Software-Praktikum

NLTK & sklearn

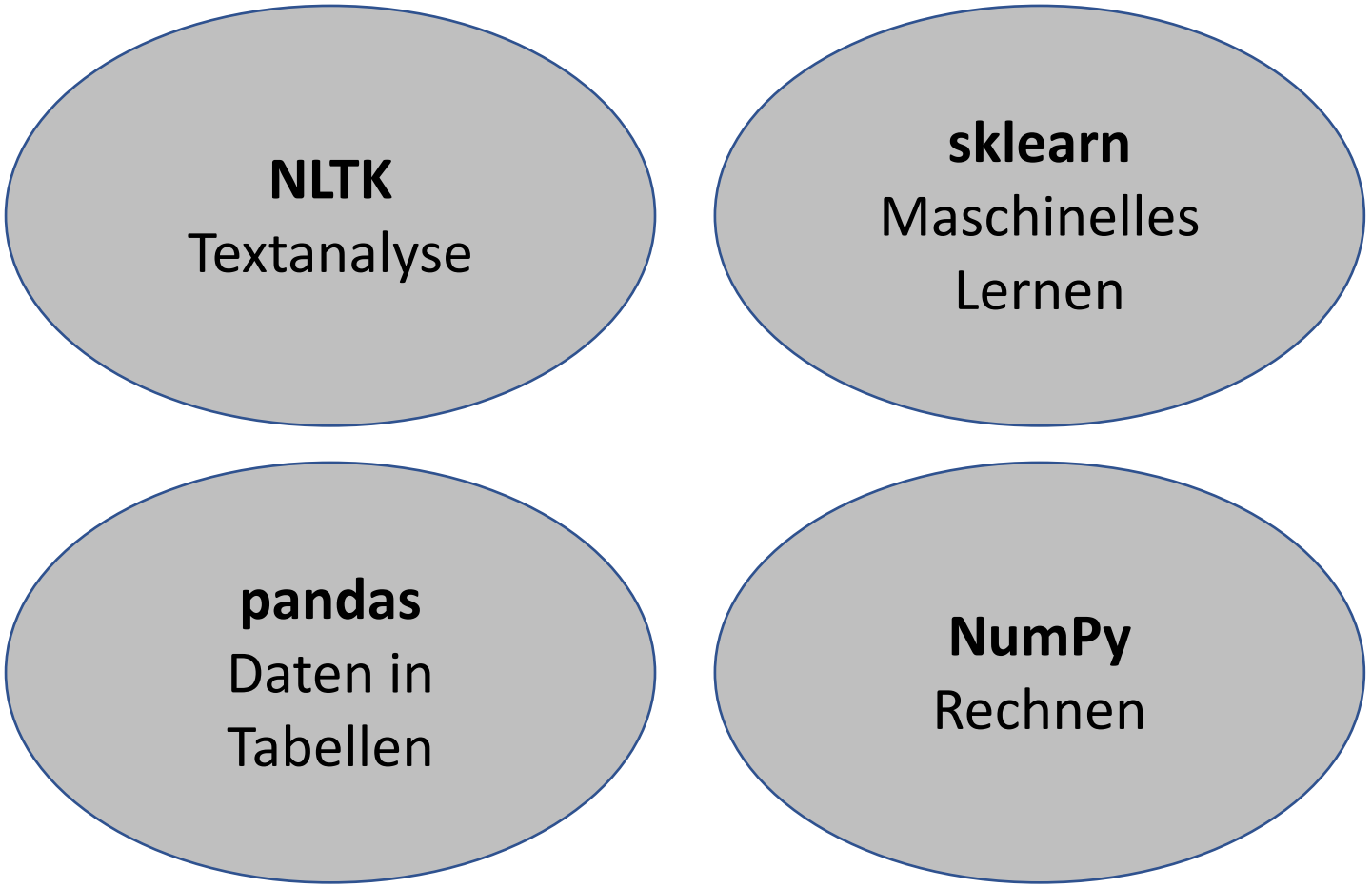
Dr. des. Johannes Hellrich

https://julielab.de

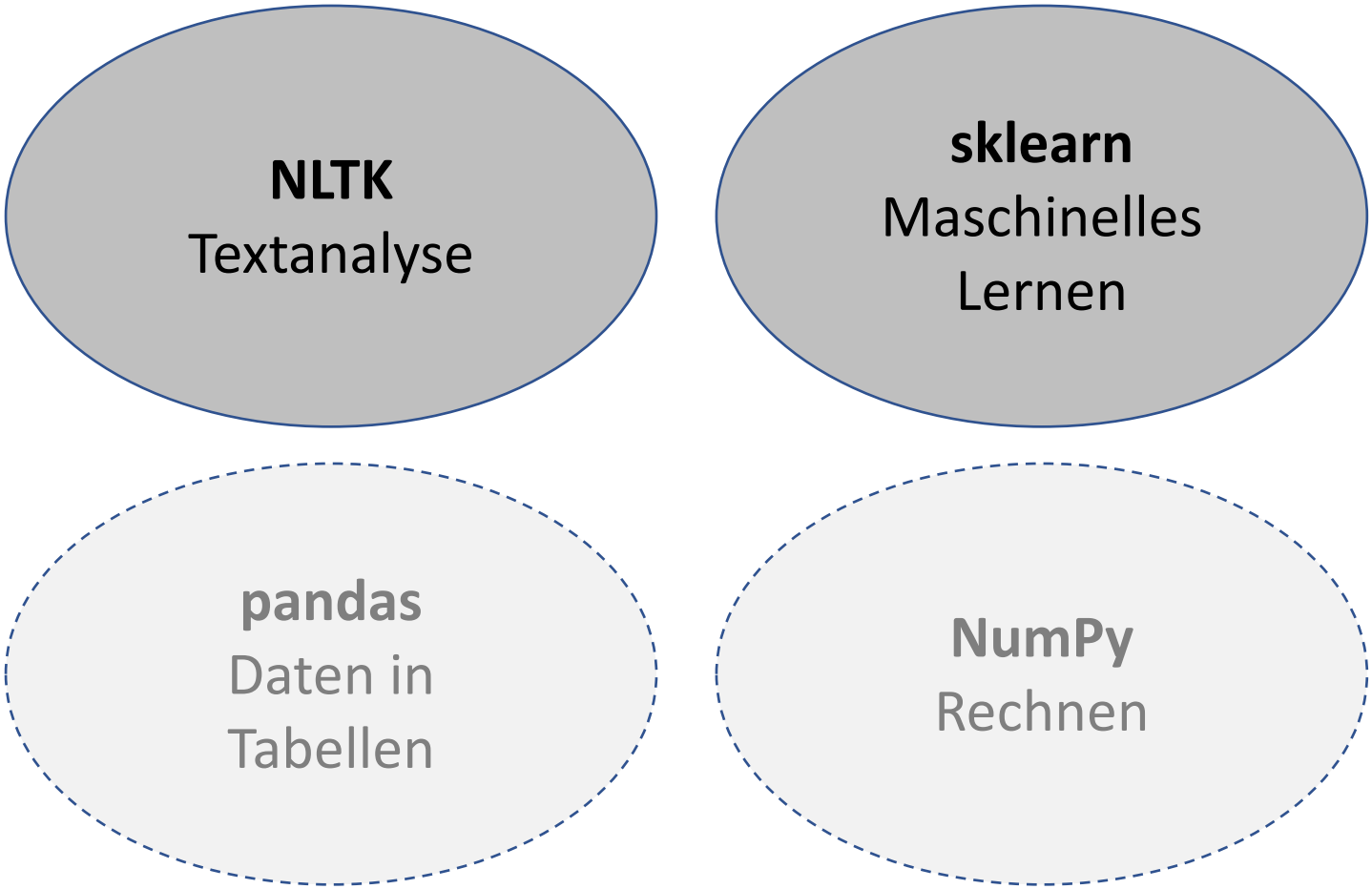
20.6.2019

Jena University Language & Information Engineering Lab

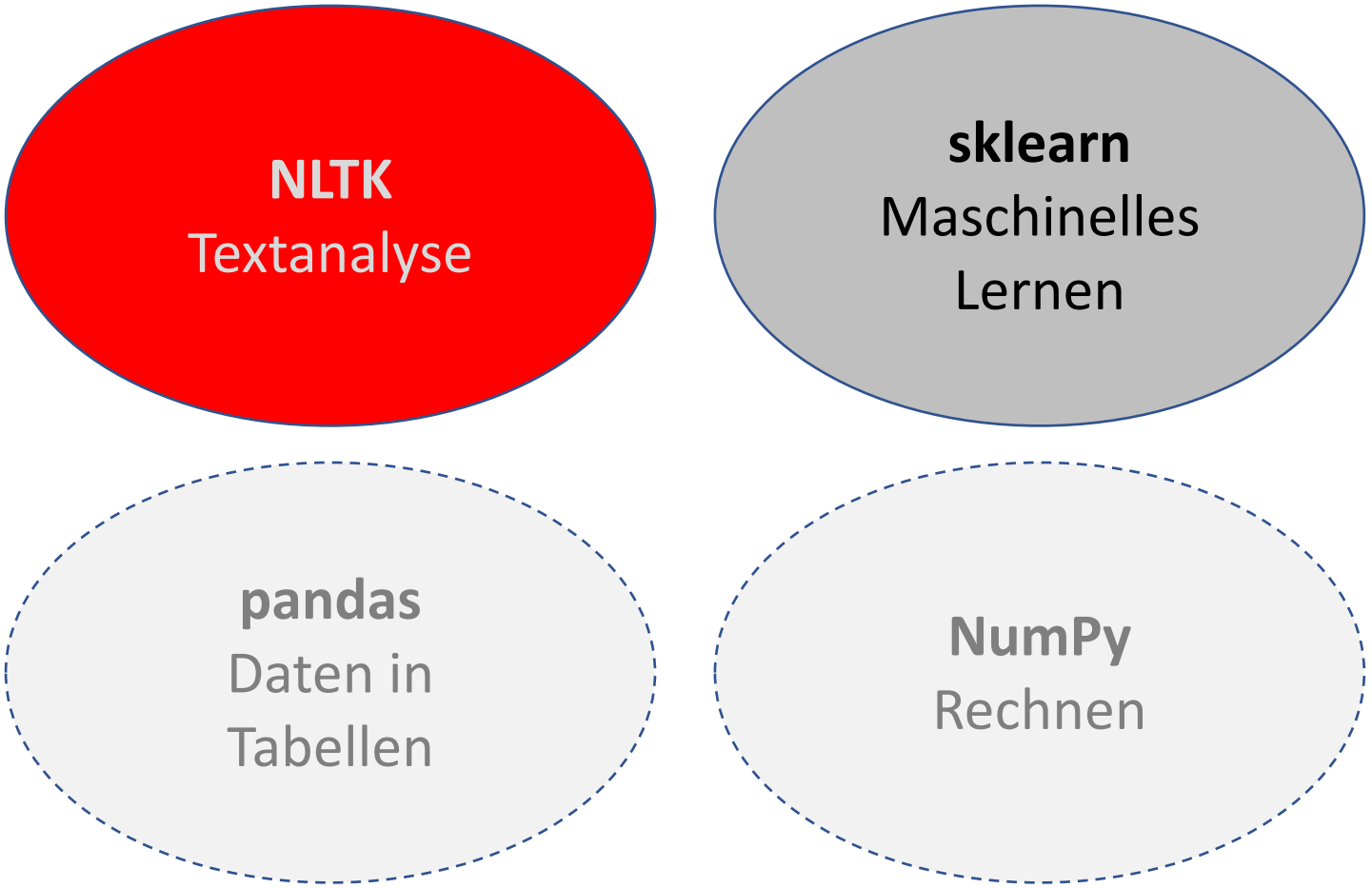
Komponenten für (modernes) NLP



Komponenten für (modernes) NLP



Komponenten für (modernes) NLP



NLTK

- Projekt: http://www.nltk.org
- E-Book: http://www.nltk.org/book/
- Zur Lehre entwickelt, wird aber auch produktiv genutzt
- Bietet Korpora und Komponenten für Verarbeitung
- String-orientiert
- Primär für Englisch

NLTK stellt Korpora & Modelle zur Verfügung

- Download in Notebooks mit `nltk.download('name‘)`
- Oder interaktiv per `nltk.download()`

```
import nltk

nltk.download()
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> l

Packages:
  [ ] abc..................... Australian Broadcasting Commission 2006
  [ ] alpino.................. Alpino Dutch Treebank
  [ ] averaged_perceptron_tagger Averaged Perceptron Tagger
  [ ] averaged_perceptron_tagger_ru Averaged Perceptron Tagger (Russian)
  [ ] basque_grammars..... Grammars for Basque
  [ ] biocreative_ppi..... BioCreAtIvE (Critical Assessment of Information
                           Extraction Systems in Biology)
```

Interaktion mit Korpora

```
import nltk
import nltk.corpus as corpora
```

```
print(corpora.brown.words()[:25])
```

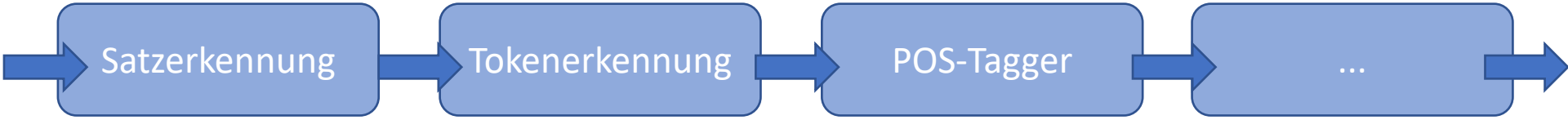
```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', "Atlanta's", 'recent', 'prima
ry', 'election', 'produced', '``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place', '.']
```

```
print(corpora.brown.tagged_sents()[0])
```

```
[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ('Grand', 'JJ-TL'), ('Jury', 'NN-TL'), ('said', 'VBD'), ('Frida
y', 'NR'), ('an', 'AT'), ('investigation', 'NN'), ('of', 'IN'), ("Atlanta's", 'NP$'), ('recent', 'JJ'), ('primary', 'NN')
, ('election', 'NN'), ('produced', 'VBD'), ('``', '``'), ('no', 'AT'), ('evidence', 'NN'), ("''", "''"), ('that', 'CS'),
('any', 'DTI'), ('irregularities', 'NNS'), ('took', 'VBD'), ('place', 'NN'), ('.', '.')]
```

- NLTK stellt Methoden zur Interaktion mit Korpora zur Verfügung:
  - `fileids()` listet alle Dateien, aus denen das Korpus besteht
  - `raw()` bietet den blanken Text
  - `words()`,`sents()`,`paras()` entsprechend die Wörter, Sätze und Paragraphen
  - bei annotierten Korpora kann über `tagged_words()`,`tagged_sents()`,`tagged_paras()` jeweils eine Liste aus Tupeln `(Token, Tag)` mit Annotationen ausgegeben werden
- siehe auch `http://www.nltk.org/howto/corpus.html` für detaillierte Informationen

NLTK Pipelines



- Zumeist sowohl sehr einfache als auch relativ moderne Lösungen vorhanden
- Segmentierung: http://www.nltk.org/book/ch03.html
- Tagging & Co.: http://www.nltk.org/book/ch05.html

Segmentierung

```
import nltk
nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
[4] nltk.sent_tokenize("This is a test sentence. And another sentence?")
```

```
['This is a test sentence.', 'And another sentence?']
```

```
[5] nltk.word_tokenize("This is a test sentence. And another sentence?")
```

```
['This', 'is', 'a', 'test', 'sentence', '.', 'And', 'another', 'sentence', '?']
```

```
for sentence in nltk.sent_tokenize("This is a test sentence. And another sentence?"):
  print(nltk.word_tokenize(sentence))
```

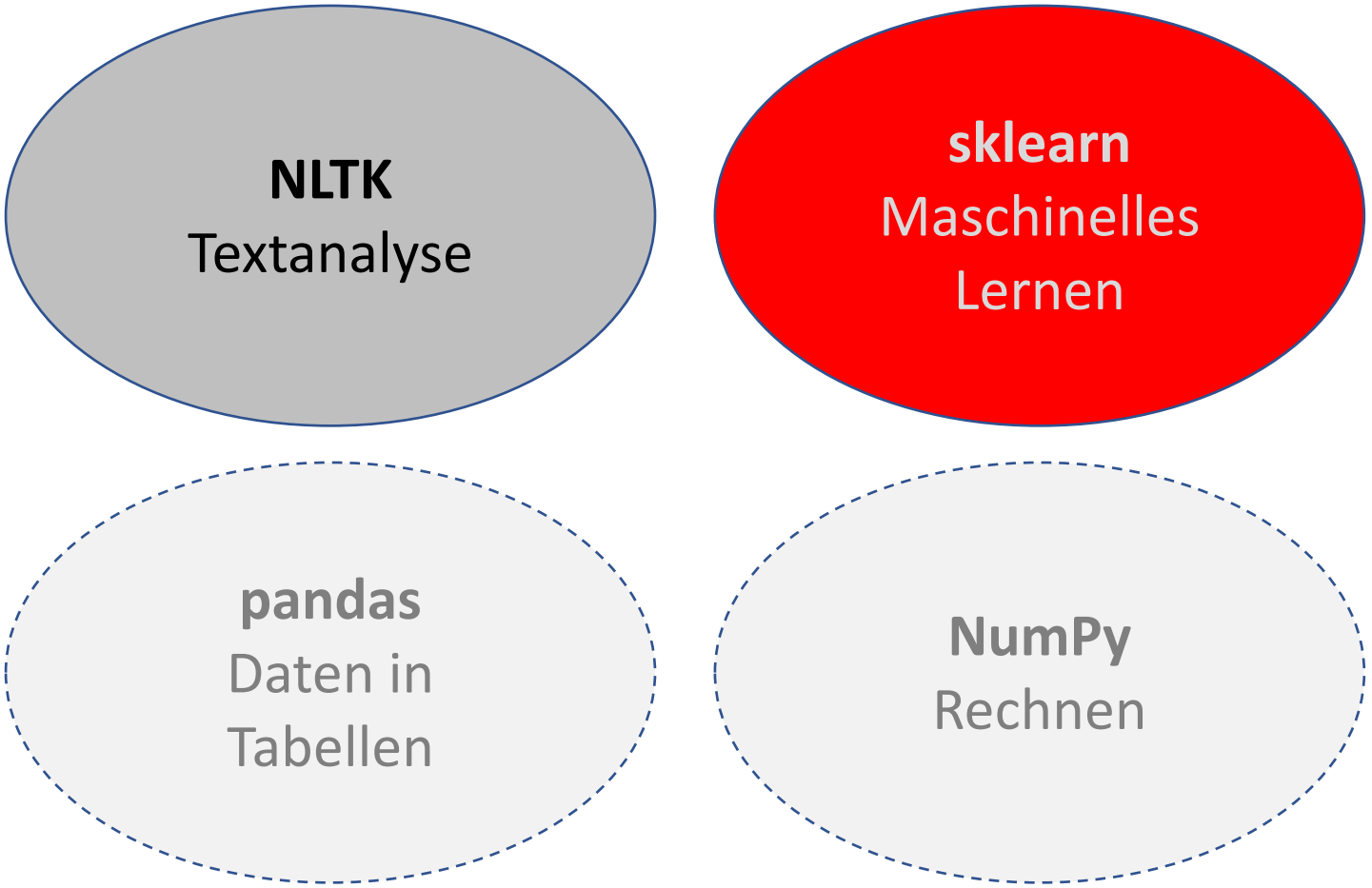
```
['This', 'is', 'a', 'test', 'sentence', '.']
['And', 'another', 'sentence', '?']
```

POS-Tagging

```python
import nltk
nltk.download('averaged_perceptron_tagger') #Installiert den NLTK default Tagger
words = nltk.word_tokenize("This is a test text.")
tagged = nltk.pos_tag(words)
print(tagged)
for word, tag in tagged: #Zugriff auf Tupel
  print(word,"\t-->\t",tag)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('test', 'NN'), ('text', 'NN'), ('.', '.')]
This    -->     DT
is      -->     VBZ
a       -->     DT
test    -->     NN
text    -->     NN
.       -->     .
```

Komponenten für (modernes) NLP



scikit learn / sklearn

- Projekt: https://scikit-learn.org/stable/
- Generelles Machine Learning
- Zugriff auf mehr und aktuellere Verfahren
- Nur sehr eingeschränkt spezifisch für Texte/NLP
- Basiert intern auf numerischen Werten

Features -> Numerisch

```
[91] import sklearn
     import numpy as np
     feature_enc = sklearn.preprocessing.OneHotEncoder()

     first_letters = np.array([["a"], ["a"], ["b"], ["c"]])
     print(first_letters)

     feature_enc.fit(first_letters)
     encoded_features = feature_enc.transform(first_letters)

     for letter in [["a"], ["b"], ["c"]]:
       vector = feature_enc.transform([letter]).toarray()
       print(letter,vector)
```

```
[['a']
 ['a']
 ['b']
 ['c']]
['a'] [[1. 0. 0.]]
['b'] [[0. 1. 0.]]
['c'] [[0. 0. 1.]]
```

Klassifikation

```
labels = ["d","d","m","w"]

from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(feature_enc.transform(first_letters),labels)
classifier.predict(feature_enc.transform([["a"]]))
```

```
array(['d'], dtype='<U1')
```

Intuition für Verfahren des maschinellen Lernens

Naive Bayes

Gesucht:

$$\text{p(Klasse} \mid \text{Feature}_1, \text{Feature}_2, \ldots)$$

Bayes Theorem:

$$\text{p(Klasse|Features)} = \frac{p(Klasse)p(Features|Klasse)}{p(Features)}$$

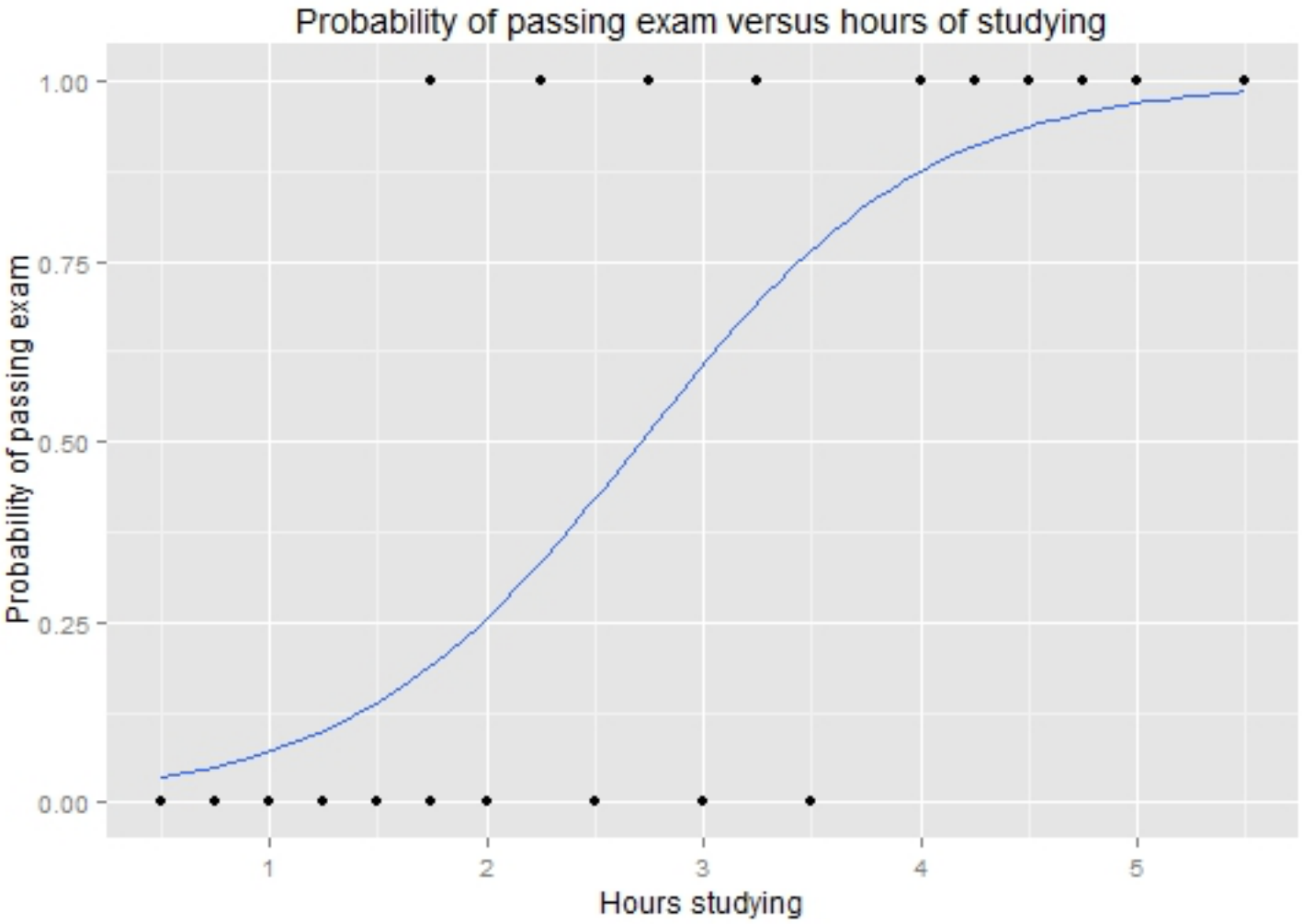
Nenner quasi konstant:

$$\text{p(Klasse|Features)} = p(Klasse)p(Features|Klasse)$$

Angenommen Features unabhängig:

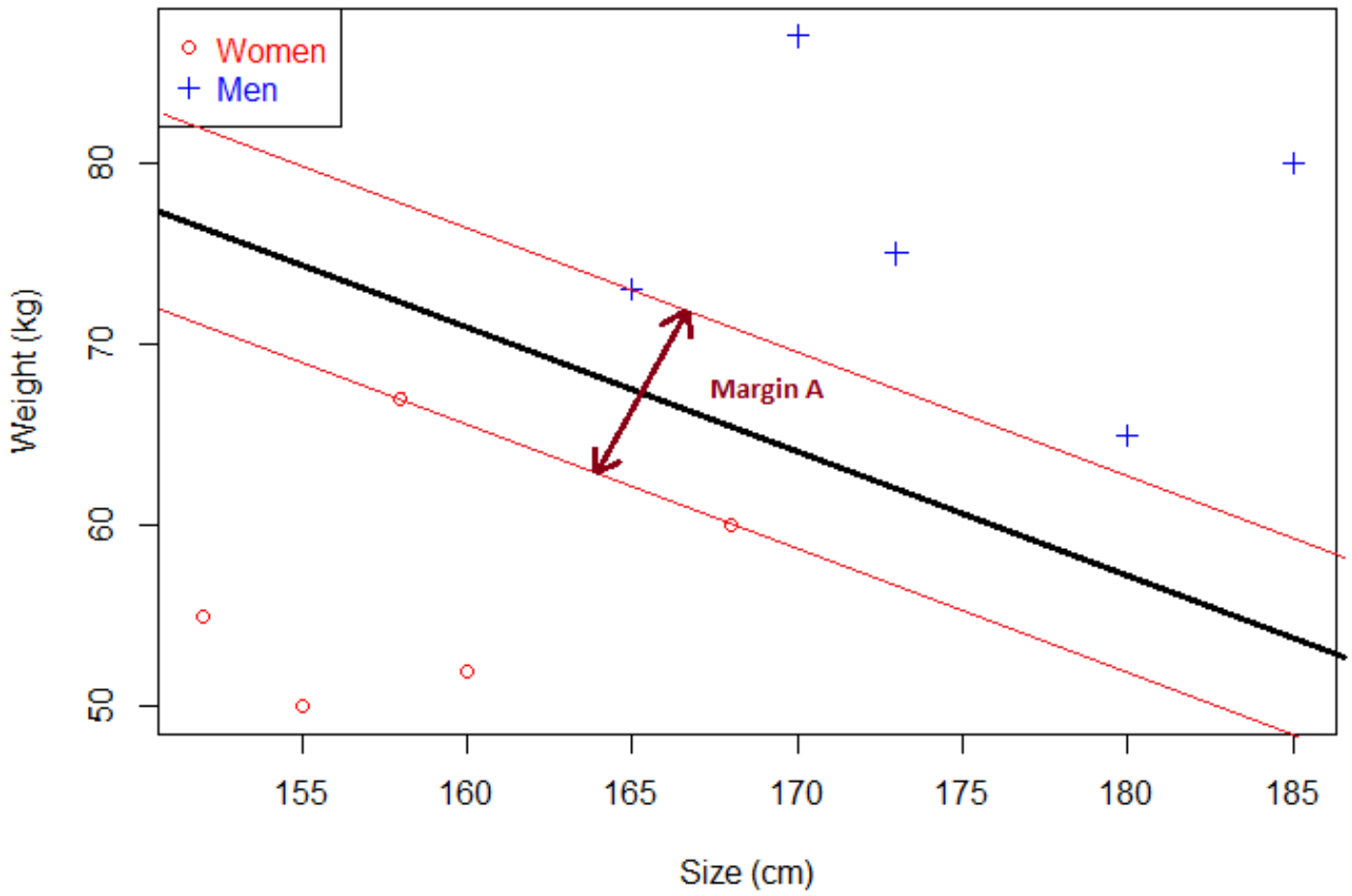
$$\text{p(Klasse|Features)} = p(K)p(F_1|K)\,p(F_2|K)\,p(F_3|K)\ldots$$

Logistische Regression



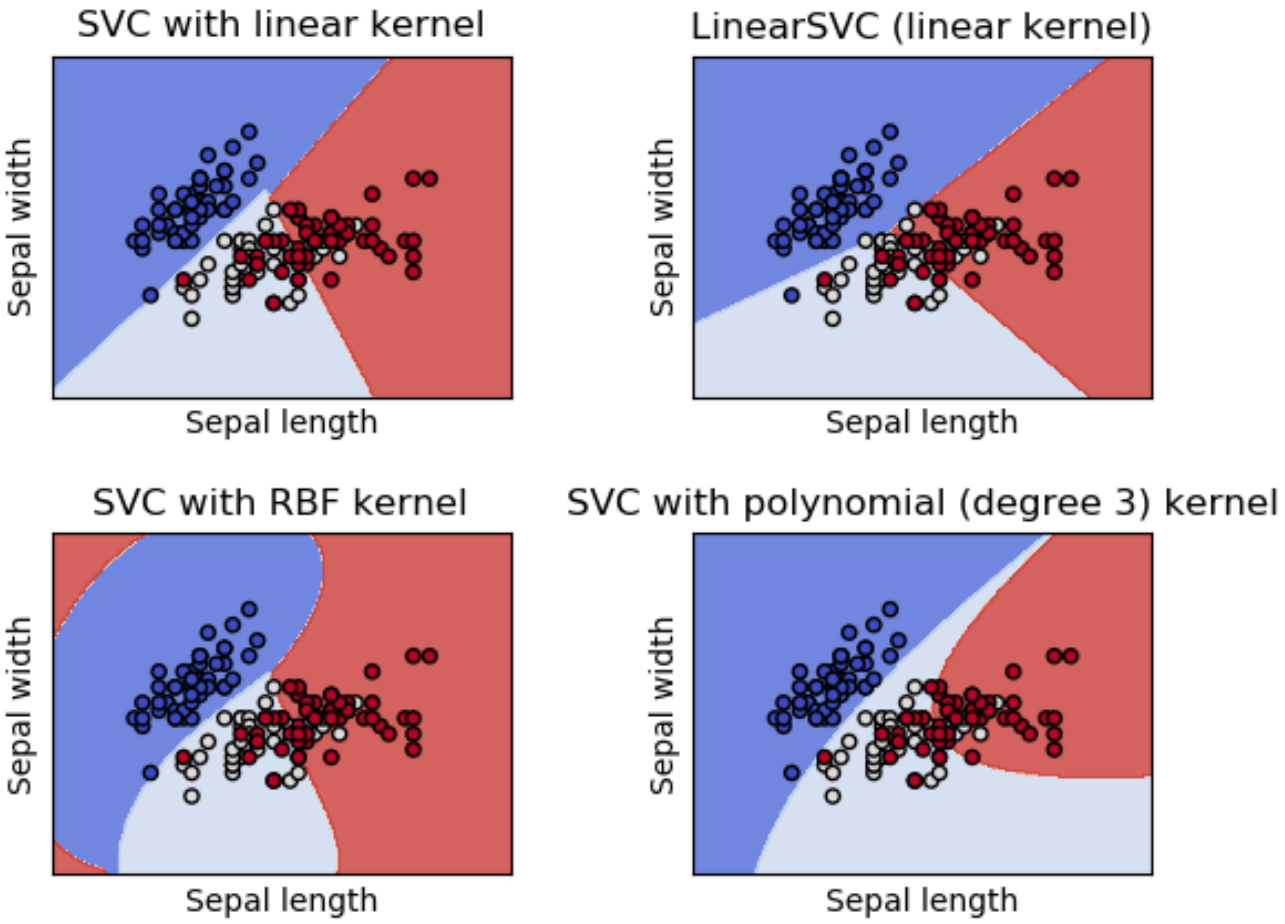
https://en.wikipedia.org/wiki/Logistic_regression

SVM I/II



https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/

SVM II/II



https://scikit-learn.org/stable/modules/svm.html#svm-classification

Faustregeln

- Jeder Algorithmus hat Stärken und Schwächen
- i.d.R. liefern SVMs die beste Kombination aus Geschwindigkeit und Genauigkeit unter den gezeigten Verfahren
- Ein Plot der Daten kann sich lohnen, eventuell ist keine lineare Trennung möglich -> SVM mit Kernel