27-day27-cross-validation

October 24, 2023

Day27: Cross-Validation By: Loga Aswin

What is Cross-Validation?

> Cross-validation is a technique for evaluating a machine learning model and testing its performance. It is a way to test how good a machine learning model is. It's like giving a test to see if the model can solve real-world problems. CV is commonly used in applied ML tasks.

The Core Algorithm of Cross-Validation

Cross-validation methods share a common algorithmic structure: > **Data Split:** The dataset is divided into two distinct subsets—one for training and the other for testing.

> **Model Training:** The machine learning model is trained on the training dataset.
>
> **Model Validation:** The trained model is then validated using the test dataset.
>
> **Repetitions:** Steps 1 to 3 are repeated a number of times, which depends on the specific cross-validation technique employed.

There are plenty of CV techniques. Some of them are commonly used:

1. Hold-out cross-validation: We simply split the data into two parts.

How It Works:

Data Split: You divide your dataset into two parts: the training set and the test set. Typically, 80% of the data goes into the training set, and 20% goes into the test set, but you can adjust these percentages as needed.

Model Training: You teach your model on the training set.

Model Testing: You test your model on the test set.

Result: You save the outcome of the test. That's it!

[4]:
```
import numpy as np
from sklearn.model_selection import train_test_split

# sample data
data = np.arange(20).reshape((10, 2))
labels = np.array([0, 0, 1, 1, 1, 0, 0, 1, 1, 0])

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.
↪3, random_state=42)

# Now X_train and y_train contain 70% of the data for training, and X_test and␣
↪y_test contain 30% for testing.
```

[10]:
```
#For Uploading image from files
from google.colab import files
from IPython.display import Image

# Upload an image
uploaded = files.upload()

# Display the uploaded image
for filename in uploaded.keys():
    display(Image(filename))
```

```
<IPython.core.display.HTML object>
```

```
Saving Screenshot 2023-10-24 232937.png to Screenshot 2023-10-24 232937 (1).png
```

2. k-Fold cross-validation: We divide the data into 'k' parts and test the model with each part.

The algorithm of the k-Fold technique:

Divide into Folds: Split your data into 'k' equal parts, like dividing a pie into slices.

Test One Slice: Take one slice (fold) as a test set and the others as training sets.

Train Model: Train a new model using the training slices.

Test Model: Test the model on the slice you set aside.

Repeat: Do this 'k' times, each time with a different slice as the test set.

Average Results: Average the results from all 'k' tests to see how well your model works overall.

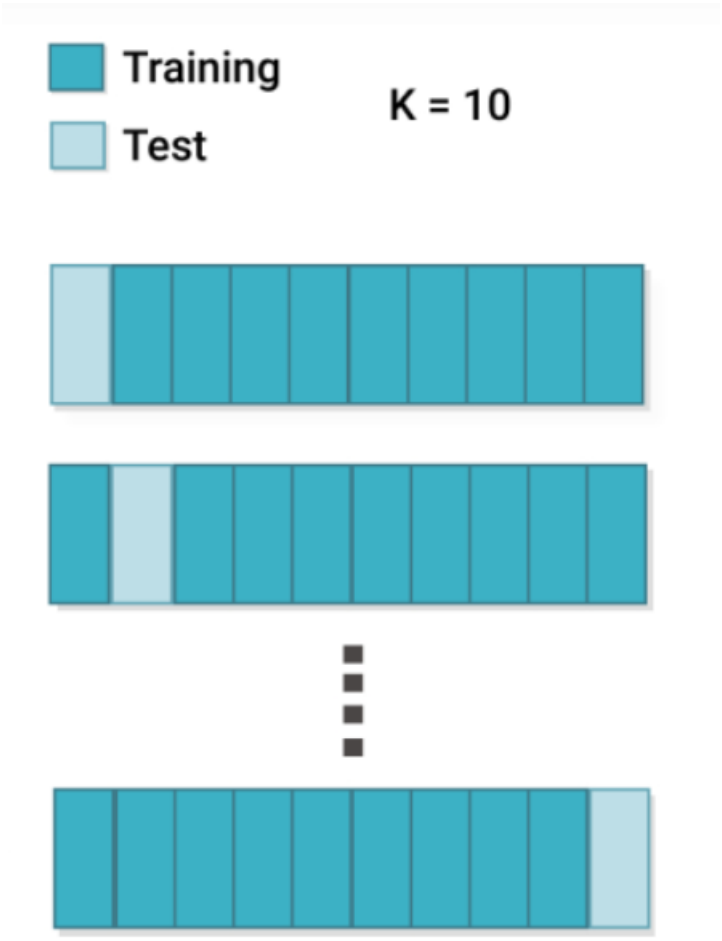
In general, it is always better to use k-Fold technique instead of hold-out.

[9]:
```
uploaded = files.upload()

# Display the uploaded image
for filename in uploaded.keys():
    display(Image(filename))
```

```
<IPython.core.display.HTML object>
```

```
Saving Screenshot 2023-10-24 233447.png to Screenshot 2023-10-24 233447 (2).png
```



3. Leave-one-out cross-validation: In this method, each data point is used as a testing instance while the rest are used for training.

4. Leave-P-Out: Similar to Leave-One-Out, but we use 'p' pieces at a time.

5. Stratified K-Folds: Like K-Folds, but it keeps the same kind of data in each part.

6. Repeated K-Folds: We repeat K-Folds many times with different data splits.

7. Nested K-Folds: A combination of K-Folds used for different kinds of tests.

8. Time Series Cross-Validation: Specifically designed for time series data to account for temporal dependencies.

Cross-Validation in Machine Learning:

scikit-learn (sklearn): This popular Python library provides various tools for machine learning, including functions for easy cross-validation techniques such as k-Fold and stratified sampling.

CatBoost: CatBoost is a gradient boosting library that supports cross-validation methods to evaluate its models. It has built-in cross-validation functionality.

Cross-Validation in Deep Learning:

Keras: Keras is an open-source neural network library that can be used with popular deep learning frameworks like TensorFlow and Theano. It doesn't have specific built-in cross-validation functions, but you can use scikit-learn or other tools for that purpose.

PyTorch: PyTorch is a deep learning framework with a rich ecosystem of libraries. You can use PyTorch in combination with scikit-learn for cross-validation or implement custom cross-validation techniques.

MxNet (MXNet): MXNet is another deep learning framework that can be integrated with cross-validation libraries for evaluating deep learning models.