



Digital Applications & Data Management

WS25/26

Dr. Jens Kohl

Roadmap Vorlesung



1. Einführung und Übersicht
2. Grundlagen Data Science
3. Vorgehen Data Science Use Case
4. Case Study Data Science
5. Grundlagen unüberwachtes Lernen
6. Grundlagen überwachtes Lernen (tabellarische Daten)
7. Case Study überwachtes Lernen (tabellarische Daten)
8. Grundlagen überwachtes Lernen (Bilddaten)
9. Case Study überwachtes Lernen und Transfer Learning (Bilddaten)
10. Grundlagen Generative AI
11. Generative AI mit Texten und Prompt Engineering
12. Agentic AI
13. Ausblick: Machine Learning in der Cloud und Reinforcement Learning



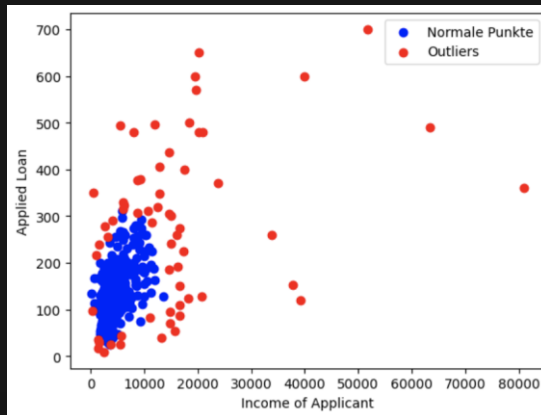
Vorlesung 5:

Grundlagen unüberwachtes Lernen



Was machen wir heute?

Motivation



Cluster 0 hat
Einkommen von mindestens 3166 bis maximal 4895
Loan von mindestens 30.0 bis maximal 255.0

Cluster 1 hat
Einkommen von mindestens 7578 bis maximal 12000
Loan von mindestens 104.0 bis maximal 308.0

Cluster 2 hat
Einkommen von mindestens 150 bis maximal 3159
Loan von mindestens 45.0 bis maximal 255.0

Cluster 3 hat
Einkommen von mindestens 4917 bis maximal 7451
Loan von mindestens 70.0 bis maximal 315.0

Wird oft zusammen gekauft



Gesamtpreis: **77,90 €**

Alle drei in den Einkaufswagen

[i](#) Einer der beiden Artikel ist schneller versandfertig. [Details anzeigen](#)

In den bisherigen Vorlesungen haben wir Zusammenhänge in den Daten selber versucht zu finden.
Unüberwachtes Lernen ermöglicht uns, dies automatisiert zu machen...



Kategorien Machine Learning

Unsupervised Learning

Algorithmus lernt ohne vorher definierte Zielwerte oder Belohnung

Supervised Learning

Algorithmus lernt Funktion, die Eingabegröße auf vorher definierte Outputs mappt

Reinforcement Learning

Algorithmus lernt selbstständig mit Ziel, eine „Belohnung“ zu maximieren

Übersicht Unsupervised Learning



Wofür brauchen wir das?

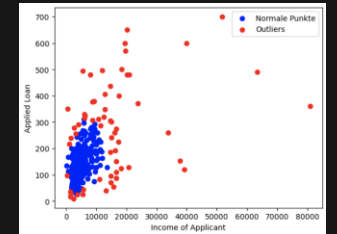
Verbesserung Datenqualität

Reduktion (sehr großer) Datenmengen

Entdecken unbekannter Zusammenhänge

Was schauen wir uns an?

Entdecken von Anomalien in Daten



Clustering/ Segmentierung

Cluster 0 hat
Einkommen von mindestens 3166 bis maximal 4895
Loan von mindestens 30.0 bis maximal 255.0

Cluster 1 hat
Einkommen von mindestens 7578 bis maximal 12000
Loan von mindestens 104.0 bis maximal 308.0

Cluster 2 hat
Einkommen von mindestens 150 bis maximal 3159
Loan von mindestens 45.0 bis maximal 255.0

Cluster 3 hat
Einkommen von mindestens 4917 bis maximal 7451
Loan von mindestens 70.0 bis maximal 315.0

Clustering/ Segmentierung
Association Rules

Wird oft zusammen gekauft

Gesamtpreis: 77,90 €
Alle drei in den Einkaufswagen

! Einer der beiden Artikel ist schneller versandfertig. Details anzeigen



Clustering



Clustering

Übersicht

Ziel: Einteilen eines Datensatzes in k verschiedene Gruppen.

Was bringt das?

- Erkennen von Strukturen in einem Datensatz
- Einordnen Datensatz in kleinere Gruppen mit ähnlichen Daten/ Verhalten

Aber: Ergebnis muß von Experten beurteilt werden, Algorithmus liefert keine Begründung!

Anwendungsgebiete:

- Kundenanalyse
- Sequenzanalyse Biologie (Entdecken von charakteristischen Teilen in der DNA)
- Image Segmentation



Clustering: Nutzerdaten im Web

index	Data Provider Name	Data Provider ID	Segment ID	Segment Name
0	Nielsen Marketing Cloud	39	4015114	Consumer Targets - Interests - Auto and Other Vehicles - Makes and Models - Makes - Hyundai (Exelate)
1	Nielsen Marketing Cloud	39	4015099	Consumer Targets - Interests - Auto and Other Vehicles - Makes and Models - Makes - Aston Martin (Exelate)
2	Nielsen Marketing Cloud	39	4015077	Consumer Targets - Custom Characteristics - Software - Browsers - Firefox (Exelate)
3	Nielsen Marketing Cloud	39	2332154	Consumer Targets - Custom Characteristics (Exelate)
4	Nielsen Marketing Cloud	39	5174108	Bombora B2B Intent Signals - Human Resources - Recruitment, Hiring and Onboarding
5	Nielsen Marketing Cloud	39	1792609	B2B Targets - Seniority - Support (Exelate)
6	Nielsen Marketing Cloud	39	1792775	B2B Targets - Industry - Wholesale - Durable Goods (Exelate)
7	Nielsen Marketing Cloud	39	1792700	B2B Targets - Industry - Manufacturing - Petroleum and Petroleum Related (Exelate)
8	Nielsen Marketing Cloud	39	4016168	Tech Targets - From WhoToo - Industry - Manufacturing - Petroleum and Petroleum Related (Exelate)
9	Nielsen Marketing Cloud	39	4016151	Tech Targets - From WhoToo - Industry - Manufacturing - Auto (Exelate)
10	Nielsen Marketing Cloud	39	4016072	Tech Targets - From WhoToo - Industry - Agricultural - Crops (Exelate)
11	Nielsen Marketing Cloud	39	4016029	Tech Targets - From WhoToo - Company - Technology Use - Vertical Market Solutions - Real Estate (Exelate)
12	Nielsen Marketing Cloud	39	4015987	Tech Targets - From WhoToo - Company - Technology Use - Data Center, Network and Platform Computing - Operating Systems and Computing Languages (Exelate)
13	Nielsen Marketing Cloud	39	4015115	Consumer Targets - Interests - Auto and Other Vehicles - Makes and Models - Makes - Infiniti (Exelate)
14	Nielsen Marketing Cloud	39	4015098	Consumer Targets - Interests - Auto and Other Vehicles - Makes and Models - Makes - Alfa Romeo (Exelate)
15	Nielsen Marketing Cloud	39	4015076	Consumer Targets - Custom Characteristics - Software - Browsers - Chrome (Exelate)
16	Nielsen Marketing Cloud	39	2332155	Consumer Targets - Custom Characteristics - Cross-channel (Exelate)
17	Nielsen Marketing Cloud	39	5174130	Bombora B2B Intent Signals - Technology - Cloud Computing
18	Nielsen Marketing Cloud	39	5174109	Bombora B2B Intent Signals - Human Resources - Training and Development
19	Nielsen Marketing Cloud	39	1792608	B2B Targets - Seniority - Staff (Exelate)
20	Nielsen Marketing Cloud	39	1792701	B2B Targets - Industry - Manufacturing - Railroad and Railroad Related (Exelate)
21	Nielsen Marketing Cloud	39	1792547	B2B Targets - Functional Area - Engineering - Chemical (Exelate)
22	Nielsen Marketing Cloud	39	4016169	Tech Targets - From WhoToo - Industry - Manufacturing - Railroad and Railroad Related (Exelate)
23	Nielsen Marketing Cloud	39	4016150	Tech Targets - From WhoToo - Industry - Manufacturing - Apparel and Apparel Related (Exelate)
24	Nielsen Marketing Cloud	39	4016073	Tech Targets - From WhoToo - Industry - Agricultural - Forestry, fish and game (Exelate)

Show 25 per page

1210100700790800

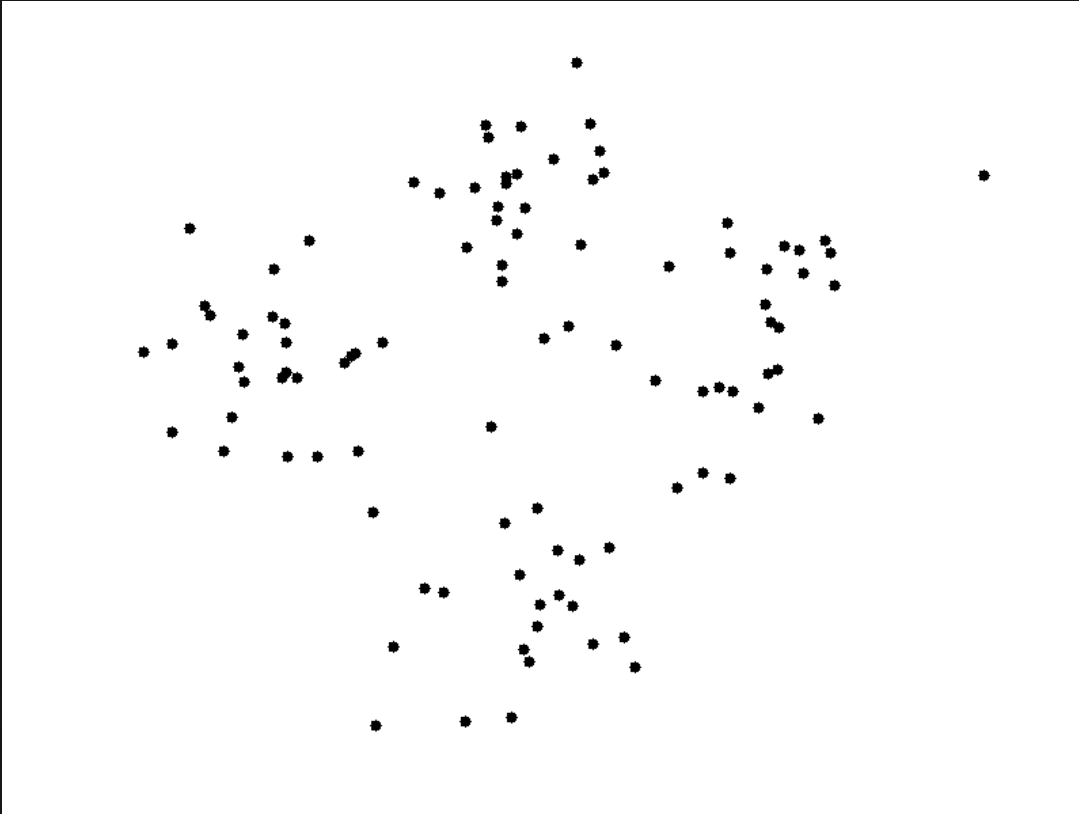
Leak von Xandr veröffentlicht, das zeigt wie anhand Ihres Such-/Browserverlaufs Sie automatisch für relevante Werbung kategorisiert werden. Enthält viele kritische Attribute wie Krankheiten, Rasse,

Quellen: <https://themarkup.org/privacy/2023/06/08/from-heavy-purchasers-of-pregnancy-tests-to-the-depression-prone-we-found-650000-ways-advertisers-label-you>



Clustering: KMeans-Algorithmus

Beschreibung Algorithmus



Algorithmus für Clustering in k-Cluster:

1. Wähle k zufällige Punkte als Cluster-Zentren (Centroid).
2. Berechne für jeden Datenpunkt x die Distanz zu einem der Centroiden. Weise x dem Centroid mit geringstem Abstand zu.
3. Finde für jeden Cluster sein neues Zentrum durch Bilden des Durchschnitts aller seiner Punkte (Cluster ist arithm. Mittel).
4. Wiederhole Schritte 2 und 3 bis sich keine Cluster-Zuweisung mehr ändert.



Clustering

Fragestellung: was könnte ein interessanter Zusammenhang sein, den wir Clustern wollen?

- Wir nehmen den Datensatz Lohnvergabe aus Vorlesung 5.
- Für 2 Features ***applicant_income*** und ***loan_amount*** wollen wir ein Clustering machen.
- Wir haben bei loan_amount nans, also fehlende Werte, schon einmal in Zeile 1 stehen.
- Schauen wir uns doch mal genauer an, wie viele Null-Werte wir haben.

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

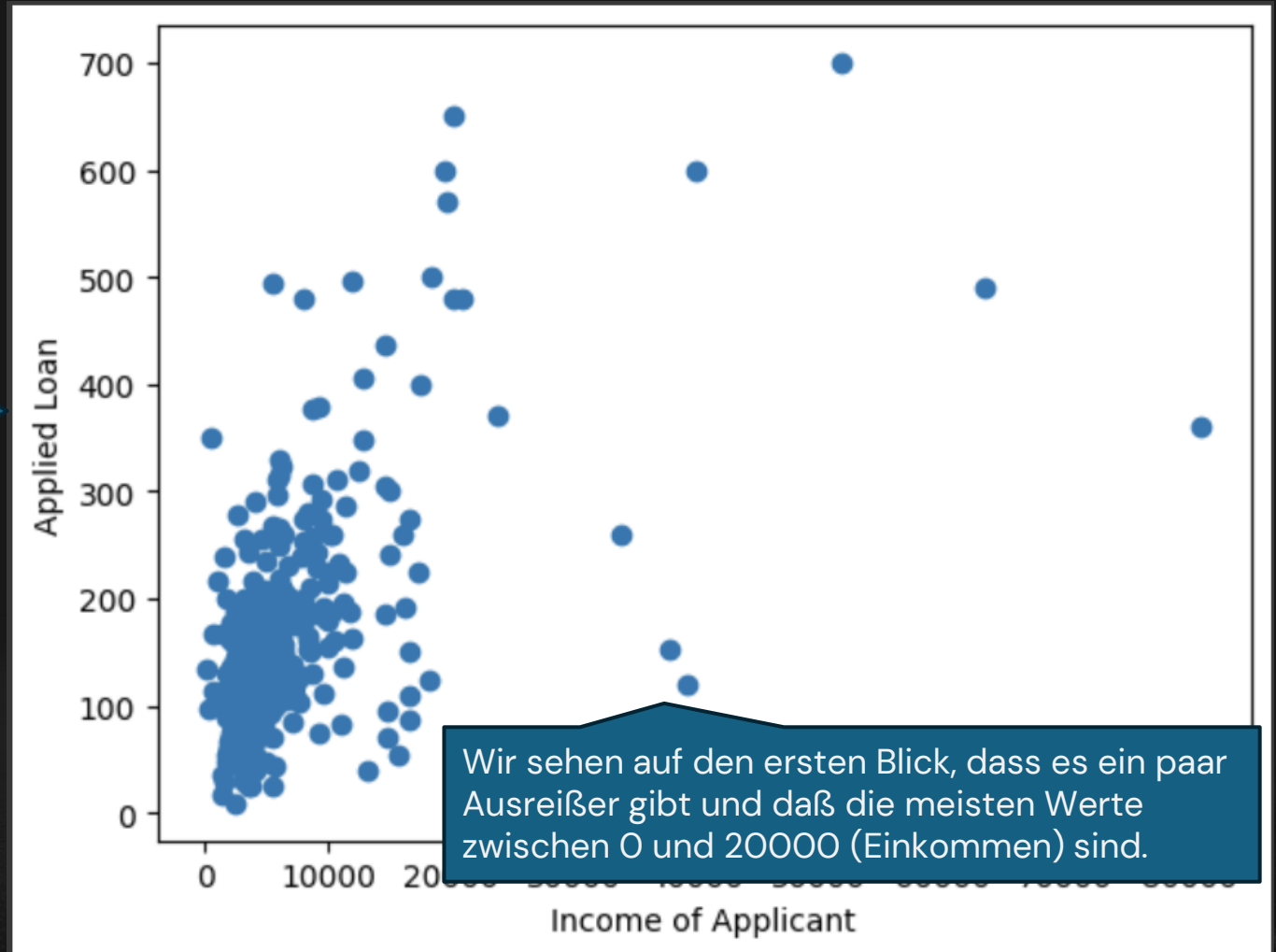
614 rows x 13 columns



Clustering

Aufgabe: Plotten der beiden Spalten Applicant_Income und Loan_Amount als Scatterplot

```
[ ] X = loan_df[["ApplicantIncome", "LoanAmount"]]  
  
plt.scatter(X["ApplicantIncome"], X["LoanAmount"]);  
plt.xlabel("Income of Applicant");  
plt.ylabel("Applied Loan");
```





Anomalieerkennung



Anomalieerkennung

Problem: Ausreißer

- Wir haben viele offensichtlicher Ausreißer, die das Cluster-Ergebnis negativ beeinträchtigen könnten.
- Diese werden wir mit dem Verfahren Isolation Forest identifizieren und den Datensatz dann um diese Ausreißer bereinigen:
 - Einteilung Daten in 2 Gruppen durch Isolation Forest: Ausreißer und Nichtausreißer
 - Visualisierung mittels Plot
 - Entfernen Ausreißer



Anomalieerkennung

Vorhersage der Ausreißer mit Isolation Forest

```
from sklearn.ensemble import IsolationForest

# Initiieren des Isolation Forest
isolation_forest = IsolationForest(
    n_estimators=100,
    contamination=0.1)

# bestimme outlier in X-Werten
y_hat = isolation_forest.fit_predict(X)
```



```
# Aufteilen der Datenmenge in 2 Untermengen mit Outliern und Nicht-outliern, damit man das schöner sehen kann per Graphik
# Herausfiltern aller Outlier.
# genauer gesagt: Non_outlier enthält alle vorhergesagten Werte die ungleich -1 sind
Non_outlier = y_hat != -1
Outlier = y_hat == -1 # analog obere Zeile

# wir erstellen zwei leere Dataframes in die wir outlier oder nicht outlier speichern
X_non_outlier = pd.DataFrame()
X_outlier = pd.DataFrame()

X_non_outlier["X_AppliantIncome_nonoutlier"], X_non_outlier["X_LoanIncome_nonoutlier"] = X["ApplicantIncome"][Non_outlier], X["LoanAmount"][Non_outlier]
X_outlier["X_AppliantIncome_outlier"], X_outlier["X_LoanIncome_outlier"] = X["ApplicantIncome"][Outlier], X["LoanAmount"][Outlier]
```



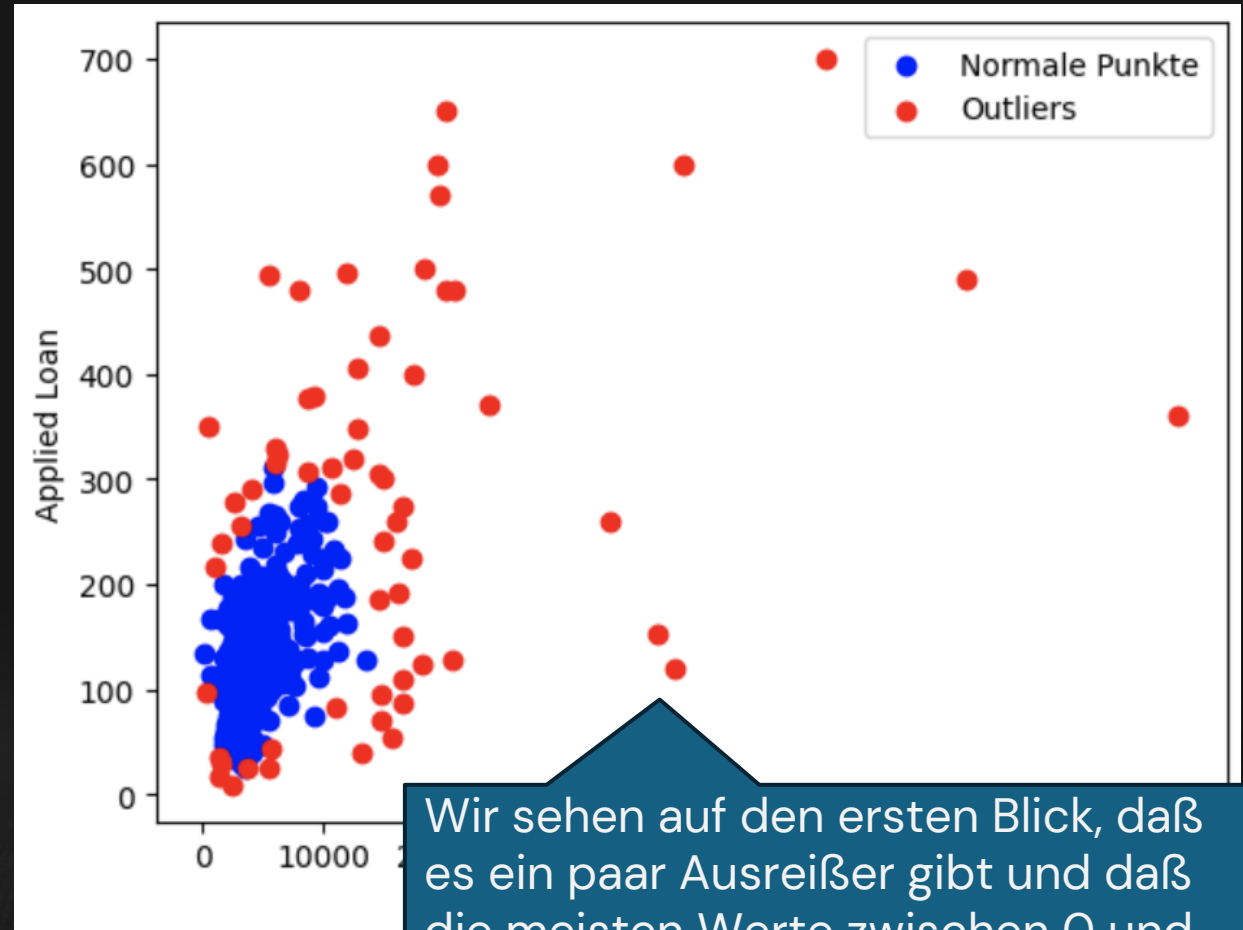
Anomalieerkennung

Scatterplot beider Gruppen

```
ax = plt.gca()
# Zeichne die Nicht-Outlier als blaue Punkte ("b")
ax.scatter(x=X_non_outlier["X_AppliantIncome_nonoutlier"],
           y=X_non_outlier["X_LoanIncome_nonoutlier"],
           c="b",
           label="Normale Punkte")

# Zeichne die Nicht-Outlier als rote Punkte ("r")
ax.scatter(x=X_outlier["X_AppliantIncome_outlier"],
           y=X_outlier["X_LoanIncome_outlier"],
           color="r",
           label="Outliers")

ax.legend() #zeichne eine Legende
plt.xlabel("Income of Applicant"); # benenne X-Achse
plt.ylabel("Applied Loan"); # benenne y-Achse
ax = plt.subplot() # zeichne Plot nachdem alle Einstellungen fertig
```

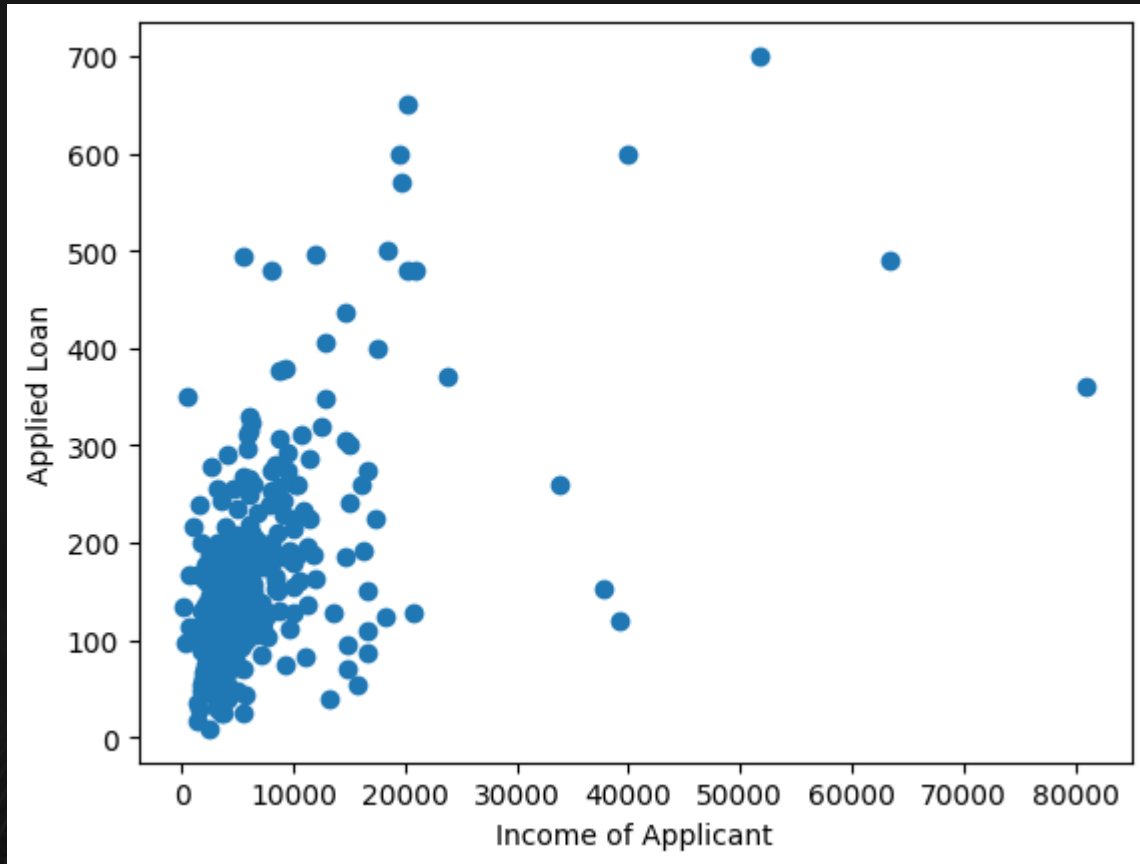


Wir sehen auf den ersten Blick, daß es ein paar Ausreißer gibt und daß die meisten Werte zwischen 0 und 20000 (Einkommen) sind.

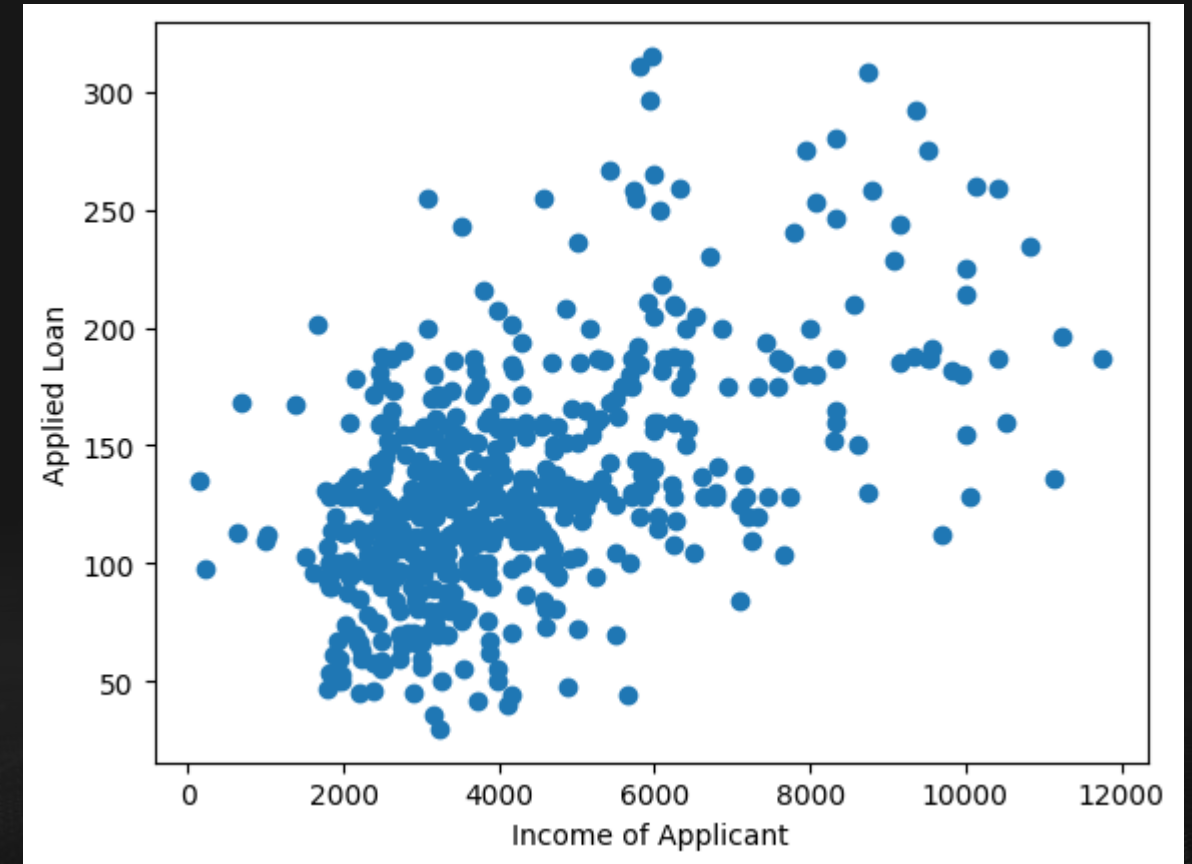


Clustering: KMeans

Ergebnis nach Filtern der Anomaliekennung



Vor Filtern anomaler Werte



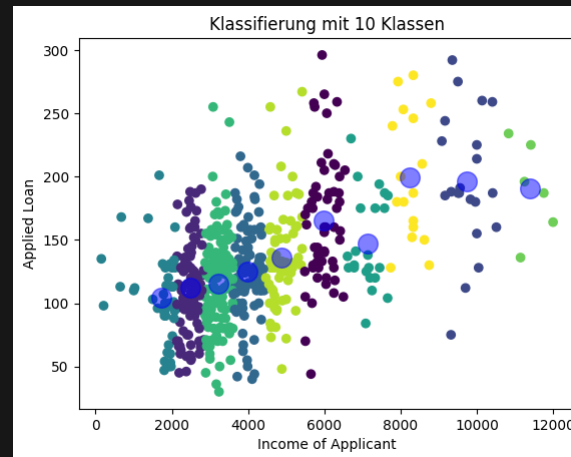
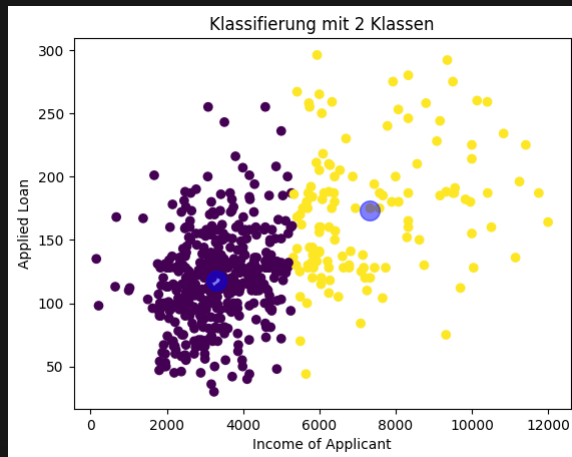
Nach Filtern anomaler Werte



Clustering: KMeans

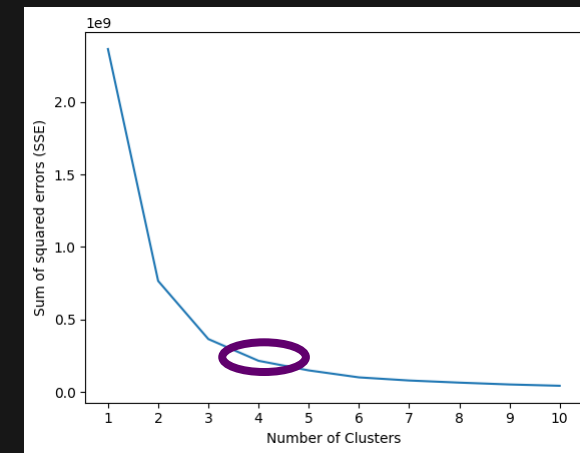
Problem: Identifikation der „idealen“ Anzahl von Clustern

Trade-off Anzahl Cluster:



- Je weniger Cluster, desto größer die Einteilung und je höher die Wahrscheinlichkeit einer Fehlklassifizierung.
- Je mehr Cluster, desto genauer die Einteilung, aber desto weniger erfolgt eine Generalisierung.

„Elbow“-trick

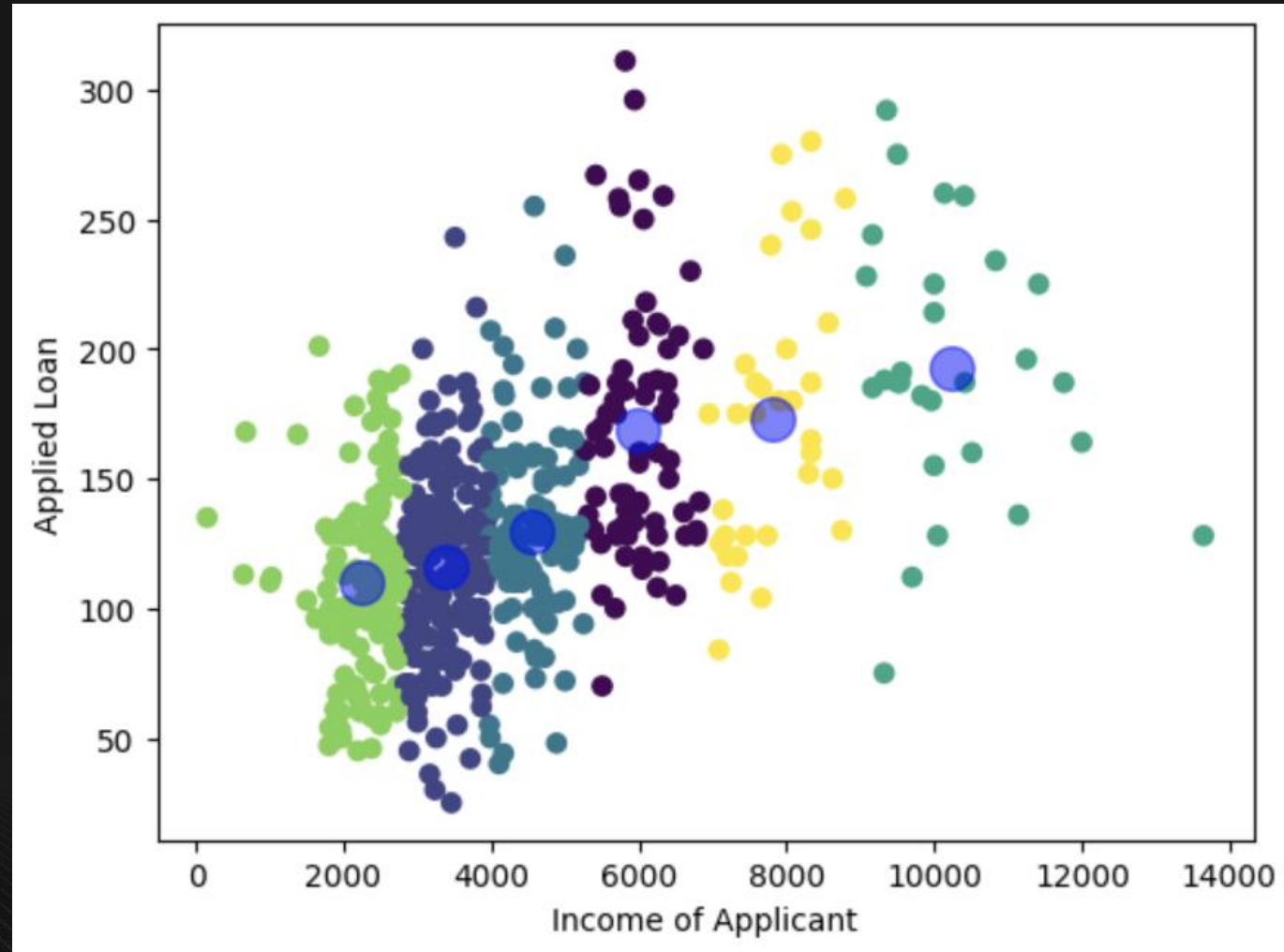


- Wiederholtes Durchführen kmeans für hohes k (>10).
- Kmeans-metrik sum of squared errors plotten (Summe der quadrierten Abstände aller Punkte zum Zentrum des nächsten Clusters; je kleiner die Summe, desto näher liegen Punkte am Mittelpunkt eines Clusters).
- Ellbogen finden



Clustering: KMeans

Identifikation der „idealen“ Anzahl von Clustern





Clustering: KMeans

Identifikation der „idealen“ Anzahl von Clustern

- Die Ellbogenmethode wird Robert I Thorndike zugeschrieben, der diese 1953 entwickelte.
- Der [Wikipedia-artikel](#) detailliert die Methode wie folgt:
 - "using the "elbow" or "knee of a curve" as a cutoff point is a common heuristic in mathematical optimization to choose a point where diminishing returns are no longer worth the additional cost. In clustering, this means one should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data.
 - The intuition is that increasing the number of clusters will naturally improve the fit (explain more of the variation), since there are more parameters (more clusters) to use, but that at some point this is over-fitting, and the elbow reflects this.
 - The idea is that the first clusters will add much information (explain a lot of variation), since the data actually consist of that many groups (so these clusters are necessary), but once the number of clusters exceeds the actual number of groups in the data, the added information will drop sharply, because it is just subdividing the actual groups. Assuming this happens, there will be a sharp elbow in the graph of explained variation versus clusters: increasing rapidly up to k (under-fitting region), and then increasing slowly after k (over-fitting region)"
- Wir verwenden 3 Schritte:
 1. Kmeans mit einer hohen Anzahl von Clustern trainieren. Dabei für jeden Durchlauf den Fehler, d.h. falsche Zuordnung, als quadrierten Wert abspeichern im Array sse. Dieser Fehler ist größer, je weniger Klassen es gibt. Man kann sich das also als Generalisierungsfehler vorstellen.
 2. Plotten der Graphik für die einzelnen Fehler im array sse
 3. Plot anschauen und den ellbogen finden. Das ist der punkt, an dem der plot schräg abfällt.



Clustering: KMeans

Identifikation der „idealen“ Anzahl von Clustern

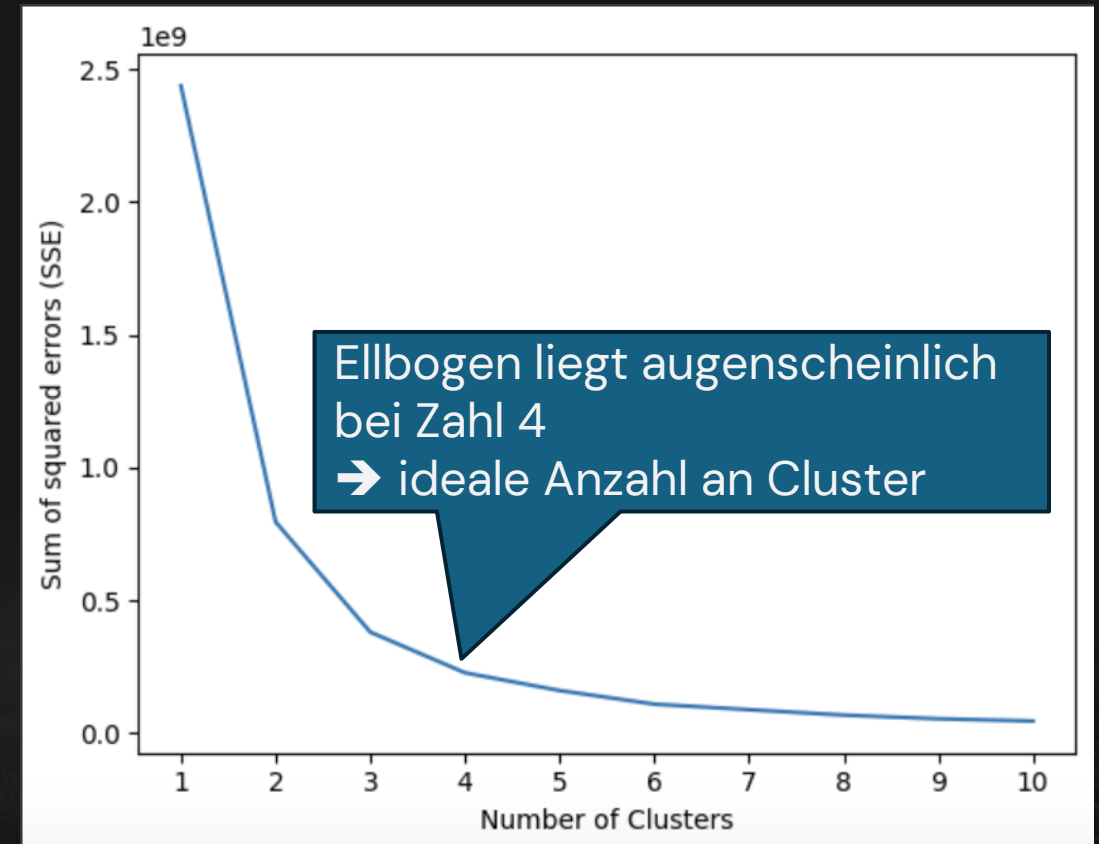
```
# Schritt 1: Trainieren
# in Liste sse speichern wir einzelnen Werte je K-Means Versuch
sse = []

# in dieser Schleife rufen wir kMeans für 1 bis 11 Cluster auf
for k in range(1, 11):
    kmeans_loop = KMeans(n_clusters=k);
    kmeans_loop.fit(X_non_outlier);
    #schreibe den Fehler in die Liste
    sse.append(kmeans_loop.inertia_);
```

„Ausprobieren“ für
wenige bis viele Cluster

```
# Schritt 2: Plotten der einzelnen Fehler je KMeans 1-11
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Sum of squared errors (SSE)")
plt
```

Visualisieren der Metrik
„falsch zugeordneter Punkte zu
Cluster“ je KMEANS





Clustering: KMeans

Identifikation der „idealen“ Anzahl von Clustern

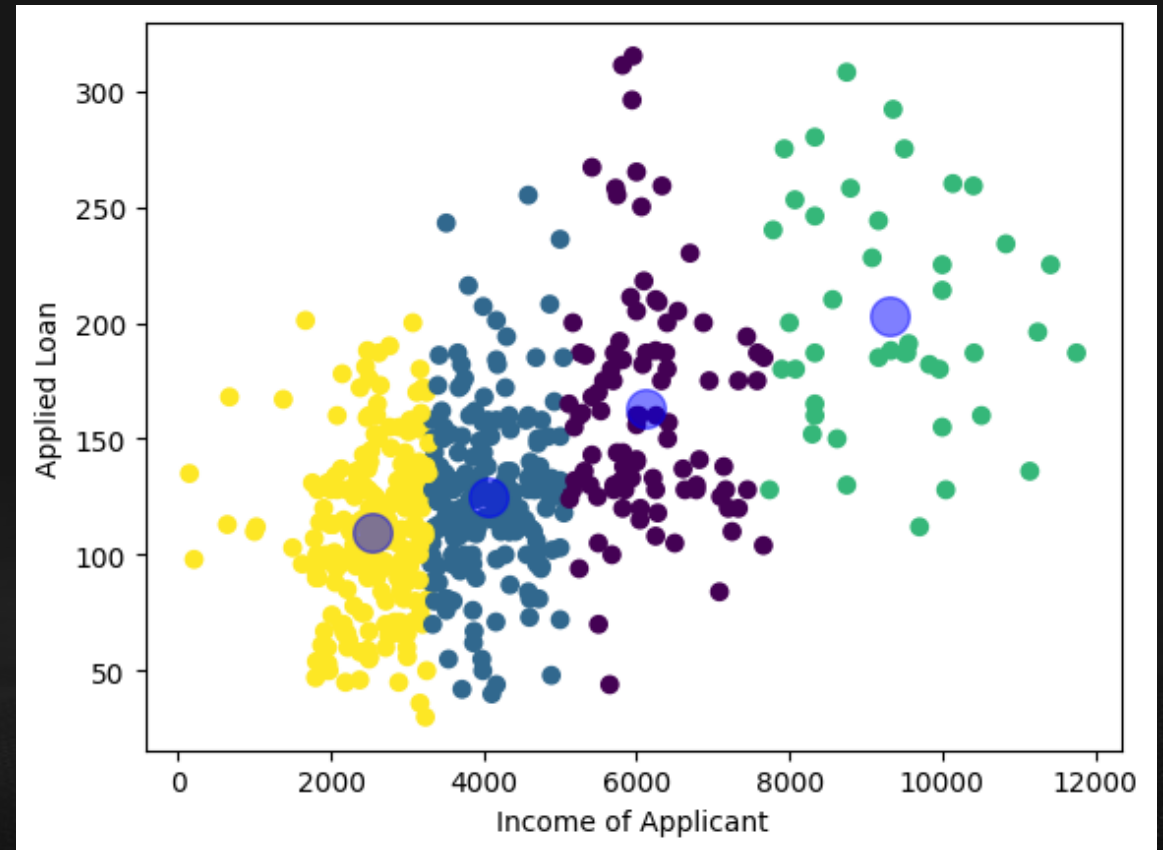
```
# deshalb nehmen wir die Zahl 4 als Parameter
kmeans_ellbow = KMeans(n_clusters=4)
kmeans_ellbow.fit(X_non_outlier)
cluster = kmeans_ellbow.predict(X_non_outlier)
```

```
centroids = kmeans_ellbow.cluster_centers_

# mach ein scatter plot für die Werte
plt.scatter(X_non_outlier["X_ApplicantIncome_nonoutlier"],
           X_non_outlier["X_LoanIncome_nonoutlier"],
           c=cluster, # zeichne kmeans-Cluster in verschiedene Farben
)

plt.xlabel("Income of Applicant");
plt.ylabel("Applied Loan");

# wir plotten die Mittelpunkte/Centroids aus dem vorigen Block
plt.scatter(centroids[:, 0],
           centroids[:, 1],
           c='blue',
           s=200,
           alpha=0.5)
```





Clustering: KMeans

Und was bringt das nun?

```
for i in range(4):  
    minIncomeValue = frame.loc[frame['cluster'] == i, "X_ApplicantIncome_nonoutlier"].min()  
    maxIncomeValue = frame.loc[frame['cluster'] == i, "X_ApplicantIncome_nonoutlier"].max()  
    minLoanValue = frame.loc[frame['cluster'] == i, "X_LoanIncome_nonoutlier"].min()  
    maxLoanValue = frame.loc[frame['cluster'] == i, "X_LoanIncome_nonoutlier"].max()  
    print("Cluster {} hat".format(i))  
    print("Einkommen von mindestens {} bis maximal {}".format(minIncomeValue, maxIncomeValue))  
    print("Loan von mindestens {} bis maximal {}".format(minLoanValue, maxLoanValue))
```



Cluster 0 hat
Einkommen von mindestens 3166 bis maximal 4895
Loan von mindestens 30.0 bis maximal 255.0

Cluster 1 hat
Einkommen von mindestens 7578 bis maximal 12000
Loan von mindestens 104.0 bis maximal 308.0

Cluster 2 hat
Einkommen von mindestens 150 bis maximal 3159
Loan von mindestens 45.0 bis maximal 255.0

Cluster 3 hat
Einkommen von mindestens 4917 bis maximal 7451
Loan von mindestens 70.0 bis maximal 315.0



Market Basket Analysis

Assoziationsregeln/ Market Basket Analyse



Ziel: automatisiertes Erkennen von Zusammenhängen/ Abhängigkeiten innerhalb eines Datensatzes

- Ermöglicht:
 - Definition einfacher, leicht verständlicher regeln: „wenn A gekauft wurde, dann auch B“.
 - **Aber: correlation does not imply causation!**

Anwendungsgebiete:

- Market Basket Analyse („Frequently bought together“)
- Recommender System

Wird oft zusammen gekauft



! Einer der beiden Artikel ist schneller versandfertig. [Details anzeigen](#)

Kunden, die diesen Artikel angesehen haben, haben auch angesehen





Assoziationsregeln/ Market Basket Analyse

Übersicht

- Das Lernen von Assoziationsregeln (Association rule learning) ist ein Verfahren, um aus Datenmengen automatisiert Regeln oder Abhängigkeiten zu lernen.
- Ein sehr bekanntes Beispiel hierfür ist die Anzeige von "Kunden, die dies kauften, kauften auch" auf Amazon. Das wird auch Market Basket Analysis genannt.
- Wir schauen uns das anhand eines beispielhaften Datensatzes eines Retail-Laden im Detail an.
- Den Datensatz können Sie inkl. kurzer Erklärung [hier](#) finden.



Assoziationsregeln/ Market Basket Analyse

Bestimmen der Regeln

Dataset

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2,55	17850	UK
536365	71053	WHITE METAL LANTERN	6	01.12.2010 08:26	3,39	17850	UK
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01.12.2010 08:26	2,75	17850	UK
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3,39	17850	UK
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3,39	17850	UK
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	01.12.2010 08:26	7,65	17850	UK
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	01.12.2010 08:26	4,25	17850	UK
536366	22633	HAND WARMER UNION JACK	6	01.12.2010 08:28	1,85	17850	UK
536366	22632	HAND WARMER RED POLKA DOT	6	01.12.2010 08:28	1,85	17850	UK
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	01.12.2010 08:34	1,69	13047	UK
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	01.12.2010 08:34	2,1	13047	UK
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	01.12.2010 08:34	2,1	13047	UK
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	01.12.2010 08:34	3,75	13047	UK
536367	22310	IVORY KNITTED MUG COSY	6	01.12.2010 08:34	1,65	13047	UK

Finde häufige Mengen

support	itemsets
0 0.102845	(6 RIBBONS RUSTIC CHARM)
1 0.100656	(JUMBO BAG WOODLAND ANIMALS)
2 0.115974	(PLASTERS IN TIN CIRCUS PARADE)
3 0.107221	(PLASTERS IN TIN SPACEBOY)
4 0.137856	(PLASTERS IN TIN WOODLAND ANIMALS)
5 0.818381	(POSTAGE)
6 0.137856	(REGENCY CAKESTAND 3 TIER)
7 0.157549	(ROUND SNACK BOXES SET OF 4 FRUITS)
8 0.245077	(ROUND SNACK BOXES SET OF 4 WOODLAND)
9 0.102845	(SPACEBOY LUNCH BOX)
10 0.126915	(WOODLAND CHARLOTTE BAG)
11 0.100656	(POSTAGE, PLASTERS IN TIN CIRCUS PARADE)
12 0.100656	(POSTAGE, PLASTERS IN TIN SPACEBOY)
13 0.118162	(POSTAGE, PLASTERS IN TIN WOODLAND ANIMALS)
14 0.120350	(POSTAGE, REGENCY CAKESTAND 3 TIER)
15 0.150985	(POSTAGE, ROUND SNACK BOXES SET OF 4 FRUITS)
16 0.225383	(POSTAGE, ROUND SNACK BOXES SET OF 4 WOODLAND)
17 0.115974	(WOODLAND CHARLOTTE BAG, POSTAGE)
18 0.131291	(ROUND SNACK BOXES SET OF 4 WOODLAND, ROUND SNA...
19 0.124726	(POSTAGE, ROUND SNACK BOXES SET OF 4 WOODLAND, ...

Erzeuge Regeln

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	shang_etriic
9	(ROUND SNACK BOXES SET OF 4 FRUITS)	(POSTAGE)	0.157549	0.818381	0.150985	0.958333	1.171012	0.022049	4.358862	0.173348
18	(ROUND SNACK BOXES SET OF 4 WOODLAND, ROUND SNA...	(POSTAGE)	0.131291	0.818381	0.124726	0.950000	1.160829	0.017280	3.632385	0.159486
3	(PLASTERS IN TIN SPACEBOY)	(POSTAGE)	0.107221	0.818381	0.100656	0.938776	1.147113	0.012909	2.966448	0.143649
11	(ROUND SNACK BOXES SET OF 4 WOODLAND)	(POSTAGE)	0.245077	0.818381	0.225383	0.919643	1.123735	0.024817	2.260151	0.145866
12	(WOODLAND CHARLOTTE BAG)	(POSTAGE)	0.126915	0.818381	0.115974	0.913793	1.116587	0.012109	2.106783	0.119591
7	(REGENCY CAKESTAND 3 TIER)	(POSTAGE)	0.137856	0.818381	0.120350	0.873016	1.066760	0.007532	1.430252	0.072589
1	(PLASTERS IN TIN CIRCUS PARADE)	(POSTAGE)	0.115974	0.818381	0.100656	0.867925	1.060539	0.005746	1.375117	0.064572
5	(PLASTERS IN TIN WOODLAND ANIMALS)	(POSTAGE)	0.137856	0.818381	0.118162	0.857143	1.047364	0.005344	1.271335	0.052453
15	(ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF 4 WOODLAND)	0.157549	0.245077	0.131291	0.833333	3.400298	0.092679	4.529540	0.837922
17	(POSTAGE, ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF 4 WOODLAND)	0.150985	0.245077	0.124726	0.826087	3.370730	0.087724	4.348810	0.828405
21	(ROUND SNACK BOXES SET OF 4 FRUITS)	(POSTAGE, ROUND SNACK BOXES SET OF 4 WOODLAND)	0.157549	0.225383	0.124726	0.791667	3.512540	0.089218	3.718162	0.849077
16	(POSTAGE, ROUND SNACK BOXES SET OF 4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.225383	0.157549	0.124726	0.563398	3.512540	0.089218	1.886357	0.923431
14	(ROUND SNACK BOXES SET OF 4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.245077	0.157549	0.131291	0.535714	3.400298	0.092679	1.814509	0.935072
20	(ROUND SNACK BOXES SET OF 4 WOODLAND)	(POSTAGE, ROUND SNACK BOXES SET OF 4 FRUITS)	0.245077	0.150985	0.124726	0.508929	3.370730	0.087724	1.728904	0.931655
10	(POSTAGE)	(ROUND SNACK BOXES SET OF 4 WOODLAND)	0.818381	0.245077	0.225383	0.275401	1.123735	0.024817	1.041850	0.606270
8	(POSTAGE)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.818381	0.157549	0.150985	0.184492	1.171012	0.022049	1.033038	0.004086
19	(POSTAGE)	(ROUND SNACK BOXES SET OF 4 WOODLAND, ROUND SNA...	0.818381	0.131291	0.124726	0.152406	1.160829	0.017280	1.024912	0.762841
6	(POSTAGE)	(REGENCY CAKESTAND 3 TIER)	0.818381	0.137856	0.120350	0.147059	1.066760	0.007532	1.010790	0.344578
4	(POSTAGE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.818381	0.137856	0.118162	0.144385	1.047364	0.005344	1.007631	0.248996
13	(POSTAGE)	(WOODLAND CHARLOTTE BAG)	0.818381	0.126915	0.115974	0.141711	1.116587	0.012109	1.017240	0.574903
2	(POSTAGE)	(PLASTERS IN TIN SPACEBOY)	0.818381	0.107221	0.100656	0.122995	1.147113	0.012909	1.017986	0.706129
9	(POSTAGE)	(PLASTERS IN TIN CIRCUS PARADE)	0.818381	0.115974	0.100656	0.122995	1.060539	0.005746	1.008096	0.314301

- Datensatz mit Transaktionen
- Schranken für Minimum Support und Minimum Confidence
- Gesucht: belastbare Regeln
FALLS {X₁, ..., X_n} → {Y₁, ..., Y_n}

- Finde alle Teilmengen mit höherer relativer Häufigkeit als minimum Support

- Nimm Elemente aus vorigem Schritt, bilde Regeln, berechne deren Konfidenz.
- Lösche Regeln falls Konfidenz kleiner oder gleich der Minimum Confidence ist
- Sortiere Regeln nach absteigendem Lift-Wert



Zusammenfassung

Vorteile:

- Einfach anwendbar.
- Kein zeitaufwendiges Labeln notwendig.
- Entdecken unbekannter Zusammenhänge.

Nachteile:

- Bewertung Ergebnisse durch Experten notwendig.
- Kein eigenständiges Ableiten von Handlungen.



Backup



Detail-Folien Clustering



Clustering

Aufgabe: Anzeigen Nullen im Datensatz

```
[ ] loan_df.isna().sum()
```

Loan_ID	0
Gender	13
Married	3
Dependents	15
Education	0
Self_Employed	32
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	22
Loan_Amount_Term	14
Credit_History	50
Property_Area	0
Loan_Status	0
dtype:	int64

Wir ersetzen für unser Beispiel die fehlenden Werte bei Loan_AMOUNT durch den Median der ganzen Spalte.
(ANM: Man sollte normalerweise ein solches Ersetzen im Detail prüfen, aber für Veranschaulichung von Clustering ist das ok).



Clustering

Aufgabe: Ersetzen NaNs in Spalte Loan_Amount durch Median der Spalte.

```
[ ] # für den Algorithmus dürfen keine leeren Zeilenwerte sein. deshalb aktualisiere Spalte LoanAmount: fülle alle leeren Einträge mit dem Median der Spalte
X["LoanAmount"] = X["LoanAmount"].fillna(X['LoanAmount'].median())
X.isna().sum()
```




Clustering: KMeans

```
[ ] from sklearn.cluster import KMeans

# Anzahl Cluster. Wir fangen mit 2 an.
kmeans = KMeans(n_clusters=2)
# Eingabedaten "fitten", d.h. einpassen in 2 Gruppen. Hier geschieht die Arbeit des Algorithmus.
kmeans = kmeans.fit(X_non_outlier)

# Einteilen der Daten in 2 Cluster
clusters = kmeans.predict(X_non_outlier)

# Auswerten der Zentrumsunkte je Cluster
centroids = kmeans.cluster_centers_
print(centroids) # From sci-kit learn
```

Vorgehen:

1. kmeans mit 2 Clustern starten
2. Kmeans trainieren
3. Vorhersage Clusterzahl
4. Ausdrucken Zentrum Cluster

Befehle:

fit() -> Modell trainieren
predict() -> Vorhersage treffen
kmeans.cluster_centers_ -> Mittelpunkte der Cluster



Clustering: KMeans

EXKURS PLOTTEN

```
[ ] # mach ein scatter plot für die Werte
plt.scatter(X_non_outlier["X_ApplicantIncome_nonoutlier"],
            X_non_outlier["X_LoanIncome_nonoutlier"],
            c=clusters, # zeichne die durch kmeans eingeteilten Gruppen in verschiedenen Farben
            )
plt.xlabel("Income of Applicant");
plt.ylabel("Applied Loan");

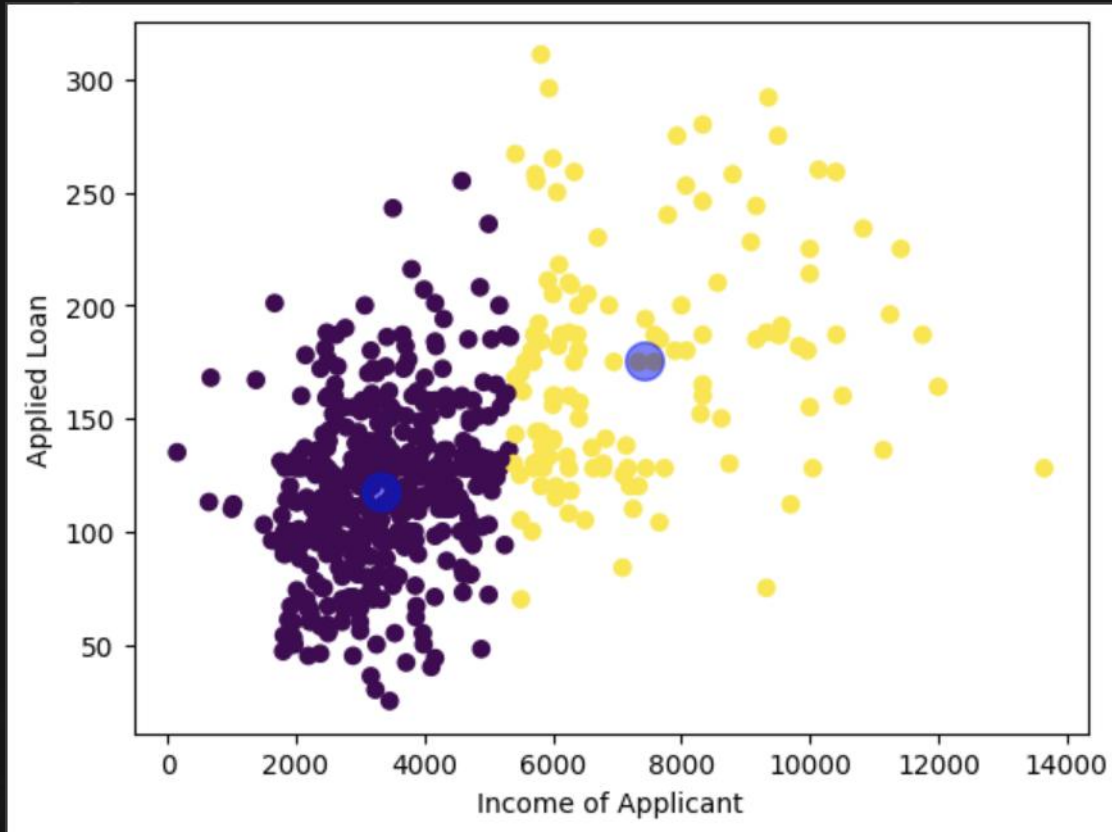
# wir plotten jetzt die Centroids aus dem vorigen Block
plt.scatter(centroids[:, 0],
            centroids[:, 1],
            c='blue',
            s=200,
            alpha=0.5)
```

HAUSAUSGABE: PLOTEINSTELLUNGEN ANSCHAUEN UND VARIANTEN PLOTTEN



Clustering: KMeans

EXKURS PLOTTEN



- Wir sehen, daß die Menge in 2 Klassen eingeteilt worden ist mit den blauen Punkten als Mittelpunkt oder Nabe der jeweiligen Gruppe.
- Mit Hilfe der Nabe kann auch veranschaulicht werden, was KMeans iterativ macht. KMeans definiert zufällig eine Anzahl von k Mittelpunkten und wiederholt das, bis es keine Aktualisierungen mehr gibt.
- Dadurch werden die gesamten Punkte in k einzelne Gruppen aufgeteilt. Dabei kommt ein Punkt in genau die Gruppe, zu deren Mittelpunkt er den geringsten Abstand hat.
- Nachdem alle Punkte in eine der k Gruppen eingeteilt wurden, ist der aktuelle Mittelpunkt nicht zwingend mehr der Mittelpunkt der Gruppe.
- Deshalb wird ein neuer Mittelpunkt je Gruppe berechnet.



Clustering: KMeans

Identifikation der „idealen“ Anzahl von Clustern

```
[ ] # Schritt 1: Trainieren
sse = [] # in dieser Liste speichern wir die einzelnen Werte je K-Means Versuch

# in dieser Schleife rufen wir kMeans für 1 bis 11 Cluster auf
for k in range(1, 11):
    kmeans_loop = KMeans(n_clusters=k);
    kmeans_loop.fit(X_non_outlier);
    sse.append(kmeans_loop.inertia_); #schreibe den Fehler in die Liste
```

sse = Die durchschnittlichen
Standardabweichungen pro Cluster



Clustering: KMeans

Identifikation der „idealen“ Anzahl von Clustern

AUFGABE : FITTEN MIT DER IDEALEN CLUSTERZAHL

```
[ ] # deshalb nehmen wir die Zahl 3 als Parameter
    kmeans_ellbow = KMeans(n_clusters=3)
    kmeans_ellbow.fit(X_non_outlier)
    cluster = kmeans_ellbow.predict(X_non_outlier)
```


Clustering: KMeans



Problem 4: wie finde ich genaue Grösse der Cluster?



Clustering: KMeans

Ermitteln der genauen Größe der Cluster

```
[ ] frame = pd.DataFrame(X_non_outlier)
    frame['cluster'] = cluster
    frame['cluster'].value_counts()
```

```
0    326
2    172
1     54
Name: cluster, dtype: int64
```

Vorgehen:

1. dataframe (zum Zählen der Cluster) erstellen
2. Cluster zuweisen
3. Clusterwerte zählen

Befehle:

pd.dataframe() -> auf Datensatz zugreifen (in pandas)



Assoziationsregeln



Assoziationsregeln

Case Study: Online Retail

Hausaufgaben :

- wir löschen alle nans aus Spalten invoice_no
- Aus dem Gespräch mit unserer Vertriebsmannschaft haben wir gelernt, dass alle invoice-nummern, die mit „c“ anfangen, abgebrochene Bestellungen sind. Diese müssen wir auch noch löschen (hint: alle löschen, die als datentyp „str“ haben).



Assoziationsregeln

Case Study: Online Retail

```
[ ] # Umwandeln der Spalte in Zeichenkette
modified_retail_df.dropna(axis = 0, subset = ['InvoiceNo'], inplace = True)
modified_retail_df["InvoiceNo"] = modified_retail_df["InvoiceNo"].astype('str')

# damit wir nach Spalten mit C für cancelled filtern können
modified_retail_df = modified_retail_df[~modified_retail_df["InvoiceNo"].str.contains("C")]
```

Vorgehen:

1. nans aus Spalte „invoiceno“ löschen
2. spalte in Zeichen umwandeln (für Filterung)
3. Filtern nach „c“

Befehle:

str.contains() -> strings filtern nach enthaltenen Werten



Assoziationsregeln

Case Study: Online Retail



```
modified_retail_df['InvoiceNo'].str.contains('C').sum()
```

9288

Datensatz hat 9288 abgebrochene Bestellungen, die gelöscht werden müssen.



Assoziationsregeln

Case Study: Online Retail

Hausaufgabe:

- Löschen der Zeilen mit abgebrochenen Bestellungen und aller weiteren Zeilen, mit denen wir nichts anfangen können (hint: bringen Bestelldaten ohne Artikelbeschreibung etwas ;)?



Assoziationsregeln

Kaufverhalten in verschiedenen Ländern

```
[ ] modified_retail_df['Country'].value_counts()

United Kingdom      354345
Germany              9042
France              8342
EIRE                 7238
Spain                2485
Netherlands          2363
Belgium              2031
Switzerland          1842
Portugal             1462
Australia            1185
Norway               1072
Italy                 758
Channel Islands      748
Finland              685
Cyprus                614
Sweden               451
Austria              398
Denmark              380
Poland               330
Japan                321
Israel               248
Unspecified          244
Singapore            222
Iceland              182
USA                  179
Canada               151
Greece               145
Malta                112
United Arab Emirates  68
European Community   60
RSA                  58
Lebanon              45
Lithuania            35
Brazil               32
Czech Republic       25
Bahrain              17
Saudi Arabia          9
Name: Country, dtype: int64
```



Assoziationsregeln

Case Study: Online Retail

Aufgaben :

- Erstellen notebook: retail
- Laden aller für Data Science notwendigen Bibliotheken und zusätzlich folgende Bibliotheken für die Algorithmen:

```
[ ] from mlxtend.frequent_patterns import fpgrowth  
    from mlxtend.frequent_patterns import apriori, association_rules
```

- Laden sie den Datensatz loan_data (excel) von der url
- Sehen sie sich den Datensatz an

```
#Load the file into pandas  
retail_df = pd.read_excel("http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx")
```



Assoziationsregeln

Case Study: Online Retail

```
[ ] modified_retail_df
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

397924 rows x 8 columns

Welche Spalte sollte man als erstes bearbeiten?



Assoziationsregeln

- Wir sehen daß die ersten 4 Zeilen die gleiche Invoice-Nummer haben, also eine gleiche Bestellung sind. Für eine Basketanalyse müssen wir jetzt noch eine Art virtuellen Einkaufskorb erstellen. Dazu verwenden wir den groupby-Befehl.
- Dieser sagt, daß wir alle Zeilen, die die gleiche Invoice-Nummer haben, gruppieren in eine neue Zeile in einer Liste, gleiche Elemente aufaddieren.



Assoziationsregeln

```
basket_GER = (modified_retail_df[modified_retail_df['Country'] == "Germany"]
              .groupby(['InvoiceNo', 'Description'])['Quantity']
              .sum()
              .unstack()
              .reset_index()
              .fillna(0)
              .set_index('InvoiceNo'))
```

Vorgehen:

1. nur numerische Werte identifizieren
2. Mittelwert berechnen
3. Objekt durch Mittelwert (bzw. median ersetzen)

Befehle:

groupby → Gruppieren anhand von Parametern
sum() → Aufaddieren von Werten, bspw. einer Spalte
reset_index() → neuen Index erstellen
set_index() → Zeilenindex auf Spalte vergeben
mean()/ median() → Mittelwert berechnen
replace() → Werte tauschen/ ersetzen



Assoziationsregeln

Für den Assoziationsregelalgorithmus müssen wir noch eine Sache machen:
der Algorithmus verlangt, daß wir statt der Anzahl an gekauften Items einen False-Wert haben, falls nichts gekauft wurde und ansonsten True.

```
[ ] def encode_values(x):  
    if(x <= 0):  
        return 0  
    if(x >= 1):  
        return 1  
  
# Encoding the datasets  
basket_GER = basket_GER.applymap(encode_values)
```

Befehle:

applymap -> definierte Funktion ausführen



Assoziationsregeln

```
▶ # Model erstellen
frq_items_GER = apriori(basket_GER,
                        min_support = 0.1, # mindestens 10% Support muss eine Regel haben
                        use_colnames = True)

# speichern der Regeln
rules_GER = association_rules(frq_items_GER,
                              metric = "lift",
                              min_threshold = 1)

# sortieren der Regeln
rules_GER = rules_GER.sort_values(['confidence', 'lift'],
                                   ascending = [False, False])
```

Antedecent = Vorbedingung

Consequent = Nachbedingung, bspw. Bier gekauft, wie hoch ist die Wahrscheinlichkeit daß zudem Windeln gekauft wurden



Assoziationsregeln

```
[ ] rules_GER
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` arg and should_run_async(code)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
8	(ROUND SNACK BOXES SET OF 4 FRUITS)	(POSTAGE)	0.157549	0.818381	0.150985	0.958333	1.171012	0.022049	4.358862	0.173348
16	(ROUND SNACK BOXES SET OF 4 FRUITS, ROUND SNAC...	(POSTAGE)	0.131291	0.818381	0.124726	0.950000	1.160829	0.017280	3.632385	0.159486
2	(PLASTERS IN TIN SPACEBOY)	(POSTAGE)	0.107221	0.818381	0.100656	0.938776	1.147113	0.012909	2.966448	0.143649

Bedeutung spalten in Zeile1:

- Support: Round snack of boxes wurde in 15% aller Fälle gekauft
- Confidence: in 95% der Fälle wenn snack of boxes gekauft wurde, wurde auch Postage gekauft
- Lift: Kauf des Antedecent einer Erhöhung der Wahrscheinlichkeit des Kaufs des Consequents um Faktor 1,17 bedeutet.

18	(ROUND SNACK BOXES SET OF4 WOODLAND, POSTAGE)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.225383	0.157549	0.124726	0.553398	3.512540	0.089218	1.886357	0.923431
15	(ROUND SNACK BOXES SET OF4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.245077	0.157549	0.131291	0.535714	3.400298	0.092679	1.814509	0.935072
20	(ROUND SNACK BOXES SET OF4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS, POSTAGE)	0.245077	0.150985	0.124726	0.508929	3.370730	0.087724	1.728904	0.931655
11	(POSTAGE)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.818381	0.245077	0.225383	0.275401	1.123735	0.024817	1.041850	0.606270
9	(POSTAGE)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.818381	0.157549	0.150985	0.184492	1.171012	0.022049	1.033038	0.804086
21	(POSTAGE)	(ROUND SNACK BOXES SET OF 4 FRUITS, ROUND SNAC...	0.818381	0.131291	0.124726	0.152406	1.160829	0.017280	1.024912	0.762841
7	(POSTAGE)	(REGENCY CAKESTAND 3 TIER)	0.818381	0.137856	0.120350	0.147059	1.066760	0.007532	1.010790	0.344578
4	(POSTAGE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.818381	0.137856	0.118162	0.144385	1.047364	0.005344	1.007631	0.248996
13	(POSTAGE)	(WOODLAND CHARLOTTE BAG)	0.818381	0.126915	0.115974	0.141711	1.116587	0.012109	1.017240	0.574903
3	(POSTAGE)	(PLASTERS IN TIN SPACEBOY)	0.818381	0.107221	0.100656	0.122995	1.147113	0.012909	1.017986	0.706129
1	(POSTAGE)	(PLASTERS IN TIN CIRCUS PARADE)	0.818381	0.115974	0.100656	0.122995	1.060539	0.005746	1.008006	0.314301