TRAINING: Data Science and AI for Medicine Training School 2026
Day 1: Machine Learning Basics – Theory and Practice

SPEAKER: Matthias Täschner

GEFÖRDERT VOM
Bundesministerium für Forschung, Technologie und Raumfahrt

SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

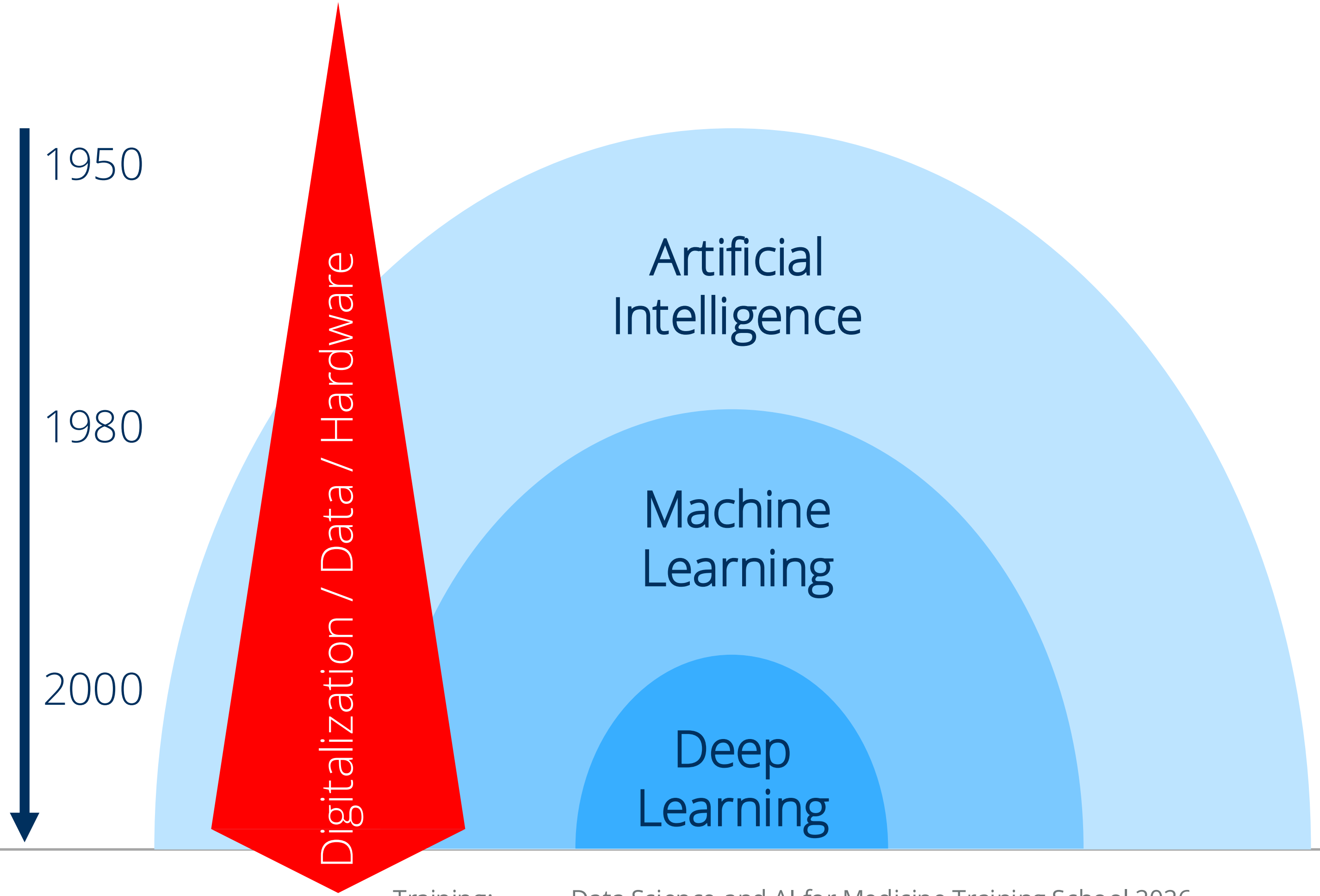
Using materials from Robert Haase (DSC ScaDS.AI / Leipzig University)

# AGENDA

- Theory and Terms
  - Areas of Artificial Intelligence (AI)
  - Paradigms of Machine Learning (ML)
  - ML Model Training
- Theory and Practice
  - Model Types in ML
  - Practical Use of ML Libraries in Python

# Areas of Artificial Intelligence (AI)

## Historical phases of AI




Programs mimic intelligent human behavior

Programs independently identify connections and patterns in (structured) data

Use of neural networks with (very) many layers

# Areas of Artificial Intelligence (AI)

## Specialized (weak/narrow) AI

- Application-specific
- Trained on labeled data
- Adaptation for other applications not possible/difficult
- Cannot extrapolate

Great for data analysis

## General (strong) AI

- Human-like capabilities
- Access to the knowledge of humanity, beyond the individual
- Can work creatively and create new solutions for universal tasks

# Areas of Artificial Intelligence (AI)

Model Family

Areas, paradigms, model families (not exhaustive)

**Paradigms**

- Supervised
- Unsupervised
- Reinforcement
- *Semi-Supervised*
- *Self-Supervised*

**Problems**

- Regression
- Classification
- Clustering
- Dimension reduction
- Anomaly detection
- Sequence-to-sequence
- Recommendations
- Generative modeling
- Time series prediction
- …

Artificial Intelligence

Machine Learning

Linear Models

Decision Trees

Kernel-based

Clustering

Ensembles

…

Neural Networks

Feed-Forward

Multi-Layer Perceptron

…

Deep Learning

Autoencoder

CNN

RNN

…

Generative AI

Transformer

…

# Areas of Artificial Intelligence (AI)
## Applications

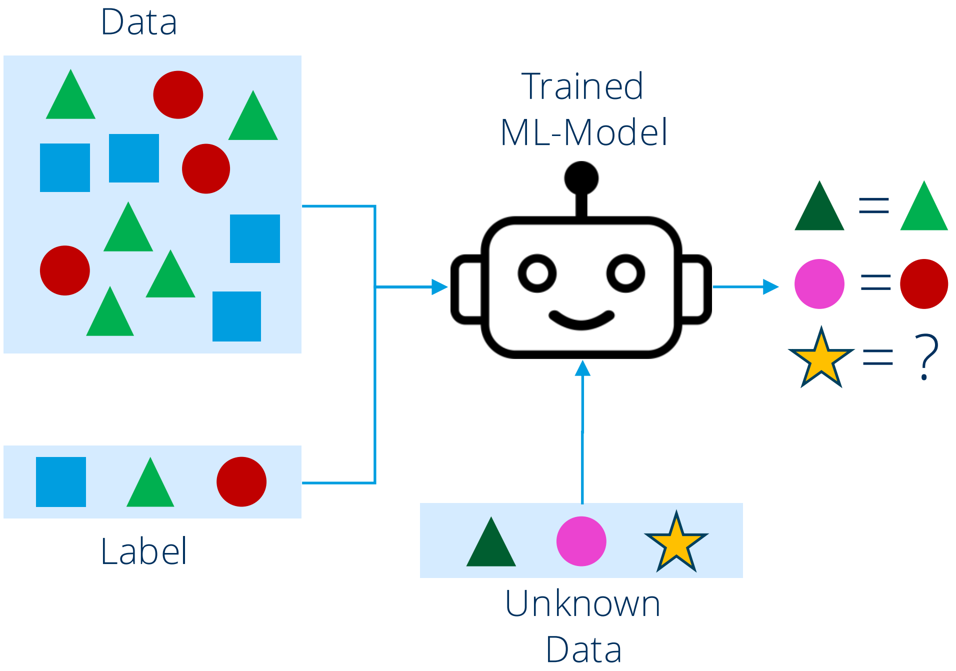
- **Regression**: Prediction of a continuous (numerical) value
- **Classification**: Prediction of a discrete label/class/category
- **Clustering**: Grouping of data points based on their properties
- **Dimensionality Reduction**: Compression of high-dimensional data to a few informative dimensions
- **Anomaly Detection**: Detection of data points that deviate from the "normal" pattern
- **Sequence-to-Sequence**: Converting one ordered sequence of data points into another
- **Recommendations**: Ranking data according to relevance or predicting user ratings
- **Generative Modeling**: Learning data distribution and using it to generate new, synthetic data

# Paradigms of Machine Learning (ML)
## Supervised Learning

### Procedure
- ML models are trained using pre-labeled data
- Input data and the desired target values are provided for training
- Prediction of target values on new, previously unknown data of the same format

### Application examples
- Classification, regression
- Anomaly detection
- Generative modeling

### Algorithms / Model types - Examples
- Linear Regression
- Decision Trees & Random Forest (DT & RF)
- Support Vector Machines (SVM)
- Artificial Neural Networks (ANN/NN)

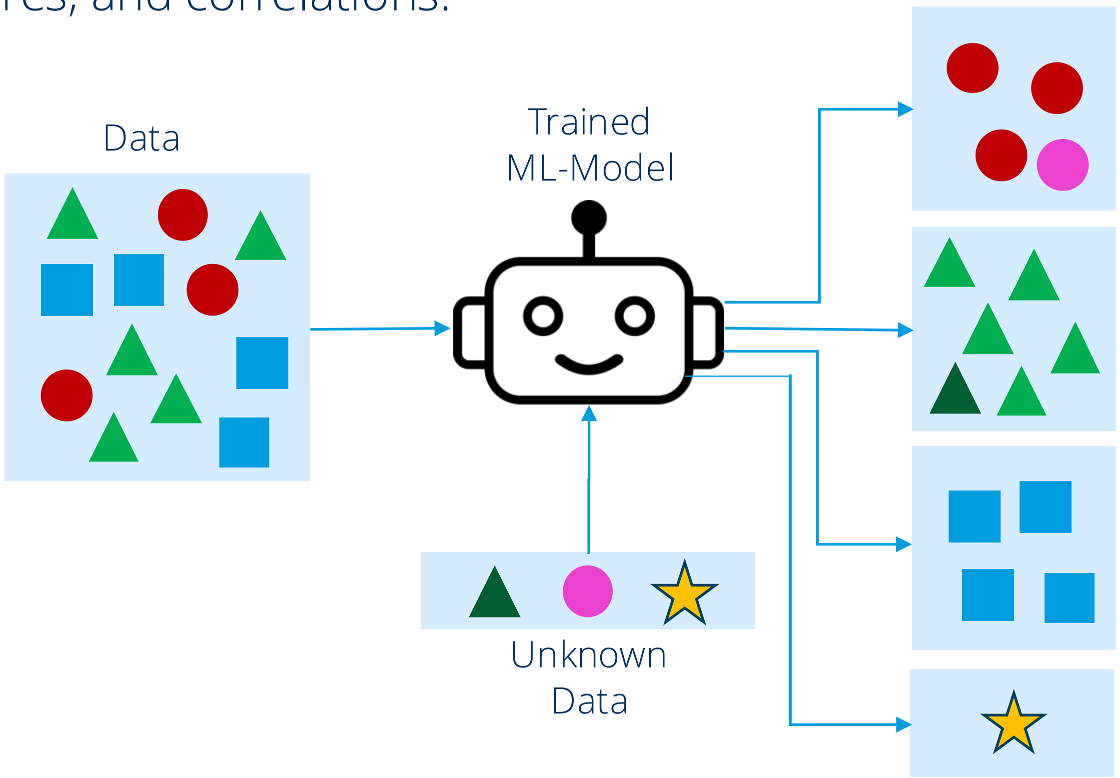


# Paradigms of Machine Learning (ML)
## Un-Supervised Learning

### Procedure
- ML models are trained with unlabeled data.
- Models independently recognize patterns, structures, and correlations.

### Application examples
- Clustering
- Dimensionality Reduction
- Anomaly Detection

### Algorithms / Model types - Examples
- K-Means Clustering
- Autoencoder

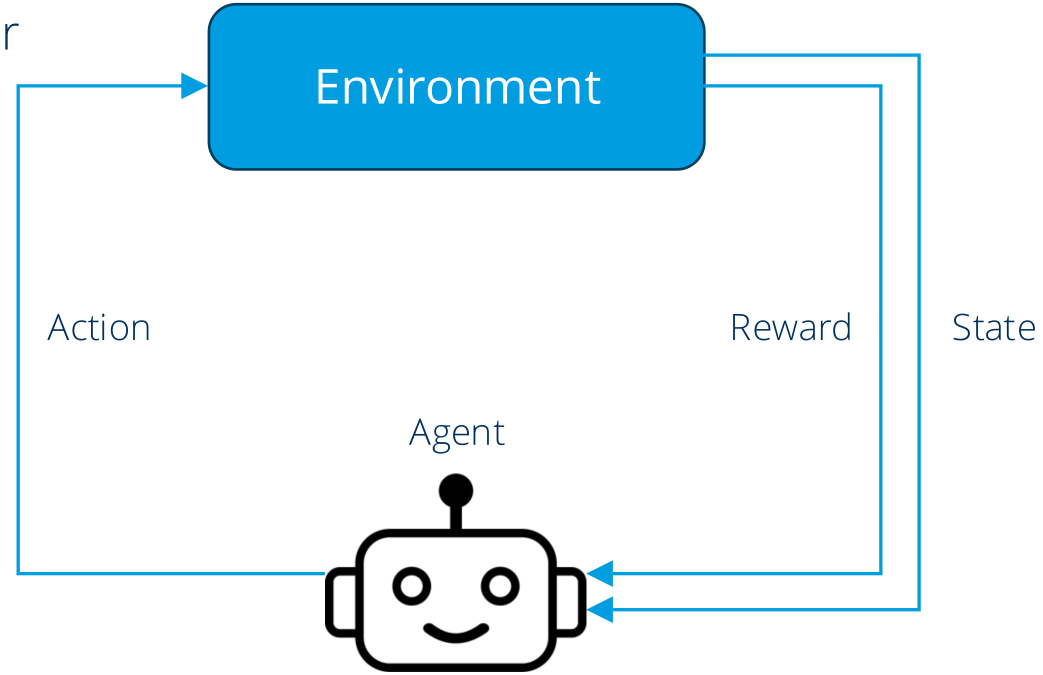


# Paradigms of Machine Learning (ML)
## Reinforcement Learning

### Procedure
- An agent is trained to maximize a certain reward in an environment through its decisions – “trial and error”
- Rules define the agent's possible actions
- Rewards and punishments influence the agent's behavior

### Application examples
- Game agents, e.g., in chess or Go
- Automation systems, robotics
- Simulations
- Training process of Large Language Models

### Algorithms / Model types - Examples
- Q-Learning
- Markov Decision Processes (MDP)
- Monte Carlo Methods



# Paradigms of Machine Learning (ML)
## Linearity: Linear vs. Non-linear Models

- Linear model ≠ straight line curve
- Linearity refers to model parameters

*A model is linear if its prediction is a linear combination of its input features.*
*(e.g., weighted sum of input data)*

- Linear models
  - Easy to interpret, based on well-understood principles, fast and efficient training
  - Can only model simple, linear relationships in the data
  - Examples: Linear or Logistic Regression, ARIMA, linear SVM, etc.
- Non-linear models
  - Also model complex, non-linear relationships in the data
  - More difficult to interpret, complex training with more parameters, often more data required
  - Examples: Decision Tree, kernel-based SVM, k-NN, Neural Networks, etc.

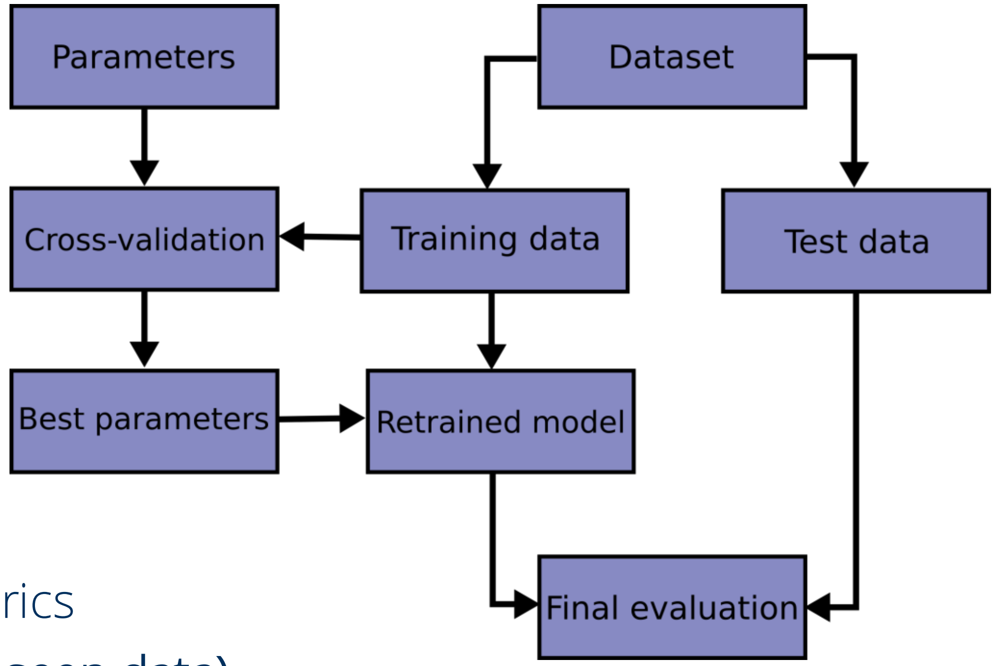
# Paradigms of Machine Learning (ML)
## Probabilistic Models

- Trained to learn the (statistical) distribution in the data instead of predicting individual points
- Input: Prior knowledge (prior) combined with input data,
- Output: Prediction a downstream distribution (posterior), sampling from this distribution
- Allows modelling uncertainty/confidence – returns “what” and “how certain”
- Examples: Naive Bayes, Bayesian neural networks, etc.

# ML Model Training
## Basic Terms and Concepts

### Core steps of ML model training process

- Data Preparation
  - Understand / prepare data (type, scale, missing values, outliers, bias, ...)
  - Feature engineering (select / create features the model can learn from)
  - Divide data for training and tests (train-validate-test split)
- Model Selection
  - Get a baseline for comparison
  - Select the models you want to work with
- Model Training with Parameter Tuning
  - Train model on training data
  - Tweak model parameters to increase prediction quality
- Model Evaluation
  - Evaluate trained model on test data with on selected metrics
  - Overall goal: good model generalization (prediction on unseen data)



That's it for the theory-**only** part ☺

Next:

Mixed theory-practice parts on
ML Models and how to use them in Python

Theory

Practice

# Types of Machine Learning Models
## Regression

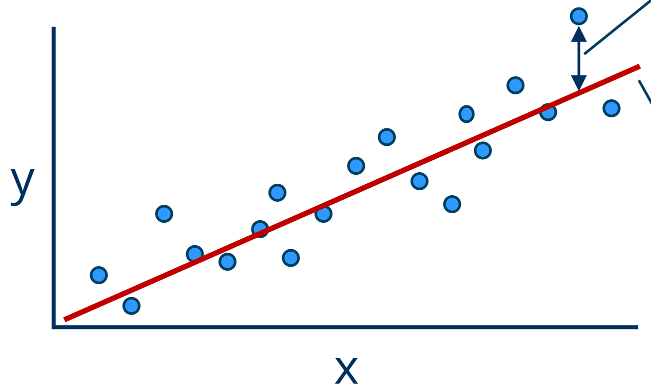
Theory

### Linear Regression

- Models a straight line → $y = w * x + b$
  - y = Target variable
  - x = Input Data / Features
  - w = Regression Weights (slope)
  - b = Constant (y intersect)
  - *[x and w can also be vectors]*

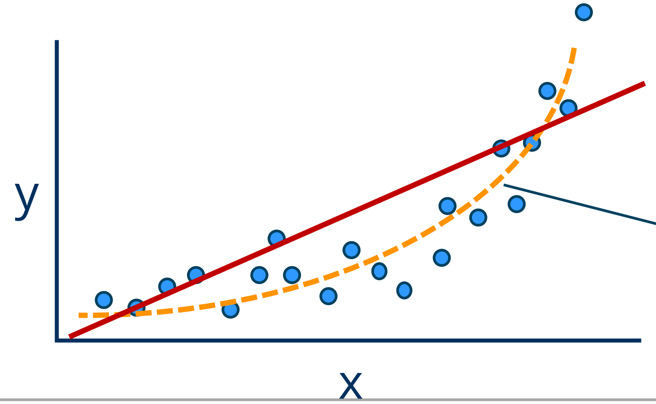
**Model params**

- Attempts to minimize the sum of squared residuals (mean squared error) **Metric**
- Prediction of continuous numerical value, assuming linear relationship between features and target
- Fast, easy to interpret
- Problems with complex, non-linear relationships in the data (e.g., curves)
- Extension: Polynomial Regression
  → $y = w_1 * x + w_2 * x^2 + \ldots + w_d * x^d + b$




"Residual" difference between predicted and observed values

Modeled linear equation for ideal features




Better modeled using polynomial regression

# Types of Machine Learning Models
## Regression

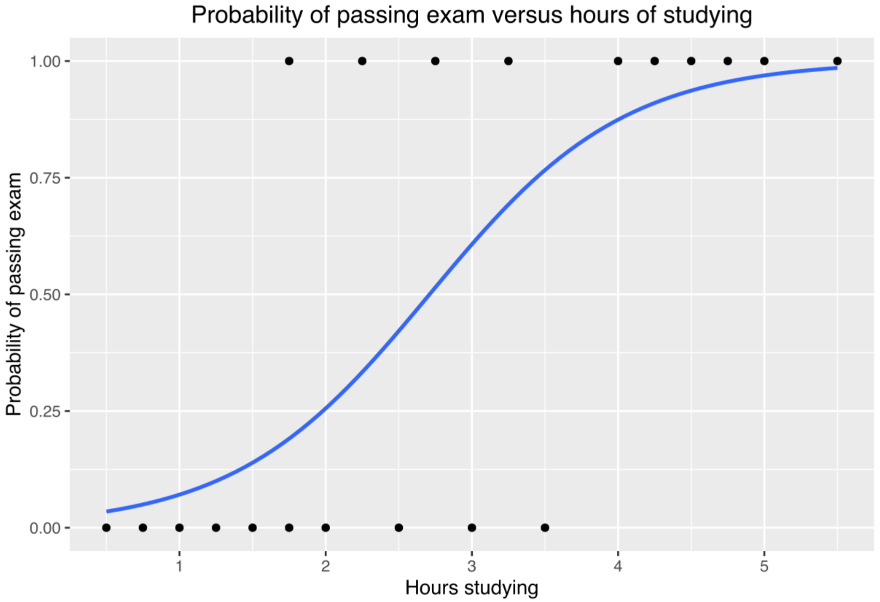
Theory

### Logistic ("Logit") Regression

- Models the probability of a binary outcome [0,1], with:
  - a linear model → $z = w * x + b$
  - a non-linear activation → $y = \frac{1}{1+e^{-z}}$
  - y = Predicted probability
  - x = Input data / features
  - w = Regression Weights
  - b = Constant

**Model params**

- Models a **sigmoid curve** (S-shape)
- Attempts to minimize the logistic loss (binary cross-entropy) **Metric**
- Suitable for linearly separable data
- Problems with complex, non-linear relationships or unbalanced data (true/false ratio)




Source: Canley, https://en.wikipedia.org/wiki/Logistic_regression, CC BY-SA 4.0

# Types of Machine Learning Models
## Regression

Practice

### Regression with scikit-learn

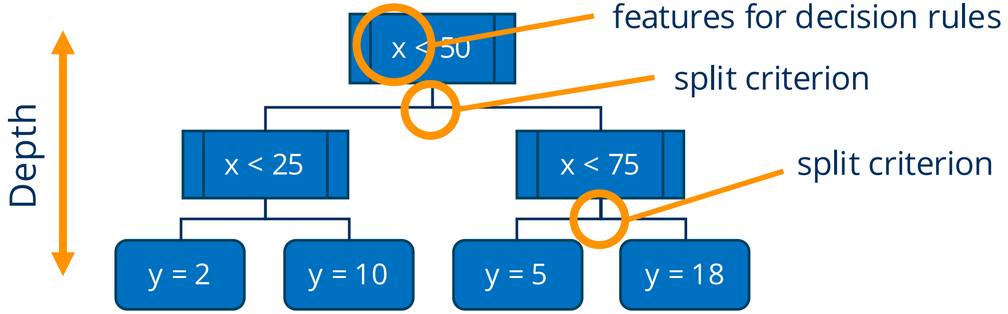
- Navigate to the training materials folder
- Use uv to start JuyperLab
- Open the notebook “day1.3_ml-basics/01_regression.ipynb”
- We will use the Python package scikit-learn and its modules:
  - sklearn.datasets: Tools for using common datasets for ML or for generating synthetic data
  - sklearn.model_selection: Tools for data splitting, parameter tuning, and more
  - sklearn.linear_model: Collection of linear models for regression and classification
  - sklearn.metrics: Collection of various metrics for model evaluation

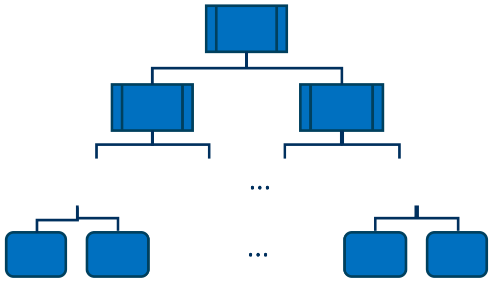
# Types of Machine Learning Models
## Decision Trees



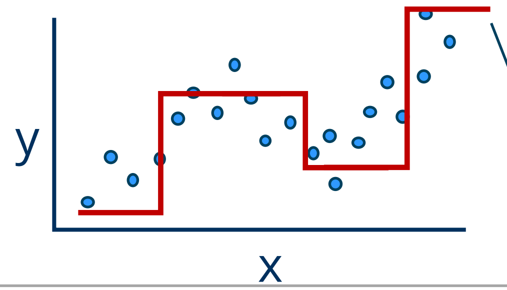
### Decision Tree

- Divides the feature space into regions using **decision rules** learned by optimizing a split criterion
- Can be used for regression, classification (Random Forests also for anomaly detection)
- Each region is assigned a constant prediction (mean value for regression, majority class for classification)
- Can capture non-linear relationships, while remaining interpretable

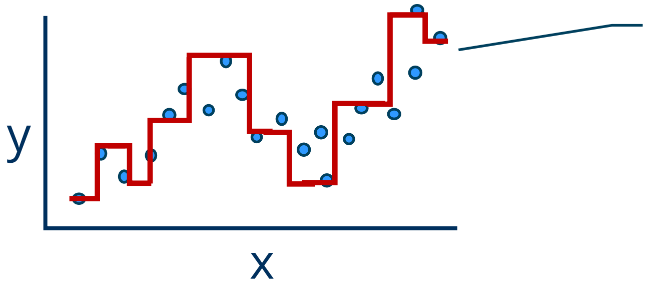



**Model params:**
- Max depth
- Split criterion
- Min samples for split
- Max leaf nodes
- ...




Piecewise constant approximation of the target

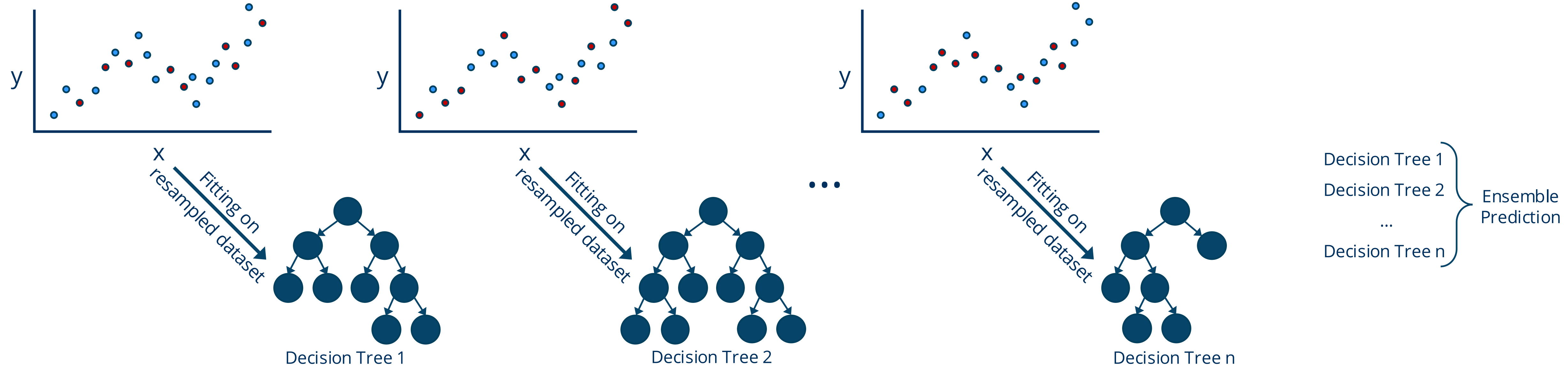



Decision tree with greater depth for closer fit to training data. But increases overfitting and reduces generalization.

# Types of Machine Learning Models
## Decision Trees

### Random Forest

- Ensemble Model consisting of *n* Decision Trees (DT)
- Each DT is trained on a randomly resampled dataset (samples may appear multiple times)
- Random subset of features considered at each split
- Prediction is obtained by averaging (regression) or majority vote (classification)
- Reduces variance and overfitting compared to a single DT



# Types of Machine Learning Models
## Decision Trees

Practice

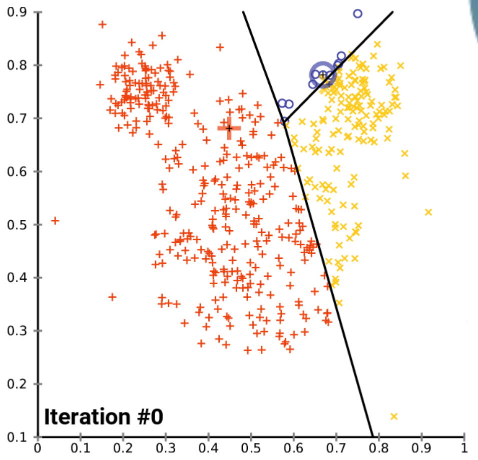
### Decision Trees with scikit-learn

- Navigate to the training materials folder
- Use uv to start JuyperLab
- Open the notebook "day1.3_ml-basics/02_random-forest.ipynb"
- We will use the Python package scikit-learn and its modules:
  - sklearn.datasets: Tools for using common datasets for ML or for generating synthetic data
  - sklearn.model_selection: Tools for data splitting, parameter tuning, and more
  - sklearn.tree: Collection of decision tree models for regression and classification
  - sklearn.ensemble: Collection of random forest models for regression and classification
  - sklearn.metrics: Collection of various metrics for model evaluation

# Types of Machine Learning Models
## Clustering

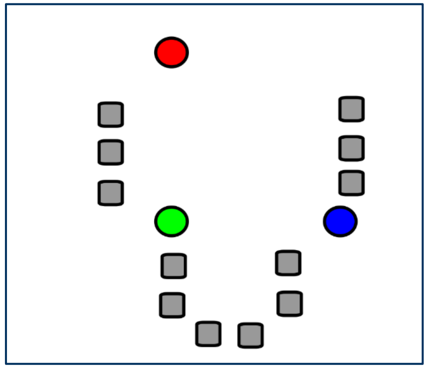
### k-Means Clustering

- Simple and fast clustering algorithm for Euclidean feature spaces
- Number of clusters k must be specified
- Expected to produce convex clusters of roughly spherical shape
- Sensitive to outliers and to feature scaling (reliance on Euclidean distance)
- Minimizes the sum of squared distances between points and their assigned cluster center

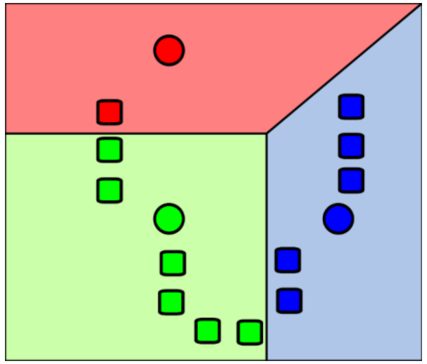



Source: Chire, https://de.wikipedia.org/wiki/K-Means-Algorithmus, CC BY-SA 4.0

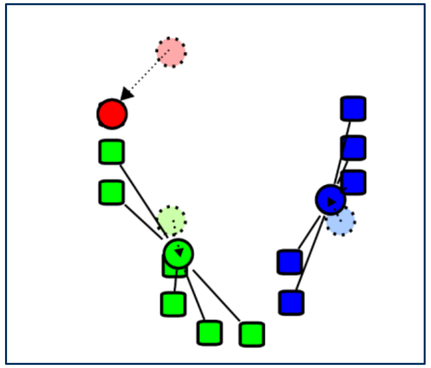
1: Initialization of cluster centers "centroids" (often random)



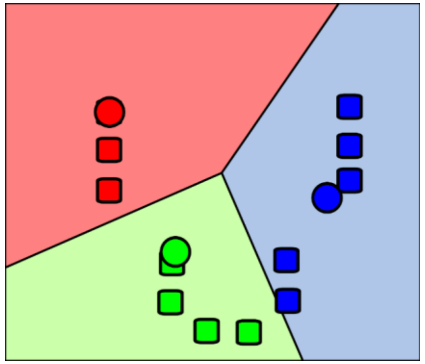
2: Assignment regions induced by centroids, assign point to nearest center



3: Recompute the centroids as mean of newly assigned points



4: Repeat steps 2 & 3 until convergence or a maximum number of iterations



Source: I, Weston.pace, https://de.wikipedia.org/wiki/K-Means-Algorithmus, CC BY-SA 3.0

# Types of Machine Learning Models
## Clustering

Practice

### Clustering with scikit-learn

- Navigate to the training materials folder
- Use uv to start JuyperLab
- Open the notebook “day1.3_ml-basics/03_clustering.ipynb”
- We will use the Python package scikit-learn and its modules:
  - sklearn.datasets: Tools for using common datasets for ML or for generating synthetic data
  - sklearn.model_selection: Tools for data splitting, parameter tuning, and more
  - sklearn.cluster: Collection of clustering models
  - sklearn.metrics: Collection of various metrics for model evaluation

# Types of Machine Learning Models
## Dimension Reduction

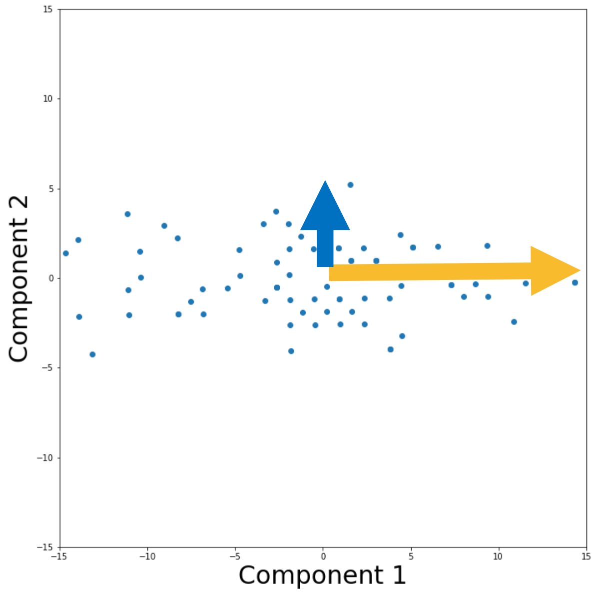
Theory

### Principal Component Analysis (PCA)

- Dimension reduction through a linear, orthogonal transformation of the data
- Projects data onto a lower-dimensional space spanned by directions of maximum variance
- Principal components are linear combinations of the original features
- PCA assumes linear relationship, sensitive to feature scaling, data should be centered and standardized
- Alternative approaches, especially for more complex non-linear relationships
  - t-SNE, UMAP, Autoencoder

| height | width | depth |
|---|---|---|
| 0.649060 | 0.213074 | 0.032167 |
| 0.983763 | 0.533933 | 0.026125 |
| 0.826448 | 0.223712 | 0.048805 |
| 0.610540 | 0.574425 | 0.116101 |
| 0.383580 | 0.042504 | 0.973645 |
| 0.222935 | 0.842952 | 0.152771 |
| 0.946367 | 0.780378 | 0.565486 |
| 0.580490 | 0.001958 | 0.945884 |
| 0.005322 | 0.019889 | 0.455281 |
| 0.359661 | 0.426161 | 0.369291 |

PCA(2) creates 2 principal components as linear combinations of height, width, and depth



# Types of Machine Learning Models
## Dimension Reduction

Practice

### Dimension Reduction with scikit-learn

- Navigate to the training materials folder
- Use uv to start JuyperLab
- Open the notebook “day1.3_ml-basics/04_dimension.ipynb”
- We will use the Python package scikit-learn and its modules:
  - sklearn.datasets: Tools for using common datasets for ML or for generating synthetic data
  - sklearn.decomposition: Collection of data transformation and decomposition methods
- We will use PCA to visualize the results of clustering high-dimensional data