1-train_test_split

October 20, 2024

1 Train Test Split

The train-test split is a critical concept in machine learning for evaluating the performance of a model. It involves dividing the dataset into two distinct parts: the training set and the test set. This method ensures that we can assess how well the model generalizes to unseen data, which is the key to making accurate predictions in real-world applications.

Purpose of Train-Test Split: The main goal is to evaluate the model's ability to generalize to new, unseen data by testing it on a separate test set that the model has never encountered during training. If a model performs well on both the training data and the test data, it's likely to perform well on new, real-world data.

Key Terminology:

1. Training Set: The subset of the dataset used to train the model. The model learns patterns, relationships, and trends from this data.
2. Test Set: The subset used to evaluate the model after training. It provides an unbiased estimate of the model's performance on unseen data.

Why Train-Test Split is Important:

- Avoid Overfitting: Without a train-test split, the model might memorize the training data (overfitting) rather than learning generalizable patterns. This would result in poor performance on new data.
- Unbiased Evaluation: It provides an unbiased measure of how well the model generalizes beyond the training data.
- Generalization Performance: It allows us to assess the model's generalization error, i.e., how it will perform on future data.

Train-Test Split Ratio:

- 80:20 Split: Commonly used when the dataset is sufficiently large. This balance ensures that there is enough data for the model to learn (80%) while still keeping a good portion (20%) for evaluating the model's performance.
- 70:30 Split: Used when you want more data for testing, but still have enough data for training. This might be more appropriate when the dataset is relatively smaller, as having more test data can give a clearer picture of generalization.

- 90:10 Split: Used when the dataset is very large. A small test set is enough to provide a good estimate of model performance since the model has a huge amount of data to train on.

[23]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

[24]:
```
data = pd.read_csv('500hits.csv', encoding='latin-1')
data.head()
```

[24]:
```
          PLAYER  YRS     G     AB     R     H   2B   3B   HR   RBI    BB  \
0        Ty Cobb   24  3035  11434  2246  4189  724  295  117   726  1249
1    Stan Musial   22  3026  10972  1949  3630  725  177  475  1951  1599
2   Tris Speaker   22  2789  10195  1882  3514  792  222  117   724  1381
3    Derek Jeter   20  2747  11195  1923  3465  544   66  260  1311  1082
4  Honus Wagner   21  2792  10430  1736  3430  640  252  101     0   963

     SO   SB   CS     BA  HOF
0   357  892  178  0.366    1
1   696   78   31  0.331    1
2   220  432  129  0.345    1
3  1840  358   97  0.310    1
4   327  722   15  0.329    1
```

[25]:
```
X = data.drop(columns=['PLAYER', 'HOF'])
y = data['HOF']

X.shape
y.shape
```

[25]:
```
(465,)
```

[26]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=11,␣
 ↪test_size=0.2)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(372, 14)
(93, 14)
(372,)
(93,)
```

[27]:
```
print(X_train.describe().round(3))
```

```
           YRS        G        AB        R        H       2B       3B  \
count  372.000  372.000   372.000  372.000  372.000  372.000  372.000
mean    17.011 2046.522  7526.078 1154.126 2177.995  382.505   78.094
std      2.662  351.233  1302.406  291.308  426.615   97.173   48.798
```

```
min     11.000  1331.000   4981.000   651.000  1660.000  177.000    3.000
25%     15.000  1797.500   6507.500   936.000  1838.000  312.000   41.000
50%     17.000  1992.000   7237.000  1099.000  2080.500  367.000   67.000
75%     19.000  2245.500   8198.250  1305.000  2383.750  436.250  108.000
max     26.000  3308.000  12364.000  2295.000  4189.000  792.000  309.000

            HR       RBI        BB        SO        SB       CS       BA
count  372.000   372.000   372.000   372.000   372.000  372.000  372.000
mean   202.642   901.073   780.105   850.323   196.927   58.987    0.289
std    141.726   484.370   327.453   472.918   185.586   49.322    0.021
min      9.000     0.000   239.000     0.000     7.000    0.000    0.246
25%     79.750   645.000   536.500   448.000    64.500   23.000    0.274
50%    185.500   977.500   719.000   844.000   141.000   52.000    0.288
75%    293.250  1218.500   961.250  1234.250   285.500   84.000    0.300
max    755.000  2297.000  2190.000  1936.000  1406.000  335.000    0.366
```

```
[28]: print(X_test.describe().round(3))
```

```
          YRS         G         AB         R         H       2B       3B  \
count  93.000    93.000     93.000    93.000    93.000   93.000   93.000
mean   17.204  2057.409   7452.968  1135.065  2139.258  374.742   80.398
std     3.154   368.580   1265.371   283.877   415.165   93.929   51.792
min    11.000  1399.000   5472.000   601.000  1660.000  206.000   14.000
25%    15.000  1820.000   6622.000   935.000  1818.000  310.000   45.000
50%    17.000  1997.000   7359.000  1108.000  2054.000  361.000   68.000
75%    19.000  2282.000   8096.000  1283.000  2256.000  432.000   99.000
max    25.000  2850.000  10876.000  1859.000  3430.000  668.000  252.000

            HR       RBI        BB        SO       SB       CS      BA
count   93.000    93.000    93.000    93.000   93.000   93.000  93.000
mean   194.677   867.011   797.387   836.065  191.817   54.473   0.287
std    151.600   495.127   328.755   552.309  166.926   42.508   0.022
min     15.000     0.000   266.000    15.000    8.000    0.000   0.248
25%     78.000   618.000   527.000   359.000   61.000   15.000   0.272
50%    151.000   926.000   750.000   745.000  137.000   50.000   0.285
75%    291.000  1138.000   937.000  1179.000  271.000   83.000   0.299
max    612.000  1922.000  1747.000  2597.000  744.000  173.000   0.340
```