# Data Splits for ML

GROKKERS

AI FOR EVERYONE

# Pre-reqs

- Python
- NumPy and PANDAS, SciPy, Visualizations
- Elementary stats and maths
- Some preprocessing steps – may need ML as well, for advanced topics

# Background

**Numeric Data:** Preprocessing involves handling missing values, scaling to a similar range, and possibly normalizing the distribution.

Text Data**:** Common preprocessing steps include text cleaning (removing stop words, punctuation, etc.), tokenization, and vectorization (converting text into numerical form, such as TF-IDF or word embeddings).

**Image Data**: Techniques like resizing, normalization of pixel values, and data augmentation (creating variations of existing images) are often used.

**Time Series Data**: Dealing with temporal aspects, handling missing values over time, and creating lag features are important steps in preprocessing time series data.

# Topics

- About Data, feature types, tabular form
- General inspection of data quality
- Handling duplicates in data
- Missing value analysis

  (2 parts)
- Handling Outliers
- Cardinality assessment
- Encoding of discrete data
- Scaling and Normalization
- Handling Skewed Distributions
- Data Imbalance Handling
- **Data Splitting**

# Various types of datasets

### 1. Training Set:

- **Purpose:** Used to train the machine learning model.
- **Size:** Typically the largest subset (60-80% of the data).
- **Usage:** The model learns patterns, relationships, and features from this set.

### 2. Validation Set:

- **Purpose:** Used for hyperparameter tuning and model selection during training.
- **Size:** Smaller than the training set (usually 10-20% of the data).
- **Usage:** Helps prevent overfitting by fine-tuning model parameters without contaminating the test set.

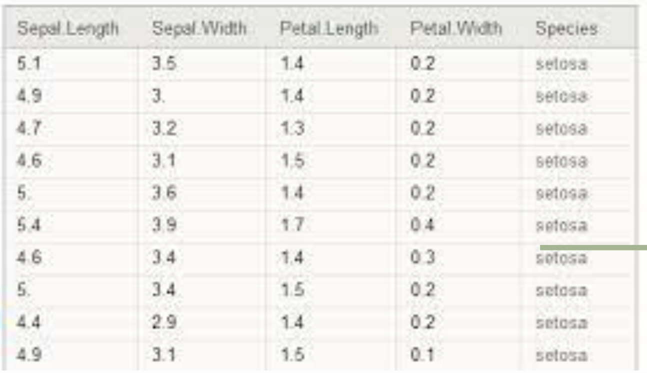
### 3. Testing Set (or Test Set):

- **Purpose:** Reserved for evaluating the model's performance on unseen data.
- **Size:** Independent subset not used during training or validation (10-20% of the data).
- **Usage:** Provides an unbiased assessment of how well the model generalizes to new, unseen instances.

# SPLITTERS/ OPERATORS

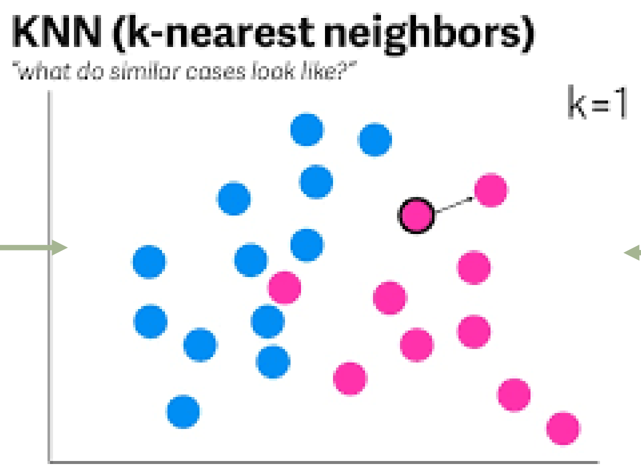
| Model evaluation methods | |
|---|---|
| Random test/input method (predict) | |
| Leave-One-out-Cross-Validation method | • N, the number of data points in the set.<br>• N separate times, the estimator is trained on all the data except for one point and a prediction is made for that point.<br>• the average error is computed and used to evaluate the model. |
| Train-test split, **holdout method** | • data set is separated into two sets, called the training set and the testing set.<br>• The function approximator fits a function using the training set only.<br>• Then the function approximator is asked to predict the output values for the data in the testing set |
| K-Fold cross-validation | • The data set is divided into k subsets, and the holdout method is repeated k times.<br>• Each time, one of the k subsets is used as the test set and the other k-1 subsets are used as training set.<br>• Then the average error across all k trials is computed. |

# EVALUATION – 1 (RANDOM INPUT TEST SAMPLE)



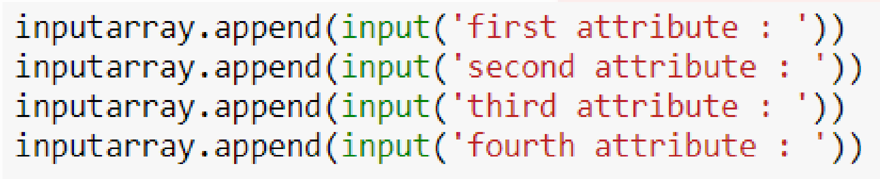
| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3. | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5. | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5. | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |

Training dataset

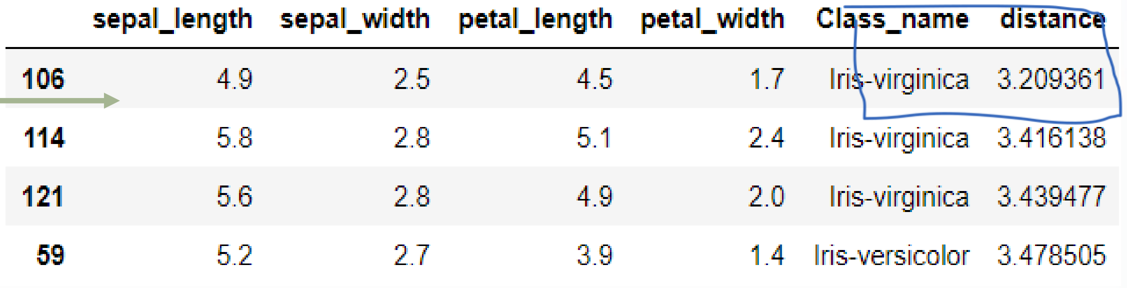

KNN (k-nearest neighbors)

"what do similar cases look like?"

Test sample input



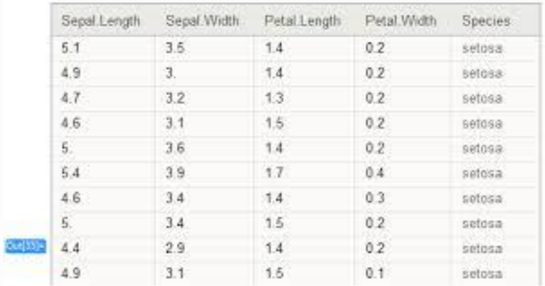
```
inputarray.append(input('first attribute : '))
inputarray.append(input('second attribute : '))
inputarray.append(input('third attribute : '))
inputarray.append(input('fourth attribute : '))
```



| | sepal_length | sepal_width | petal_length | petal_width | Class_name | distance |
|---|---|---|---|---|---|---|
| 106 | 4.9 | 2.5 | 4.5 | 1.7 | Iris-virginica | 3.209361 |
| 114 | 5.8 | 2.8 | 5.1 | 2.4 | Iris-virginica | 3.416138 |
| 121 | 5.6 | 2.8 | 4.9 | 2.0 | Iris-virginica | 3.439477 |
| 59 | 5.2 | 2.7 | 3.9 | 1.4 | Iris-versicolor | 3.478505 |

- Entire training set considered
- Test sample entered externally

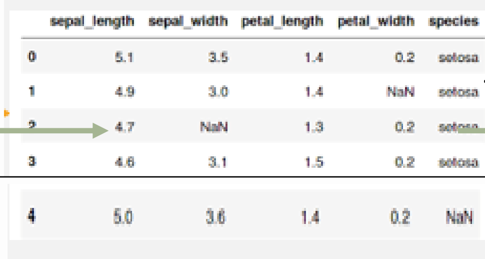
# EVALUATION – 2 (LEAVE-ONE-OUT CROSS-VALIDATION - LOOCV)



| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3. | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5. | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5. | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |

Training dataset

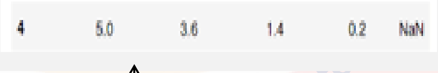
Split into



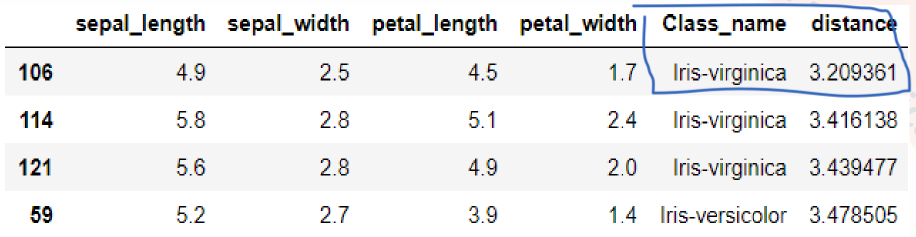
| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 1 | 4.9 | 3.0 | 1.4 | NaN | setosa |
| 2 | 4.7 | NaN | 1.3 | 0.2 | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | NaN |

Training

1 test sample



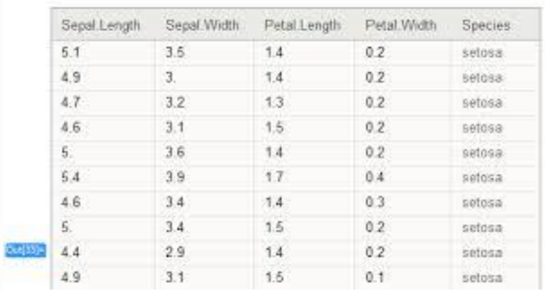
| | | | | | |
|---|---|---|---|---|---|
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | NaN |



| | sepal_length | sepal_width | petal_length | petal_width | Class_name | distance |
|---|---|---|---|---|---|---|
| 106 | 4.9 | 2.5 | 4.5 | 1.7 | Iris-virginica | 3.209361 |
| 114 | 5.8 | 2.8 | 5.1 | 2.4 | Iris-virginica | 3.416138 |
| 121 | 5.6 | 2.8 | 4.9 | 2.0 | Iris-virginica | 3.439477 |
| 59 | 5.2 | 2.7 | 3.9 | 1.4 | Iris-versicolor | 3.478505 |

- is a special case of cross-validation method in which each instance is used once as the test case and all other instances are used as the training set.
- also called as n-fold cross validation.
- utilizes the utmost training instances
- but due to its expensive nature it is usually applied  to small datasets.

# EVALUATION – 3 (HOLDOUT METHOD)



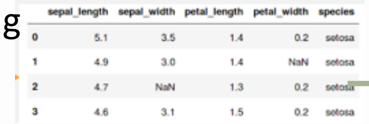
Out[35]:

| Sepal Length | Sepal Width | Petal Length | Petal Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3. | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5. | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5. | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |

Training dataset

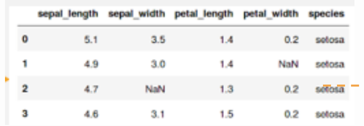
Split into

Training



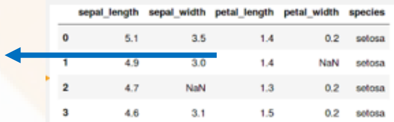
| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 1 | 4.9 | 3.0 | 1.4 | NaN | setosa |
| 2 | 4.7 | NaN | 1.3 | 0.2 | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |

Test

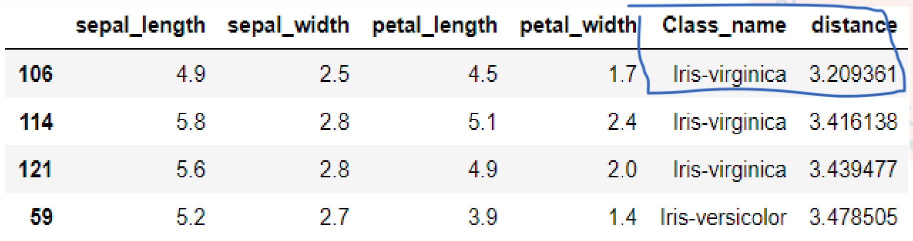


| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 1 | 4.9 | 3.0 | 1.4 | NaN | setosa |
| 2 | 4.7 | NaN | 1.3 | 0.2 | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |

Test sample



| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 1 | 4.9 | 3.0 | 1.4 | NaN | setosa |
| 2 | 4.7 | NaN | 1.3 | 0.2 | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |



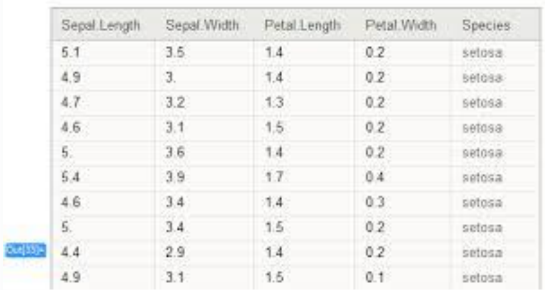
| | sepal_length | sepal_width | petal_length | petal_width | Class_name | distance |
|---|---|---|---|---|---|---|
| 106 | 4.9 | 2.5 | 4.5 | 1.7 | Iris-virginica | 3.209361 |
| 114 | 5.8 | 2.8 | 5.1 | 2.4 | Iris-virginica | 3.416138 |
| 121 | 5.6 | 2.8 | 4.9 | 2.0 | Iris-virginica | 3.439477 |
| 59 | 5.2 | 2.7 | 3.9 | 1.4 | Iris-versicolor | 3.478505 |

- original data set is partitioned into two parts
- 50-50 or 70-30 or 60-40
- randomly divided into the training and test sets

Repeated for each test sample

REPEAT

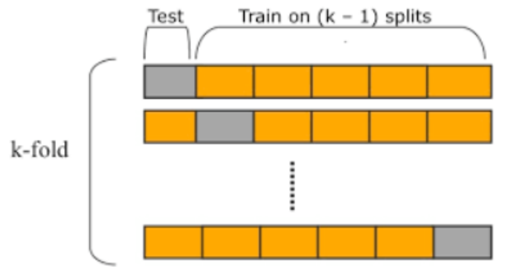
# EVALUATION – 4 (K-FOLD CROSS-VALIDATION METHOD)



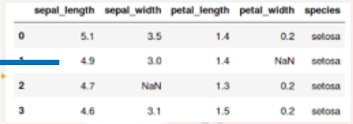
| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3. | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5. | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5. | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |

Training dataset

Split into



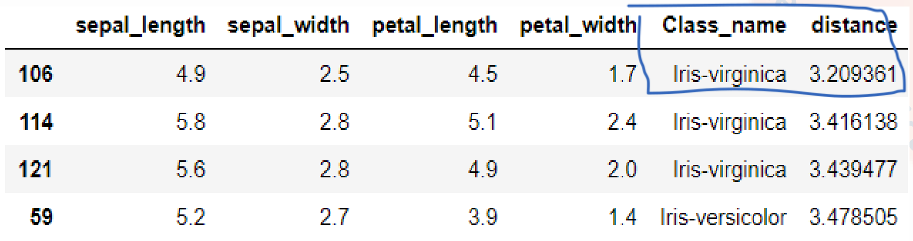
Test sample



|  | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 1 | 4.9 | 3.0 | 1.4 | NaN | setosa |
| 2 | 4.7 | NaN | 1.3 | 0.2 | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |

REPEAT

Repeated for each fold



|  | sepal_length | sepal_width | petal_length | petal_width | Class_name | distance |
|---|---|---|---|---|---|---|
| 106 | 4.9 | 2.5 | 4.5 | 1.7 | Iris-virginica | 3.209361 |
| 114 | 5.8 | 2.8 | 5.1 | 2.4 | Iris-virginica | 3.416138 |
| 121 | 5.6 | 2.8 | 4.9 | 2.0 | Iris-virginica | 3.439477 |
| 59 | 5.2 | 2.7 | 3.9 | 1.4 | Iris-versicolor | 3.478505 |

REPEAT

Repeated for each test sample

- Popular choices for K are 3, 5 and 10 as they’re manageable computationally
- The cross-validation method is used with moderate datasets having instances around hundreds or more.

# SCIKIT LEARN - SPLITTERS

## sklearn.model_selection: Model Selection

| Splitter Classes | |
|---|---|
| model_selection.GroupKFold([n_splits]) | K-fold iterator variant with non-overlapping groups. |
| model_selection.GroupShuffleSplit([…]) | Shuffle-Group(s)-Out cross-validation iterator |
| model_selection.KFold([n_splits, shuffle, …]) | K-Folds cross-validator |
| model_selection.LeaveOneGroupOut() | Leave One Group Out cross-validator |
| model_selection.LeavePGroupsOut(n_groups) | Leave P Group(s) Out cross-validator |
| model_selection.LeaveOneOut() | Leave-One-Out cross-validator |
| model_selection.LeavePOut(p) | Leave-P-Out cross-validator |
| model_selection.PredefinedSplit(test_fold) | Predefined split cross-validator |
| model_selection.RepeatedKFold([n_splits, …]) | Repeated K-Fold cross validator. |
| model_selection.RepeatedStratifiedKFold([…]) | Repeated Stratified K-Fold cross validator. |
| model_selection.ShuffleSplit([n_splits, …]) | Random permutation cross-validator |
| model_selection.StratifiedKFold([n_splits, …]) | Stratified K-Folds cross-validator |
| model_selection.StratifiedShuffleSplit([…]) | Stratified ShuffleSplit cross-validator |
| model_selection.TimeSeriesSplit([n_splits, …]) | Time Series cross-validator |
| model_selection.train_test_split(*arrays, …) | Split arrays or matrices into random train and test subsets |

# Demo using python/sklearn

# (data splitters in SCIKIT-LEARN)

17-08-2024

# Thanks !!

AI FOR EVERYONE

Grokkers

CONCEPTS AND EQUATIONS MADE SIMPLE