Different types of data splitting methods - Mejbah Ahammad

In order to prevent overfitting and guarantee that our model can generalize to new data, data splitting is essential in machine learning. Let's examine a few typical data splitting techniques:

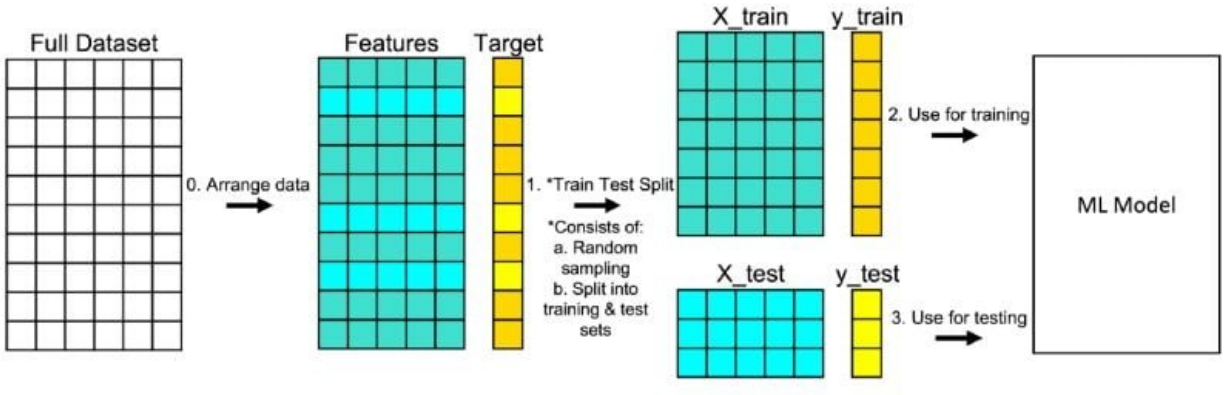

Image Credit: Train test split procedure. | Image: Michael Galarnyk | Built In

1. **Train/Test Split** This is the simplest method. We split our data into a training set and a testing set.

```
from sklearn.model_selection import train_test_split

X, y = [...]  # Your data and labels
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
```

2. **K-Fold Cross Validation** This method involves splitting the data into 'k' subsets. The model is trained on k-1 of these folds and tested on the remaining one. This process is repeated k times, each time with a different fold as the test set.

```
from sklearn.model_selection import KFold

kf = KFold(n_splits=5, shuffle=True, random_state=42)
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # Train and test your model here
```

3. **Stratified K-Fold Cross Validation** Like K-Fold, but it ensures that each fold maintains the same distribution of classes as the entire dataset.

```
from sklearn.model_selection import StratifiedKFold

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
for train_index, test_index in skf.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # Train and test your model here
```

4. **Time Series Split** Useful for time series data. In each split, the test set consists of the next 'n' points in the data. This avoids "looking into the future" during training.

```
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=5)
for train_index, test_index in tscv.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # Train and test your model here
```

5. **Leave One Out Cross Validation (LOOCV)** This involves training on all data points except one and testing on that single left out point. This is repeated for all data points. It's computationally intensive but can be useful for small datasets.

```
from sklearn.model_selection import LeaveOneOut

loo = LeaveOneOut()
for train_index, test_index in loo.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # Train and test your model here
```

Always be sure that no information from the test set leaks into the training set with your data. This is crucial in instances like time series forecasting, where utilizing past data to predict the future may provide erroneous findings.

Credit: BytesOfIntelligence - > https://github.com/BytesOfIntelligences