The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

In this tutorial, you will discover how to evaluate machine learning models using the train-test split.

After completing this tutorial, you will know:

- The train-test split procedure is appropriate when you have a very large dataset, a costly model to train, or require a good estimate of model performance quickly.
- How to use the scikit-learn machine learning library to perform the train-test split procedure.
- How to evaluate machine learning algorithms for classification and regression using the train-test split.

Train-Test Split Evaluation

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

When to Use the Train-Test Split

The idea of “sufficiently large” is specific to each predictive modeling problem. It means that there is enough data to split the dataset into train and test datasets and each of the train and test datasets are suitable representations of the problem domain. This requires that the original dataset is also a suitable representation of the problem domain.

A suitable representation of the problem domain means that there are enough records to cover all common cases and most uncommon cases in the domain. This might mean combinations of input variables observed in practice. It might require thousands, hundreds of thousands, or millions of examples.

Conversely, the train-test procedure is not appropriate when the dataset available is small. The reason is that when the dataset is split into train and test sets, there will not be enough data in the training dataset for the model to learn an effective mapping of inputs to outputs. There will also not be enough data in the test set to effectively evaluate the model performance. The estimated performance could be overly optimistic (good) or overly pessimistic (bad).

If you have insufficient data, then a suitable alternate model evaluation procedure would be the k-fold cross-validation procedure.

How to Configure the Train-Test Split

The procedure has one main configuration parameter, which is the size of the train and test sets. This is most commonly expressed as a percentage between 0 and 1 for either the train or test datasets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

There is no optimal split percentage.

You must choose a split percentage that meets your project's objectives with considerations that include:

- Computational cost in training the model.
- Computational cost in evaluating the model.
- Training set representativeness.
- Test set representativeness.

Nevertheless, common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

Now that we are familiar with the train-test split model evaluation procedure, let's look at how we can use this procedure in Python.

function.

The function takes a loaded dataset as input and returns the dataset split into two subsets.

```
...
# split into train test sets
train, test = train_test_split(dataset, ...)
```

Ideally, you can split your original dataset into input (*X*) and output (*y*) columns, then call the function passing both arrays and have them split appropriately into train and test subsets.

```
...
# split into train test sets
X_train, X_test, y_train, y_test = train_test_s
```

The size of the split can be specified via the "*test_size*" argument that takes a number of rows (integer) or a percentage (float) of the size of the dataset between 0 and 1.

The latter is the most common, with values used such as 0.33 where 33 percent of the dataset will be allocated to the test set and 67 percent will be allocated to the training set.

```
...
# split into train test sets
X_train, X_test, y_train, y_test = train_test_s
```

We can demonstrate this using a synthetic classification dataset with 1,000 examples.

The complete example is listed below.

```
# split a dataset into train and test sets
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_
# create dataset
X, y = make_blobs(n_samples=1000)
# split into train test sets
X_train, X_test, y_train, y_test = train_test_s
print(X_train.shape, X_test.shape, y_train.shap
```

Running the example splits the dataset into train and test sets, then prints the size of the new dataset.

We can see that 670 examples (67 percent) were allocated to the training set and 330 examples (33 percent) were allocated to the test set, as we specified.

```
(670, 2) (330, 2) (670,) (330,)
```

Alternatively, the dataset can be split by specifying the "*train_size*" argument that can be either a number of rows (integer) or a percentage of the original dataset between 0 and 1, such as 0.67 for 67 percent.

```
...
# split into train test sets
X_train, X_test, y_train, y_test = train_test_s
```