Statistics Foundation

Model Evaluation

Digital Vidya

Model Evaluation

- **Loss/Objective function**
  - This is a function of the difference between estimated (Y`) and the ground truth (Y)
  - During training, we minimize the loss function in order to learn the functional relationship
- **Parameter & Hyper-parameters**
  - Parameters of a model establish the rules of how X will translate into Y`
  - They are learnt during the model training (minimizing loss function)
  - For ex: In regression, coefficients of features are the parameters
  - Hyper-parameters control how model will learn it’s parameters
  - They are fixed before the model training starts
  - For ex: k in k-nearest neighbours algorithm is a hyper-parameter.

Digital Vidya

Trade-off

- **What is bias?**

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

- **What is variance?**

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

Trade-off

- **Mathematical Derivation**

Let the variable we are trying to predict as Y and other covariates as X. We assume there is a relationship between the two such that

Y=f(X) + e

Where e is the error term and it's normally distributed with a mean of 0.
We will make a model f^(X) of f(X) using linear regression or any other modeling technique.
So the expected squared error at a point x is

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

The Err(x) can be further decomposed as

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

Trade-off

- **Mathematical Derivation**

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

Err(x) is the sum of
Bias^2, variance
and the irreducible error

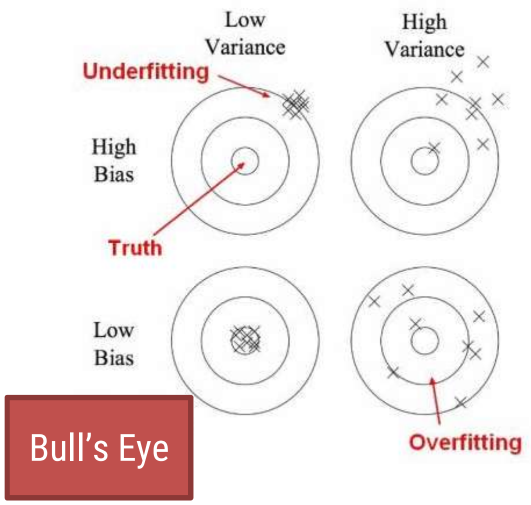
Irreducible error is the error that can't be reduced by creating good models.
It is a measure of the amount of noise in our data.
No matter how good we make our model, our data will have certain amount of noise or irreducible error that can not be removed.

Digital Vidya

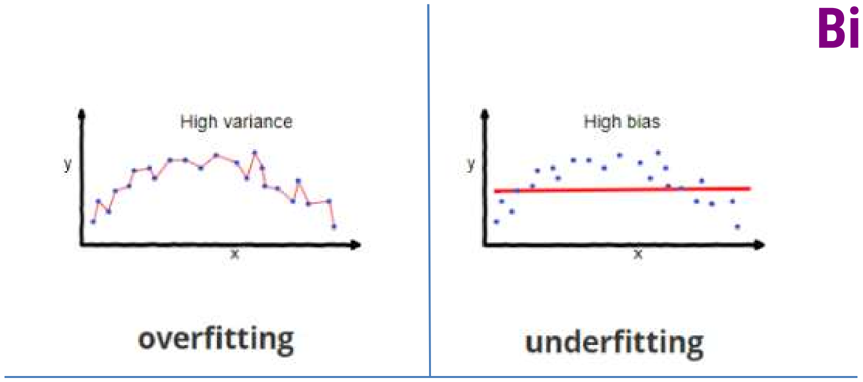
Trade-off

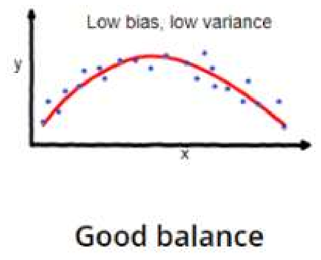
Bias and variance trade-off

- In supervised learning, under-fitting happens when a model unable to capture the underlying pattern of the data
- These models usually have high bias and low variance
- It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data
- Also, these kind of models are very simple to capture the complex patterns in data like Linear and logistic regression



Trade-off





Bias and variance trade-off

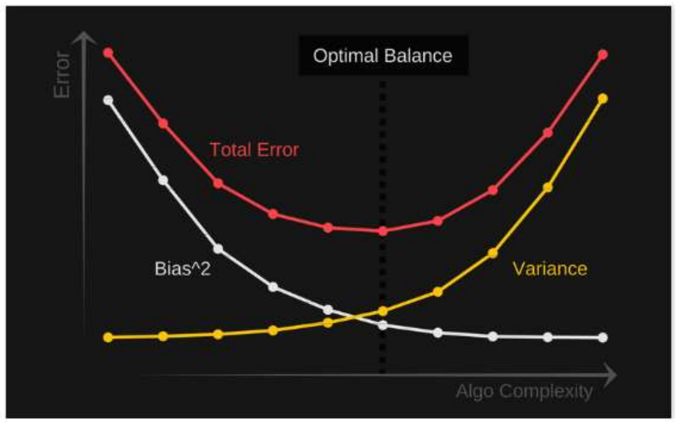
- In supervised learning, overfitting happens when our model captures the noise along with the underlying pattern in data
- It happens when we train our model a lot over noisy dataset
- These models have low bias and high variance
- These models are very complex like Decision trees which are prone to overfitting

Model Evaluation

Total Error

- To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error
- An optimal balance of bias and variance would never over-fit or under-fit the model

Total Error = Bias^2 + Variance + Irreducible Error



The testing data/environment for model evaluation should be as close to what is expected post deployment

This should inspire the choice of validation strategy

Digital Vidya

Model Evaluation

Validation Strategy

Validation strategies can be broadly divided into 2 categories:

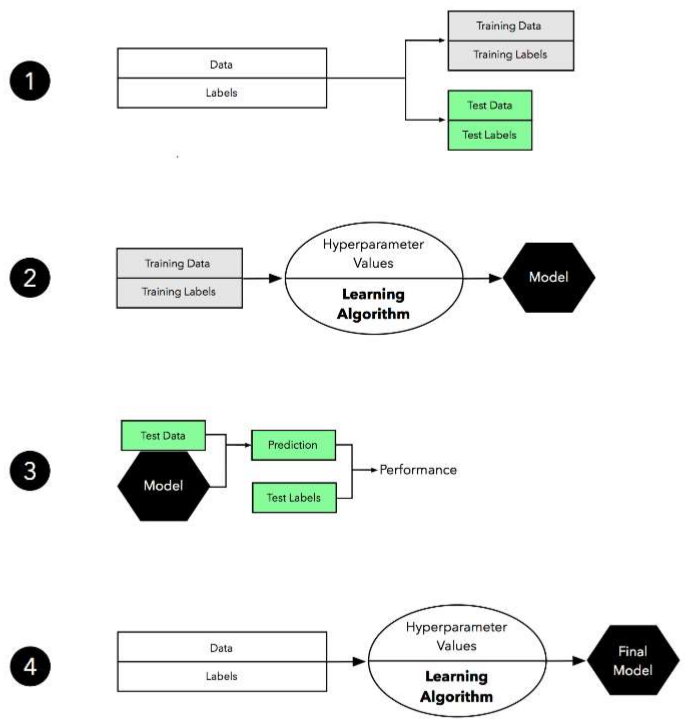
- Holdout validation
- Cross validation

Model Evaluation

Holdout Validation

Single Holdout

- The basic idea is to split our data into a training set and a holdout test set
- Train the model on the training set and then evaluate model performance on test set
- We take only a single holdout—hence the name



Source: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part1.html

Digital Vidya

Model Evaluation

Holdout Validation

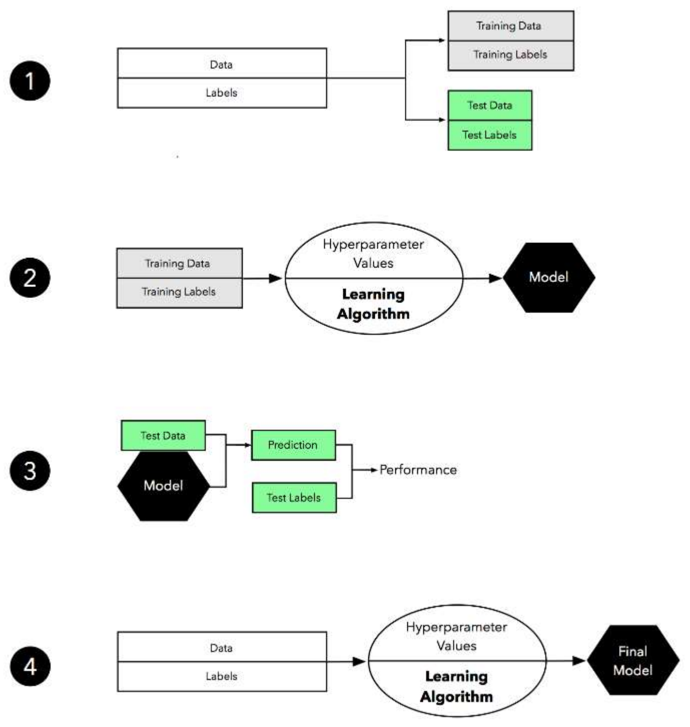
Single Holdout

Step 1: Split the labelled data into 2 subsets (train and test).

Step 2: Choose a learning algorithm. (For ex: Random Forest). Fix values of hyper-parameters. Train the model to learn the parameters.

Step 3: Predict on the test data using the trained model. Choose an appropriate metric for performance estimation (ex: accuracy for a classification task). Assess predictive performance by comparing predictions and ground truth.

Step 4: If the performance estimate computed in the previous step is satisfactory, combine the train and test subset to train the model on the full data with the same hyper-parameters.



Source: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part1.html

Digital Vidya

Model Evaluation

- **Stratified sampling**
  - While splitting, we need to ensure that the distribution of features as well as target remains the same in the training and test sets
  - For ex: Consider a problem where we're trying to classify an observation as fraudulent or not
  - While splitting, if the majority of fraud cases went to the test set, the model won't be able to learn the fraudulent patterns, as it doesn't have access to many fraud cases in the training data
  - In such cases, stratified sampling should be done, as it maintains the proportion of different classes in train and test set
  - Stratified sampling should be used for splitting the data almost always
  - ***In production, if we expect the distribution to be very different from what we have in our present data, stratification may not be a good choice***

With more training data, the model's predictive power should improve

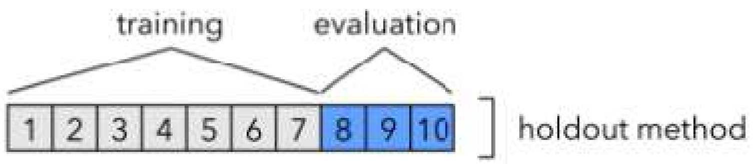
Model Evaluation

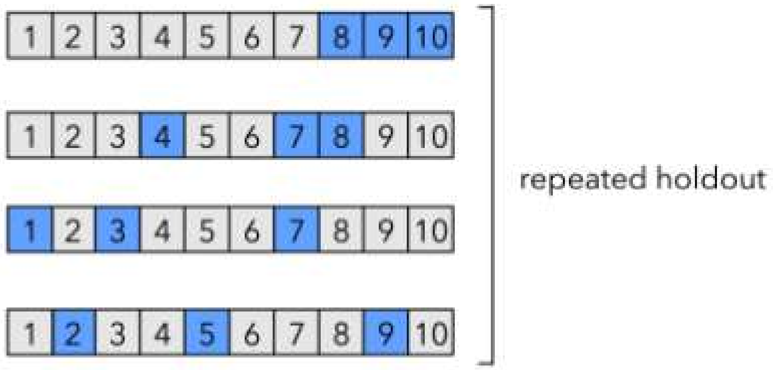
- **Repeated holdout**
  - One way to obtain a more robust performance estimate that's less variant to how we split the data is to repeat the holdout method k times with different random seeds
  - The estimate of predictive power of the model will be the average performance over these k repetitions
  - In case of accuracy as a proxy for predictive power:

$$\mathrm{ACC}_{avg} = \frac{1}{k}\sum_{j=1}^{k} ACC_j,$$

Model Evaluation

- **Repeated holdout**
  - In repeated holdout, we can test our model on a higher number of samples compared to single holdout. This reduces the variance of the final estimate.



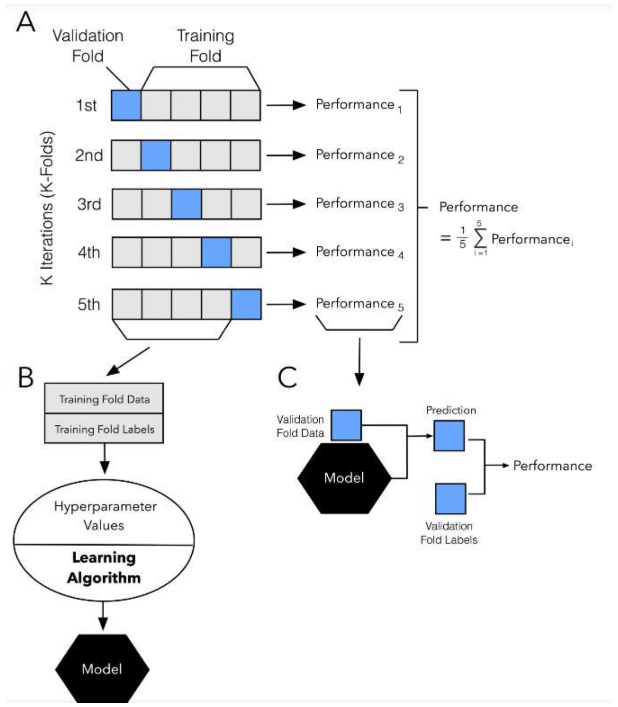


Digital Vidya

Model Evaluation

- **Cross Validation**
  - **k-fold cross validation**

- In repeated holdout, we can test our model on a higher number of samples compared to single holdout
- This reduces the variance of the final estimate

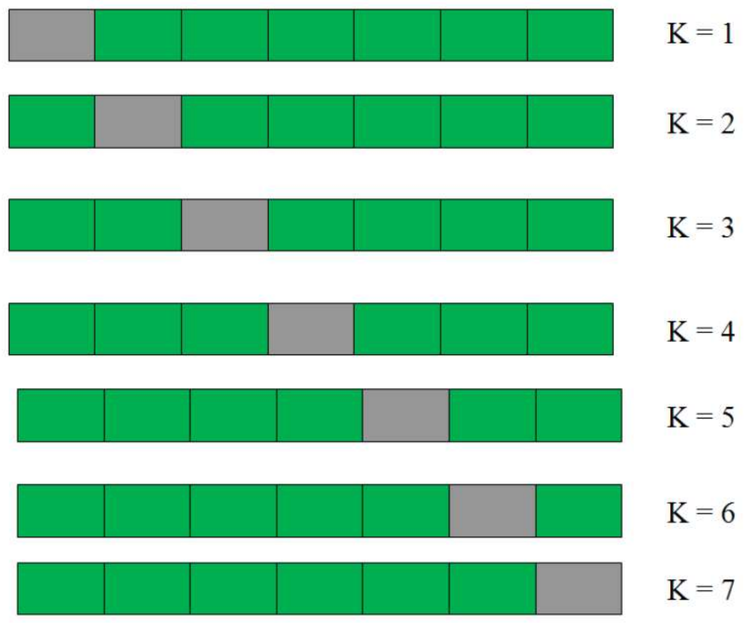


Digital Vidya

Model Evaluation

- **Cross Validation**
  - **k-fold cross validation**

    - Choose a value of k and divide the data into k equal subsets
    - Combine k-1 subsets and consider it as a training fold and the remaining one as a test fold
    - Conduct the holdout method to get test performance (let’s choose accuracy for now)
    - Repeat 2nd and 3rd steps, k times with a different subset as test fold
    - Point estimate of predictive power is the average of the k different test accuracies

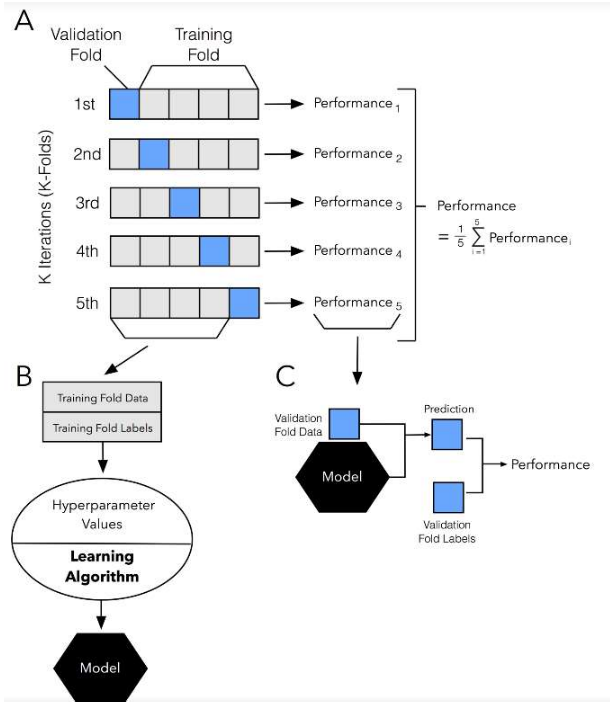


7 Fold Cross-Validation

Digital Vidya

Model Evaluation

- **Cross Validation**
  - **k-fold cross validation**
    - In cross-validation, every sample in our data is part of the test set exactly once. The model is tested on every sample
    - Experimental studies have suggested that cross validation estimates are less biased compared to holdout estimates
    - A special case when k = n (# of samples on data) is also called leave one out cross validation is useful when working with extremely small datasets
    - Another variation of k-fold is to repeat k-fold multiple times and take the average of performances across all the iterations. This reduces the variance further
    - Computationally less expensive than repeated holdout but more expensive than single holdout
    - For data with dependent observations like time series data, the cross validation scheme discussed above should not be used



Digital Vidya

Thank You

Digital Vidya