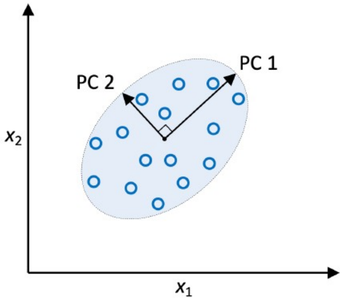
Machine learning

5

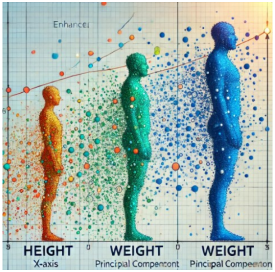
PCA

PCA is the most commonly used dimensionality reduction method and helps us to identify patterns and correlations in the original dataset to transform it into **a lower-dimension dataset with no loss of information**.



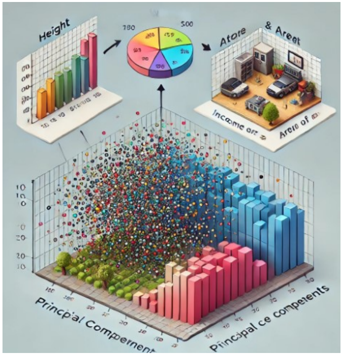
PCA example

- Let's assume that we have a dataset containing 100 observations (100 people) in 5 variables (e.g. height, weight, age, income, area of the apartment).
- It can be assumed that the variables **height** and **weight** will be strongly positively correlated with each other (since the taller someone is, the more he weighs).
- In order to achieve greater transparency of data or to avoid duplication of data, it is sometimes useful to replace two variables with one variable - called **component**, which can be called, for example, **size**.



PCA example

- Similarly, the variables **income and area of the apartment** will be correlated, which can perhaps be replaced by the factor **wealth**.



PCA

The main concept of PCA is the discovery of **unseen relationships and correlations among variables** in the original dataset.

Highly correlated attributes are so similar as to be redundant. Therefore, PCA removes such redundant attributes.

PCA reduces the dimensionality without affecting the significant information.

PCA

If we use PCA for dimensionality reduction, we construct a $d \times k$-dimensional transformation matrix, W, that allows us to map a vector of the features of the training example, x, onto a new k-dimensional feature subspace that has fewer dimensions than the original d-dimensional feature space.

Let us assume, that we have a feature vector, $x = [x_1, x_2, ..., x_d] \in \mathbb{R}^d$, which is transformed by transformation matrix $W \in \mathbb{R}^{d \times k}$ and $z = xW$, where $z = [z_1, z_2, ..., z_k] \in \mathbb{R}^k$.

PCA example

- Standardize the d-dimensional dataset.
- Construct the covariance matrix.
- Decompose the covariance matrix into its eigenvectors and eigenvalues.
- Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
- Select k eigenvectors, which correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace $k \leq d$.
- Construct a projection matrix, W, from the top k eigenvectors.
- Transform the d-dimensional input dataset, X, using the projection matrix, W, to obtain the new k-dimensional feature subspace.

the PCA equation can be written as follows

$$PC_j = w_{1j}x_1 + w_{2j}x_2 + ...w_{dj}x_d, j = 1, ..., k$$

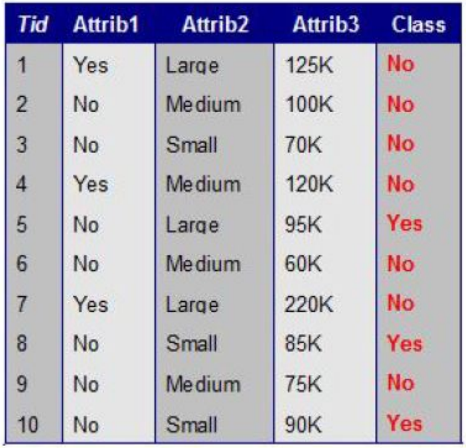
Prediction problems

Classification and **numerical prediction problems**, i.e. assigning an observation to a category or assigning to a numerical value.

$$x_n \rightarrow (\langle x_{n1}, ..., x_{nk} \rangle, y_n)$$

- customers who are likely to buy or not buy a particular product in a supermarket;
- customer status (current customer, past customer, noncustomer)
- people who are at high, medium or low risk of acquiring a certain illness;
- educational degree obtained (none, bachelor's, master's, doctorate);
- house prices;
- salaries of company employees;

Classification



Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

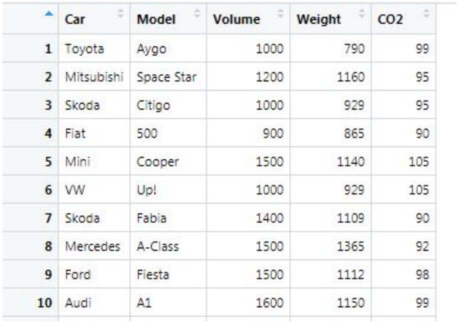
Classification

A **classifier** is an algorithm, that find the mapping function to map the observation x **input** to y **discrete output**.

The algorithms estimate discrete values - in other words, binary values such as 0 and 1, yes and no, true or false, based on a particular set of independent variables.

Regression

The regression algorithm's task is finding the mapping function. So we can map the input variable of x to the **continuous output variable of y**.



	Car	Model	Volume	Weight	CO2
1	Toyota	Aygo	1000	790	99
2	Mitsubishi	Space Star	1200	1160	95
3	Skoda	Citigo	1000	929	95
4	Fiat	500	900	865	90
5	Mini	Cooper	1500	1140	105
6	VW	Up!	1000	929	105
7	Skoda	Fabia	1400	1109	90
8	Mercedes	A-Class	1500	1365	92
9	Ford	Fiesta	1500	1112	98
10	Audi	A1	1600	1150	99

Popular algorithms for classification

- Base Classifiers
  - k-Nearest Neighbors
  - naive Bayes
  - decision trees
  - Support Vector Machine
  - neural networks
  - linear regression
  - logistic regression
- Ensemble Classifiers
  - random forest
  - boosting
  - bagging

Machine learning process

1. Data pre-processing
    - import the data
    - clean the data
    - split the data into training and test sets
2. Modelling
    - build the model
    - fit the model
    - make predictions
3. Evaluation
    - calculate performance metrics
    - make a verdict about the model, whether it's a good-fitting model and if it works for the data or not.

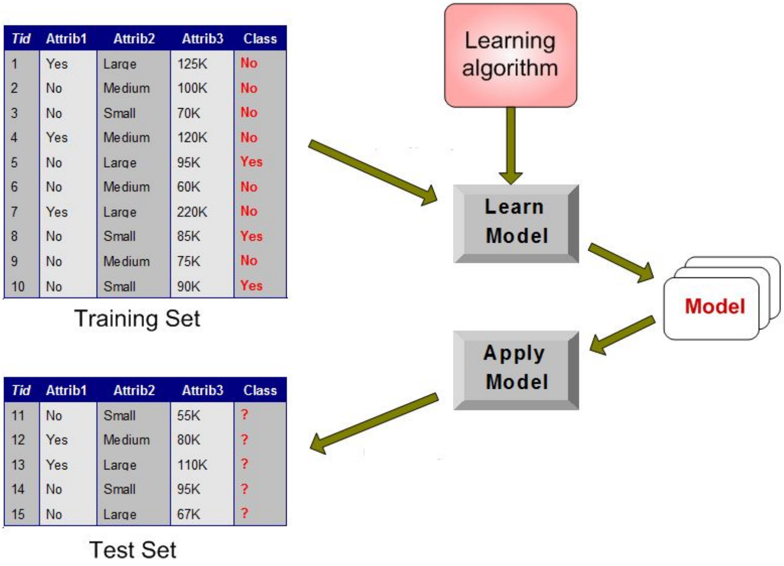
Splitting the dataset into a training set and a test set

1. **training set** - we'll use our training set to build the model. **(70% observations)** The model should observe and learn from the training set, optimizing any of its parameters.

2. **test set** - we'll use our testing set to evaluate the model. **(30% observations)** This set of data has the goal of ranking the model's accuracy and can help with model selection.

Our goal is for the split to have an exact representation of the full data set.

Machine learning process



Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Evaluation of models

The performance of a model (classifier) can be evaluated by comparing the predicted labels against the true labels of instances. This information can be summarized in a table called a **confusion matrix.**

Confusion matrix for a binary classification problem:

		Actual class	
		Class=1	**Class=0**
Predicted class	**Class=1**	**TP**	FP
	Class=0	FN	**TN**

Binary confusion matrix

$$\begin{array}{ccc|cc} & & & \multicolumn{2}{c}{\textit{actual}} \\ & & & 1 & 0 \\ \textit{predicted} & & 1 & \mathbf{TP} & \mathbf{FP} \\ & & 0 & \mathbf{FN} & \mathbf{TN} \end{array}$$

- **TP (True Positives)** – are when you predict an observation belongs to a class and it actually does belong to that class.;
- **FN (False Negatives)** – occur when you predict an observation does not belong to a class when in fact it does;
- **FP (False Positives)** – occur when you predict an observation belongs to a class when in reality it does not;
- **TN (True Negatives)** – are when you predict an observation does not belong to a class and it actually does not belong to that class.

Evaluation metric

		actual	
		1	0
predicted	1	**TP**	**FP**
	0	**FN**	**TN**

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

For binary clas.

$$ACC = \frac{TP + TN}{TN + FP + FN + TP}$$

Accuracy alone is not too informative regarding a model's effectiveness.

Evaluation metric

		actual	
		1	0
predicted	1	**TP**	**FP**
	0	**FN**	**TN**

Error rate is defined as the percentage of incorrect predictions for the test data. It can be calculated easily by dividing the number of incorrect predictions by the number of total predictions.

For binary clas.

$$error = \frac{FP + FN}{TN + FP + FN + TP}$$

Machine learning process

	age	brand	risk.of.a.crash
1	22	VW	high
2	25	VW	high
3	20	Fiat	low
4	34	Fiat	high
5	61	VW	low
6	19	Skoda	low
7	24	Opel	low
8	21	VW	high

classifier

$$\begin{aligned} \textbf{if}\ \ brand &= VW\ \textbf{and}\ age < 25 \\ \textbf{then} &= risk.of.a.crash = high \end{aligned}$$

New data:

John Smith	**23**	**VW**

$\Longrightarrow$

risk of a crash = high

Machine learning process

if *brand* = *VW* **and** *age* < 25 **then** *risk.of.a.crash* = *high*

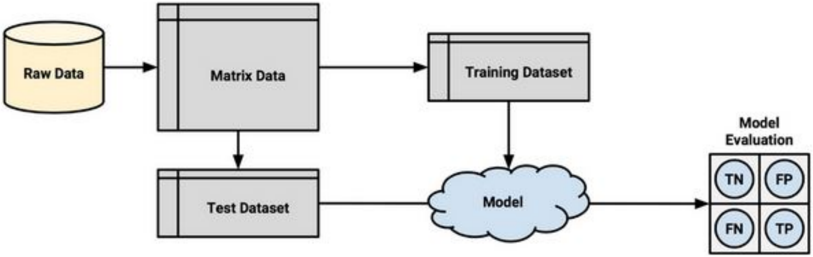
	age	brand	risk.of.a.crash
1	19	VW	high
2	19	Opel	high
3	55	Fiat	low
4	20	Skoda	high
5	22	VW	high
6	62	VW	low

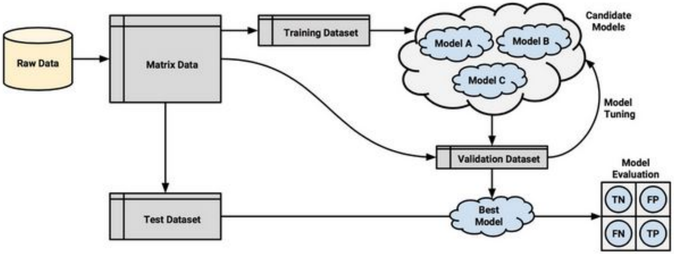
	risk.of.a.crash
1	high
2	low
3	low
4	low
5	high
6	low

		Risks from the test set	
		high	low
Risk from the classifier estimated	high	**2**	0
	low	2	**2**

$$ACC = \frac{TP+TN}{TN+FP+FN+TP} = \frac{4}{6}, \ error = \frac{FP+FN}{TN+FP+FN+TP} = \frac{2}{6}$$

Machine learning process





Properties of classification methods

- **accuracy** - refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data;
- **speed** - refers to the computational costs involved in generating and using the given classifier;
- **robustness** - ability of the classifier to make correct predictions given noisy data or data with missing values;
- **scalability** - ability to construct the classifier efficiently given large amounts of data;
- **interpretability** - refers to the level of understanding and insight that is provided by the classifier. Interpretability is subjective and therefore more difficult to assess.

k-nearest neighbors algorithm (k-NN)

kNN can be shortly described: **if it walks like a duck, looks like a duck, and talks like a duck, it is probably a duck.**

Classifier **kNN**

- belongs to the group of case study-based algorithms;
- the classification process is done online when there is a need to classify a new observation;

lazy learning methods

- **k-NN algorithm** assumes that the training dataset is not only a dataset for which we know the value of the target variable, but is at the same time a classification model.

k-nearest neighbors algorithm (k-NN) - idea

- kNN trains all samples and classifies new instances based on a dissimilarity (distance) measure;
- if kNN uses distance measures the data must be transformed. Most often this is a normalisation;
- for a new observation, we count the distances to all observations in the training set;
- In kNN, a new instance is classified to a label (class) that is common among the k-nearest neighbors. If $k = 1$, then the new instance is assigned to the class where its nearest neighbor belongs.

If we give a small k input, it may lead to overfitting. On the other hand, if we give a large k input, it may result in underfitting.

k-nearest neighbors algorithm (k-NN) - idea

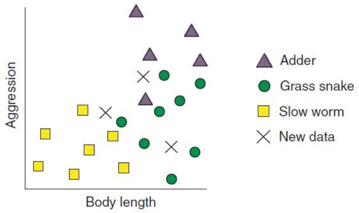
Let's introduce an example from the book Machine learning with R. This example is from the UK.

In the UK, as in Poland, we have a few native species of snakes

- **grass snake**
- **adder**
- **slow worm** - which is commonly mistaken for a snake.

k-nearest neighbors algorithm (k-NN) - idea

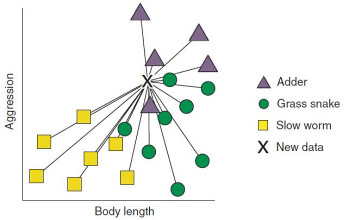
At the plot of data, we have body length and aggression of reptiles. Labeled cases for adders, grass snakes, and slow worms are indicated by their shape. New, unlabeled data are shown by black crosses.



k-nearest neighbors algorithm (k-NN) - idea

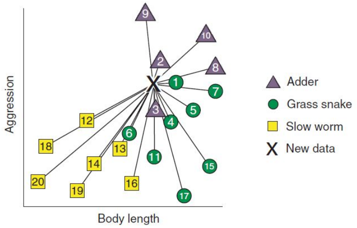
The first step of the kNN algorithm: **calculating distance** - the lines represent the distance between one of the unlabeled cases (the cross) and each of the labeled cases.

The kNN algorithm calculates the distance between each new, unlabeled case and all the labeled cases.



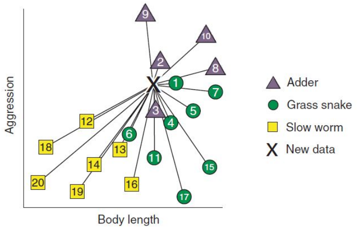
k-nearest neighbors algorithm (k-NN) - idea

The second step of the kNN algorithm: **ranking the neighbors**. The lines represent the distance between one of the unlabeled cases (the cross) and each of the labeled cases. The numbers represent the ranked distance between the unlabeled case (the cross) and each labeled case (1 = closest).



k-nearest neighbors algorithm (k-NN) - idea

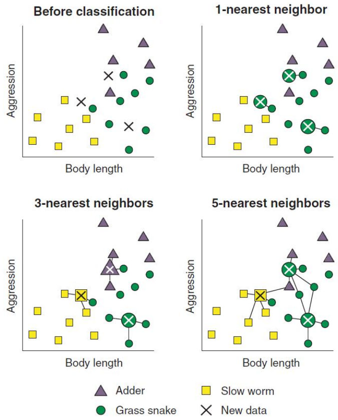
- The algorithm identifies the k-labeled cases (neighbors) nearest to each unlabeled case.
- **k is an integer specified by user.**
- in other words, find the k-labeled cases that are most similar in terms of their variables to the unlabeled case.
- finally, each of the k-nearest neighbor cases **votes** on which class the unlabeled data belongs in, based on the nearest neighbor's own class.



k-nearest neighbors algorithm (k-NN) - selecting k

- **small value of k** - then the classification is strongly influenced by outlying points or unusual observations (noise). E.g. for k=1 the algorithm will return the closest observation, and so may produce an **overfitting** of the set, at the expense of generalisability;
- **selecting too large a k** - may result in an interesting local behaviour of the data being omitted.**underfitting**;

k-nearest neighbors algorithm (k-NN) - idea



The final step of the kNN algorithm: **identifying the k-nearest neighbors and taking the majority vote.** Lines connect the unlabeled data with their one, three, and five nearest neighbors. The majority vote in each scenario is indicated by the shape drawn under each cross.

What happens if the vote is tied?

- in a two-class classification problem (when the data can only belong to one of two, mutually exclusive groups) is to ensure that we pick **odd numbers of k**. This way, there will always be a deciding vote.
- we have more than two groups - is to decrease k until a majority vote can be won.

Strengths and weaknesses of this algorithm

Advantages

- simple;
- makes no assumptions about the underlying data distribution;

Disadvantages

- does not produce a model, limiting the ability to understand how the features are related to the class;
- requires selection of an appropriate k;
- slow classification phase;
- nominal features and missing data require additional processing.

Thank you for your attention!!!