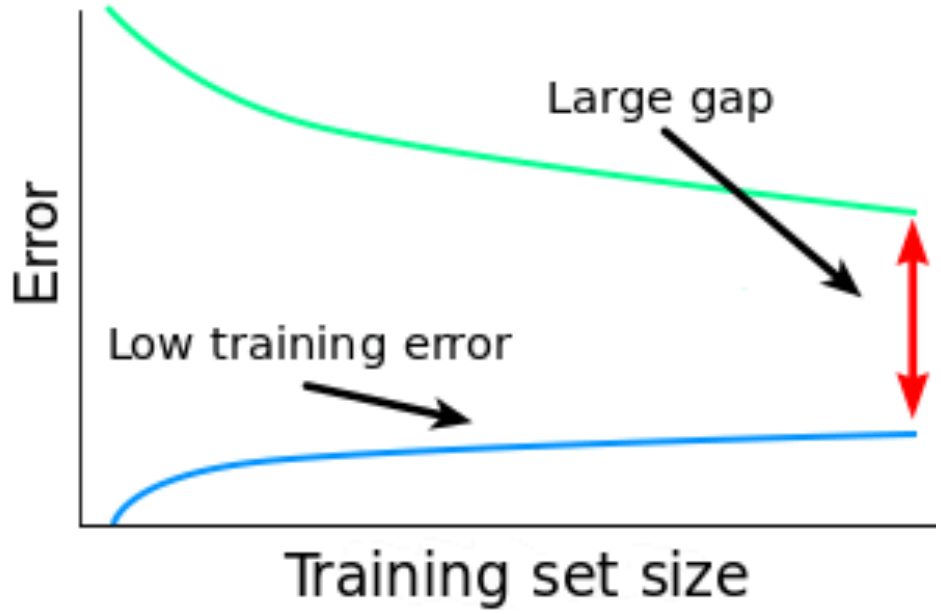
- The training error is high, the training data is not fitted well enough by the estimated model.
- The model is performing similarly bad for both the training and validation sets.
- Both the training and validation error is high and it doesn't seem to improve with more training examples.



high bias and low variance
underfitting the training data

Learning curves – overfitting

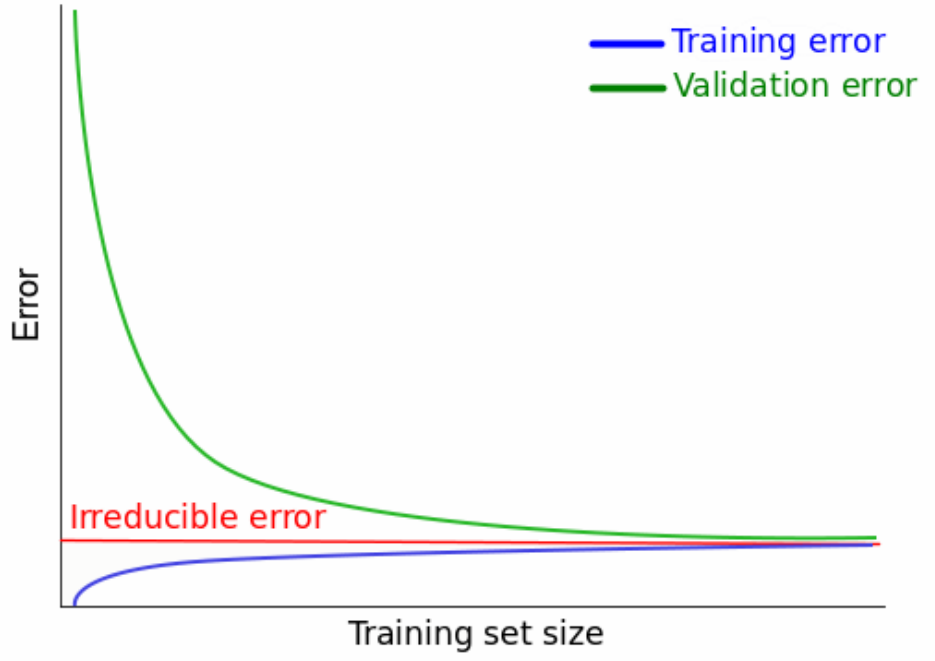
- The large gap and the low training error indicates an overfitting problem.



high variance and low bias
overfitting the training data

The ideal learning curves

- Irreducible error cannot be reduced by building better models.
- The best possible learning curves converge to the value of some irreducible error.



Learning Curves using Accuracy

- Accuracy describes how good the model is. The higher the accuracy, the better.
- Accuracy = 1 - Error