Cross-Validation

Cross Validation

The problem :

The Hold out approach (Train - Test -split)

- When we use train test split, the data gets divided into some proportion (generally 80 for train, 20 for test). But train test split randomly selects data point and where it should go.

Therefore, everytime when we run the model, the accuracy score would be different.

Problem with Hold out approach :

Variability: The performance of the model can be very sensitive to how the data is divided into training and testing sets. If the split is unfortunate, the training set may not be representative of the overall distribution of data, or the test set might contain unusually easy or difficult examples. This leads to high variance in the estimation of the model's performance.

1. **Data inefficiency:** The holdout method only uses a portion of the data for training and a different portion for testing. This means that the model doesn't get to learn from all available data, which can be particularly problematic if the dataset is small.
2. **Bias in performance estimation:** If some classes or patterns are over- or under-represented in the training set or the test set due to the random split, it can lead to a biased performance estimation.
3. **Less reliable for hyperparameter tuning:** If the holdout method is used for hyperparameter tuning, there's a risk of overfitting to the test set because information might leak from the test set into the model. This means that the model's performance on the test set might be overly optimistic and not representative of its performance on unseen data

Why hold out approach is still use then ?

1. Simplicity : The holdout method is straightforward and easy to understand. You simply divide your data into two sets: a training set and a test set. This simplicity makes it appealing, especially for initial exploratory analysis or simple projects.

2. Computational Efficiency: The holdout method is computationally less intensive than methods like k-fold cross-validation. In k-fold cross-validation, you need to train and test your model k times, which can be computationally expensive, especially for large datasets or complex models. With the holdout method, you only train the model once.

3. Large Datasets: For very large datasets, even a small proportion of the data may be sufficient to form a representative test set. In these cases, the holdout method can work quite well.

Cross Validation

The idea of cross-validation is to divide the data into several subsets or "folds". The model is then trained on some of these subsets and tested, on the remaining ones.

The process is repeated multiple times, with different subsets used for training and validation each time. The results from each round are usually averaged to estimate the model's overall performance.

Leave one out Cross Validation (LOOCV)

Each data point in the dataset is used once as a test sample while the remaining data points are used for training. This process repeats for every data point, resulting in as many iterations as there are samples.

Advantages and Disadvantages of Leave-One-Out Cross-Validation (LOOCV)

Advantages

- **Use of Data:**
  LOOCV uses almost all of the data for training, which is beneficial when the dataset is small and every data point is valuable.
- **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.
- **No Randomness:**
  There's no randomness in the train/test split, so the evaluation is stable and does not vary due to different random splits.

Disadvantages

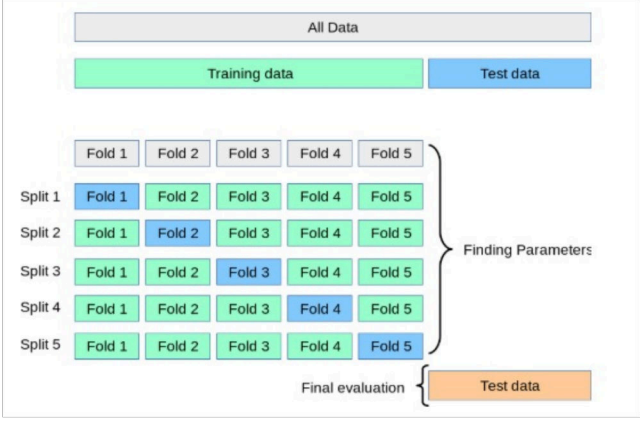
- **Computational Expense:**
  LOOCV requires fitting the model N times, which can be computationally expensive and time-consuming for large datasets.
- **High Variance:**
  LOOCV can lead to higher variance in the model performance estimate since the training sets in all iterations are very similar to each other.
- **Inappropriate Performance Metric:**
  Performance metrics like R2R^2R2 are not appropriate to be used with LOOCV as they are not defined when the validation set only has one sample.
- **Not Ideal for Imbalanced Data:**
  In classification problems, if you have imbalanced classes, LOOCV may not provide a reliable estimate of model performance because the single validation sample in each iteration may not be representative of the overall class distribution.

When to Use LOOCV

- **Small Datasets:**
  LOOCV is most beneficial when you have a limited amount of data. With small datasets, you want to use as much data as possible for training, which is exactly what LOOCV offers by using all but one data point for training.
- **Balanced Datasets:**
  LOOCV might not perform well on imbalanced datasets, especially in classification problems, because the training set might end up missing some classes. Thus, it's more appropriate to use LOOCV when you have a balanced dataset.
- **Need for Less Biased Performance Estimate:**
  Since LOOCV uses nearly all the data for training, it gives a less biased estimate of model performance compared to other methods like k-fold cross-validation.

K-Fold Cross Validation

K-Fold Cross Validation is a method used to assess the performance and generalizability of machine learning models. It involves splitting the dataset into K equally sized folds (subsets), training the model on K-1 folds, and validating it on the remaining fold. This process is repeated K times, with each fold used exactly once as the validation data.



Advantages of K-Fold Cross Validation:

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 Cross Validation:

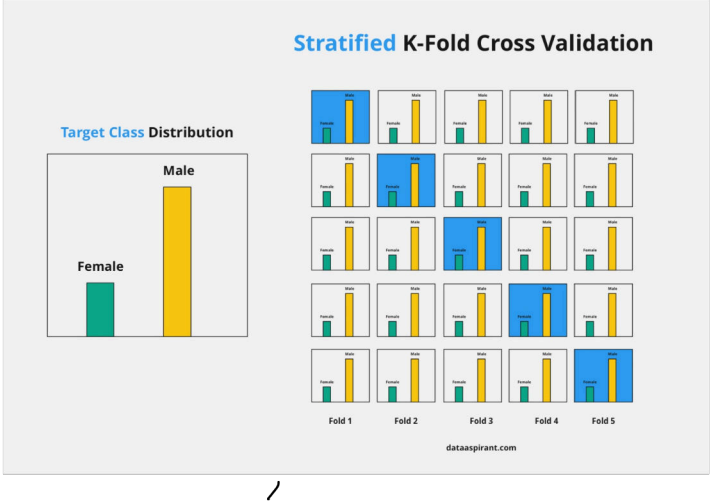
1. Potential for High Bias: If k is too small, there could be high bias if the test set is not representative of the overall population.
2. May not work well with Imbalanced Classes: If the data has imbalanced classes, there's a risk that in the partitioning, some of the folds might not contain any samples of the minority class, which can lead to misleading performance metrics.

When to use:

1. When you have a sufficiently large dataset: K-Fold Cross Validation requires the model to be trained K times, so it can be computationally intensive. However, if your dataset is large enough, this increased computational cost can be justified by the more reliable estimate of model performance.
2. When your data is evenly distributed: K-Fold Cross Validation works best when the data is evenly distributed. If your dataset is imbalanced (i.e., one class has significantly more samples than another), it might be better to use a technique like Stratified K-Fold Cross Validation, which aims to ensure each fold is a good representative of the whole dataset.

Stratified K-Fold Cross Validation

Stratified K-Fold Cross Validation is a variation of K-Fold Cross validation, that ensures each fold is representative of the overall class distribution. This method is particularly useful for imbalanced datasets, where the number of samples in each class is not evenly distributed.



Maintaining the even class ratio across all folds. Therefore, each part will have same class distribution.

Advantages of Stratified K-Fold Cross Validation:

1. Maintains Class Distribution: Stratified K-Fold Cross Validation ensures that each fold contains approximately the same percentage of samples of each target class as the complete dataset, preserving the true class distribution and reducing the risk of biased outcomes.
2. More Robust and Reliable Performance Estimates: By maintaining class proportions in each fold, it provides a more accurate and robust estimate of model performance, especially with imbalanced datasets.

Disadvantages of Stratified K-Fold Cross Validation:

1. Increased Complexity: The stratification process adds complexity to the data splitting procedure compared to standard K-Fold Cross Validation.
2. May Not Be Necessary for Balanced Data: If the dataset is already balanced, the benefits of stratification are minimal and a standard K-Fold may suffice.

When to use:

1. When Dealing with Imbalanced Classes: Stratified K-Fold Cross Validation is particularly useful when the target variable is imbalanced, as it ensures that each fold is a good representative of the overall class distribution.
2. When Reliable Performance Metrics Are Needed: Use this method when you require a more accurate and fair evaluation of models, especially for classification problems where minority class performance is critical.

Data Leakage

Data Leakage in Machine Learning

Data leakage, in the context of machine learning and data science, refers to a problem where information from outside the training dataset is used to create the model. This additional information can come in various forms, but the common characteristic is that it is information that the model wouldn't have access to when it's used for prediction in a real-world scenario.

This can lead to overly optimistic performance estimates during training and validation, as the model has access to extra information. However, when the model is deployed in a production environment, that additional information is no longer available, and the performance of the model can drop significantly. This discrepancy is typically a result of mistakes in the experiment design.

Ways in which data leakage can occur:

1. **Target Leakage:**
   Target leakage occurs when your predictors include data that will not be available at the time you make predictions.
2. **Multicollinearity with target col**
3. **Duplicated Data**
4. **Preprocessing Leakage → Train test contamination & Improper Cross Validation**
5. **Hyperparameter Tuning**

How to detect?

1. **Review Your Features:**
   Carefully review all the features being used to train your model. Do they include any data that wouldn't be available at the time of prediction, or any data that directly or indirectly reveals the target? Such features are common sources of data leakage.
2. **Unexpectedly High Performance:**
   If your model's performance on the validation or test set is surprisingly good, this could be a sign of data leakage. Most predictive modelling tasks are challenging, and exceptionally high performance could mean that your model has access to information it shouldn't.
3. **Inconsistent Performance Between Training and Unseen Data:**
   If your model performs significantly better on the training and validation data compared to new, unseen data, this might indicate that there's data leakage.
4. **Model Interpretability:**
   Interpretable models, or techniques like feature importance, can help understand what the model is learning. If the model places too much importance on a feature that doesn't seem directly related to the output, it could be a sign of leakage.

How to remove data leakage?

1. **Understand the Data and the Task:**
   Before starting with any kind of data processing or modelling, it's important to understand the problem, the data, and how the data was collected. You should understand what each feature in your data represents, and whether it would be available at the time of prediction.
2. **Careful Feature Selection:**
   Review all the features used in your model. If any feature includes information that wouldn't be available at the time of prediction, or that directly or indirectly gives away the target variable, it should be removed or modified.
3. **Proper Data Splitting:**
   Always split your data into training, validation, and testing sets at an early stage of your pipeline, before doing any pre-processing or feature extraction.
4. **Pre-processing Inside the Cross-Validation Loop:**
   If you're using techniques like cross-validation, make sure to do any pre-processing inside the cross-validation loop. This ensures that the pre-processing is done separately on each fold of the data, which prevents information from the validation set leaking into the training set.

```
# Incorrect way
X_normalized = normalize(X)  # normalize the whole dataset
cross_val_score(model, X_normalized, y, cv=5)  # perform cross-validation
```

```
# Correct way
pipeline = make_pipeline(normalizer, model)
cross_val_score(pipeline, X, y, cv=5)  # pe
```

5. **Avoid Overlapping Data:**
   If the same individuals, or the same time periods, appear in both your training and test sets, this can cause data leakage. It's important to ensure that the training and test sets represent separate, non-overlapping instances.