

# **Feature Engineering**

WS 23

# Pandas Intro

# Series vs DataFrame

Eine Spalte eines DataFrames ist ein Series Objekt

Rückgabe eines Series Objekts mit [ ]

Rückgabe eines DataFrame Objekts mit [[ ]]

# Ersten Blick auf DataFrame

`df.head()`

`df.tail()`

`df.describe()`

`df.index`

`df.sort_index()`

`df.sort_values()`

Achtung: DataFrames sind mutable

# Auswahl

Mit slicing erhält man Zeilen: `df[0:3]`. Rückgabe: DataFrame

Die Schweizer Taschenmesser: `.loc`, `.iloc`. Rückgabe: It depends

Für einzelne Werte: `.at`, `.iat`. Rückgabe: einzelner Wert

# Boolsche Masken

Erst mit Boolescher Anweisung eine Lochmaske erstellen

Dann auf Daten anwenden

# Neue Werte setzen

Am einfachsten mit .at oder mit .loc

Vorsicht beim Abschneiden von Series Indizes

# Fehlende Werte

Wichtigste Methoden: `.dropna()`, `.fillna()`, `isna()`

# Eigene Transformation

Wichtigste Methoden: `.agg()`, `.transform()`

# DataFrames verkleben

Wichtigste Methoden: concat, merge

Ganz schlechte Idee: DataFrames zeilenweise mit concat aufbauen

# Group By

Implementiert das Paradigma split apply combine

Rückgabe: Bei mehr als einer group by Spalte DataFrame mit Multi-Indizes, kann tricky sein

# DataFrames schmäler machen

Wenn Observables zu Spaltennamen geworden sind, helfen `melt` und `stack`

Unterschied: `melt` generiert eine neue Spalte, `stack` einen neuen Index

# DataFrames breiter machen

Wenn Spaltennamen zu Observanzen geworden sind, helfen  
`df.pivot()` oder `pd.pivot_table()`

Unterschied: `pd.pivot_table()` hat deutlich mehr Optionen

# Kategorien

Eigener Datentyp: `.astype('category')`

Hauptvorteil: Unterstützt Kategorien, die nicht im Datensatz vorhanden sind

# Tidy Data

# What is tidy data?

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Source: <https://vita.had.co.nz/papers/tidy-data.pdf>

# Examples

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Source: <https://vita.had.co.nz/papers/tidy-data.pdf>

# Typical Errors

1. Column headers are values, not variable names
2. Multiple variables are stored in one column

*Source: <https://vita.had.co.nz/papers/tidy-data.pdf>*

# Example for Error Type 1

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Source: <https://vita.had.co.nz/papers/tidy-data.pdf>

# Example for Error Type 2

machinedate	key	value
M	1/1/22 t_min	15,00
M	1/1/22 t_max	17,00
M	1/2/22 t_min	15,20
M	1/2/22 t_max	17,40
M	1/3/22 t_min	15,90
M	1/3/22 t_max	18,30

Machine	date	t_min	t_max
M	1/1/22	15	17,00
M	1/2/22	15,20	17,40
M	1/3/22	15,90	18,30

# Pandas methods

`pandas.melt`

Unpivot data

From wide to long

`df.pivot`

Pivot data

From long to wide

`df.groupby`

Aggregate

From long to short

# Ihre Aufgabe

Bringen Sie die folgenden Datensätze in Ordnung:

- TidyData.xlsx
- TidyData Error Type Two.xlsx
- pew.xlsx

# Ihre Aufgabe

Bringen Sie die folgenden Datensätze in Ordnung:

- tb.csv

# Data Sources

# Flat Files und Tabellen

Die Klassiker:

- Excel, csv, parquet etc.
- Tabellen in RDBMS

# File Formate für hierarchische Daten

Bekanntestes Format ist hdf5

- Jedes Element eines n-dimensionalen Datensatzes darf beliebig komplex sein
- Erlaubt die Speicherung von Metadaten direkt an den Daten

Beliebt zum Speichern von annotierten Bilddaten

# File Formate für hierarchische Daten

Ihre Aufgabe:

- Lesen Sie die Daten `mnist.hdf5` in python ein
- Lesen Sie das erste Bild als array aus. Finden Sie das erste Label
- Stellen Sie das gefundene Array als Bild dar und überprüfen Sie, dass das Label korrekt ist

# REST APIs

- A client-server architecture made up of clients, servers, and resources, with requests managed through HTTP.
- Stateless client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.
- Cacheable data that streamlines client-server interactions.
- A uniform interface between components so that information is transferred in a standard form.
  - resources requested are identifiable and separate from the representations sent to the client.
  - resources can be manipulated by the client via the representation they receive because the representation contains enough information to do so.
  - self-descriptive messages returned to the client have enough information to describe how the client should process it.
  - hypertext/hypermedia is available, meaning that after accessing a resource the client should be able to use hyperlinks to find all other currently available actions they can take.
- A layered system that organizes each type of server (those responsible for security, load-balancing, etc.) involved the retrieval of requested information into hierarchies, invisible to the client.

Source: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>

# OpenAPI ist eine Spezifikation von REST

- “The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to HTTP APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.”
- *Source: <https://swagger.io/specification/>*

# Aufgabe zu OpenAPI

- Gehen Sie auf <https://petstore3.swagger.io/>
- Finden Sie die ID eines verkauften Tiers
- Finden Sie die Details zu diesem Tier

# **ODATA ist eine spezielle REST Implementierung**

Slogan: “The best way to rest”

Highlights:

- “OData helps you focus on your business logic”
- “OData RESTful APIs are easy to consume”
- “The OData metadata, a machine-readable description of the data model of the APIs, ...”

# **ODATA ist eine spezielle REST Implementierung**

ODATA ist besonders im Business Intelligence Context weit verbreitet

Startpunkt ist die Basis URL, gefolgt von \$metadata

Zum Experimentieren empfiehlt sich ein Programm wie Postman oder die VS Code Extension Thunder Client

# Ihre Aufgabe zu ODATA

- Laden Sie die Daten aus dem People Endpunkt des Trip Pin Service mit python
- Nutzen Sie pandas um die Daten in eine DataFrame zu bringen
- Sind die Daten tidy?

# JSON Dateien verarbeiten

- Mit pd.read\_json() erzeugen Sie ein DataFrame, dass die Struktur der json Datei identisch abbildet

```
l_nested = [{"name": "Alice", "age": 25, "id": {"x": 2, "y": 8}},  
            {"name": "Bob", "id": {"x": 10, "y": 4}}]
```

```
print(pd.DataFrame(l_nested))  
#      name    age          id  
# 0   Alice  25.0  {'x': 2, 'y': 8}  
# 1     Bob    NaN  {'x': 10, 'y': 4}
```

# JSON Dateien verarbeiten

- Mit pd.json\_normalize() können Sie einfache Strukturen flach machen

```
l_nested = [{"name": "Alice", "age": 25, "id": {"x": 2, "y": 8}},  
            {"name": "Bob", "id": {"x": 10, "y": 4}}]
```

```
print(pd.json_normalize(l_nested))  
#      name    age  id.x  id.y  
# 0  Alice  25.0     2     8  
# 1    Bob    NaN    10     4
```

# JSON Dateien verarbeiten

- Für komplexe Strukturen muss man stückweise vorgehen oder Ebenen definieren

```
l_complex = [{"label": "X",
    "info" : {"n": "nx", "m": "mx"},
    "data": [{"a": 1, "b": 2},
              {"a": 3, "b": 4}]},
    {"label": "Y",
    "info" : {"n": "ny", "m": "my"},
    "data": [{"a": 10, "b": 20},
              {"a": 30, "b": 40}]]
```

```
print(pd.json_normalize(l_complex, record_path='data'))
#      a   b
# 0    1   2
# 1    3   4
# 2   10  20
# 3   30  40
```

```
print(pd.json_normalize(l_complex, record_path='data',
                           meta='label'))
#      a   b   label
# 0    1   2       X
# 1    3   4       X
# 2   10  20      Y
# 3   30  40      Y
```

```
print(pd.json_normalize(l_complex, record_path='data',
                           meta=['label', ['info', 'n'], ['info', 'm']],
                           sep='_'))
#      a   b   label   info_n   info_m
# 0    1   2       X     nx     mx
# 1    3   4       X     nx     mx
# 2   10  20      Y     ny     my
# 3   30  40      Y     ny     my
```

# JSON Dateien verarbeiten

- Arrays in der json Datei sind schwierig aufzulösen, da sie variable Längen haben

```
l_list = [ {'label': 'X',
            'info' : {'n': 'nx', 'm': 'mx'},
            'data': [1, 2, 3]},
           {'label': 'Y',
            'info' : {'n': 'ny', 'm': 'my'},
            'data': [10, 20, 30, 40]}]
```

```
pd.json_normalize(l_list)
```

✓ 0.0s

	label	data	info.n	info.m
0	X	[1, 2, 3]	nx	mx
1	Y	[10, 20, 30, 40]	ny	my

# Ihre Aufgabe zu JSON

- Vollziehen Sie die Beispiele oben im Notebook `read_json_starter.ipynb` nach

# Ihre Aufgabe zu JSON

- Laden Sie die Daten aus dem People Endpunkt des Trip Pin Service mit python
- Nutzen Sie pandas um die Daten in eine DataFrame zu bringen
- Machen Sie die Daten so tidy wie möglich

# Detecting Redundant Attributes

# Redundante Spalten

Ein Spalte ist redundant, wenn sie von einer anderen Spalte oder einer Menge von anderen Spalten abgeleitet werden kann

In Formeln: Es gibt eine Funktion  $f$  so dass die Werte in Spalte  $y$  aus den Werten in Spalte  $x$  berechnet werden können

# Hinweis auf Redundanz

Sind die Werte in zwei Spalten nicht unabhängig voneinander, kann das ein Hinweis auf Redundanz sein

Reminder: Zwei Zufallsvariablen  $X$  und  $Y$  heißen unabhängig, falls

$$P(Y = y \mid X = x) = P(Y = y)$$

# Wann schaden abhängige Variablen?

$X$  sind die Eingangsdaten,  $y$  die Zielvariable

- Abhängige Variablen in  $X$  können Modellen schaden
- Zur Vorhersage von  $y$  brauche ich dagegen Variablen in  $X$ , die nicht unabhängig von  $y$  sind

# Unabhängigkeit für kategoriale Variablen

Unabhängigkeit zwischen zwei Variablen lässt sich mit dem  $\chi^2$ -Test überprüfen

# Wiederholung $\chi^2$ Test

Starten mit einer Tabelle:

Ziel/Note	4	5	6	Total
Grades	49	50	69	168
Popular	24	36	38	98
Sports	19	22	28	69
Total	92	108	135	335

Daraus berechnen wir erwartete Werte,

$$\text{z.Bsp. } 168 \cdot \frac{92}{335} = 46,14$$

Ziel/Note	4	5	6
Grades	46,14	54,16	67,70
Popular	26,91	31,59	39,49
Sports	18,95	22,24	27,81

# Wiederholung $\chi^2$ Test

Ziel/Note	4	5	6	Total
Grades	49	50	69	168
Popular	24	36	38	98
Sports	19	22	28	69
<b>Total</b>	<b>92</b>	<b>108</b>	<b>135</b>	<b>335</b>

Ziel/Note	4	5	6
Grades	46,14	54,16	67,70
Popular	26,91	31,59	39,49
Sports	18,95	22,24	27,81

Die quadratischen Differenzen der Zellen ergeben die Test Statistik:

$$\chi^2 = \sum \frac{(obs - exp)^2}{exp}$$

# Wiederholung $\chi^2$ Test

$H_0$  beim  $\chi^2$ -Test: Die Variablen sind unabhängig

$p$ -Wert: Wahrscheinlichkeit für Daten unter der Annahme, dass  $H_0$  gilt

# Ihre Aufgabe zu Unabhängigkeit

- Laden Sie die Datei `nhl_short.pckl`
- Suchen Sie sich zwei Kandidaten aus dem Datensatz
- Erstellen Sie eine contingency table für diese Daten
- Prüfen Sie mit dem chi Quadrat Test, ob die Variablen unabhängig sind
- Würden Sie die Spalten entfernen?

# Bonusaufgabe zu Unabhängigkeit

- Was macht `from sklearn.feature_selection import chi2`?
- In welcher Situation wendet man diese Funktion an?

# Redundanz für numerische Variablen

Linearer Zusammenhang lässt sich mit dem Korrelationskoeffizienten überprüfen

Reminder: Der empirische Korrelationskoeffizient zweier Variablen ist als

$$\text{Corr}(X, Y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2(y_i - \bar{y})^2}}$$

definiert. Wichtigste Eigenschaft für uns:

$$Y = aX + b \text{ fast sicher genau dann wenn } \text{Corr}(X, Y) = 1$$

# Zusammenhang Unabhängigkeit und Korrelation

Unabhängig impliziert Unkorreliert, denn:

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}} \text{ und } \text{Cov}(X, Y) = E[XY] - E[X]E[Y].$$

Falls  $X$  und  $Y$  unabhängig, gilt  $E[XY] = E[X]E[Y]$ , also wird der Zähler Null.

# Aufgabe: Beispiele für Korrelation vs Kausalität

- Gehen Sie auf <https://www.tylervigen.com/spurious-correlations> und suchen Sie Ihr Lieblingsbeispiel
- Machen Sie sich den Unterschied zwischen Korrelation und Kausalität klar

# Ihre Aufgabe zu Korrelation

- Laden Sie die Datei `nhl_short.pkl`
- Suchen Sie sich zwei numerische Kandidaten aus dem Datensatz
- Berechnen Sie den Korrelationskoeffizienten

# Ihre Aufgabe zu Korrelation

- Laden Sie die Datei `nhl_short.pkl`
- Suchen Sie sich mindestens vier numerische Spalten aus dem Datensatz
- Plotten Sie die Korrelationsmatrix

# Bonusaufgabe zu Korrelation

- Was macht die Funktion `from sklearn.feature_selection import r_regression?`
- In welcher Situation wenden Sie diese Funktion an?

# Detecting Duplicate Rows

# Duplizierte Zeilen sind ein Problem

- Führen zu mehr Daten als nötig
- Verlangsamen Algorithmen
- Führen zu Inkonsistenzen in Ergebnissen

# Duplizierte Zeilen erkennen: Die einfachen Fälle

- Bei eindeutiger ID im Datensatz
- Falls alle Werte identisch sind (`df.drop_duplicates()`)

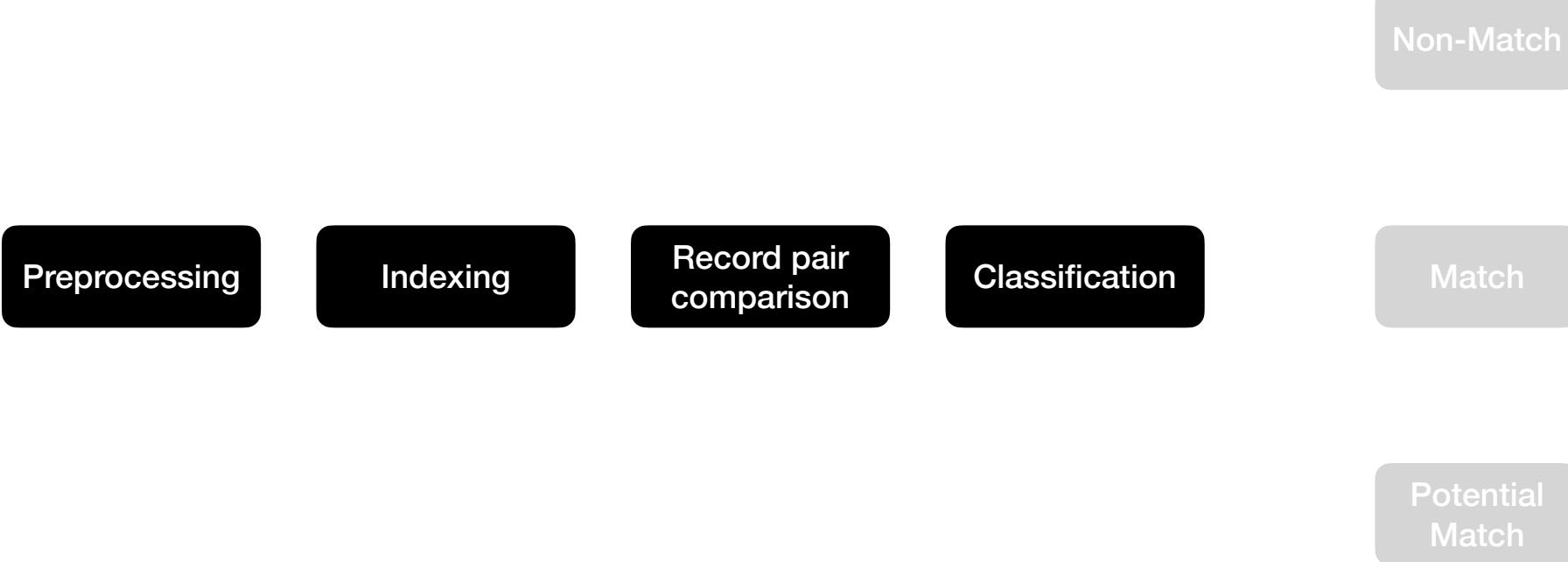
# Duplizierte Zeilen erkennen: Die schwierigen Fälle

- Ähnliche, aber nicht gleiche kategoriale Variablen. Z.Bsp. DEU, GER, Deutschland für das gleiche Land
- Tipfehler bei Handeingaben
- Schreibweisen bei Adressen

# Zentrale Probleme bei der Erkennung

- Jeden mit jedem vergleichen erzeugt quadratischen Aufwand
- Wie vergleicht man zwei Strings?
- Wann nennen wir zwei nicht exakt gleiche numerische Werte gleich?

# High Level Vorgehen



# Beispiel: Zwei Datenquellen

Database A

RecID	Surname	GivenName	Street	Suburb	Postcode	State	DateOfBirth
a1	Smith	John	42 Miller St	O'Connor	2602	A.C.T.	12-11-1970
a2	Neighan	Joanne	Brown Pl	Dickson	2604	ACT	8 Jan 1968
a3	Meyer	Marie	3/12-14 Hope Cnr	SYDNEY	2050	NSW	01-01-1921
a4	Smithers	Lyn	Browne St	DIXON	2012	N.S.W.	13/07/1970
a5	Nguyen	Ling	1 Milli Rd	Nrth Sydeny	2022	NSW	10/08/1968
a6	Faulkner	Christine	13 John St	Glebe	2037	NSW	02/23/1981
a7	Sandy	Robert	RMB 55/326 West St	Stuart Park	2713	NSW	7/10/1970

Database B

RecID	Name	Address	BYear	BMonth	BDay
b1	Meier, Mary	14 (App 3) Hope Corner, Sydney 2000	1927	4	29
b2	Janice Meyer	Bryan St, O'Connor ACT 2604	1968	11	20
b3	Jonny Smith	47 Miller Street, 2619 Canberra ACT	1970	12	11
b4	Lyng Nguyen	1 Millie Road, 2002 North Sydney, NSW	1968	8	10
b5	Kristina Fawkner	13 St John Street, 2031 Glebe	1981	2	23
b6	Bob Santi	55 East St; Stuart's Point; NSW 2113	1970	12	11
b7	Lynette Cain	6 / 12 Hope Corner, 2020 Sydney N.S.W.	1970	7	13

# Beispiel: Preprocessing

**Database A**

RecID	Surname	GivenName	Street	Suburb	Postcode	State	DateOfBirth
a1	Smith	John	42 Miller St	O'Connor	2602	A.C.T.	12-11-1970
a2	Neighan	Joanne	Brown Pl	Dickson	2604	ACT	8 Jan 1968
a3	Meyer	Marie	3/12-14 Hope Cnr	SYDNEY	2050	NSW	01-01-1921
a4	Smithers	Lyn	Browne St	DIXON	2012	N.S.W.	13/07/1970
a5	Nguyen	Ling	1 Milli Rd	Nrth Sydeny	2022	NSW	10/08/1968
a6	Faulkner	Christine	13 John St	Glebe	2037	NSW	02/23/1981
a7	Sandy	Robert	RMB 55/326 West St	Stuart Park	2713	NSW	7/10/1970

**Database A – Cleaned and standardised**

RecID	GivenName	Surname	Gender	StrPrefix	StrNum	StrName	StrType	Suburb	Postcode	State	BDay	BMonth	BYear
a1	john	smith	m		42	miller	street	oconnor	2602	act	12	11	1970
a2	joanne	neighan	f			brown	place	dickson	2604	act	8	1	1968
a3	mary	meier	f	3	12-14	hope	corner	sydney	2050	nsw	1	1	1921
a4	lynette	smithers	f			browne	street	dixon	2012	nsw	13	7	1970
a5	ling	nguyen	?		1	milli	road	north sydney	2022	nsw	10	8	1968
a6	christine	faulkner	f		13	john	street	glebe	2037	nsw	23	2	1981
a7	robert	sandy	m	rmb 55	326	west	street	stuart park	2713	nsw	7	10	1970

**Database B**

RecID	Name	Address	BYear	BMonth	BDay
b1	Meier, Mary	14 (App 3) Hope Corner, Sydney 2000	1927	4	29
b2	Janice Meyer	Bryan St, O'Connor ACT 2604	1968	11	20
b3	Jonny Smith	47 Miller Street, 2619 Canberra ACT	1970	12	11
b4	Lyng Nguyen	1 Millie Road, 2002 North Sydney, NSW	1968	8	10
b5	Kristina Fawkner	13 St John Street, 2031 Glebe	1981	2	23
b6	Bob Santi	55 East St; Stuart's Point; NSW 2113	1970	12	11
b7	Lynette Cain	6 / 12 Hope Corner, 2020 Sydney N.S.W.	1970	7	13

**Database B – Cleaned and standardised**

RecID	GivenName	Surname	Gender	StrPrefix	StrNum	StrName	StrType	Suburb	Postcode	State	BDay	BMonth	BYear
b1	mary	meier	f	apt 3	14	hope	corner	sydney	2000	nsw	29	4	1927
b2	janice	meier	f			bryan	street	oconnor	2604	act	20	11	1968
b3	john	smith	m		47	miller	street	canberra	2619	act	11	12	1970
b4	lyng	nguyen	?		1	millie	road	north sydney	2002	nsw	10	8	1968
b5	kristina	fawkner	f		13	saint john	street	glebe	2037	nsw	23	2	1981
b6	robert	santi	m		55	east	street	stuarts point	2113	nsw	11	12	1970
b7	lynette	cain	f	6	12	hope	corner	sydney	2020	nsw	13	7	1970

# Beispiel: Indexing mit Blocking

**Database A**

RecID	Surname	GivenName	Street	Suburb	Postcode	State	DateOfBirth
a1	Smith	John	42 Miller St	O'Connor	2602	A.C.T.	12-11-1970
a2	Neighan	Joanne	Brown Pl	Dickson	2604	ACT	8 Jan 1968
a3	Meyer	Marie	3/12-14 Hope Cnr	SYDNEY	2050	NSW	01-01-1921
a4	Smithers	Lyn	Browne St	DIXON	2012	N.S.W.	13/07/1970
a5	Nguyen	Ling	1 Milli Rd	Nrth Sydeny	2022	NSW	10/08/1968
a6	Faulkner	Christine	13 John St	Glebe	2037	NSW	02/23/1981
a7	Sandy	Robert	RMB 55/326 West St	Stuart Park	2713	NSW	7/10/1970

**Database A – Blocking information**

RecID	Surname	Sndx-SN	Postcode	F3D-PC
a1	smith	s530	2602	260
a2	neighan	n250	2604	260
a3	meier	m600	2050	205
a4	smithers	s536	2012	201
a5	nguyen	n250	2022	202
a6	faulkner	f425	2037	203
a7	sandy	s530	2713	271

- (a1, b2)
- (a1, b3)
- (a1, b6)
- (a2, b2)
- (a2, b4)
- (a3, b1)
- (a3, b2)
- (a5, b4)
- (a5, b7)
- (a6, b5)
- (a7, b3)
- (a7, b6)

**Statt 7x7 nur  
12 Kandidaten**

**Database B**

RecID	Name	Address	BYear	BMonth	BDay
b1	Meier, Mary	14 (App 3) Hope Corner, Sydney 2000	1927	4	29
b2	Janice Meyer	Bryan St, O'Connor ACT 2604	1968	11	20
b3	Jonny Smith	47 Miller Street, 2619 Canberra ACT	1970	12	11
b4	Lyng Nguyen	1 Millie Road, 2002 North Sydney, NSW	1968	8	10
b5	Kristina Fawkner	13 St John Street, 2031 Glebe	1981	2	23
b6	Bob Santi	55 East St; Stuart's Point; NSW 2113	1970	12	11
b7	Lynette Cain	6 / 12 Hope Corner, 2020 Sydney N.S.W.	1970	7	13

**Database B – Blocking information**

RecID	Surname	Sndx-SN	Postcode	F3D-PC
b1	meier	m600	2000	200
b2	meier	m600	2604	260
b3	smith	s530	2619	261
b4	nguyen	n250	2002	200
b5	fawkner	f256	2037	203
b6	santi	s530	2113	211
b7	cain	c500	2020	202

# Beispiel: Record Pair Comparison

RecID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear	SimSum
a1	john	smith	42	miller	oconnor	12	11	1970	
b2	janice	meier		bryan	oconnor	20	11	1968	
	0.61	0.6	0.0	0.0	1.0	0.0	1.0	0.5	3.71
a1	john	smith	42	miller	oconnor	12	11	1970	
b3	john	smith	47	miller	canberra	11	12	1970	
	1.0	1.0	0.5	1.0	0.6	0.5	0.5	1.0	6.10
a1	john	smith	42	miller	oconnor	12	11	1970	
b6	robert	santi	55	east	stuarts point	11	12	1970	
	0.47	0.6	0.0	0.0	0.31	0.5	0.5	1.0	3.39
a2	joanne	neighan		brown	dickson	8	1	1968	
b2	janice	meier		bryan	oconnor	20	11	1968	
	0.78	0.56	0.0	0.73	0.51	0.0	0.5	1.0	4.08
a2	joanne	neighan		brown	dickson	8	1	1968	
b4	lyng	nguyen	1	millie	north sydney	10	8	1968	
	0.47	0.64	0.0	0.0	0.45	0.0	0.0	1.0	2.56

# Beispiel: Classification

Candidate pair	SimSum	Classification
(a1, b2)	3.71	Non-match
(a1, b3)	6.10	Match
(a1, b6)	3.39	Non-match
(a2, b2)	4.08	Potential match
(a2, b4)	2.56	Non-match
(a3, b1)	5.15	Potential match
(a3, b2)	2.91	Non-match
(a5, b4)	7.78	Match
(a5, b7)	3.07	Non-match
(a6, b5)	7.12	Match
(a7, b3)	2.98	Non-match
(a7, b6)	4.85	Potential match

# Aufgabe zu preprocessing

- Laden Sie die Datei `names_adress.csv` mit pandas
- Nutzen Sie das `recordlinkage` Paket um
  - den Namen vorzuverarbeiten
  - den Namen mit soundex phonetisch zu kodieren
  - die Telefonnummer vorzuverarbeiten

# Aufgabe zu Indizierung

- Erstellen Sie mit `recordlinkage` einen full Index auf den Frames `df_a` und `df_b`
- Erstellen Sie mit `recordlinkage` einen block Index auf den Frames `df_a` und `df_b` auf der Spalte `names`
- Erstellen Sie mit `recordlinkage` einen zufälligen Index auf den Frames `df_a` und `df_b`
- Erstellen Sie mit `recordlinkage` einen sorted neighbourhood Index auf den Frames `df_a` und `df_b` auf der Spalte `names` und der Fenstergröße 3

# String distances

- *Levenshtein* oder *edit* distance
- minimum number of insertions, deletions or replacements of characters needed to convert one string in the other
- Beispiel: kitten und sitting hat Abstand 3
- *Affine gap* distance
  - adds two operations to edit distance: opening gap and extending gap

Beispiel: Deu ähnlich zu Deutschland, but it's complicated...

-

# String distances

- Jaro distance

- $$\text{Jaro}(s_1, s_2) = \frac{1}{3} \left( \frac{c}{s_1} + \frac{c}{s_2} + \frac{c-t}{c} \right)$$

- Mit  $c$  die Anzahl gemeinsamer Zeichen, und  $t$  die Anzahl der Transpositionen.

- Zwei Zeichen sind gemeinsam wenn sie gleich sind und nicht mehr als  $\left\lfloor \frac{\max(s_1, s_2)}{2} \right\rfloor - 1$  Stellen entfernt

- Transpositionen werden berechnet als Anzahl gemeinsamer Zeichen, die nicht an der richtigen Stelle stehen, geteilt durch 2

- Beispiel: FAREMVEL und FARMVILLE hat Abstand  $\frac{1}{3} \left( \frac{8}{9} + \frac{8}{9} + \frac{8-1}{9} \right) = 0.88$

# String distances

- *Jaro-Winkler* distance
- $\text{Jaro-Winkler}(s_1, s_2) = \text{Jaro}(s_1, s_2) + lp(1 - \text{Jaro}(s_1, s_2))$
- mit  $l$  die Anzahl gemeinsamer Zeichen am Beginn des Strings, aber maximal 4, und  $p$  eine Gewichtung, normalerweise 0.1.
- Beispiel: FAREMVEL und FARMVILLE hat Abstand  
 $0.88 + 3 \cdot 0.1 \cdot (1 - 0.88) = 0.92$

# Numeric distances

- *Maximum absolute distance with linear interpolation*

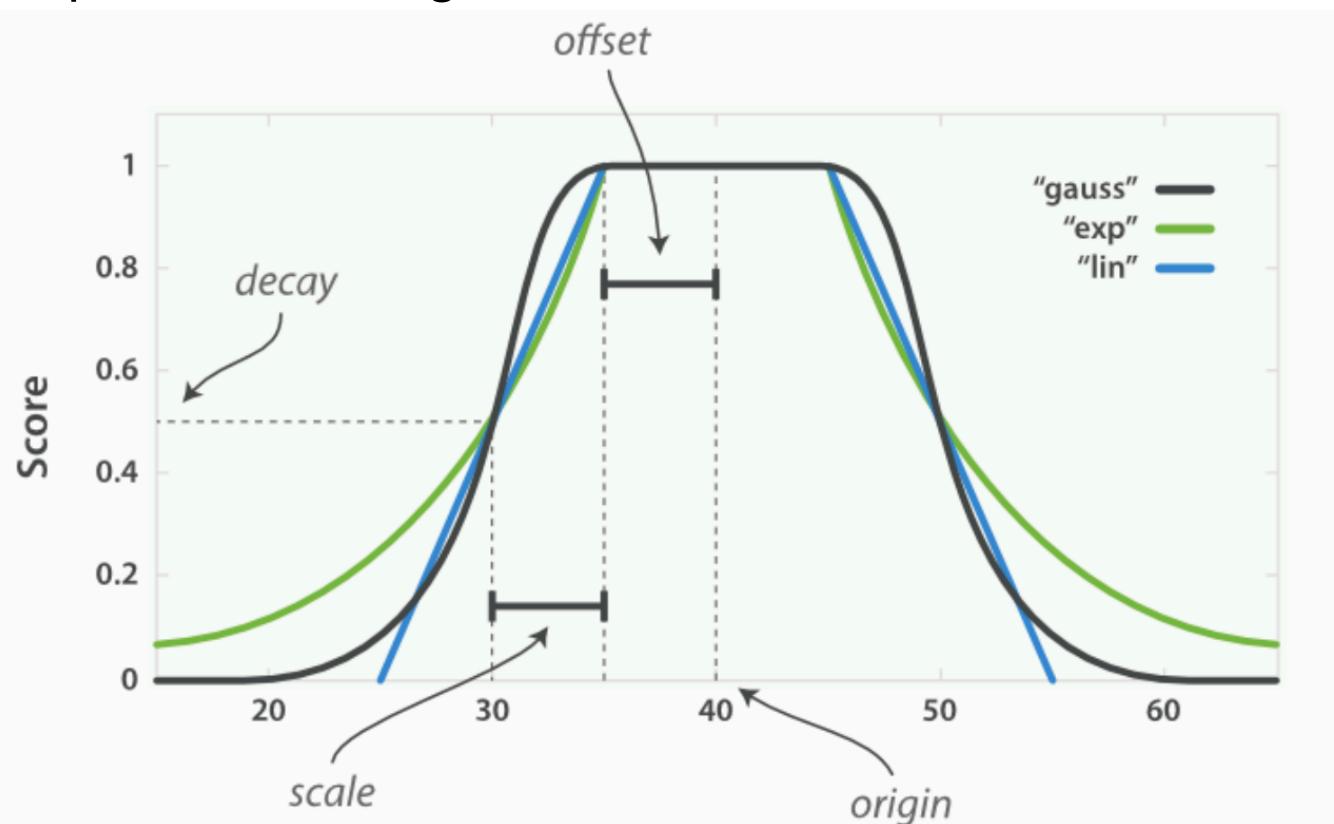
- $$\text{sim}(n_1, n_2) = \begin{cases} 1 - \frac{n_1 - n_2}{d_{\max}} & \text{if } |n_1 - n_2| < d_{\max} \\ 0 & \text{sonst} \end{cases}$$

- Beispiel:  $d_{\max} = 1000$ ,  $n_1 = 2000$ ,  $n_2 = 2500$ . Dann  $\text{sim}(n_1, n_2) = 1 - \frac{500}{1000} = \frac{1}{2}$

- Aber auch:  $d_{\max} = 1000$ ,  $n_1 = 400.000$ ,  $n_2 = 400.500$ . Dann  $\text{sim}(n_1, n_2) = \frac{1}{2}$ , obwohl der relative Fehler viel kleiner

# Numeric distances

- Andere Interpolationen möglich



# Aufgabe zu Record Pair Comparison

- Laden Sie aus recordlinkage.datasets das Datenset  
`load_febrl1()`
- Erstellen Sie einen index
- Experimentieren Sie mit recordlinkage.Compare und vergleichen Sie verschiedene Spalten

# Strategien zur Klassifizierung

- *Threshold based classification (unsupervised)*: Bilde Summe der Scores. Dann Histogramm, und entscheiden wo guter Punkt zum abschneiden
- *Threshold based classification (supervised)*: Bilde Summe der Scores. Dann Threshold so wählen, dass entweder Anzahl false matches oder Anzahl false non-matches minimiert wird
- *Threshold based classification (supervised)*: Bilde gewichtete Summe der Scores, wobei Gewichte durch eine logistische Regression gebildet werden

# Strategien zur Klassifizierung

- *Cost based classification:* Einem nicht interessierten Kunden einen Flyer schicken verursacht fast keine Kosten. Einem interessierten Kunden keinen Flyer schicken verursacht dagegen hohe Kosten

Cost	Classification	True match status
$c_{U,M}$	Non-Match	True match ( $M$ )
$c_{U,U}$	Non-Match	True non-match ( $U$ )
$c_{P,M}$	Potential Match	True match ( $M$ )
$c_{P,U}$	Potential Match	True non-match ( $U$ )
$c_{M,M}$	Match	True match ( $M$ )
$c_{M,U}$	Match	True non-match ( $U$ )

$$\begin{aligned} c = & c_{U,M} \cdot P(r \in \text{Non-Match}, r \in M) + c_{U,U} \cdot P(r \in \text{Non-Match}, r \in U) + \\ & c_{P,M} \cdot P(r \in \text{Potential Match}, r \in M) + c_{P,U} \cdot P(r \in \text{Potential Match}, r \in U) + \\ & c_{M,M} \cdot P(r \in \text{Match}, r \in M) + c_{M,U} \cdot P(r \in \text{Match}, r \in U), \end{aligned}$$

•

# Strategien zur Klassifizierung

- *Supervised Machine learning:* Entscheidungsbäume, support vector machines, ...
- *Unsupervised Machine learning:* Expectation Maximisation, k-means, ...

# Scaling and Normalizing Numerical Values

# Vorüberlegungen

Reicht es für mein Ziel, zu wissen ob die Zahl positiv oder negativ ist?

Reicht die grobe Größenordnung?

Spielt die Skalierung eine Rolle?

Wie ist die Verteilung meiner Daten?

# Kriterien für Skalierung

Modelle, die die euklidische Norm verwenden, reagieren empfindlich auf Skalierung: k-means, RBF kernels, ...

Modelle, die logische Operatoren verwenden, sind nicht empfindlich:  
Entscheidungsbäume, random forests, ...

Ausnahme: Datadrift im Zeitverlauf, z.Bsp. Besuchszahlen auf Webseiten

# Skalieren

Verschieben der Verteilung einer Spalte in einen gewünschten Bereich, ohne die Verteilung der Daten zu ändern

# Min Max Skalieren

Spalte A mit Minimum  $\min_A$  und Maximum  $\max_A$ . Gewünschter neuer Bereich ist  $[n, n + R]$ . Dann ist die Skalierung

$$x' = n + R \left( \frac{x - \min_A}{\max_A - \min_A} \right)$$

Normalerweise nimmt man als Zielbereich  $[-1,1]$  oder  $[0,1]$

Beispieldaten: housing.csv, Spalte total\_rooms

Python: `sklearn.preprocessing.MinMaxScaler` ↗

Erstellen Sie vorher und nachher ein Histogramm mit ausreichend bins, damit Sie die Verteilung der Daten sehen können

# z-score Skalierung

Spalte A mit bekanntem Mittelwert  $\mu_A$  und bekannter Standardabweichung  $\sigma_A$ . Dann ist die Skalierung

$$x' = \frac{x - \mu_A}{\sigma_A}$$

Dann hat die transformierte Spalte Mittelwert 0 und Standardabweichung 1.

Beispieldaten: housing.csv, Spalte `housing_median_age`

Python: `sklearn.preprocessing.StandardScaler` ↗

Erstellen Sie vorher und nachher ein Histogramm mit ausreichend bins, damit Sie die Verteilung der Daten sehen können. Überprüfen Sie Mittelwert und Standardabweichung nach der Transformation

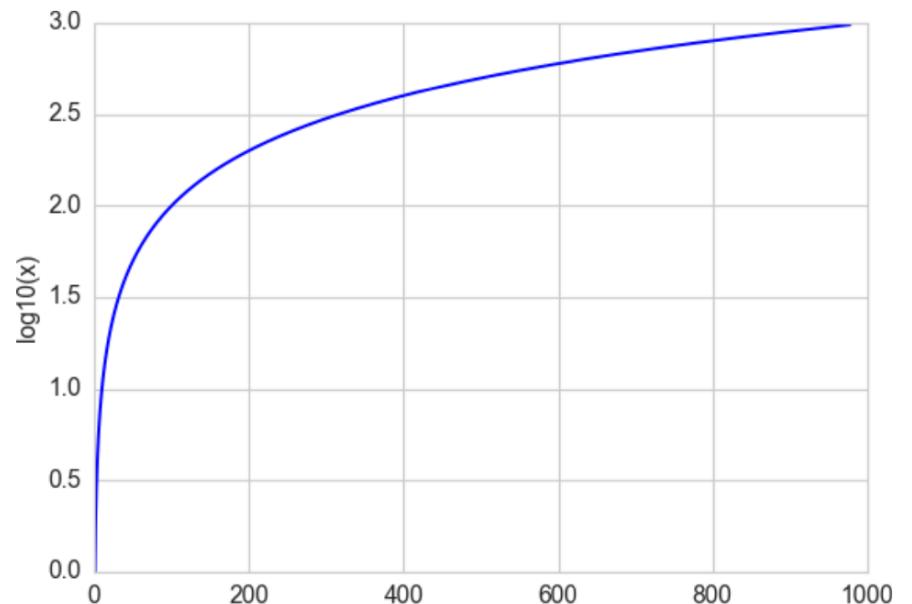
# Normalisieren

Anwendung einer Funktion  $f(x)$  so dass die Daten annähernd normal verteilt sind

# Log Transformation

Der 10er Logarithmus bildet

- $(0,1)$  auf das Intervall  $(-\infty,0)$  ab
- $(1,10)$  auf das Intervall  $(0,1)$
- $(10,100)$  auf das Intervall  $(1,2)$



# Log Transformation

Eignet sich besonders für heavy tailed distributions

Beispiele:

1. yelp Business Reviews, Datensatz

`yelp_academic_dataset_business.json`, Spalte  
`'review_count'`

2. Online News Popularity, Spalte '`n_tokens_content`'. Vorsicht, Falle.

# Log Transformation

Anwendung:

1. Lineare Regression im Yelp Datensatz : review\_count **auf** stars  
**versus** log\_review\_count **auf** stars
2. Lineare Regression im Online News Datensatz: n\_tokens\_content **auf** shares **versus** log\_n\_tokens\_count **auf** shares

Führen Sie zur Evaluation des Modells eine 10-fache Kreuzvalidierung mit dem scikit-learn Modul `from sklearn.model_selection import cross_val_score` durch. Interpretieren Sie die Ergebnisse mit Hilfe von Scatterplots.

# Box Cox Normalisierung

Was ist die optimale Normalisierung meiner Daten?

$$y = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

Mit  $\lambda$  zwischen -5 und 5, oder ähnliches Intervall

Python: `scipy.stats.boxcox`

# Box Cox Normalisierung

$$y = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

Beispielwerte

- $\lambda = 1$ : Keine Transformation
- $\lambda = 0$ : Log Transformation
- $\lambda = 0.5$ : Wurzel
- $\lambda = -1$ : Invertieren

# Box Cox Normalisierung

Anwendung:

1. Führen Sie beim housing Datensatz eine Box Cox Normalisierung der Spalte `total_rooms` durch
2. Erstellen Sie ein Histogramm der transformierten Spalte, um die Verteilung nach der Transformation zu visualisieren

# Transforming Numerical Values

# Transformieren

Eine Spalte mit Hilfe einer Funktion  $f(x)$  in eine andere Form bringen

# Logit Transformation

Eine Spalte die Werte zwischen 0 und 1 liegt nach  $[-\infty, +\infty]$  transformieren

$$\text{logit}(x) = \log \left( \frac{x}{1-x} \right)$$

Anwendung auf Proportionen oder Wahrscheinlichkeiten, da ein Modell auch Werte außerhalb von  $[0,1]$  annehmen kann

Frage: Relevant für Prädiktoren oder Zielvariable?

# Logit Transformation

Ihre Aufgabe:

- Laden Sie den Datensatz `MLBPlayerStats2022Batting.csv`
- Erstellen Sie ein Histogramm für die Spalte `BA`
- Führen Sie eine logit Transformation auf die Spalte `BA` durch
- Erstellen Sie ein Histogramm für die transformierte Spalte
- Lösen Sie das Problem der  $-\infty$  Werte

Python: `scipy.special.logit`

# Glätten: Moving average

Wird besonders bei Zeitreihen zum glätten verwendet

$$\bar{x}_t = \frac{x_t + \dots + x_{t-n+1}}{n}$$

Aufgabe: Plotten Sie die original Zeitreihe und den moving average für verschiedene n

Daten: <https://fred.stlouisfed.org/series/TXBPPRIV>

Python: `pandas.Series.rolling(n).mean`

# Glätten: Moving median

Auch hautsächlich für Zeitreihen

$$\bar{x}_t = \text{Median}(x_1, \dots, x_{t-n+1})$$

Robuster gegen Outlier als der moving mean

Python: `pandas.Series.rolling.median`

# Glätten: Wichtige Punkte

Zu viel glätten kann nichtlineare Trends verwischen

Glätten muss für Trainings und Testdaten separat durchgeführt werden, um Leakage zu vermeiden

Glätten kann helfen, noise aus den Daten zu eliminieren

# Potenzen von Features

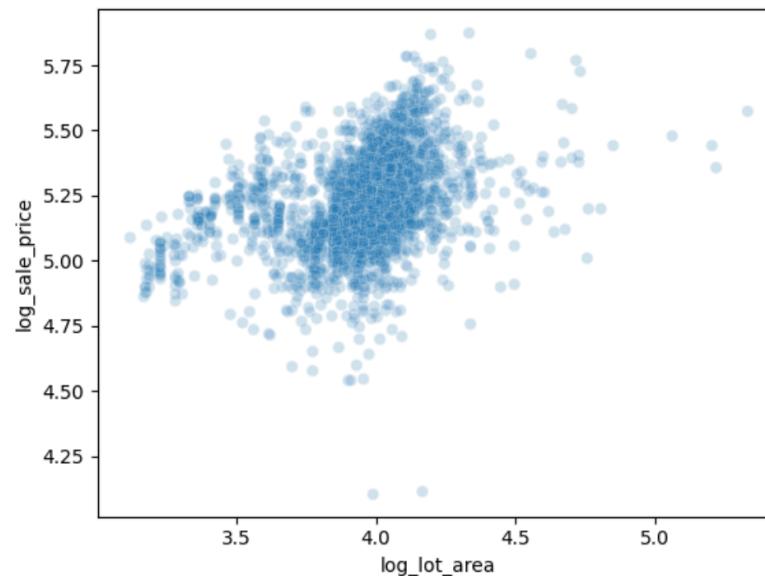
Sinnvoll bei nichtlinearen, sondern quadratischen oder ähnlichen Zusammenhängen

Vorgehen:

x	$x^2$	$x^3$
3	9	27
5	25	125

# Potenzen von Features

Beispiel: Zusammenhang zwischen Grundstücksgröße und Verkaufspreis bei Häusern in Iowa nach log Transformation



# Potenzen von Features

Ihre Aufgabe:

- Laden Sie die Ames housing daten
- Führen Sie eine log Transformation auf lot area und SalePrice durch
- Fügen Sie quadratische und kubische Terme zur log lot area hinzu
- Führen Sie eine lineare Regression von log lot area auf log SalePrice durch, mit und ohne die neuen Terme. Vergleichen Sie die Ergebnisse.

# Ein Feature in Stücke schneiden

Mit Hinge Functions lässt sich ein Feature in einzelne Regionen zerteilen

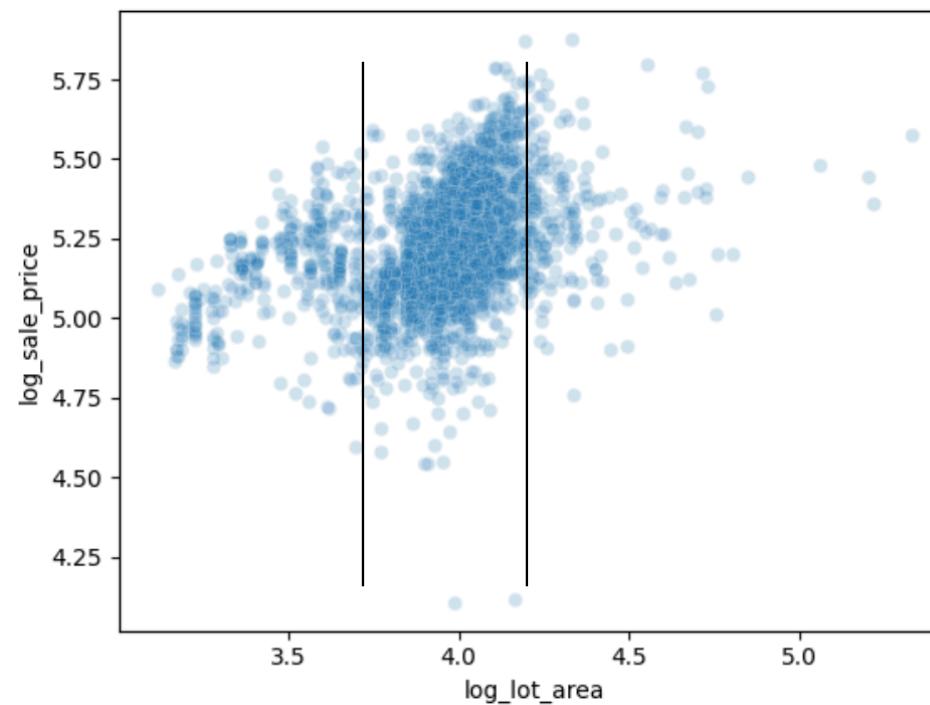
Approximation des Beispiels mit

$$5 \cdot x_1 + 15 - 3 \cdot x_2$$

x	y	$x_1=\max(0, x-2)$	$x_2=\max(2-x, 0)$
0	9	0	2
1	12	0	1
2	15	0	0
3	20	1	0
4	25	2	0
5	30	3	0

# Ein Feature in Stücke schneiden

Im Ames Housing Beispiel



# Aufgabe zu Hinge Functions

Ihre Aufgabe:

- Fügen Sie vier hinge Variablen zur log lot area hinzu, zwei zum Abschneidepunkt 3.75 und zwei zum Abschneidepunkt 4.25. Formel für erste Variable:  $\max(0, 3.75 - x)$
- Plotten Sie eine der hinge Variablen gegen log lot area
- Führen Sie erneut eine lineare Regression durch

# Interaction Features

Beispiel:

- Wasser hat Effekt auf Pflanzenwachstum
- Dünger hat Effekt auf Pflanzenwachstum
- Der gemeinsame Effekt von Wasser und Dünger ist größer als die Einzeleffekt

Wie modellieren wir das in den Daten?

# Interaction Features

Bei einem linearen Modell mit zwei Eingangsvariablen:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \text{error}$$

Wir fügen also das Produkt als zusätzliche Variable hinzu.

Der Koeffizient  $\beta_3$  repräsentiert die inkrementelle Änderung auf Grund des kombinierten Effekts von  $x_1$  und  $x_2$ , der über den Effekt von nur  $x_1$  und  $x_2$  hinausgeht.

# Interaction Features

Welche Features kombiniert werden sollen, ist vom Problem und gewählten Modell abhängig.

Bei wenig Features kann man alle Kombinationen ausprobieren:  $\frac{p(p - 1)}{2}$  Möglichkeiten

Tree models brauchen z.Bsp. keine interaction features.

# Binning

If all else fails, try binning.

Synonym: Diskretisierung

“Binning is the process of translating a quantitative variable into a set of two or more qualitative buckets (i.e., categories).”

Z.Bsp. werden oft Quartile genutzt. Daraus entstehen dann 4 bins.

# Binning

Warum?

- Kann Ergebnisse interpretierbarer machen (Z.Bsp. Bei über 40 jährigen wächst das Risiko für ... um 23%)
- Binning kann die Variation in den Daten drastisch reduzieren

Vorsicht!

- Trenderkennung wird schwieriger
- Cut-Offs sind beliebig
- Modell Ergebnisse können von Wahl der Cut-Offs abhängen
- nicht existente Trends werden erkannt
- “eyeballing” von Cut-Offs führt zu overfitten

# Binning

Ihre Aufgabe:

- Nutzen Sie den yelp business review Datensatz
- Diskretisieren Sie den review count in Quartil-bins
- Outcome soll eine neue Spalte sein, die nur die Werte 0-3 enthält

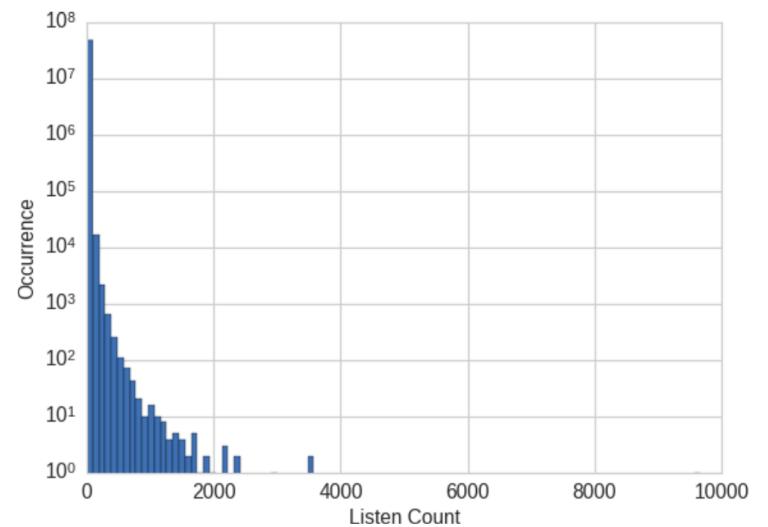
Python: `pandas.qcut`

# If even binning fails...

... try binarizing

Beispiel: Million Song Dataset

- 48 Mio Tripel aus User ID, Song ID, Listen Frequency
- Über 1 Mio Nutzer und über 380.000 Songs



# If even binning fails...

Ein User der 20 mal einen Song hört, mag ihn nicht doppelt so sehr wie jemand der einen Song 10 mal hört

Listen Count ist kein robuster Indikator für Geschmack

Robustes Maß: If count > 0 then 1 else 0

# Encoding categorical variables

# Wann ist eine Variable kategorisch?

Kategorische Variablen beschreiben qualitative Phänomene

Beispiele

- Quantitativ: Aktienkurs
- Qualitativ: Industriesektor der Aktie

# Geordnete und ungeordnete kategoriale Variablen

Kategoriale Variablen können entweder geordnet oder ungeordnet sein

Beispiele

- Geordnet: “Schlecht”, “Gut”, “Am Besten”
- Ungeordnet: “Französisch”, “Indisch”, “Amerikanisch”

# Unabhängig vom Datentyp

Auch numerische Werte können kategorische Variablen sein

Beispiele

- PLZ: “64295”, “55118”. Darmstadt ist nicht um 9177 besser als Mainz

# Kompatible Modelle zu kategorischen Daten

Ob kategorische Werte transformiert werden müssen, hängt vom gewählten Modell ab

## Beispiele

- Entscheidungsbäume können mit kategorischen Werten als cut-offs arbeiten
- Naive Bayes Modelle arbeiten mit contingency tables nach kategorischen Werten
- Aber: Die meisten Modelle erwarten numerische Daten

# One Hot Encoding

Jede Ausprägung der Variable erhält ein Bit



Person	Stadt	Person	DA	FFM	MZ
1	DA	1	1	0	0
2	FFM	2	0	1	0
3	MZ	3	0	0	1
4	DA	4	1	0	0

In Python: `sklearn.preprocessing.OneHotEncoder`

Aufgabe:

- OKCupid Data, one hot encoding der Spalte `body_type`
- Bonus: Wo sind die feature Namen?

# One Hot Encoding

Da jede Ausprägung eine Spalte bekommt, haben wir folgende Gleichung:

$$e_1 + \dots + e_k = 1,$$

Also eine lineare Abhängigkeit zwischen den Spalten

# Dummy Coding

Lösung für lineare Abhängigkeit: Eine Ausprägung wird nur implizit abgebildet, durch den Null-Vektor

Person	Stadt
1	DA
2	FFM
3	MZ
4	DA



Person	DA	FFM
1	1	0
2	0	1
3	0	0
4	1	0

Python: `pandas.get_dummies, OneHotEncoder ( drop='first' )`

Fragen:

- Warum nur noch 11 statt 13 Werte?
- Warum eignet sich `sklearn.preprocessing.OneHotEncoder` besser für maschinelles Lernen?

# Effect Coding

Implizite Ausprägung wird durch -1-Vektor abgebildet

Person	Stadt
1	DA
2	FFM
3	MZ
4	DA



Person	DA	FFM
1	1	0
2	0	1
3	-1	-1
4	1	0

Vorteil: Bei linearer Regression ist der y-Achsenabschnitt der Mittelwert der Zielvariable

Python: `pandas.get_dummies`, dann von Hand werte anpassen

# Pros and Cons

Method	Pro	Con
One Hot	Missing data bekommt 0 Vektor	Redundant
Dummy	Nicht redundant	Was tun mit missing data?
Effect	Nicht redundant, Ergebnisse leichter interpretierbar	-1-Vektor ist dicht Teuer für Berechnungen

# Probleme bei zu vielen Ausprägungen

Seltene Ausprägungen schaffen es nicht ins Test Data set

In einem Sample kann ein Prädiktor konstant oder fast konstant sein, z.Bsp.  
999 mal 0 und nur ein mal 1

Beide Möglichkeiten können zu Problemen beim Training führen

# Aufgabe zu vielen Ausprägungen

Führen Sie ein one-hot-encoding auf die Spalte `location` durch

Erstellen Sie ein train test split

Identifizieren Sie die Variablen, die es nicht ins training set geschafft haben

# Lösung mit other Kategorie

Wir berechnen pro Spalte die Frequenz, wie häufig ein Wert vorhanden ist

Alle Variablen, deren Frequenz unter einem cutoff liegen, kommen in eine other Kategorie

# Feature Hashing

Falls zu viele Ausprägungen vorhanden sind: Hash function anwenden

Hash function bildet  $\mathbb{N} \rightarrow [1,m]$  ab

Wichtige Begriffe zu Hash Functions:

Collision: Zwei Ausprägungen werden auf die gleiche Zahl abgebildet

Uniform: Ungefähr die gleiche Anzahl an Zahlen pro bin

# Feature Hashing

Collisions können wir nicht vermeiden, da wir die Zahl der Features reduzieren möchten

Uniform ist nur in Kryptographie ein Ziel, für uns irrelevant

# Feature Hashing

Vorgehen bei n gewünschten Features:

String → Hash-Wert → (Hash-Wert mod n)

San Francisco ↠ 823883475 ↠ 15

Also bekommt San Francisco eine 1 in der 15. Spalte, sonst 0

# Feature Hashing

Mit einer (-1) als zusätzlichen Wert erhalten wir sogar  $2n$  Features mit nur  $n$  Spalten

Vorgehen bei  $2n$  gewünschten Features in  $n$  Spalten:

String → Hash-Wert → (Hash-Wert mod  $2n$ ) - n

Palo Alto ↠ 834834297 ↠ -14

Also bekommt San Francisco eine -1 in der 14. Spalte, sonst 0

# Feature Hashing

Ihre Aufgabe zu Feature Hashing:

Datensatz: OK Cupid Data, Spalte location

Python: `sklearn.feature_extraction.FeatureHasher`

Tips:

- Nutzen Sie `.unique` um die einzelnen Werte von location zu ermitteln
- Bringen Sie die Daten in das Format aus dem minimal working example
- Nutzen Sie dann erst den FeatureHasher

# Bin Counting

Statt Ausprägung des Features betrachten wir die *bedingte Wahrscheinlichkeit* des Targets unter der Ausprägung

Bin Counting konvertiert damit eine kategorische Variable in Statistiken über den Wert der Variable.

Statt einer großen, fast leeren Darstellung der Variable wie bei one-hot-encoding erhalten wir eine kleine dichte numerische Darstellung.

# Bin Counting

Starten mit einfachen counts zu Feature User

User	Query	Ad Domain	Ad Click
Alice	Food	<u>foo.com</u>	1
Alice	Far	<u>foo.com</u>	0
Alice	Bear	<u>bar.org</u>	0
Bob	Food	<u>foo.com</u>	0
Bob	Bar	<u>bar.org</u>	1
...			

User	Number of clicks	Number of non-clicks
Alice	5	120
Bob	20	230
...		
...		

# Bin Counting

Aus den Counts können wir 2-way contingency tables errechnen...

	Click	Non-click	Total
Alice	5	120	125
Not Alice	995	18880	19875
Total	1000	19000	20000

... und damit die odds ratio:

$$\text{Odds ratio} = \frac{P(Y = 1 \mid X = 1)/P(Y = 0 \mid X = 1)}{P(Y = 1 \mid X = 0)/P(Y = 0 \mid X = 0)} = \frac{(5/125)/(120/125)}{(995/19875)/(18880/19875)}$$

Das Verhältnis zwischen “wie viel eher wahrscheinlich ist es dass Alice klickt statt nicht klickt” and “wie viel eher wahrscheinlich ist es dass andere Leute klicken statt nicht klicken.”

# Bin Counting

Einfacher zu implementieren: Nur Zähler des odds ration

$$\text{Numerator odds ratio(Alice)} = \frac{P(Y = 1 \mid X = 1)}{P(Y = 0 \mid X = 1)} = \frac{(5/125)}{(120/125)}$$

	Click	Non-click	Total
Alice	5	120	125
Not Alice	995	18880	19875
Total	1000	19000	20000

# Bin Counting

Klappt auch für die Kombination von Features:

User	Query	Ad Domain	Ad Click
Alice	Food	<u>foo.com</u>	1
Alice	Far	<u>foo.com</u>	0
Alice	Bear	<u>bar.org</u>	0
Bob	Food	<u>foo.com</u>	0
Bob	Bar	<u>bar.org</u>	1
...			

Query-Add Domain	Number of clicks	Number of non-clicks
Food, <u>foo.com</u>	5000	30000
Far, <u>foo.com</u>	100	900
Bear, <u>bar.org</u>	6	18
Bar, <u>bar.org</u>	347	389472

# Aufgabe zu Bin Counting

Berechnen Sie im avazu click through rate Datensatz den Zähler des odds ratio pro device id. Tipps: groupby und agg

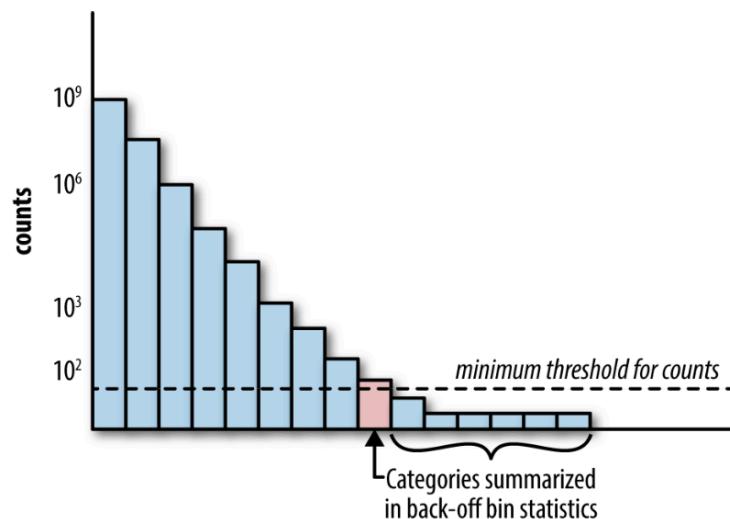
Erinnerung: Numerator odds ratio = 
$$\frac{P(Y = 1 \mid X = 1)}{P(Y = 0 \mid X = 1)}$$

	Click	Non-click	Total	Numerator
Alice	5	120	125	$(5/125) / (120/125)$

# Bin Counting für seltene Kategorien

Problem: Der User, der sich nur ein mal im Jahr einloggt

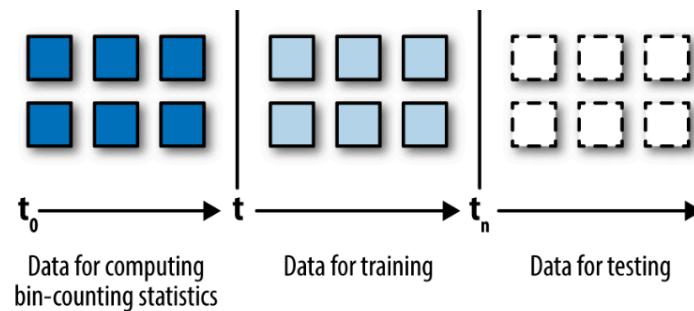
Lösung: Wieder eine `other` Kategorie. Falls sich die Anzahl über den Threshold entwickelt, dann eigene Kategorie



# Bin Counting und Leakage

Nachteil bei Bin Counting: *Leakage*. Man verwendet direkt Wissen über die Zielvariable, um diese vorherzusagen.

Lösung: Drei Phasen beim Daten sammeln



# Missing data



# Warum sind missing values ein Problem?

- Bei Predictors
  - Viele Modelle können mit missing values nicht umgehen
  - Viele Feature Engineering Techniken können nicht mit missing values umgehen
  - Ergebnisse eines Modells können schlechter werden
- Bei Zielvariable
  - Fast alle Modelle können mit missing values nicht umgehen

# Wie entstehen missing values?

Drei Hauptursachen:

- Zwei Datasets werden gemerged
- Zufällige Ereignisse
  - Batterie im Messgerät ausgefallen
  - Netzwerkverbindung ausgefallen
- Messfehler
  - Kamera unscharf
  - Patient nicht erschienen

# Typen von missing values

- Strukturelle Defizite
- Zufällig fehlende Werte
- Spezifische Gründe

# Strukturelle Defizite

Beispiel: Ames Housing Data, Spalte Alley.

Warum fehlt die Spalte? Womit kann man sie ersetzen?

# Zufällig fehlende Werte

Unterscheiden zwei Typen von zufällig fehlenden Werten:

- MCAR: Missing completely at random
- MAR: Missing at random

# Beispiele zu zufällig fehlenden Werten

- MCAR: Die Batterie einer Waage ist leer geworden, und Ladestand der Batterie ist nicht im Datensatz
- MAR:
  - Einer der Prädiktoren ist “Surface Type”, und auf weichen Oberflächen produziert die Waage mehr Messfehler
  - Personen mit sehr niedrigem oder sehr hohem Gehalt beantworten eine Frage häufig nicht, und sie haben das Gehalt der Personen zur Verfügung

# Zufällig fehlende Werte, in Formeln ausgedrückt

Maske  $R$  für fehlende Werte:

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Teile Beobachtungen  $X$  auf in  $(X_{\text{obs}}, X_{\text{mis}})$

Setzen  $\phi$  für die Parameter des wahren Modells

Dann:

- Missing completely at random:  $P(R = 0 | X_{\text{obs}}, X_{\text{mis}}, \phi) = P(R = 0 | \phi)$
- Missing at random:  $P(R = 0 | X_{\text{obs}}, X_{\text{mis}}, \phi) = P(R = 0 | X_{\text{obs}}, \phi)$

# Spezifische Gründe

- NMAR: Not missing at random
- Keine Vereinfachung der Formel  $P(R = 0 | X_{\text{obs}}, X_{\text{mis}}, \phi)$
- Beispiele
  1. Waage produziert im Zeitverlauf immer mehr Messfehler, und schwere Gegenstände werden später gemessen
  2. Ein Patient scheidet aus einer Studie aus, da er schwere Nebenwirkungen bekommt

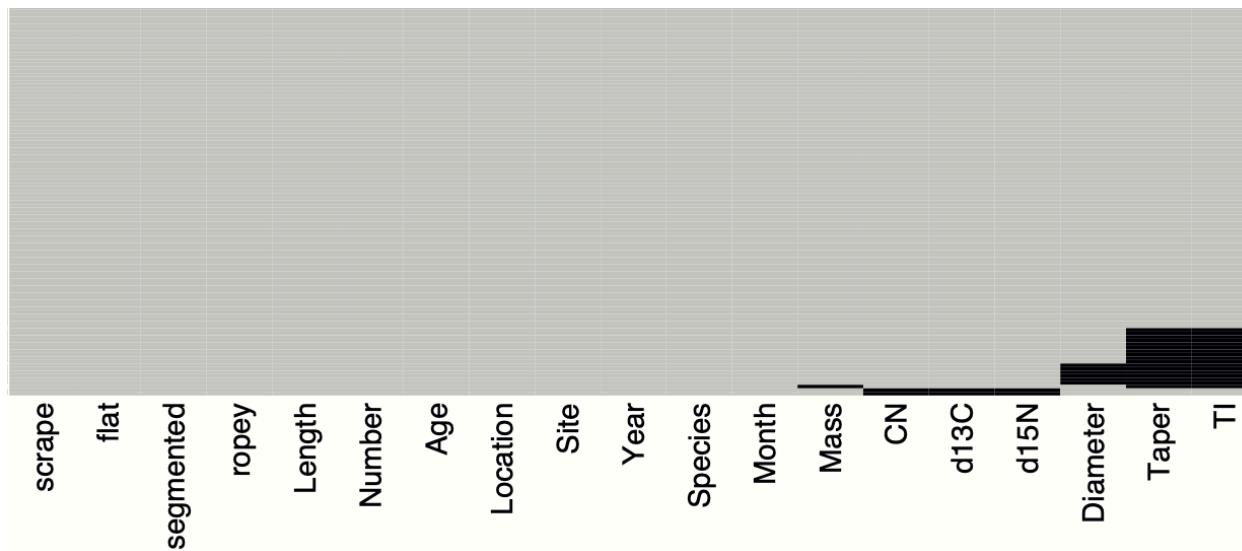
# NMAR in der Praxis

In der Praxis führt NMAR zu erheblichen Problemen.

Gängige Methoden zum Ersetzen fehlender Werte führen zu falschen oder verzerrten Ergebnissen

# Fehlende Werte visualisieren

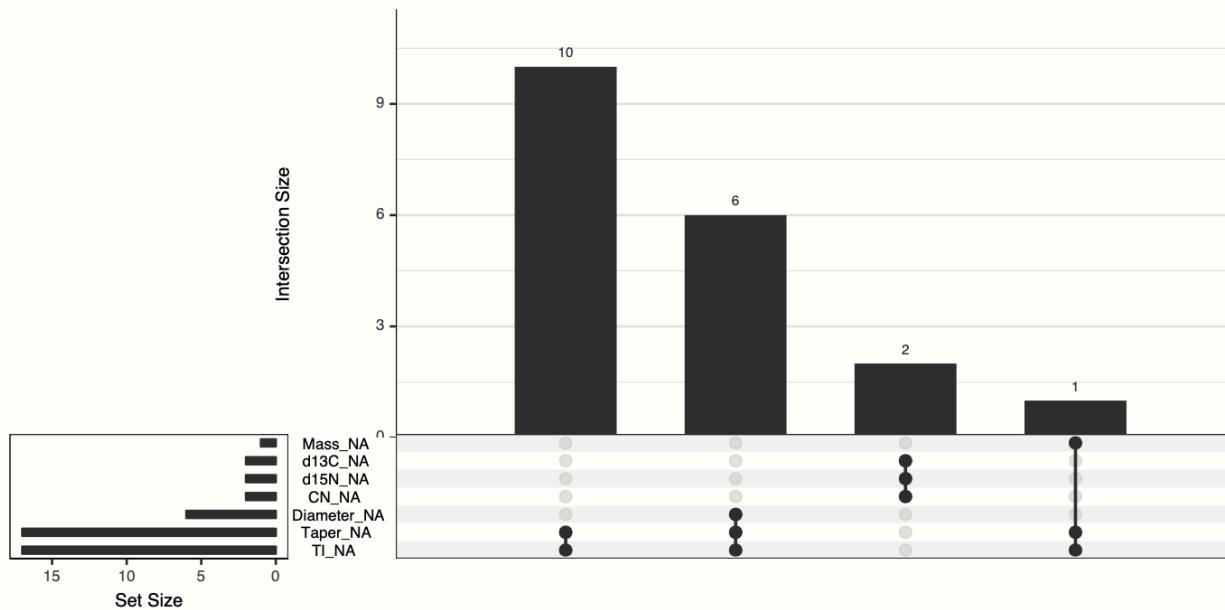
Heatmap



Aufgabe: Bauen Sie die Visualisierung oben für den scat dataset nach

# Fehlende Werte visualisieren

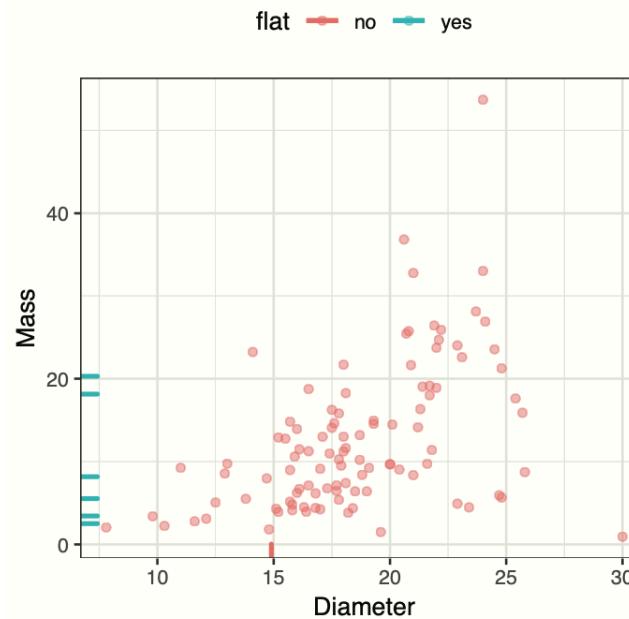
Co-occurrence plot



Aufgabe: Bauen Sie die Visualisierung oben für den scat dataset nach

# Fehlende Werte visualisieren

Scatterplot



Aufgabe: Bauen Sie die Visualisierung oben für den scat dataset nach

# Für große Datensets

Hauptkomponentenanalyse (PCA) liefert einen ersten Eindruck von Zeilen mit vielen fehlenden Werten.

Dazu wird eine Maske für fehlende Werte erzeugt, mit 1 für fehlende Werte:

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Anschließend werden die ersten beiden Hauptkomponenten visualisiert.

# Für große Datensets

Für Spalten mit vielen fehlenden Werten: Einfach die Maske transponieren.

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^T$$

Anschließend werden wieder die ersten beiden Hauptkomponenten visualisiert.

Aufgabe: Probieren Sie das Vorgehen mit den Chicago Train Daten aus

# Für große Datensets

Numerische Zusammenfassung der fehlenden Werte

- Anteil fehlende Werte pro Predictor
- Anteil fehlende Werte pro Sample

Aufgabe: Erstellen Sie diese Tabellen für den Scat Datensatz

# Strategien: Do not Impute

Do Not Impute: Man tut einfach gar nichts

- Viele Modelle haben eine default Strategie für fehlende Werte
- Manche Modelle können mit fehlenden Werten umgehen
- Ziel: Baseline für andere Strategien

# Strategien: Delete Missing

Zwei Varianten:

- Samples (entspricht Zeilen) mit fehlenden Werten werden gelöscht
- Predictors (entspricht Spalten) mit fehlenden Werten werden gelöscht

Faustregeln:

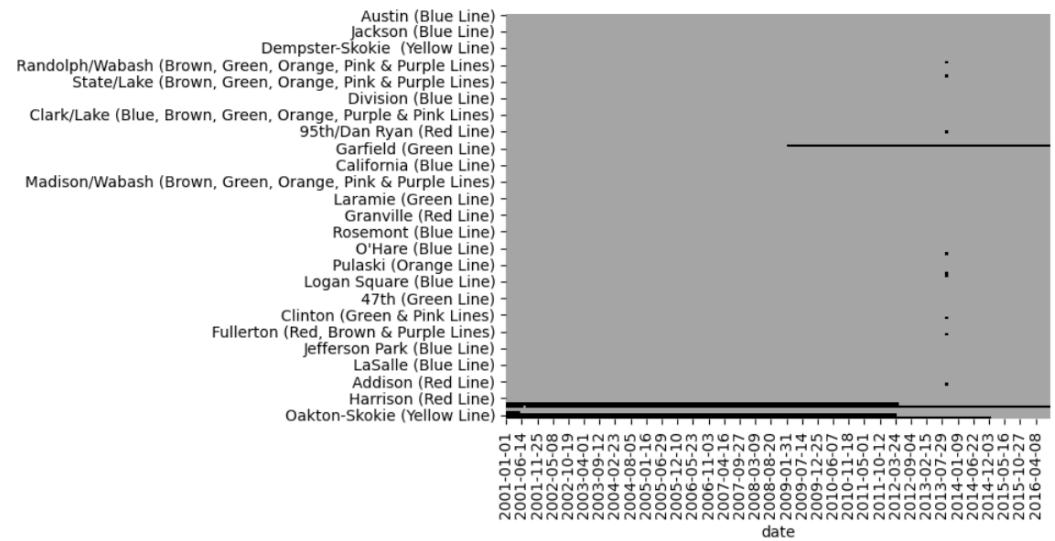
- Samples sind meist wichtiger als Predictors
- Falls missing completely at random, ist das trainierte Modell unbiased
- Falls nicht missing completely at random, ist das trainierte Modell biased
  - Bsp: Leute mit geringeren Einkommen beantworten seltener Frage nach Einkommen

# Delete Missing bei Chicago Train Ridership Data

Jedes sample (Daten für ein Tag) hat mindestens einen fehlenden Wert

Die allermeisten fehlenden Werten konzentrieren sich auf einzelne Predictors (Stationen)

Wie sollten wir also vorgehen?



# Strategien: Impute Most common

Most common: Fehlende Werte durch häufigsten Wert ersetzen

- Für kategoriale Variablen: Durch den am häufigsten vorkommenden Wert ersetzen
  - Für numerische Werte: Durch den Durchschnitt ersetzen
- 
- Aufgabe: Ersetzen Sie nan Werte im Chicago Train Datensatz mit dem globalen Durchschnitt. Visualisieren Sie ihr Ergebnis mit einer Heatmap.

# Strategien: Impute Concept most common

Concept Most common: Fehlende Werte durch häufigsten Wert in der Gruppe der samples mit gleichen sonstigen Prädiktoren ersetzen, oder Teilmenge gleicher Prädiktoren

- Aufgabe: Ersetzen Sie nan Werte im Chicago Train Datensatz mit dem Durchschnitt des Predictors. Visualisieren Sie ihr Ergebnis mit einer Heatmap für die Station '87th (Red Line)'

# Strategien: Impute Using k Nearest Neighbours

Angenommen im  $i$ -ten Sample  $y_i$  fehlt der  $h$ -te Wert, also  $y_{ih}$ .

Finde dann die  $k$  volle Samples  $\{x_1, \dots, x_k\}$ , die am nächsten bei  $y_i$  liegen.

Der ersetzte Wert  $\hat{y}_{ih}$  ist dann der Durchschnitt  $\sum_i^k \frac{x_{ih}}{k}$ .

- Aufgabe: Ersetzen Sie nan Werte im Chicago Train Datensatz mit k Nearest Neighbours mit k=2. Visualisieren Sie ihr Ergebnis mit einer Heatmap für die Station '87th (Red Line)'

# Strategien: Iterative Impute Using Estimator

Eine multivariate Methode. Zum Schätzen der fehlenden Werte werden alle vorhanden Predictors genutzt.

Dieses Verfahren wird iterativ reihum für alle Predictors genutzt, die fehlende Werte enthalten.

Das Verfahren wird so lange fortgesetzt, bis es konvergiert (oder eine Warnung generiert wird).

- Aufgabe: Ersetzen Sie nan Werte im Chicago Train Datensatz mit einem iterative imputer und max\_iter=2. Visualisieren Sie ihr Ergebnis mit einer Heatmap für die Station '87th (Red Line) '

# Beispiel Iterative Impute Using Estimator

Drei Merkmale: A, B und C. Fehlende Werte in allen drei Merkmalen

Erste Iteration:

Schätze A mit B und C.

Schätze B mit A (im vorherigen Schritt geschätzt) und C.

Schätze C mit den Merkmalen A und B (beide in vorherigen Schritten geschätzt).

Zweite Iteration:

Schätze A erneut mit den neu geschätzten B und C aus der ersten Iteration.

Schätze B erneut mit den neu geschätzten A und C.

Schätze Merkmal C erneut mit den neu geschätzten A und B.

# Imputation: Expectation Maximization

Annahme: Parameter  $\theta \in \mathbb{R}^d$ , der die probability density function  $f_\theta$  der Daten  $(X_{\text{obs}}, X_{\text{Miss}})$  charakterisiert

Die  $X_{\text{miss}}$  betrachten wir als latente Variablen, d.h. theoretisch messbar, aber praktisch nicht messbar

Suchen eine Maximum Likelihood Lösung für  $f_\theta$

Chicken-and-Egg Problem: Wir brauchen  $X_{\text{Miss}}$  um  $f_\theta$  zu berechnen und umgekehrt

# Strategien: Expectation Maximization

Chicken-and-Egg Problem: Wir brauchen  $X_{\text{Miss}}$  um  $f_\theta$  zu berechnen und umgekehrt

Vorgehen:

1.  $f_\theta$  auf Basis von  $X_{\text{obs}}$  schätzen
2. Damit  $X_{\text{miss}}$  schätzen
3. Weiter bei 1.

# Imputation: Expectation Maximization

Chicken-and-Egg Problem: Wir brauchen  $X_{\text{Miss}}$  um  $f_{\theta}$  zu berechnen und umgekehrt

Beispiel:

ID	depression	age	height	wage		ID	depression	age	height	wage
1	5	32		32,010		1	5	32	<b>181.43</b>	32,010
2		17	173	31,600	Depression = -15.3 + .01 x age + .004 x height + .0005 x wage	2	<b>1.362</b>	17	173	31,600
3	7		169	48,020	Age = 7.3 + .34 x depression + .002 x height + .0003 x wage	3	7	<b>19.53</b>	169	48,020
4	5	24	186	17,400	Height = 19.2 + .53 x depression + .021 x age + .0004 x wage	4	5	24	186	17,400
.	.	.	.		Wage = 7.3 + .44 x depression + .031 x age + .0021 x height	.	.	.	.	.
.	.	.	.			.	.	.	.	.
100	4	45	201	7,800		100	4	45	201	7,800

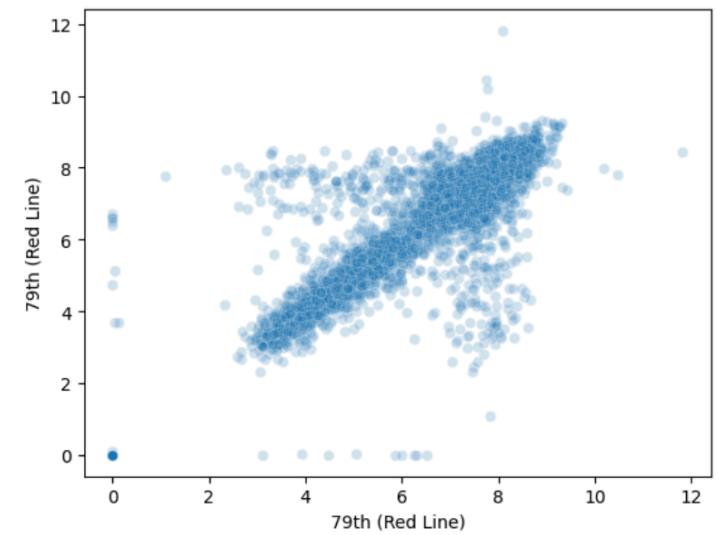
Depression = -14.9 + .03 x age + .005 x height + .0010 x wage  
Age = 7.1 + .24 x depression + .001 x height + .0002 x wage  
Height = 20.2 + .61 x depression + .022 x age + .0003 x wage  
Wage = 8.3 + .49 x depression + .041 x age + .0022 x height

...

# Strategien: Lineares Modell

Falls ein lineare Zusammenhang zwischen einem vorhandenen Predictor und einem Predictor mit fehlenden Werten besteht, kann ein lineares Modell genutzt werden.

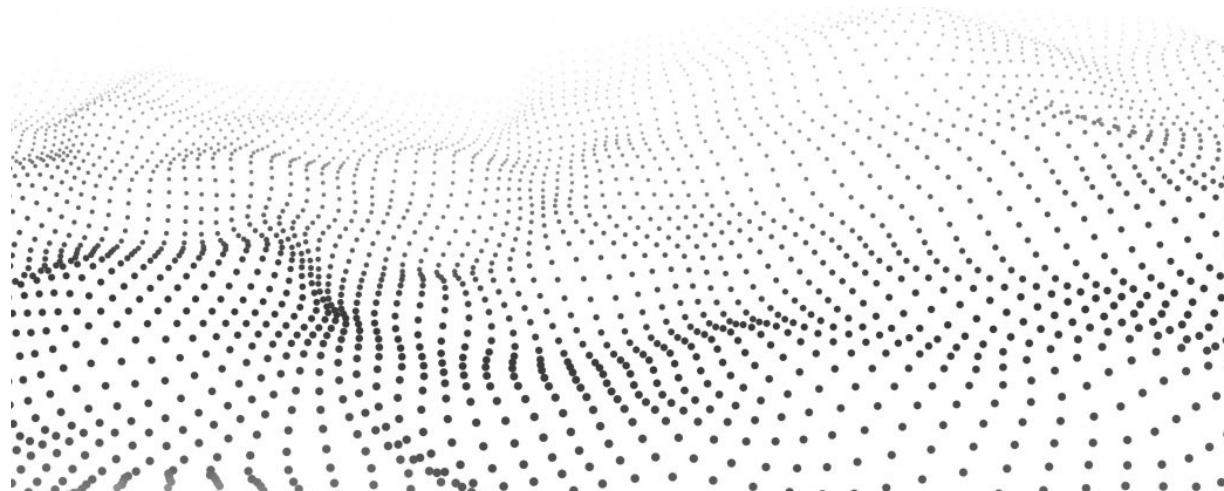
Beispiel: Der 14 Tage Lag in Chicago Train Data



# **Dimensionsreduktion**

# Daten als Punktewolke

Ein Datensatz mit n Samples und m Features wird als n Punkte im m-dimensionalen Raum interpretiert



# Ziel der Dimensionsreduktion

Anzahl der Features reduzieren, ohne zu viel Information zu verlieren

Zu viele Features

- machen Training langsamer
- erschweren das Finden einer guten Lösung

# Fluch der hohen Dimension

Hochdimensionale Geometrie ist für uns unintuitiv

In 2D ist die Wahrscheinlichkeit dass ein zufällig gewählter Punkt in einem 1x1 Quadrat weniger als 0.001 vom Rand entfernt liegt bei 0.4%

In 10000D ist die gleiche Wahrscheinlichkeit > 99.999999%

Der mittlere Abstand zwischen zwei zufällig gewählten Punkten in einem 3D Würfel liegt bei 0.66

In 1000000D ist der mittlere Abstand ungefähr 408.25

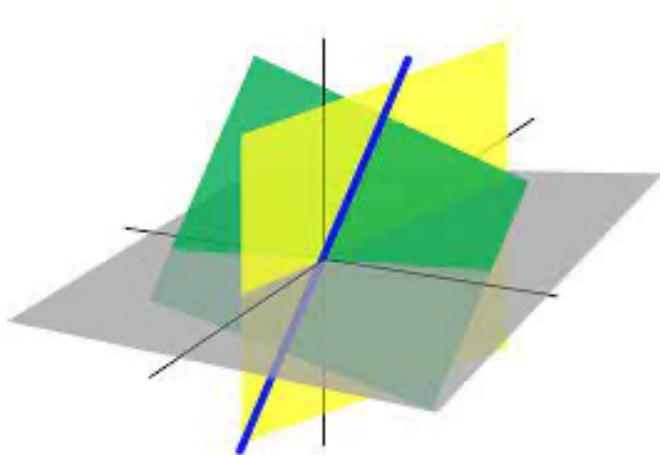
# Fluch der hohen Dimension

Faustregel: Bei nichtparametrischen Modellen wächst die Anzahl benötigter Samples exponentiell mit der Dimension

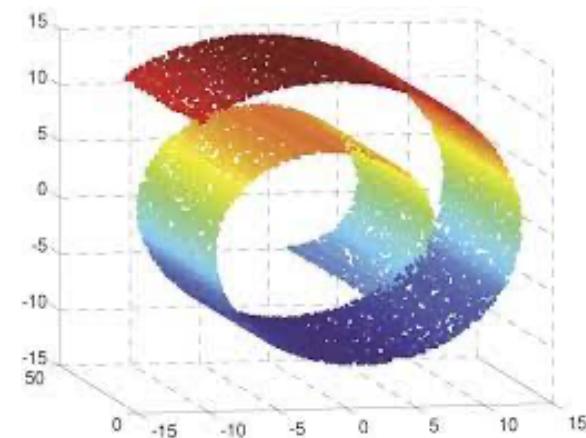
Beispiele für nichtparametrische Modelle: kNN, Support Vector Machines

# Liegen die Daten in einem Unterraum?

Linear

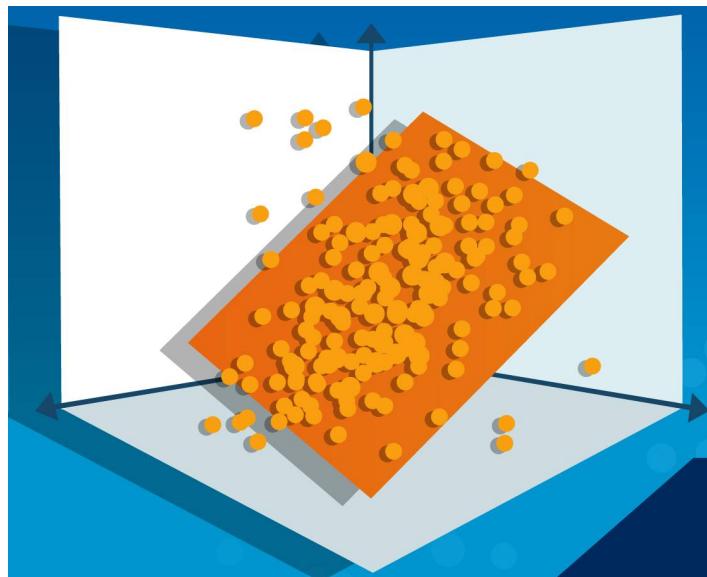


Gekrümmmt

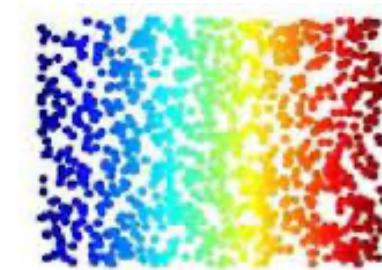


# Ansätze zur Dimensionsreduktion

- Linear: Projektion auf einen Unterraum



- Gekrümmmt: Manifold Learning



THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG  
PILE OF LINEAR ALGEBRA, THEN COLLECT  
THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL  
THEY START LOOKING RIGHT.



# Projektion auf Unterraum mit PCA

Um einen guten Unterraum zu finden, kann man PCA = *Principal Component Analysis* nutzen

Wichtigste Zutaten sind

- Covarianz Matrix
- Basiswechsel
- Singulärwertzerlegung

# Die Covarianz Matrix

Ein Feature entspricht einer Spalte  $x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$  im Datensatz

Angenommen  $E[x] = 0$ , ist die Varianz von  $x$  gleich  $\sum_0^n x_i^2$

In Vektorschreibweise:  $\text{Var}(x) = x^T x$

# Die Covarianz Matrix

Zwei Features entspricht zwei Spalten  $x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$ ,  $y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}$  im Datensatz

Angenommen  $E[x] = E[y] = 0$ , ist die Covarianz von  $x$  und  $y$  gleich

$$\sum_1^n x_i \cdot y_i$$

In Vektorschreibweise:  $\text{CoVar}(x) = x^T y$

# Die Covarianz Matrix

Für alle Features gleichzeitig können wir Varianzen und Kovarianzen in einer Matrix zusammen fassen (wieder angenommen Matrix ist bei 0 zentriert):

$$\text{CoVar}(X) = X^T X$$

Der Eintrag  $\text{CoVar}(X)_{i,j}$ ,  $i \neq j$  ist die Covarianz zwischen i-tem und j-tem Feature

Der Eintrag  $\text{CoVar}(X)_{i,i}$  ist die Varianz des i-ten Feature

Die Matrix  $\text{CoVar}(X)$  ist eine symmetrische  $m \times m$  Matrix

# Aufgabe zu Covarianz Matrix

- Laden Sie den 8x8 MNIST Datensatz (ist schon im Starter Notebook)
- Berechnen Sie die Covarianz Matrix

# Basiswechsel

Neben der Standardbasis  $\begin{pmatrix} 1 \\ 0 \\ \dots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \end{pmatrix}$  gibt es noch jede Menge andere Basen

Falls Sie das nicht mehr parat haben: <https://www.youtube.com/watch?v=P2LTAUO1TdA>

# Ziel von PCA

Finde einen orthonormalen Basiswechsel  $P$  so, dass die Transformierten Daten

$$Y = PX$$

eine diagonale Covarianzmatrix  $Y^T Y$  haben.

Diese Basis maximiert die Varianz (Diagonal-Terme) und minimiert die Covarianz (Off-Diagonal-Terme).

Die Off-Diagonal-Terme entsprechen Redundanz in den Daten.

# Singulärwertzerlegung

Da  $X^T X$  symmetrisch und quadratisch ist, hat diese Matrix  $r$  orthonormale Eigenvektoren und Eigenwerte, wobei  $r$  der Rang der Matrix ist.

Also:  $X^T X v_i = \lambda_i v_i$ , und als Matrix  $X^T X = V D V^T$ , wobei  $D = \text{diag}(\lambda_1, \dots, \lambda_n)$ .

Setze  $u_i = \frac{1}{\sqrt{\lambda_i}} X v_i$ . Dann gilt:

$$X v_i = \sqrt{\lambda_i} u_i$$

Sieht aus wie Eigenwerte, aber nicht ganz. Heißen *Singulärwerte*.

# Singulärwertzerlegung

Packen die  $v_i$  in eine Matrix  $V = [v_1, \dots, v_n]$  und die  $u_i$  in  $U = [u_1, \dots, u_n]$ .

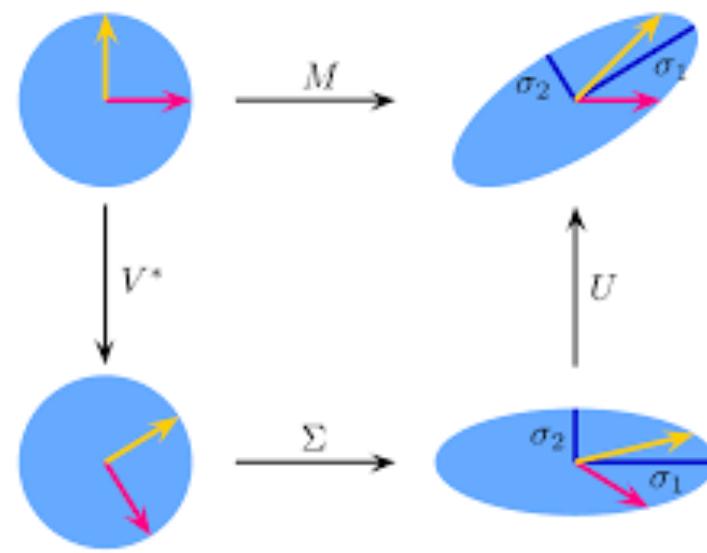
Außerdem die  $\sqrt{\lambda_i}$  in eine Diagonalmatrix  $\Sigma$ , mit Nullen aufgefüllt.

Dann:

$$X = U\Sigma V^T$$

# Singulärwertzerlegung

Geometrisch:



$$M = U \cdot \Sigma \cdot V^*$$

# PCA mit Singulärwertzerlegung

Gesuchte Basiswechselmatrix ist  $V^T$ . Denn:

$$V^T X (V^T X)^T = V^T X X^T V = V^T V D V^T V = D$$

# Aufgabe zu Singulärwertzerlegung

Nutzen Sie `np.linalg.svd` um die Basiswechselmatrix für den MNIST Datensatz zu finden

Vergessen Sie nicht, die Matrix vorher bei Null zu zentrieren

Bonus: Versuchen Sie die Eingangsmatrix aus den Rückgabewerten von `np.linalg.svd` wieder zu rekonstruieren. Den Vergleich können Sie mit `np.allclose` durchführen.

# Dimensionsreduktion mit PCA

Statt des kompletten Basiswechsels  $V^T$  nutzen wir einfach nur die ersten  $k$  Spalten des Basiswechsels.

Wir projizieren also auf den Unterraum, der von den ersten  $k$  neuen Basisvektoren aufgespannt wird.

Erinnerung: Projektion eines Vektors  $x$  auf den von  $u$  und  $v$  aufgespannten Unterraum ist durch  $(x^T \cdot u) \cdot u + (x^T \cdot v) \cdot v$  gegeben.

Die Koordinaten in der neuen Basis sind also  $\begin{pmatrix} x^T u \\ x^T v \end{pmatrix}$ .

# Aufgabe zu Projektion

Projizieren Sie den MNIST Datensatz auf die ersten beiden Hauptkomponenten

Beachten Sie dabei, dass unsere Daten bereits transponiert sind

Wir können also in Matrixschreibweise die Daten durch  $X \cdot \begin{pmatrix} v_{11} & v_{21} \\ \dots & \\ v_{1m} & v_{2m} \end{pmatrix}$  projizieren.

## **Vergleich mit sklearn.decomposition.PCA**

Führen Sie die gleiche Projektion auf die ersten beiden Hauptkomponenten mit scikit learn durch und vergleichen Sie die Ergebnisse

# Explained variance

Die Varianz der Daten hat sich durch die Transformation nicht geändert. In Formeln:  $\sum_i (X^T X)_{i,i} = \sum_i (V^T X)^T (V^T X)_{i,i} = \sum_i D_{i,i}$ .

Links die alte Covarianz-Matrix, rechts die Neue, und  $D = \text{diag}(\lambda_1, \dots, \lambda_n)$ .

Die *explained variance* der  $i$ -Hauptkomponente ist  $\frac{\lambda_i}{\sum_j \lambda_j}$

# Aufgabe zu Explained variance

Nutzen Sie `pca.explained_variance_ratio` um die explained variance der ersten beiden Hauptkomponenten zu finden

Wie viele Hauptkomponenten brauchen Sie, um 0.95 der Varianz zu erhalten? Tipp: `PCA(n_components=xx)` lässt sich auch mit Zahlen kleiner 1 initialisieren

# Rekonstruktionsfehler

Unsere Projektion war

$$X_p = X \cdot V_k$$

wobei  $V_k$  die ersten  $k$  Spalten von  $V$  sind.

Die Rekonstruktion ist

$$X_r = X_p \cdot V_k^T$$

Das ist die obere Gleichung von rechts mit  $V_k^T$  multipliziert.

# Aufgabe zu Rekonstruktionsfehler

Führen Sie für die PCA, die 95% der Varianz erhalten hat, eine Rekonstruktion durch. Nutzen Sie dazu `pca.inverse_transform`

Visualisieren Sie die ersten zwei Bilder Ihres Ergebnis.

# Kernel PCA

Der Kernel Trick funktioniert für PCA ähnlich wie für andere Verfahren

Die lineare Projektion auf einen Unterraum im Feature Space entspricht einer nichtlinearen Transformation im Original Space

Kernel Trick: Man braucht den Feature Space nur implizit

# Aufgabe zu Kernel PCA

Führen Sie auf den Swiss roll dataset einen kernel PCA mit einem rbf und einen sigmoid Kernel durch

Experimentieren Sie mit verschiedenen Werten für  $\gamma$

# Random Projection

- Statt Varianz maximieren kann man auch so projizieren, dass Abstände erhalten bleiben
- Das geht erstaunlicherweise mit zufällig gewählten Unterräumen
- Nach genug Versuchen ist das Ergebnis ziemlich gut: Johnson-Lindenstrauss Lemma

# Random Projection

Exakte mathematische Formulierung:

- Sei  $0 < \epsilon < 1$ ,  $n$  Punkte im  $\mathbb{R}^d$ . Wähle  $k > \frac{4 \log(n)}{\epsilon^2/2 - \epsilon^3/3}$ . Dann existiert eine Projektion  $p: \mathbb{R}^d \rightarrow \mathbb{R}^k$  so dass

$$(1 - \epsilon) \|x_i - x_j\|^2 \leq \|p(x_i) - p(x_j)\| \leq (1 + \epsilon) \|x_i - x_j\|^2$$

- D.h. es existiert immer eine Projektion, die den Abstand erhält

# Random Projection Implementierung

Brute Force:

- Ziehe  $k$  Vektoren aus dem  $\mathbb{R}^d$
- Falls nicht linear unabhängig: starte von vorne
- Projiziere auf Unterraum, der von gezogenen Vektoren aufgespannt wird
- Falls Unterraum nicht gut: starte von vorne

# Aufgabe zu Random Projection

Führen Sie mit

`sklearn.random_projection.GaussianRandomProjection` eine  
Projektion des MNIST Datensatzes auf 10 Dimensionen durch

Was passiert, wenn man keine Dimension vorgibt?

Wie sieht die Rekonstruktion aus?

# Random Projection Implementierung

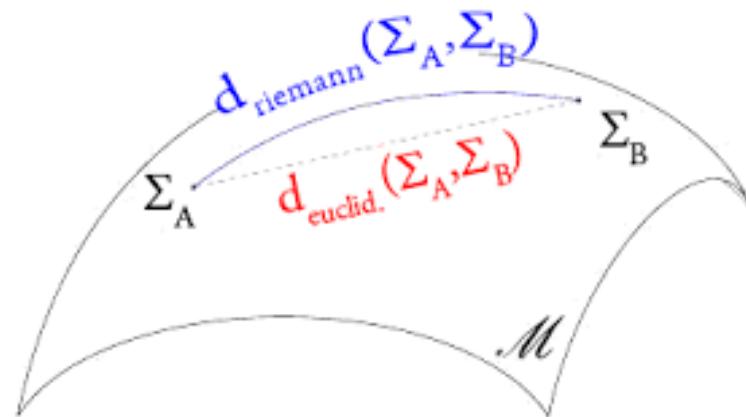
Ohne Brute Force:

- Konstruiere eine  $d \times k$  Matrix  $R$  mit Einträgen 0 oder 1. An jeder Stelle mit Wahrscheinlichkeit 1/2 gewürfelt
- Die Projektion  $\frac{1}{\sqrt{k}} \cdot R \cdot X$  hat mit hoher Wahrscheinlichkeit die gewünschten Eigenschaften

**Jetzt gekrümmte Daten**

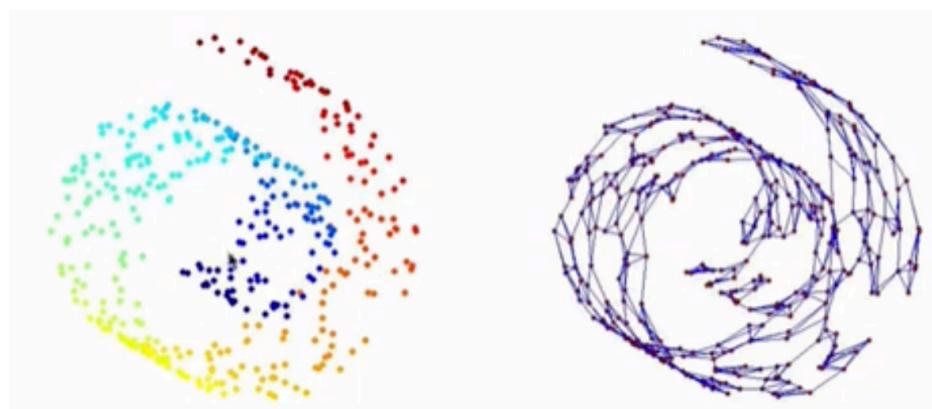
# Erster Ansatz für gekrümmte Daten

- Große Abstände werden anders gemessen (mit Geodäten)
- Kleine Abstände dagegen wie gewohnt (Euklidischer Abstand)
- Suche Vektoren in niedrig dimensionalen Raum mit ähnlichen Abständen



# Isomap

- Bauen einen k Nearest Neighbour Graph
- Abstand = Kürzester Weg im Graph



# Isomap

- Gemessenen Abstände werden in einer Matrix  $D$  erfasst

$$D = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ \vdots & & & \\ d_{m1} & d_{m2} & \dots & d_{mm} \end{pmatrix}$$

- Suche dann Vektoren  $x_1, \dots, x_m$  in einem Raum kleiner Dimension (z.Bsp. 2) so dass

$$d(x_i, x_j) = d_{ij}$$

gilt. Vorgehen heißt Multi-Dimensional-Scaling

# Aufgabe zu Isomap

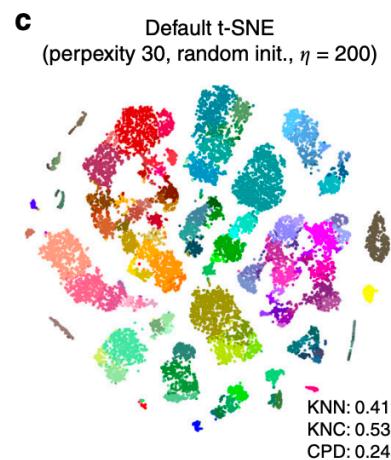
- Wenden Sie den Isomap Algorithmus auf das Swiss Roll Dataset an
- Ab welchem Wert für den Parameter `n_neighbors` beginnt der Algorithmus zu versagen?

# Weitere Vertreter dieser Algorithmenklasse

- Locally linear embedding
- UMAP
- Laplacian eigenmaps
- ...

# Zweiter Ansatz für gekrümmte Daten

- Cluster Strukturen in den Daten erkennen
- Nicht geeignet für Preprocessing, sondern Visualisierungstechnik



# t-SNE

- Gesprochen: tee-Snee.
- Starten mit Daten  $x_1, \dots, x_m \in \mathbb{R}^d$ . Ziel: Einbettung in  $\mathbb{R}^2$ .
- Lege um jeden Punkt  $x_i$  eine Dichtefunktion und berechne Wahrscheinlichkeiten  $p_{i,j}$ , das Punkt  $x_i$  zu  $x_j$  gehört. Achtung, nicht symmetrisch, hängt stark von Parameterwahl ab!
- Definiere Ähnlichkeitsfunktion  $q_{i,j}$  auf  $\mathbb{R}^2$  mit der  $t$ -Verteilung
- Versuche, die beiden Verteilungsfunktionen so ähnlich wie möglich zu machen

# t-SNE in Formeln

$$p_{ij} = \frac{\exp\left(\|x_i - x_j\|^2/(2\sigma_i^2)\right)}{\sum_{k \neq i} \exp\left(\|x_i - x_k\|^2/(2\sigma_i^2)\right)}$$

- Die Parameter  $\sigma_i$  hängen vom Punkt  $x_i$  ab. Dichte Regionen bekommen kleine  $\sigma_i$ , unbesiedelte große  $\sigma_i$
- Grobe Definition: Wähle  $\sigma_i$  so, dass Perplexity der Dichteschätzung zum vorgegebenen Parameter passt

# t-SNE in Formeln

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq i} \left(\|x_i - x_k\|^2\right)^{-1}}$$

- Modellierung durch  $t$ -Verteilung. Keine Parameterwahl

# t-SNE Heuristik

- Große Werte von  $p_{i,j}$  bei kleinen Werten von  $q_{i,j}$  werden hoch bestraft
- Kleine Werte von  $p_{i,j}$  bei großen Werten von  $q_{i,j}$  werden nur gering bestraft
- Heuristisch hat man abstoßende Kräfte zwischen allen Punkten, aber benachbarte Punkte ziehen sich außerdem noch an

# Aufgabe zu t-SNE

- Führen Sie mit t-SNE eine Projektion der ersten 500 Zeilen des MNIST Datensatzes auf zwei Dimensionen durch
- Variieren Sie den perplexity Parameter und visualisieren Sie Ihre Ergebnisse

# t-SNE Bewertung

- Macht Spaß und gibt schöne Bilder
- Es gibt außer Perplexity noch viele Parameter, an denen man tunen kann
  - Gefahr: Man tuned so lange, bis man das sieht, was man erwartet
- Geometrie geht verloren
  - Clustergröße, Abstände zwischen Clustern etc. haben keine Bedeutung
- Insgesamt sind die Plots schwierig zu interpretieren

# Feature Selection

# Definition Feature Selection

Die Menge der Predictors  $P = \{x_1, \dots, x_m\}$  auf eine Teilmenge  $Q \subset P$  einschränken

Achtung: Das ist nicht das gleiche wie Dimensionsreduktion, sondern eine spezielle Form der Dimensionsreduktion.

# Ziele Feature Selection

Beschränken uns auf supervised learning.

## 1. Ein Problem im Zusammenspiel zwischen Predictors und Modell lösen

- Support Vector Machines und Neuronale Netze reagieren empfindlich auf irrelevante Predictors
- Lineare und logistische Regression reagiert empfindlich auf korrelierte Predictors

## 2. Modell Komplexität reduzieren

- Reduziert Kosten für zusätzliche Daten
- Reduziert Kosten für Training durch Reduktion der Datenmenge
- Erhöht Wartbarkeit für Modell

# Missverständnis bei Feature Selection

**Filtering out uninformative predictors will lead to greater interpretability.**

Bei wenigen Samples und vielen Features gibt es viele optimale Kombinationen. *Eine* lokal optimale Kombination sagt nichts über Wichtigkeit der Features aus.

Selbst bei global optimaler Lösung haben wir nie perfekte Daten.

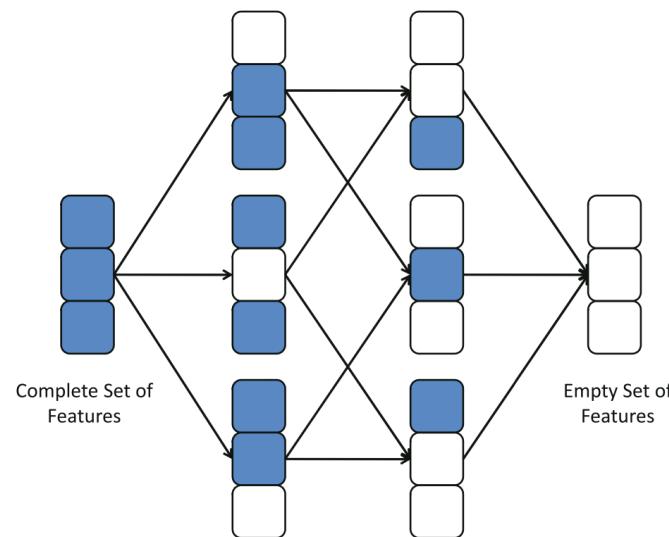
# Fragestellungen

- Wie suche ich nach optimalen subsets meiner Features?
- Wie evaluiere ich ein gegebenes subset?
- Nach welchen Prinzipien nehme ich ein Feature hinzu oder nehme eins weg?

# Komplexität des Problems

Bei  $m$  Features gibt es  $2^m$  Kombinationen der Features.

Z.Bsp.  $m = 3$ , dann 8 Kombinationen



# Überblick Feature Selection

Drei übergeordnete Methoden:

- Intrinsische oder implizite Methoden
- Filter
- Wrapper

# Intrinsische Methoden

Das Modell erkennt selber, welche Features irrelevant sind

Beispiele:

- Bäume. Nicht genutzte Features können eliminiert werden
- Regularisierer: Koeffizienten von Predictors werden zu null gestaucht. Diese können entfernt werden.

# Filter und Wrapper

## Filter

- Vorauswahl **vor** Trainieren des Modells mit statistischen Methoden
- Filter sind simpel und schnell

## Wrapper

- Wiederholtes Trainieren des Modells mit verschiedenen Features
- Wrapper haben lange Laufzeit und sind ressourcenintensiv

# Bewertung Intrinsischer Methoden

## Positiv

- Direkter Zusammenhang zwischen Feature Auswahl und Zielfunktion
- Schnell

## Negativ

- Auswahl ist Modell abhängig
- Auswahl ist fast immer *greedy*

# Bewertung Filter

## Positiv

- Filter erkennen gut individuelle Zusammenhänge zwischen Predictor und Zielvariable
- Schnell und einfach zu implementieren

## Negativ

- Disconnect zwischen statistischer Bewertung (z.Bsp. Signifikanz) und Ziel (bessere predictive Performance)
- Die meisten Filter bewerten nur ein Feature auf einmal. Sie verpassen so Interaktionen zwischen Features

# Bewertung Wrapper

## Positiv

- Evaluieren viele verschiedene Kombinationen von Teilmengen der Features
- Das meiste Potential, eine optimale Lösung zu finden

## Negativ

- Brauchen sehr lange
- Teure Modelle haben oft den meisten Bedarf an Feature Selection (SVM, NN)
- Gefahr von Overfitting

**Deepdive Filter**

# Welcher Filter passt?

- Für kategorischen Predictor und kategorisches Target
  - Contingency Table und  $\chi^2$  Test
  - Falls nur zwei Level im Predictor: odds ratio
- Für kategorischen Predictor und numerisches Target
  - Anova Test
  - Falls nur zwei Level im Predictor: t Test
  - Fläche unter ROC Kurve oder Precision Recall Kurve (mit vertauschten Rollen)

# Welcher Filter passt?

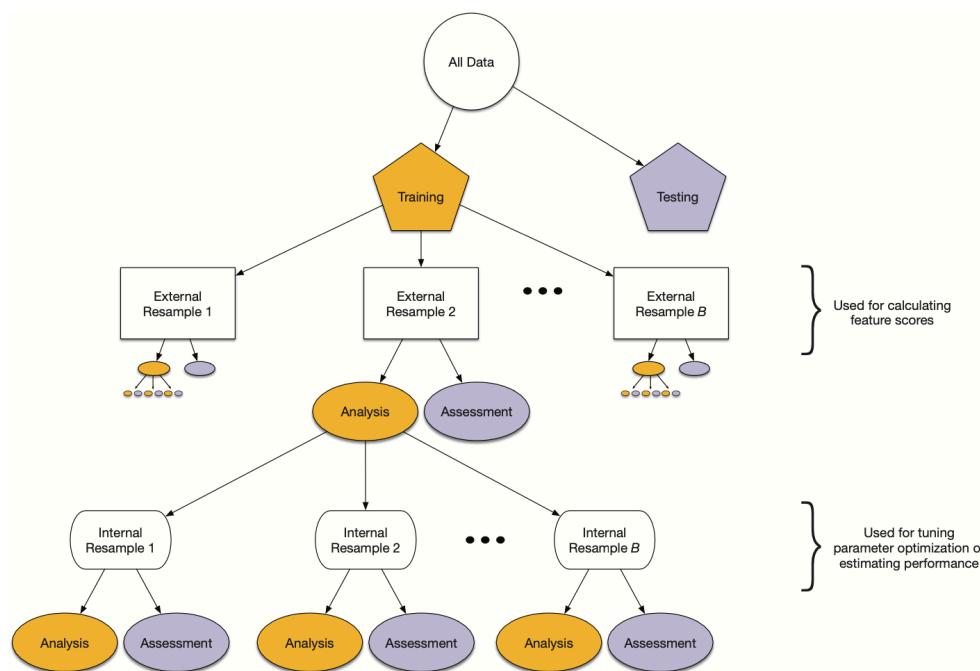
- Für numerischen Predictor und kategorisches Target
  - Contingency Table und  $\chi^2$  Test (mit vertauschten Rollen)
  - Mutual Information Classifier
- Für numerischen Predictor und numerisches Target
  - Korrelation
  - Maximaler Informations Koeffizient
  - Generalised Linear Models

# Vergleich der Ergebnisse

- Wenn alle Predictors vom gleichen Typ gibt es kein Problem
- Sind kategorische und numerische Predictors vorhanden, ist es nicht klar.  
Z.Bsp. Fläche unter ROC Kurve vs.  $t$ -Statistik.
- Meistens werden alle Ergebnisse in einen  $p$ -Wert transformiert.
- Nachteil:  $p$ -Werte sind anfällig für false positives

# Verhindern von false positives

Ganz viel resamplen! Vorsicht: Resamplen für hyper-parameter Tuning muss nachgelagert werden



# Aufgabe zu Filter

- Nutzen Sie `from sklearn.feature_selection import mutual_info_classif` um die zwei besten Predictors für den Parkinson Datensatz zu identifizieren
- Nutzen Sie wieder den Mutual Info Classifier um die top 20% der Predictors zu identifizieren

# Kombinierte Methoden

- Führen zunächst eine einfache Methode aus, die intrinsisch eine Bewertung der Features vornimmt. Z.Bsp. ein lineares Modell mit L1 Norm.
- Uninteressante Features werden heraus gefiltert
- Auf den verbleibenden Features trainieren wir unser Ziel-Modell

# Aufgabe zu kombinierten Methoden

- Nutzen Sie `from sklearn.svm import LinearSVC` um einen Linearen Support Vector Classifier mit L1 Penalty zu trainieren
- Nutzen Sie das Attribut `coeff_` des Classifiers um die 10 besten Features zu identifizieren

# Aufgabe zu kombinierten Methoden

- Führen Sie die gleiche Transformation mit der Methode `SelectFromModel` aus. Setzen Sie dabei den Regularisierungsparameter  $C=0.01$
- Nutzen Sie anschließend die Methode `.transform` des Outputs von `SelectFromModel`

# Aufgabe zu kombinierten Methoden

- Bauen Sie eine Pipeline, die zuerst mit `SelectFromModel` und einem `lsvc` die besten Features heraus sucht
- Dann soll auf dem reduzierten Datensatz ein `RandomForest` trainiert werden
- Wie gut ist die confusion Matrix auf dem reduzierten Test Trainingsset?  
Den `StandardScaler` auf den Testdaten nicht vergessen!

# **Deepdive Wrapper**

# Vorteil gegenüber selectFromModel

- Funktioniert auch für Modelle ohne `coef_` oder `feature_importances_` Attribut
- Auswahl der zu entfernenden Features erfolgt über Kreuzvalidierungsscore des Modells

# Hyperparameter bei Wrappern

- Anzahl Teilmengen, die evaluiert werden sollen
- Größe der Teilmengen
- Hyperparameter sind nicht in scikit learn implementiert
- Hyperparametertuning führt zu noch mehr Kreuzvalidierungen und damit noch mehr Trainingsläufen

# Wrapper: Backward

- Starten mit dem vollen leeren Feature Set
- Auf Basis des kreuzvalidierten Scores des ersten Trainingslauf wird ein Feature entfernt.
- Stopping Kriterium: Fahre fort, bis keine Änderung mehr im Score oder gewünschte Anzahl Features erreicht ist

# Aufgabe zu Backwards Wrapper

- Nutzen Sie `from sklearn.feature_selection import SequentialFeatureSelector` um einen Backwards Wrapper zu implementieren
- Der Backwards Wrapper soll den Datensatz auf 150 Features reduzieren
- Das genutzte Modell soll wieder ein Random Forest Classifier sein
- Brechen Sie die Berechnung ab sobald der Code läuft, sonst dauert es zu lange

# Wrapper: Forward

- Starten mit einem leeren Feature Set
- Auf Basis des kreuzvalidierten Scores des ersten Trainingslauf wird ein Feature hinzugefügt.
- Stopping Kriterium: Fahre fort, bis keine Änderung mehr im Score oder gewünschte Anzahl Features erreicht ist

# Aufgabe zu Forwards Wrapper

- Nutzen Sie from sklearn.feature\_selection import SequentialFeatureSelector um einen Forwards Wrapper zu implementieren
- Der Forwards Wrapper soll den Datensatz auf 2 Features reduzieren
- Reduzieren Sie die Anzahl der Kreuzvalidierungen auf 2
- Das genutzte Modell soll wieder ein Random Forest Classifier sein
- Finden Sie heraus, in welchem Attribut die genutzten Spalten stehen

# Forward und Backward sind Greedy

- Können nicht gewährleisten, dass wir die optimale Teilmenge gefunden haben
- Vielleicht war eines der Features, dass wir im Schritt  $n$  nicht berücksichtigt haben, doch Teil des optimalen Lösungssets

# Weitere Suchstrategien

- Exhaustive: Alle Teilmengen werden evaluiert. Nicht praktikabel
- Stochastic Methods: Simulated Annealing und Genetic Algorithms
  - Starte mit zufälligem Feature Set und fester Anzahl Iterationen
  - Füge zufällig neue Features hinzu oder entferne zufällig Features
  - Akzeptanzkriterium für neues Featureset mit Toleranz