Inhaltsverzeichnis

Einleitung

Im Rahmen des Moduls Data Mining and Machine Learning werden wir einen von uns ausgewählten Datensatz mit der Clustering Methode statistisch analysieren und auswerten. Dabei wählen wir geeignete Clustering-Algorithmen an, um Muster und Strukturen in den Daten zu identifizieren. Durch diese statistische Analyse erhalten wir Einblicke in die zugrunde liegenden Zusammenhänge und können die Daten entsprechend interpretieren. Die Ergebnisse unserer Auswertung ermöglichen es uns, Erkenntnisse zu gewinnen und möglicherweise Trends oder Gruppierungen in den Daten zu erkennen, die für weitere Entscheidungen oder Analysen relevant sein könnten.

Eigenständigkeitserklärung

Hiermit erkläre/n ich/wir, dass ich/wir die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe/n. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht.

Genutzter Datensatz

Datensatz Anpassung

Damit unser Output nur 90% des originalen Datensatzes beinhaltet, werden mit dem unteren Skript zufällig Daten aus dem Datensatz ausgewählt.

In [ ]:
```python
import os
import random
import csv

# Specify the path to the CSV files
input_file = './tripadvisor_review.csv'
# Datei heißt output2.csv weil schon eine Datei output.csv existiert und
output_file = './output2.csv'
# Read the input CSV file
with open(input_file, 'r') as file:
    reader = csv.reader(file)
    data = list(reader)
# Remove the first column from the data (it contains the user IDs)
data = [row[1:] for row in data]
# Keep the first row in the data for the header
header = data[0]
# Filter out possible duplicates from the data
input_file = list(set(tuple(row) for row in input_file))
# Calculate the number of rows to keep (90% of the total rows)
num_rows = int(len(data) * 0.9)
# Select random num_rows rows from the data
selected_data = random.sample(data[1:], num_rows)
# Add the header back to the selected data
selected_data.insert(0, header)
# Write the selected data to the output CSV file
with open(output_file, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(selected_data)
```

Was bilden die Daten ab

Die Daten bilden das User Feedback zu verschiedenen Angeboten an Ferien Resorts in Asien auf einer Skala von 0 bis 4 ab. Diese werden dann mit der Durchschnittsbewertung des jeweiligen Angebots verrechnet, damit generell schlecht bewertete Angebote nicht zu stark ins Gewicht fallen. Gleiches gilt für sehr gut bewertete Angebote.

Aufbau des Datensatzes

Variablen

Attribute 1 : Unique user id
Attribute 2 : Average user feedback on art galleries
Attribute 3 : Average user feedback on dance clubs
Attribute 4 : Average user feedback on juice bars
Attribute 5 : Average user feedback on restaurants
Attribute 6 : Average user feedback on museums
Attribute 7 : Average user feedback on resorts
Attribute 8 : Average user feedback on parks/picnic spots
Attribute 9 : Average user feedback on beaches

Attribute 10 : Average user feedback on theaters
Attribute 11 : Average user feedback on religious institutions

Attribut 1 ist die User ID und die Attribute 2 bis 11 sind die Durchschnittsbewertungen der User für die jeweiligen Kategorien. Daher haben wir uns dafür entschieden, die User ID zu entfernen, da sie für unsere Analysen nicht relevant ist. Diese Entscheidung wurde getroffen, da wir die Daten in Kategorien clustern wollen und die User ID keine Kategorie darstellt. Die User ID wird erst interessant, wenn die Daten ausgewertet werden, da wir dann die User ID mit den Clustern in Verbindung bringen können.

Analysen

Ziel der Analysen

Das Ziel der Analysen ist es, die User zu clustern und die Cluster zu analysieren. Dadurch können die Resorts besser auf die Bedürfnisse der User angepasst werden. Und die Seite, die den Usern die Resorts anbietet, kann besser auf die Bedürfnisse der User eingehen, indem sie besser auf den User zugeschnittene Reiseziele vorschlägt.

KMeans

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [ ]:
```
# Load the data from the CSV file
df = pd.read_csv('./output.csv')
df
```

Out[ ]:

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8
0	1.25	1.24	0.43	0.62	0.72	2.70	3.18	3.07
1	0.91	1.96	1.82	0.40	1.52	1.82	3.19	2.74
2	0.77	1.36	0.43	0.54	1.72	2.91	3.18	2.94
3	0.96	1.32	0.56	0.42	0.98	2.06	3.17	2.82
4	0.74	2.00	1.66	0.49	1.62	1.76	3.18	2.54
...	...	...	...	...	...	...	...	...
877	0.77	1.08	1.82	0.39	1.06	2.32	3.19	2.85
878	0.93	1.28	0.93	0.46	0.42	1.50	3.18	2.70
879	0.54	1.52	0.24	0.37	0.74	1.50	3.18	2.89
880	0.58	1.80	0.70	0.53	1.52	2.22	3.18	2.78
881	0.74	1.68	2.89	0.57	0.94	1.92	3.20	2.86

882 rows × 10 columns

In [ ]:
```
# Describe the data
df.describe()
```

Out[ ]:

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Cat
count	882.000000	882.000000	882.000000	882.000000	882.000000	882.000000	882.
mean	0.894127	1.357914	1.035317	0.532449	0.945601	1.852880	3
std	0.324176	0.483453	0.794687	0.282624	0.442922	0.542676	0.
min	0.340000	0.000000	0.130000	0.170000	0.060000	0.380000	3.
25%	0.700000	1.080000	0.275000	0.410000	0.640000	1.460000	3.
50%	0.830000	1.280000	0.890000	0.500000	0.900000	1.820000	3.
75%	1.020000	1.600000	1.600000	0.580000	1.220000	2.220000	3.
max	3.220000	3.640000	3.620000	3.440000	3.300000	3.380000	3.

In [ ]:
```
# Scale the data and fit the KMeans model
scaler = StandardScaler()
df[['Category 1_T','Category 2_T','Category 3_T','Category 4_T','Category
]] = scaler.fit_transform(df[['Category 1','Category 2','Category 3','Cat
]])
df
```

Out[ ]:

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8
0	1.25	1.24	0.43	0.62	0.72	2.70	3.18	3.07
1	0.91	1.96	1.82	0.40	1.52	1.82	3.19	2.74
2	0.77	1.36	0.43	0.54	1.72	2.91	3.18	2.94
3	0.96	1.32	0.56	0.42	0.98	2.06	3.17	2.82
4	0.74	2.00	1.66	0.49	1.62	1.76	3.18	2.54
...	...	...	...	...	...	...	...	...
877	0.77	1.08	1.82	0.39	1.06	2.32	3.19	2.85
878	0.93	1.28	0.93	0.46	0.42	1.50	3.18	2.70
879	0.54	1.52	0.24	0.37	0.74	1.50	3.18	2.89
880	0.58	1.80	0.70	0.53	1.52	2.22	3.18	2.78
881	0.74	1.68	2.89	0.57	0.94	1.92	3.20	2.86

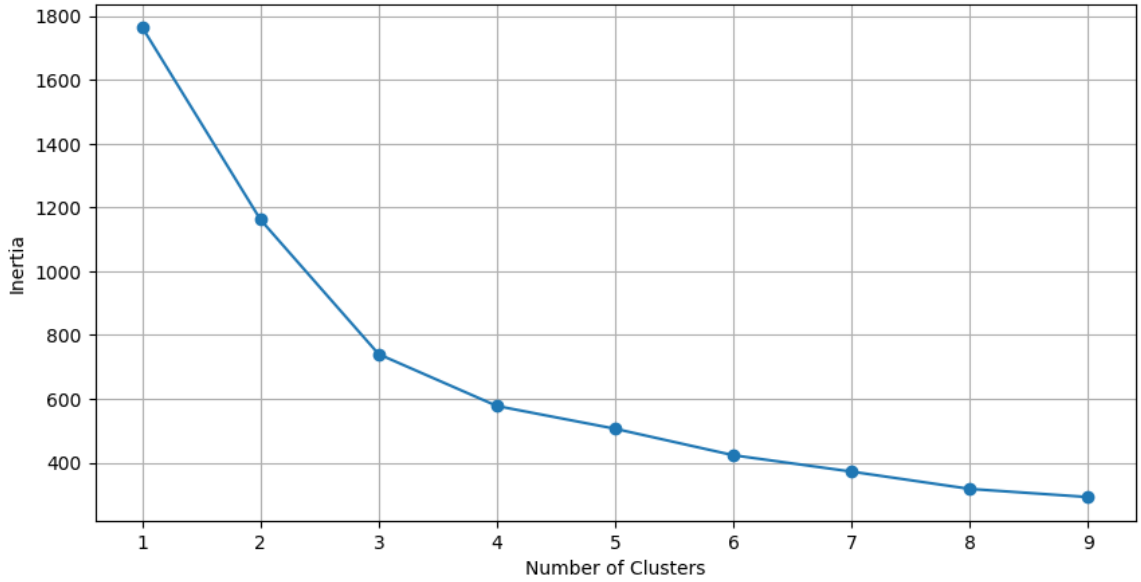
882 rows × 20 columns

Anzahl an Cluster bestimmen

Die Methode "optimise_k_means" ermittelt die optimale Anzahl von Clustern für die Anwendung der K-Means-Methode, um eine angemessene Gruppierung der Daten zu erreichen.

In [ ]:
```
# Defining a function to optimise the number of clusters
# This function shows the inertia(distance between the points in one clus
def optimise_k_means(data, max_k):
    means = []
    inertias = []

    for k in range(1, max_k):
        kmeans = KMeans(n_clusters=k)
        kmeans.fit(data)

        means.append(k)
        inertias.append(kmeans.inertia_)

    fig = plt.subplots(figsize=(10, 5))
    plt.plot(means, inertias, 'o-')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.grid(True)
    plt.show()
```

In [ ]:
```
# Using the function to optimise the number of clusters
optimise_k_means(df[['Category 1_T','Category 2_T']], 10)
```



Anhand der Methode "optimise_k_means" wird die optimale Anzahl von Clustern ermittelt. Die Methode berechnet die Summe der quadrierten Abweichungen (Sum of Squared Distances, SSD) für eine Vielzahl von Clustergrößen und gibt die optimale Anzahl von Clustern zurück. Die Methode wird verwendet, um die optimale Anzahl von Clustern zu bestimmen, um eine angemessene Gruppierung der Daten zu erreichen. Daher haben wir die Entscheidung getroffen, im weiteren Verlauf der Analyse die Anzahl von Clustern zu verwenden, die von der Methode "optimise_k_means" zurückgegeben wird. In unserem Fall ist die optimale Anzahl von Clustern 5.

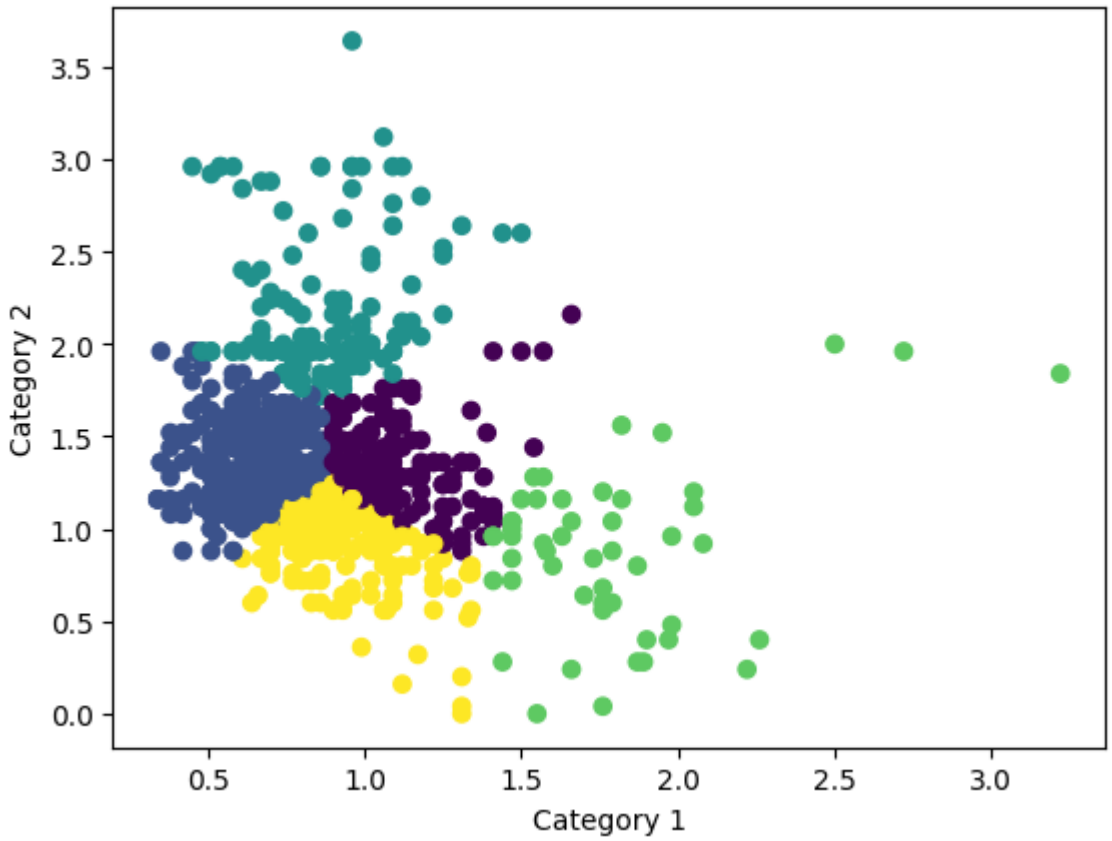
In [ ]:
```
# Doing the KMeans clustering with 5 clusters
kmeans = KMeans(n_clusters=5)
kmeans.fit(df[['Category 1_T','Category 2_T']])
df['kmeans_5'] = kmeans.labels_
df
```

Out[ ]:

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8
0	1.25	1.24	0.43	0.62	0.72	2.70	3.18	3.07
1	0.91	1.96	1.82	0.40	1.52	1.82	3.19	2.74
2	0.77	1.36	0.43	0.54	1.72	2.91	3.18	2.94
3	0.96	1.32	0.56	0.42	0.98	2.06	3.17	2.82
4	0.74	2.00	1.66	0.49	1.62	1.76	3.18	2.54
...	...	...	...	...	...	...	...	...
877	0.77	1.08	1.82	0.39	1.06	2.32	3.19	2.85
878	0.93	1.28	0.93	0.46	0.42	1.50	3.18	2.70
879	0.54	1.52	0.24	0.37	0.74	1.50	3.18	2.89
880	0.58	1.80	0.70	0.53	1.52	2.22	3.18	2.78
881	0.74	1.68	2.89	0.57	0.94	1.92	3.20	2.86

882 rows × 21 columns

In [ ]:
```
# Plotting the clusters
plt.scatter(df['Category 1'], df['Category 2'], c=df['kmeans_5'],)
plt.xlabel('Category 1')
plt.ylabel('Category 2')
plt.show()
```



Hierarchisch Agglomerativ

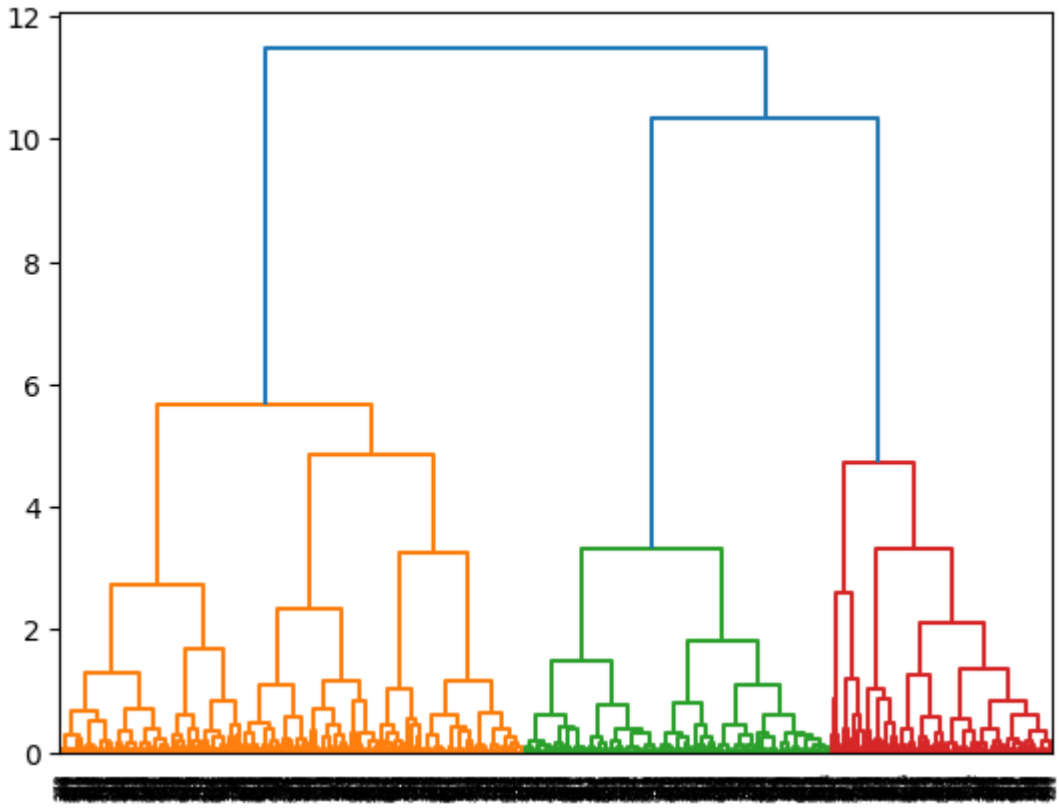
In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch

from sklearn.cluster import AgglomerativeClustering
```

In [ ]:
```
# Load the data from the CSV file and select the relevant columns
dataset = pd.read_csv('./output.csv')
points = dataset.iloc[:, [8, 9]].values
```

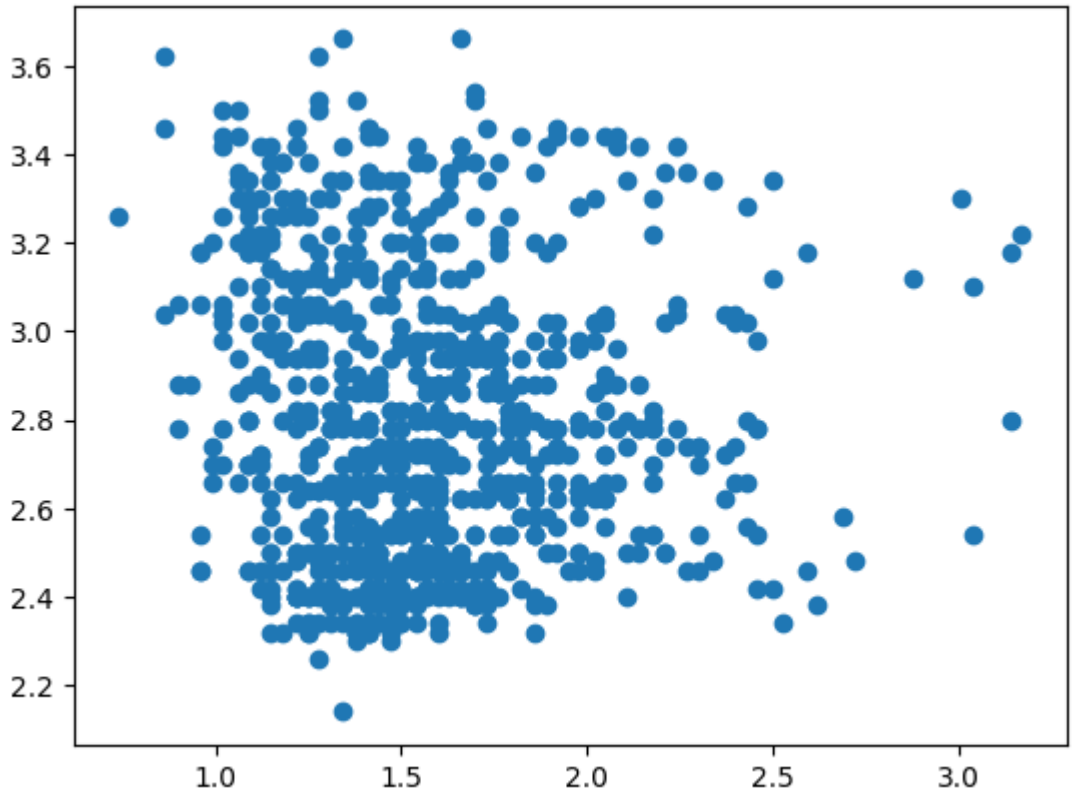
Distanzmatrix

Wir verwenden die Funktion "sch.linkage", um direkt die Verknüpfungen zwischen den Punkten zu berechnen. Die Methode 'ward' in dieser Funktion weist jedoch darauf hin, dass die Ward-Methode für das agglomerative Clustering verwendet wird, und intern wird eine Distanzmatrix erstellt, um die Hierarchie der Cluster zu erstellen. Die Berechnung der Distanzen erfolgt durch die Verwendung der Ward-Methode.

In [ ]:
```
# Show the Data in a Dendrogram
dendrogram = sch.dendrogram(sch.linkage(points, method='ward'))
```



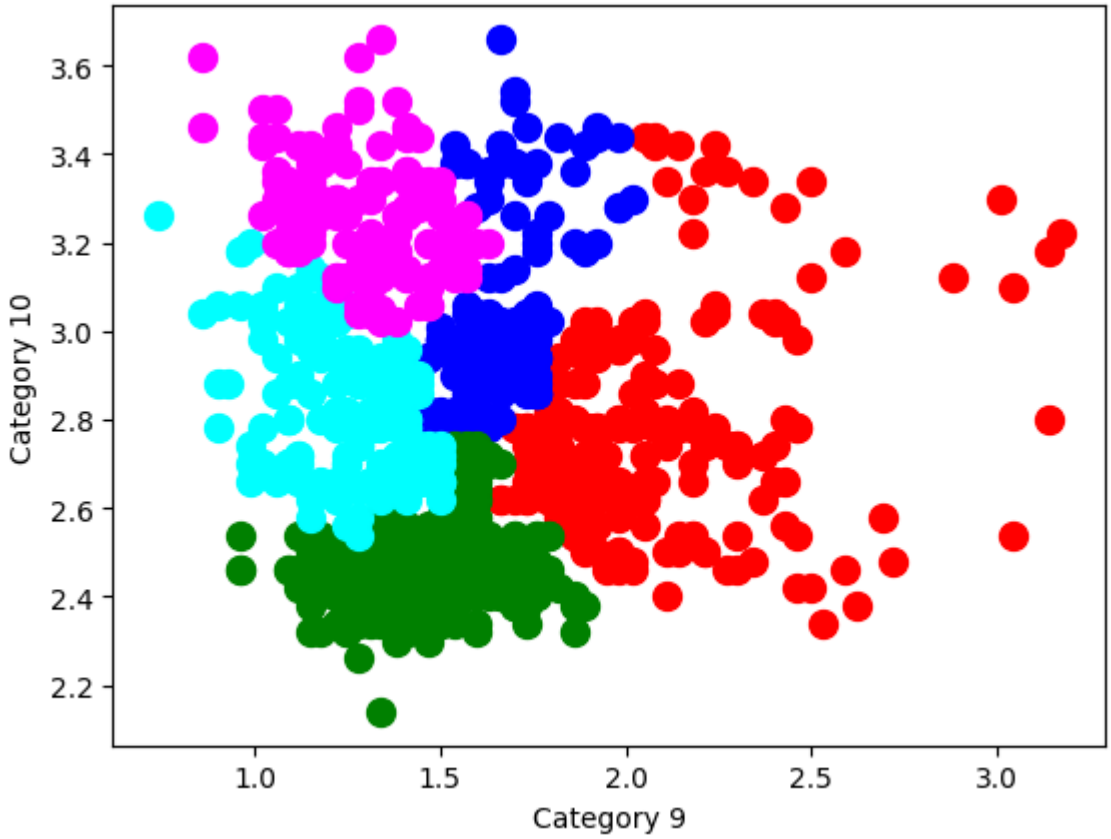
In [ ]:
```
# # Show the Data Points
plt.scatter(dataset.iloc[:, 8], dataset.iloc[:, 9])
plt.show()
```



In [ ]:
```
# Perform agglomerative clustering on the given points
hc = AgglomerativeClustering(n_clusters=5, linkage='ward')
y_hc = hc.fit_predict(points)
```

In [ ]:
```
plt.scatter(points[y_hc == 0, 0], points[y_hc == 0, 1], s=100, c='red', l
plt.scatter(points[y_hc == 1, 0], points[y_hc == 1, 1], s=100, c='blue',
plt.scatter(points[y_hc == 2, 0], points[y_hc == 2, 1], s=100, c='green',
plt.scatter(points[y_hc == 3, 0], points[y_hc == 3, 1], s=100, c='cyan',
plt.scatter(points[y_hc == 4, 0], points[y_hc == 4, 1], s=100, c='magenta
plt.xlabel('Category 9')
plt.ylabel('Category 10')
```

Out[ ]:
```
Text(0, 0.5, 'Category 10')
```



Auswertung

Bezogen auf die K_means Methode kann untersucht werden, ob es einen Zusammenhang zwischen dem Feedback zu Kunstgalerien(1) und Tanzclubs(2) besteht und ob bestimmte Nutzergruppen dazu neigen, ähnliche Feedback Muster zu haben. Dies könnte Einblicke in das Verhalten und die Vorlieben der Nutzer liefern. Das Cluster zeigt, dass die meisten Teilnehmer die Attraktionen in Bereich von 0.5 bis 1.5 bezogen Kunstgalerien und 0.5 bis 2 bezogen Tanzclubs bewertet haben. In dem Bereich mit den meisten Einträgen, sind die Tanzclubs besser bewertet als die Kunstgalerien. Teilnehmer, die eine der beiden Kategorien gut bis sehr gut bewertet haben, bewerteten die andere schlechter. Vor allem haben die Teilnehmer die Tanzclubs gut bewertet und die Kunstgalerien schlechter bewertet. Im Vergleich beider Attraktionen bekommen die Tanzclubs mehr gute Bewertungen. Dass die meisten Punkte eher im unteren linken Bereich sind und nicht oben rechts, zeigt, dass die meisten Teilnehmer entweder die Kunstgalerien oder die Tanzclubs mochten.

Bezogen auf die hierarchisch Agglomerative Methode kann untersucht werden, ob es einen Zusammenhang zwischen dem Feedback zu Theatern(9) und religiösen(10) Institutionen besteht und ob bestimmte Nutzergruppen dazu neigen, ähnliche Feedback Muster zu haben. Dies könnte Einblicke in das Verhalten und die Vorlieben der Nutzer liefern. Der Großteil der religiösen Institutionen wird gut bis sehr gut bewertet, im Gegensatz dazu werden die Theater eher schlecht bis mittelmäßig bewertet. Es gibt sehr wenige Teilnehmer, die Theater 3 oder besser bewertet haben. Im rechten, oberen und unteren Bereich des Graphen gibt es wenige Punkte. Das

lässt darauf schließen, dass Teilnehmer, die religiöse Institutionen mit 3 oder besser bewerten, die Theater schlechter bewerten. Die untere Lücke weist darauf hin, dass Leute, die das Theater gut bewerten, religiöse Institutionen auch gut bewerten.