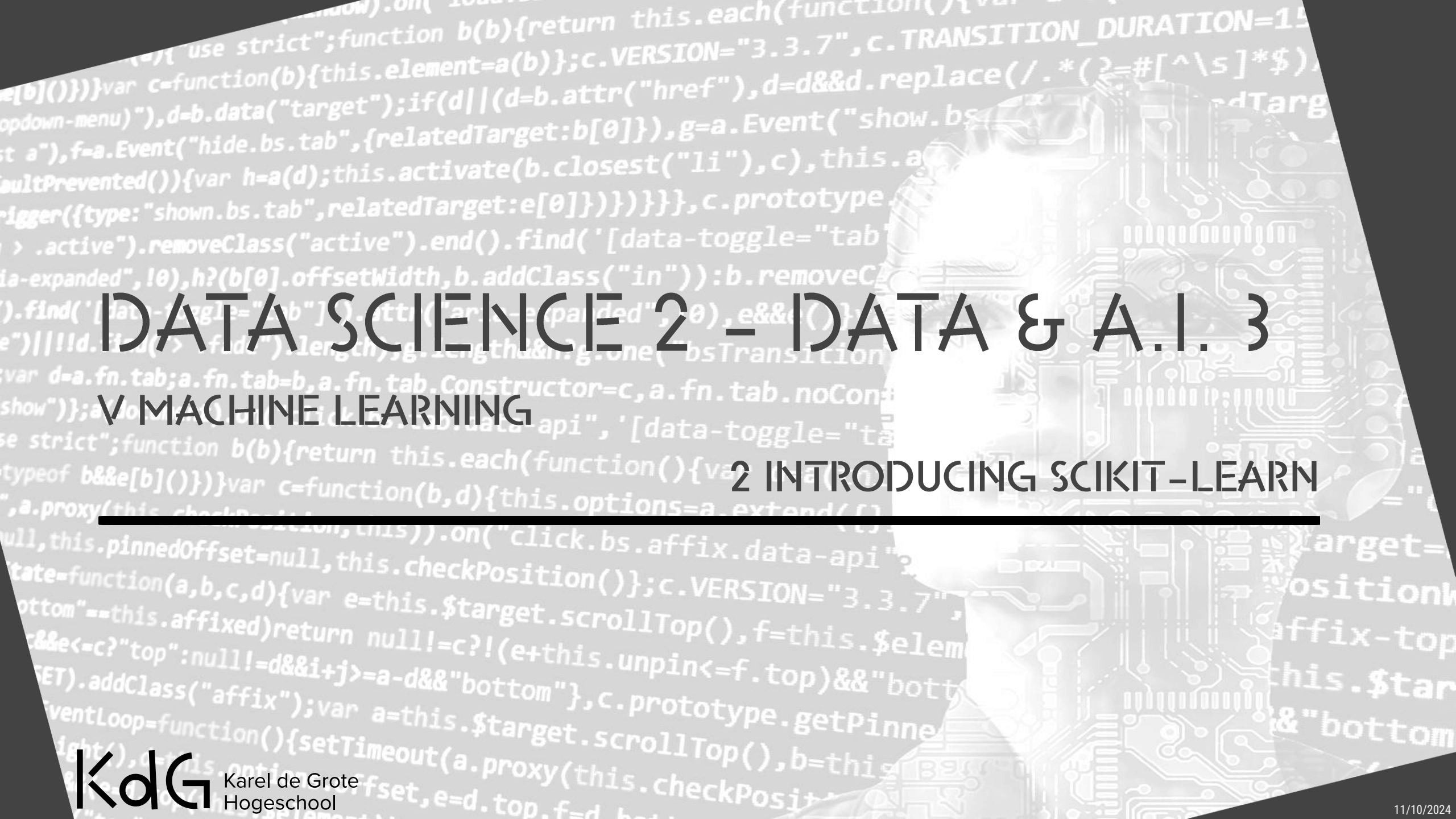
DATA SCIENCE 2 - DATA & A.I. 3

V MACHINE LEARNING

2 INTRODUCING SCIKIT-LEARN



KdG Karel de Grote Hogeschool

DATA SCIENCE 2
DATA & A.I. 3

I

PYTHON BASICS

Python for data science

II

WORKING WITH ARRAYS

Numpy

III

DATA ENGINEERING

pandas

IV

DATA VISUALISATION

Matplotlib

V

MACHINE LEARNING

Automatically find patterns

V

MACHINE LEARNING

scikit-learn

MACHINE LEARNING

01 WHAT IS MACHINE LEARNING

Automatically find patterns

02 **INTRODUCING SCIKIT-LEARN**

Machine learning with Python

03 HYPERPARAMETERS AND CROSS VALIDATION

Holdout samples and cross-validation

04 REGRESSION

Best fitting line

05 DECISION TREES

Best separating lines

06 K-MEANS CLUSTERING

Object grouping

07 ASSOCIATION RULES

Frequent itemsets

08 ARTIFICIAL NEURAL NETWORK

Imitate the human brain

INTRODUCING SCIKIT-LEARN

Machine learning with Python

SCIKIT-LEARN

PYTHON PACKAGE/LIBRARY FOR MACHINE LEARNING

- General purpose machine learning library (for supervised and unsupervised learning)
- Very commonly used

CONSISTENT INTERFACE TO MANY MACHINE LEARING TECHNIQUES

Common methods to.

- Transform data
- Train model
- Predict data
- Validate model

=> Uniform access to many machine learning techniques

DATA PREPARATION

FEATURES

Supervised learning is a kind of dependency model, i.e. a model in which a feature/variable (the dependent variable) is estimated based on a series of other features/variables (independent variables)

In statistics, one talks about dependent and independent variables, in machine learning one talks about target features and predictors or independent features.

- Predictors (independent features/variables) : the features that can be used to estimate the target feature
- Target feature (dependent feature/variable) : the feature to be estimated

LABELED DATASET

Supervised machine learning is based on training, i.e. deriving a model based on examples, with one example being one set of predictors and the known outcome for that set of predictors. Hence, for supervised machine learning, one needs a labeled dataset, a dataset with example with for every example the known outcome (the known outcome of an example is also called a label)

So a labeled dataset contains a set of example features (dependent features or predictors) and the known outcome or label for each of those examples.

DATA PREPARATION

SCIKIT-LEARN : FEATURE MATRIX AND TARGET ARRAY

- **X** = feature matrix : the feature matrix is commonly referred to as X (capital X because it is a matrix)
- **y** = target array : the target array is commonly referred to as y (small y because it is mostly a vector,)
  (capital Y if it is a matrix – for multivariate scikit)

Mind that feature matrix X must be a 2-dimensional structure or matrix in scikit-lean, even it there is only one single feature or predictor. So in case of a single feature in a Pandas Series or a one-dimensional NumPy array, that Pandas series needs to be converted to a Pandas DataFrame explicitly, or the one-dimensional NumPy array must be converted to a two-dimensional NumPy array explicitly.

DATA PREPARATION

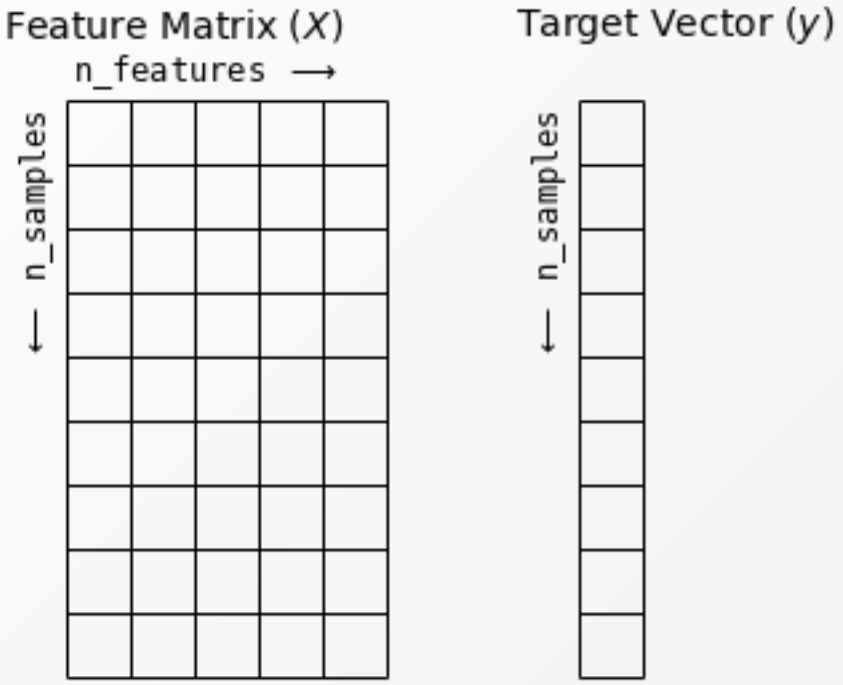
SCIKIT-LEARN : FEATURE MATRIX AND TARGET ARRAY

So the starting point of supervised machine learning is a labeled dataset. In scikit-learn, this labeled dataset is split into two arrays or matrices:

- **Feature matrix** : the table with the values of the independent features or predictors. It is a table structure because it consists of rows (examples) and columns (independent features or predictors. As it is a table, it can be represented in Python by a Pandas DataFrame, or by a 2-dimensional NumPy array (matrix)
- **Target array** : the vector with the labels, the values of the dependent features or target feature. It is mostly a vector because it contains the labels for a single target feature (one label for every example). As it is a vector, it can be represented in Python by a Pandas Series, or by a 1-dimensional NumPy array (vector). However, some scikit-learn methods can handle multile target features at once, and in that case the targe array also becomes a table that can be represented by a pandas dataframe or a 2-dimensional NumPy array (matrix)

Mind that in many cases the independent features ore predictors and the target feature are interchangeable, i.e. any feature can play the role of predictor or target.

DATA PREPARATION



DATA PREPARATION

SCIKIT-LEARN : NORMALIZATION

It is often a good idea to normalize the feature matrix. Normalization is a statistical method that helps mathematical-based algorithms to interpret features with different magnitudes and distributions equally. The most used normalization techniques is transforming your input data are:

- **StandardScaler** : scaling method so that all features have a mean of 0 and a standard deviation of 1. These are the Z-scores: x_norm = (x - mean) / std_dev
- **MinMaxScaler** : scaling method so that all features are in a range between 0 and 1 : x_norm = (x - min) / (max - min)

NORMALIZATION WITH SCIKIT-LEARN

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
scaler = StandardScaler()  # or MinMaxScaler()
X_norm = scaler.fit_transform(X)
```

Methods

scaler.fit(X) : compute the mean and standard deviation of all features (or minimum and maximum) to be used for later scaling.

X_norm = scaler.transform(X): transforms (scales) all the features, based on the fit. Returns array of shape (n_samples, n_features).

X_norm = scaler.fit_transform(X): fit all features and transform (scale) them. Returns array of shape (n_samples, n_features).

X = scaler.inverse_transform(X_norm): undo the scaling for all features. Returns array of shape (n_samples, n_features).

REMARK

If you want the result in a Pandas DataFrame, the following can be used:

X_norm = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

MODEL VALIDATION BASICS (SUPERVISED LEARNING)

COMPARE PREDICTED VALUES WITH REAL VALUES (LABELS/GROUND TRUTH/GOLDEN STANDARD)

For supervised learning, model validation is easy. As you start from a labeled dataset, you have a set of examples for which you know the real outcome (labels). This is also called the ground truth or gold standard. So you can compare the predicted values with the real values (labels):

- Derive (train/fit) a model based on labeled data
- Use the derived model to predict the target variable in the labeled data based on the independent features/predictors in the labeled data
- Compare the predicted target feature values from the previous step with the real target feature values (labels) in the labeled dataset

HOW TO COMPARE REAL AND PREDICTED VALUES FOR REGRESSION METHODS (PREDICTION OF NUMERICAL FEATURES)

Start calculating the difference between the real and predicted values. Based on those differences, validation metrics can be calculated:

- MAE : mean absolute error
- MAPE : mean absolute percentage error
- RMSE : root mean squared error
- …

MODEL VALIDATION BASICS (SUPERVISED LEARNING)

HOW TO COMPARE REAL AND PREDICTED VALUES FOR CLASSIFICATION METHODS (PREDICTION OF CATEGORICAL FEATURES)

Start compiling the confusion matrix based on the number of correctly and wrongly predicted cases. Based on this confusion matrix, validation metrics can be calculated:

- Accuracy
- Precison
- Recall
- F-value
- AUC
- ROC
- …

SEE DATA SCIENCE 1 / DATA & A.I. 2 AND SCIKIT-LEAN DOCUMENTATION

- sklearn.metrics : https://scikit-learn.org/stable/api/sklearn.metrics.html

DATA ANALYTICS PIPELINE

DATA PREPARATION

- Load (labeled) source data
- Compile feature matrix
- Compile target array (for supervised methods)

MODEL SELECTION AND HYPERPARAMETER SELECTION (MODEL SPECIFIC)

- Decide on the method to use (linear regression, decision tree, K-means clustering, …)
- Decide on the hyperparameter to use (degree of polynomial, tree depth, number of clusters, …)

  Hyperparameters are parameters that the algorithm uses to derive a model

  Hyperparameters depend on the method (every methods has it's on kind of hyperparameters)

DERIVE MODEL FROM LABELED DATA (TRAIN MODEL/FIT MODEL)

- Apply the method/algorithm on the labeled data to derive the model

DISPLAY MODEL (MODEL SPECIFIC)

- Display the resulting model

  The resulting model depends on the method used (equation of regression line with intercept and slope, decision tree with nodes and split conditions, groups of observations with centroid, …)

DATA ANALYTICS PIPELINE

VALIDATE MODEL USING LABELED DATA (SUPERVISED LEARNING)

- Predict target feature for the labeled data
- Calculate the difference between predicted and real values for the labeled data / calculate confusion matrix (classification)
- Calculate validation metrics for the labeled data

  Regression : MAE, MAPE, RMSE, …

  Classification : accuracy, precision, recall, f-score, AUC, ROC

APPLY MODEL ON NEW DATA

- Apply the model on new data

  In case of supervised machine learning methods, this will predict the target feature for new data

  In case of unsupervised machine learning methods, this will restructure the feature data (clusters, association rules, new feature matrix with reduced dimensions)

MODEL VALIDATION THE WRONG WAY

DO NOT TRAIN AND VALIDATE THE MODEL ON THE SAME DATA

Training and validating a model on the same data is not a good idea. It is like evaluating students based on an examination with questions that the students got beforehand, including the answers. If a student performs well on that kind of examination, you will never know whether that student really understands the questions and answers, or just learned the answers by heart.

The same problem arises with machine learning; machine learning builds a model by deriving patterns from example data. If you use the same dataset for training and validation, you will never know whether the model really captured the underlying patterns, or just learned the target value by heart for every example (e.g. a decision tree with rules that are too detailed, classifying every single example with a specific rule, instead of a limited set of general rules classifying a large set of similar cases).

OVERFITTING VERSUS GENERALISATION

Mind that the purpose of supervised machine learning is not to predict the labeled training data (you know already the outcome for that data), but to predict new data, data never seen before (and hence not included during training). That means the model should be able to generalise, i.e. be able to capture the overall patterns to be able to predict new data.

So, for validation, we are not that much interested in how well the model can predict the data used for training. We are interested in how well the model can predict new data. A model that scores well on training data but scores bad on new data is an overfitted model. That means it tries too hard to predict the training data, but is not able to grasp the general patterns in the data. It is not able he generalise, so it predicts the training data very well, but is not able to predict new data. Such a model is useless in practice.

What we need is a model that score well on new data.

MODEL VALIDATION THE RIGHT WAY

DO NOT USE THE LABELED TRAINING DATA FOR VALIDATION

Again, we are not interested in how well the model scores on the labeled training data, but on new data. That means that the labeled training data **MUST NOT** be used for validation. Hence, after training, a new labeled dataset needs to be constructed for the validation.

Constructing a new labeled dataset after training is not a practical approach; constructing a labeled dataset might require expert knowledge in the domain at hand. As a data scientist, you cannot have expert knowledge for every potential domain. So labeling a dataset might require external experts. It is not always possible to call those in after every training job.

The solutions is to find a way to use the labeled dataset for both training and validation.

USE A HOLDOUT SAMPLE FOR VALIDATION

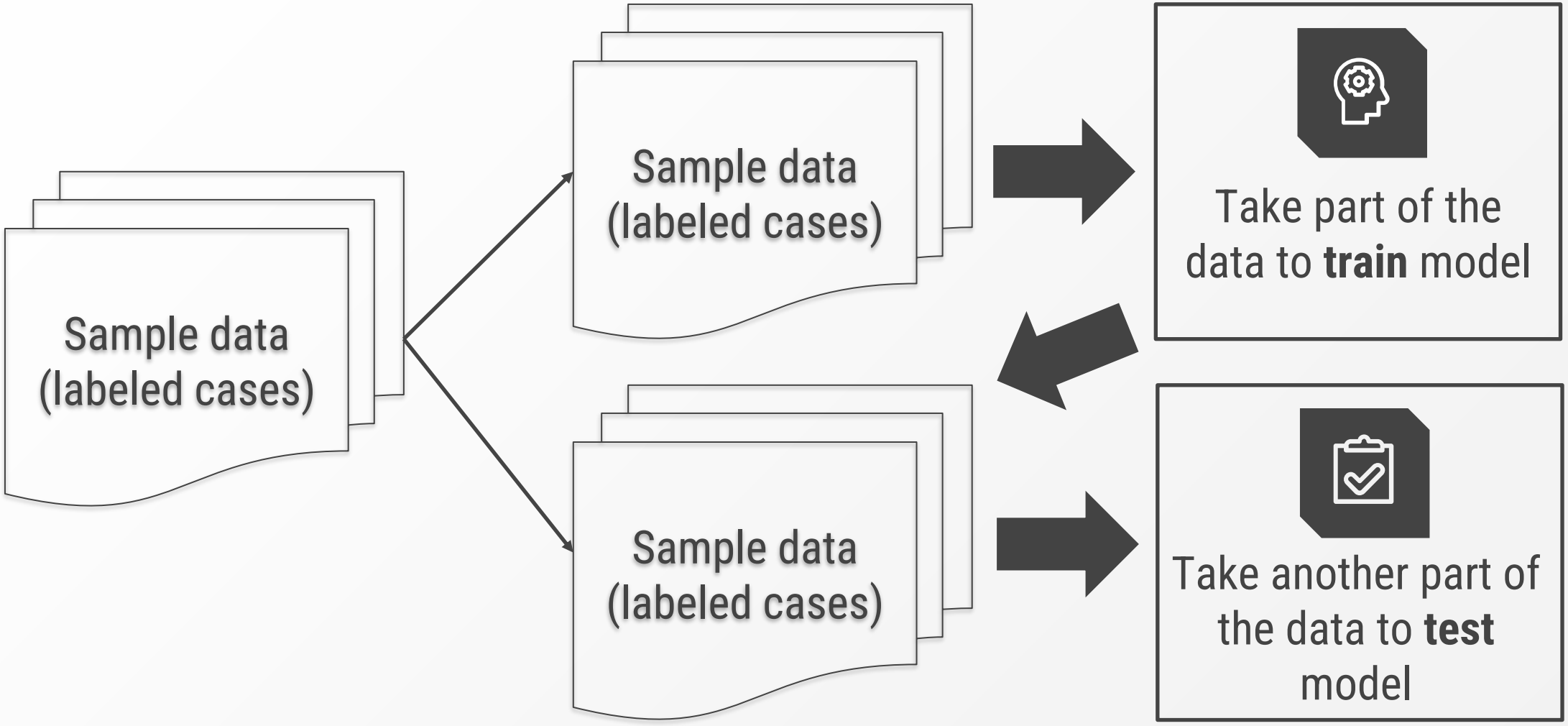
The answer to that problem is to use a holdout sample. That means that some labeled examples are set aside before training and are not used during training. After training, the holdout sample, i.e. the labeled examples that were set aside and not used in training, can now be used for an independent validation as they are new to the model.

SPLIT LABELED DATA INTO TRAINING SET AND TEST SET

In practice, it is as simple as to split the labeled data set in two subsets, one subset to be used for training, and one subset to be used for testing (validation).

The split does not need to be 50/50. The more training examples the better, but of course there need to be enough test examples to get a representative sample for testing. Common ratio's are 70%/80% for training and 30%/20% for testing.

TRAIN - TEST SPLIT



TRAIN – TEST SPLIT WITH SCIKIT-LEARN

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)
```

Parameters

X , y : feature matrix and target vector (more general: can be any number of NumPy or Pandas DataFrames, but they all need to have the same length)

test_size :

- Default: complement of train_size, if train_size is also None: 0.25
- int: number of test samples
- float: proportion of test samples (should be between 0.0 and 1.0)

train_size :

- Default: complement of test_size
- int: number of trainingt samples
- float: proportion of training samples (should be between 0.0 and 1.0)

shuffle (default=True): whether or not to shuffle the data before splitting

stratify (default=None) : for classification, if not None, the distribution of the classes in training and test sets are similar as in this array (usually = y)

Returns

X_train, X_test, y_train, y_test: splitted Numpy arrays of Pandas DataFrames (more general: 2 times the number of arrays given as input)

MODEL VALIDATION BASICS (UNSUPERVISED LEARNING)

METHOD SPECIFIC

For unsupervised learning (clustering, dimensionality reduction, association rules), model validation is not easy. There are no labels to compare with, there is no ground truth or golden standard.

The way to validate this is very method specific.

Some methods allow to derive some metrics that can be used in validation:

- Clustering : Inter cluster distance (should by high), intra cluster distance (should be low), silhouette (should be high)
- Association rules : lift, confidence

MIND VALIDATION METRICS AS THESE ARE USED TO DERIVE THE MODEL

Do not blindly rely on validation metrics of unsupervised methods, as these metrics are used under the hood to derive the model:

- Clustering : inter cluster distance and intra cluster distance is used to form the clusters
- Association rules : lift and confidence are used to select relevant rules

Using these metrics for validation of unsupervised models is a bit like using training data to validate supervised models. If unsupervised models score low on those metrics, you can be sure the model is bad, but the opposite is not true, if unsupervised models score high on those metrics, you still can not know for sure the model is good.

MODEL VALIDATION BASICS (UNSUPERVISED LEARNING)

THE PROOF OF THE PUDDING IS IN THE EATING

The only way to check whether an unsupervised model is good is to implement it and check if it does the job that has to be done, or solve the problem it has to solve. That kind of validation can be very hard and time consuming.