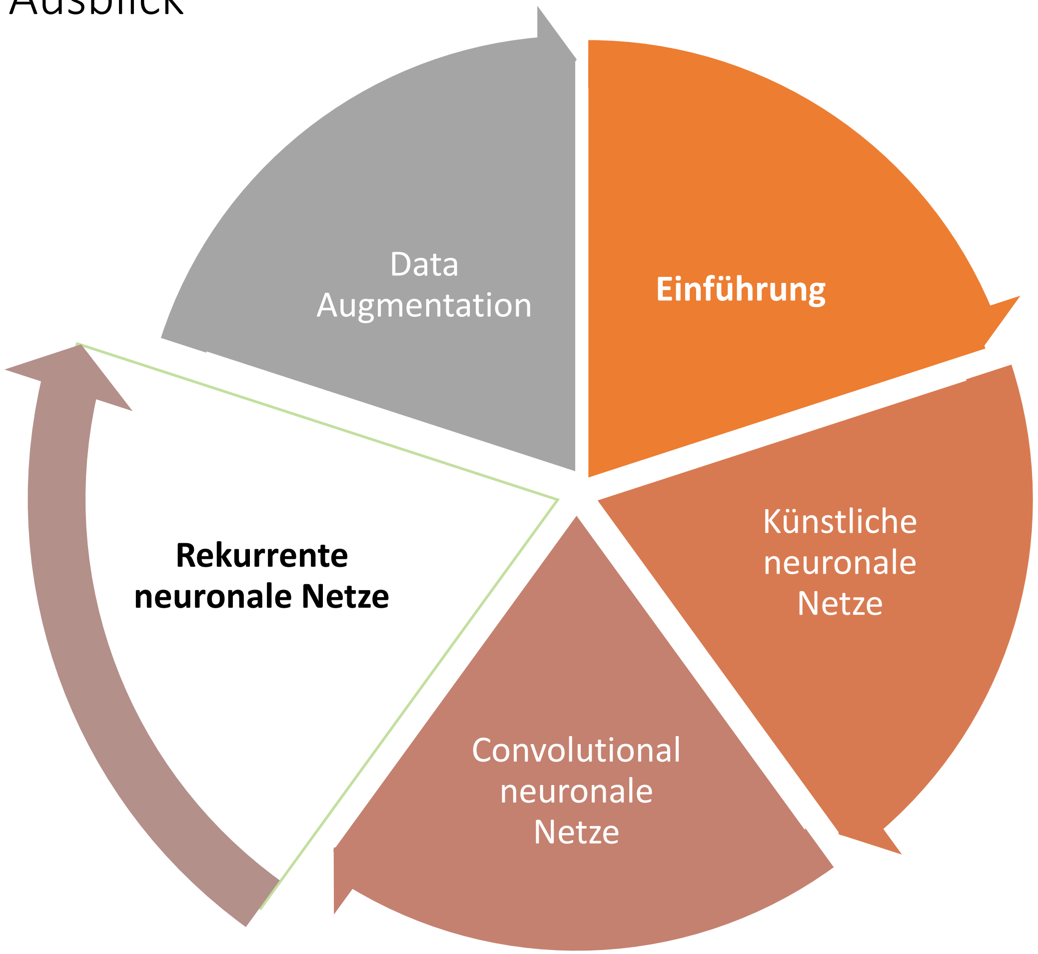
Deep Learning

Ausblick



Recurrent neural networks (RNN)

introduction
Sequences: What time is it?
Embedding Layers
Building an ANN
Long Short-Term Memory (LSTM)
Keras code

Recurrent neural network - RNN
Feedforward Vs. RNNs

Recurrents neural network
Output depends on previous calculations

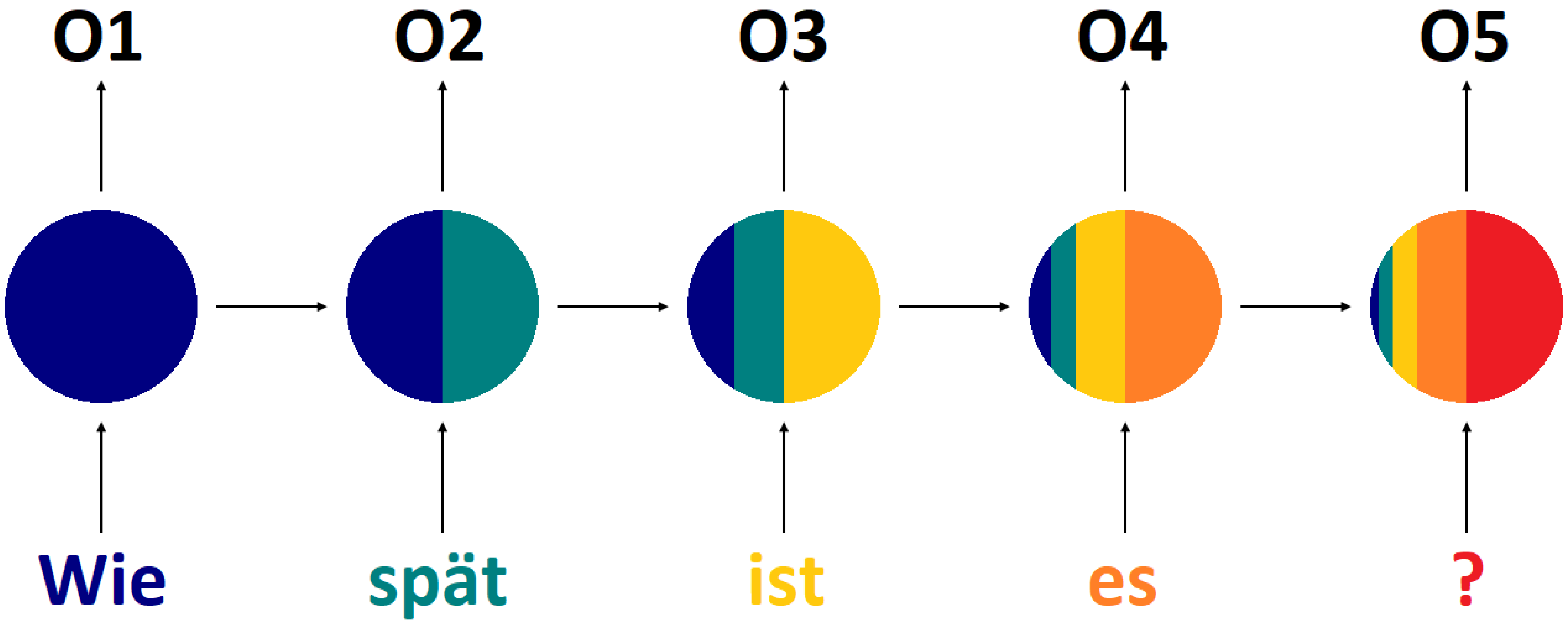
	Architecture & Signal Flow	Inspired by	memory	input and output	Detection of
KNN & CNN	• Feedforward • one direction (from the entrance to the exit) •	• Neuronen • visueller Cortex	• None to static memory (FNN-TD) •	• Fixed size • pictures • text • audio •	• pattern
RNN	• Feedback • Both directions (loops) •	• Neocortex	• Dynamic Memory •	• Any size • Data in sequential form (e.B. time series) •	• Time-coded information •

Artificial Neural Networks - KNN Notations

- N: Number of (training) instances
- p: Dimension of input (number of features)
- K: Dimension of output (or number of classes)
- L: Number of layers of a neural network
- $\boldsymbol{x}^{(i)} \in \mathbb{R}^p / \boldsymbol{y}^{(i)} \in \mathbb{R}^K$: the ith input vector / output vector represented as a line vector
- $\boldsymbol{x}_j^{(i)} / \boldsymbol{y}_k^{(i)}$: The jth element of the ith input vector / output vector (scalar)
- $\mathbf{X} \in \mathbb{R}^{N \times p} / \mathbf{Y} \in \mathbb{R}^{N \times K} : \mathrm{X} = \begin{pmatrix} \left(\boldsymbol{x}^{(1)}\right)^T \\ \vdots \\ \left(\boldsymbol{x}^{(N)}\right)^T \end{pmatrix} = \begin{pmatrix} x_1^{(1)}\ x_2^{(1)} \dots x_p^{(1)} \\ \vdots \\ x_1^{(N)}\ x_2^{(N)} \dots x_p^{(N)} \end{pmatrix}$ Input matrix
- $\omega_{ji}^{[\ell]}$: Weight of the jth input of the ith neuron of the ℓ-ten layer (alternatively θ)
- $\mathbf{W}^{[\ell]} \in \mathbb{R}^{(\text{Number of neurons of the layer}\ell) \times (\text{Number of neurons of the layer}\ell - 1)}$: Weight matrix of the layer ℓ.
- $\hat{\boldsymbol{y}}^{(i)} \in \mathbb{R}^K$, $\hat{\boldsymbol{y}}^{(i)} = h\left(\boldsymbol{x}^{(i)}\right)$: Is the predicted output vector (estimator)
  h: is the prediction function of your system called a hypothesis
- $\sigma^{[\ell]}(\cdot)$: Activation function of the ℓ-ten layer, for the step function we write $\sigma(\cdot) = f(\cdot)$

Recurrent neural network - RNN
Example: What time is it?

- **Output**
  Time Step 1: O1
  Time Step 2: O2
- **network**:
  Circle represents the entire network at different time steps
-
- **Input**



Recurrent neural network - RNN
Embedding Layer - Working with Text Data

- **Text vectorization** (Assign a unique number to words)
  - Create a vocabulary
  - Creating One-Hot Vectors
  - Keras: Tokenizer

  disadvantages:
  - Inefficient in matrix multiplication
  - Can be very high-dimensional

V = {„have“, „ a“, „ nice“, „day“ }

have $= [1,0,0,0]^T$

a = $[0,1,0,0]^T$

day = $[0,0,0,1]^T$

Embedded Layer
- First layer of a neural network
- Multiplication of one-hot vectors with the embedded wei
- Dimension reduction

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 8 & 2 & 1 & 9 \\ 6 & 5 & 4 & 0 \\ 7 & 1 & 6 & 2 \\ 1 & 3 & 5 & 8 \\ 0 & 4 & 9 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 5 & 8 \end{bmatrix}$$

One-hot vector

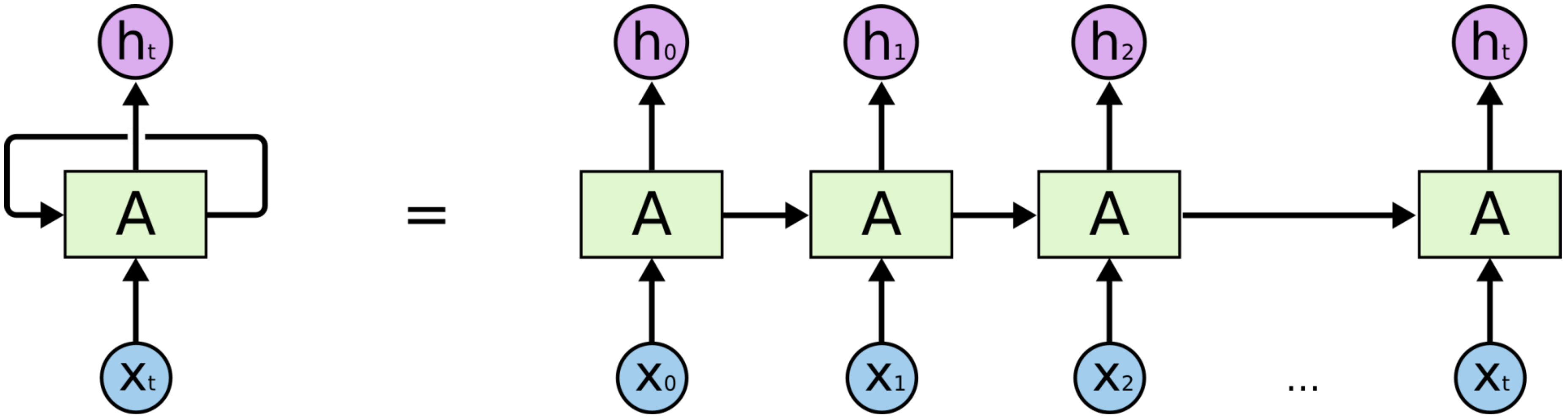
Embedding Weight Matrix

Hidden layer output

Recurrent neural network - RNN
Building an RNN

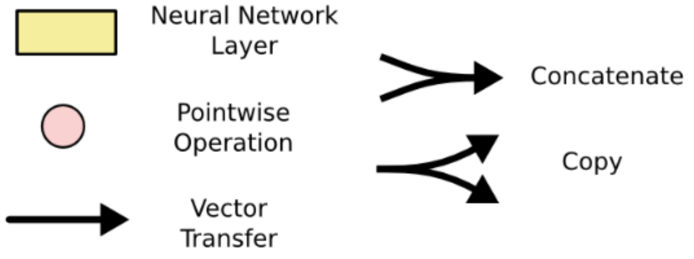
- **Rekurrente neuronale Netze**
  - Have loops
  - Grinding allows to hold information
  - Repeating modules from NNs
  - Vanishing Gradient Problem
  -

The problem of long-term dependency
1. The clouds are in the sky
2. I grew up in France... I speak fluent French

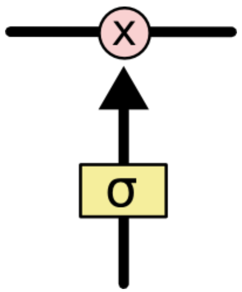


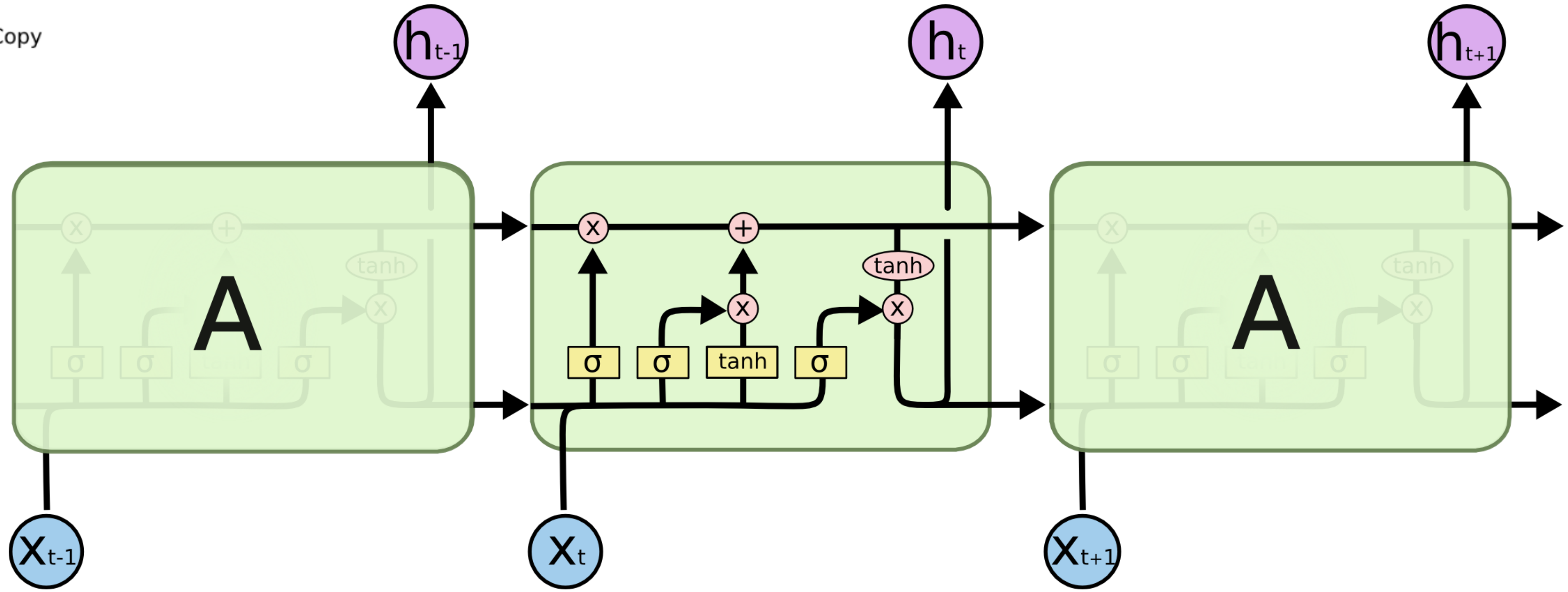
Rekurrentes Neuronales Netz - RNN

Long Short-Term Memory (LSTM)



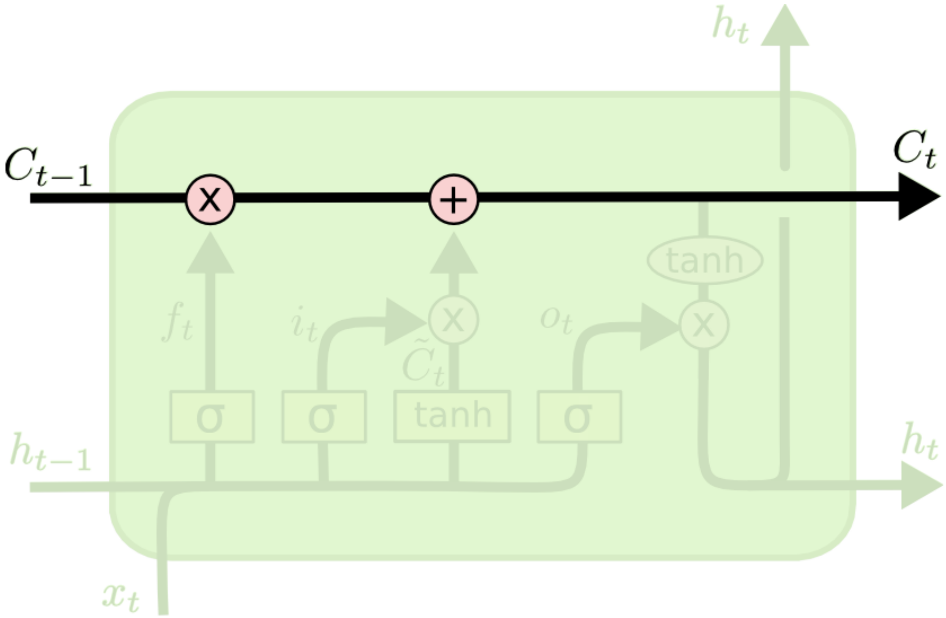
Gates:





Rekurrentes Neuronales Netz - RNN

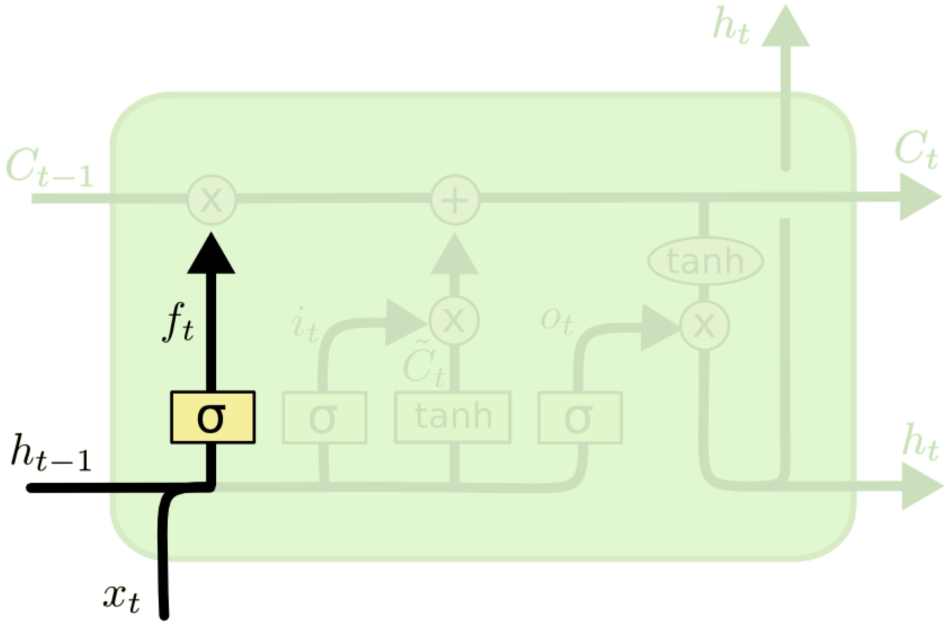
LSTM – Der Zellzustand



- Cell condition C_T
  - Conveyor belt for information
  - Extends through the full chain
  - Add/remove information through two linear operations
  -

Rekurrentes Neuronales Netz - RNN

LSTM – Forget Gate Layer

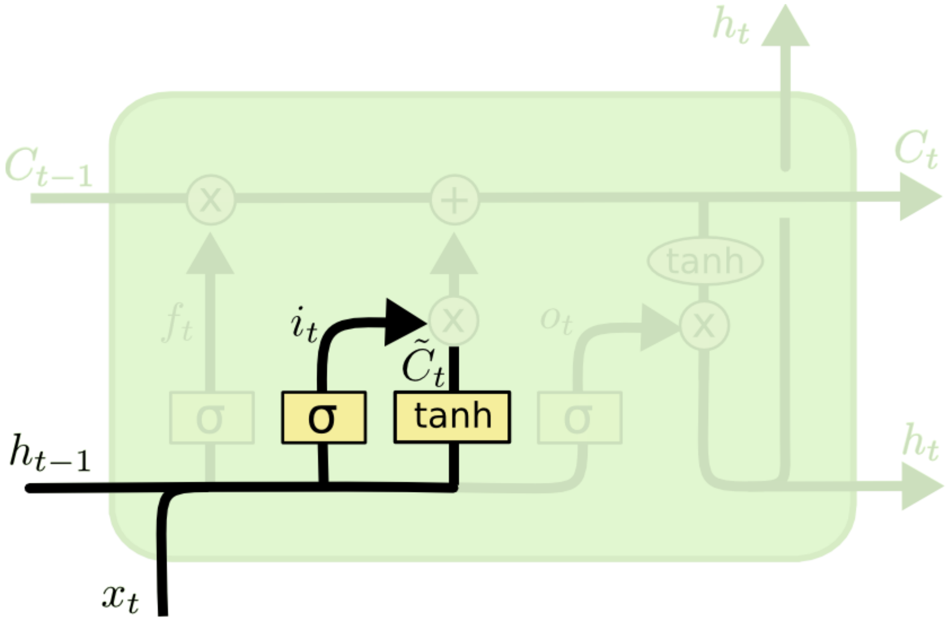


- Forget Gate Layer
  - Links the vectors h_{t-1} und x_t
  - f_t is the output of the sigmoid layer
  - 0(1) means the information of C_{t-1} to discard (to maintain)

- $f_t = \sigma\left(W_f \cdot [h_{t-1}. x_t] + b_f\right)$

Rekurrentes Neuronales Netz - RNN

LSTM – Input Gate Layer



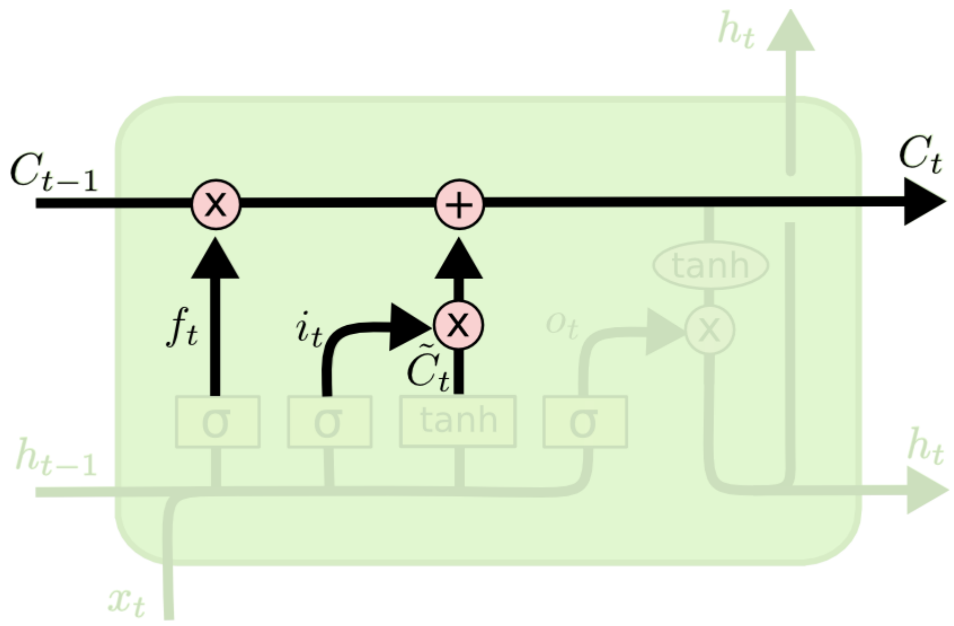
update of Cell state

1. Input Gate Layer
   - What values to update
2. $tanh$ - layer
   - Create potential candidates
3. Unification of both steps:
   → receive cell state updates

- $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
- $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

Rekurrentes Neuronales Netz - RNN

LSTM – Aktualisierung des Zellzustandes

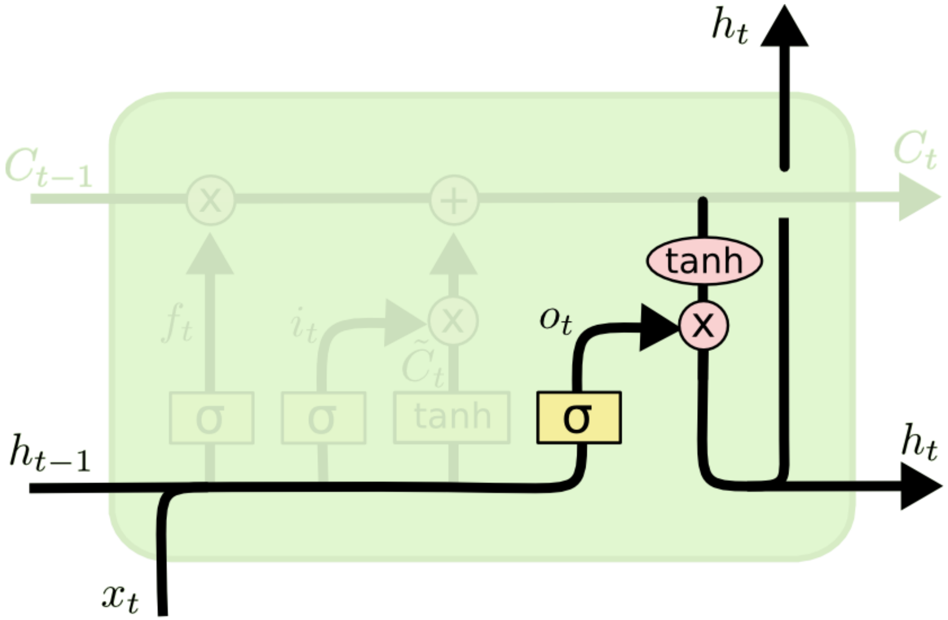


- Perform an update
  - Multiply the old cell state C_{t-1} with f_t (forget about selected information)
  - Add up the new candidate values $i_t \cdot \tilde{C}_t$ (new scaled candidates)

- $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$

Rekurrentes Neuronales Netz - RNN

LSTM – Generieren einer Ausgabe



Generating the output (Output Gate Layer)

1. Set the cell state through the tanh layer to the values between -1 and +1
   → Filters Information
2. Multiply the filtered values by the Sigmoid-Gate

→ Output is limited to information we have chosen

- $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
- $h_t = o_t \cdot \tanh(C_t)$

Rekurrentes Neuronales Netz - RNN

Keras Implementierung eines LSTMs

1. Input-Sequence-Arrays
   – Instances: 15631,
   – Sequence Length: 50, Features: 25
2. Creating the model
   – LSTM layer:
     Input dimension: (50, 25)
     Units: 100
   – Dropout layer
   – LSTM layer:
     Input dimension: (50, 25)
     Units: 50
   – dropout
   – Fully crosslinked layer
     Units: 50
   –

```
seq_array.shape, label_array.shape
```

```
((15631, 50, 25), (15631, 1))
```

```
model = Sequential()

model.add(LSTM(
         input_shape=(seq_array.shape[1], seq_array.shape[2]),
         units=100,
         return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(
          units=50,
          return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(units=label_array.shape[1], activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Rekurrentes Neuronales Netz - RNN

Keras Implementierung eines LSTMs

3. Summary of the model

```
print(model.summary())
```

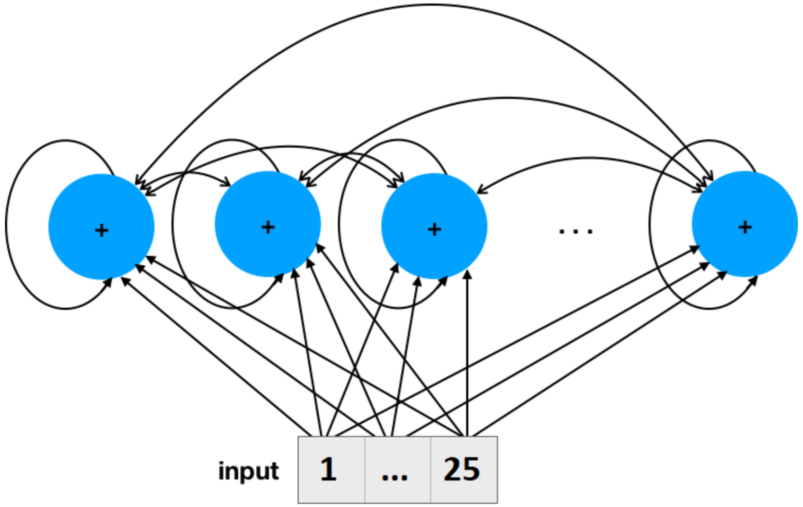
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50, 100)           50400
_________________________________________________________________
dropout_1 (Dropout)          (None, 50, 100)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                30200
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 80,651
Trainable params: 80,651
Non-trainable params: 0
```

Recurrent neural network - RNN
Calculation of parameters of an LSTMa

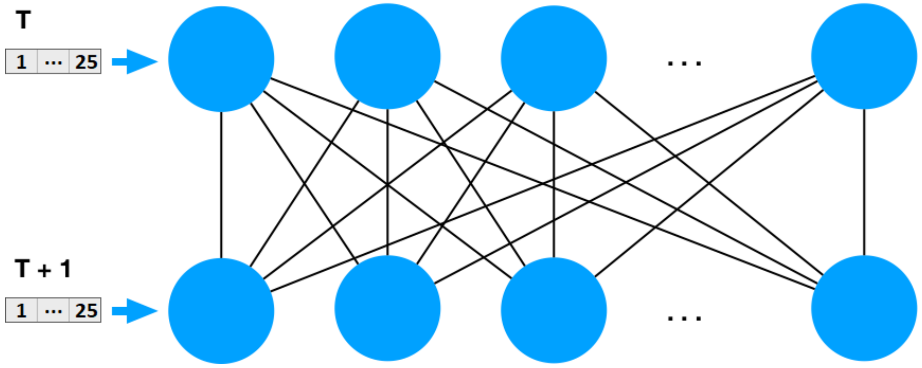
First, consider simple RNN:
- 100 cells (blue)
- 25 Features
- Calculation of parameters:
  - Recurrent connections:
    *num_units * num_units* = 100^2
  - input:
    *input_dim * num_units + num_units* = $25 \cdot 100 + 100$
  - together:
$$100^2 + 25 \cdot 100 + 100 = 12600$$

LSTM layer:

Issue times 4:
$$12600 \cdot 4 = 50400$$



Unrolled



Rekurrentes Neuronales Netz - RNN

Keras Implementierung eines LSTMs

4. Train the model

```
%%time
# fit the network
model.fit(seq_array, label_array, epochs=10, batch_size=200, validation_split=0.05, verbose=1,
          callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=0, verbose=0, mode='auto')])
```

```
Train on 14849 samples, validate on 782 samples
Epoch 1/10
14849/14849 [==============================] - 10s 643us/step - loss: 0.2576 - accuracy: 0.8873 - val_loss: 0.0635 - val_accura
cy: 0.9731
Epoch 2/10
14849/14849 [==============================] - 8s 545us/step - loss: 0.1090 - accuracy: 0.9567 - val_loss: 0.0610 - val_accurac
y: 0.9706
Epoch 3/10
14849/14849 [==============================] - 8s 540us/step - loss: 0.0777 - accuracy: 0.9679 - val_loss: 0.0340 - val_accurac
y: 0.9885
Epoch 4/10
14849/14849 [==============================] - 8s 537us/step - loss: 0.0694 - accuracy: 0.9706 - val_loss: 0.0417 - val_accurac
y: 0.9859
Wall time: 34.8 s
```

Rekurrentes Neuronales Netz - RNN

Keras Implementierung eines LSTMs

5. Evaluating the model
6.

```
# test metrics
scores_test = model.evaluate(seq_array_test_last, label_array_test_last, verbose=2)
print('Accurracy: {}'.format(scores_test[1]))
```

```
Accurracy: 0.9677419066429138
```

```
# make predictions and compute confusion matrix
y_pred_test = model.predict_classes(seq_array_test_last)
y_true_test = label_array_test_last
print('Confusion matrix\n- x-axis is true labels.\n- y-axis is predicted labels')
cm = confusion_matrix(y_true_test, y_pred_test)
cm
```

```
Confusion matrix
- x-axis is true labels.
- y-axis is predicted labels

array([[67,  1],
       [ 2, 23]], dtype=int64)
```

```
# compute precision and recall
precision_test = precision_score(y_true_test, y_pred_test)
recall_test = recall_score(y_true_test, y_pred_test)
f1_test = 2 * (precision_test * recall_test) / (precision_test + recall_test)
print( 'Precision: ', precision_test, '\n', 'Recall: ', recall_test,'\n', 'F1-score:', f1_test )
```

```
Precision:  0.9583333333333334
 Recall:  0.92
 F1-score: 0.9387755102040817
```

Deep Learning

Gegenüberstellung: KNNs, CNNs und RNNs

	KNNs (MLP)	**CNNs**	**RNNs**
data	tabular	Image	sequences
Recurrent connections	N	N	Y
Shared parameters	N	Y	Y
Identifying spatial relationships	N	Y	No
Vanishing & Exploding Gradients	Y	Y	Y

Data Preparation

(Data pre-processing)

Data Preparation

- Introduction to Data Preparation
- Types of Data
- Discretization of Continuous Variables
- Outliers
- Data Transformation
- Missing Data
- Handling Redundancy
- Sampling and Unbalanced Datasets

INTRODUCTION TO DATA PREPARATION

Why Prepare Data?

- Some data preparation is needed for all mining tools
- The purpose of preparation is to transform data sets so that their information content is best exposed to the mining tool
- Error prediction rate should be lower (or the same) after the preparation as before it

Why Prepare Data?

- Preparing data also prepares the miner so that when using prepared data the miner produces better models, faster

- GIGO - good data is a prerequisite for producing effective models of any type

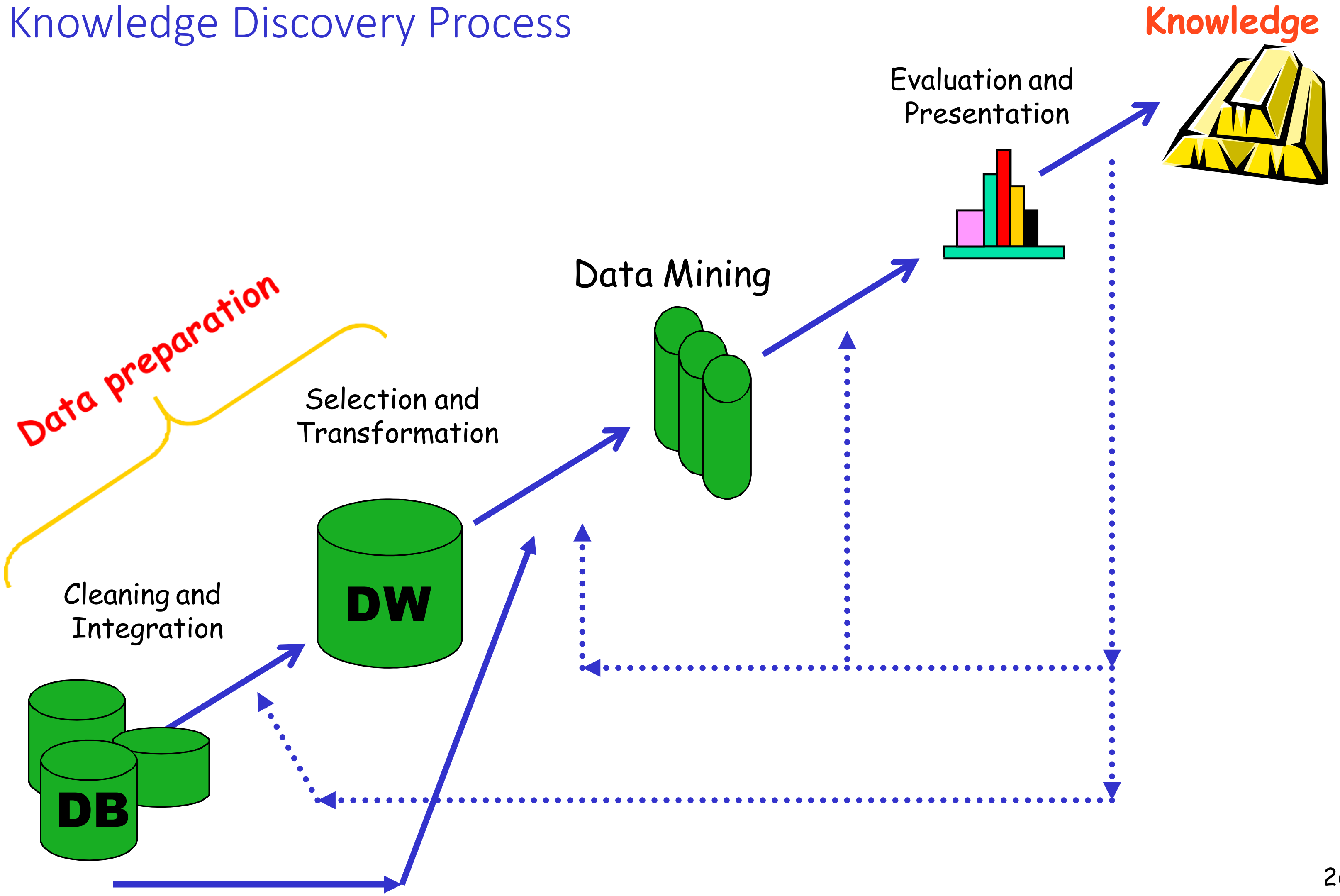
Why Prepare Data?

- Data need to be formatted for a given software tool
- Data need to be made adequate for a given method
- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“”
  - noisy: containing errors or outliers
    - e.g., Salary=“-10”, Age=“222”
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age=“42” Birthday=“03/07/1997”
    - e.g., Was rating “1,2,3”, now rating “A, B, C”
    - e.g., discrepancy between duplicate records
    - e.g., ***Endereço:*** travessa da Igreja de Nevogilde ***Freguesia:*** Paranhos

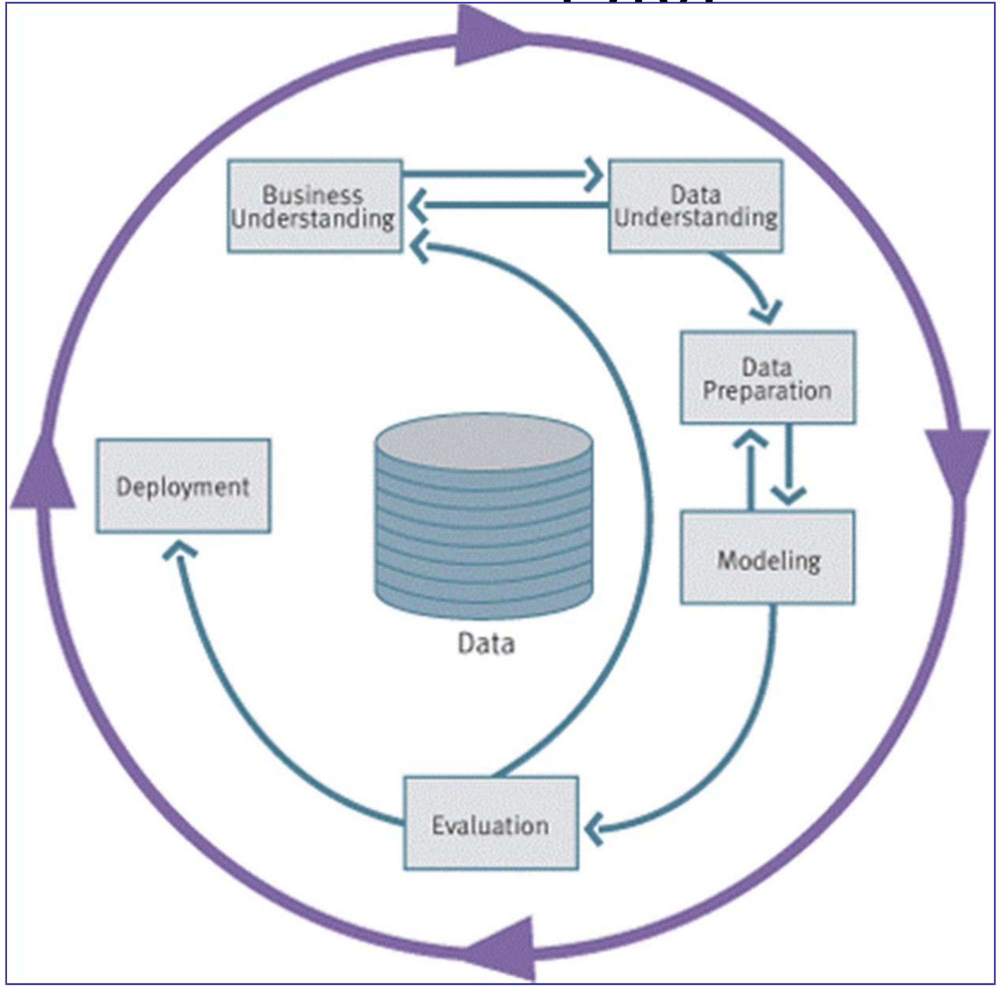
Major Tasks in Data Preparation

- Data discretization
  - Part of data reduction but with particular importance, especially for numerical data
- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases, data cubes, or files
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results

Data Preparation as a step in the Knowledge Discovery Process



CRISP-DM



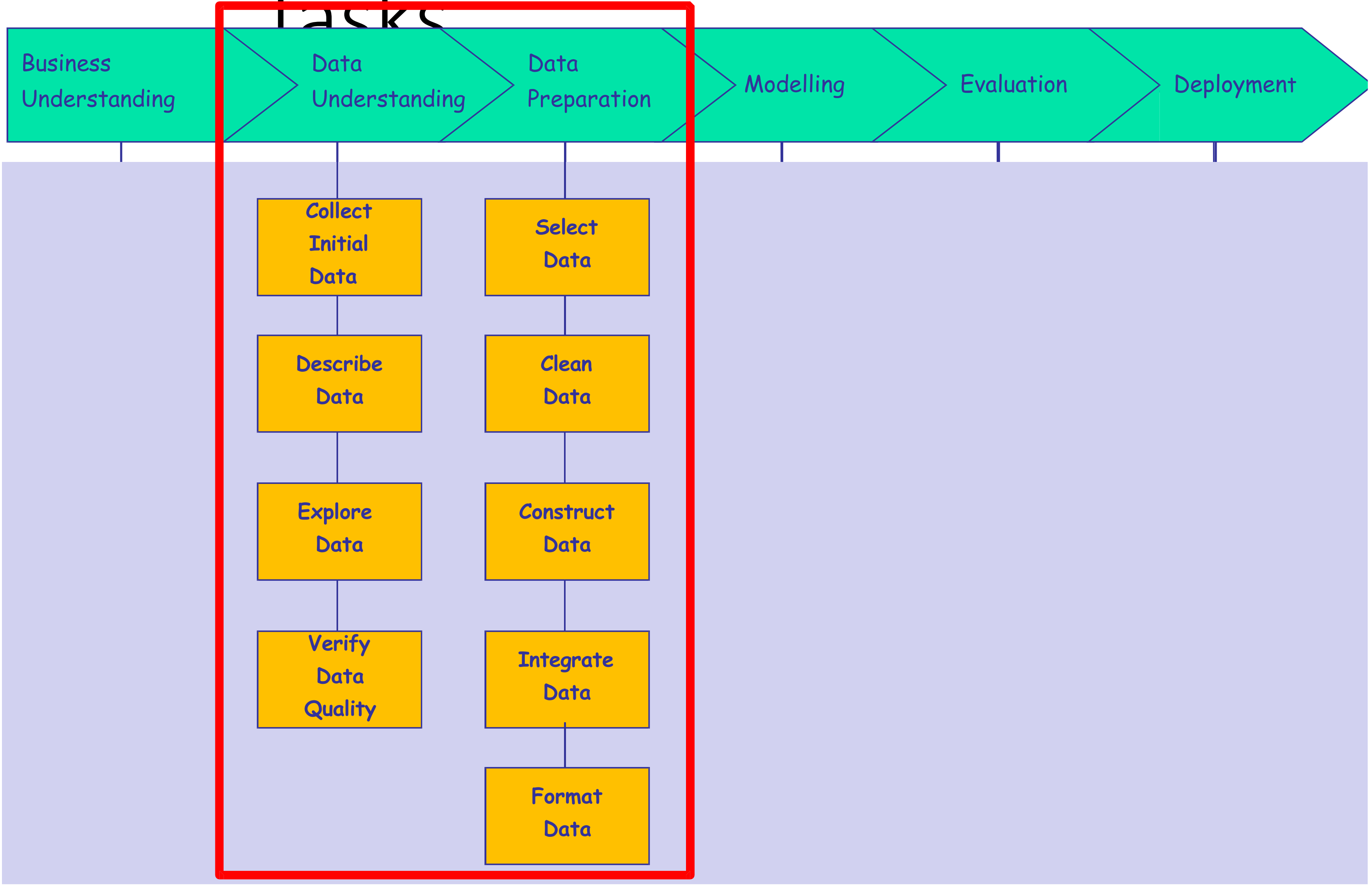
CRISP-DM is a comprehensive data **mining methodology** and process model that provides anyone—from novices to data mining experts—with a complete blueprint for conducting a data mining project.

A methodology enumerates the steps to reproduce success

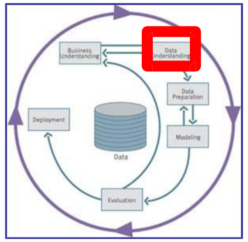
CRISP-DM Phases and Tasks

Business Understanding	Data Understanding	Data Preparation	Modelling	Evaluation	Deployment
Determine Business Objectives	Collect Initial Data	Select Data	Select Modeling Technique	Evaluate Results	Plan Deployment
Assess Situation	Describe Data	Clean Data	Generate Test Design	Review Process	Plan Monitoring & Maintenance
Determine Data Mining Goals	Explore Data	Construct Data	Build Model	Determine Next Steps	Produce Final Report
Produce Project Plan	Verify Data Quality	Integrate Data	Assess Model		Review Project
		Format Data			

CRISP-DM Phases and Tasks

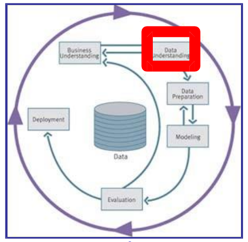


CRISP-DM: Data Understanding



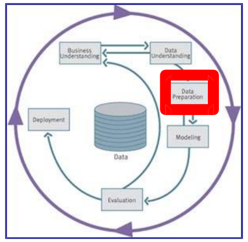
- **Collect data**
  - List the datasets acquired (locations, methods used to acquire, problems encountered and solutions achieved).
- **Describe data**
  - Check data volume and examine its gross properties.
  - Accessibility and availability of attributes. Attribute types, range, correlations, the identities.
  - Understand the meaning of each attribute and attribute value in business terms.
  - For each attribute, compute basic statistics (e.g., distribution, average, max, min, standard deviation, variance, mode, skewness).

CRISP-DM: Data Understanding



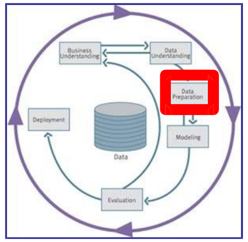
- **Explore data**
  - Analyze properties of interesting attributes in detail.
    - *Distribution, relations between pairs or small numbers of attributes, properties of significant sub-populations, simple statistical analyses.*
- **Verify data quality**
  - Identify special values and catalogue their meaning.
  - Does it cover all the cases required? Does it contain errors and how common are they?
  - Identify missing attributes and blank fields. Meaning of missing data.
  - Do the meanings of attributes and contained values fit together?
  - Check spelling of values *(e.g., same value but sometime beginning with a lower case letter, sometimes with an upper case letter).*
  - Check for plausibility of values, e.g. all fields have the same or nearly the same values.

CRISP-DM: Data Preparation



- **Select data**
  - Reconsider data selection criteria.
  - Decide which dataset will be used.
  - Collect appropriate additional data (internal or external).
  - Consider use of sampling techniques.
  - Explain why certain data was included or excluded.
- **Clean data**
  - Correct, remove or ignore noise.
  - Decide how to deal with special values and their meaning *(99 for marital status)*.
  - Aggregation level, missing values, etc.
  - Outliers?

CRISP-DM: Data Preparation



- **Construct data**
  - Derived attributes.
  - Background knowledge.
  - How can missing attributes be constructed or imputed?
- **Integrate data**
  - Integrate sources and store result (new tables and records).
- **Format Data**
  - Rearranging attributes *(Some tools have requirements on the order of the attributes, e.g. first field being a unique identifier for each record or last field being the outcome field the model is to predict).*
  - Reordering records *(Perhaps the modelling tool requires that the records be sorted according to the value of the outcome attribute).*
  - Reformatted within-value *(These are purely syntactic changes made to satisfy the requirements of the specific modelling tool, remove illegal characters, uppercase lowercase).*

TYPES OF DATA

Types of Measurements

- Nominal scale
- Categorical scale
- Ordinal scale
- Interval scale
- Ratio scale

Qualitative

Quantitative

Discrete or Continuous

More information content

Types of Measurements: Examples

- Nominal:
  - ID numbers, Names of people
- Categorical:
  - eye color, zip codes
- Ordinal:
  - rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
- Interval:
  - calendar dates, temperatures in Celsius or Fahrenheit, GRE (Graduate Record Examination) and IQ scores
- Ratio:
  - temperature in Kelvin, length, time, counts

Types of Measurements: Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
1	Sunny	85	85	Light	No
2	Sunny	80	90	Strong	No
3	Overcast	83	86	Light	Yes
4	Rain	70	96	Light	Yes
5	Rain				
6	Rain				
7	Overcast				
8	Sunny				
9	Sunny				
10	Rain				
11	Sunny				
12	Overcast				
13	Overcast				
14	Rain				

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

Data Conversion

- Some tools can deal with nominal values but other need fields to be numeric
- Convert ordinal fields to numeric to be able to use ">" and "<" comparisons on such fields.
  - A → 4.0
  - A- → 3.7
  - B+ → 3.3
  - B → 3.0
- Multi-valued, unordered attributes with small no. of values
  - e.g. Color=Red, Orange, Yellow, …, Violet
  - for each value *v* create a binary "flag" variable *C_v* , which is 1 if Color=*v*, 0 otherwise

Conversion: Nominal, Many Values

- Examples:
  - US State Code (50 values)
  - Profession Code (7,000 values, but only few frequent)

- Ignore ID-like fields whose values are unique for each record
- For other fields, group values "naturally":
  - e.g. 50 US States → 3 or 5 regions
  - Profession – select most frequent ones, group the rest
- Create binary flag-fields for selected values

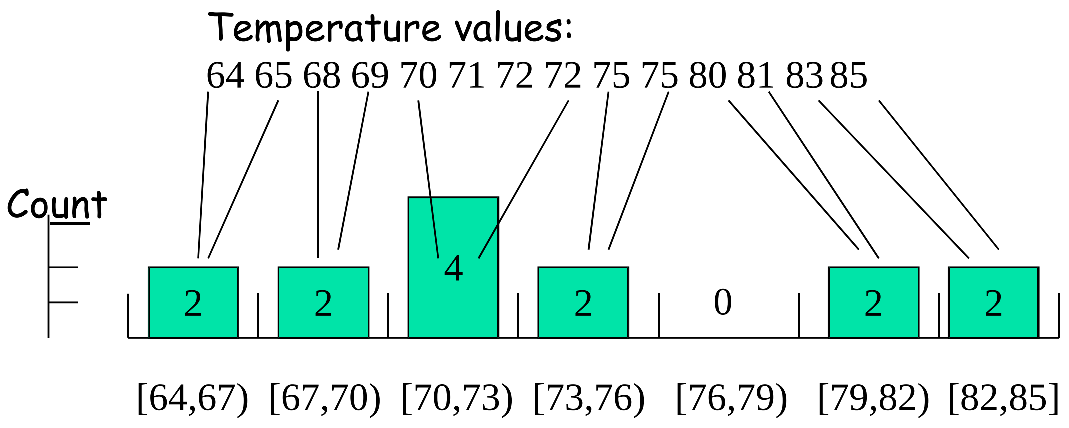
DISCRETIZATION OF CONTINUOUS VARIABLES

Discretization

- Divide the range of a continuous attribute into intervals
  - Some methods require discrete values, e.g. most versions of Naïve Bayes, CHAID
  - Reduce data size by discretization
  - Prepare for further analysis
  - Discretization is very useful for generating a summary of data
- Also called "binning"

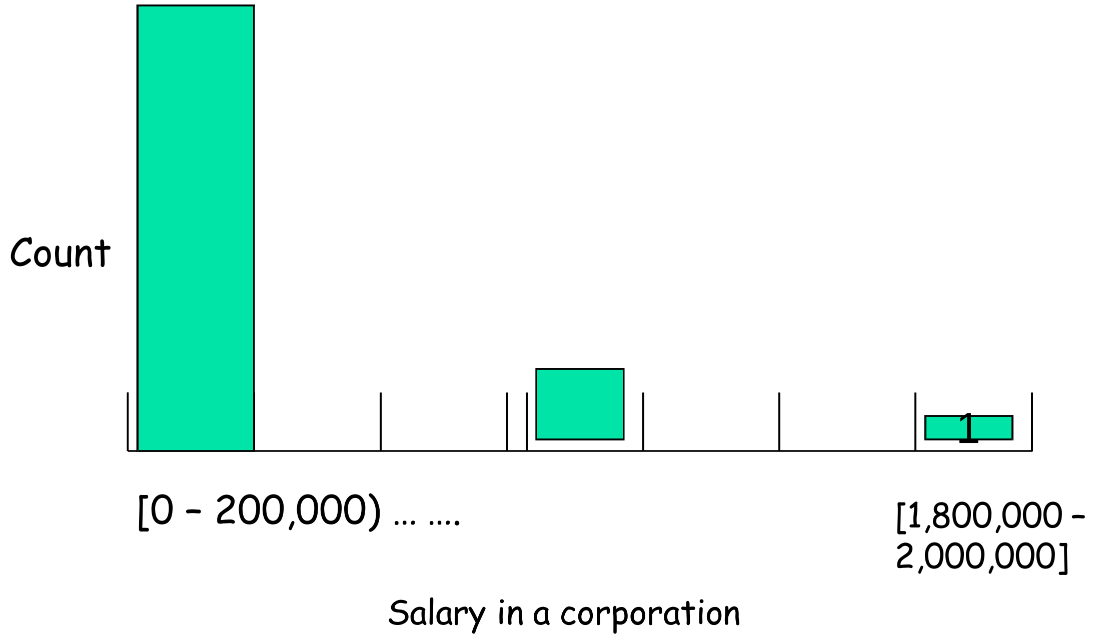
Equal-width Binning

- *It divides the range into N intervals of equal size (range): uniform grid*
- *If A and B are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.*



Equal Width, bins Low <= value < High

Equal-width Binning



Disadvantage

(a) Unsupervised

(b) Where does N come from?

(c) Sensitive to outliers

Advantage

(a) simple and easy to implement

(b) produce a reasonable abstraction of data

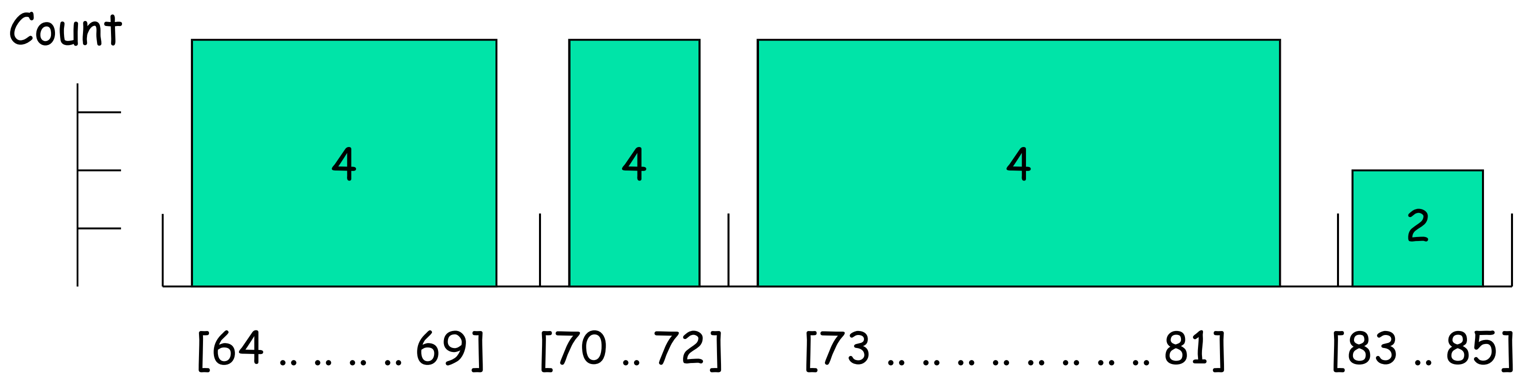
Equal-depth (or height) Binning

- It divides the range into *N* intervals, each containing *approximately* the same number of samples
  - Generally preferred because avoids clumping
  - In practice, “almost-equal” height binning is used to give more intuitive breakpoints

- Additional considerations:
  - don’t split frequent values across bins
  - create separate bins for special values (e.g. 0)
  - readable breakpoints (e.g. round breakpoints

Equal-depth (or height) Binning

Temperature values:
64 65 68 69 70 71 72 72 75 75 80 81 83 85



Equal Height = 4, except for the last bin

Discretization considerations

- Class-independent methods
  - Equal Width is simpler, good for many classes
    - can fail miserably for unequal distributions
  - Equal Height gives better results
- Class-dependent methods can be better for classification
  - Decision tree methods build discretization on the fly
  - Naïve Bayes requires initial discretization
- Many other methods exist …

Method 1R

- Developed by Holte (1993).
- It is a supervised discretization method using binning.
- After sorting the data, the range of continuous values is divided into a number of disjoint intervals and the boundaries of those intervals are adjusted based on the class labels associated with the values of the feature.
- Each interval should contain a given minimum of instances (6 by default) with the exception of the last one.
- The adjustment of the boundary continues until the next values belongs to a class different to the majority class in the adjacent interval.

1R Example

Interval contains at leas 6 elements

Adjustment of the boundary continues until the next values belongs to a class different to the majority class in the adjacent interval.

	1	2	3	4	5	6	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4
Var	65	78	79	79	81	81	82	82	82	82	82	82	83	83	83	83	83	84	84	84	84	84	84	84	84	84	85	85	85	85	85
Class	2	1	2	2	2	1	1	2	1	2	2	2	2	1	2	2	2	1	2	2	1	1	2	2	1	1	1	2	2	2	2
majority						2						2							2							1					
new class	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3

Comment: The original method description does not mention the criterion of making sure that the same value for Var is kept is the same interval (although that seems reasonable).

The results above are given by the method available in the R package Dprep.

See the following papers for more detail:

Very Simple Classification Rules Perform Well on Most Commonly Used Datasets by Robert C. Holte

The Development of Holte's 1R Classifier by Craig Nevill-Manning, Geoffrey Holmes and Ian H. Witten

Exercise

- Discretize the following values using EW and ED binning
- 13, 15, 16, 16, 19, 20, 21, 22, 22, 25, 30, 33, 35, 35, 36, 40, 45

Entropy Based Discretization

Class dependent (classification)

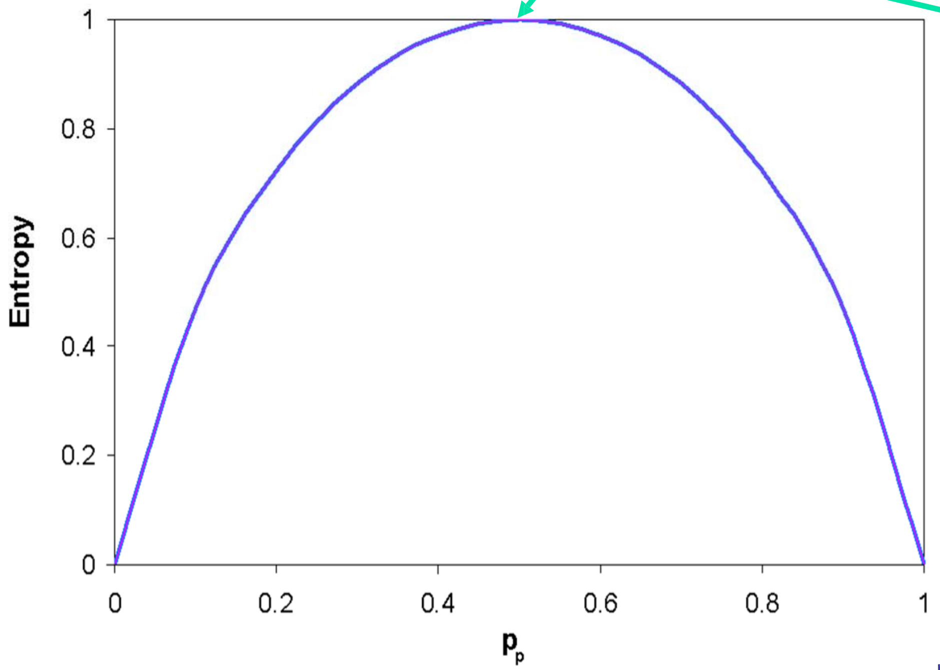
1. Sort examples in increasing order
2. Each value forms an interval ('m' intervals)
3. Calculate the entropy measure of this discretization
4. Find the binary split boundary that minimizes the entropy function over all possible boundaries. The split is selected as a binary discretization.

$$E(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$$

5. Apply the process recursively until some stopping criterion is met, e.g.,

$$Ent(S) - E(T,S) > \delta$$

Entro
py



p	1-p	Ent
0.2	0.8	0.72
0.4	0.6	0.97
0.5	0.5	1
0.6	0.4	0.97
0.8	0.2	0.72

$\log_2(2)$

$$Ent = -\sum_{c=1}^{N} p_c \cdot \log_2 p_c$$

$\log_2(3)$

p1	p2	p3	Ent
0.1	0.1	0.8	0.92
0.2	0.2	0.6	1.37
0.1	0.45	0.45	1.37
0.2	0.4	0.4	1.52
0.3	0.3	0.4	1.57
0.33	0.33	0.33	1.58

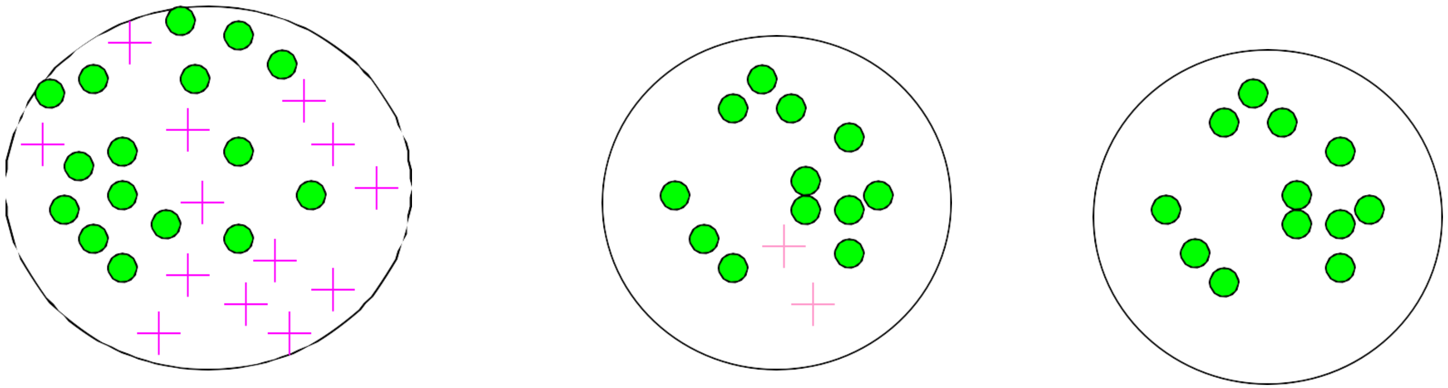
Entropy/Impurity

- S - training set, $C_1,\ldots,C_N$ classes
- Entropy E(S) - measure of the impurity in a group of examples
  - p_c - proportion of C_c in S

$$\text{Impurity}(S) = -\sum_{c=1}^{N} p_c \cdot \log_2 p_c$$

Impurity

Very impure group | **Less impure** | **Minimum impurity**



high entropy

null entropy

An example of entropy disc.

Test split temp < 71.5

Temp.	Play?
64	Yes
65	No
68	Yes
69	Yes
70	Yes
71	No
72	No
72	Yes
75	Yes
75	Yes
80	No
81	Yes
83	Yes
85	No

(4 yes, 2 no)

(5 yes, 3 no)

	yes	no
< 71.5	4	2
> 71.5	5	3

$$Ent(split\ 71.5) = \frac{6}{14} \cdot \left(\frac{4}{6} \log \frac{4}{6} + \frac{2}{6} \log \frac{2}{6} \right) + \frac{8}{14} \cdot \left(\frac{5}{8} \log \frac{5}{8} + \frac{3}{8} \log \frac{3}{8} \right) = 0.939$$

	yes	no
< 77	7	3
> 77	2	2

$$Ent(split\ 77) = \frac{10}{14} \cdot \left(\frac{7}{10} \log \frac{7}{10} + \frac{3}{10} \log \frac{3}{10} \right) + \frac{4}{14} \cdot \left(\frac{2}{4} \log \frac{2}{4} + \frac{2}{4} \log \frac{2}{4} \right) = \boxed{0.915}$$

An example (cont.)

Temp.	Play?
64	Yes
65	No
68	Yes
69	Yes
70	Yes
71	No
72	No
72	Yes
75	Yes
75	Yes
80	No
81	Yes
83	Yes
85	No

6th split

5th split

4th split

3rd split

2nd split

1st split

The method tests all split possibilities and chooses the split with smallest entropy.

In the first iteration a split at 84 is chosen.

The two resulting branches are processed recursively.

The fact that recursion only occurs in the first interval in this example is an artifact. In general both intervals have to be split.

The stopping criterion

Previous slide did not take into account the stopping criterion.

$$Ent(S) - E(T,S) > \delta$$

$$\partial > \frac{\log(N-1)}{N} + \frac{\Delta(T,S)}{N}$$

$$\Delta(T,S) = \log_2(3^c - 2) - [cEnt(S) - c_1 Ent(S_1) - c_2 Ent(S_2)]$$

c is the number of classes in S

c_1 is the number of classes in S_1

c_2 is the number of classes in S_2.

This is called the Minimum Description Length Principle (MDLP)

Exercise

- Compute the gain of splitting this data in half

Humidity	play
65	Yes
70	No
70	Yes
70	Yes
75	Yes
80	Yes
80	Yes
85	No
86	Yes
90	No
90	Yes
91	No
95	No
96	Yes

OUTLIERS

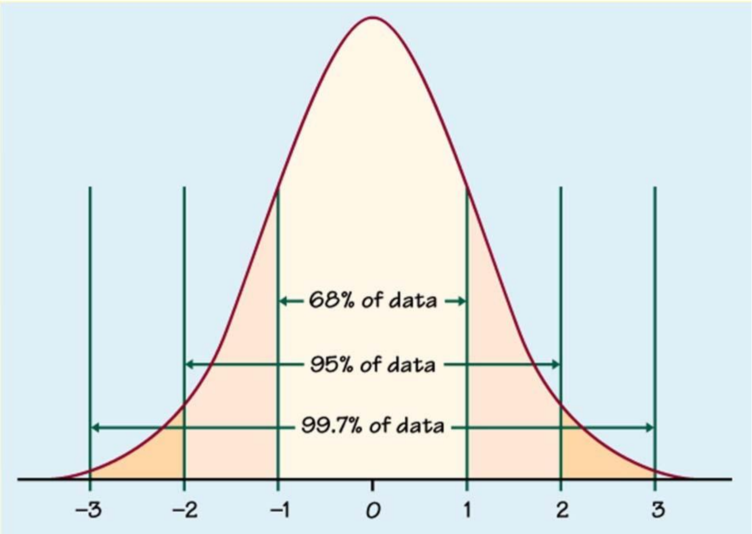
Outliers

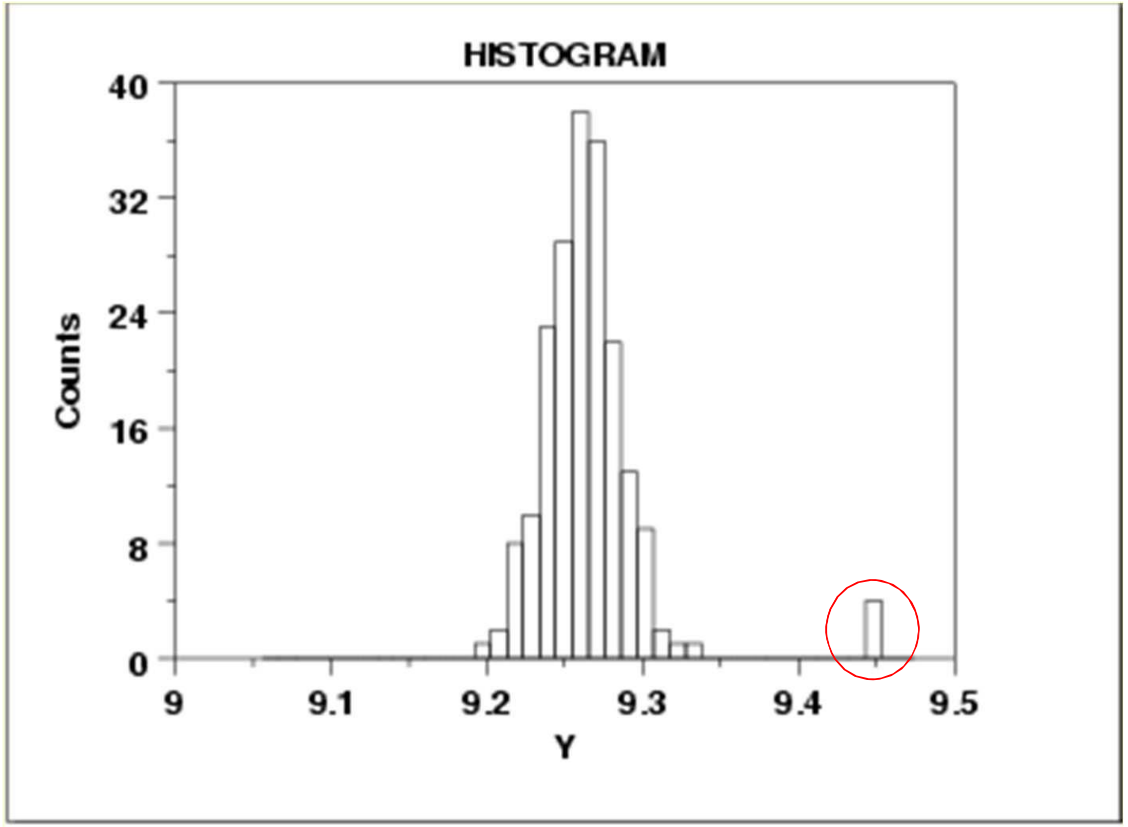
- Outliers are values thought to be out of range.
  - ***"An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism"***
  - Can be detected by standardizing observations and label the standardized values outside a predetermined bound as outliers
  - Outlier detection can be used for fraud detection or data cleaning
- Approaches:
  - do nothing
  - enforce upper and lower bounds
  - let binning handle the problem

Outlier detection

- **Univariate**
  - Compute mean and std. deviation. For k=2 or 3, x is an outlier if outside limits (normal distribution assumed)

$$(\bar{x} - ks, \bar{x} + ks)$$



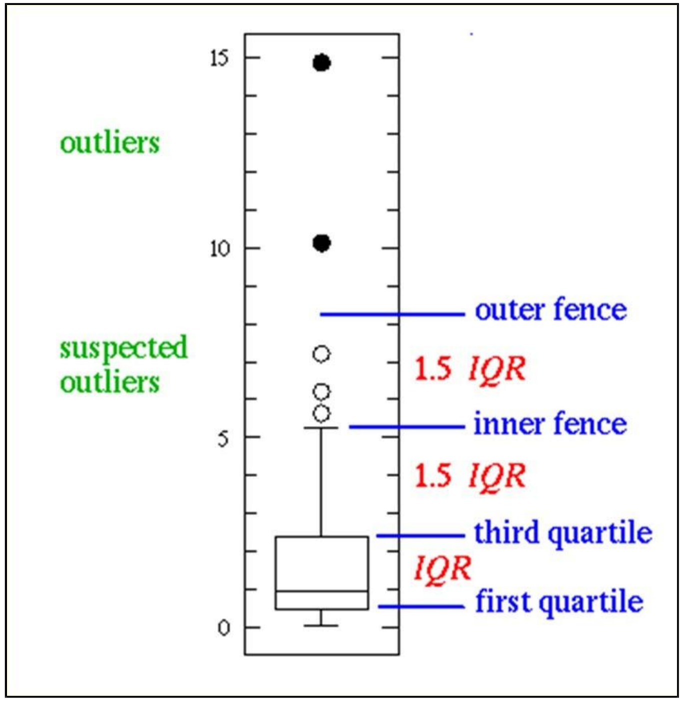

HISTOGRAM
Counts
40
32
24
16
8
0
9
9.1
9.2
9.3
9.4
9.5
Y

Outlier detection

- **Univariate**
  - Boxplot: An observation is an **extreme** outlier if

(Q1-3×IQR, Q3+3×IQR),   where IQR=Q3-Q1

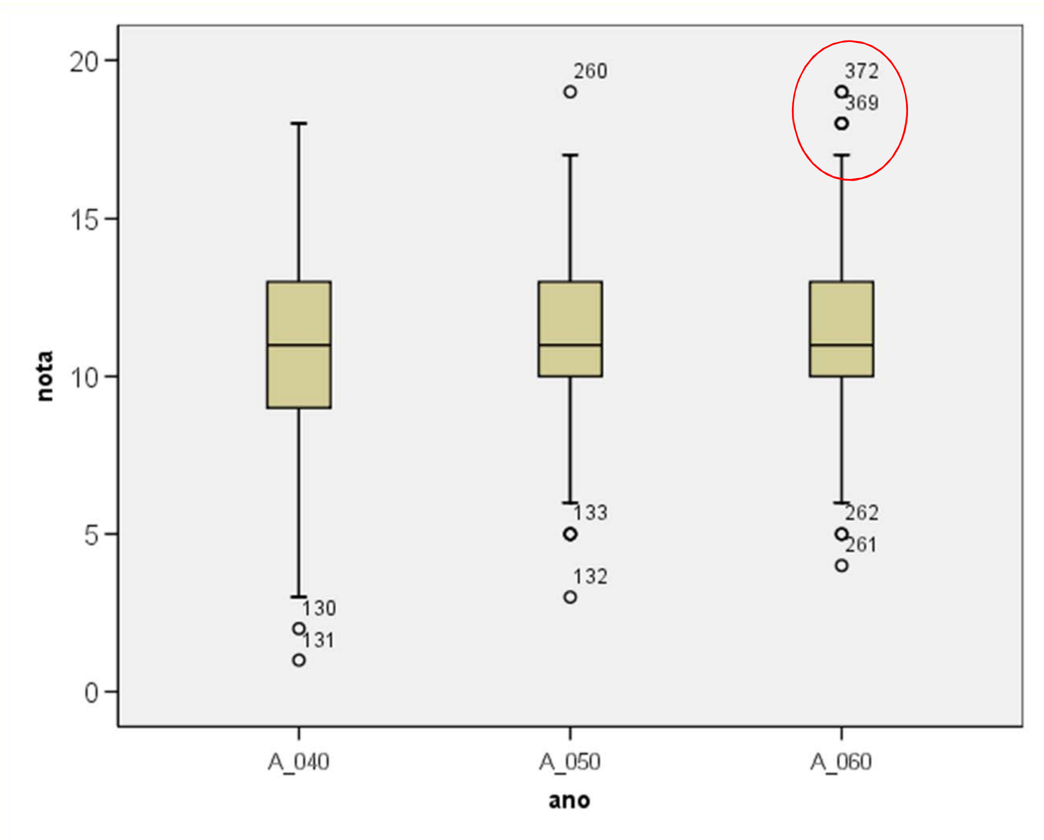
(IQR = Inter Quartile Range)

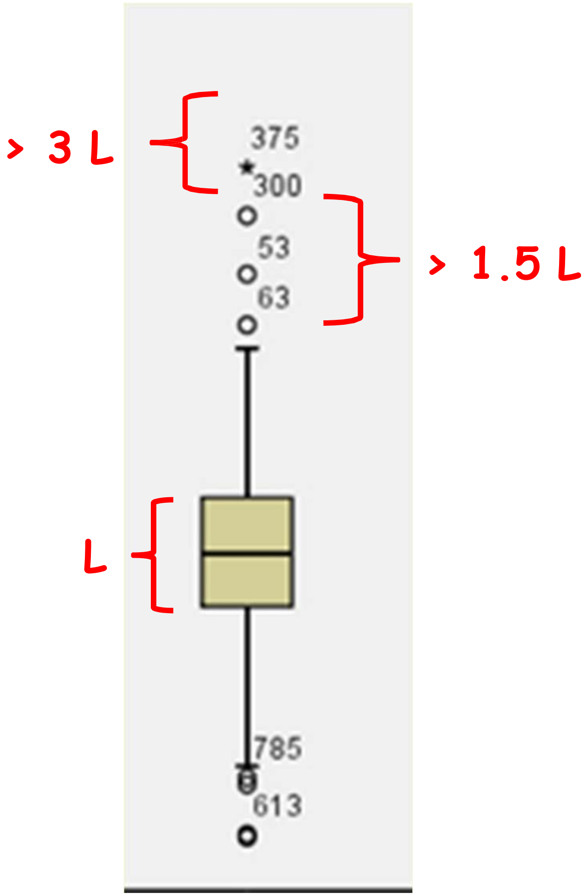


and declared a **mild** outlier if it lies outside of the interval

(Q1-1.5×IQR, Q3+1.5×IQR).

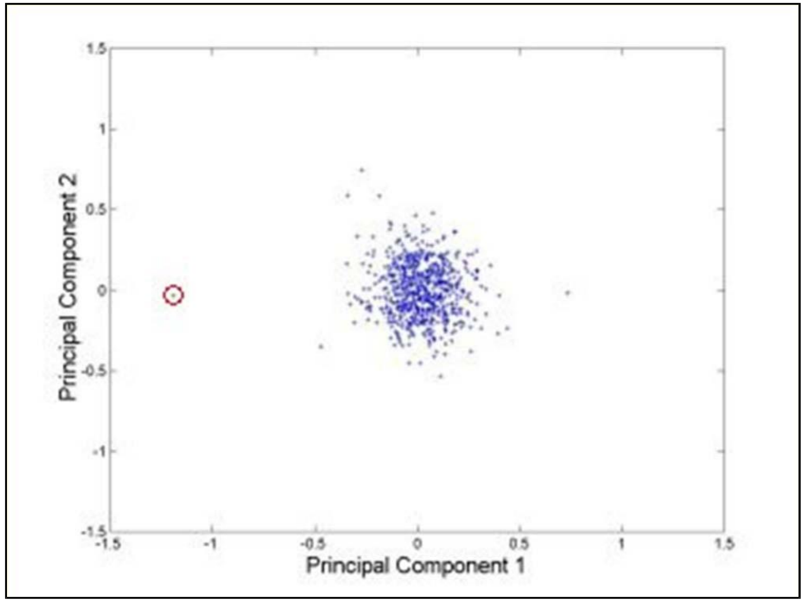
http://www.physics.csbsju.edu/stats/box2.html

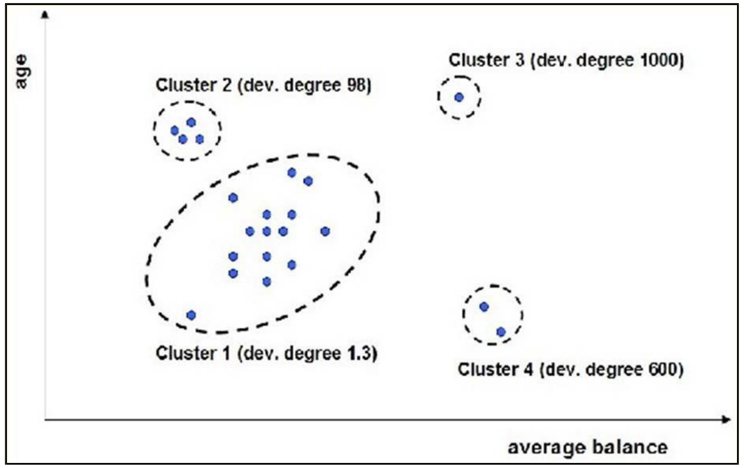

20
15
10
5
0
nota
260
372
369
133
132
262
261
130
131
A_040
A_050
A_060
ano


> 3 L
375
300
53
63
> 1.5 L
L
785
613

Outlier detection

- Multivariate
  - Clustering
    - Very small clusters are outliers



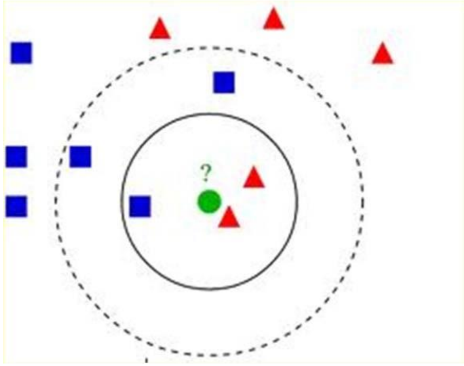


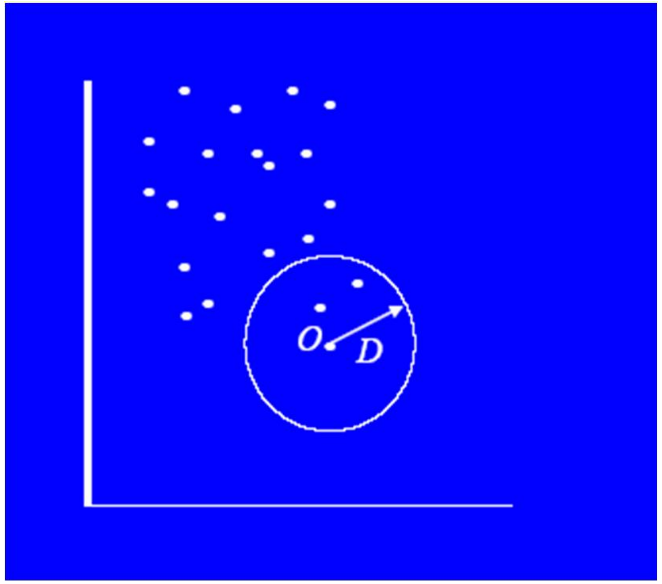
http://www.ibm.com/developerworks/data/library/techarticle/dm-0811wurst/

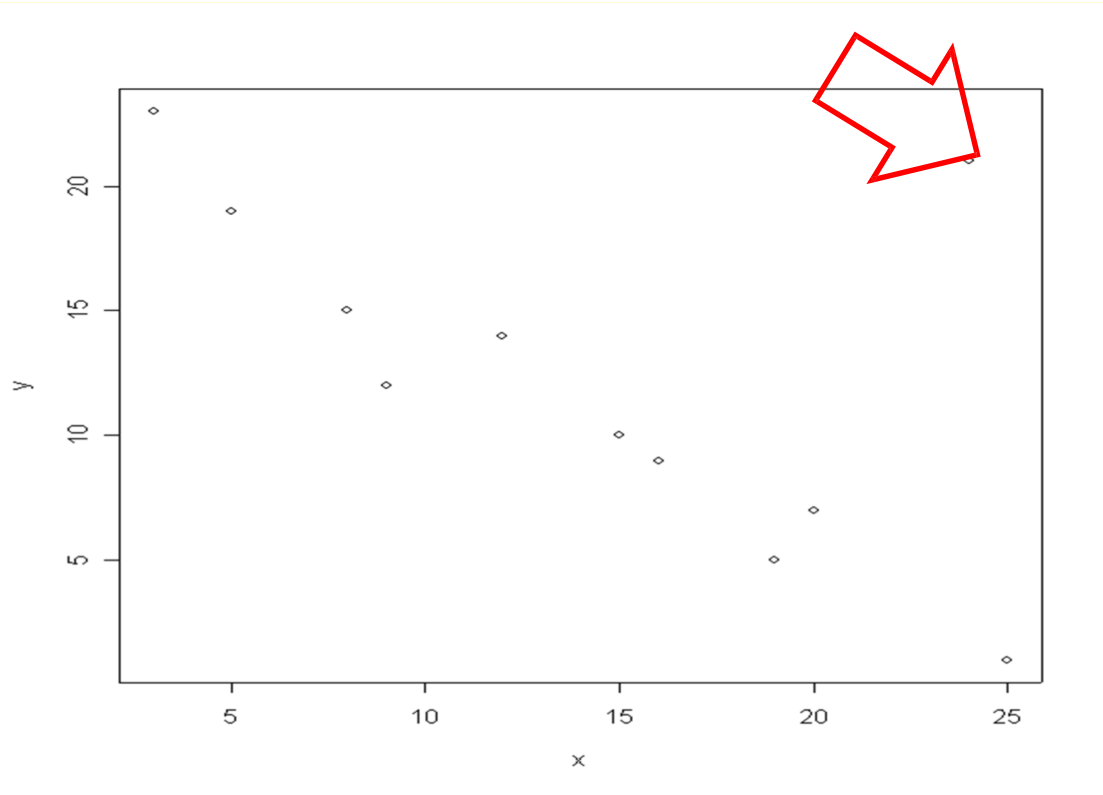
Outlier detection

- Multivariate
  - Distance based
    - An instance with very few neighbors within D is regarded as an outlier

Knn algorithm







A bi-dimensional outlier that is not an outlier in either of its projections.

DATA TRANSFORMATION

Normalization

- For distance-based methods, normalization helps to prevent that attributes with large ranges out-weight attributes with small ranges
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling

Normalization

- min-max normalization

$$v' = \frac{v - \min_v}{\max_v - \min_v}(new\_max_v - new\_min_v) + new\_min_v$$

- z-score normalization

$$v' = \frac{v - \bar{v}}{\sigma_v}$$

does not eliminate outliers

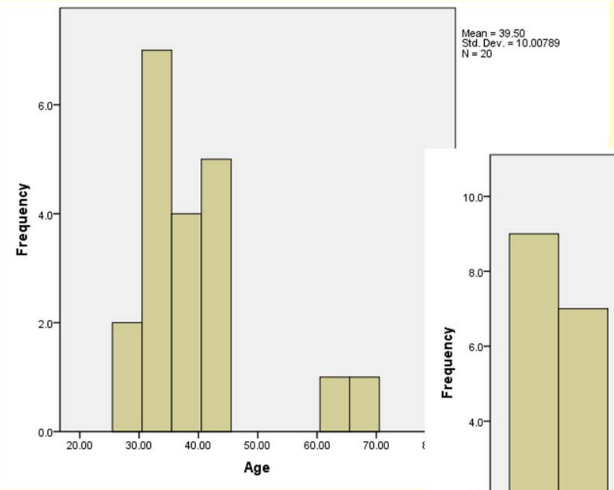
- normalization by decimal scaling

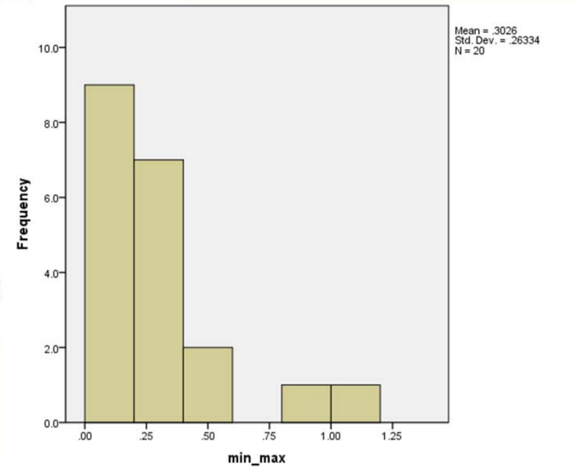
$$v' = \frac{v}{10^j}$$

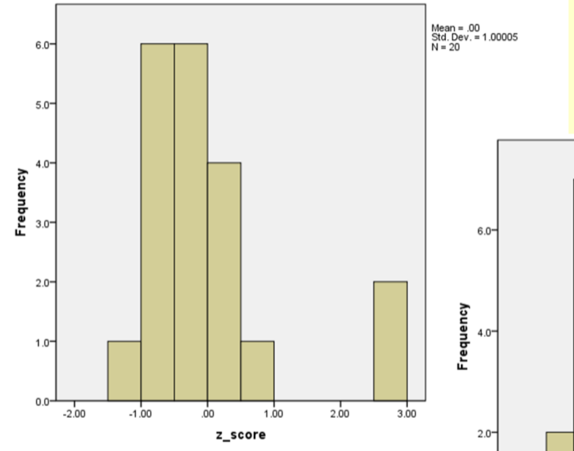
Where j is the smallest integer such that Max(| v' |)<1

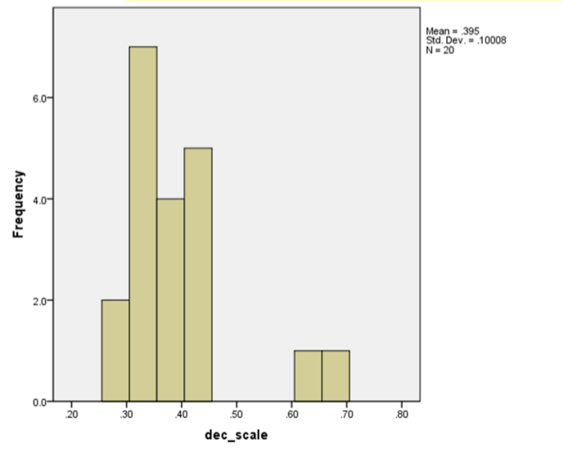
range: -986 to 917 => j=3 -986 -> -0.986 917 -> 0.917

Age	min-max (0-1)	z-score	dec. scaling
44	0.421	0.450	0.44
35	0.184	-0.450	0.35
34	0.158	-0.550	0.34
34	0.158	-0.550	0.34
39	0.289	-0.050	0.39
41	0.342	0.150	0.41
42	0.368	0.250	0.42
31	0.079	-0.849	0.31
28	0.000	-1.149	0.28
30	0.053	-0.949	0.3
38	0.263	-0.150	0.38
36	0.211	-0.350	0.36
42	0.368	0.250	0.42
35	0.184	-0.450	0.35
33	0.132	-0.649	0.33
45	0.447	0.550	0.45
34	0.158	-0.550	0.34
65	0.974	2.548	0.65
66	1.000	2.648	0.66
38	0.263	-0.150	0.38
28	minimun		
66	maximum		
39.50	avgerage		
10.01	standard deviation		









MISSING DATA

Missing Data

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred.
- Missing values may carry some information content: e.g. a credit application may carry information by noting which field the applicant did not complete

Missing Values

- There are always MVs in a real dataset
- MVs may have an impact on modelling, in fact, they can destroy it!
- Some tools ignore missing values, others use some metric to fill in replacements
  - The modeller should avoid default automated replacement techniques
    - Difficult to know limitations, problems and introduced bias
- Replacing missing values without elsewhere capturing that information removes information from the dataset

How to Handle Missing Data?

- Ignore records (use only cases with all values)
  - Usually done when class label is missing as most prediction methods do not handle missing data well
  - Not effective when the percentage of missing values per attribute varies considerably as it can lead to insufficient and/or biased sample sizes
- Ignore attributes with missing values
  - Use only features (attributes) with all values (may leave out important features)
- Fill in the missing value manually
  - tedious + infeasible?

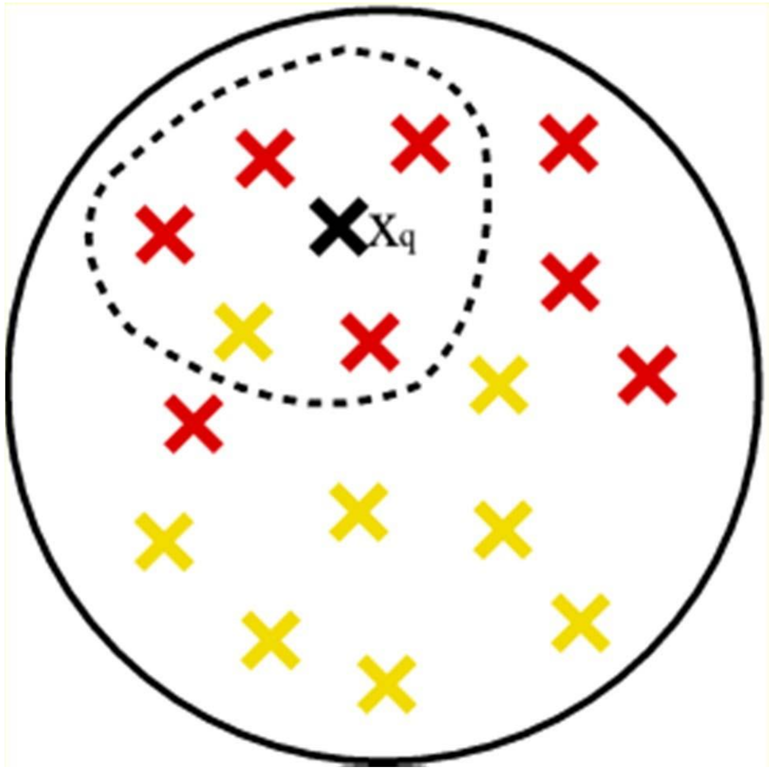
How to Handle Missing Data?

- Use a global constant to fill in the missing value
  - e.g., "unknown". (May create a new class!)
- Use the attribute mean to fill in the missing value
  - It will do the least harm to the mean of existing data
  - If the mean is to be unbiased
  - What if the standard deviation is to be unbiased?
- Use the attribute mean for all samples belonging to the same class to fill in the missing value

How to Handle Missing Data?

- Use the most probable value to fill in the missing value
  - Inference-based such as Bayesian formula or decision tree
  - Identify relationships among variables
    - Linear regression, Multiple linear regression, Nonlinear regression
  - Nearest-Neighbour estimator
    - Finding the k neighbours nearest to the point and fill in the most frequent value or the average value
    - Finding neighbours in a large dataset may be slow

Nearest-Neighbour



How to Handle Missing Data?

- Note that, it is as important to avoid adding bias and distortion to the data as it is to make the information available.
  - bias is added when a wrong value is filled-in
- No matter what techniques you use to conquer the problem, it comes at a price. The more guessing you have to do, the further away from the real data the database becomes. Thus, in turn, it can affect the accuracy and validation of the mining results.

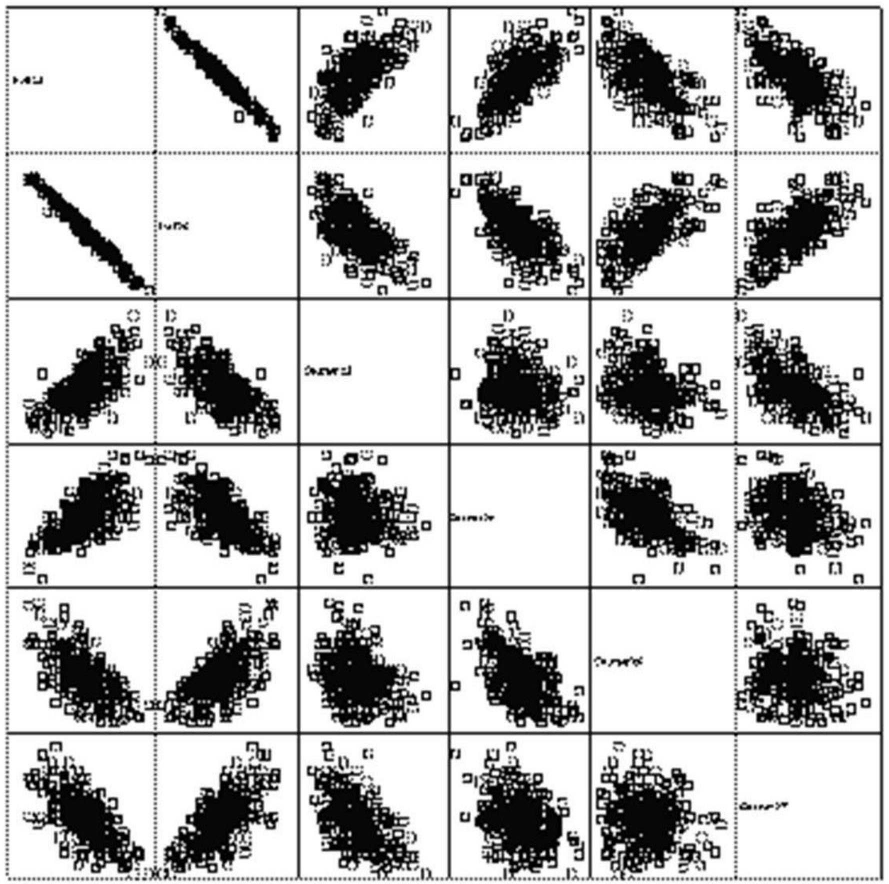
HANDLING REDUNDANCY

Handling Redundancy in Data Integration

- Redundant data occur often when integrating databases
  - The same attribute may have different names in different databases
  - False predictors are fields correlated to target behavior, which describe events that happen at the same time or *after* the target behavior
    - Example: Service cancellation date is a leaker when predicting attriters
  - One attribute may be a "derived" attribute in another table, e.g., annual revenue
  - For numerical attributes, redundancy may be detected by correlation analysis

$$r_{XY} = \frac{\frac{1}{N-1}\cdot\sum_{n=1}^{N}(x_n-\bar{x})\cdot(y_n-\bar{y})}{\sqrt{\frac{1}{N-1}\cdot\sum_{n=1}^{N}(x_n-\bar{x})^2}\cdot\sqrt{\frac{1}{N-1}\cdot\sum_{n=1}^{N}(y_n-\bar{y})^2}} \quad (-1 \le r_{XY} \le 1)$$

Scatter Matrix



SAMPLING AND UNBALANCED DATASETS

Sampling

- The cost of sampling is proportional to the sample size and not to the original dataset size, therefore, a mining algorithm's complexity is potentially sub-linear to the size of the data

- Choose a representative subset of the data
  - Simple random sampling (SRS) (with or without reposition)
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data

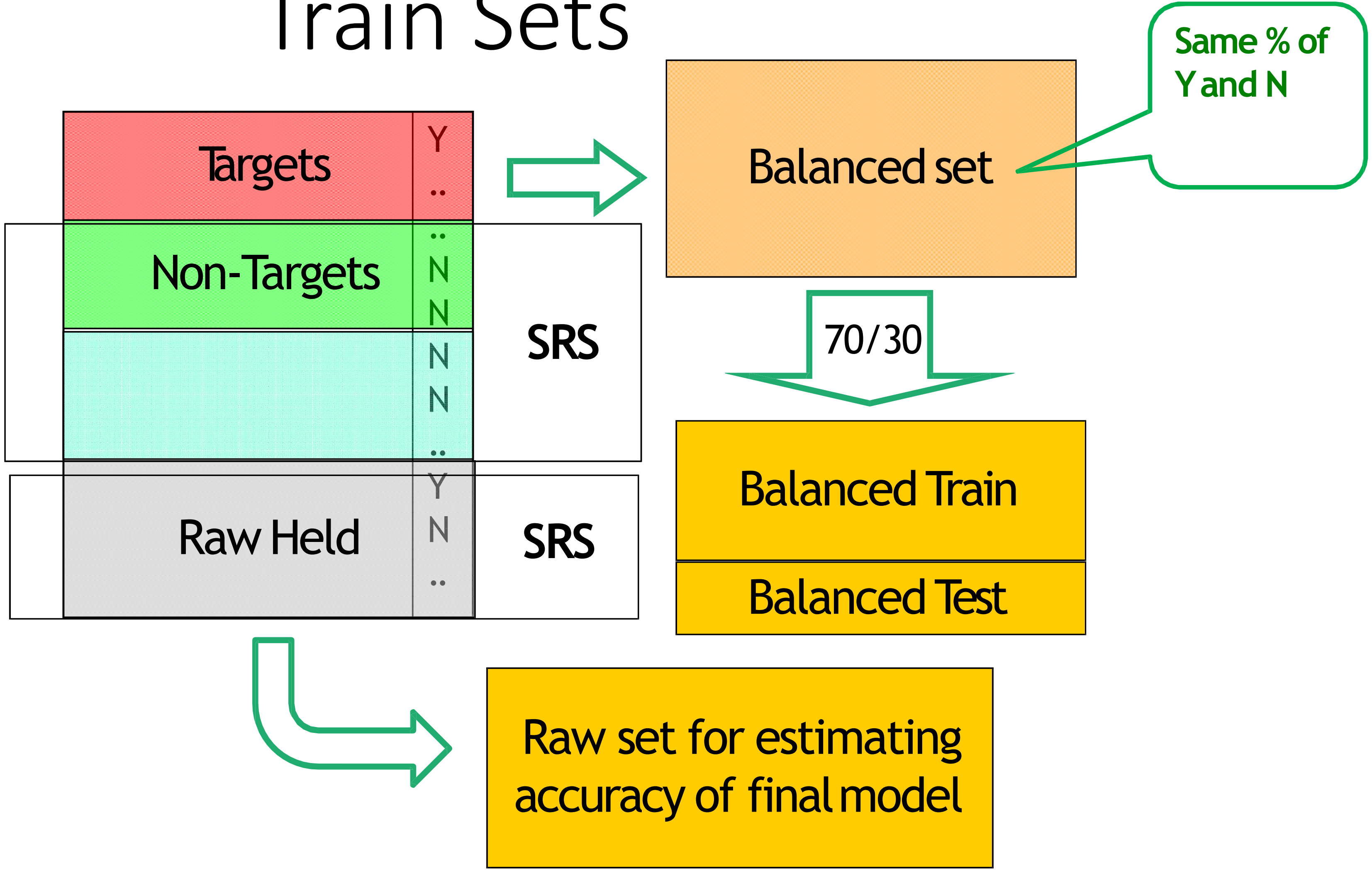
Unbalanced Target Distribution

- Sometimes, classes have very unequal frequency
  - Attrition prediction: 97% stay, 3% attrite (in a month)
  - medical diagnosis: 90% healthy, 10% disease
  - eCommerce: 99% don't buy, 1% buy
  - Security: >99.99% of Americans are not terrorists
- Similar situation with multiple classes
- Majority class classifier can be 97% correct, but useless

Handling Unbalanced Data

- With two classes: let positive targets be a minority
- Separate raw held-aside set (e.g. 30% of data) and raw train
  - put aside raw held-aside and don't use it till the final model
- Select remaining positive targets (e.g. 70% of all targets) from raw train
- Join with equal number of negative targets from raw train, and randomly sort it
- Separate randomized balanced set into balanced train and balanced test

Building Balanced Train Sets



Summ
ary

- Every real world data set needs some kind of data pre-processing
  - Deal with missing values
  - Correct erroneous values
  - Select relevant attributes
  - Adapt data set format to the software tool to be used
- In general, data pre-processing consumes more than 60% of a data mining project effort

Referenc es

- ‘Data preparation for data mining’, Dorian Pyle, 1999
- ‘Data Mining: Concepts and Techniques’, Jiawei Han and Micheline Kamber, 2000
- ‘Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations’, Ian H. Witten and Eibe Frank, 1999
- ‘Data Mining: Practical Machine Learning Tools and Techniques second edition’, Ian H. Witten and Eibe Frank, 2005
- DM: Introduction: Machine Learning and Data Mining, Gregory Piatetsky-Shapiro and Gary Parker