DATA SCIENCE

-

PROJEKT

MARKETING - CAMPAIGN

VORGEHENSWEISE:

- ✔ Datensatz beschaffen → kostenloser Kaggle Datensatz
- ✔ Deskriptive Analyse des Datensatzes
- ✔ Daten bereinigen
- ✔ Korrelationen einsehen und visualisieren
- ✔ Visualisierungen in Tableau
- ✔ Visualisierung des Datensatzes – Plots
- ✔ PCA
- ✔ Supervised Learning
- ✔ Unsupervised Learning

DATENSATZ:

VORÜBERLEGUNGEN:

- Auswahl des Datensatzes: „Marketing “ - Akzeptanz vom Kunden in Bezug auf verschiedene Kampagnen

- Marketing ist ein wichtiger Bestandteil der Strategie eines jeden Unternehmens.
  Gezieltes Marketing fördert die Bekanntheit, den Ruf oder den Absatz eine Marke, eines Produkt oder einer Dienstleistung bei bestimmten Personen

- Die Analyse der Zielgruppen oder das Preismanagement sind ebenso wichtig wie die Planung und Gestaltung von Werbemaßnahmen, wie in diesem Datensatz die unterschiedlichen Kampagnen

ZIEL:

➔ Ziel dieses Projektes ist es, anhand verschiedener Algorithmen aus dem Machine-Learning-Bereich vorherzusagen, ob ein Kunde eine weitere Kampagne annimmt oder nicht

Der Datensatz enthält 29 features, hat 2240 samples und liefert uns mit dem Label „Response“ Informationen, ob ein Kunde eine weitere Kampagne angenommen hat oder nicht:

FEATURES:

AcceptedCmp1	
AcceptedCmp2	
AcceptedCmp3	
AcceptedCmp4	Akzeptanz des Angebots: 1 – ansonsten: 0
AcceptedCmp5	
Response(Label – letzte Kampagne)	
Complain	Beschwerde des Kunden in den letzten 2 Jahre: 1 - ansonsten 0
DtCustomer	Datum der Anmeldung des Kunden beim Unternehmen
Eduction	Bildungsgrad
Marital	Familienstand des Kunden
Kidhome	Anzahl der kleinen Kinder im Haushalt des Kunden
Teenhome	Anzahl der Teenager im Haushalt des Kunden
Income	jährliches Haushaltseinkommen des Kunden
MntFishProducts	Betrag, der in den letzten 2 Jahren für Fischprodukte ausgegeben wurde
MntMeatProducts	Betrag, der in den letzten 2 Jahren für Fleischprodukte ausgegeben wurde

MntFruits	Betrag Ausgaben für Früchte in den letzten 2 Jahren
MntSweetProducts	Betrag, der in den letzten 2 Jahren für Süßigkeiten ausgegeben wurde
MntWines	Betrag, der in den letzten 2 Jahren für Wein ausgegeben wurde
MntGoldProds	Betrag, der in den letzten 2 Jahren für Goldschmuck ausgegeben wurde
NumDealsPurchases	Anzahl der Käufe mit Rabatt
NumCatalogPurchases	Anzahl der über den Katalog getätigten Käufe
NumStorePurchases	Anzahl der direkt in Geschäften getätigten Einkäufe
NumWebPurchases	Anzahl der über die Website des Unternehmens getätigten Einkäufe
NumWebVisitsMonth	Anzahl der Besuche auf der Website des Unternehmens im letzten Monat
Recency	Anzahl der Tage seit dem letzten Kauf

DESKRIPTIVE ANALYSE DES DATENSATZES:

- ✗ Ausschnitt aus dem Datensatz:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	88	546	172
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	1	6	2
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	49	127	111
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	4	20	10
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	43	118	46

MntSweetProducts	MntGoldProds	NumDealsPurchases	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumWebVisitsMonth
88	88	3	8	10	4	7
1	6	2	1	1	2	5
21	42	1	8	2	10	4
3	5	2	2	0	4	6
27	15	5	5	3	6	5

AcceptedCmp3	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Z_CostContact	Z_Revenue	Response
0	0	0	0	0	0	3	11	1
0	0	0	0	0	0	3	11	0
0	0	0	0	0	0	3	11	0
0	0	0	0	0	0	3	11	0
0	0	0	0	0	0	3	11	0

DATEN BEREINIGEN:

- ✗ Der Datensatz hat Nullwerte in der Spalte „Income“ und einige Ausreißer von sehr hohem Einkommen, die mit dem Durchschnittswert aufgefüllt bzw. ersetzt werden:

```
FÜLLEN NULLWERTE MIT MEAN BEI INCOME

df["Income"].fillna(df["Income"].mean(),inplace=True)
```

```
AUSREISSER BEI INCOME (7 ZEILEN) MIT FUNKTION NEU BEFÜLLEN

def income_ausreisser(x):
    if x > 150000:
        return 52000
    else:
        return x

df['Income'] = df['Income'].apply(income_ausreisser)
```

✗ eine neue Spalte „Age“ (löschen der Spalte „Year_Birth“):

```
SPALTE AGE HINZUFÜGEN UND SPALTE YEAR_BIRTH LÖSCHEN

df["Age"] = 2023 - df["Year_Birth"]

df.drop("Year_Birth", axis = 1, inplace = True)
```

✗ eine neue Spalte „AcceptedCmp_Total“(alle akzeptieren Kampagnen pro Kunde aufsummiert):

```
NEUE SPALTE TOTAL_CAMPAIGNS_ACCEPTED

df['AcceptedCmp_Total'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4',
'AcceptedCmp5', 'Response']].sum(axis=1)
```

✗ Löschen nicht benötigter Spalten ID, Dt_Customer, Z_CostContact und Z_Revenue:

```
df.drop(["ID","Dt_Customer", "Z_CostContact", "Z_Revenue"],axis=1, inplace = True)
```

✗ Ausreißer bei der Spalte Age mit 3 Werten weit über 100 – ersetzen mit Mittelwert:

```
def ausreisser_age(x):
    if x > 100:
        return 54
    else:
        return x

df['Age'] = df['Age'].apply(ausreisser_age)
```

✗ somit verbleiben 26 features

✗ der Datensatz hat kategorische Werte, die in numerische umgewandelt werden:

KATEGORISCHE WERTE - OBJECTS

```
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
```

NULLWERTE:

```
1 df.isnull().sum()
```

```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                24
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

✗ Umwandeln der kategorischen features in numerische Werte:

```python
def mari(x):
    if x == "Absurd" or x == "YOLO" or x == "Alone":
        return 0
    if x == "Single":
        return 1
    if x == "Widow":
        return 2
    if x == "Divorced":
        return 3
    if x == "Together":
        return 4
    if x == "Married":
        return 5

df["Marital_Status"] = df["Marital_Status"].apply(mari)
```

```python
def educ(x):
    if x == "Basic":
        return 0
    if x == "Graduation":
        return 1
    if x == "2n Cycle":
        return 2
    if x == "Master":
        return 3
    if x == "PhD":
        return 4

df["Education"] = df["Education"].apply(educ)
```

✗ bereinigter Datensatz ohne Nullwerte und mit ausschließlich numerischen Daten:

0	Education	2240	non-null	int64
1	Marital_Status	2240	non-null	int64
2	Income	2240	non-null	float64
3	Kidhome	2240	non-null	int64
4	Teenhome	2240	non-null	int64
5	Recency	2240	non-null	int64
6	MntWines	2240	non-null	int64
7	MntFruits	2240	non-null	int64
8	MntMeatProducts	2240	non-null	int64
9	MntFishProducts	2240	non-null	int64
10	MntSweetProducts	2240	non-null	int64
11	MntGoldProds	2240	non-null	int64
12	NumDealsPurchases	2240	non-null	int64
13	NumWebPurchases	2240	non-null	int64
14	NumCatalogPurchases	2240	non-null	int64
15	NumStorePurchases	2240	non-null	int64
16	NumWebVisitsMonth	2240	non-null	int64
17	AcceptedCmp3	2240	non-null	int64
18	AcceptedCmp4	2240	non-null	int64
19	AcceptedCmp5	2240	non-null	int64
20	AcceptedCmp1	2240	non-null	int64
21	AcceptedCmp2	2240	non-null	int64
22	Complain	2240	non-null	int64
23	Response	2240	non-null	int64
24	Age	2240	non-null	int64
25	AcceptedCmp_Total	2240	non-null	int64

✗ Überblick über einzigartige Werte (nunique):

```
Education              5
Marital_Status         6
Income              1968
Kidhome                3
Teenhome               3
Recency              100
MntWines             776
MntFruits            158
MntMeatProducts      558
MntFishProducts      182
MntSweetProducts     177
MntGoldProds         213
NumDealsPurchases     15
NumWebPurchases       15
NumCatalogPurchases   14
NumStorePurchases     14
NumWebVisitsMonth     16
AcceptedCmp3           2
AcceptedCmp4           2
AcceptedCmp5           2
AcceptedCmp1           2
AcceptedCmp2           2
Complain               2
Response               2
Age                   56
AcceptedCmp_Total      6
dtype: int64
```

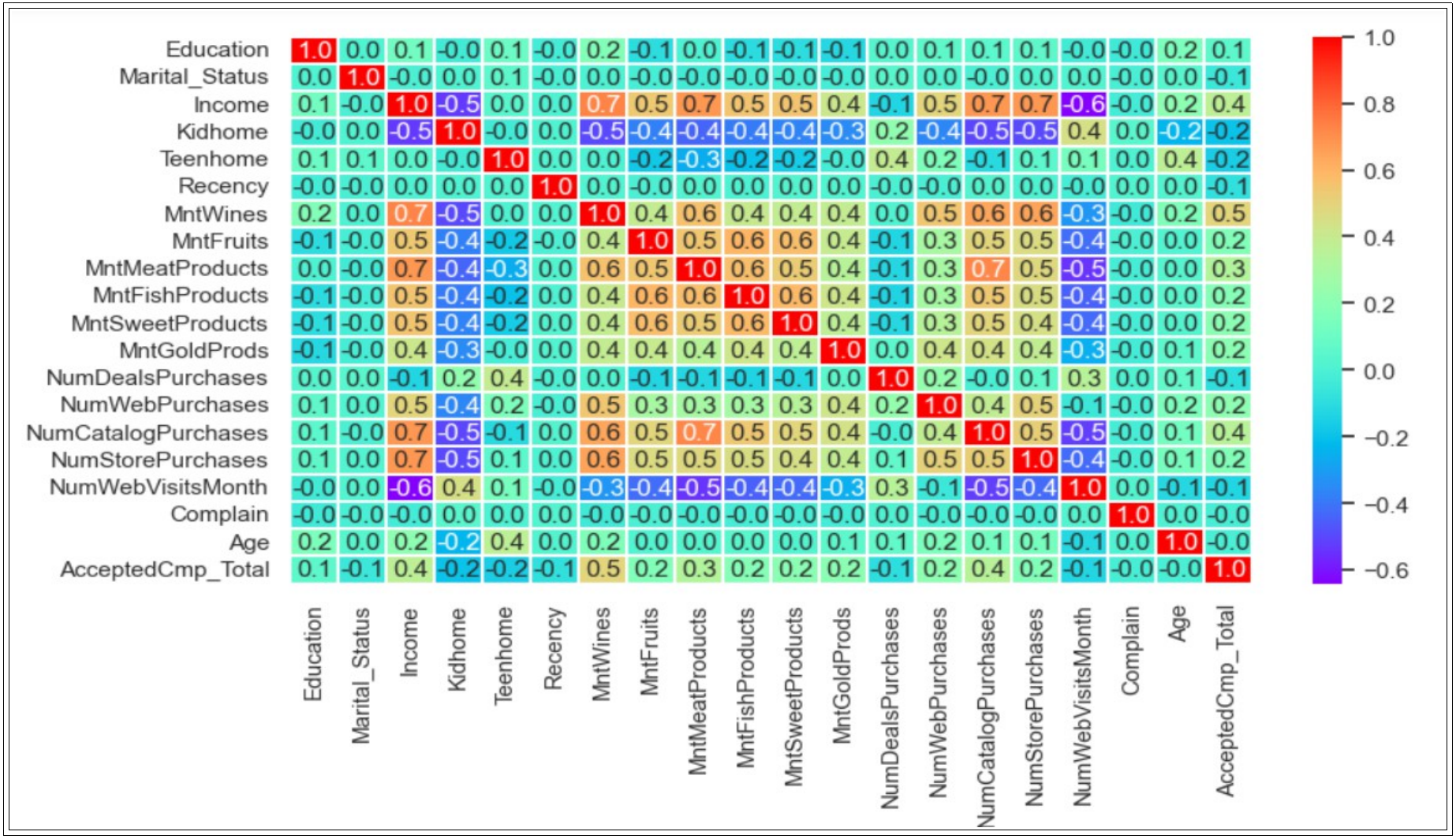
✗ deskriptive Analyse:

	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
count	2240.00	2240.00	2240.00	2240.00	2240.00	2240.00	2240.00	2240.00	2240.00	2240.00	2240.00
mean	2.05	3.56	51641.52	0.44	0.51	49.11	303.94	26.30	166.95	37.53	27.06
std	1.28	1.55	20564.93	0.54	0.54	28.96	336.60	39.77	225.72	54.63	41.28
min	0.00	0.00	1730.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	1.00	2.00	35538.75	0.00	0.00	24.00	23.75	1.00	16.00	3.00	1.00
50%	1.00	4.00	51741.50	0.00	0.00	49.00	173.50	8.00	67.00	12.00	8.00
75%	3.00	5.00	68098.25	1.00	1.00	74.00	504.25	33.00	232.00	50.00	33.00
max	4.00	5.00	113734.00	2.00	2.00	99.00	1493.00	199.00	1725.00	259.00	263.00

MntGoldProds	NumDealsPurchases	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumWebVisitsMonth
2240.00	2240.00	2240.00	2240.00	2240.00	2240.00
44.02	2.33	4.08	2.66	5.79	5.32
52.17	1.93	2.78	2.92	3.25	2.43
0.00	0.00	0.00	0.00	0.00	0.00
9.00	1.00	2.00	0.00	3.00	3.00
24.00	2.00	4.00	2.00	5.00	6.00
56.00	3.00	6.00	4.00	8.00	7.00
362.00	15.00	27.00	28.00	13.00	20.00

AcceptedCmp3	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Response	Age	AcceptedCmp_Total
2240.00	2240.00	2240.00	2240.00	2240.00	2.24e+03	2240.00	2240.00	2240.00
0.07	0.07	0.07	0.06	0.01	9.37e-03	0.15	54.10	0.45
0.26	0.26	0.26	0.25	0.11	9.64e-02	0.36	11.69	0.89
0.00	0.00	0.00	0.00	0.00	0.00e+00	0.00	27.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00e+00	0.00	46.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00e+00	0.00	53.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00e+00	0.00	64.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00e+00	1.00	83.00	5.00

KORRELATIONEN VISUALISIEREN:



✗ wichtige Korrelationen einzelner features untereinander ohne „Accepts“:

WICHTIGE KORRELATIONEN:

KORRELATION 0.7:

- Income - MntWines
- Income - MntMeatProducts
- Income - NumCatalogPurchases
- Income - NumStorePurchases
- MntMeatProducts - NumCatalogPurchases

KORRELATION 0.6:

- MntWines - MntMeatProducts
- MntWines - NumCatalogPurchases
- MntWines - NumCatalogStore
- MntFruits - MntFishProducts
- MntFruits - MntSweetProducts
- MntMeatProducts - MntFish
- MntFish - MntSweetProducts

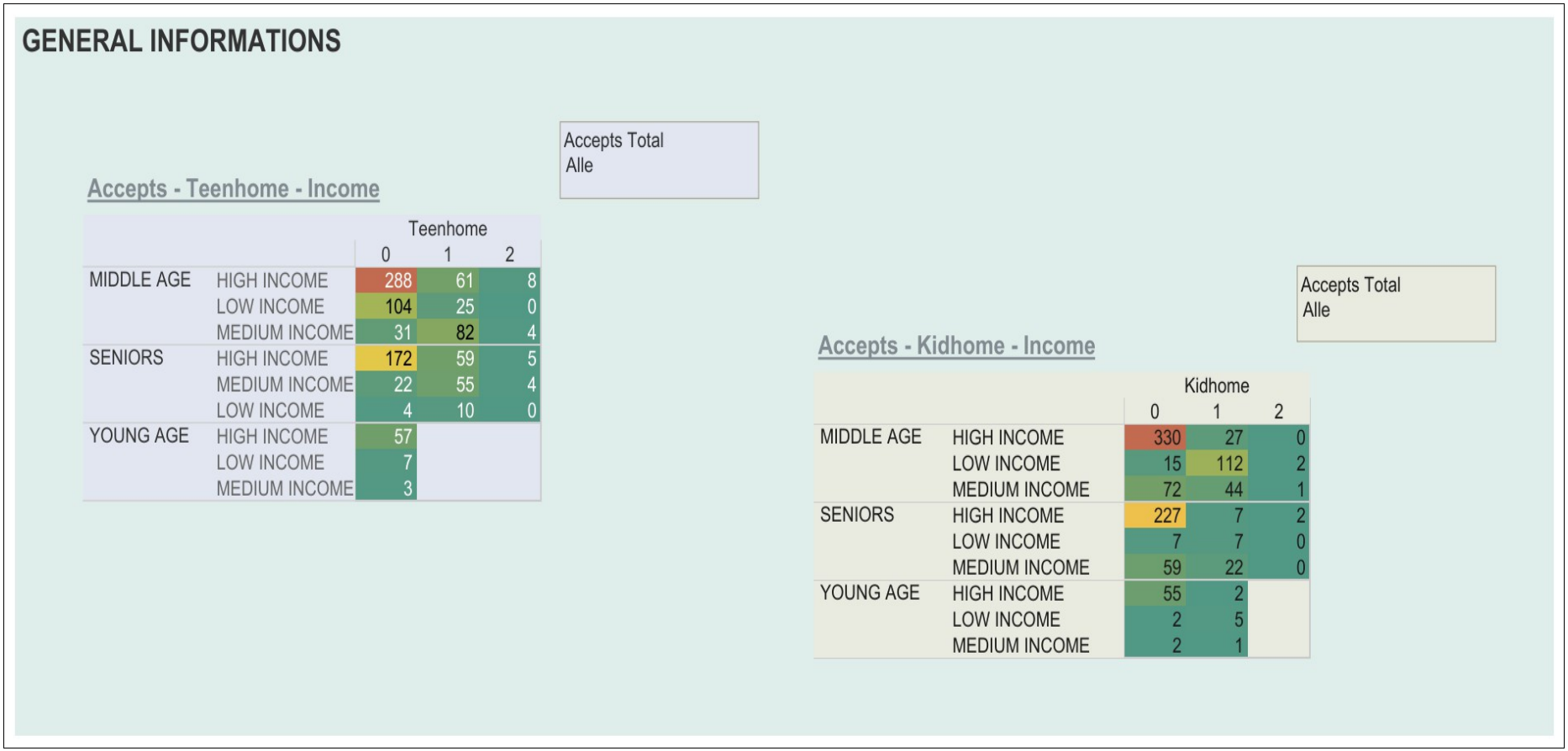
✗ Korrelation vom Label „Response“ mit allen features:

	index	Response
0	AcceptedCmp_Total	0.72
1	AcceptedCmp5	0.33
2	AcceptedCmp1	0.29
3	MntWines	0.25
4	AcceptedCmp3	0.25
5	MntMeatProducts	0.24
6	NumCatalogPurchases	0.22
7	AcceptedCmp4	0.18
8	Income	0.17
9	AcceptedCmp2	0.17
10	NumWebPurchases	0.15
11	MntGoldProds	0.14
12	MntFruits	0.13

	index	Response
13	MntSweetProducts	0.12
14	MntFishProducts	0.11
15	Education	0.09
16	NumStorePurchases	0.04
17	NumDealsPurchases	0.00
18	NumWebVisitsMonth	-0.00
19	Complain	-0.00
20	Age	-0.02
21	Kidhome	-0.08
22	Marital_Status	-0.14
23	Teenhome	-0.15
24	Recency	-0.20

VISUALISIERUNGEN IN TABLEAU:

LINK ZU TABLEAU: https://public.tableau.com/app/profile/manuela.holzner/viz/Marketing_campaign_16980756488600/General01

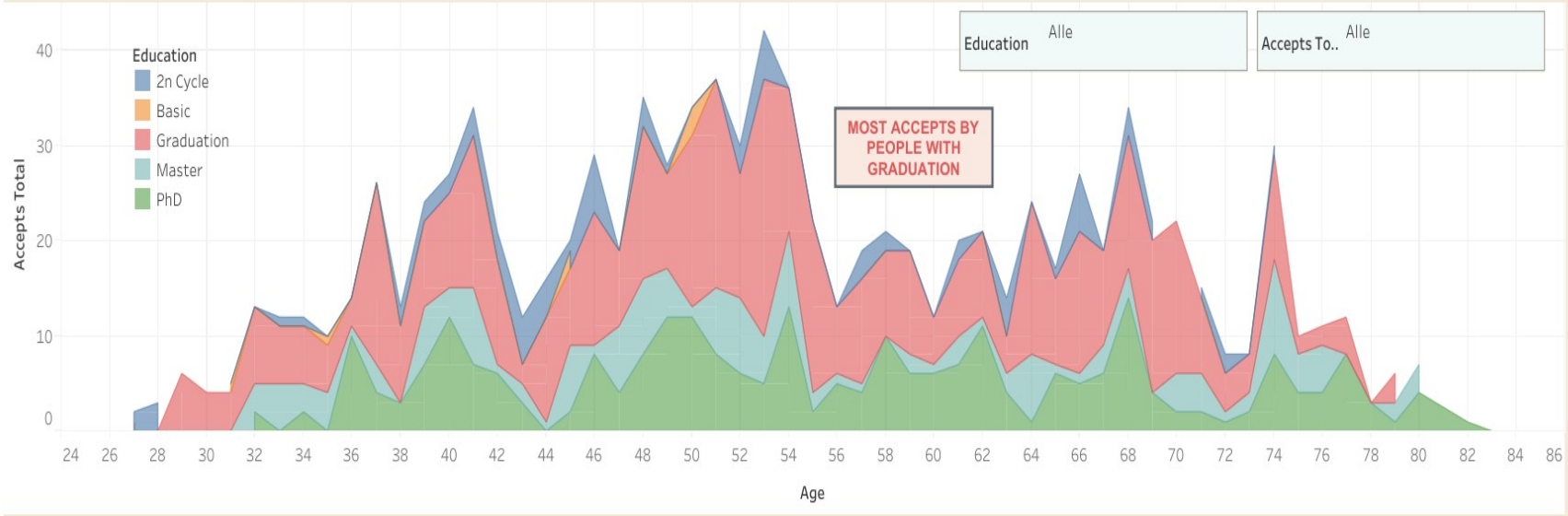


GENERAL INFORMATIONS

Age and Income

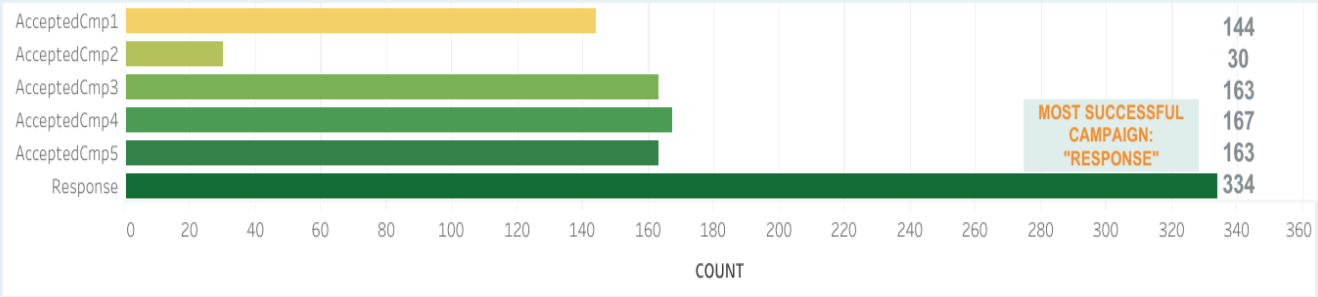
	HIGH INCOME	LOW INCOME	MEDIUM INCOME
MIDDLE AGE	424	546	454
SENIORS	311	130	255
YOUNG AGE	50	58	12

Accepts Total - Education



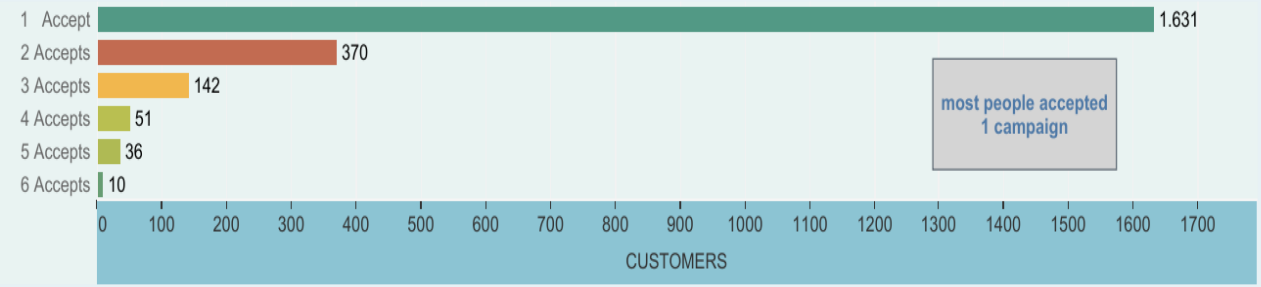
ACCEPTED CAMPAIGNS

Accepted Campaigns



Accepts Total
Alle

Numeric Accepted Campaigns by Customer



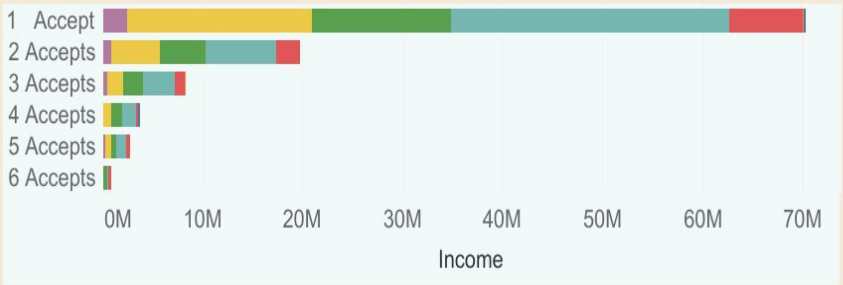
ACCEPTS MARITAL STATUS/INCOME

Income
1730 bis 113734

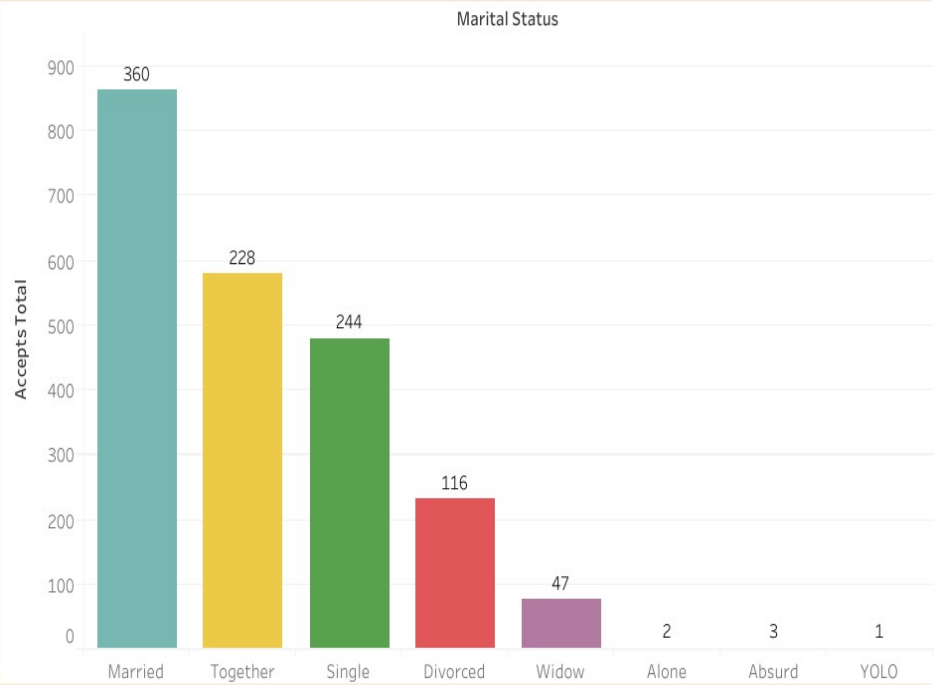
Marital Status
Alle

Marital Status
- Absurd
- Alone
- Divorced
- Married
- Single
- Together
- Widow
- YOLO

Accepts Total - Income/Marital Status

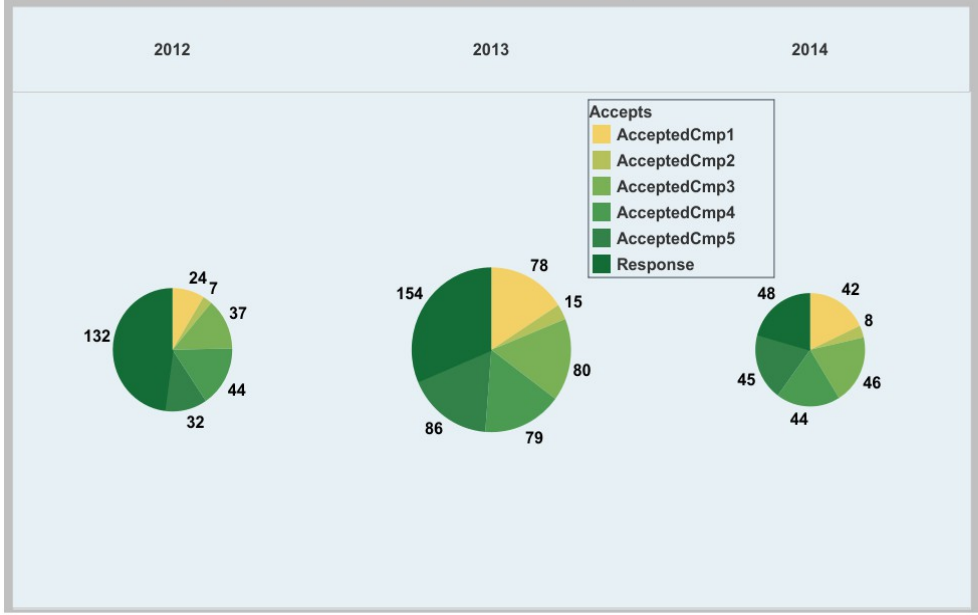


Accepts Total - Marital Status



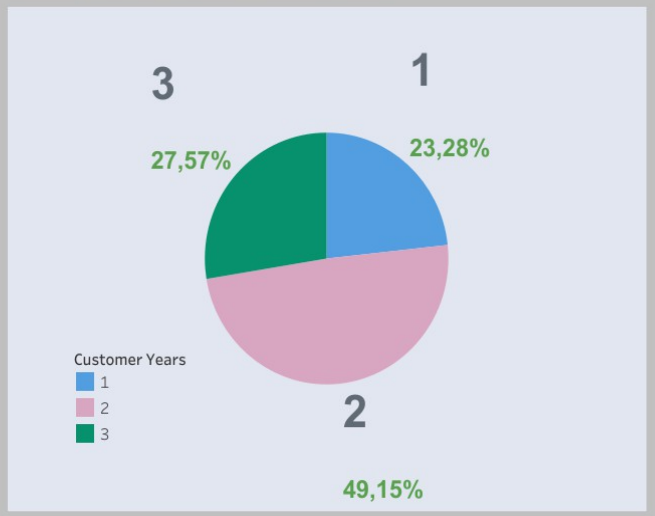
ACCEPTS CUSTOMER-YEARS

Duration Customer Relationship - Different Accepts



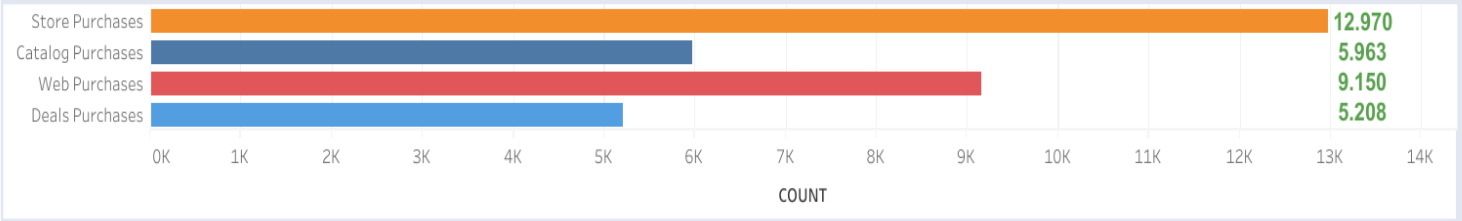
Accepts Total
Alle

Accepts Total - Customer Years



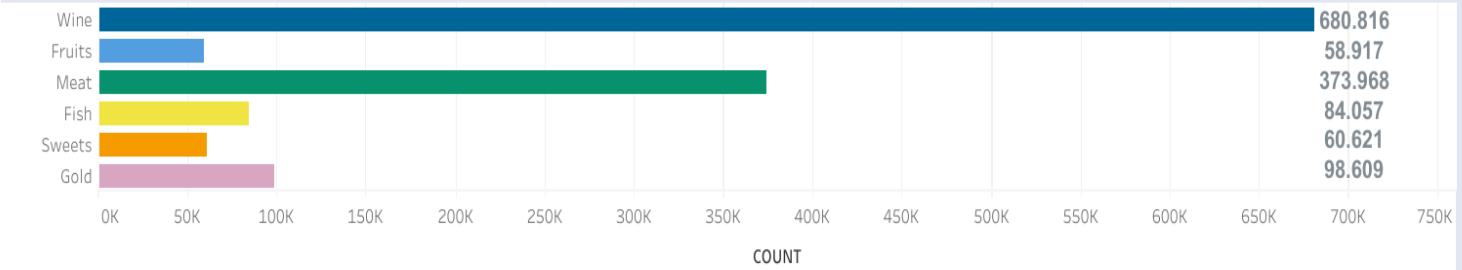
ACCEPTS PURCHASES AND SALED PRODUCTS

Purchases Store - Catalog - Web - Deals



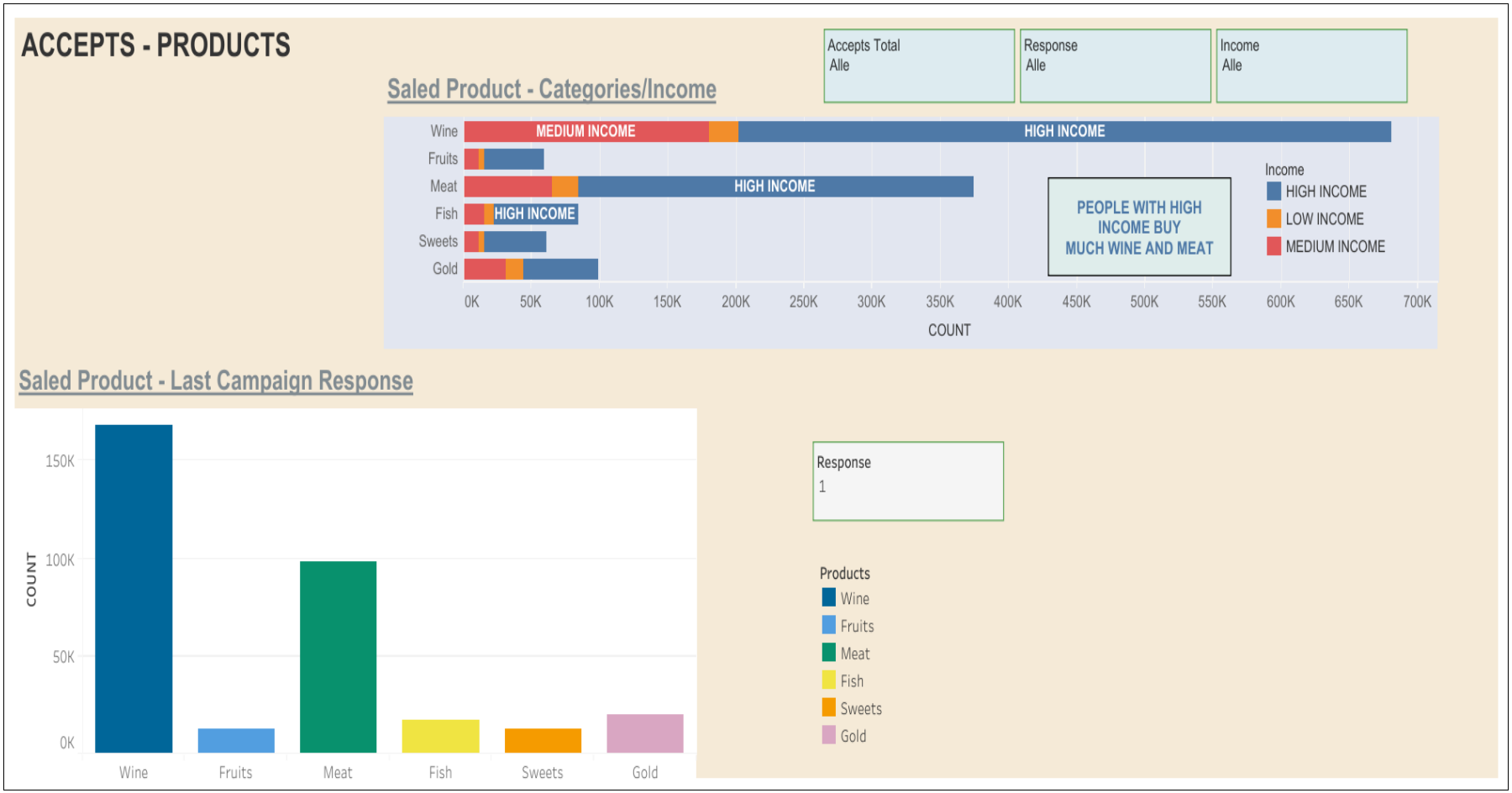
Purchases
- Store Purchases
- Catalog Purchases
- Web Purchases
- Deals Purchases

Saled Products - Categories



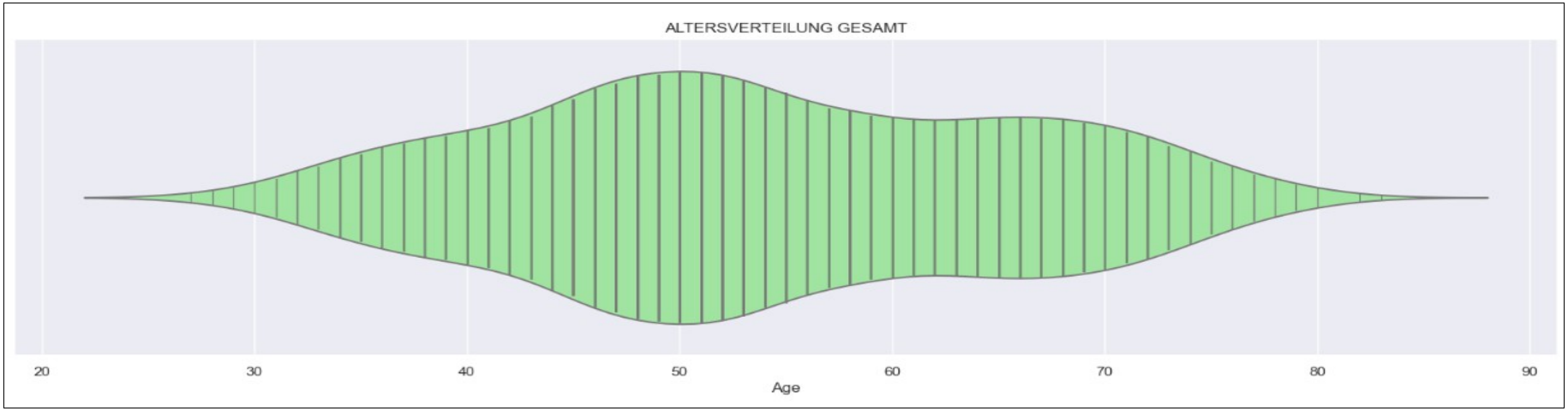
Accepts Total
Alle

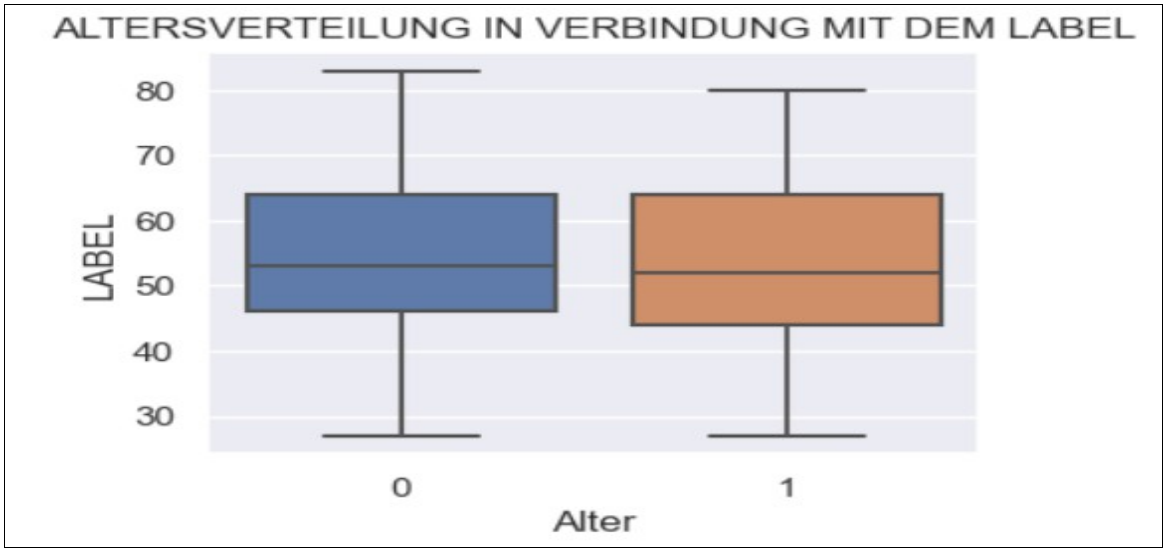
Response
Alle


ACCEPTS - PRODUCTS
Accepts Total
Alle
Response
Alle
Income
Alle
Saled Product - Categories/Income
Wine
Fruits
Meat
Fish
Sweets
Gold
MEDIUM INCOME
HIGH INCOME
HIGH INCOME
HIGH INCOME
PEOPLE WITH HIGH INCOME BUY MUCH WINE AND MEAT
Income
HIGH INCOME
LOW INCOME
MEDIUM INCOME
0K
50K
100K
150K
200K
250K
300K
350K
400K
450K
500K
550K
600K
650K
700K
COUNT
Saled Product - Last Campaign Response
150K
100K
50K
0K
COUNT
Wine
Fruits
Meat
Fish
Sweets
Gold
Response
1
Products
Wine
Fruits
Meat
Fish
Sweets
Gold

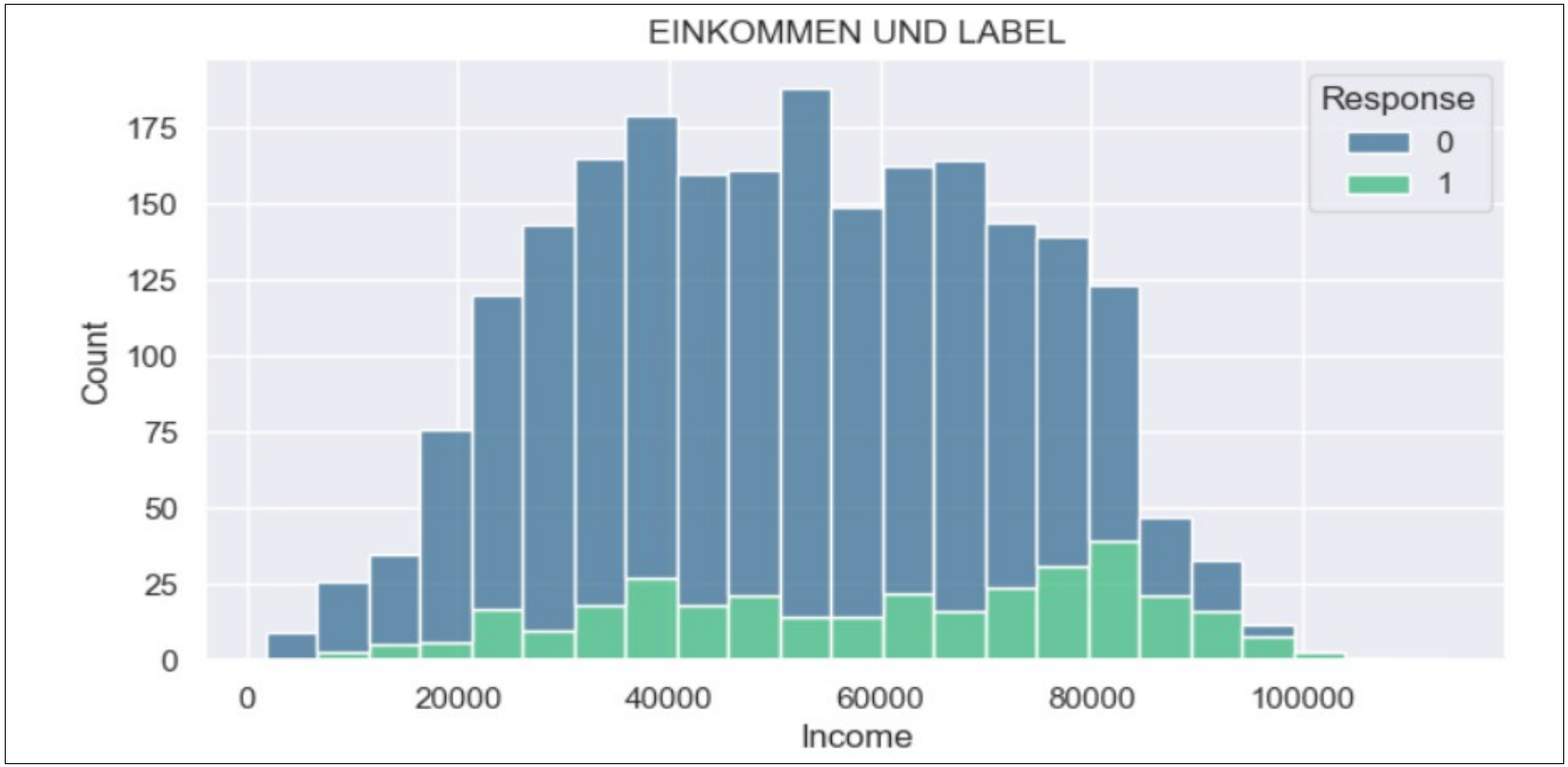
VISUALISIERUNG DES DATENSATZES JUPYTER NOTEBOOK:

- Altersverteilung:

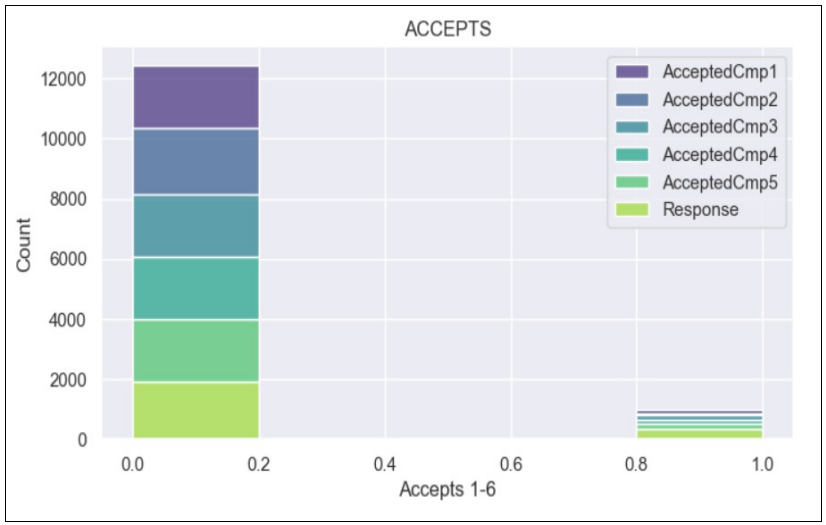




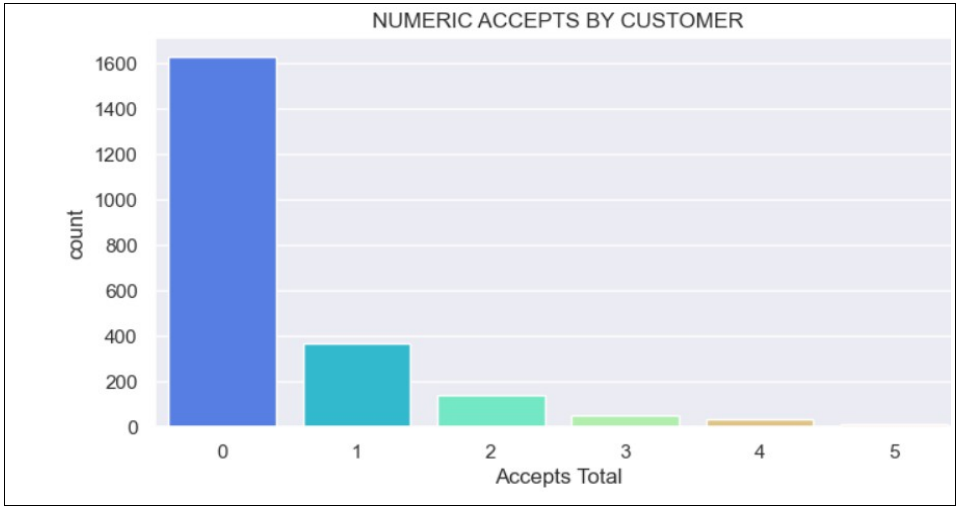
✗ Verhältnis von Einkommen und Label “Response“:



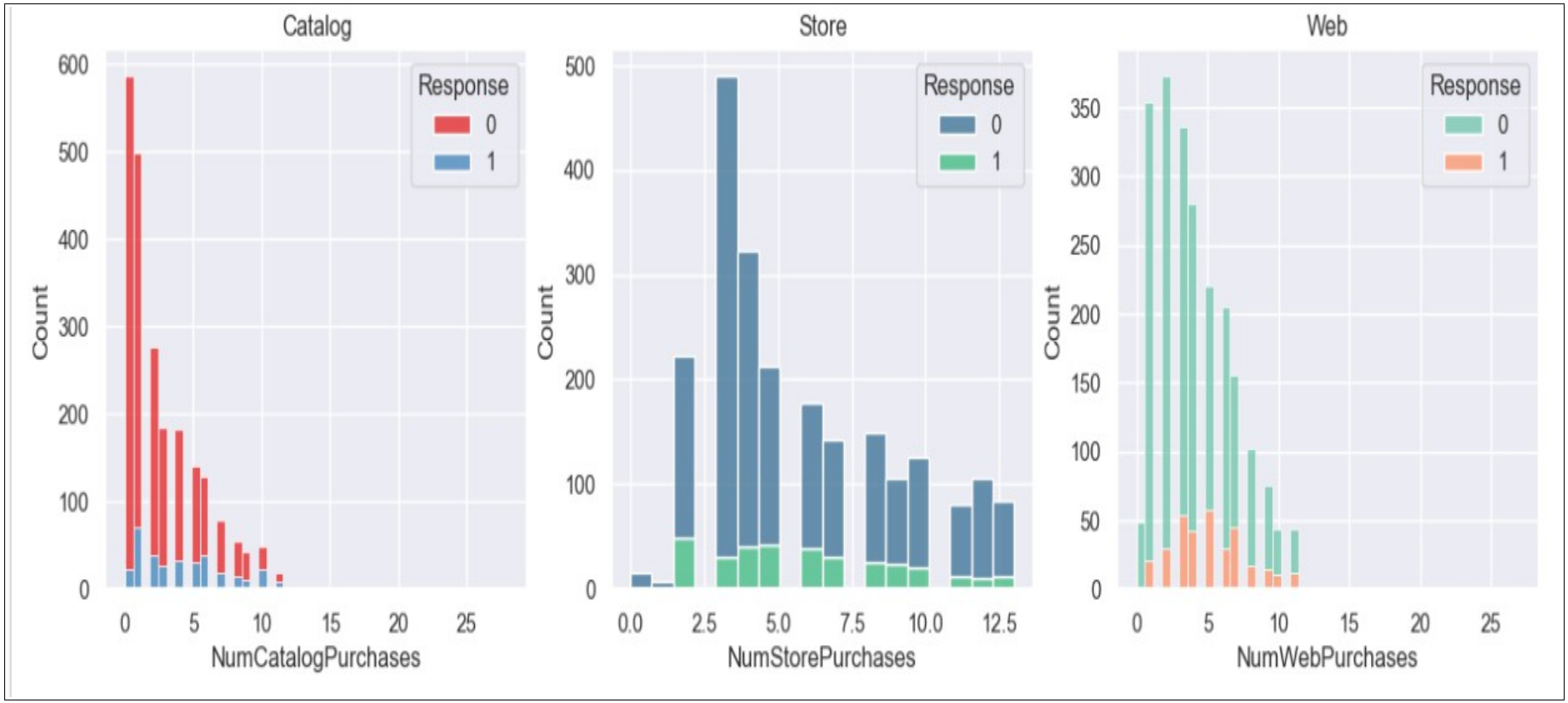
✗ einzelne Kampagnen 1-6, die angenommen wurden (1) oder nicht (0):



✗ Anzahl angenommene Kampagnen pro Kunde:

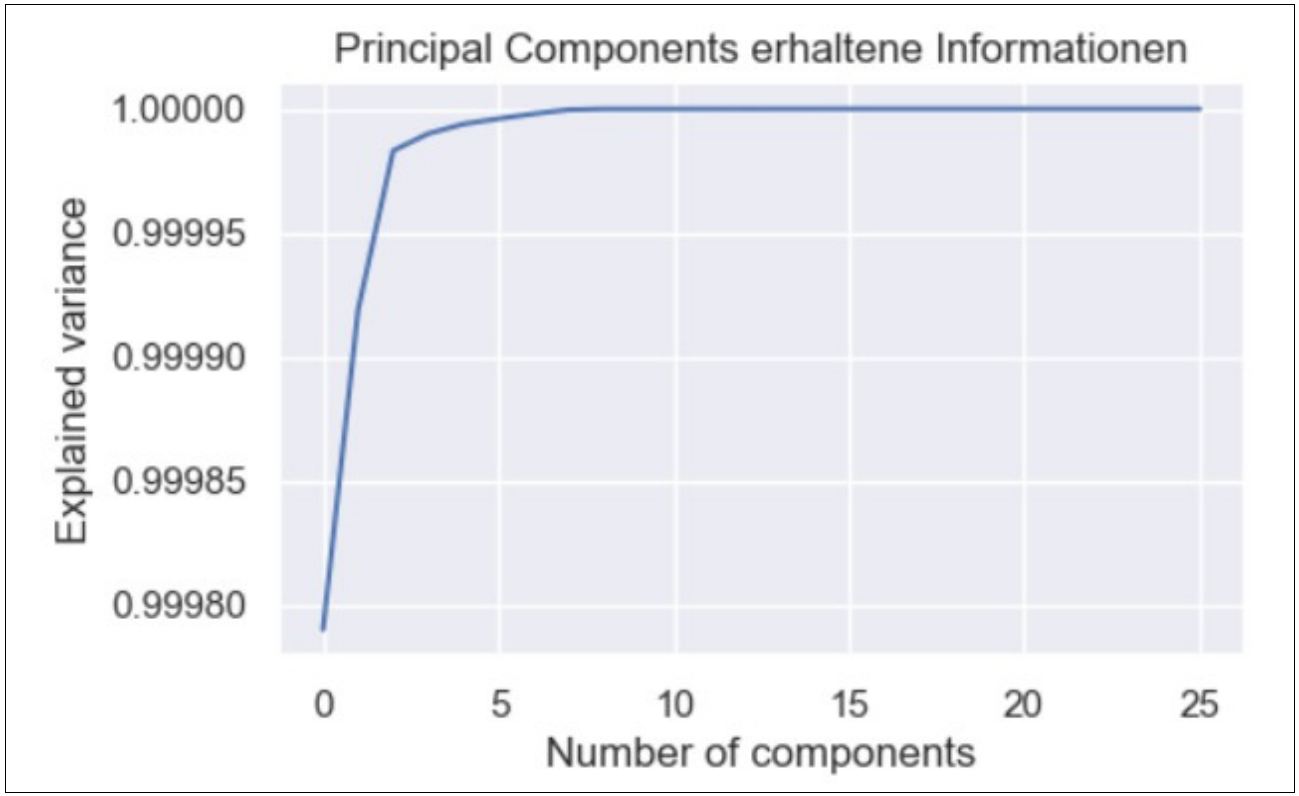


✗ Käufe im Geschäft, Katalog oder Web in Verbindung mit dem Label „Response“:



PCA:

- ✗ Überprüfung der enthaltenen Informationen in Abhängigkeit der Principal Components :



✗ Erste Komponente enthält nahezu 100 % der Informationen

```
Erste Komponente  enthält nahezu 100 % der Informationen

for i,value in enumerate(pca_check.explained_variance_ratio_):
    print(f"{i+1}. Principal Component erklärt {value*100:.4f}% der Varianz ")
```

```
1. Principal Component erklärt 99.9790% der Varianz
2. Principal Component erklärt 0.0129% der Varianz
3. Principal Component erklärt 0.0064% der Varianz
4. Principal Component erklärt 0.0007% der Varianz
5. Principal Component erklärt 0.0004% der Varianz
6. Principal Component erklärt 0.0002% der Varianz
7. Principal Component erklärt 0.0002% der Varianz
8. Principal Component erklärt 0.0002% der Varianz
9. Principal Component erklärt 0.0000% der Varianz
10. Principal Component erklärt 0.0000% der Varianz
```

✗ X ohne „Response“ und „AcceptedCmp_Total“:

```
X = df.drop(["AcceptedCmp_Total", "Response"], axis  = 1)
```

✗ Standardscaler für Streuung und Varianz:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)
```

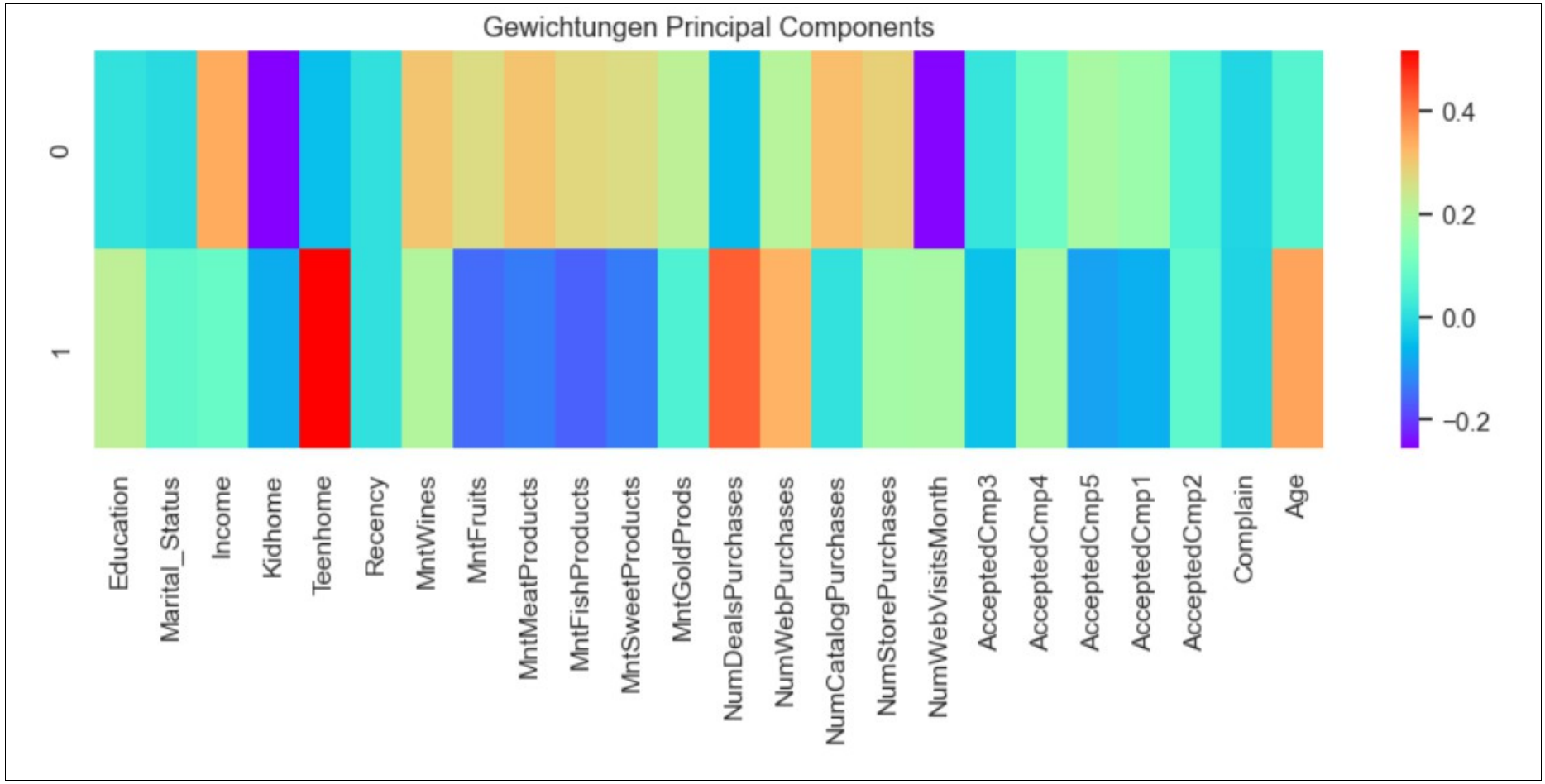
✗ PCA mit 2 Komponenten:

```
pca = PCA(n_components=2, random_state=33)
```

✗ Trainieren und Transformieren:

```
x_pca = pca.fit_transform(scaled_X)

x_pca.shape
```

✗ Gewichtung Principal Components mit den maximal enthaltenen Informationen:



SUPERVISED LEARNING:

- ✗ Verwendete Algorithmen:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

log = LogisticRegression()
knc = KNeighborsClassifier()
svc = SVC()
nab = GaussianNB()
rfc = RandomForestClassifier()
```

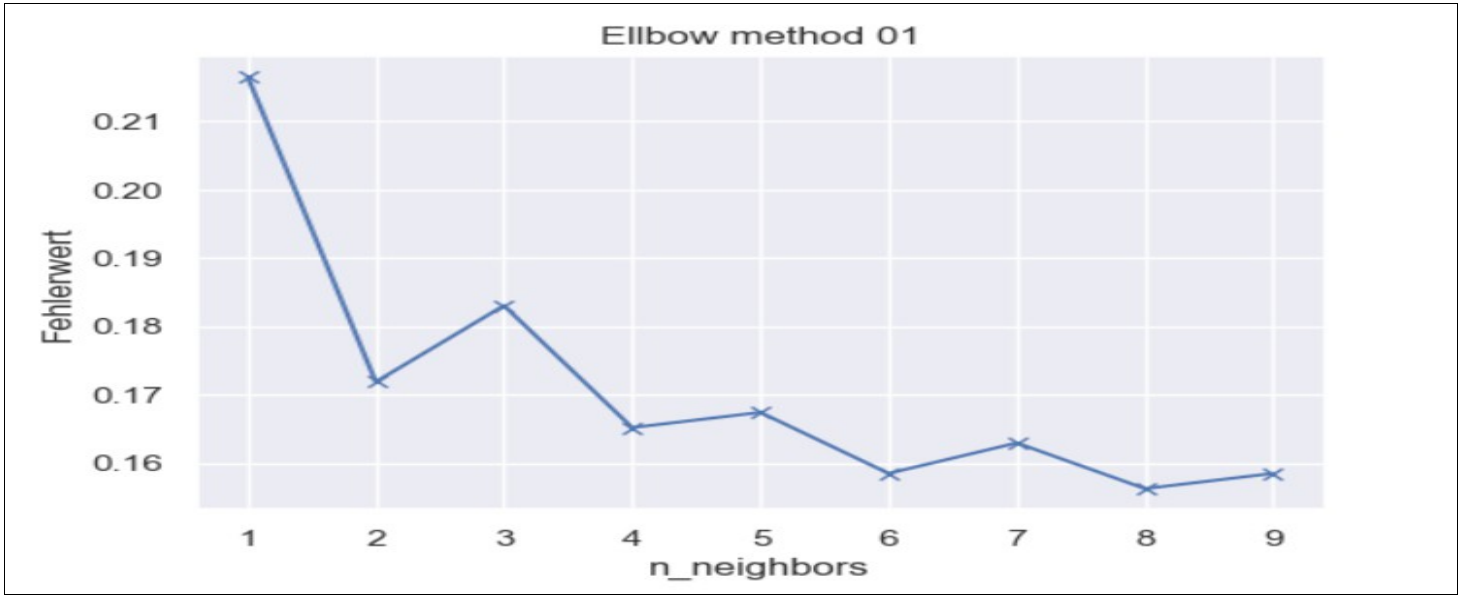
- ✗ alle features mit Label "Response" - PCA-reduziert:

X01

```
X01 = x_pca
y01 = df["Response"]
```

```
from sklearn.model_selection import train_test_split

X01_train, X01_test, y01_train, y01_test = train_test_split(X01, y01, test_size = 0.20, random_state = 33)
```

✗ Ellbow-method für KNN X01 – mit 4 neighbors von KNN bestes Ergebnis:



✗ Gridsearch für beste Parameter von SVC für X01:

```
from sklearn.model_selection import GridSearchCV

# dictionary mit Werten für C und gamma
hyperparameter = {"C" : [0.1, 1, 10, 100, 1000], "gamma" : [1, 0.1, 0.01, 0.001, 0.0001]}
```

```
1 grid01.best_params_
```

```
{'C': 10, 'gamma': 1}
```

✗ mit allen Algorithmen trainieren, vorhersagen und reports:

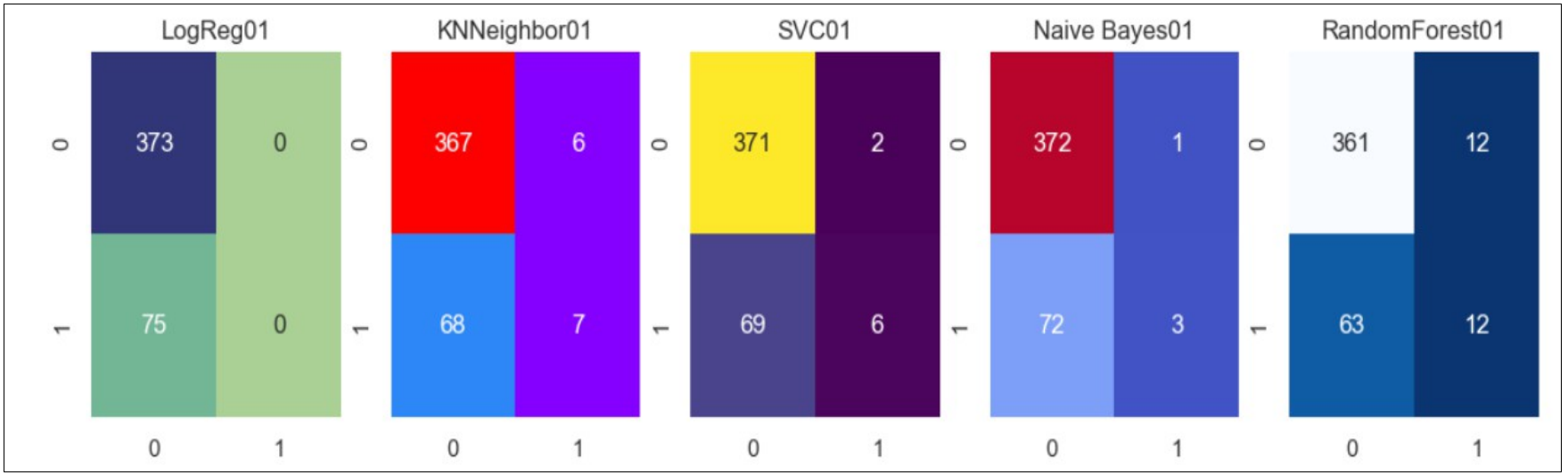
```
TRAINIEREN

log01 = LogisticRegression().fit(X01_train, y01_train)
knc01 = KNeighborsClassifier(n_neighbors = 4).fit(X01_train, y01_train) # bester Wert 4
svc01 = SVC(C=10, gamma=1).fit(X01_train, y01_train)
nab01 = GaussianNB().fit(X01_train, y01_train)
rfc01 = RandomForestClassifier(random_state = 33, n_estimators = 1000).fit(X01_train, y01_train)
```

```
VORHERSAGEN

pred_log01 = log01.predict(X01_test)
pred_knc01 = knc01.predict(X01_test)
pred_svc01 = svc01.predict(X01_test)
pred_nab01 = nab01.predict(X01_test)
pred_rfc01 = rfc01.predict(X01_test)
```

```
REPORTS

print("ALLE FEATURES:\n")
print("Genauigkeit LogReg01: {:>12.2f}%".format((accuracy_score(y01_test, pred_log01)*100)))
print("Genauigkeit KNN01   : {:>12.2f}%".format((accuracy_score(y01_test, pred_knc01)*100)))
print("Genauigkeit SVC01   : {:>12.2f}%".format((accuracy_score(y01_test, pred_svc01)*100)))
print("Genauigkeit NAIVE01 : {:>12.2f}%".format((accuracy_score(y01_test, pred_nab01)*100)))
print("Genauigkeit Random01: {:>12.2f}%\n".format((accuracy_score(y01_test, pred_rfc01)*100)))
```

✗ Report and Confusion matrix:

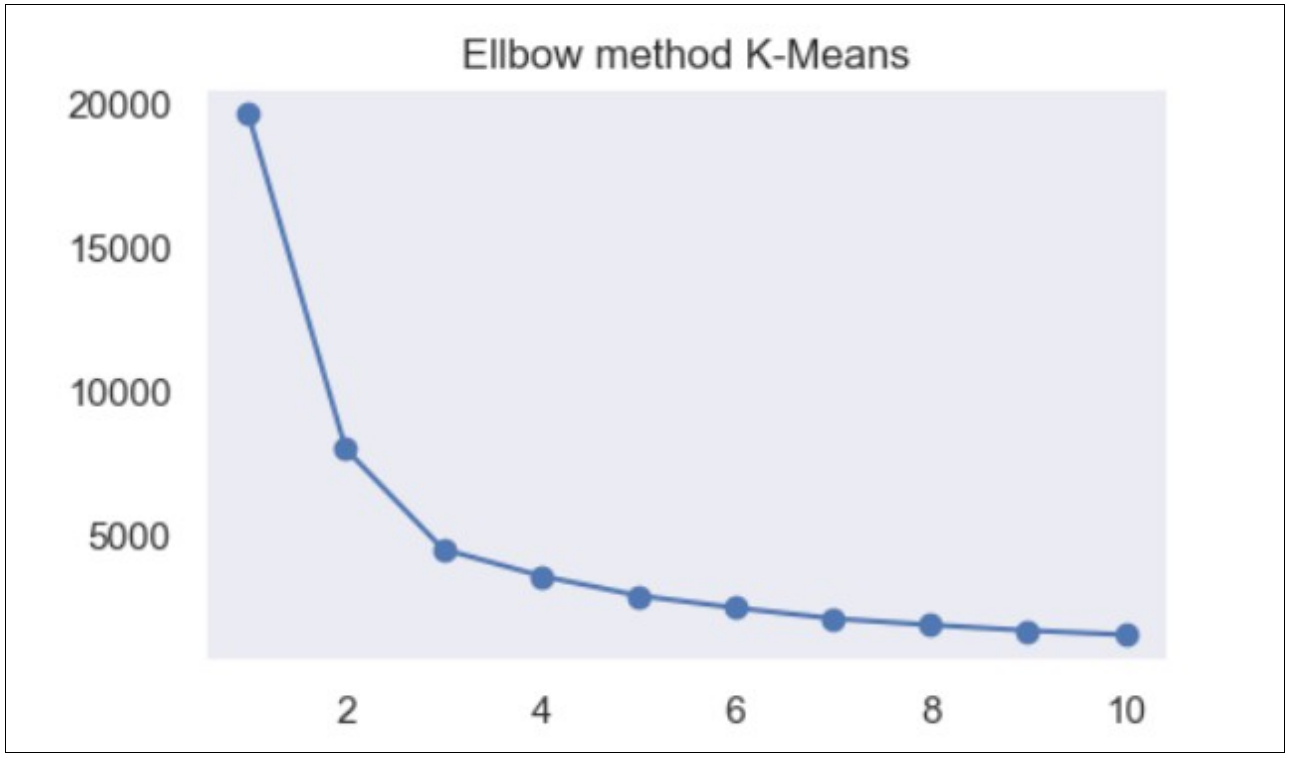
```
ALLE FEATURES PCA-REDUZIERT:

Genauigkeit LogReg01:        83.26%
Genauigkeit KNN01   :        83.48%
Genauigkeit SVC01   :        84.15%
Genauigkeit NAIVE01 :        83.71%
Genauigkeit Random01:        83.26%
```



UNSUPERVISED LEARNING:

- ✗ K-MEANS PCA-REDUZIERT

- ✗ Ellbow-method für K-Means:



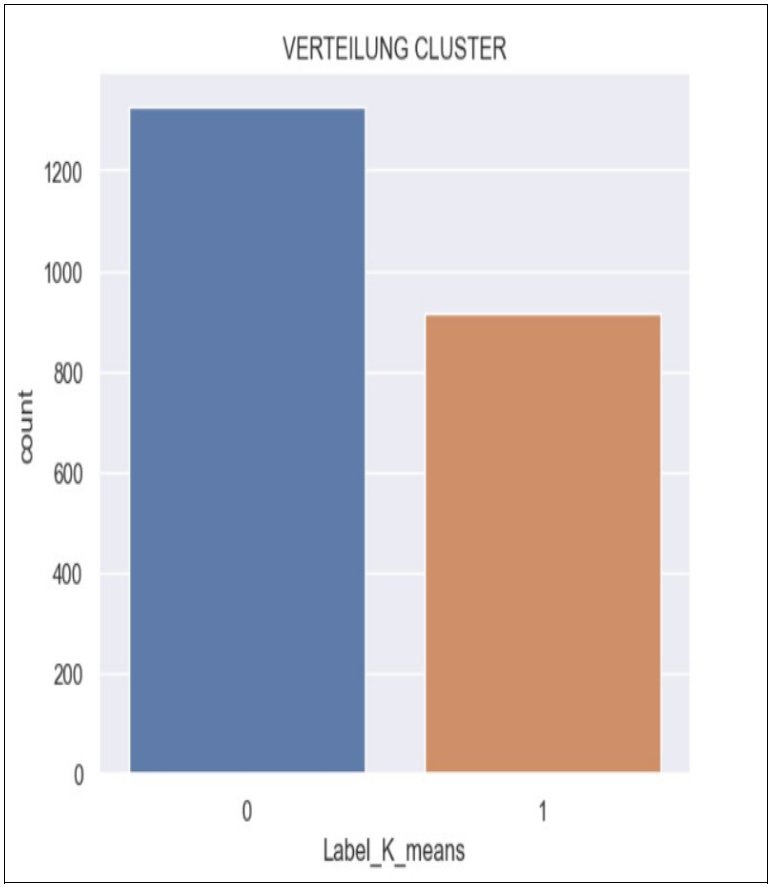
✗ mit K-Means trainieren und vorhersagen:

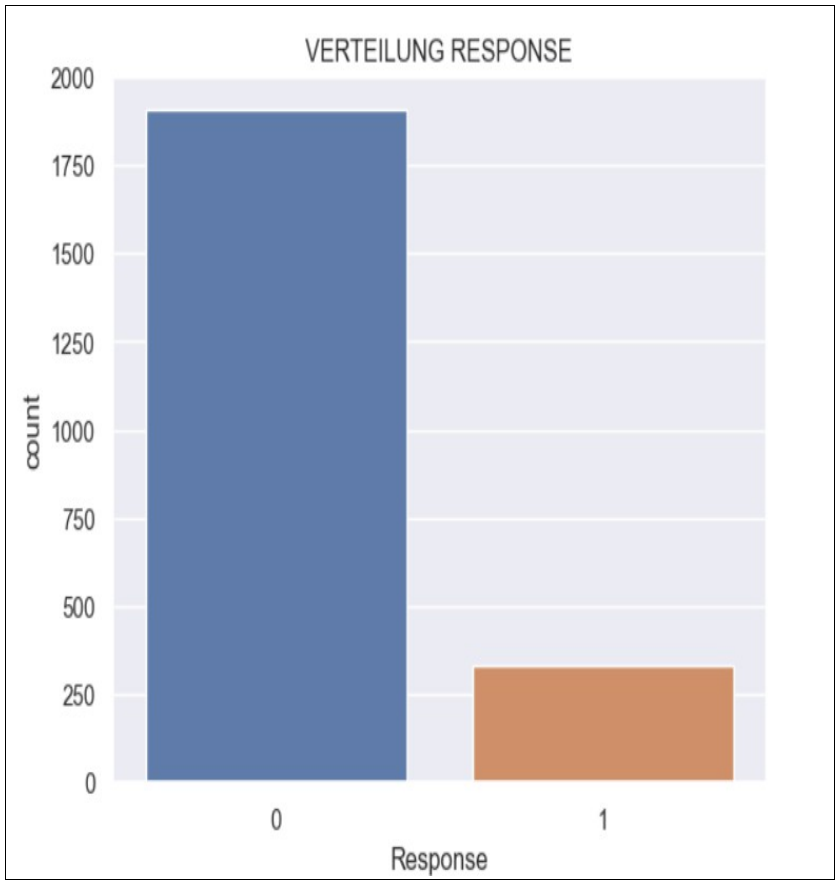
```
TRAINIEREN UND VORHERSAGE MIT 2 CLUSTERN:

km01 = KMeans(n_clusters=2, random_state=33)

pred_km01 = km01.fit_predict(x_pca)
```

✗ neues Label mit den vorhergesagten Clustern an ursprünglichen Datensatz hinzufügen:

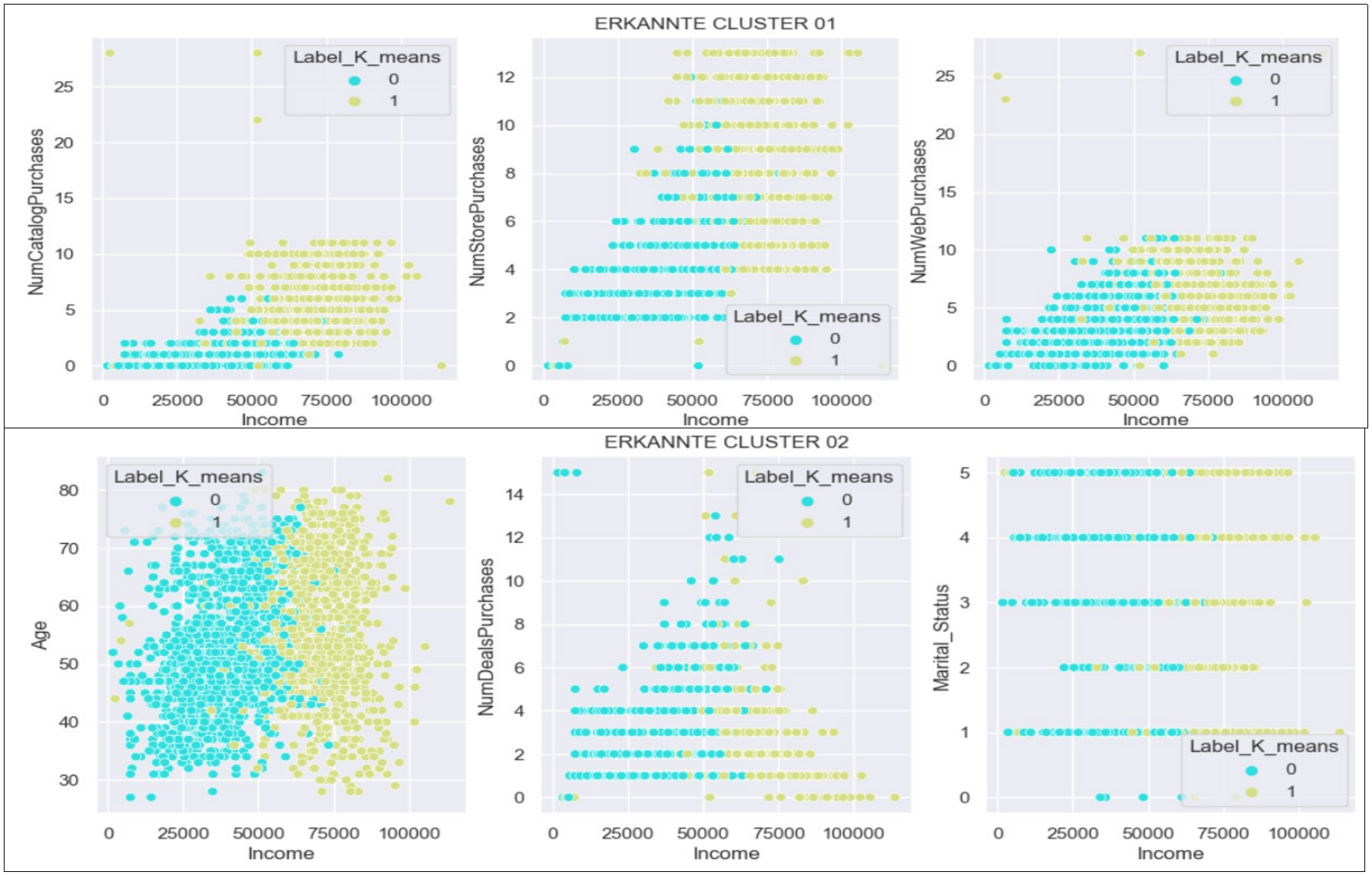
```
df["Label_K_means"] = pred_km01
```

✗ Visualisierung Verteilung neues Label „K_means“ und ursprüngliches Label „Response“ in einem countplot:

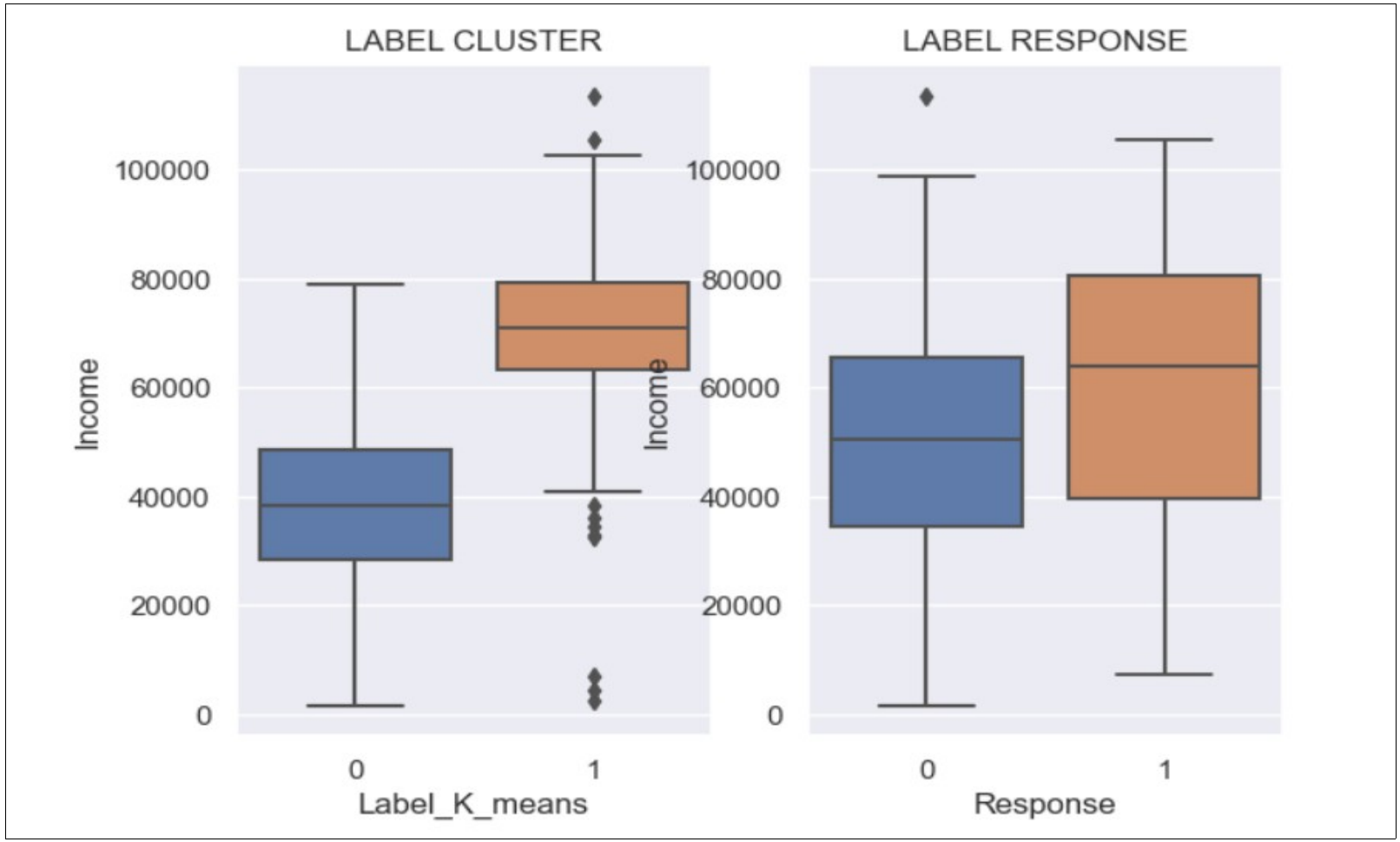




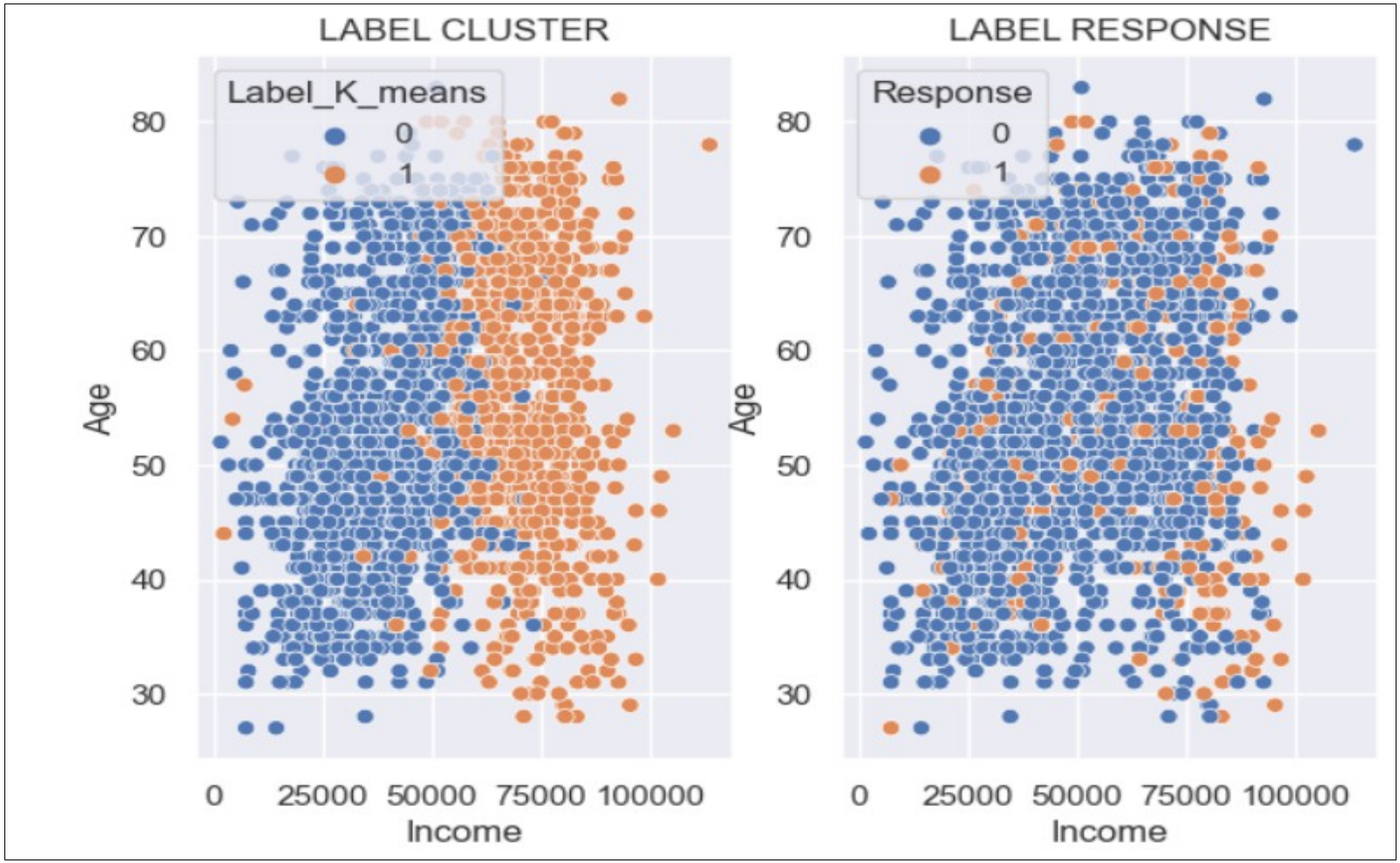
✗ Visualisierung Cluster mit verschiedenen features in einem scatterplot:



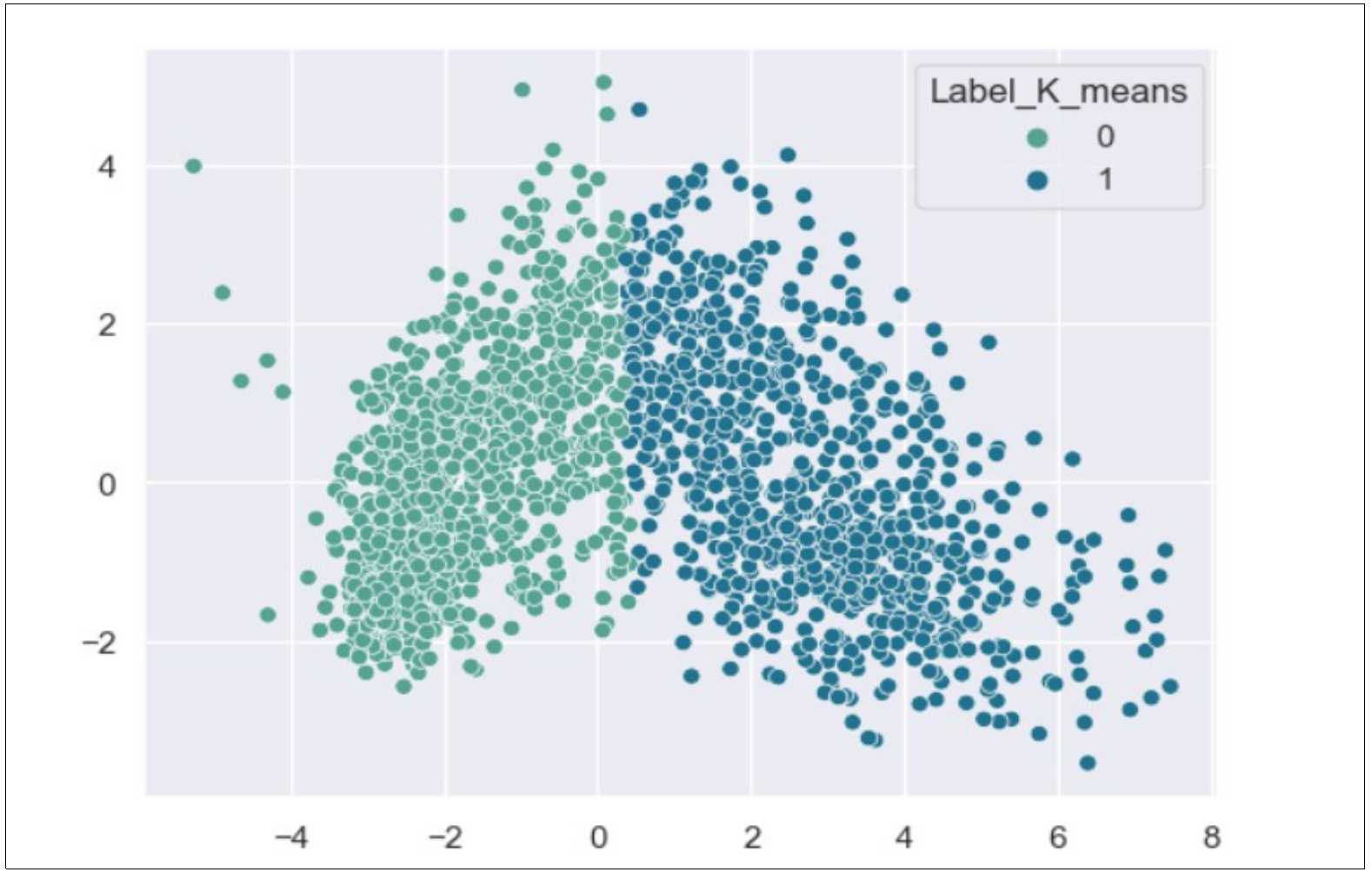
✗ Visualisierung „Response/Income“ - „Label K-Means/Income“ im boxplot:



✗ Vergleich ursprüngliches Label "Response" mit „K-Means Label“:



✗ Label K-Means in einem Scatterplot:



✗ Vergleich Label "Response" mit Vorhersagen "Clustering K-Means":

```
3 print(classification_report(df["Response"], pred_km01))
```

	precision	recall	f1-score	support
0	0.90	0.63	0.74	1906
1	0.23	0.62	0.33	334
accuracy			0.63	2240
macro avg	0.56	0.62	0.54	2240
weighted avg	0.80	0.63	0.68	2240

FAZIT:

➢ Ergebnisse Supervised Learning:

- die Algorithmen lieferten sehr gute Ergebnisse:

```
ALLE FEATURES PCA-REDUZIERT:

Genauigkeit LogReg01:        83.26%
Genauigkeit KNN01   :        83.48%
Genauigkeit SVC01   :        84.15%
Genauigkeit NAIVE01 :        83.71%
Genauigkeit Random01:        83.26%
```

- der Datensatz wurde im Bereich des Supervised - Learnings auch **ohne** PCA-reduzierte features bearbeitet – die Ergebnisse waren folgende:

```
Genauigkeit LogReg01:        88.62%
Genauigkeit KNN01   :        86.83%
Genauigkeit SVC01   :        87.72%
Genauigkeit NAIVE01 :        83.04%
Genauigkeit Random01:        86.16%
```

- durch die PCA-reduzierten features gibt es zwar Einbußen bei der accuracy, welche meiner Meinung nach aber zu verkraften sind

➢ Ergebnisse Unsupervised Learning – K-Means PCA-reduziert:

- die Ergebnisse des Clusterings sind im Vergleich zum ursprünglichen Label recht unterschiedlich, aber es ist ein Clustering zu erkennen
- außerdem sieht man, dass bei K-Means die Anzahl von „Response 01 – Kampagne angenommen“ dreimal höher ist als beim ursprünglichen Label Response
- K-Means kann man meiner Meinung nach zwar für die Fragestellung in diesem Datensatz, ob ein Kunde die nächste Kampagne annehmen wird oder nicht, verwenden, er klassifiziert teilweise etwas anders als es beim ursprünglichen Label „Response“ der Fall ist
- weiterführend könnte man überprüfen, ob mehr als 2 Cluster bessere Ergebnisse liefern würden, da die Werte des ursprünglichen Targets „Response“ mit 0 und 1 vielleicht nicht ideal sind
- grundsätzlich sind aber andere Algorithmen für die Fragestellung “Angebot der Kampagne angenommen: Ja/Nein“ sicher geeigneter als der K-Means