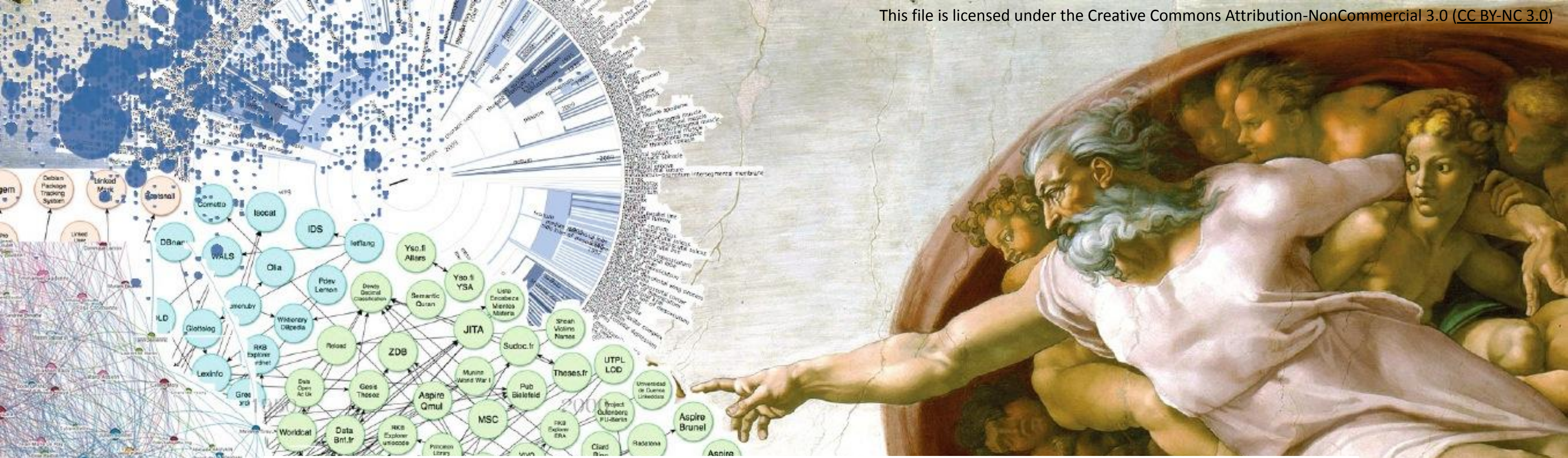


Information Service Engineering

Lecture 11: Basic Machine Learning - 2

Karlsruher Institut für Technologie

FIZ Karlsruhe
Leibniz Institute for Information Infrastructure

Prof. Dr. Harald Sack
FIZ Karlsruhe - Leibniz Institute for Information Infrastructure
AIFB - Karlsruhe Institute of Technology

Summer Semester 2021

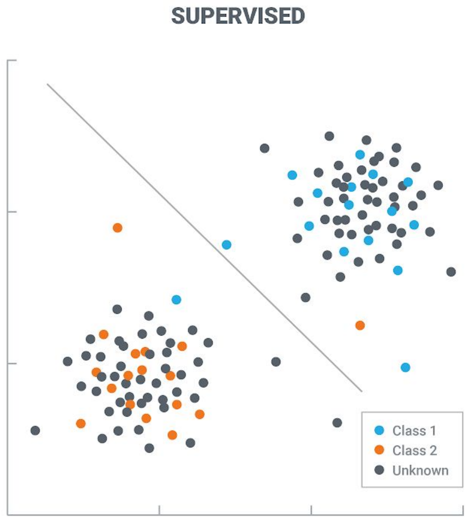
Last Lecture: Basic Machine Learning - 1

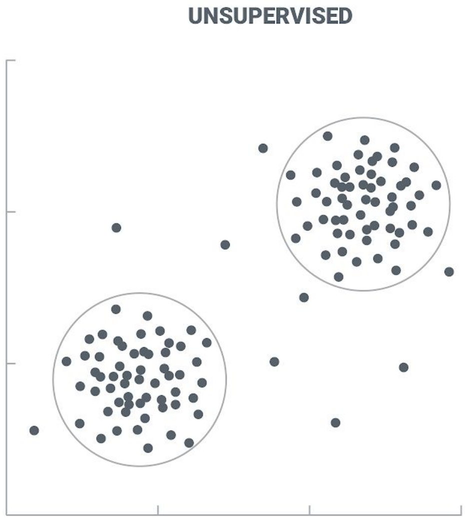
- AI and Machine Learning
- Machine Learning Applications
- Classification vs. Regression
- Supervised vs. Unsupervised Learning
- Machine Learning Challenges
- Overfitting
- Data Cleaning
- Sampling
- Cross Fold Validation

Supervised and Unsupervised Learning

- In **supervised learning** aka **predictive analytics**, data consists of observations $(x_i, y_i),\ x_i \in \mathbb{R}^p,\ i = 1, \ldots, n$
- Such data is called **labeled data**, and the y_i are thought of as the labels for the data.

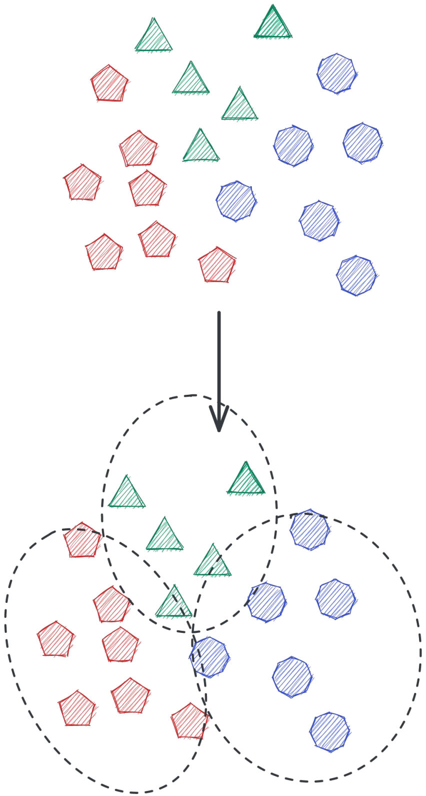
- In **unsupervised learning**, we just look at data $x_i \in \mathbb{R}^p,\ i = 1, \ldots, n$ to detect patterns.
- This is called **unlabeled data**.
- *Even if we have labels y_i, we may still wish to temporarily ignore the y_i and conduct unsupervised learning on the inputs x_i.*





Examples for Unsupervised Learning Tasks

- Identify similar groups of **online shoppers** based on their browsing and purchasing history.
- Identify similar groups of **music listeners** or **movie viewers** based on their ratings or recent listening/viewing patterns.
- Identify similar groups of patients based on their medical records.
- Determine how to **place sensors, broadcasting towers, law enforcement, or emergency-care centers** to guarantee that desired coverage criteria are met.



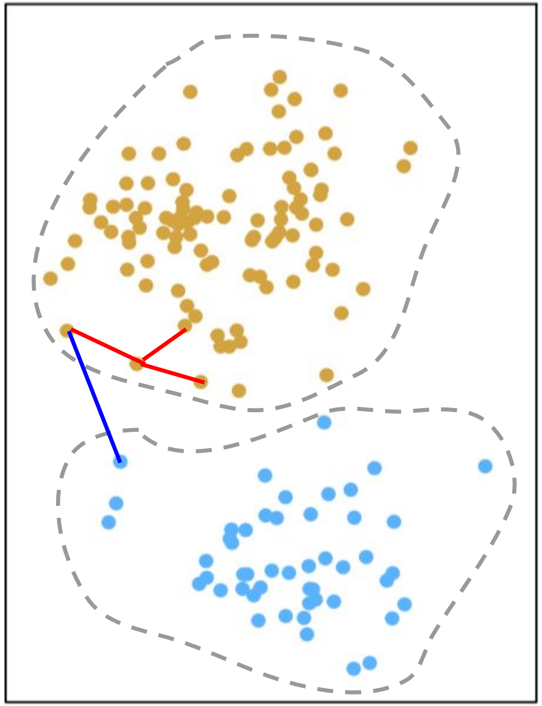
Unsupervised Learning

- If there are no labels, how do we know if results are meaningful?
  - Experts might interpret the result (**external evaluation**).

- However, we need **Unsupervised Learning**, because:
  - Labeling large datasets is very costly.
  - We may have no idea what/how many classes there are (data mining).
  - We may want to use **clustering** to gain some insight into the structure of the data before designing a classifier.

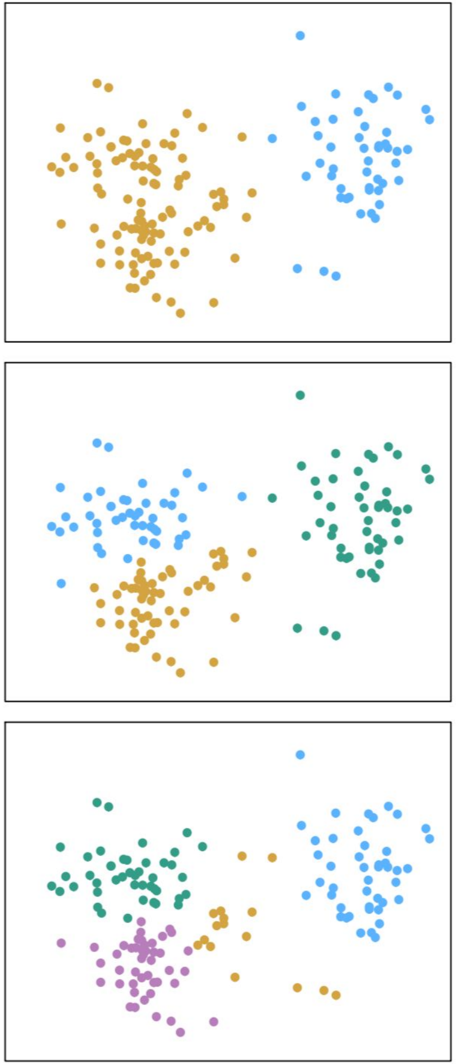
Clustering

- What is a good clustering?
  - **Internal distances** (within the cluster) should be **small**.
  - **External distances** (inter-cluster) should be **large**.
- Clustering is a way to discover new categories.



Clustering

- A **clustering** is a partition $\{C_1, \ldots, C_K\}$, where each C_k denotes a subset of the observations.
- Each observation belongs to one and only one of the clusters.
- To denote that the i^{th} observation is in the k^{th} cluster, we write $i \in C_k$.

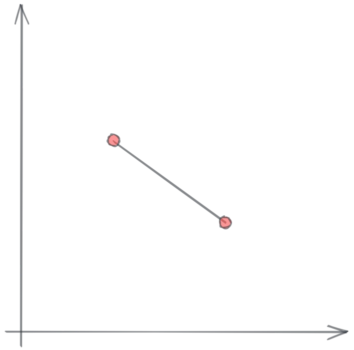


Clustering - What does 'Similarity' mean?

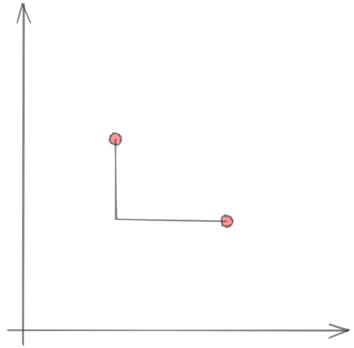
- To start with, we need a **proximity measure:**
  - **Similarity measure**
    $s(x_i,x_k)$: large, if x_i and x_k are similar
  - **Dissimilarity measure** (distance)
    $d(x_i,x_k)$: small, if x_i and x_k are similar

Distance Measures

- **Euclidean distance:** $d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_i^{(k)} - x_j^{(k)})^2}$
  - translation invariant



- **Manhattan distance:** $d(x_i, x_j) = \sum_{k=1}^{d} |x_i^{(k)} - x_j^{(k)}|$
  - less complex to compute



k-Means Clustering

- **Main idea:**
  A good clustering is one for which the **within-cluster variation** is as small as possible.
- The **within-cluster variation** **$WCV(C_k)$** for cluster C_k is some measure of the amount by which the observations within each class differ from one another.
- **Goal**: Find $C_1, \ldots, C_K$ that minimize $\sum_{k=1}^{K} WCV(C_k)$.

"Partition the observations into K clusters such that the WCV summed up over all K clusters is as small as possible."

k-Means Clustering

- **Determine within-cluster variation**
  - **Goal**: Find $C_1, \ldots, C_K$ that minimize $\sum_{k=1}^{K} WCV(C_k)$.
  - Use Euclidean distance: $$WCV(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$ where $|C_k|$ denotes the number of observations in cluster k.
  - The total number of clusters K is a fixed parameter.

k-Means Clustering

- WCV(C_k) can be rewritten:

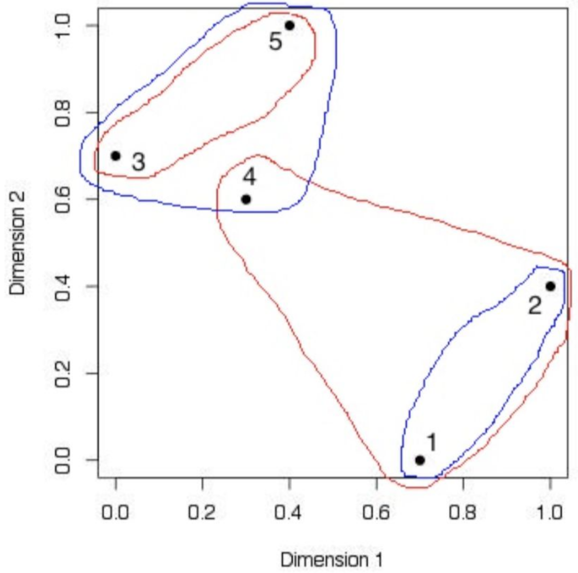
$$WCV(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \|(x_i - x_{i'})\|_2^2$$
$$= 2 \sum_{i \in C_k} \|x_i - \bar{x}_k\|^2$$

  - with $\bar{x}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$ is just the **average of all the points in Cluster C_k** (**Cluster Centroid** or **Cluster Center**).

k-Means Clustering

- **Simple Example**
  - n=5 and K =2
  - distance matrix

	1	2	3	4	5
1	0	0.25	0.98	0.52	1.09
2	0.25	0	1.09	0.53	0.72
3	0.98	1.09	0	0.10	0.25
4	0.52	0.53	0.10	0	0.17
5	1.09	0.72	0.25	0.17	0



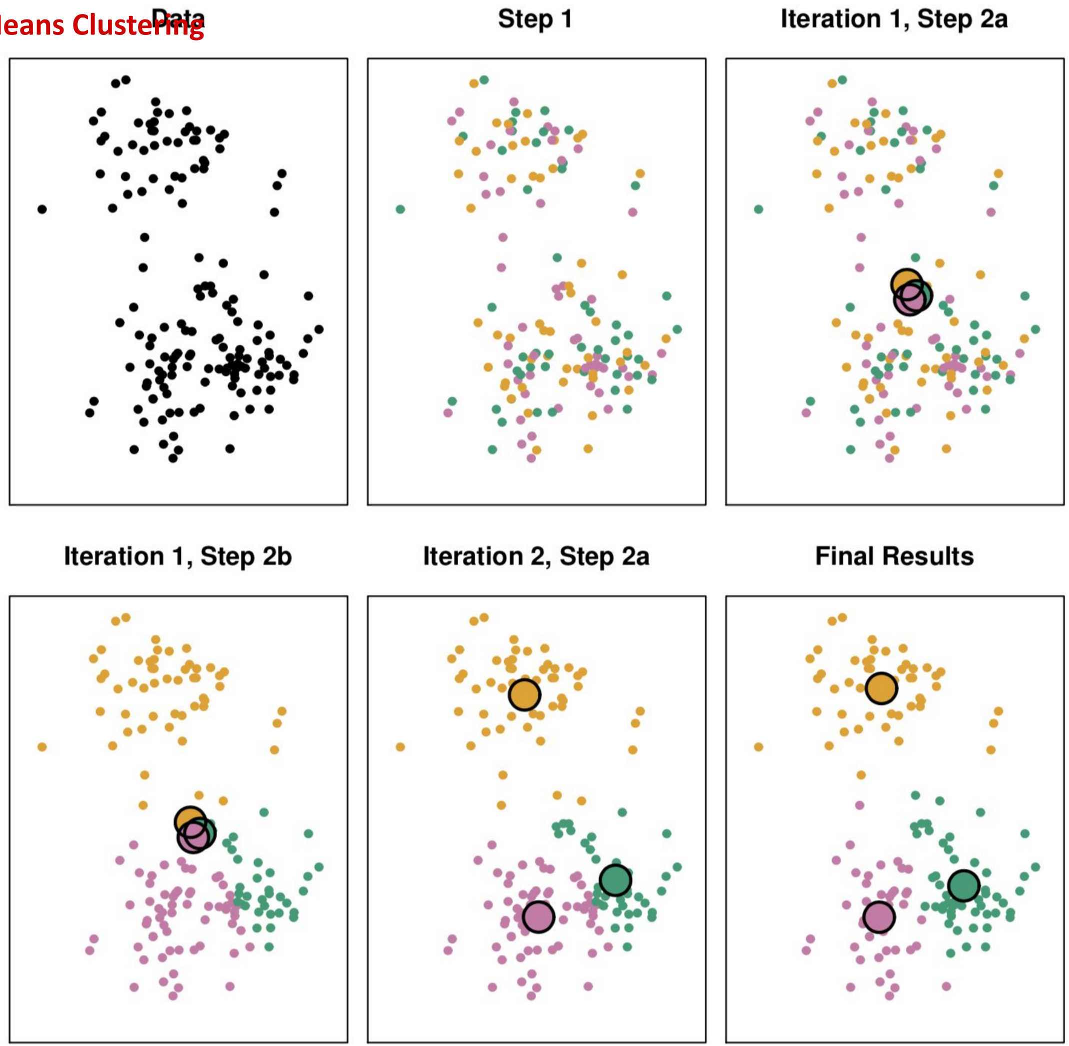
  - Red clustering: $\sum \mathrm{WCV}(C_k) = (0.25 + 0.53 + 0.52)/3 + 0.25/2 = 0.56$
  - Blue clustering: $\sum \mathrm{WCV}(C_k) = 0.25/2 + (0.10 + 0.17 + 0.25)/3 = 0.30$

k-Means Algorithm

1. Start by **randomly** partitioning the observations into **K clusters**.
2. Until the clusters stop changing, repeat:
   a. For each cluster, compute the **cluster centroid**.
   b. Assign each observation to the cluster whose centroid is the closest.

k-Means Algorithm

- Example: K=3



k-Means Summary

Karlsruher Institut für Technologie

FIZ Karlsruhe
Leibniz Institute for Information Infrastructure

- It is infeasible to actually optimize $WCV(C_k)$ in practice, but K-means provides a so-called **local optimum** of this objective.
- The achieved result depends both on K, and also on the random initialization.
- It is a good idea to try different random starts and pick the best result among them.
- There is a method called **K-means++** that improves how the clusters are initialized.

k-Means Clustering Hands On

11 - ISE2021 - k-Means Example.ipynb

File Edit View Insert Runtime Tools Help Last saved at 09:41

Comment Share

+ Code + Text

RAM Disk Editing

Basic Machine Learning - k-Means Examples

This is the colab notebook example for lecture 11 Basic Machine Learning 2, chapter 4.5 Basic ML Algorithms 1 - k-Means Clustering, of AIFB/KIT Lecture "Information Service Engineering" Summer Semester 2021.

In this colab notebook you will learn how to make use of the SciKit Learn library for applying machine learning algorithms, in particular the k-means clustering, mathplotlib to draw diagrams, numpy to manage multi-dimensional arrays.

Please make a copy of this notebook to try out your own adaptions via "File -> Save Copy in Drive"

k-Means Clustering

Clustering algorithms seek to learn, from the properties of the data, an optimal division or discrete labeling of groups of points.

Many clustering algorithms are available in **Scikit-Learn**. This colab notebook will give you an example how to use *k-means clustering*, which is implemented in `sklearn.cluster.KMeans`.

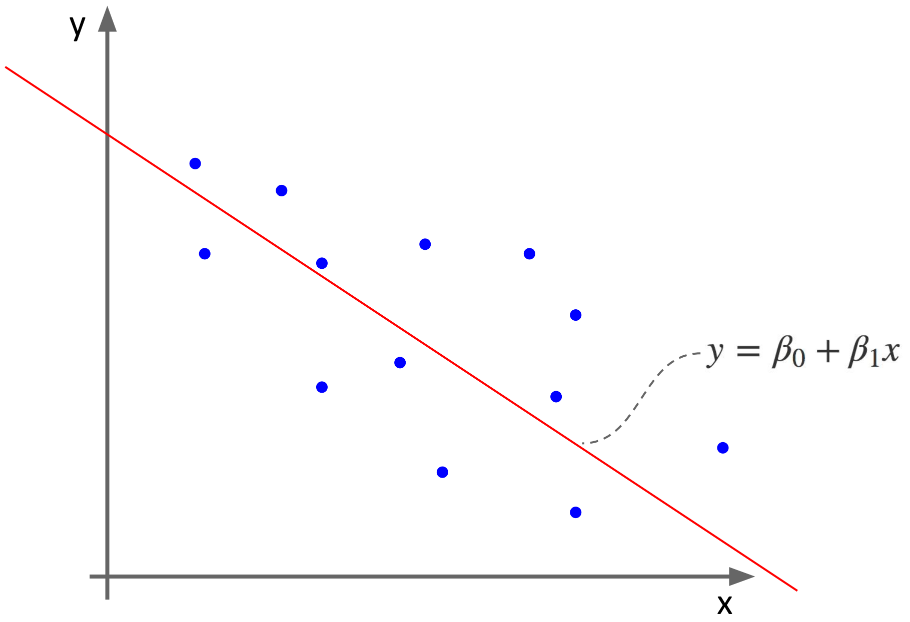
As usual we start with importing required libraries.

```
[ ]  %matplotlib inline
     import matplotlib.pyplot as plt
     import seaborn as sns; sns.set()  # for plot styling
     import numpy as np
```

k-Means Clustering Python Colab Notebook

Linear Regression

- With **Linear Regression** we aim to fit a line to a scattering of data.

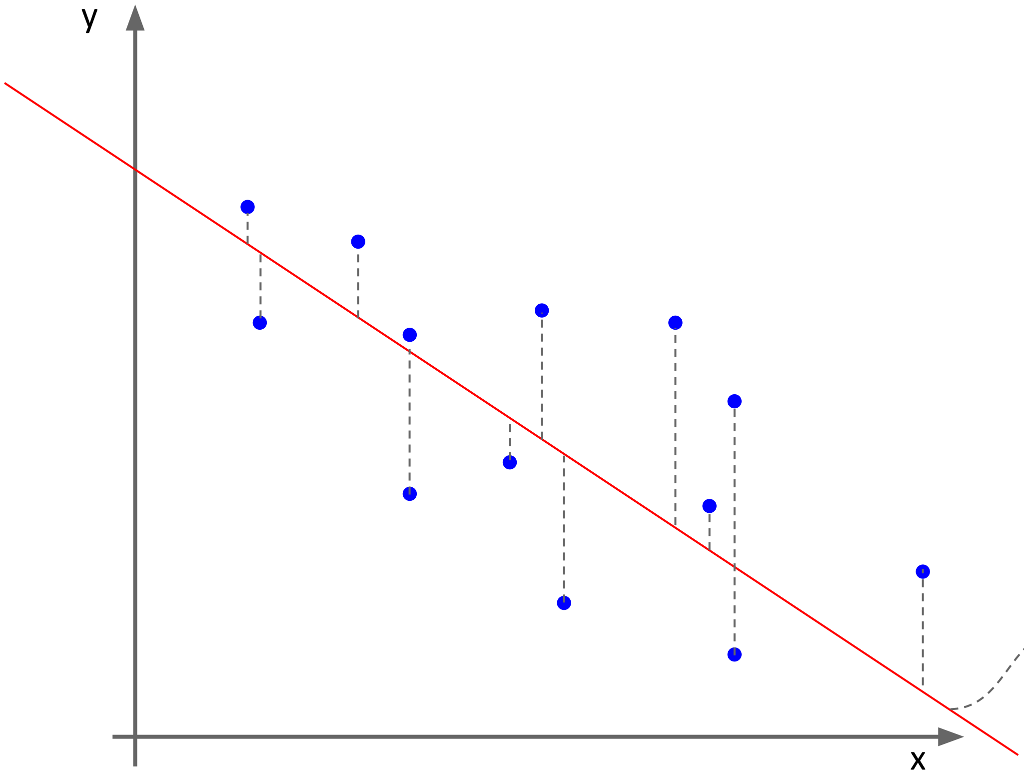


Linear Regression

- Notation:
  - **Input** vector $x \in \mathbb{R}^N$
  - **Output** vector $y \in \mathbb{R}$
  - Parameters $\beta = (\beta_0, \beta_1, \dots, \beta_N)^\top \in \mathbb{R}^{N+1}$
  - **Linear** model: $f(x) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j$

- Given **training data** $D = \{(x_i, y_i)\}_{i=1}^{P}$
- We define the **least squares costs** (or "loss") $L^{ls}(\beta) = \sum_{i=1}^{P} (\underbrace{y_i}_{\text{observed result}} - \underbrace{f(x_i)}_{\text{predicted result}})^2$

Linear Regression



$$f(x) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j$$

- Try to find the line (hyperplane) that fits best to the given data by **minimizing its distance between the data and the line**.
- The **Least Squares Cost ($L^{ls}(\beta)$)** framework aims at recovering the line (hyperplane) that **minimized** the total squared length of the error.

$$L^{ls}(\beta) = \sum_{i=1}^{P} (y_i - f(x_i))^2$$

Linear Regression

- **Find optimal parameters $\boldsymbol{\beta}$**
  - Augment input vector with a 1 in front: $\mathbf{x} = (1, x) = (1, x_1, x_2, \ldots, x_N)^\top \in \mathbb{R}^{N+1}$
    $\beta = (\beta_0, \beta_1, \ldots, \beta_N)^\top \in \mathbb{R}^{N+1}$
  - Simplified linear model: $f(x) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j = \mathbf{x}^\top \beta$
  - Rewrite LSC: $L^{ls}(\beta) = \sum_{i=1}^{P} (y_i - \mathbf{x}^\top \beta)^2 = \|y - \mathbf{X}\beta\|^2$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_P^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,N} \\ \vdots & & & & \vdots \\ 1 & x_{1P,1} & x_{P,2} & \ldots & x_{P,N} \end{pmatrix}, \; y = \begin{pmatrix} y_1 \\ \vdots \\ y_P \end{pmatrix}$$

Linear Regression

- **Find optimal parameters β**
  - Rewrite LSC:

$$L^{ls}(\beta) = \sum_{i=1}^{P} (y_i - \mathbf{x}^\top \beta)^2 = \|y - \mathbf{X}\beta\|^2$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_P^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,N} \\ \vdots & & & & \vdots \\ 1 & x_{1P,1} & x_{P,2} & \dots & x_{P,N} \end{pmatrix}, \; y = \begin{pmatrix} y_1 \\ \vdots \\ y_P \end{pmatrix}$$

  - Compute optimum by setting the gradient to zero:

$$0_P^\top = \frac{\partial L^{is}(\beta)}{\partial \beta} = -2(y - \mathbf{X}\beta)^\top \mathbf{X}$$

$$\Leftrightarrow 0_P = \mathbf{X}^\top \mathbf{X}\beta - \mathbf{X}^\top y$$

$$\Leftrightarrow \hat{\beta}^{ls} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top y$$

via chain rule:

$$f(x) = u(v(x)) \Rightarrow f'(x) = u'(v(x))v'(x)$$

Example - Berlin Climate Data

Climate Dataset

● **Date**	year-mm-dd
● **AverageTemperature**	average surface temperature
● **AverageTemperatureUncertainty**	uncertainty of measurement
● **City**	city of measurement
● **Country**	country of measurement
● **Latitude**	latitude
● **Longitude**	longitude

New:

● **Year**	extracted from date
● **12MonthAvgTemperature**	12 month average of the temperature

Berlin Average Temperatures 1753-2013

http://bit.do/eZUHz

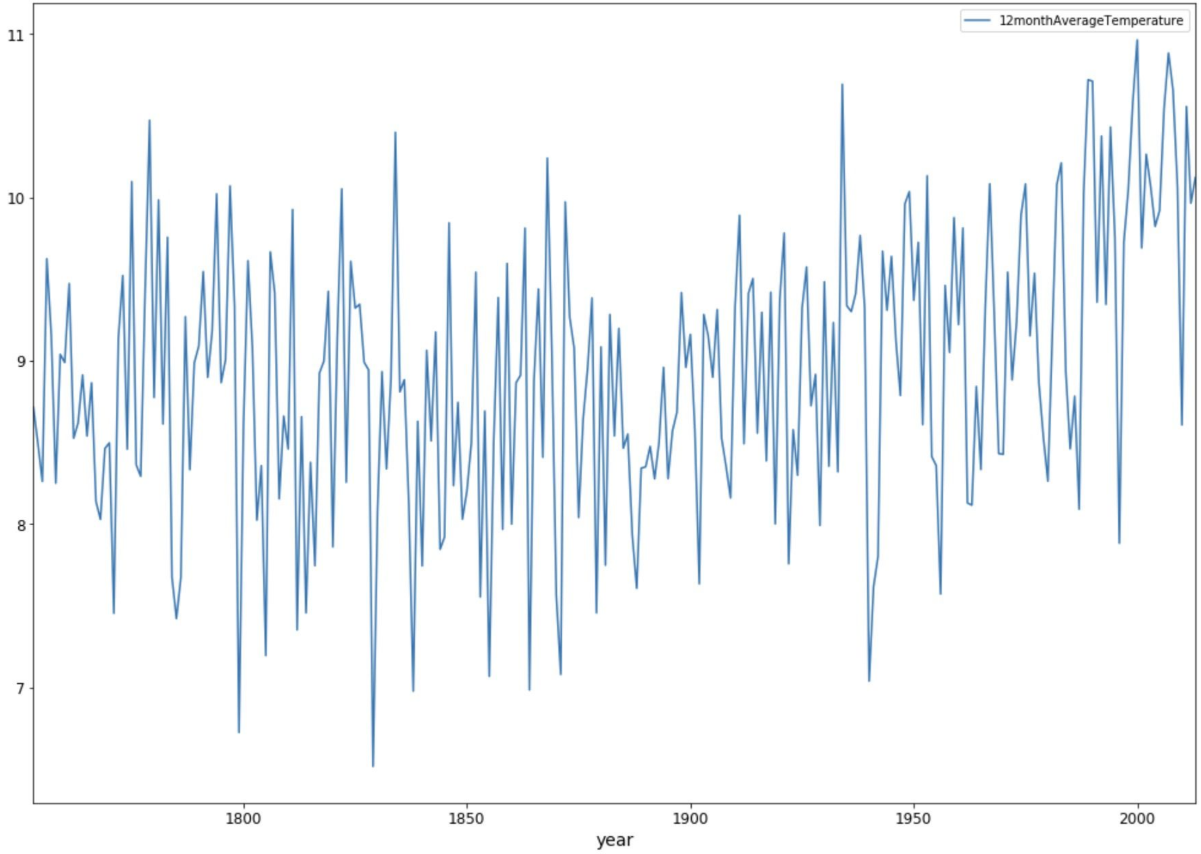
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

100% $ % .0 .00 123 Arial 10 B I S A

	A	B	C	D	E	F	G	H	I	J	K
1	**date**	**year**	**12monthAverag**	**AverageTemper**	**AverageTemper**	**City**	**Country**	**Latitude**	**Longitude**		
2	1753-01-01	1753	8.715666667	-2.452	7.998	Berlin	Germany	52.24N	13.14E		
3	1754-01-01	1754	8.492833333	-1.341	2.212	Berlin	Germany	52.24N	13.14E		
4	1755-01-01	1755	8.261083333	-6.032	6.072	Berlin	Germany	52.24N	13.14E		
5	1756-01-01	1756	9.624833333	2.619	6.38	Berlin	Germany	52.24N	13.14E		
6	1757-01-01	1757	9.153666667	-1.883	4.262	Berlin	Germany	52.24N	13.14E		
7	1758-01-01	1758	8.250833333	-4.598	1.266	Berlin	Germany	52.24N	13.14E		
8	1759-01-01	1759	9.03925	2.143	7.459	Berlin	Germany	52.24N	13.14E		
9	1760-01-01	1760	8.989166667	-2.831	5.388	Berlin	Germany	52.24N	13.14E		
10	1761-01-01	1761	9.47275	-0.742	8.045	Berlin	Germany	52.24N	13.14E		
11	1762-01-01	1762	8.52625	1.105	3.177	Berlin	Germany	52.24N	13.14E		
12	1763-01-01	1763	8.619916667	-5.068	6.828	Berlin	Germany	52.24N	13.14E		
13	1764-01-01	1764	8.913666667	1.933	5.876	Berlin	Germany	52.24N	13.14E		
14	1765-01-01	1765	8.539666667	0.679	11.287	Berlin	Germany	52.24N	13.14E		
15	1766-01-01	1766	8.865166667	-2.678	7.946	Berlin	Germany	52.24N	13.14E		
16	1767-01-01	1767	8.138416667	-8.253	7.235	Berlin	Germany	52.24N	13.14E		
17	1768-01-01	1768	8.02925	-5.705	13.971	Berlin	Germany	52.24N	13.14E		
18	1769-01-01	1769	8.463583333	0.72	7.133	Berlin	Germany	52.24N	13.14E		
19	1770-01-01	1770	8.4975	-1.689	8.09	Berlin	Germany	52.24N	13.14E		
20	1771-01-01	1771	7.454	-2.867	8.727	Berlin	Germany	52.24N	13.14E		
21	1772-01-01	1772	9.13475	-0.944	9.804	Berlin	Germany	52.24N	13.14E		
22	1773-01-01	1773	9.522083333	1.64	2.79	Berlin	Germany	52.24N	13.14E		
23	1774-01-01	1774	8.458666667	-1.993	9.485	Berlin	Germany	52.24N	13.14E		
24	1775-01-01	1775	10.09666667	-0.523	5.569	Berlin	Germany	52.24N	13.14E		
25	1776-01-01	1776	8.363	-8.336	6.452	Berlin	Germany	52.24N	13.14E		

Example - Berlin Climate Data

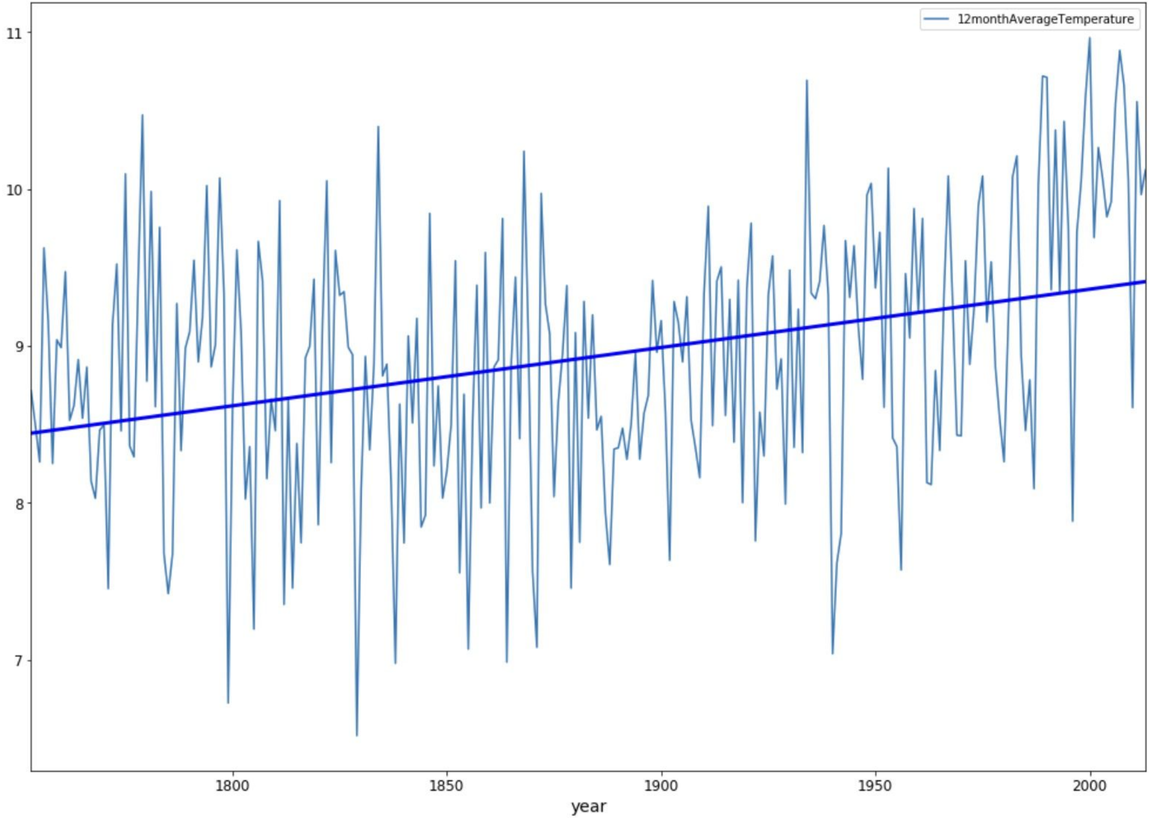
- There might be a correlation between the **year** of a measurement and the **temperature**.



Example Berlin Climate Data

- The linear regression prediction might underfit the data.
- There might be a **non linear correlation**.
- The **temperature** might also be dependent on other factors, such as e.g. **latitude**, etc.

RMSE(linear): 0.7995



Root Mean Square Error (RMSE)

- A popular measure for the achieved quality of the prediction is the **Root Mean Square Error RMSE**:

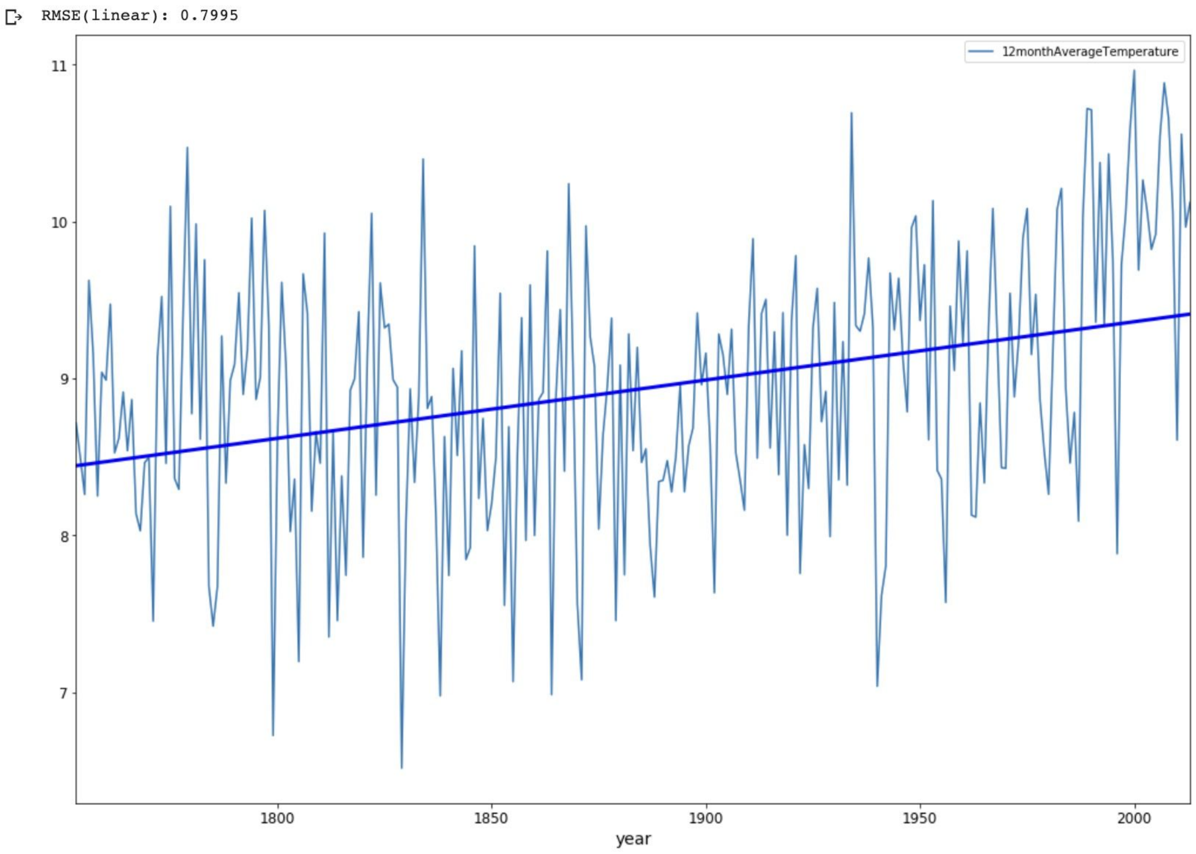
$$RMSE(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h(\mathbf{x}_i^\top) - y_i\right)^2}$$

with
m … number of dataset instances
$\boldsymbol{X}_i^T$ … feature vector for ith instance
y_i … label (desired output) of ith instance
h … hypothesis (prediction function)

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_P^\top \end{pmatrix}$$

Example Berlin Climate Data

- The linear regression prediction might underfit the data.
- There might be a **non linear correlation.**
- The **temperature** might also be dependent on other factors, such as e.g. **latitude**, etc.
- **RMSE=0.7995**



Non Linear (Polynomial) Regression

Karlsruher Institut für Technologie

FIZ Karlsruhe
Leibniz Institute for Information Infrastructure

- Replace the inputs $x_i \in \mathbb{R}^d$ by some **non-linear features** $\phi(x_i) \in \mathbb{R}^k$

- The optimal β is the same $\hat{\beta}^{ls} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top y$ but with $\mathbf{X} = \begin{pmatrix} \phi(x)_1^\top \\ \vdots \\ \phi(x)_P^\top \end{pmatrix} \in \mathbb{R}^{n \times k}$

- **What are "features"?**

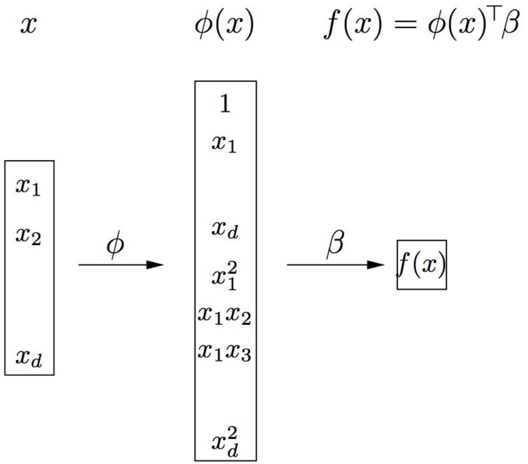
  a) Features are an arbitrary set of basis functions.

  b) Any function linear in β can be written as $f(x) = \phi(x)^\top \beta$

  for some $\boldsymbol{\phi}$ - which we denote as "**features**".

Non Linear (Polynomial) Regression

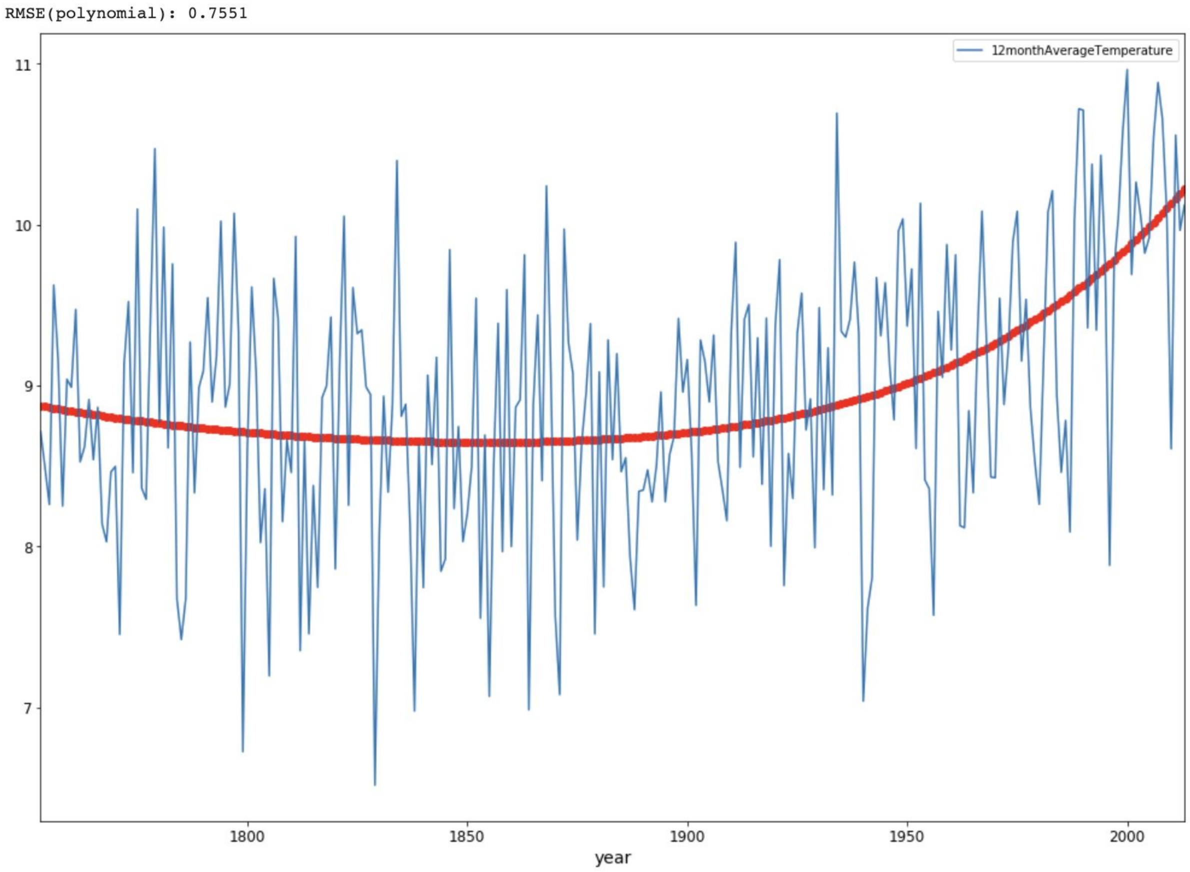
- Linear features: $\phi(x) = (1, x_1, \dots, x_N) \in \mathbb{R}^{1+N}$
- Quadratic features $\phi(x) = (1, x_1, \dots, x_N, x_1^2, x_1 x_2, x_1 x_3, \dots, x_n^2) \in \mathbb{R}^{1+N+\frac{N(N+1)}{2}}$



Example - Berlin Climate Data

- We might also try a **non linear (polynomial) regression.**

- Here degree=8,
  RMSE = 0.7551



Berlin Climate Change Regression in a Notebook

11 - ISE 2021 - ClimateData Regression HandsOn

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editi

Basic Machine Learning - Linear Regression Examples

This is the colab notebook example for lecture 11 Basic Machine Learning 2, chapter 4.6 Basic ML Algorithms 2 - Linear Regression, of AIFB/KIT Lecture "Information Service Engineering" Summer Semester 2021.

In this colab notebook you will learn how to make use of the SciKit Learn library for applying machine learning algorithms, in particular linear and polynomial regression, mathplotlib for data visualization, pandas for data analysis, numpy to manage multi-dimensional arrays.

Please make a copy of this notebook to try out your own adaptions via "File -> Save Copy in Drive"

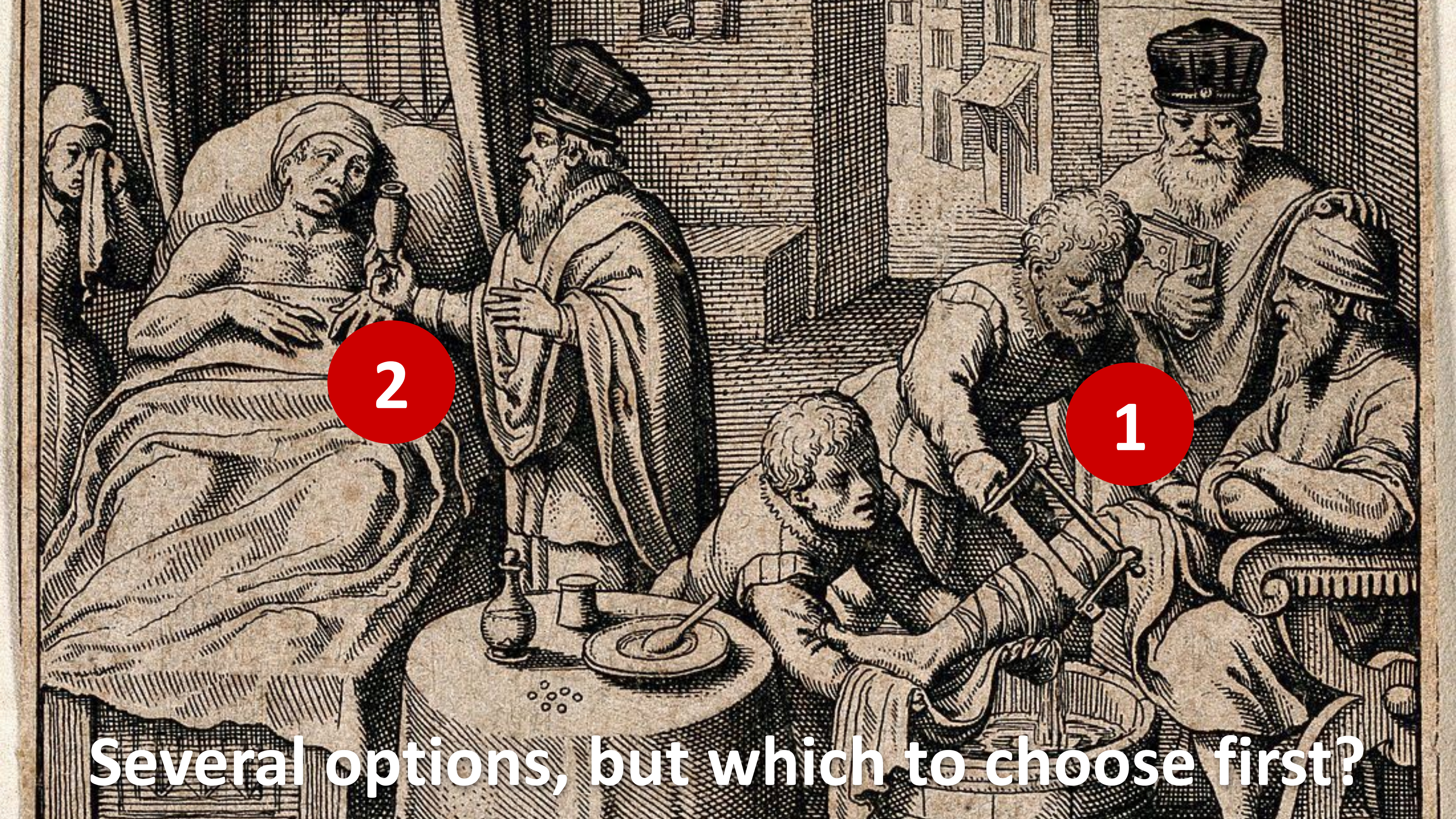
Set Up

```
[ ]  # Common imports
     import numpy as np
     import pandas as pd

     # To plot pretty figures
     %matplotlib inline
     import matplotlib
     import matplotlib.pyplot as plt
     plt.rcParams['axes.labelsize'] = 14
     plt.rcParams['xtick.labelsize'] = 12
     plt.rcParams['ytick.labelsize'] = 12
```

Get the Data

Linear Regression Python Colab Notebook

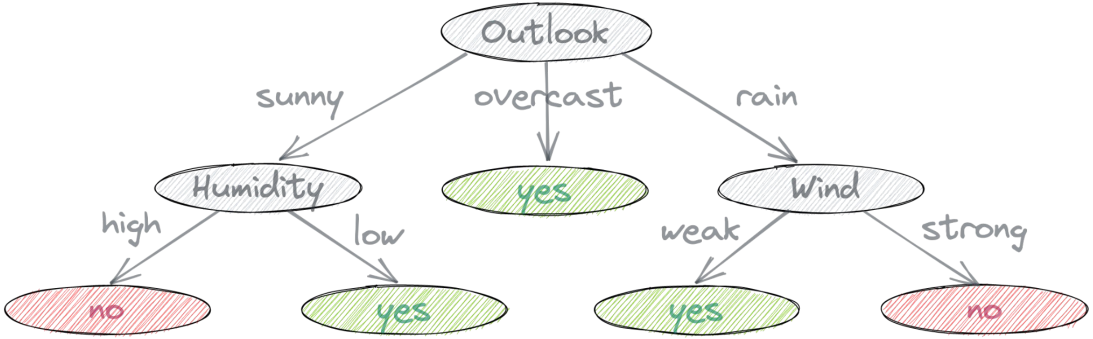

2
1
Several options, but which to choose first?

Decision Trees

- In many classification or regression problems,
  we expect **all attribute values present at the same time**.
- However…
  - sometimes **new attributes occur only over time**,
  - sometimes it is more efficient,
    first to check the most promising attributes only.

Should We Play Tennis?

- Depends on
  - Outlook
  - Wind
  - Temperature
  - Humidity
  - …



- Internal node: test attribute X_i
- Branch: selects value for X_i
- Leaf node: predict Y

- Given: a set of possible instances: **<Humidity=low, Wind=weak, Outlook=rain, Temp=high>**

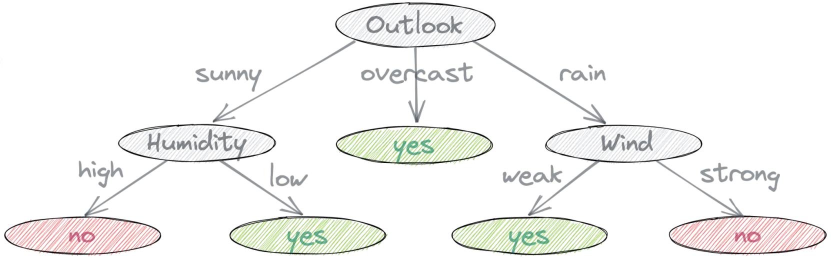
Decision Tree Learning

- **Problem Setting:**
  - Set of possible instances *X* ,
  - each instance $x \in X$ is a feature vector $x = \langle x_1, x_2, \ldots, x_n \rangle$.
- Unknown **target function** $f = X \rightarrow Y$
  - *Y* is discrete valued.
- Set of **function hypotheses** $H = \{h \mid h : X \rightarrow Y\}$
  - each hypothesis *h* is a decision tree.
- **Input**:
  - Training examples $\{\langle x^{(i)}, y^{(i)} \rangle\}$ of unknown target function *f* .
- **Output**:
  - Hypothesis $h \in H$ that best approximates target function *f* .

Decision Tree Learning

- In general, decision trees represent a **disjunction of conjunctions** of constraints on the attribute values of instances.
- The given decision tree corresponds to:

$$(outlook = sunny \wedge humidity = low) \vee$$
$$(outlook = overcast) \vee$$
$$(outlook = rain \wedge wind = weak)$$



- Decision Trees can represent any function.

Decision Tree Learning

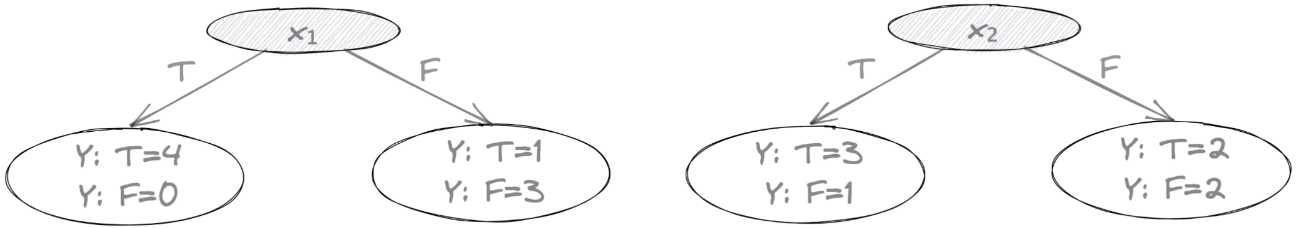
- Constructing a decision tree is easy:
  - Given ***n* features**, there are ***n!* possible decision trees**.
- Unfortunately, a decision tree can grow very large.
  - Given ***n* features** with ***m* possible choices,**
  - then a decision tree might grow up to size $\mathbf{m^n}$.
- The **size of a decision tree** depends on the **order of its features**.
- **Learning algorithm** tries to determine a good ordering.
- Learning the simplest (smallest) decision tree is an **NP complete problem** (if you are interested, check: Hyafil & Rivest'76).

Decision Tree Learning - ID3 Algorithm

- Resort to a **greedy heuristic**:
  - Start from an empty decision tree,
  - split on next “best” attribute,
  - recurse.
- What is the best attribute?
- We use **information theory** to guide us.

Decision Tree Learning - ID3 Algorithm

- Which attribute is better to split on, X_1 or X_2?



- **Idea:** Use counts at leaves to define probability distributions, so we can measure uncertainty.

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Decision Tree Learning - ID3 Algorithm

Entropy

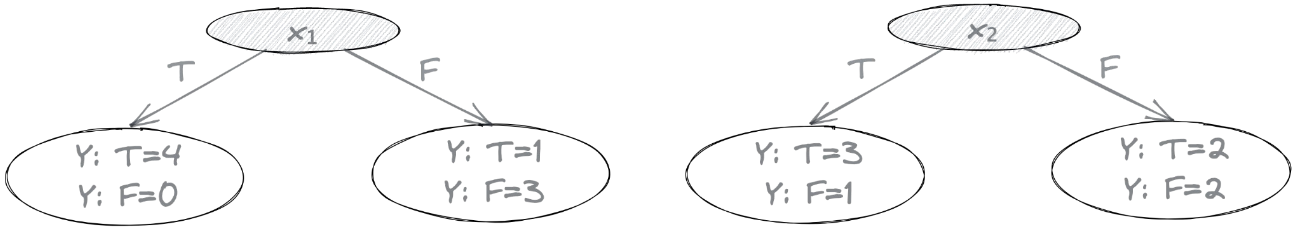
- The **Information Content** (**Entropy**) ***H*** of a feature variable *x* with *n* possible feature values is based on its relative frequency of occurrence (probability):

$$H(x) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

- The unit to measure information content is **bit** (*binary digit, basic indissoluble information unit*).

Decision Tree Learning - ID3 Algorithm

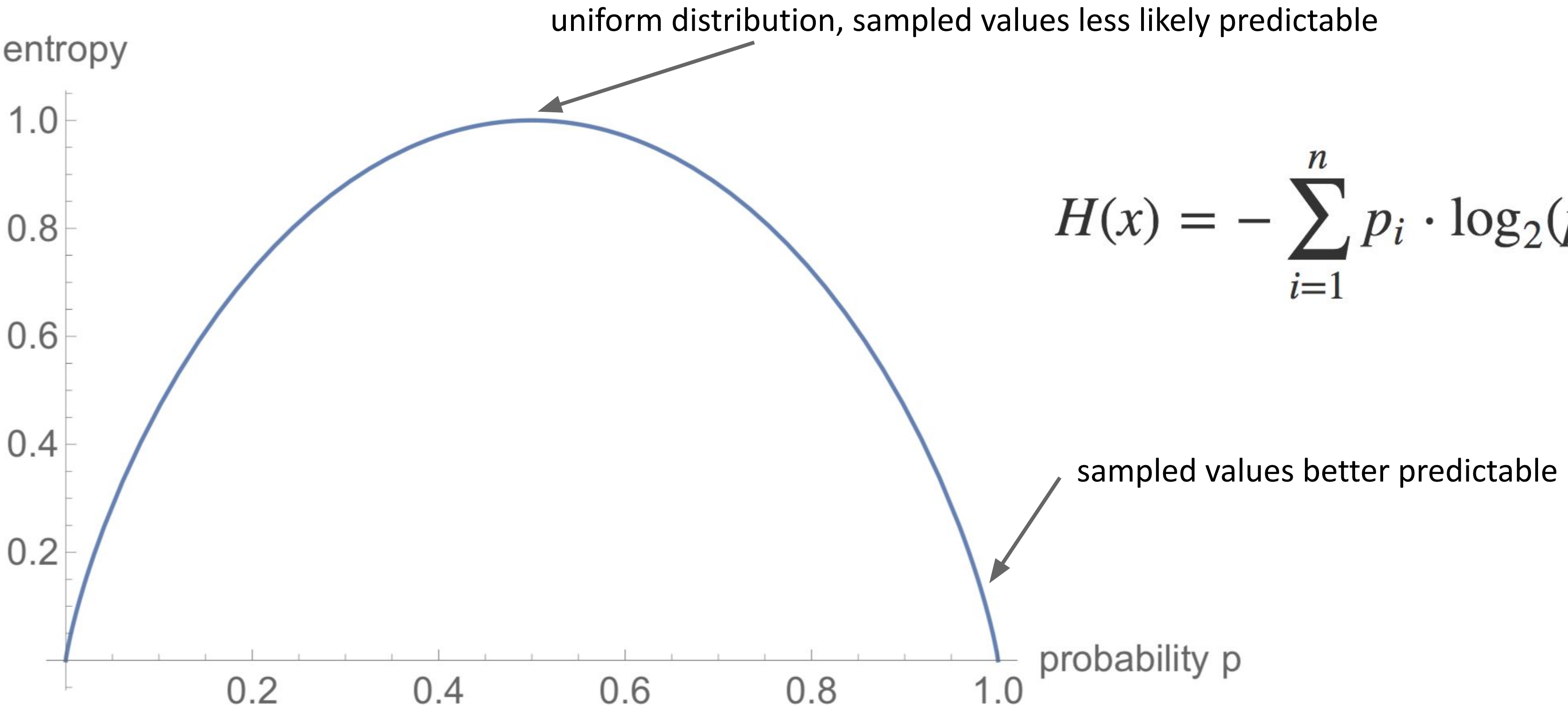
- Which attribute is better to split on, X_1 or X_2?



- **Entropy of Y:** [T=5, F=3] $H(Y) = -\frac{5}{8}\log_2(\frac{5}{8}) - \frac{3}{8}\log_2(\frac{3}{8}) = 0.95$ bit

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Decision Tree Learning - ID3 Algorithm



$$H(x) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

Decision Tree Learning - ID3 Algorithm

Information Gain

- The **Information Gain** of a feature ***Y*** due to a feature ***X*** is a measure of the effectiveness of a feature in classifying the training data.
- It is simply the **expected reduction in entropy** caused by partitioning the examples according to this feature:

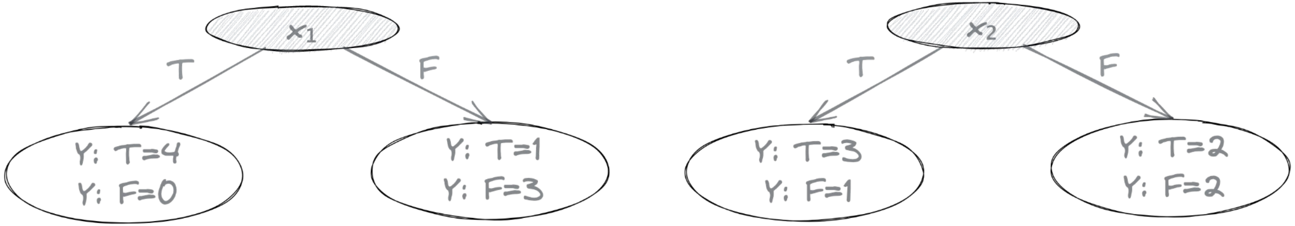
$$IG(Y|X) = H(Y) - H(Y|X)$$
$$= H(Y) - \sum_{x \in X} \frac{|Y_x|}{|Y|} H(Y_x)$$

- Where x are all possible values of feature X, and Y_x the subset of all instances in Y with $X=x$

Decision Tree Learning - ID3 Algorithm

- Which attribute is better to split on, X_1 or X_2?

$$IG(Y|X) = H(Y) - \sum_{x \in X} \frac{|Y_x|}{|Y|} H(Y_x)$$

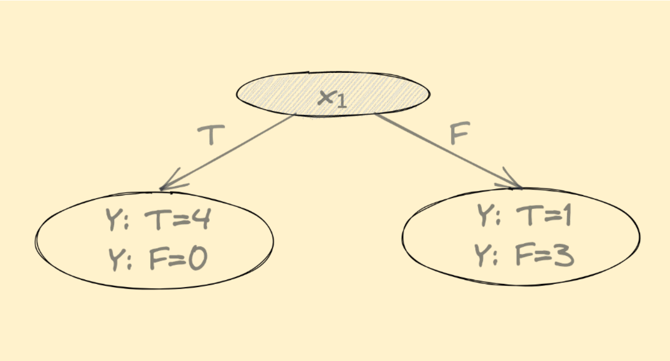


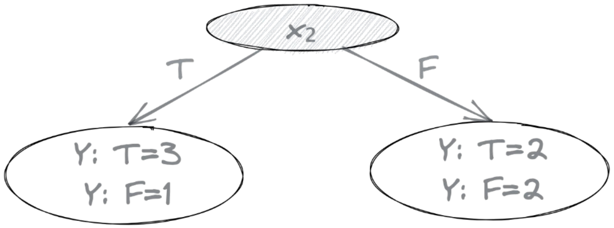
- IG(Y|X_1) = 0.95 - (4/8 H(X_1=True) + 4/8 H(X_1=False)) = 0.95 - (0 + 0.4) = **0.55**
  IG(Y|X_2) = 0.95 - (4/8 H(X_2=True) + 4/8 H(X_2=False))= 0.95 - (0.4 + 0.5) = **0.05**

Decision Tree Learning - ID3 Algorithm

- Which attribute is better to split on, X_1 or X_2?

$$IG(Y|X) = H(Y) - \sum_{x \in X} \frac{|Y_x|}{|Y|} H(Y_x)$$

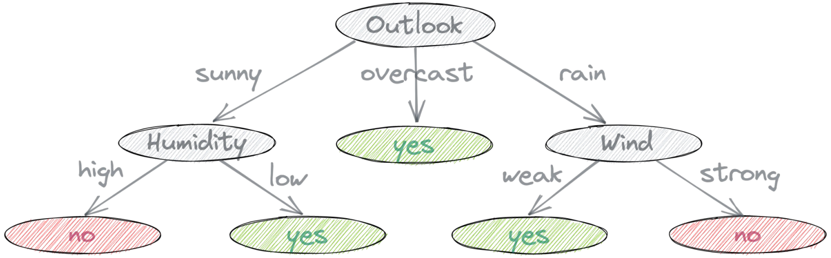




- $IG(Y|X_1) = 0.95 - (4/8\ H(X_1{=}\text{True}) + 4/8\ H(X_1{=}\text{False})) = 0.95 - (0 + 0.4) =$ **0.55**
  $IG(Y|X_2) = 0.95 - (4/8\ H(X_2{=}\text{True}) + 4/8\ H(X_2{=}\text{False})) = 0.95 - (0.4 + 0.5) =$ **0.05**

Decision Tree Learning - ID3 Algorithm

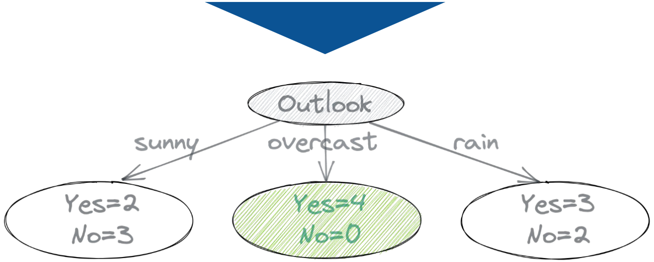
- ID3 **greedy heuristic**:
  1. Start from an empty decision tree.
  2. Split on next "best" attribute,
     i.e. the attribute with the **next highest information gain**.
  3. Recurse.



ID3 Algorithm - Example

Outlook	Temperature	Humidity	Wind	PlayTennis
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	low	weak	yes
rain	cool	low	strong	no
overcast	cool	low	strong	yes
sunny	mild	high	weak	no
sunny	cool	low	weak	yes
rain	mild	low	weak	yes
sunny	mild	low	strong	yes
overcast	mild	high	strong	yes
overcast	hot	low	weak	yes
rain	mild	high	strong	no

IG(PlayTennis,**Outlook**) = **0.246**
IG(PlayTennis,**Humidity**) = 0.151
IG(PlayTennis,**Wind**) = 0.048
IG(PlayTennis,**Temperature**) = 0.029



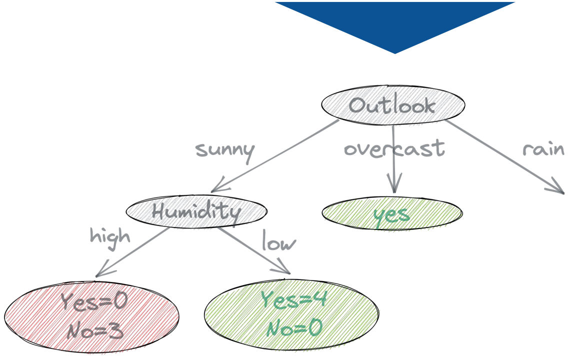
ID3 Algorithm - Example

Outlook	Temperature	Humidity	Wind	PlayTennis
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	low	weak	yes
rain	cool	low	strong	no
overcast	cool	low	strong	yes
sunny	mild	high	weak	no
sunny	cool	low	weak	yes
rain	mild	low	weak	yes
sunny	mild	low	strong	yes
overcast	mild	high	strong	yes
overcast	hot	low	weak	yes
rain	mild	high	strong	no

$IG(Outlook_{sunny}, \textbf{Humidity}) = \textbf{0.97}$

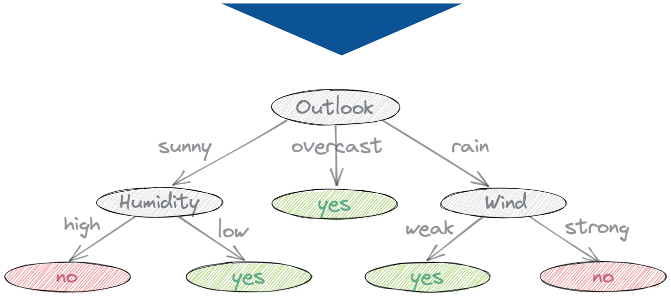
$IG(Outlook_{sunny}, \textbf{Temperature}) = \textbf{0.57}$

$IG(Outlook_{sunny}, \textbf{WInd}) = \textbf{0.019}$



ID3 Algorithm - Example

Outlook	Temperature	Humidity	Wind	PlayTennis
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	low	weak	yes
rain	cool	low	strong	no
overcast	cool	low	strong	yes
sunny	mild	high	weak	no
sunny	cool	low	weak	yes
rain	mild	low	weak	yes
sunny	mild	low	strong	yes
overcast	mild	high	strong	yes
overcast	hot	low	weak	yes
rain	mild	high	strong	no



Example from: Tom Mitchel, *Machine Learning*, MacGraw-Hill (1997), p.59-61

Decision Trees - Pros and Cons

- **Simple to understand** and interpret.
- Able to handle both **numerical and categorical data**.
- Requires **little data preparation**.
- Uses a **white box model**.
- Possible to **validate a model using statistical tests**.
- Performs well with **large datasets**.
- **Mirrors human decision making** more closely than other approaches.
- **BEWARE**: Decision Tree models easily overfit.

- DT Learning Algorithms:
  - **ID3** (Iterative Dichotomiser 3)
  - **C4.5** (successor of ID3)
  - **CART** (Classification And Regression Tree)
  - **CHAID** (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.
  - **MARS:** extends decision trees to handle numerical data better.

Karlsruher Institut für Technologie

FIZ Karlsruhe
Leibniz Institute for Information Infrastructure

Decision Trees Hands On Example

11 - ISE2021 - Decision Trees Example.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment

+ Code + Text

RAM
Disk

Basic Machine Learning - Decision Trees Examples

This is the colab notebook example for lecture 11 Basic Machine Learning 2, chapter 4.7 Basic ML Algorithms 3 - Decision Trees, of AIFB/KIT Lecture "Information Service Engineering" Summer Semester 2021.

In this colab notebook you will learn how to make use of the SciKit Learn library for applying machine learning algorithms, in particular the decision trees and random forest classifier, mathplotlib and seaborn for data visualization, pandas to analyze and manipulate data.

Please make a copy of this notebook to try out your own adaptions via "File -> Save Copy in Drive"

[2]

```
# Common imports
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score,confusion_matrix

# for data analysis and manipulation
import pandas as pd

# To plot pretty figures
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['axes.labelsize'] = 10
plt.rcParams['xtick.labelsize'] = 10
plt.rcParams['ytick.labelsize'] = 10
```

We will load a dataset with **weather observations**, including the information whether it has rained or not. We will use this dataset to train a **decision tree classifier** that predicts - based on the available weather data - **whether it will rain the next day or not**.

[Decision Tree Python Colab Notebook]

4. Basic Machine Learning - 2
Bibliography

- T. Mitchel, ***Machine Learning***, MacGraw-Hill, 1997
  - Chap 3.4.2 (pp. 59-61, The Tennis Decision Tree Example)
- S. Marsland, ***Machine Learning, An Algorithmic Perspective***, 2nd. ed., Chapman & Hall / CRC Press, 2015
  - Chap. 14.1 (k-Means Algorithm)
  - Chap. 3.5 (Linear Regression)
  - Chap. 12 (Decision Trees)

(Both books should also be available on the Web as pdf, just keep looking…)

4. Basic Machine Learning - 2

Syllabus Questions

- What’s the difference between supervised and unsupervised ML?
- Explain the basic concept of k-Means Clustering.
- Explain the concept of Linear Regression.
- What kind of problems can be solved via Linear Regression?
- Explain the difference between Linear, Multilinear, and Polynomial Regression.
- When are Decision Trees preferred over Linear (Multilinear or Polynomial) Regression?
- Explain the concept of Information Gain.
- What are the Pros and Cons of Decision Trees?