HPC.NRW

NATIONALES HOCHLEISTUNGS RECHNEN

EDIH Rheinland

WEST AI KI-Servicezentrum

PPCES: Machine and Deep Learning

Introduction to AI, Machine Learning and Preprocessing Techniques

DAS KOMPETENZNETZWERK FÜR HOCHLEISTUNGSRECHNEN.

Agenda – Friday, March 14

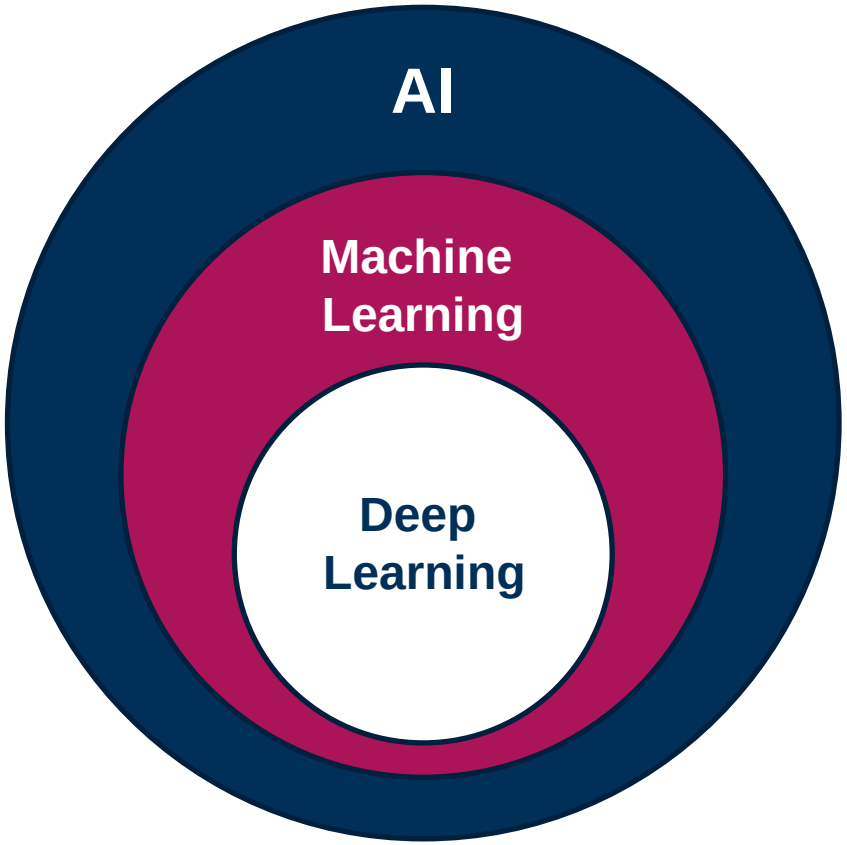
HPC.NRW

Zeit	Thema	Berichtende/r
09:00 – 10:30	**Introduction to AI, Machine and Preprocessing Techniques – Part 1**	Jannis Klinkenberg (RWTH)
10:30 – 11:00	Break	
11:00 – 12:30	Introduction to AI, Machine and Preprocessing Techniques – Part 2 Execution Options for ML / DL Software Lab: Machine Learning with scikit-learn	Jannis Klinkenberg (RWTH)
12:30 – 14:00	Lunch Break	
14:00 – 14:45	Introduction to Deep Learning + Example	Jannis Klinkenberg (RWTH)
14:45 – 15:30	Introduction to Distributed Deep Learning + Example	Jannis Klinkenberg (RWTH)
15:30 – 16:00	Break	
16:00 – 17:00	Lab: Deep Learning with PyTorch	Jannis Klinkenberg (RWTH)

Acknowledgements / Contributors:

- Georg Zitzlsberger, NVIDIA (formerly IT4I Supercomputing Center, Ostrava)
- Dominik Viehhauser and Radita Liem, IT Center RWTH Aachen University

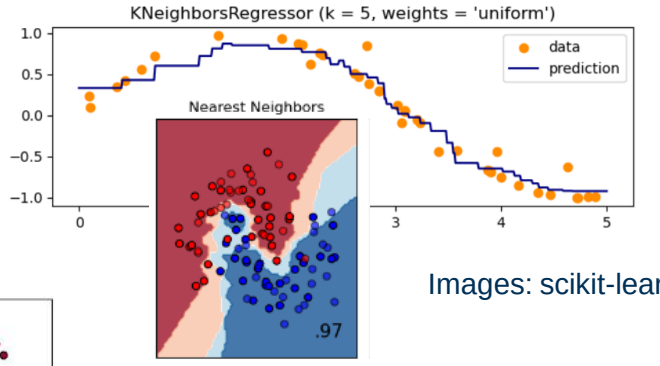
Categorization: AI, Machine and Deep Learning



- **Artificial Intelligence (AI)**
  - Simulation of human intelligence:
    Learning, Reasoning, Problem Solving and Understanding
  - Broad and multi-disciplinary field:
    Computer science, Psychology, Linguistics, …
  - Main types
    - **Narrow AI:** Designed to perform a special task such as image recognition
    - …
    - **General AI:** Can perform any intellectual task like a human
- **Machine Learning (ML)**
  - Subset of AI teaching computers how to learn from data
  - Often requires structured data
  - Typically used to process numerical and categorical data
- **Deep Learning (DL)**
  - Subset of ML
  - Can handle vast amounts and also unstructured data (images, text, …)
  - Uses sophisticated algorithms and (deep) neural networks

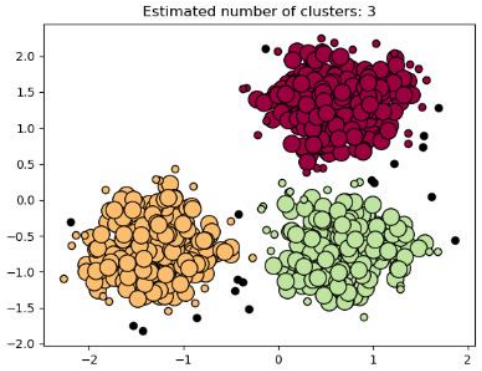
Machine Learning – Disciplines

- **Supervised learning (data is labeled)**
  - Classification → Categorize objects to one or more classes
  - Regression → Predict one or more continuous-valued attributes



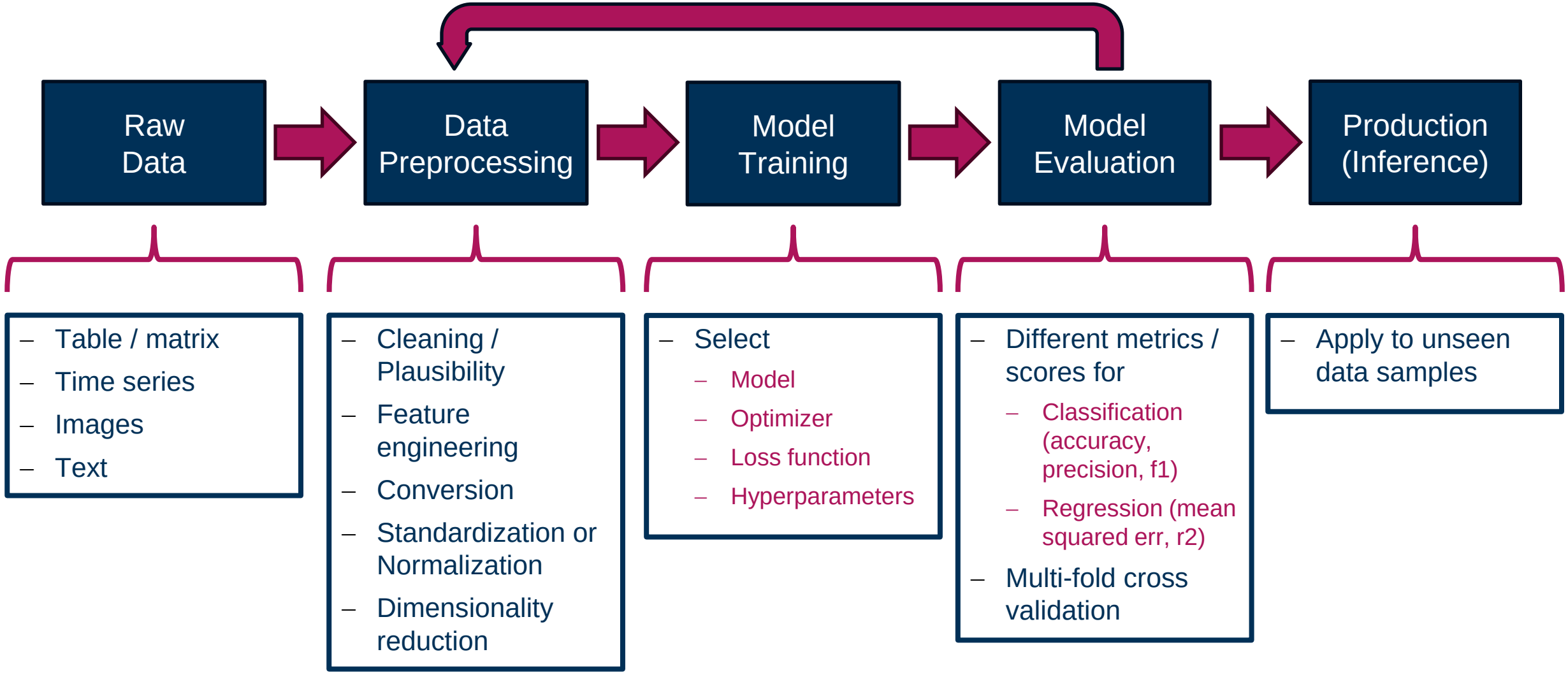
Images: scikit-learn

- **Unsupervised learning (data is not labeled)**
  - Goal: Find structure or pattern in data
  - Clustering → Group similar objects
  - Dimensionality reduction (PCA, Autoencoders)



- **Semi-supervised learning**
  - Combines both supervised and unsupervised techniques
  - Uses both labeled and unlabeled data
- **Reinforcement learning**
  - Develop strategy dynamically using environment feedback (reward)
- …

Typical Machine / Deep Learning Workflow



Example: Feature Engineering and Classification

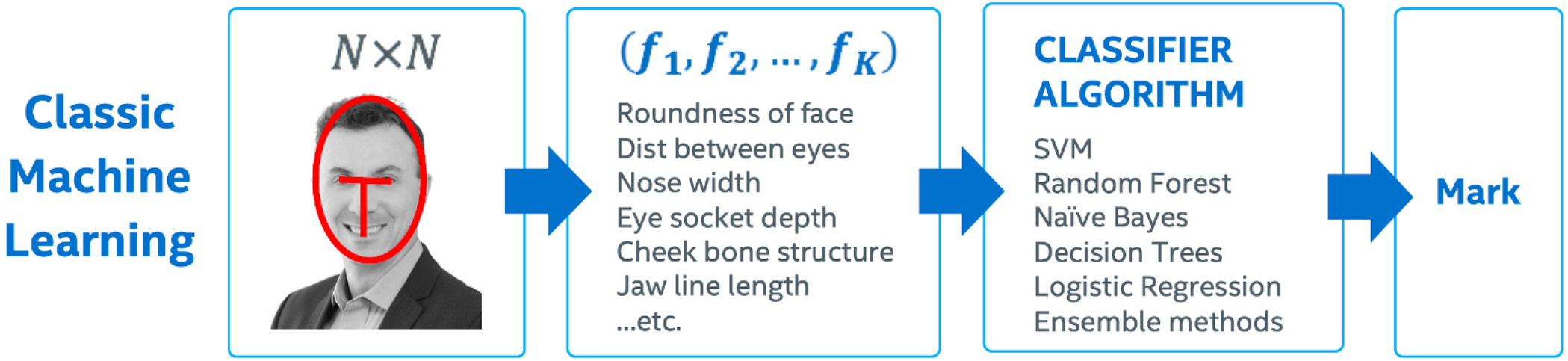


Image: Intel

- **Step 1:** Train selected model with many different images
- **Step 2:** Use model to identify (classify) unseen images
  - **Note:** Data preprocessing steps should be the same for training and inference

scikit-learn – What is it?

HPC.NRW

- **Simple and efficient tools for predictive data analysis**
  - Data preprocessing and other utilities
  - Machine learning methods
  - Visualization
- **Built on NumPy, SciPy, and matplotlib**
  - No separate data types (unlike pandas)
  - Benefit from NumPy and SciPy optimizations
- **Accessible to everybody, and reusable in various contexts**
  - Documented API and ton of examples (Link)
  - Building block for your data analysis
- **Many contributors**
  - Development since 2007
  - Currently version 1.4
  - Open source (BSD) and commercially usable

scikit-learn – Overview

HPC.NRW

- **Available as Python module**

```
# install using pip
pip3 install -U scikit-learn
# install using conda (Intel version)
conda install -c intel scikit-learn
```

```
# create SVM classifier instance
from sklearn import svm
clf = svm.SVC(gamma=0.001, C=100.0)
```

- **Offers supervised / unsupervised methods**
  - Classification
  - Regression
  - Clustering
  - …
- **But also**
  - Data preprocessing
  - Dimensionality reduction
  - Model selection
  - Toy datasets (`sklearn.datasets`)

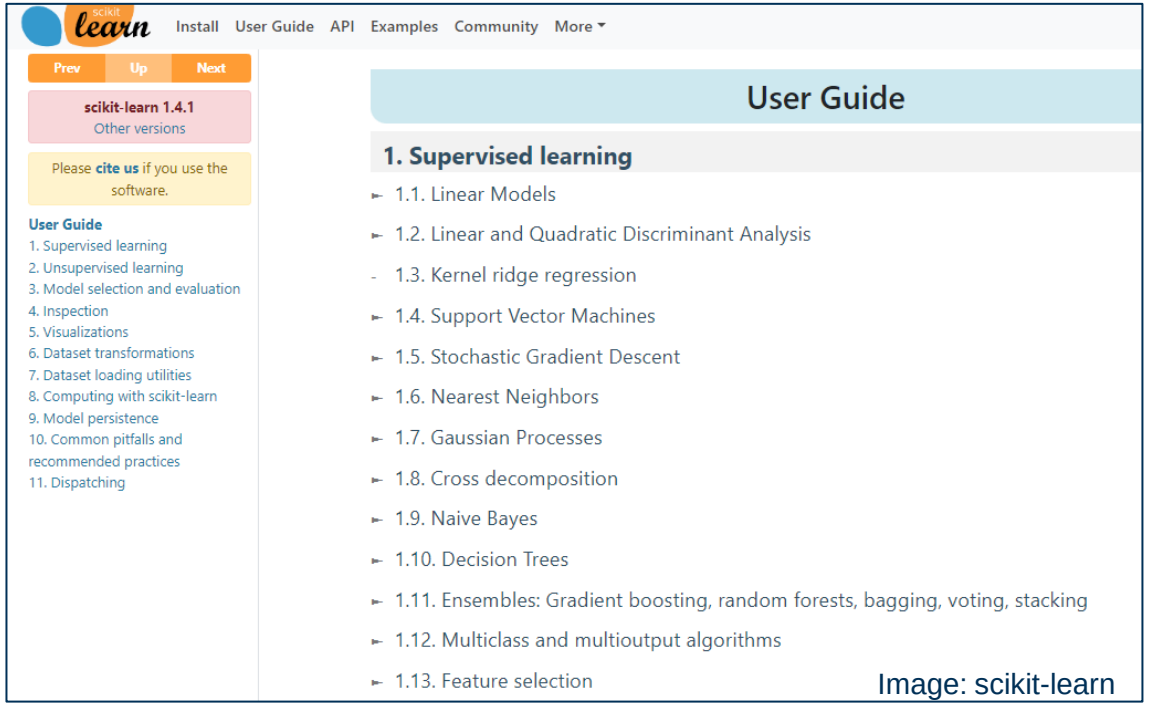


Image: scikit-learn

Datasets

HPC.NRW

- **Multiple toy and real world datasets available**

Function	Description
`load_iris(*[, return_X_y, as_frame])`	Load and return the iris dataset (classification).
`load_diabetes(*[, return_X_y, as_frame, scaled])`	Load and return the diabetes dataset (regression).
`load_digits(*[, n_class, return_X_y, as_frame])`	Load and return the digits dataset (classification).
`load_linnerud(*[, return_X_y, as_frame])`	Load and return the physical exercise Linnerud dataset.
`load_wine(*[, return_X_y, as_frame])`	Load and return the wine dataset (classification).
`load_breast_cancer(*[, return_X_y, as_frame])`	Load and return the breast cancer wisconsin dataset (classification).

- **Parameters / return values and meanings**
  - `X`: Data or feature matrix used to train the model
  - `y`: Labels (targets) in case of supervised learning
  - `n_class`: Number of classes to return
  - `return_X_y`: Boolean whether tuple of (data, label) desired
  - `as_frame`: Boolean whether pandas DataFrame should be returned

Datasets

HPC.NRW

– Example how to use `digits` dataset

```
[1]: from sklearn.datasets import load_digits
     digits = load_digits()
     print(digits.keys())
```

```
dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])
```

```
[2]: print(digits.data.shape)
```

```
(1797, 64)
```

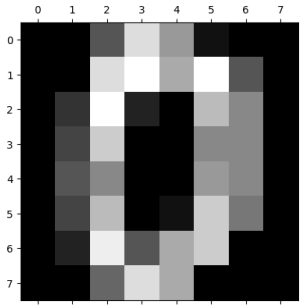
```
[3]: print(digits.target.shape)
```

```
(1797,)
```

```
[4]: print(digits.target_names)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
[5]: import matplotlib.pyplot as plt
     plt.gray()
     plt.matshow(digits.images[0])
     plt.show()
```



Returns:

data : *Bunch*

Dictionary-like object, with the following attributes.

data : *(ndarray, dataframe) of shape (1797, 64)*
The flattened data matrix. If `as_frame=True`, `data` will be a pandas DataFrame.

target: {ndarray, Series} of shape (1797,)
The classification target. If `as_frame=True`, `target` will be a pandas Series.

feature_names: list
The names of the dataset columns.

target_names: list
The names of target classes.

New in version 0.20.

frame: DataFrame of shape (1797, 65)
Only present when `as_frame=True`. DataFrame with `data` and `target`.

New in version 0.23.

images: {ndarray} of shape (1797, 8, 8)
The raw image data.

DESCR: str
The full description of the dataset.

(data, target) : *tuple if `return_X_y` is True*
A tuple of two ndarrays by default. The first contains a 2D ndarray of shape (1797, 64) with each row representing one sample and each column representing the features. The second ndarray of shape (1797) contains the target samples. If `as_frame=True`, both arrays are pandas objects, i.e. `X` a dataframe and `y` a series.

New in version 0.18.

Image: scikit-learn

Data Preprocessing

HPC.NRW

- **Missing values and feature engineering**
  - Imputation of missing values
  - Generating polynomial features
- **Transformation and scaling**
  - Standardization (using mean and std)
  - Normalization
  - Non-linear transformations
  - Custom transformers
- **Special treatments**
  - Encoding categorical features
  - Discretization (binarization, bins)

```
from sklearn import preprocessing as pre
import numpy as np
# data to be scaled
X = np.array([[ 1., -1.,  2.],
              [ 2.,  0.,  0.],
              [ 0.,  1., -1.]])

# create and fit scaler
scaler = pre.StandardScaler().fit(X)

# return mean and scaling factors for features
scaler.mean_
# --> array([1. ..., 0. ..., 0.33...])
scaler.scale_
# --> array([0.81..., 0.81..., 1.24...])

# appy transformation
X_scaled = scaler.transform(X)
# --> array([[ 0.  ..., -1.22...,  1.33...],
#            [ 1.22...,  0.  ..., -0.26...],
#            [-1.22...,  1.22..., -1.06...]])
```

Feature Selection & Dimensionality Reduction (if feature space too high)

HPC.NRW

- **Option 1: Only use a subset of features**
  - Examples: `SelectKBest`, `SelectPercentile`
    - Considers the labels / target variable(s)
    - Ranks feature space based on scores / importance
- **Option 2: Create new features space**
  - Reduce number of features
  - Try to preserve as much information as possible
  - Example: Principal Component Analysis (PCA)
    - Uses Singular Value Decomposition (SVD)
    - Looks for linear combination of features that capture well the variance of original features
    - Note: Not that easy to interprete anymore

```python
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# load data set
X, y = load_digits(return_X_y=True)
X.shape # --> (1797, 64)

# apply selection
X_new = SelectKBest(chi2, k=8).fit_transform(X, y)
X_new.shape # --> (1797, 8)
```

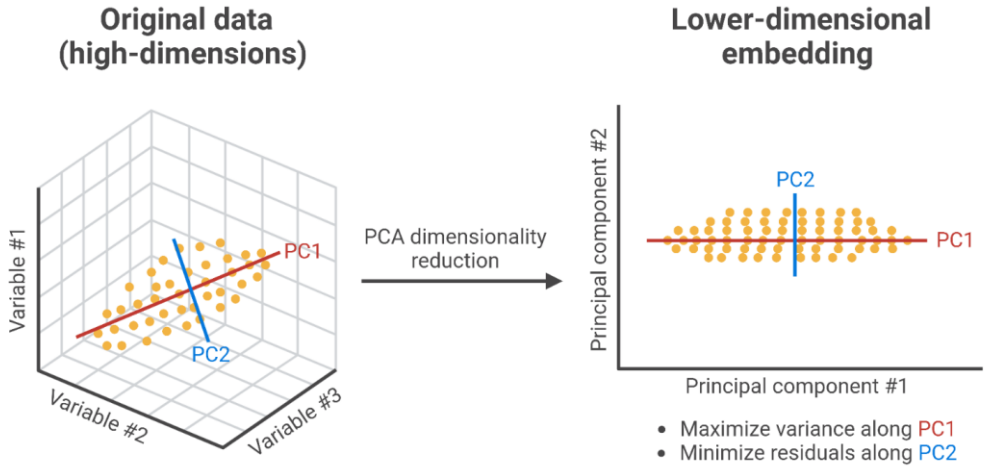


Image: biorender.com

Classification

HPC.NRW

- **Many classification methods available (classifier comparison)**
  - Classification performance might differ depending on your data set

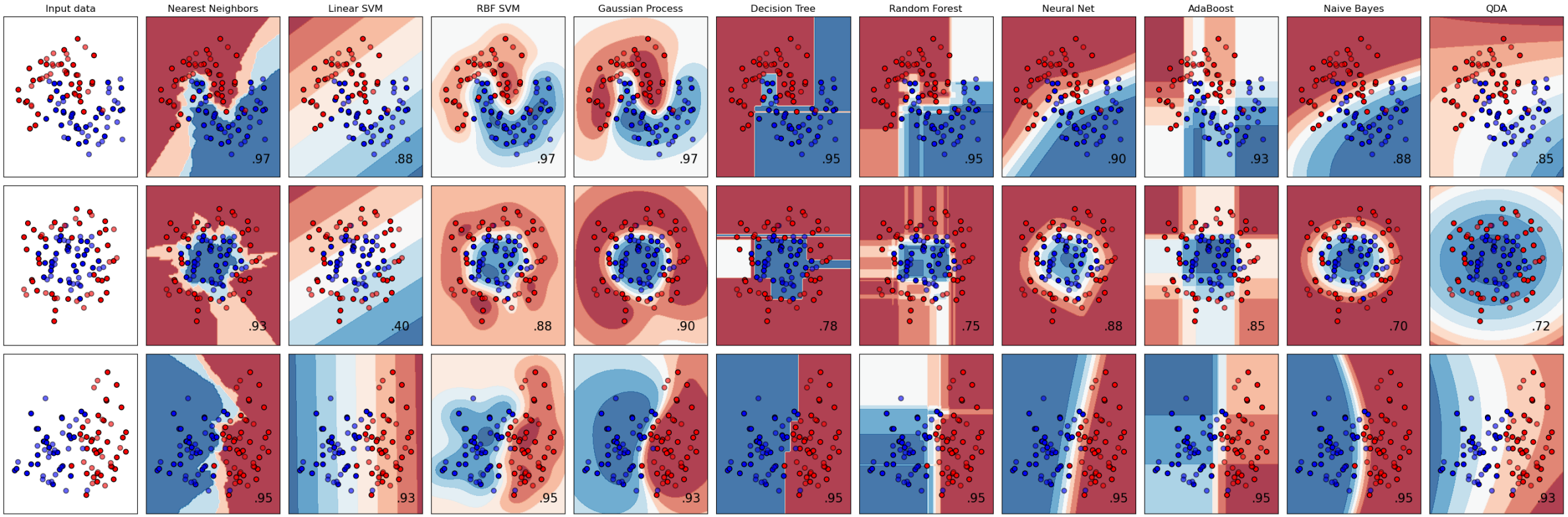


Image: scikit-learn

Classification – Example (`digits`)

HPC.NRW

– **Goal:** Recognizing hand-written digits

```python
from sklearn import datasets, metrics, svm
from sklearn.model_selection import train_test_split

# load dataset & flatten the images
digits = datasets.load_digits()
data = digits.images.reshape((len(digits.images), -1))

# create a support vector classifier
clf = svm.SVC(gamma=0.001)

# split into train and test data
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False
)

# train and predict
clf.fit(X_train, y_train)
predicted = clf.predict(X_test)
```

```python
from sklearn.metrics import ConfusionMatrixDisplay as cf

# evaluate classifier performance
print(f"{metrics.classification_report(y_test, pred)}")
disp = cf.from_predictions(y_test, pred)
plt.show()
```

```
Classification report for classifier SVC(gamma=0.001):
              precision    recall  f1-score   support

           0       1.00      0.99      0.99        88
           1       0.99      0.97      0.98        91
           2       0.99      0.99      0.99        86
           3       0.98      0.87      0.92        91
           4       0.99      0.96      0.97        92
           5       0.95      0.97      0.96        91
           6       0.99      0.99      0.99        91
           7       0.96      0.99      0.97        89
           8       0.94      1.00      0.97        88
           9       0.93      0.98      0.95        92

    accuracy                           0.97       899
   macro avg       0.97      0.97      0.97       899
weighted avg       0.97      0.97      0.97       899
```

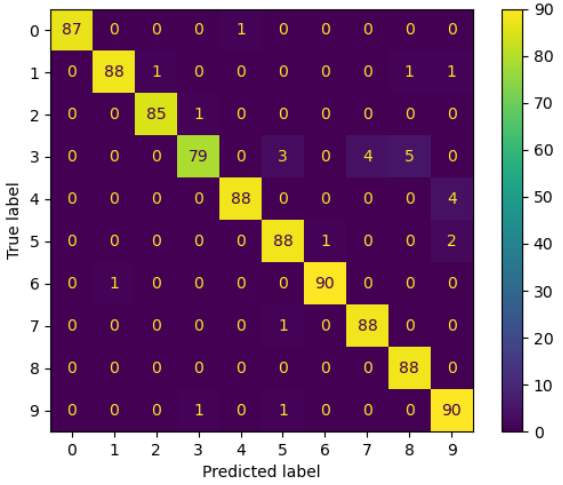


Image: scikit-learn

Regression

- **Reminder**
  - Classification: predict categorical attributes
  - Regression: predict continuous-valued attributes
- **Several types of regressions**
  - Linear models
    - Ridge
    - Bayesian
    - Stochastic Gradient Descent (SGD)
  - Non-linear models
    - Support Vector Regression (SVR)
    - Kernel Ridge Regression (KRR)
  - Other methods
    - Nearest Neighbors
    - Decision Trees
    - Ensembles: Random Forests, Boosting

Example: Decision Tree Regression

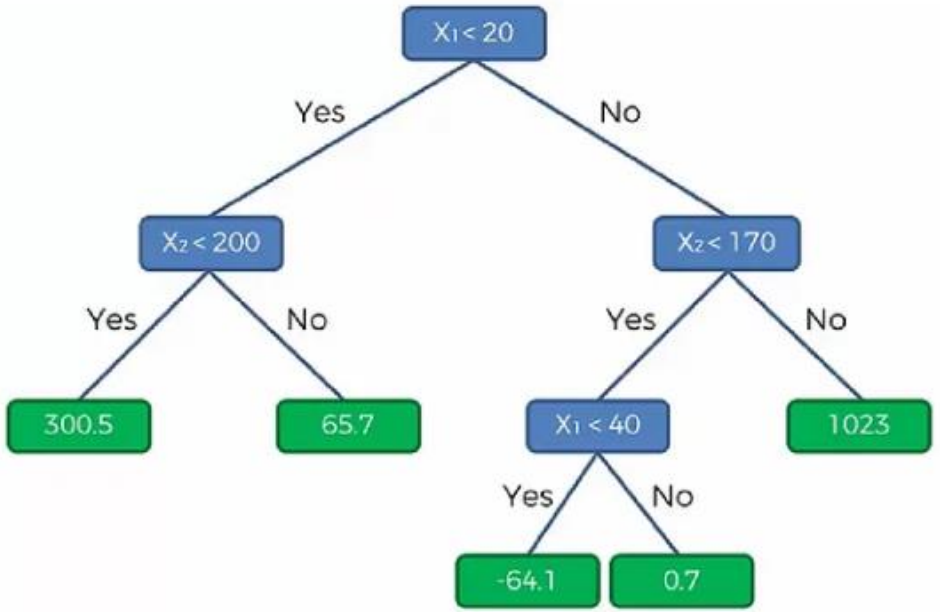
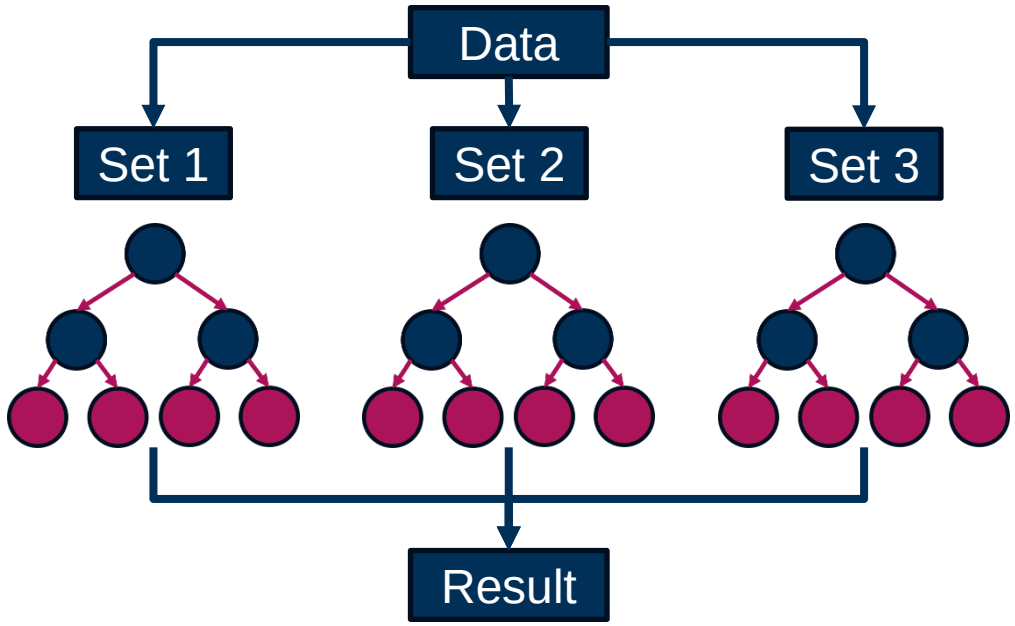


Image: Source

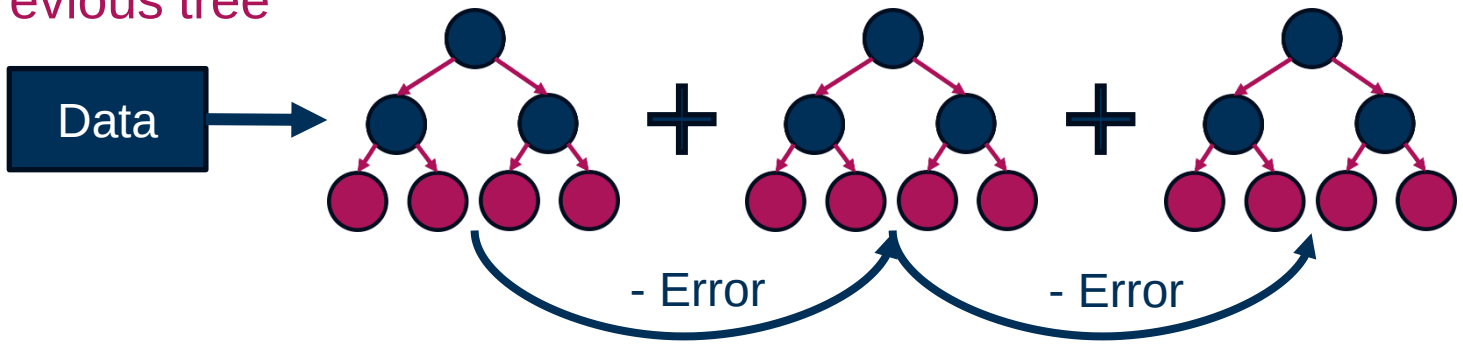
Regression – Ensembles Techniques

HPC.NRW

- **Example:** Random Forests
  - Train multiple separate classifiers
  - Prediction is majority vote or average over all trees
  - Easily extends to multi-output problems



- **Example:** Decision Tree Boosting
  - Train multiple classifiers and apply after each other
  - Each tree compensates error from previous tree
  - Requires differentiable loss function



Clustering

HPC.NRW

- **Clustering is an unsupervised learning technique**
  - Training: Identify patterns or groups automatically in data set (based on similarity or distance)
  - Retrieve labels / metrics from trained model

- **Several methods for different use cases (Documentation)**

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance

Image: scikit-learn

Clustering – Examples

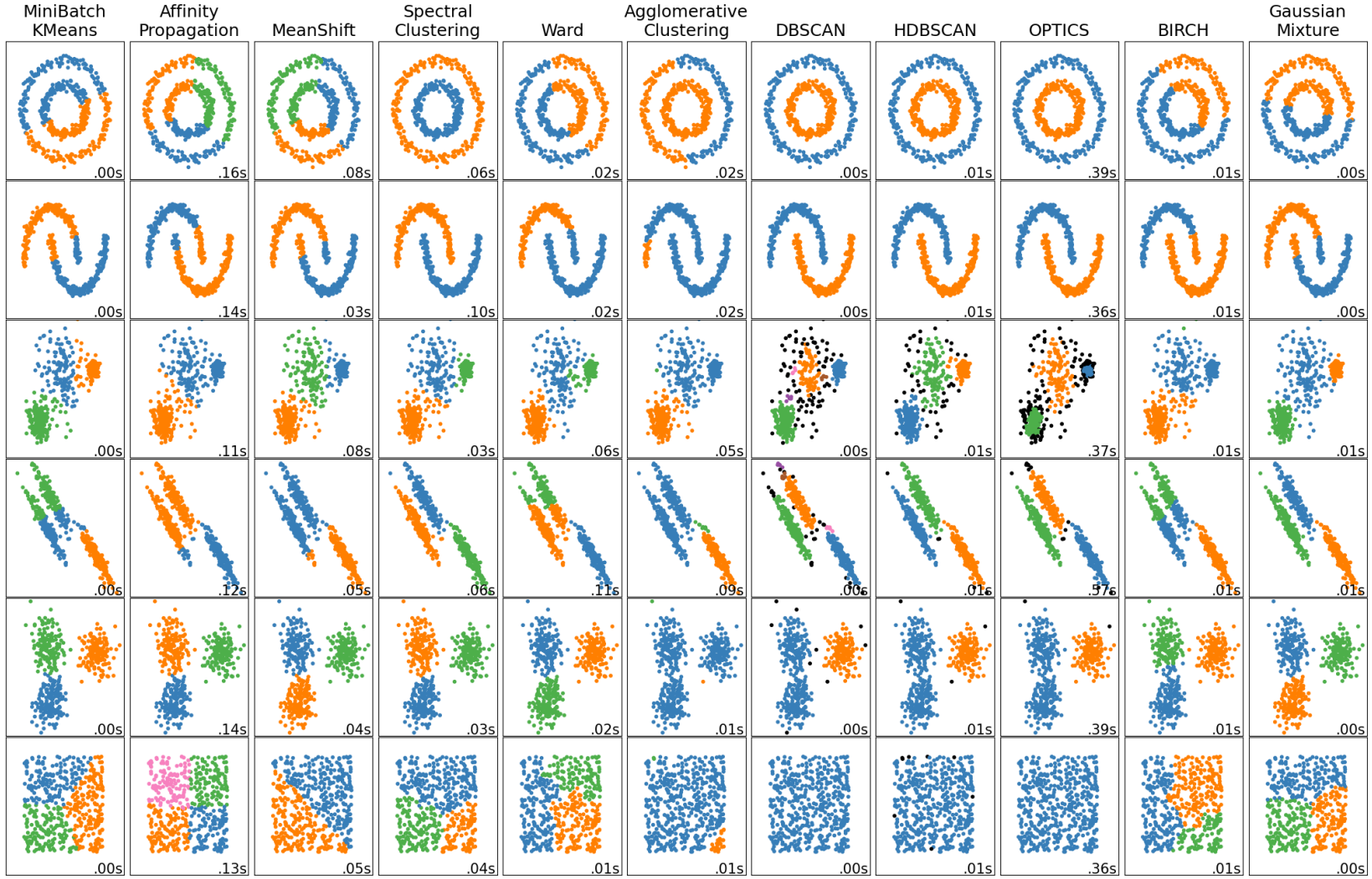


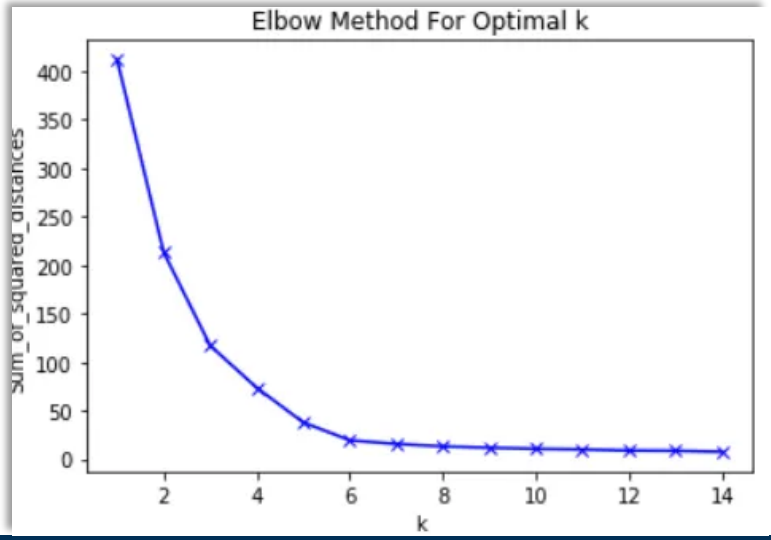
Image: scikit-learn

Clustering – Numer of Clusters

HPC.NRW

- **Question:** How to estimate number of clusters?
  - Silhouette score [-1, 1]
    - Evaluates how well clusters are separated
    - `+1`: best value
    - `-1`: worst value (values assigned to wrong cluster)
    - `0`: indicator for overlapping clusters
  - Ellbow plots
    - Empirically tries to determine the best value
    - Based on Silhouette score or distances
    - Here: best value is 5

```python
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
# create dataset
X, y = make_blobs(random_state=42)
# train clustering model
kmeans = KMeans(n_clusters=2, random_state=42)
# determine score
score = silhouette_score(X, kmeans.fit_predict(X))
```



What is the best method for my task?

- **Highly depends on the use case / problem**
  - How much training data do you have?
  - Is your problem continuous or discrete?
  - What is the ratio between $\#_{features}$ and $\#_{samples}$?
  - Would reducing dimensionality be an option?
  - Do you have a multi-task/-label problem?

- **Further hints / overviews**
  - scikit-learn recommendations (Link)
  - Data science cheat sheet (Link)

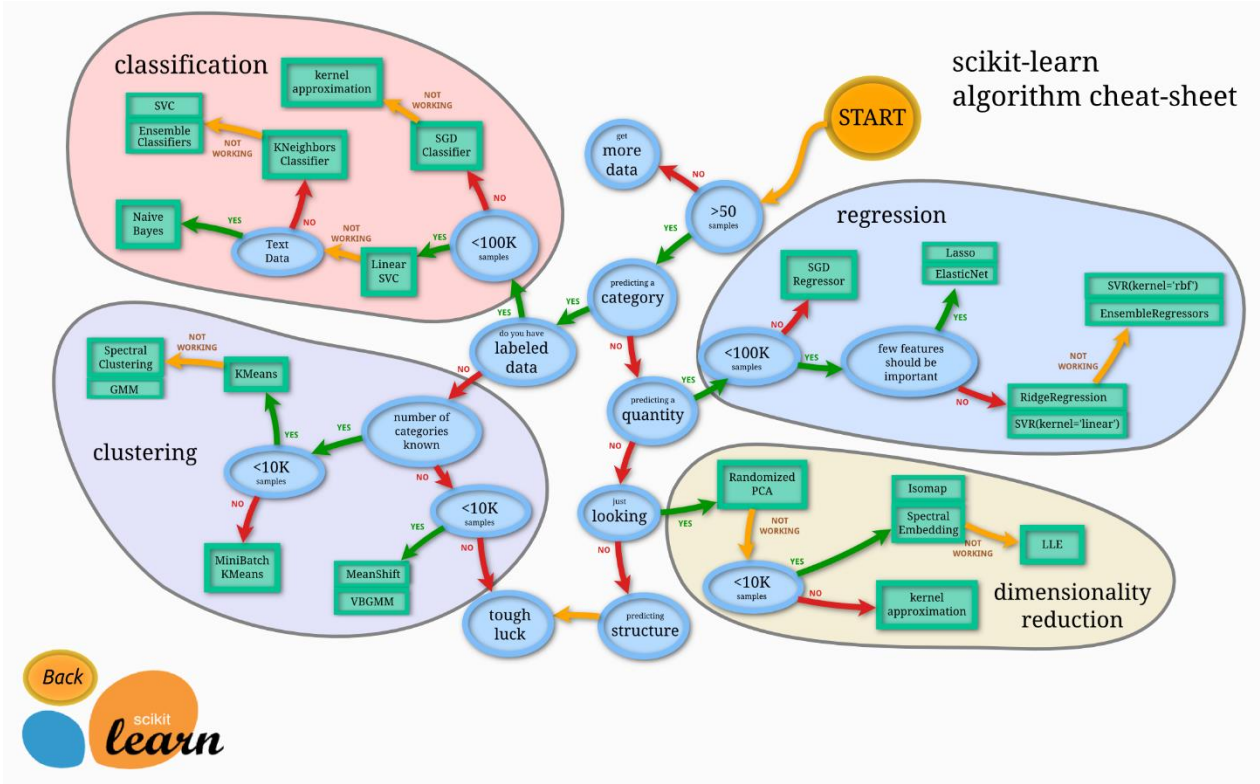


Image: scikit-learn