

Model Evaluation Techniques in Machine Learning

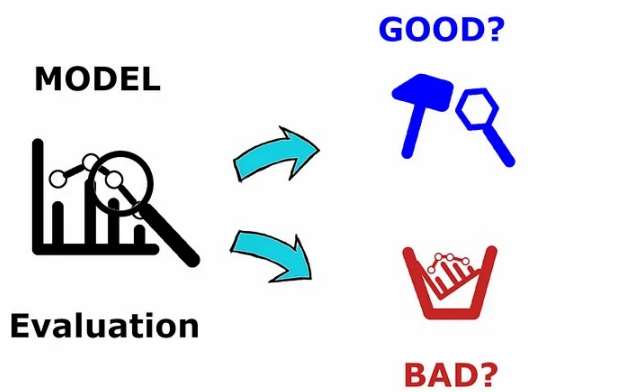


Sachinsoni · Follow

11 min read · Jun 14, 2023



8



Model evaluation is a fundamental step in machine learning and predictive modeling. It enables us to assess the performance, reliability, and generalization capabilities of models, leading to informed decision-making and improved business outcomes. There are several model evaluation techniques which are as follow:

A. Hold-out Approach:

Sign up to discover human stories that deepen your understanding of the world.

Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

1. The original dataset is randomly divided into a training set and a testing set. The common split ratio is 70–30 or 80–20, but it can vary depending on the size of the dataset and the specific problem.
2. The training set is used to train the model. The model learns patterns and relationships in the data based on the input features and corresponding target variables.
3. Once the model is trained, the testing set is used to evaluate its performance. The model's predictions are compared against the actual target variables in the testing set to calculate performance metrics such as accuracy, precision, recall, or others, depending on the problem type.
4. The performance metrics obtained from the testing set provide an estimate of how well the model is likely to perform on unseen data.

and the code for train-test- split is:

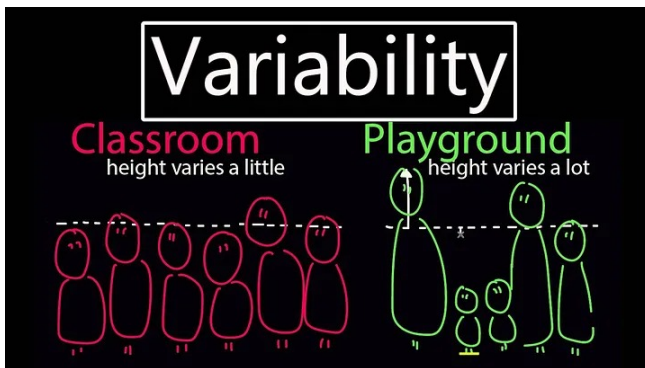
```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

Problems with Hold-out Approach:

While the hold-out approach is a straightforward and commonly used technique for model evaluation, it does have some potential problems and limitations:

1. **Variability** :- The performance of the model can be very sensitive to how the data is divided into training and testing sets. A different split may result in different evaluation results, leading to less reliable performance estimates.



Let's assume we have a dataset of 1000 samples and we want to split it into a training set and a testing set using an 80–20 ratio. We will evaluate the performance of a classification model on the testing set.

If we set the **random state** to a **specific value**, such as 42, the data will be split in a consistent manner. We train the model on the training set and evaluate its performance on the testing set. **Let's assume we obtain an accuracy of 85%.**

Now, if we **change the random state to a different value**, let's say 100, and repeat the same process, we might get a **different accuracy**, such as 82%. This discrepancy arises due to the different samples that are randomly selected for the training and testing sets with a different random state.

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=2)

# So,if we change random state value model accuracy will be change every time.
# This is called variability and creates a confusion how to select best model fo
```

By repeating this process with various random state values, we will observe fluctuations in the model's performance. For instance, **using random state 42 might yield an accuracy of 85%, while random state 100 could result in 82%, and random state 123 might give us 84%.** These variations highlight the impact of randomization on the hold-out approach and demonstrate the variability in the evaluation results.

2. **Data inefficiency** :- The holdout method only uses a portion of 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.

3. **Bias in performance estimation** :- Let's consider a real-life example to understand how bias can affect performance estimation.

Suppose we are building a spam email classifier, and we have collected a dataset of 10,000 emails. Out of these, 9,000 emails are legitimate and 1,000 emails are spam. The class distribution is imbalanced, with a much higher

number of legitimate emails compared to spam emails.

Now, let's say we use the hold-out approach to evaluate our spam classifier by randomly splitting the data into a training set (80% of the data) and a testing set (20% of the data). We train the model on the training set and evaluate its performance on the testing set using a performance metric such as accuracy.

In this scenario, there is a possibility of biased performance estimation due to the imbalanced class distribution. The model might achieve high accuracy simply by correctly classifying the majority class (legitimate emails) while performing poorly on the minority class (spam emails).

Let's assume that the model achieves an accuracy of 95% on the testing set. At first glance, this may appear to indicate excellent performance. However, upon closer inspection, we discover that the model incorrectly classifies 50% of the spam emails while correctly classifying 98% of the legitimate emails.

This bias in performance estimation occurs because the evaluation metric (accuracy) does not account for the imbalanced class distribution. The majority class (legitimate emails) heavily influences the accuracy, leading to a misleadingly high value. In reality, the model's performance on the minority class (spam emails) is significantly worse than the overall accuracy suggests.

So, 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.

4. Less reliable for hyperparameter tuning :- If the holdout method is used for hyperparameter tuning, there's 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 of unseen data.

When these are the problems in hold-out approach then why we use this approach?

- 1. Simplicity :-** straightforward and easy to understand.
- 2. Computational Efficiency :-** Since with this method you train your model only once hence it is computationally less expensive as compare to other techniques which you study in this article because they train the model multiple times.
- 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 and if you change the random state value multiple times, model accuracy will change very very small and then this is ok for model evaluation.

B. Cross Validation : (based on Resampling Technique)

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. This 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.

There are several methods of cross-validation commonly used in machine learning. Here are some of the most widely used techniques:

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

(n rows-> n models ->(n-1) for training -> 1 for testing)

Let's understand Leave-One-Out Cross-Validation (LOOCV) with a simpler explanation and a real-life example.

Imagine you are a student studying for an exam. You have a collection of practice questions that you want to use to evaluate your knowledge and predict how well you will perform on the actual exam. However, you want to make sure your prediction is as accurate as possible.

In LOOCV, you simulate a situation where you test your knowledge by leaving out one practice question at a time and using the remaining questions to assess your understanding.

Here's how it works:

1. Imagine you have 20 practice questions numbered from 1 to 20.
2. For the first round of LOOCV, you decide to leave out question 1 and use questions 2 to 20 as your training set. You study those questions and try to predict the answer to question 1 based on your understanding of the other questions.
3. After making your prediction for question 1, you compare it to the actual answer. This gives you an idea of how well you would have performed on that particular question if it had been part of the training set.
4. You repeat this process for each question, leaving out one question at a time and predicting its answer based on the rest of the questions. Each time, you assess your prediction against the actual answer.
5. In the end, you will have gone through all 20 questions, leaving out each one and evaluating your prediction. You can calculate the overall accuracy(average accuracy) or any other performance metric by comparing all your predictions to the actual answers.

In this example, LOOCV simulates a situation where you assess your knowledge by leaving out one question at a time and evaluating your performance. By going through this process for all the questions, you can get a more accurate estimate of how well you understand the material and how you might perform on the actual exam.

For implementing LOOCV technique see my Jupyter notebook on my github. [Click here!](#)

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 in each iteration of validation is performed on just one data point, LOOCV is less biased than other methods, such as k-fold validation.
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 and Time Consuming
2. Not ideal for imbalanced dataset

When to use LOOCV:

1. Small datasets
2. Balanced datasets
3. **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.

2. K-Fold Cross-Validation:

- In k-fold cross-validation, the dataset is divided into k equal-sized folds or subsets.
- The model is trained and evaluated k times, each time using a different fold as the validation set while the remaining folds are used for training.
- The performance metrics obtained from each iteration are averaged to obtain an overall performance estimate.

Let's understand k-fold cross-validation with an example:

Suppose you have a dataset of 100 images, and you want to build an image classifier. To evaluate the performance of your classifier, you decide to use 5-fold cross-validation.

Here's how it works:

1. Dataset Preparation:

- You divide your dataset of 100 images into 5 equal-sized folds, each containing 20 images.
- Each fold represents a subset of the data that will be used for training and validation.

2. Iteration 1:

- In the first iteration, you use Fold 1 as the validation set and the remaining Folds 2 to 5 as the training set.
- You train your image classifier on Folds 2 to 5, using 80 images, and evaluate its performance on Fold 1, which has 20 images.
- You record the performance metrics obtained, such as accuracy, precision, recall, or any other relevant metric.

3. Iteration 2:

- In the second iteration, you use Fold 2 as the validation set and Folds 1, 3, 4, and 5 as the training set.

- You train your image classifier on Folds 1, 3, 4, and 5, using 80 images, and evaluate its performance on Fold 2, which has 20 images.
- Again, you record the performance metrics obtained.

4. Iterations 3, 4, and 5:

- You repeat the same process for Folds 3, 4, and 5, using them as the validation set while the remaining folds serve as the training set.
- Each time, you train the model on the training set and evaluate its performance on the respective validation fold.
- Performance metrics are recorded for each iteration.

5. Performance Estimation:

- Once all 5 iterations are completed, you have obtained performance metrics for each fold.
- To obtain an overall performance estimate, you average the performance metrics obtained from the 5 iterations.
- This average represents the performance of your image classifier across all the folds and provides a reliable estimate of its effectiveness.

For implementing K-fold cross validation technique see my Jupyter notebook on my GitHub. [Click here!](#)

Advantages of K-fold cross validation:-

1. **Reduction of Variance** : LOOCV has a higher variance because the models are highly correlated due to the minimal differences in training data. In k-fold cross-validation, the larger differences between the validation sets reduce the correlation among the models, resulting in lower variance.
2. Computationally inexpensive as compared to LOOCV.

Disadvantages of K-fold cross validation:-

1. **May not work well with imbalanced Classes** : If the dataset 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.
2. **Potential for High Bias**

When to use K-fold cross validation:

1. When you have a sufficiently large dataset
2. When your data is evenly distributed

3. Stratified K-fold cross validation :

Stratified k-fold cross-validation is a variation of k-fold cross-validation that addresses the issue of imbalanced class distributions in the dataset. It is commonly used when the target variable or the classes of interest are unevenly distributed.

In standard k-fold cross-validation, the dataset is randomly divided into k equal-sized folds. However, this random partitioning might lead to some folds having significantly imbalanced class distributions, especially when the original dataset has a class imbalance problem.

Stratified k-fold cross-validation aims to preserve the class distribution across different folds. It ensures that each fold maintains the same class distribution as the original dataset, thereby providing a more reliable estimate of model performance, particularly for imbalanced datasets.

The process of stratified k-fold cross-validation involves the following steps:

1. First, the dataset is divided into k folds, typically using random sampling.
2. Next, for each fold, the class distribution of the target variable is analyzed.
3. The folds are then constructed in such a way that each fold contains approximately the same proportion of each class as the original dataset. This ensures that each fold is representative of the overall class distribution.
4. The model is trained and evaluated using the k iterations, where in each iteration, one fold is used as the validation set and the remaining k-1 folds are used as the training set.
5. The performance measures, such as accuracy or F1 score, are calculated for each fold, and the results are averaged to obtain the overall performance estimate of the model.

For implementing Stratified K-fold cross validation technique see my Jupyter notebook on my GitHub. [Click here!](#)

I hope this will improve your knowledge regarding Model Evaluation techniques in machine Learning. Thank you for reading this article!



Written by Sachinsoni

74 Followers



More from Sachinsoni



Mastering t-SNE(t-distributed stochastic neighbor embedding)

A better dimensionality reduction technique as compared to PCA (Principal Component...

10 min read · Feb 11, 2024



K Nearest Neighbours— Introduction to Machine Learning...

Used to solve classification type problems

7 min read · Jun 11, 2023

👤 Sachinsoni

Unlocking the ideas behind of SVM(Support Vector Machine)

Imagine teaching a computer to be like a clever detective, finding the best ways to...

17 min read · Aug 22, 2023

👤 98 💬

🔖

👤 Sachinsoni

Naïve Bayes Machine Learning Algorithm From Basic to Advanced

The naïve Bayes algorithm is a family of probabilistic classification algorithms used...

16 min read · Jun 30, 2023

👤 29 💬

🔖

See all from Sachinsoni

Recommended from Medium

👤 Data Notes

Understanding Confusion Matrix with Python

What is a Confusion Matrix?

6 min read · Feb 26, 2024

👤 2 💬

🔖

👤 Sachinsoni

Mastering t-SNE(t-distributed stochastic neighbor embedding)

A better dimensionality reduction technique as compared to PCA (Principal Component...

10 min read · Feb 11, 2024

👤 53 💬

🔖

Lists

Staff Picks

646 stories · 986 saves

Stories to Help You Level-Up at Work

19 stories · 613 saves

Self-Improvement 101

20 stories · 1857 saves

Productivity 101

20 stories · 1701 saves

👤 Gen. David L.

10 Python Methods for Handling Imbalanced Data

Imbalanced data is a common challenge in machine learning, where the quantity of one...

3 min read · Jan 6, 2024

👤 4 💬

🔖

👤 Shawkygamal


Precision vs. Recall

Imagine someone told you that he had made a model with 99.9% accuracy. Would you...

5 min read · Apr 18, 2024

👤 5 💬

🔖


 Gupta Anshul

Hypothesis Testing: Chi-Square Test

The chi-square distribution is a continuous probability distribution. It's denoted as χ^2 ...

3 min read · Mar 29, 2024



 Ayesha sidhikha

Unveiling Outliers: Exploring Z-Score and IQR Methods for...

In the realm of data analysis, the identification of outliers holds significant importance as...

11 min read · Jan 15, 2024



See more recommendations