MODEL EVALUATION

Why do we need Model Evaluation?

Let's understand why Model Evaluation is important. Suppose you are working on a supervised machine learning problem and have built a model for it. Can you deploy this model without understanding its success or performance? Of course not! You need to evaluate the performance of your model before deploying it to ensure it is reliable and effective.This is where Model Evaluation comes into play. It helps you assess your model's performance and determine whether it is suitable for deployment.

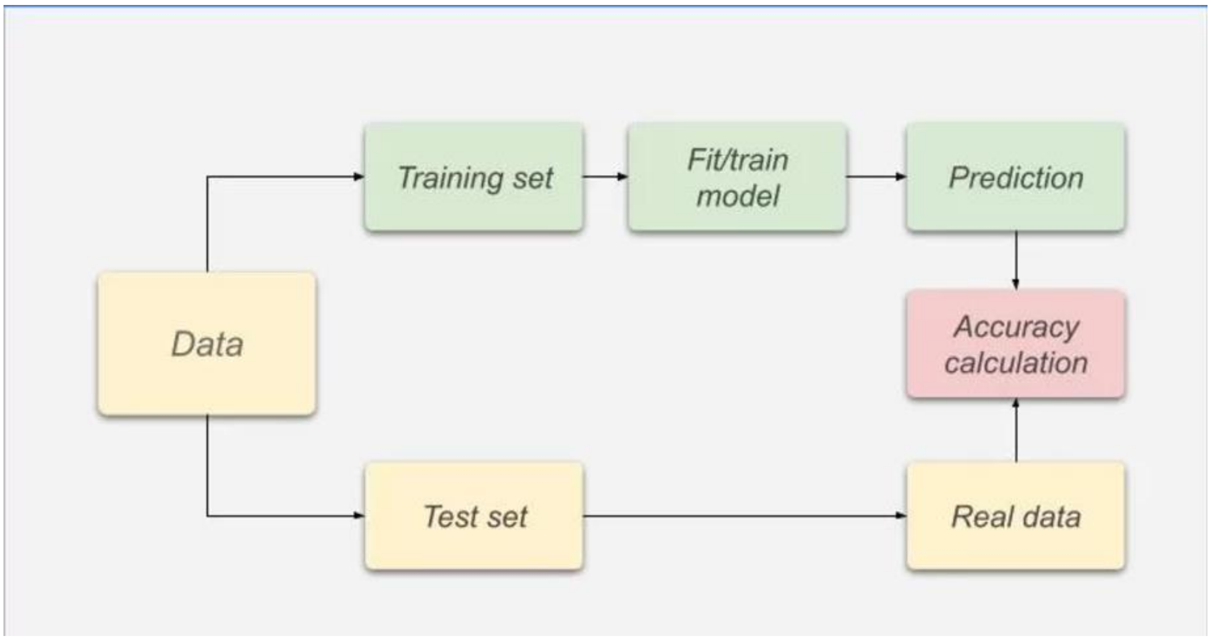
To evaluate your model's performance, you generally use two approaches:

1. The Holdout Approach

2. Cross-Validation

1. The Holdout Approach :

The holdout approach is a very straightforward method. You simply shuffle the data and divide it into two parts: a training set and a test set. You train your model on the training data and then make predictions on the test data. The predicted values are compared with the actual values using performance metrics.



Yogeshchouhan263@gmail.com

This is a simple and intuitive method, but it comes with several limitations:

Problems with the Holdout Approach:

1. **Variability:**
   The performance of the model can be highly sensitive to how the data is divided into training and testing sets. If the split is not representative, the training set might not capture the overall distribution of the data, or the test set could contain unusually easy or difficult examples. This results in high variance in the performance estimation.
2. **Data Inefficiency:**
   The holdout method only uses a portion of the data for training and another portion for testing. This means the model doesn't get to learn from the entire dataset, which can be particularly problematic when the dataset is small.
3. **Bias in Performance Estimation:**
   If certain classes or patterns are overrepresented or underrepresented in the training or test sets due to the random split, it can lead to biased performance estimates.
4. **Less Reliable for Hyperparameter Tuning:**
   When the holdout method is used for hyperparameter tuning, there is a risk of overfitting to the test set. This happens because information from the test set can inadvertently influence the model. Consequently, the performance estimation might be overly optimistic and not representative of the model's performance on unseen data.

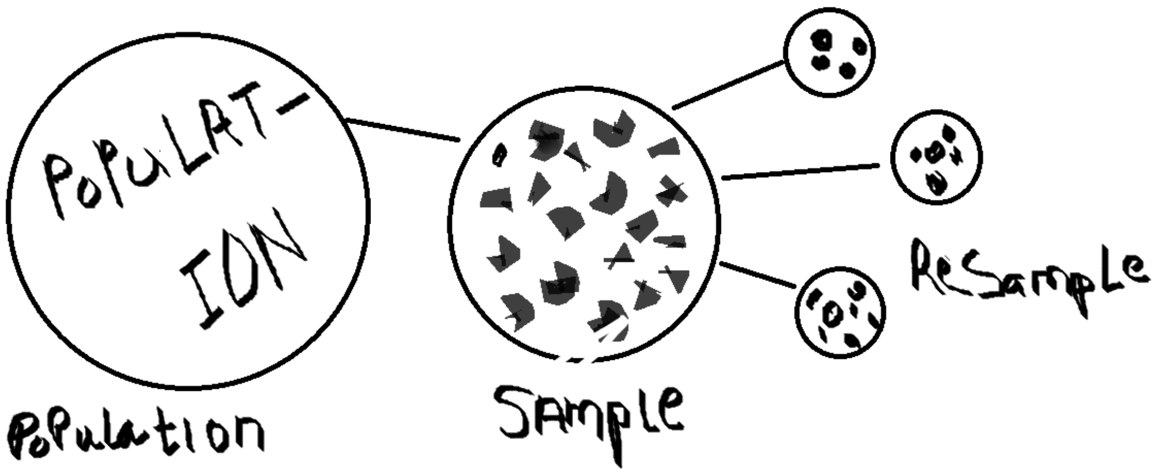
Why Use the Holdout Approach Despite Its Limitations?

1. **Simplicity:**
   The holdout method is straightforward and easy to understand. You simply divide your data into two sets: training and test. This simplicity makes it appealing, especially for initial exploratory analysis or simpler projects.
2. **Computational Efficiency:**
   The holdout method is computationally less expensive compared to methods like k-fold cross-validation. In k-fold cross-validation, you need to train and test your model multiple times (k times), which can be resource-intensive, especially for large datasets or complex models. With the holdout method, you train the model only once.
3. **Effective for Large Datasets:**
   For very large datasets, even a small proportion of the data may be sufficient to form a representative test set. In such cases, the holdout method can work effectively without compromising the evaluation's reliability.

2. Cross Validation :

'Cross-validation works on the concept of resampling, which means taking samples from an existing sample dataset. For example, consider the Iris dataset, which itself is a sample of all flower species. Generating subsets (samples) from the Iris dataset is an example of resampling.

In essence, cross-validation relies on the idea of resampling, where you have a sample dataset (most datasets are sample data because storing population data is practically impossible in this world). Cross-validation allows us to generate subsets (samples) from our sample dataset using various approaches. These approaches are essentially the types of cross-validation.



The most well-known and widely used cross-validation techniques are:

1. **Leave-One-Out Cross-Validation (LOOCV)**
2. **K-Fold Cross-Validation**
3. **Stratified K-Fold Cross-Validation**

1. Leave-One-Out Cross-Validation (LOOCV) :

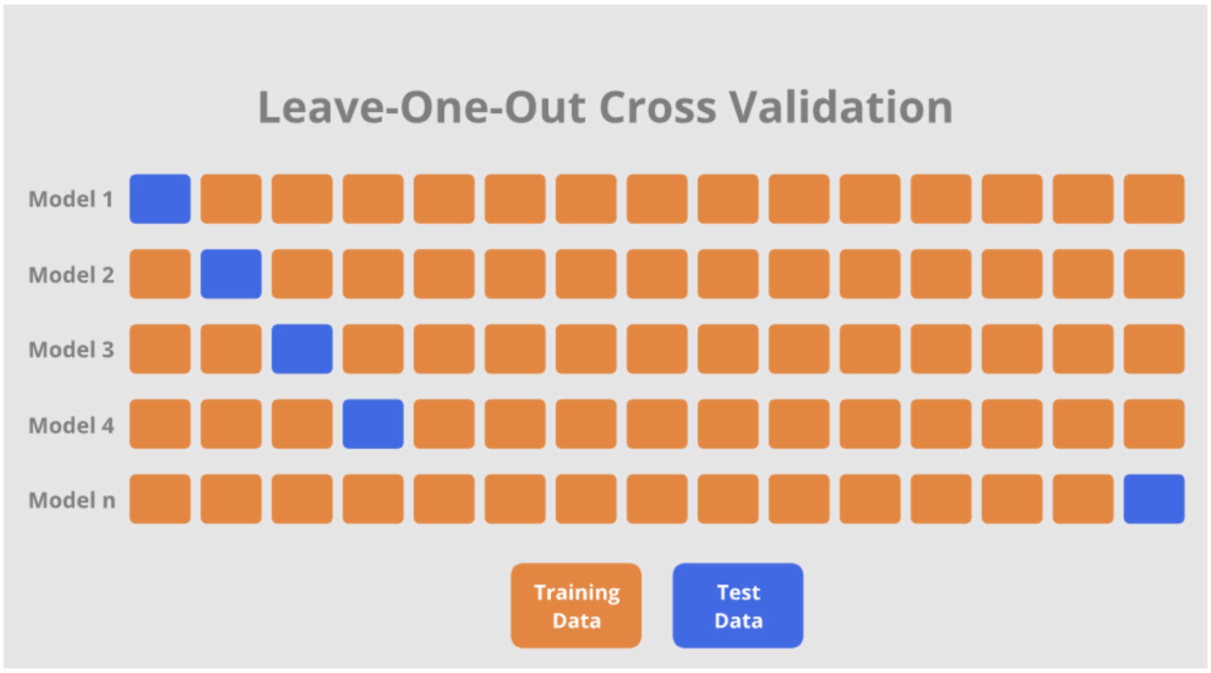
Working:

- **In this method, the dataset is divided into nnn subsets (where nnn is the total number of samples).**

- For each iteration, one data point is used as the test set, and the remaining n−1n-1n−1 data points are used as the training set.
- This process is repeated nnn times, ensuring that each data point is used exactly once as a test sample.
- The final performance is averaged across all iterations.

Thumb Rules (When to Use):

- Use LOOCV when the dataset is very small, as it uses almost all the data for training in each iteration.
- Ideal for cases where you want an unbiased performance estimate.



Advantages of LOOCV:

1.**Use of Data:** LOOCV uses almost all of the data for training, which can be beneficial in situations where the dataset is small and every data point is valuable.

2.**Less Bias**: Since each iteration of validation is performed on just one data point, LOOCV is less biased than other methods, such as k-fold cross-validation. The validation process is less dependent on the random partitioning of data.

3.**No Randomness**: There's no randomness in the train/test split, so the evaluation is stable, without variation in the results due to different random splits.

Disadvantages of LOOCV:

1.Computational Expense: LOOCV requires fitting the model N times, which can be computationally expensive and time-consuming for large datasets.

2.High Variance: LOOCV can lead to higher variance in the model performance since the training sets in all iterations are very similar to each other.

3.Inappropriate Performance Metric: Performance metrics like R^2 are not appropriate to be used with LOOCV as they are not defined when the validation set only has one sample.

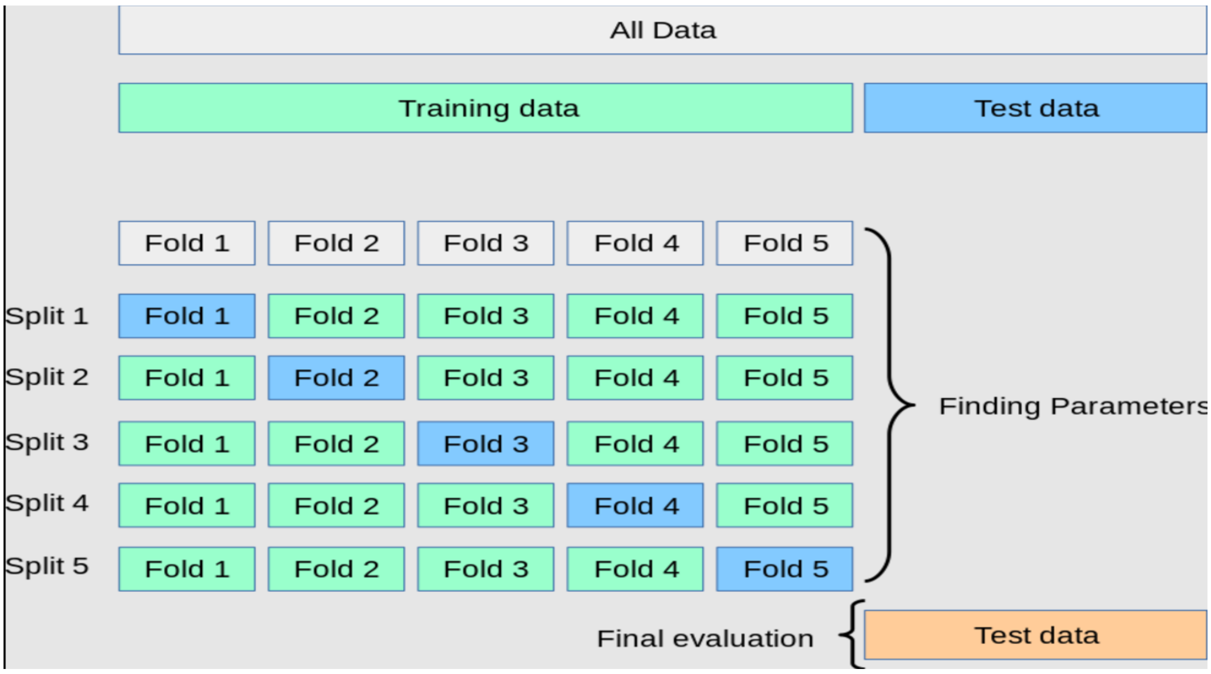
2. K-Fold Cross-Validation :

Working:

- The dataset is split into k equally sized subsets (folds).
- For each iteration, one fold is used as the test set, and the remaining k−1 folds are used as the training set.
- This process is repeated k times, with each fold being used exactly once as the test set.
- The final performance is calculated as the average of the performances from all k iterations.

Thumb Rules (When to Use):

- Use k-fold cross-validation for medium to large datasets where computational cost is manageable.
- Commonly used when LOOCV is too computationally expensive.



Advantages of K-Fold:

1.**Reduction of Variance**: By averaging over k different partitions, the variance of the performance estimate is reduced. This is beneficial because it means that the performance estimate is less sensitive to the particular random partitioning of the data.

2.**Computationally Inexpensive**: Take less time and space in comparison to LOOCV

Disadvantages of K-Fold:

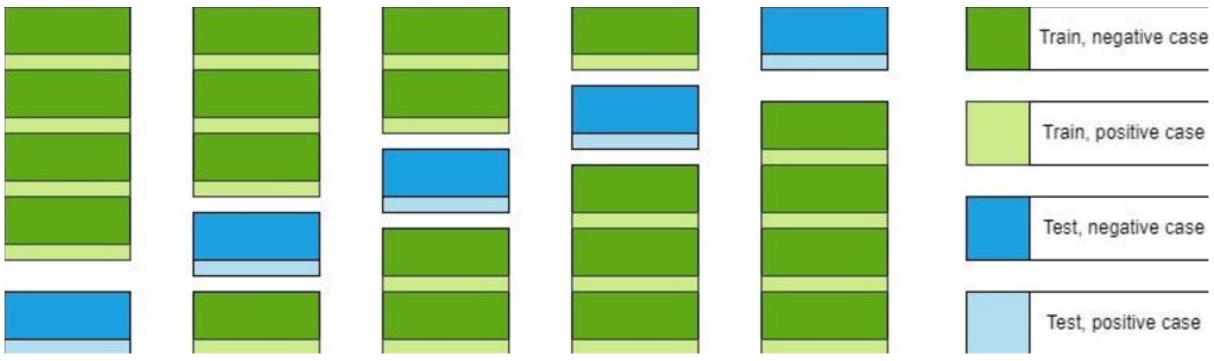
1.**Potential for High Bias:** If k is too small, there could be high bias if the test set is not representative of the overall population.

2.**May not work well with Imbalanced Classes**: If the data has imbalanced classes, there's a risk that in the partitioning, some of the folds might not contain any samples of the minority class, which can lead to misleading performance metrics.

3.Stratified K-Fold Cross-Validation :

Working:

1. The dataset is divided into k equal-sized subsets (folds).
2. The splitting ensures that each fold preserves the proportion of samples for each class.
3. For each iteration, one fold is used as the test set, and the remaining k−1 folds are used as the training set.
4. The process is repeated k times, and the performance is averaged across all iterations.



MODEL EVALUATION

Advantages:

1. Handles Imbalanced Data: Ensures that all folds have a representative distribution of each class, making it suitable for imbalanced datasets.
2. Reduces Bias: Prevents the risk of overestimating or underestimating performance due to class imbalance in the train-test split.
3. Improves Consistency: Produces more reliable and stable performance estimates compared to simple K-Fold Cross-Validation when working with imbalanced datasets.

Disadvantages:

1. Increased Complexity: The splitting process is more complex than standard K-Fold Cross-Validation.
2. Computational Cost: Similar to K-Fold Cross-Validation, it requires training the model k times, which can be computationally expensive for large datasets or complex models.

Thumb Rules (When to Use):

- Use Stratified K-Fold Cross-Validation when working with imbalanced datasets to ensure fair representation of all classes in each fold.
- Ideal for classification problems where maintaining class distribution is critical.

Follow MY Notebook Link For Code.......

MODEL EVALUATION