Notenbonusprojekt Gruppe 2

Segmentierung von Stammkunden

Für dieses Projekt liegen 200 Stammkundendaten vor. Es wird im Folgenden ein Cluster-Algorithmus entwickelt, mit welchem dann die Segmentierung der Kunden stattfindet. Daraus werden dann im Anschluss Handlungsempfehlungen abgeleitet.

1. Um den Datensatz verwenden zu können, wird dieser zu Beginn eingelesen. Anschließend werden die Attribute Alter, Jährliches Einkommen und Monatsausgaben standardisiert, um im Späteren besser mit den Daten arbeiten zu können. Die Standardisierung erfolgt mithilfe der *z-Transformation* . Dies ist sinnvoll, da Daten in einem Datensatz oftmals unterschiedlich skaliert sind. Die Transformation liefert eine einheitliche Skalierung.

```
9  #Zu Beginn werden die Kundendaten mit dem Package Pandas eingelesen;
   #importiere noch weitere wichtige Packages

   import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
   from sklearn import cluster, metrics



   kunden = pd.read_csv('Kundendaten.csv', index_col= 0, delimiter=";", decimal=",")

   #kunden = pd.get_dummies(kunden, columns=['Geschlecht'])


   kunden.head()
   kunden.rename(columns={'Alter':'Alter', 'Jährliches Einkommen (k€)':'Einkommen', 'Monatsausgaben (Ind



   for col in kunden.columns: #iteriert über alle Spalten im Datensatz
       if col == 'Geschlecht':
           continue
       kunden['z_' + col] = (kunden[col]-kunden[col].mean())/kunden[col].std()
   X = kunden[['z_Alter', 'z_Einkommen', 'z_Monatsausgaben']].values #Variablen auswählen auf deren Basi
```

Die folgende Tabelle zeigt nun den Datensatz mit allen Werten. Es wurden die standardisierten Werte hinzugefügt, welche mit einem **z** beginnen und die Dummyvariable für *Geschlecht*.

```
10 #Dummy hinzugefügt für Geschlecht
   #kunden['Dummy'] = kunden.Geschlecht.map( lambda Geschlecht: '1' if Geschlecht == 'Female' else '0')

   kunden.head()
   #print(X)
```

10

	Geschlecht	Alter	Einkommen	Monatsausgaben	z_Alter	z_Einkommen	z_M
KundenID							
1	Male	19	15	39	-1.421003	-1.734646	-0.4
2	Male	21	15	81	-1.277829	-1.734646	1.19
3	Female	20	16	6	-1.349416	-1.696572	-1.7
4	Female	23	16	77	-1.134655	-1.696572	1.03
5	Female	31	17	40	-0.561958	-1.658498	-0.3

2. Da wir nun die standardisierten Werte errechnet haben, können wir die erste Methode zur Kundensegmentierung anwenden. Diese nennt sich die *Ellbogenmethode* .

11

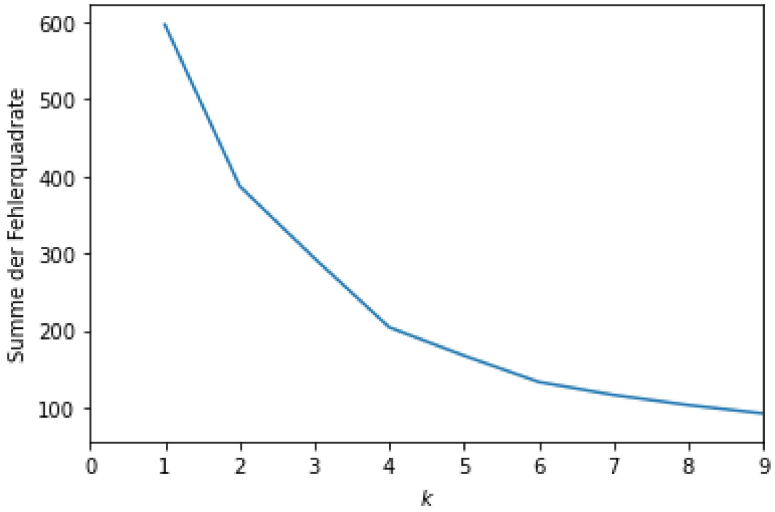
```
#Ellbogenmethode
ss = []
krange = list(range(1, 11)) #Hier einstellen, für welches Intervall die sum of squared errors berechn
for n in krange:
    model = cluster.KMeans(n_clusters=n, random_state=10)
    model.fit_predict(X)
    cluster_assignments = model.labels_
    centers = model.cluster_centers_
    ss.append(np.sum((X - centers[cluster_assignments]) ** 2))
```

```
C:\Users\mdrek\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is know
  warnings.warn(
```

Die Grafik zeigt nun das Ergebnis der *Ellbogenmethode* . Wir sehen, dass bei k=2,4 und 6 die Summe der Fehlerquadrate weniger stark als davor abfallen. Somit sollte laut dieser Methode das Clustering für k=2,4 und 6 stattfinden. Dies prüfen wir nun noch mit einer anderen Methode. Diese wird in **Abschnitt 3** gezeigt.

12

```
plt.plot(krange, ss)
plt.xlabel("$k$")
plt.ylabel("Summe der Fehlerquadrate")
plt.xlim(0,9)
plt.show()  #k=2, k=4, k=6
```

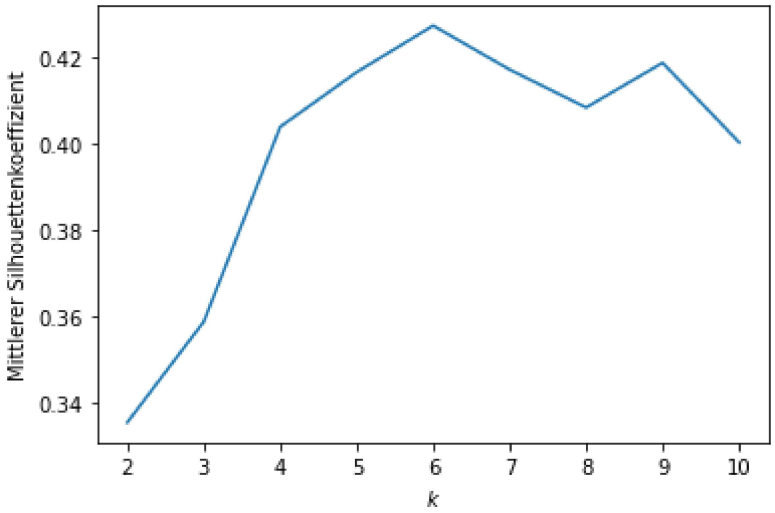


3. Es folgt nun die zweite Methode nach dem Silhouettenkoeffizient. Hier suchen wir ebenfalls das k nach dem höchsten mittleren Silhouettenkoeffizienten. Dieser wird mithilfe einer *for-Schleife* berechnet. Er gibt an, wie gut Cluster voneinander abgegrenz sind. Anschließend wird das Ergebnis grafisch dargestellt.

```
13  #Silhouettenkoeffizient
    krange = list(range(2, 11))
    avg_silhouettes = []
    for n in krange:
        model = cluster.KMeans(n_clusters=n, random_state=10)
        model.fit_predict(X)
        cluster_assignments = model.labels_
        silhouette_avg = metrics.silhouette_score(X, cluster_assignments)
        avg_silhouettes.append(silhouette_avg)
```

Die Grafik zeigt uns nun das Ergebnis der Berechnung. Wie schon bei der Ellbogenmethode, ist auch hier wieder k=6. Somit ist k=6 basierend auf dem Silhouettenkoeffizienten die beste Wahl.

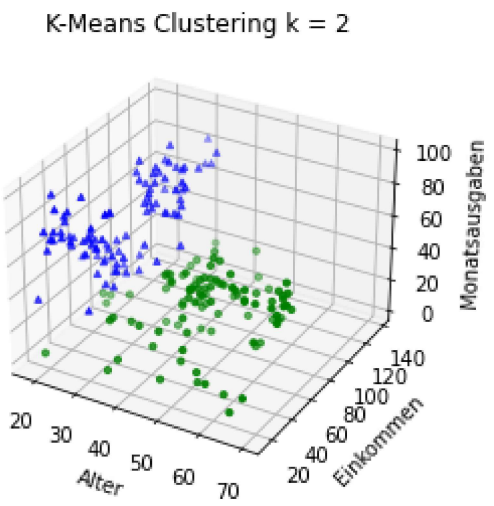
```
14  plt.plot(krange, avg_silhouettes)
    plt.xlabel("$k$")
    plt.ylabel("Mittlerer Silhouettenkoeffizient")
    plt.show() #k = 6 ist basierend auf dem Silhouettenkoeffizienten die beste Wahl
```

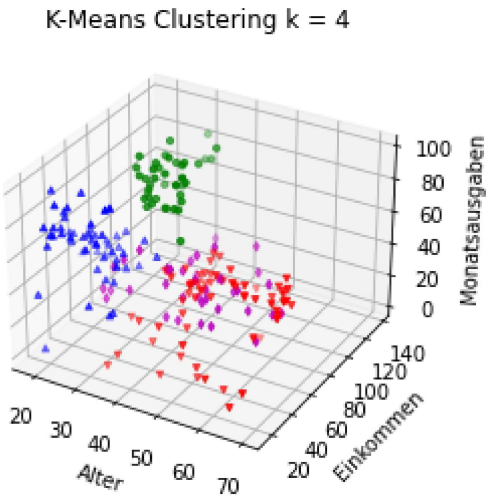


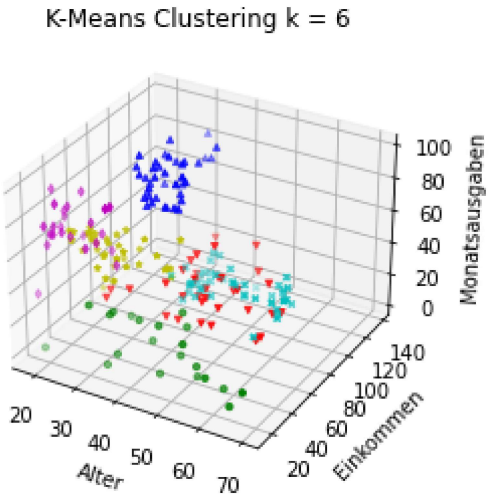
4. Die Methoden haben uns gezeigt, wie gut die Anzahl an Clustern zu unseren Daten passt. Nun wird die k-Means-Methode angewandt. Wir nehmen dazu k=2,4 und 6 wie oben bereits berechnet. Hierbei wird jede Dateninstanz dem nächstgelegenen Clusterzentrum zugeordnet. Dies wird farblich in der Grafik dargestellt.

Die Grafik ist 3-Dimensional, wobei die Attribute das Alter, Einkommen und die Montasausgaben sind.

22
```
#k-Means Clustering

for n in [2,4,6]:
    model = cluster.KMeans(n_clusters=n, random_state=10)
    cluster_assignments = model.fit_predict(X)
    kunden['cluster']=model.labels_
    colors = ['b','g','m','red','y','c']
    markers = ['^','o','d','v','*','x']


    fig = plt.figure()
    ax = fig.add_subplot(111, projection = '3d')

    for c in kunden['cluster'].unique():
        d = kunden[kunden['cluster'] == c]
        xs = d['Alter']
        ys = d['Einkommen']
        zs = d['Monatsausgaben']
        ax.scatter(xs, ys, zs,marker=markers[c], color=colors[c], s=10)
    plt.title('K-Means Clustering k = ' + str(n))
    ax.set_xlabel('Alter')
    ax.set_ylabel('Einkommen')
    ax.set_zlabel('Monatsausgaben')

    plt.show()
```







Auswertung der k-Means Methode

Hier sehen wir nun die verschiedenen Cluster nach der k-Means Methode. Da sowohl die Ellbogenmethode als auch der Silhouettenkoeffizient auf sechs Cluster kommen und durch die oberen Plots auch sichtbar ist, dass k=6 das sinnvollste Ergebnis liefert, werden wir im Folgenden sechs Cluster annehmen.

Cluster	Alter	Einkommen	Monatsausgaben
Blau	<30	>120	hoch
Lila	<20	<=80	hoch
Gelb	<30	80-120	mittel
Rot	30-50	>100	niedrig
Grün	20-60	<100	niedrig
Türkis	30-50	>120	niedrig

Wir sehen, dass das Einkommen bei der jüngeren Generation eine untergeordnete Rolle spielt, da in jeder Monatsausgabenkategorie das gesamte Spektrum der Einkommen

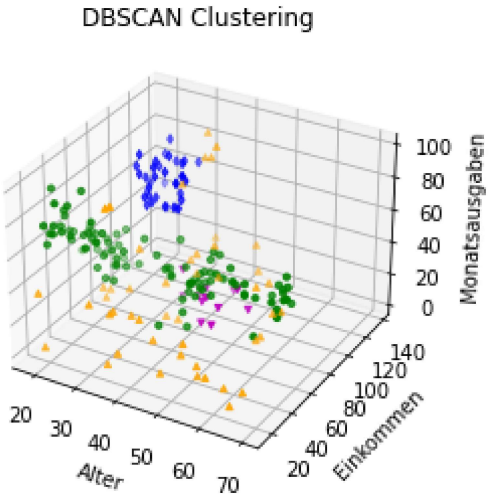
abgebildet ist. Der Hauptkundenstamm umfasst überwiegend junge Personen bis 30 Jahre. Das ältere Publikum ist weniger rentabel, obwohl hier das höchste Einkommen generiert wird.

5. Wir nutzen nun die **DBSCAN** Methode für die Segmentierung. Die folgende Grafik zeigt nun das Ergebnis des DBSCAN. Wir sehen deutlich 3 Cluster in Blau, Lila und Grün. Die Gelben Dateninstanzen sind überall verteilt.

21

```
colors = ['orange', 'g','b','m']
markers = ['^','o','d','v']

model3=cluster.DBSCAN(eps=0.61,min_samples=7,metric='euclidean')#epsilon=0.61 und minsamples=11 berei
cluster_assignments3 = model3.fit_predict(X)
kunden['cluster3'] = model3.labels_
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
print(kunden['cluster3'].unique())
for c in kunden['cluster3'].unique():
    d = kunden[kunden['cluster3'] == c]
    xs = d['Alter']
    ys = d['Einkommen']
    zs = d['Monatsausgaben']
    ax.scatter(xs, ys, zs,marker=markers[c+1], color=colors[c+1], s=10)

plt.title('DBSCAN Clustering')
ax.set_xlabel('Alter')
ax.set_ylabel('Einkommen')
ax.set_zlabel('Monatsausgaben')

plt.show()
```

```
[-1  0  1  2]
```



Auswertung der DBSCAN Methode

Bei dieser Methode ist nur ein eindeutiges Cluster (Blau) zu sehen. Durch diese Methode bekommen wir keine sinnvolle Erkenntnisse bezüglich des Kundenstamms. Auch eine Unterteilung in Männer und Frauen liefert keine guten Ergebnisse (siehe folgender Code).

5.1 DBSCAN für Frauen

24
```
#Neue Tabellen für Female

kunden2 = pd.read_csv('Kundendaten.csv', index_col= 0, delimiter=";", decimal=",")

#kunden = pd.get_dummies(kunden, columns=['Geschlecht'])


kunden2.head()
kunden2.rename(columns={'Alter':'Alter', 'Jährliches Einkommen (k€)':'Einkommen', 'Monatsausgaben (In

female = kunden2.loc[kunden2["Geschlecht"] == 'Female']
for col in female.columns: #iteriert über alle Spalten im Datensatz
    if col == 'Geschlecht':
        continue
    female['z_' + col] = (female[col]-female[col].mean())/female[col].std()
X_female = female[['z_Alter', 'z_Einkommen', 'z_Monatsausgaben']].values #Variablen auswählen auf der
```

```
C:\Users\mdrek\AppData\Local\Temp/ipykernel_1136/4281688992.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  female['z_' + col] = (female[col]-female[col].mean())/female[col].std()
```

157
```
#female.head()
```

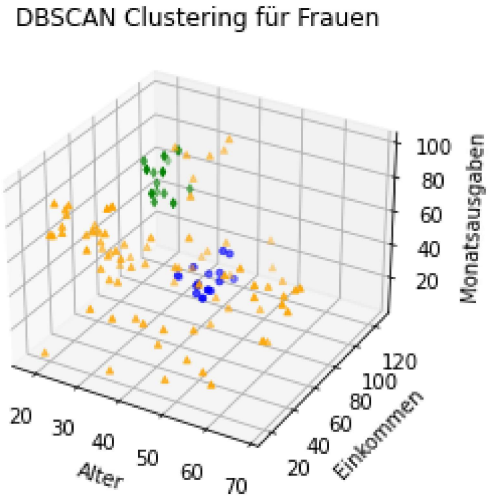
157

	Geschlecht	Alter	Einkommen	Monatsausgaben	z_Alter	z_Einkommen	z_M
KundenID							
3	Female	20	16	6	-1.431357	-1.662697	-1.8
4	Female	23	16	77	-1.194092	-1.662697	1.05
5	Female	31	17	40	-0.561386	-1.624253	-0.4
6	Female	22	17	76	-1.273180	-1.624253	1.01
7	Female	35	18	6	-0.245032	-1.585810	-1.8

158
```
##DBSCAN für female

colors = ['orange','b','g','m']
markers = ['^','o','d','v']

model4=cluster.DBSCAN(eps=0.61,min_samples=11,metric='euclidean')#epsilon=0.61 und minsamples=11 bere
cluster_assignments4 = model4.fit_predict(X_female)
female['cluster4'] = model4.labels_
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
print(female['cluster4'].unique())
for c in female['cluster4'].unique():
    d = female[female['cluster4'] == c]
    xs = d['Alter']
    ys = d['Einkommen']
    zs = d['Monatsausgaben']
    ax.scatter(xs, ys, zs,marker=markers[c+1], color=colors[c+1], s=10)

plt.title('DBSCAN Clustering für Frauen')
ax.set_xlabel('Alter')
```

```
ax.set_ylabel('Einkommen')
ax.set_zlabel('Monatsausgaben')

plt.show()
```

```
C:\Users\mdrek\AppData\Local\Temp/ipykernel_7008/820017732.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  female['cluster4'] = model4.labels_

[-1  0  1]
```



In der obigen Grafik ist nun die Methode des DBSCAN nur für Frauen zu sehen. Hier erkennen wir 3 CLuster. Ein grünes Cluster, welches Frauen mit hohem Einkommen und hohen Monatsausgaben zeigt. Diese sind auch eher jünger und befinden sich im Bereich der 20 - 30 Jährigen.

Ein anderes Cluster befindet sich im Bereich der 40 Jährigen mit mittleren Ausgaben und eher geringerem Einkommen.

Die anderen Dateninstanzen, in der Grafik Gelb gekennzeichnet, sind überall verteilt und keinem der beiden Cluster direkt zuzuordnen.

5.2 DBSCAN für Männer

```
#Neue Tabellen für Male

kunden3 = pd.read_csv('Kundendaten.csv', index_col= 0, delimiter=";", decimal=",")

#kunden = pd.get_dummies(kunden, columns=['Geschlecht'])


kunden3.head()
kunden3.rename(columns={'Alter':'Alter', 'Jährliches Einkommen (k€)':'Einkommen', 'Monatsausgaben (In

male = kunden3.loc[kunden3["Geschlecht"] == 'Male']
for col in male.columns: #iteriert über alle Spalten im Datensatz
    if col == 'Geschlecht':
        continue
```

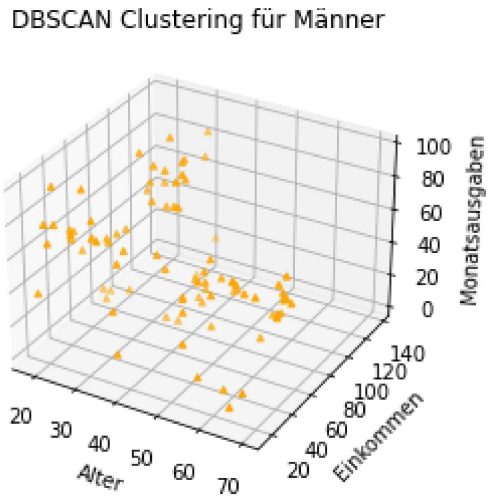
```
    male['z_' + col] = (male[col]-male[col].mean())/male[col].std()
X_male = male[['z_Alter', 'z_Einkommen', 'z_Monatsausgaben']].values #Variablen auswählen auf deren B

C:\Users\mdrek\AppData\Local\Temp/ipykernel_1136/3062113878.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  male['z_' + col] = (male[col]-male[col].mean())/male[col].std()
```

160
```
#male.head()
```

160

	Geschlecht	Alter	Einkommen	Monatsausgaben	z_Alter	z_Einkommen	z_M
KundenID							
1	Male	19	15	39	-1.341094	-1.772904	-0.3
2	Male	21	15	81	-1.212185	-1.772904	1.16
9	Male	64	19	3	1.559360	-1.622744	-1.6
11	Male	67	19	14	1.752724	-1.622744	-1.2
15	Male	37	20	13	-0.180912	-1.585205	-1.2

161
```
##DBSCAN für male

colors = ['orange','b','g','m']
markers = ['^','o','d','v']

model5=cluster.DBSCAN(eps=0.61,min_samples=11,metric='euclidean')#epsilon=0.61 und minsamples=11 bere
cluster_assignments5 = model5.fit_predict(X_male)
male['cluster5'] = model5.labels_
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
print(male['cluster5'].unique())
for c in male['cluster5'].unique():
    d = male[male['cluster5'] == c]
    xs = d['Alter']
    ys = d['Einkommen']
    zs = d['Monatsausgaben']
    ax.scatter(xs, ys, zs,marker=markers[c+1], color=colors[c+1], s=10)

plt.title('DBSCAN Clustering für Männer')
ax.set_xlabel('Alter')
ax.set_ylabel('Einkommen')
ax.set_zlabel('Monatsausgaben')

plt.show()

C:\Users\mdrek\AppData\Local\Temp/ipykernel_7008/3978180370.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  male['cluster5'] = model5.labels_

[-1]
```



Hier sind keine Cluster anhand des DBSCAN erkennbar. Alle Dateninstanzen sind verteilt. Es ist hier also kein bestimmtes Cluster zu erkennen. Dies führt nun dazu, dass wir eine andere Methode zur Kundensegmentierung anwenden müssen.

k-means Methode nach Geschlecht

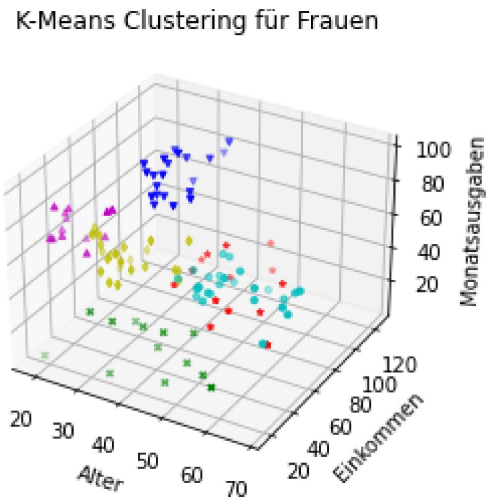
6. Nehmen wir nun nochmals das k-Means Clustering Verfahren zur Hand. Diesmal ebenfalls unterteilt in Frauen und Männer.

6.1 Zuerst schauen wir uns die Frauen an und nehmen k=6, da hier das Clustering am eindeutigsten ist.

```
#k-Means Clustering für Frauen

for n in [6]:
    model6 = cluster.KMeans(n_clusters=n, random_state=10)
    cluster_assignments6 = model6.fit_predict(X_female)
    female['cluster6']=model6.labels_
    colors = ['m','c','y','b','red','g']
    markers = ['^','o','d','v','*','x']


    fig = plt.figure()
    ax = fig.add_subplot(111, projection = '3d')

    for c in female['cluster6'].unique():
        d = female[female['cluster6'] == c]
        xs = d['Alter']
        ys = d['Einkommen']
        zs = d['Monatsausgaben']
        ax.scatter(xs, ys, zs,marker=markers[c], color=colors[c], s=10)
    plt.title('K-Means Clustering für Frauen')
    ax.set_xlabel('Alter')
    ax.set_ylabel('Einkommen')
    ax.set_zlabel('Monatsausgaben')

    plt.show()
```

```
C:\Users\mdrek\AppData\Local\Temp/ipykernel_1136/3104055329.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

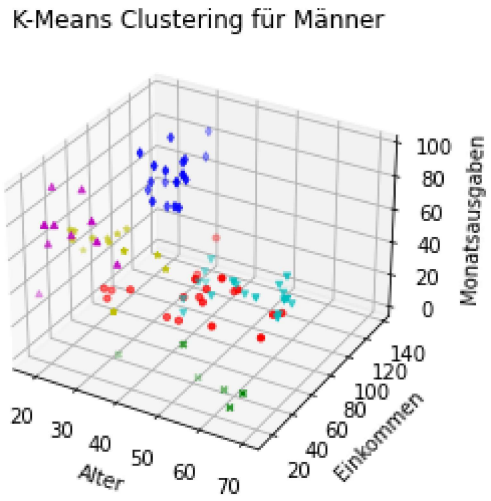
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  female['cluster6']=model6.labels_
```



6.2 Schauen wir uns nun nochmals das Clustering für Männer an:

28
```
#k-Means Clustering für Männer

for n in [6]:
    model7 = cluster.KMeans(n_clusters=n, random_state=10)
    cluster_assignments7 = model7.fit_predict(X_male)
    male['cluster7']=model7.labels_
    colors = ['m','red','b','c','y','g']
    markers = ['^','o','d','v','*','x']


    fig = plt.figure()
    ax = fig.add_subplot(111, projection = '3d')

    for c in male['cluster7'].unique():
        d = male[male['cluster7'] == c]
        xs = d['Alter']
        ys = d['Einkommen']
        zs = d['Monatsausgaben']
        ax.scatter(xs, ys, zs,marker=markers[c], color=colors[c], s=10)
    plt.title('K-Means Clustering für Männer')
    ax.set_xlabel('Alter')
    ax.set_ylabel('Einkommen')
    ax.set_zlabel('Monatsausgaben')

    plt.show()
```

```
C:\Users\mdrek\AppData\Local\Temp/ipykernel_1136/3557585589.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  male['cluster7']=model7.labels_
```



Auswertung der k-means Methode nach Geschlecht

Die Aufteilung in Männer und Frauen liefert identische Cluster. Wie bereits am Anfang, befinden sich die auch hier die Hauptcluster im Bereich *jünger*. In diesen Gruppen haben wir die höchten Ausgaben. Die Cluster der älteren Generation liefern ebenfalls keine eindeutigen Ergebnisse.

Unsere Handlungsempfehlungen werden deshalb nicht zwischen Männern und Frauen differenzieren. Sie beziehen sich auf die gesamte jüngere oder ältere Generation, unabhängig vom Geschlecht.

Handlungsempfehlungen

Cluster	Ziel	Handlungsempfehlung
Blau	Kunden halten + weiterempfehlung	Treuepunkte, Umweltprogramm-Werbung, Werbung in Sozialen Medien, Bonus für das Anwerben von Neukunden
Lila	Kunden halten + weiterempfehlung	Treuepunkte, Umweltprogramm-Werbung, Werbung in Sozialen Medien, Bonus für das Anwerben von Neukunden
Gelb	Kunden halten + weiterempfehlung	Treuepunkte, Umweltprogramm-Werbung, Werbung in Sozialen Medien, Bonus für das Anwerben von Neukunden, Rabattaktionen
Rot	Kunden anwerben + über Kinder ansprechen	Werbung in der Zeitung, Säulen, TV, Prospekte, Aktionstage mit Sektempfang, Kinderbetreuung

Cluster	Ziel	Handlungsempfehlung
Grün	--	Zu unspezifisch für gezielte Maßnahmen; Gruppe wird eventuell durch andere vorhandene Maßnahmen angesprochen
Türkis	Kunden anwerben + über Kinder ansprechen	Werbung in der Zeitung, Säulen, TV, Prospekte, Aktionstage mit Sektempfang, Kinderbetreuung