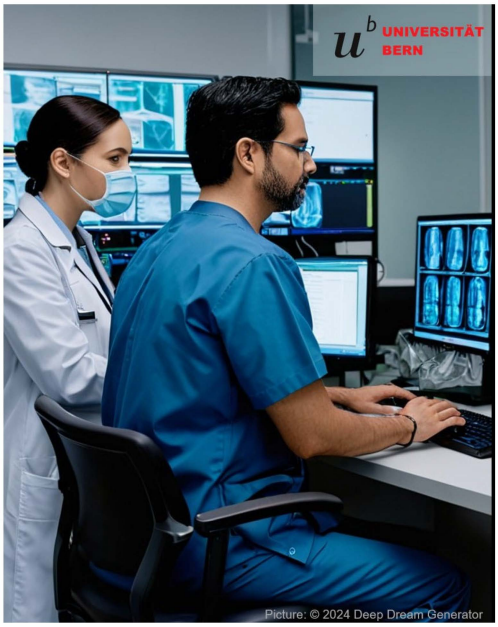
Unravelling the Human Interaction with Generative AI-Based Decision Support in Healthcare

CAS ADS Module 3 by Mayra Spizzo

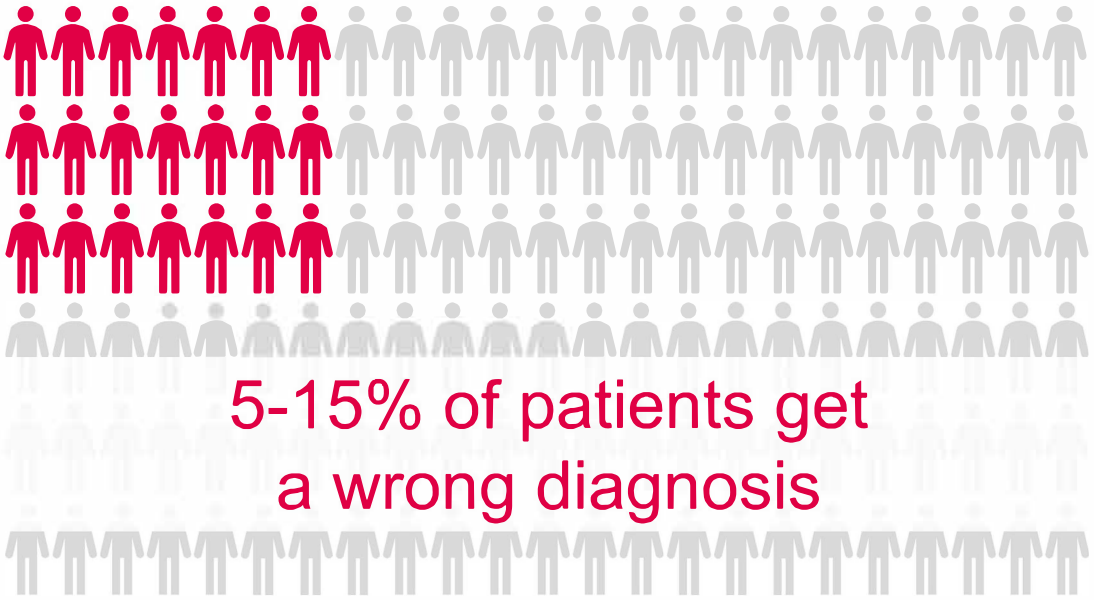
Unravelling the Human Interaction with Generative AI-Based Decision Support in Healthcare:

Types of Chat Users When Having ChatGPT vs. a Human Expert as a Chat Partner



Mayra Spizzo
CAS Applied Data Science, Module 3

Introduction



How can we reduce that number?

- At the same time, AI systems are increasingly implemented in various industries
- Also in the field of medicine, AI-based systems are tested for different purposes

- How could AI be used to reduce the number of incorrect patient diagnoses?
- One approach could be the implementation of a LLM – like ChatGPT

- This leads to the question: What do we know about ChatGPT and its implementation in the diagnostic decision-making process?

- Accuracy of ChatGPT for differential diagnoses is 60.3%
- But at the same time, there is potential in augmenting diagnostic decisions

- Difference to a traditional AI system: Involvement of the human in the process of the recommendation generation

How and why do the interactions with generative AI differ from the interactions with a human coach?

Data Collection and Description

- An experiment is conducted in which participants have to solve two patient cases (i.e., generate a diagnosis for a patient). Participants are split into two groups:
  - Group 1: chat with ChatGPT as a support to solve the patient case
  - Group 2: chat with a human expert (i.e., a physician) as a partner to solve the case
- Participation:
  - possible for all medical students at the Charité Medical School in Berlin that are in the fourth year of their studies
  - at least 18 years old
  - have given their written consent for participation
- The medical students are given two diagnostic tasks that are presented in random order. The patient cases in each of the two diagnostic tasks are based on real emergency cases.
- Participants are randomly assigned to one condition (i.e., either to the human expert

or ChatGPT). To solve the diagnostic task, they can chat with their assigned assistant in real time (i.e., a human coach or ChatGPT). During the assignment, all clicks, noted differential diagnoses, and chat interactions are logged with timestamps

In [378...

```
# libraries
import pandas as pd
import matplotlib.pyplot as plt
```

Descriptive Statistics

Based on the data collection process, the following data for each participant is collected:

- duration of the exchange with the expert (for both ChatGPT and human expert)
- number of interactions with respective expert
- categorization of interaction (technical question, request, statement, diagnosis exclusion)
- patient background info reviewed by the med student

In [391…

```
# load data
df = pd.read_csv("Daten_Experiment.csv", sep=';')
df2 = pd.read_csv("Daten_Experiment2.csv", sep=';')
```

In [380…

```
# data set overview
#get variables
n_participants = df2['Count'].nunique()
human_interaction = df2['Condition'].value_counts()['Human']
gpt_interaction = df2['Condition'].value_counts()['GPT']
time_per_patient = df2['TimePatientSeconds'].mean()
time_per_patient_human = df2[df2['Condition'] == 'Human']['TimePatientSeconds'].
time_per_patient_gpt = df2[df2['Condition'] == 'GPT']['TimePatientSeconds'].mean
amount_interaction = df2['AmountInteraction'].mean()
amount_interaction_gpt = df2[df2['Condition'] == 'GPT']['AmountInteraction'].mea
amount_interaction_human = df2[df2['Condition'] == 'Human']['AmountInteraction']
amount_patient_ratio = df2['AmountPatientRatio'].mean()
```

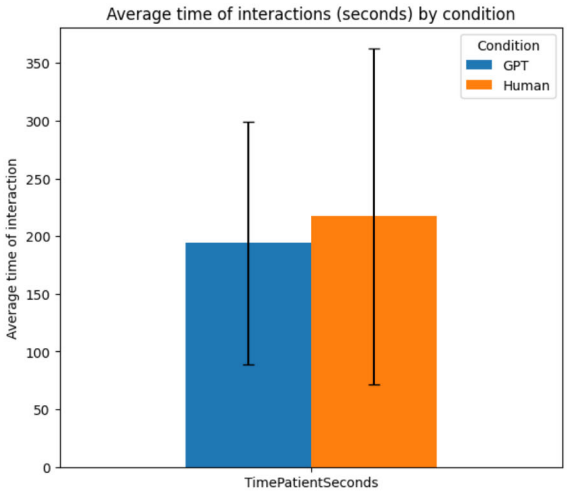
In [381…

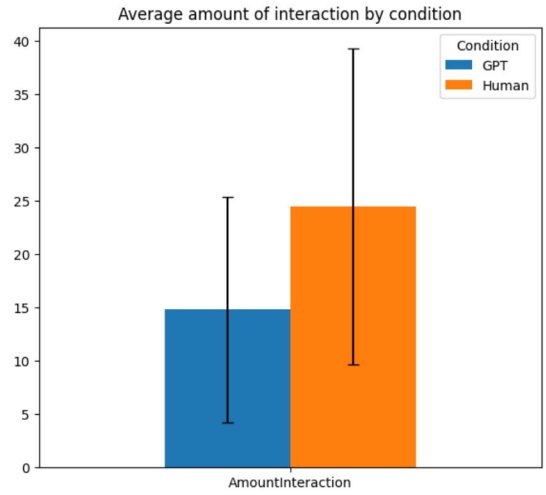
```
from IPython.display import Markdown, display

table_md = f"""

| Variable              | Observations|
|-----------------------|-------|
|Number of participants | {n_participants} |
|Human interactions | {human_interaction} |
|GPT interactions| {gpt_interaction} |
|Average time per patient (seconds)| {round(time_per_patient)} |
|Average time per patient - Human (seconds)| {round(time_per_patient_human)} |
|Average time per patient - GPT (seconds)| {round(time_per_patient_gpt)} |
|Average amount of interaction| {round(amount_interaction)} |
|Average amount of interaction - Human| {round(amount_interaction_human)} |
|Average amount of interaction GPT | {round(amount_interaction_gpt)} |
|Patient information viewed by medical student | {round(amount_patient_ratio,2)}

"""
# Display Table
display(Markdown(table_md))
```

Variable	Observations
Number of participants	114
Human interactions	54
GPT interactions	60
Average time per patient (seconds)	205
Average time per patient - Human (seconds)	217
Average time per patient - GPT (seconds)	194
Average amount of interaction	19
Average amount of interaction - Human	24
Average amount of interaction GPT	15
Patient information viewed by medical student	0.58

In [382…

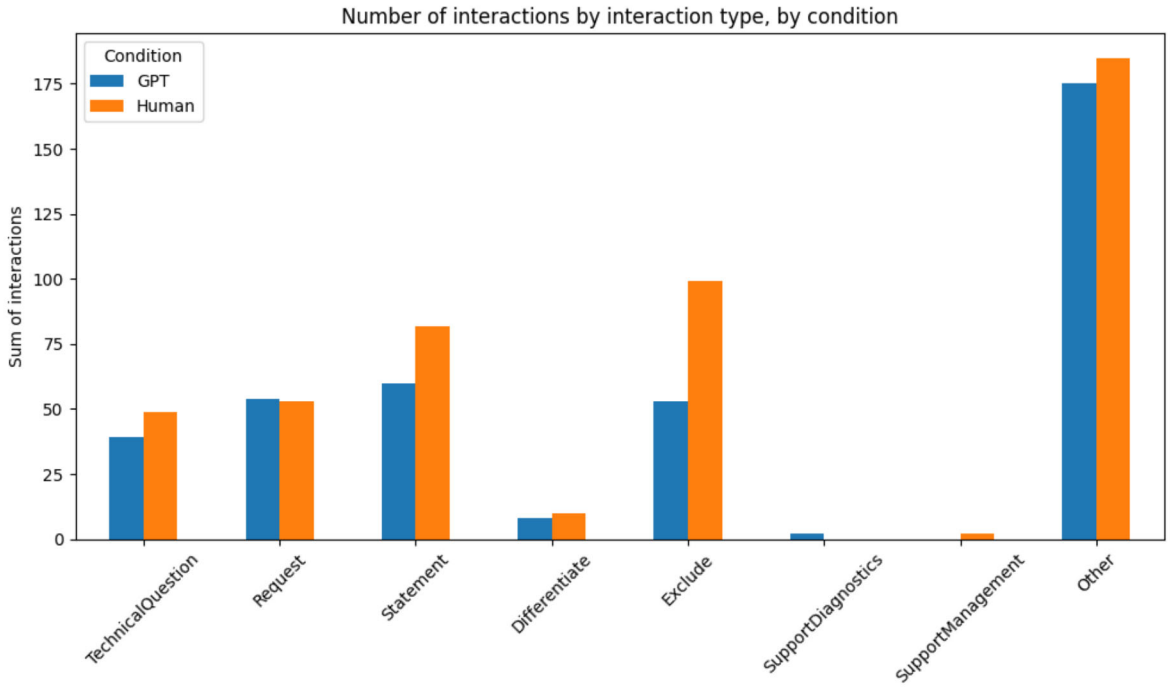
```
df = pd.DataFrame(df2)

# Selecting columns and computing the means and standard deviations
selected_columns_time = ['TimePatientSeconds', 'Condition']  # Replace with desi
selected_columns_interaction = ['AmountInteraction', 'Condition']  # Replace wit

average_time = df[selected_columns_time]
average_interaction = df[selected_columns_interaction]

# Group by 'Condition' and calculate mean and standard deviation
grouped_average_time = average_time.groupby('Condition').mean()
grouped_std_time = average_time.groupby('Condition').std()  # Standard deviation

grouped_average_interaction = average_interaction.groupby('Condition').mean()
grouped_std_interaction = average_interaction.groupby('Condition').std()  # Stan

# Transpose the data to have categories on x-axis
grouped_average_time = grouped_average_time.T
grouped_std_time = grouped_std_time.T  # Transpose standard deviation for time

grouped_average_interaction = grouped_average_interaction.T
grouped_std_interaction = grouped_std_interaction.T  # Transpose standard deviat

# Plotting side by side with error bars
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# First plot: Average time of interactions with error bars
grouped_average_time.plot(kind='bar', yerr=grouped_std_time, ax=axes[0], capsize
axes[0].set_title('Average time of interactions (seconds) by condition')
axes[0].set_xlabel(' ')
axes[0].set_ylabel('Average time of interaction')
axes[0].tick_params(axis='x', rotation=0)
axes[0].legend(title='Condition')

# Second plot: Average amount of interaction with error bars
grouped_average_interaction.plot(kind='bar', yerr=grouped_std_interaction, ax=ax
axes[1].set_title('Average amount of interaction by condition')
axes[1].set_xlabel(' ')
axes[1].tick_params(axis='x', rotation=0)
```





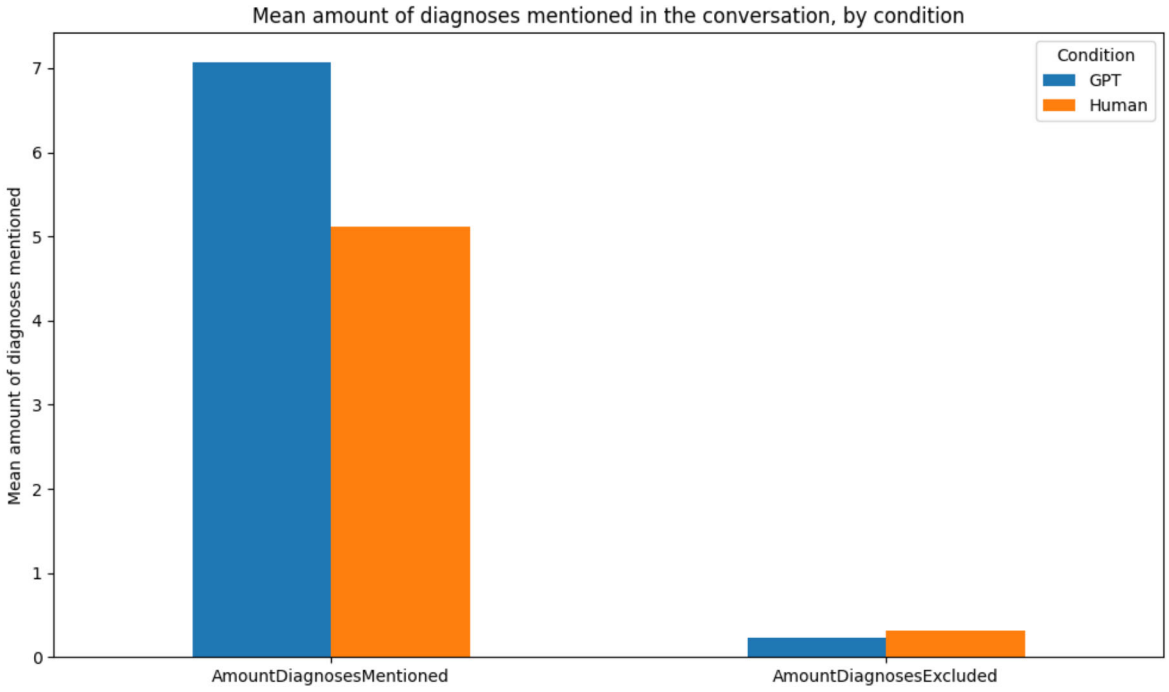
In [383…

```
df = pd.DataFrame(df2)
selected_columns = ['Condition','TechnicalQuestion', 'Request', 'Statement', 'Di

sum_replies = df[selected_columns]

grouped_sum = sum_replies.groupby('Condition').sum()
# Transpose the data to have categories on x-axis and genders as separate bars
grouped_sum = grouped_sum.T  # Transpose so categories are on x-axis

# Plotting
grouped_sum.plot(kind='bar', figsize=(10, 6))

# Customize the plot
plt.title('Number of interactions by interaction type, by condition')
plt.xlabel('')
plt.ylabel('Sum of interactions')
plt.xticks(rotation=45)
plt.legend(title='Condition')
plt.tight_layout()

# Show the plot
plt.show()
```



In [385…

```
selected_columns = ['Condition','AmountDiagnosesMentioned', 'AmountDiagnosesExcl

sum_replies = df[selected_columns]
grouped_sum = sum_replies.groupby('Condition').mean()
# Transpose the data to have categories on x-axis and genders as separate bars
grouped_sum = grouped_sum.T  # Transpose so categories are on x-axis

# Plotting
grouped_sum.plot(kind='bar', figsize=(10, 6))

# Customize the plot
plt.title('Mean amount of diagnoses mentioned in the conversation, by condition'
plt.xlabel('')
plt.ylabel('Mean amount of diagnoses mentioned')
plt.xticks(rotation=0)
plt.legend(title='Condition')
plt.tight_layout()

# Show the plot
plt.show()
```



Unsupervised Machine Learning Methods

K-means

In [386…

```
#libraries
import numpy as np
from sklearn.cluster import KMeans
```

In [392…

```
#Create a dataframe with the necessary variables
clustering_df = df[['Count', 'Condition', 'AmountPatientRatio', 'TimePatientSeco

#Create a binary column for the variable 'Condition'
#1 = ChatGPT, 0 = Human expert
clustering_df['Condition'] = clustering_df['Condition'].apply(lambda x: 1 if x =
```

```
C:\Users\ms20s658\AppData\Local\Temp\ipykernel_31912\144954572.py:6: SettingWithC
opyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  clustering_df['Condition'] = clustering_df['Condition'].apply(lambda x: 1 if x
== "GPT" else 0)
```

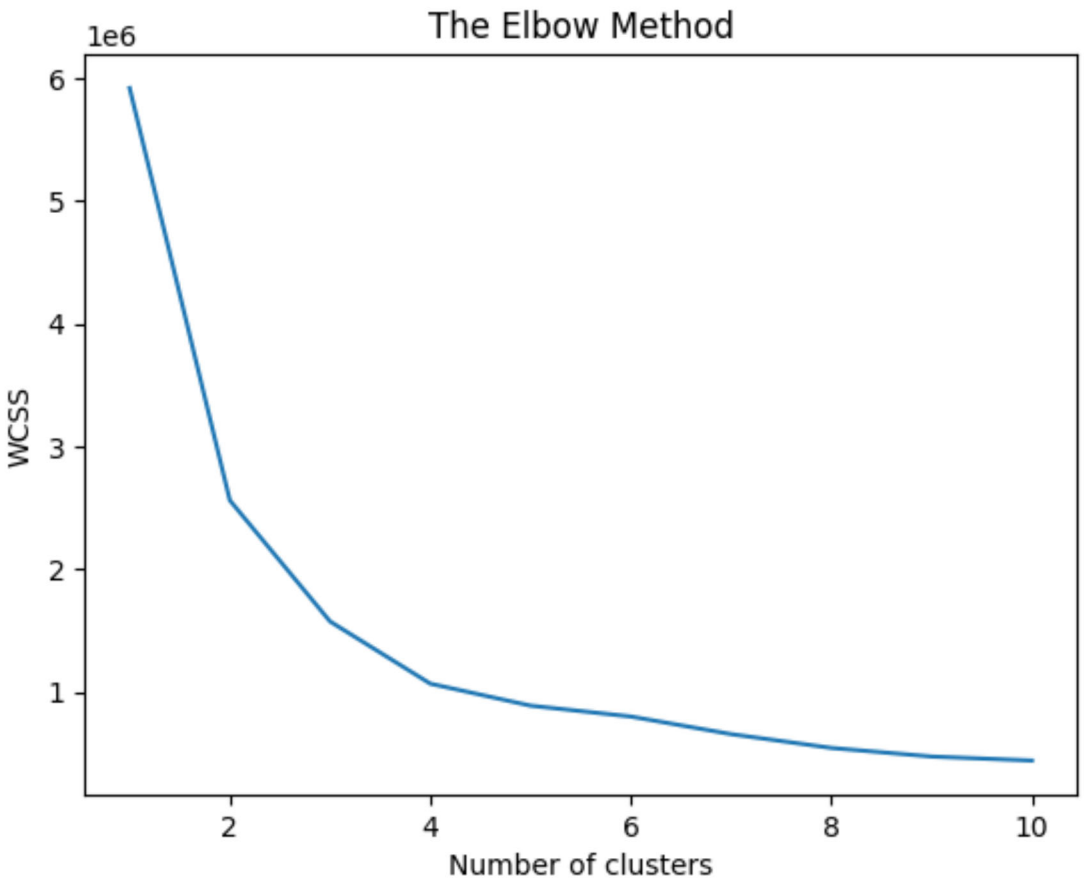
In [393…
```python
#Create two subsets for each condition
clustering_gpt = clustering_df[clustering_df['Condition'] == 1]
clustering_human = clustering_df[clustering_df['Condition'] == 0]

#Check if it worked
#clustering_gpt.head()
#clustering_human.head()
```

Apply a PCA to reduce the dimensionality:

In [394…
```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(clustering_df)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

Use of the elbow method to find the optimal number of clusters:

In [395…
```python
# K-Means++ initialization for the elbow method:
# Instead of randomly placing centroids, K-Means++ selects them strategically:
# 1. The first centroid is chosen randomly from the data points.
# 2. For each remaining centroid, the algorithm selects a point farthest from th
#    This spreads out the centroids across the dataset.
# 3. After initializing the centroids, K-Means proceeds with clustering as usual
# K-Means++ improves the clustering efficiency and helps get better results when
# like the Elbow Method to find the optimal number of clusters.
wcss = []
for i in range(1, 11): # we'll do it for 10 clusters to find the optimal number
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(clustering_df)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



The optimal number of clusters is **4**. We can see visually that with a number of clusters of 4, the descent of the WCSS value is reducing its descent.

Let's now train the model with this number of clusters:

In [396…
```
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(clustering_df)
```

Explanation of `y_kmeans` :

The `y_kmeans` variable contains the classification in a cluster of the participant / user.

For example, participant 1 is in cluster 2, participant 2 is in cluster 1 and participant 3 is in cluster 2.

In [397…
```
#print(clustering_df.head())
y_kmeans
```

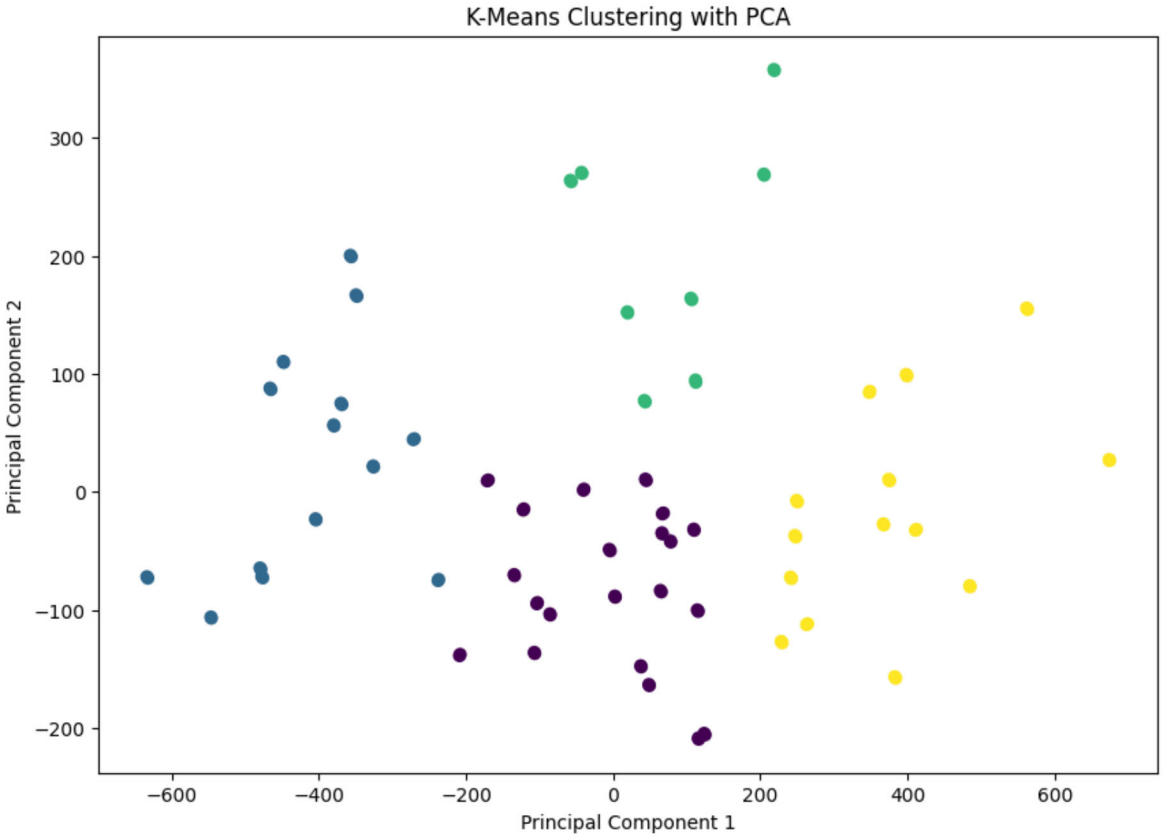
Out[397…
```
array([1, 0, 1, 1, 2, 1, 2, 1, 0, 3, 3, 3, 2, 2, 3, 3, 0, 2, 2, 1, 3, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1,
       2, 3, 2, 3, 2, 1, 2, 2, 0, 1, 1, 2, 0], dtype=int32)
```

Visualization of the results

In a next step, let's visualize the clusters:

In [37]:
```
labels = kmeans.labels_

plt.figure(figsize=(10, 7))
```

```
plt.scatter(principalDf['principal component 1'], principalDf['principal compone
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-Means Clustering with PCA')
plt.show()
```



In [38]:
```
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(clustering_df, labels)
print("Silhouette Score:", silhouette_avg)
```

```
Silhouette Score: 0.46693286310598936
```

To further understand and interpret the clusters, let's have a look at the distribution of the values for each cluster:

In [399…
```
# Group the data by cluster and calculate the mean of each feature
clustering_df.loc[:, 'cluster'] = y_kmeans
df_grouped = clustering_df.groupby('cluster').mean()

#Filter the DataFrame for each cluster
cluster1_data = clustering_df[clustering_df['cluster'] == 0]
cluster2_data = clustering_df[clustering_df['cluster'] == 1]
cluster3_data = clustering_df[clustering_df['cluster'] == 2]
cluster4_data = clustering_df[clustering_df['cluster'] == 3]

#Calculate the mean of each feature for each cluster
cluster1_mean = cluster1_data.mean()
cluster2_mean = cluster2_data.mean()
cluster3_mean = cluster3_data.mean()
cluster4_mean = cluster4_data.mean()

# Create a Markdown table string
table_md1 = f"""
```

```
| Feature | Mean value for cluster 1 | Mean value for cluster 2 | Mean value for
|---|---|---|---|
| Condition | **{cluster1_mean['Condition']:.2f}** | {cluster2_mean['Condition']
| Ratio of patient information | {cluster1_mean['AmountPatientRatio']:.2f} | **{
| Time of patient information (in seconds) | {cluster1_mean['TimePatientSeconds'
| Q-type: Technical question | {cluster1_mean['TechnicalQuestion']:.2f} | {clust
| Q-type: Request | {cluster1_mean['Request']:.2f} | **{cluster2_mean['Request']
| Q-type: Statement | {cluster1_mean['Statement']:.2f} | {cluster2_mean['Stateme
| Q-type: Differentiate | {cluster1_mean['Differentiate']:.2f} | {cluster2_mean[
| Q-type: Exclude | {cluster1_mean['Exclude']:.2f} | {cluster2_mean['Exclude']:.
| First question was self-formulated (and not copied) | {cluster1_mean['FirstQue
| Amount of mentioned diagnoses | **{cluster1_mean['AmountDiagnosesMentioned']:.
| Duration of the total chat interaction (in seconds) | **{cluster1_mean['Durati
| Amount of interactions in the chat | **{cluster1_mean['AmountInteraction']:.2f
"""

# Display Table
display(Markdown(table_md1))
```

Feature	Mean value for cluster 1	Mean value for cluster 2	Mean value for cluster 3	Mean value for cluster 4
Condition	**0.79**	0.57	**0.29**	0.38
Ratio of patient information	0.64	**0.41**	0.62	**0.83**
Time of patient information (in seconds)	201.00	**123.48**	211.36	**415.25**
Q-type: Technical question	0.50	0.76	**1.14**	0.25
Q-type: Request	0.71	**1.19**	1.07	0.38
Q-type: Statement	0.14	1.86	**2.00**	0.62
Q-type: Differentiate	0.07	0.19	0.14	**0.25**
Q-type: Exclude	0.36	1.67	**2.14**	0.50
First question was self-formulated (and not copied)	0.71	0.76	0.79	0.75
Amount of mentioned diagnoses	**3.14**	**8.00**	7.50	4.25
Duration of the total chat interaction (in seconds)	**360.14**	771.67	**1145.00**	832.12
Amount of interactions in the chat	**4.71**	25.48	**28.07**	13.62

Interpretation

- Cluster 1 - Quickies (or low intensity chat user)
  - The group that did use the chat the least (lowest duration of chat interaction and amount of chat interactions among all clusters)
  - Subsequently, the group that generated the lowest amount of differential

diagnoses

- Cluster 2 - Generate me solutions, please! (or low informer and high requester)
  - The group that contains users who did not inform themselves before interacting in the chat (lowest values for ratio and time of patient information acquisition prior to entering the chat)
  - During the interaction they request the generation of diagnoses (highest value for Q-type: Request) to have a palette of diagnoses (highest amount of generated diagnoses among all clusters)
- Cluster 3 - Novice chatterbox (or high intensity chat user with low expert knowledge in the medical field)
  - The group that did use the chat the most (highest duration of chat interaction and amount of chat interactions among all clusters)
  - In the chat interaction, they mostly ask technical questions to close their knowledge gap (highest value for Q-type: Technical question). They also make the most statements (highest value for Q-type: Statement), but this is probably because of additional information or the interaction for the technical questions.
  - They also asked the most questions to exclude diagnoses (highest value for Q-type: Exclude). The information gained from the technical questions could possibly be used to then exclude diagnoses they thought of.
- Cluster 4 - Help me to get my ideas straight (or high informer and high differentiater)
  - The group with users who did inform themselves a lot about the patient case before interacting in the chat (highest values for ratio and time of patient information acquisition prior to entering the chat)
  - With this, they built themselves an image about the situation and had already a list of diagnoses in mind. Then, during the chat interaction, they asked questions to differentiate multiple diagnoses from each other (highest value for Q-type: Differentiate)

Condition ChatGPT vs. Human expert

- It is noticeable that cluster 1 and 2 contain participants who had in a majority ChatGPT as their interaction partner and cluster 3 and 4, on the other hand, contain mostly participants with a human expert as their chat partner.
- The clusters seem to differentiate between the conditions. Therefore, it makes sense to do an additonal clustering for each condition separately and to compare the results.

K-means for the condition ChatGPT

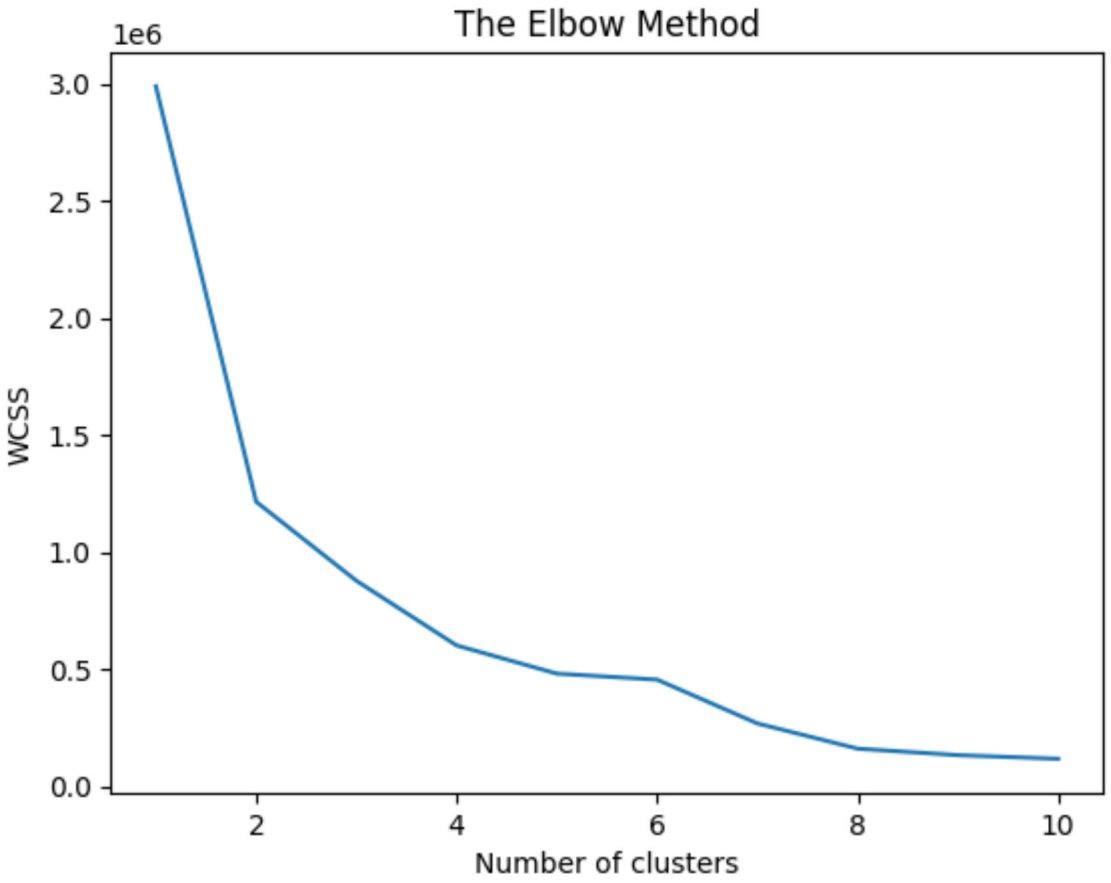
PCA to reduce the dimensionality:

In [40]:
```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(clustering_gpt)
principalDf = pd.DataFrame(data = principalComponents
```

```
                , columns = ['principal component 1', 'principal component 2'])
```

Use of the elbow method to find the optimal number of clusters:

In [400…
```
wcss = []
for i in range(1, 11): # we'll do it for 10 clusters to find the optimal number
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(clustering_gpt)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



The optimal number of clusters is **4**.

Train the model with this number of clusters:

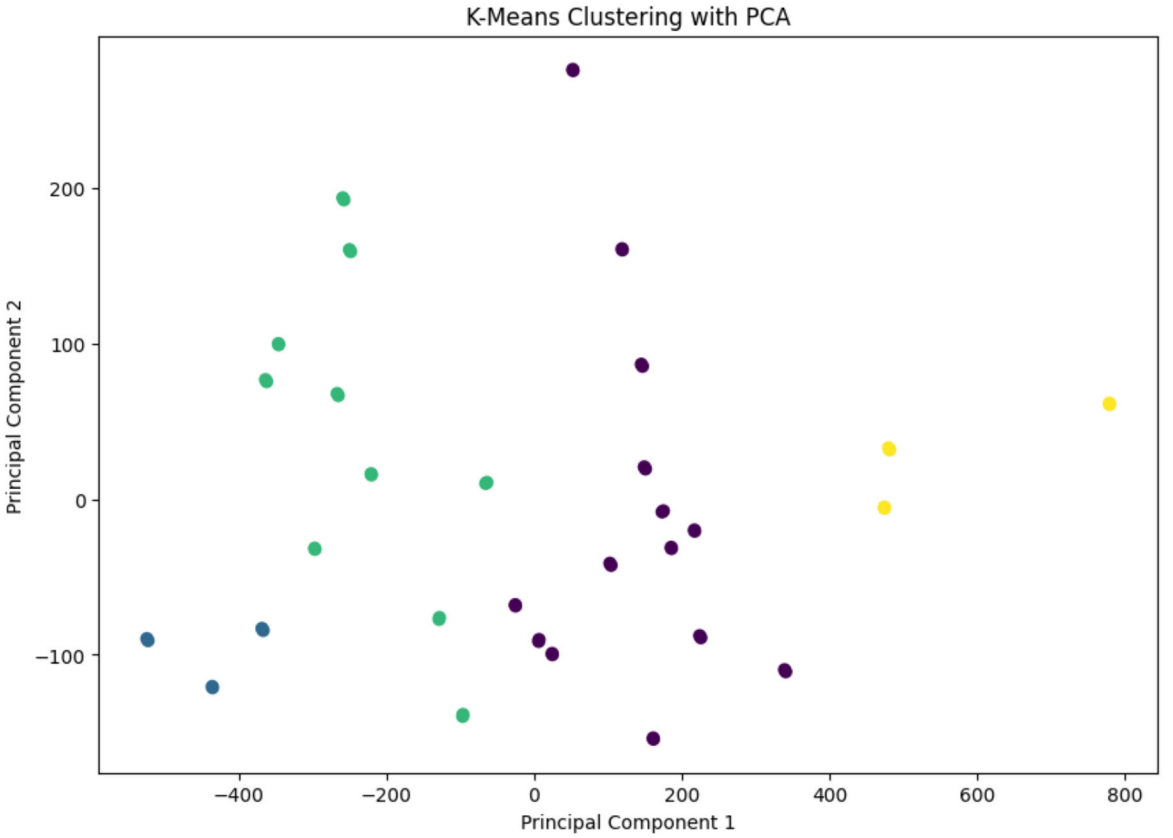
In [401…
```
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(clustering_gpt)
```

Visualization of the results

Visualization of the clusters:

In [43]:
```
labels = kmeans.labels_

plt.figure(figsize=(10, 7))
plt.scatter(principalDf['principal component 1'], principalDf['principal compone
```

```
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-Means Clustering with PCA')
plt.show()
```



In [44]:
```
silhouette_avg = silhouette_score(clustering_gpt, labels)
print("Silhouette Score:", silhouette_avg)
```

```
Silhouette Score: 0.41596230451124055
```

Distribution of the values for each cluster:

In [403…
```
# Group the data by cluster and calculate the mean of each feature
clustering_gpt.loc[:, 'cluster'] = y_kmeans
df_grouped = clustering_gpt.groupby('cluster').mean()

#Filter the DataFrame for each cluster
cluster1_data = clustering_gpt[clustering_gpt['cluster'] == 0]
cluster2_data = clustering_gpt[clustering_gpt['cluster'] == 1]
cluster3_data = clustering_gpt[clustering_gpt['cluster'] == 2]
cluster4_data = clustering_gpt[clustering_gpt['cluster'] == 3]

#Calculate the mean of each feature for each cluster
cluster1_mean = cluster1_data.mean()
cluster2_mean = cluster2_data.mean()
cluster3_mean = cluster3_data.mean()
cluster4_mean = cluster4_data.mean()

# Create a Markdown table string
table_md1 = f"""
| Feature | Mean value for cluster 1 | Mean value for cluster 2 | Mean value for
|---|---|---|---|---|
| Ratio of patient information | {cluster1_mean['AmountPatientRatio']:.2f} | {cl
| Time of patient information (in seconds) | {cluster1_mean['TimePatientSeconds'
```

```
| Q-type: Technical question | {cluster1_mean['TechnicalQuestion']:.2f} | {clust
| Q-type: Request | {cluster1_mean['Request']:.2f} | {cluster2_mean['Request']:.
| Q-type: Statement | {cluster1_mean['Statement']:.2f} | {cluster2_mean['Stateme
| Q-type: Differentiate | {cluster1_mean['Differentiate']:.2f} | **{cluster2_mea
| Q-type: Exclude | {cluster1_mean['Exclude']:.2f} | {cluster2_mean['Exclude']:.
| First question was self-formulated (and not copied) | {cluster1_mean['FirstQue
| Amount of mentioned diagnoses | **{cluster1_mean['AmountDiagnosesMentioned']:.
| Duration of the total chat interaction (in seconds) | **{cluster1_mean['Durati
| Amount of interactions in the chat | **{cluster1_mean['AmountInteraction']:.2f
"""

# Display Table
display(Markdown(table_md1))
```

Feature	Mean value for cluster 1	Mean value for cluster 2	Mean value for cluster 3	Mean value for cluster 4
Ratio of patient information	0.68	0.71	**0.40**	**1.00**
Time of patient information (in seconds)	212.90	214.08	**121.29**	**278.00**
Q-type: Technical question	0.60	0.58	**0.71**	0.00
Q-type: Request	0.80	0.92	**1.14**	0.00
Q-type: Statement	0.10	1.92	0.71	**3.00**
Q-type: Differentiate	0.10	**0.17**	0.14	0.00
Q-type: Exclude	0.20	1.08	**1.43**	1.00
First question was self-formulated (and not copied)	0.60	0.67	0.71	1.00
Amount of mentioned diagnoses	**3.60**	9.50	7.57	**10.00**
Duration of the total chat interaction (in seconds)	**331.90**	891.58	641.71	**1442.00**
Amount of interactions in the chat	**4.30**	22.33	15.14	**26.00**

Interpretation

- Cluster 1 - Low performer
  - The group that did use the chat the least (lowest duration of chat interaction and amount of chat interactions among all clusters) and that generated the lowest amount of diagnoses
- Cluster 2 - Differentiater
  - The group that asked the most questions to differentiate between two or more differential diagnoses (highest values for Q-type: Differentiate)
  - Otherwise they are average users / perfomers (no peak values in the other categories)

- Cluster 3 - I know nothing, please do everyting for me! (or low informer and high requester as well as excluder)
    - The group that contains users who did not inform themselves before interacting in the chat (lowest values for ratio and time of patient information acquisition prior to entering the chat)
    - In the chat interaction, they theny mostly ask technical questions to close their knowledge gap (highest value for Q-type: Technical question). They also ask the most requests to ChatGPT to generate diagnoses (highest value for Q-type: Request), and additionally ask the most questions to exclude diagnoses (highest value for Q-type: Exclude). Basically, they ask ChatGPT to generate them diagnoses and then want ChatGPT to reduce the amount of target diagnoes by excluding differential diagnoses.
    - This cluster is to some extent similar to cluster 2 in the overall model
- Cluster 4 - High performer
    - The group that did inform themselves the most (highest values for patient information acquisition variables), use the chat the most (highest duration of chat interaction and amount of chat interactions among all clusters), and also generated the highest amount of diagnoses
    - In addition, they made the most statements during the chat interaction (highest value for Q-type: Statement)

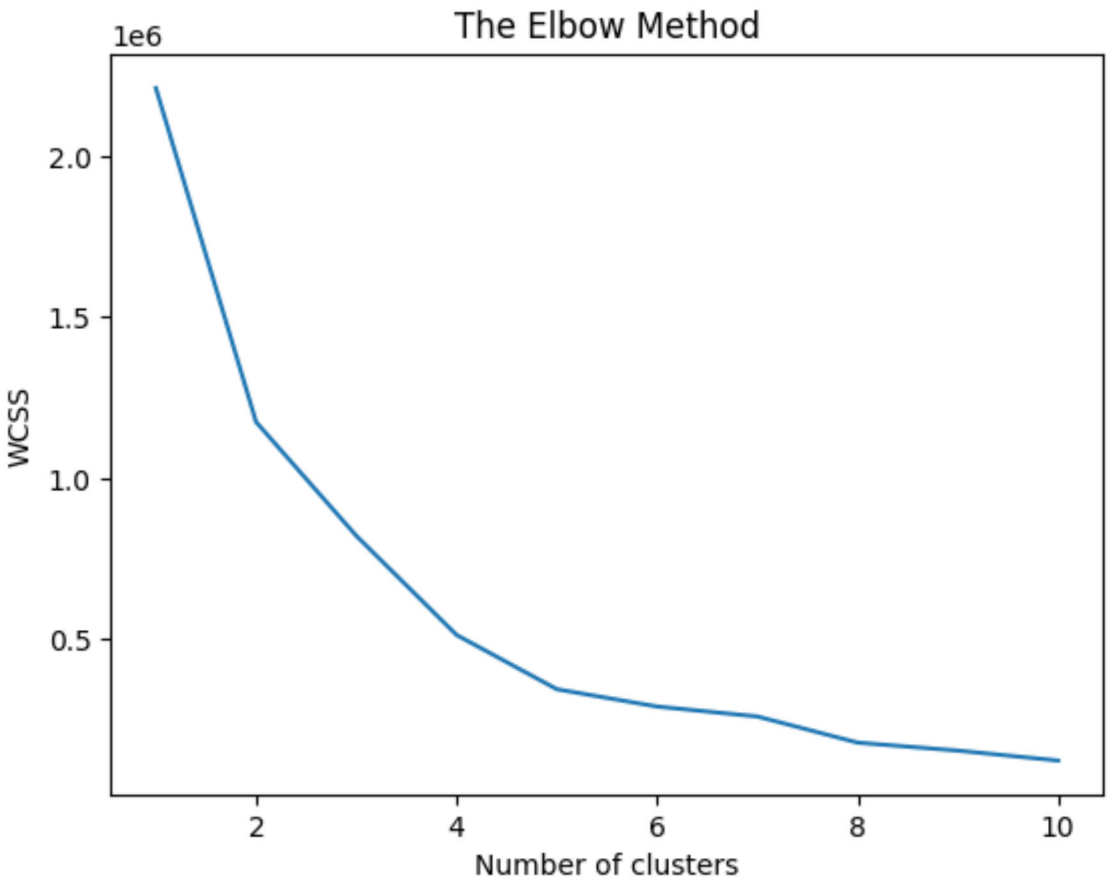
K-means for the condition Human expert

PCA to reduce the dimensionality:

In [46]:
```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(clustering_human)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

Use of the elbow method to find the optimal number of clusters:

In [404…
```
wcss = []
for i in range(1, 11): # we'll do it for 10 clusters to find the optimal number
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(clustering_human)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



The optimal number of clusters is **5**.

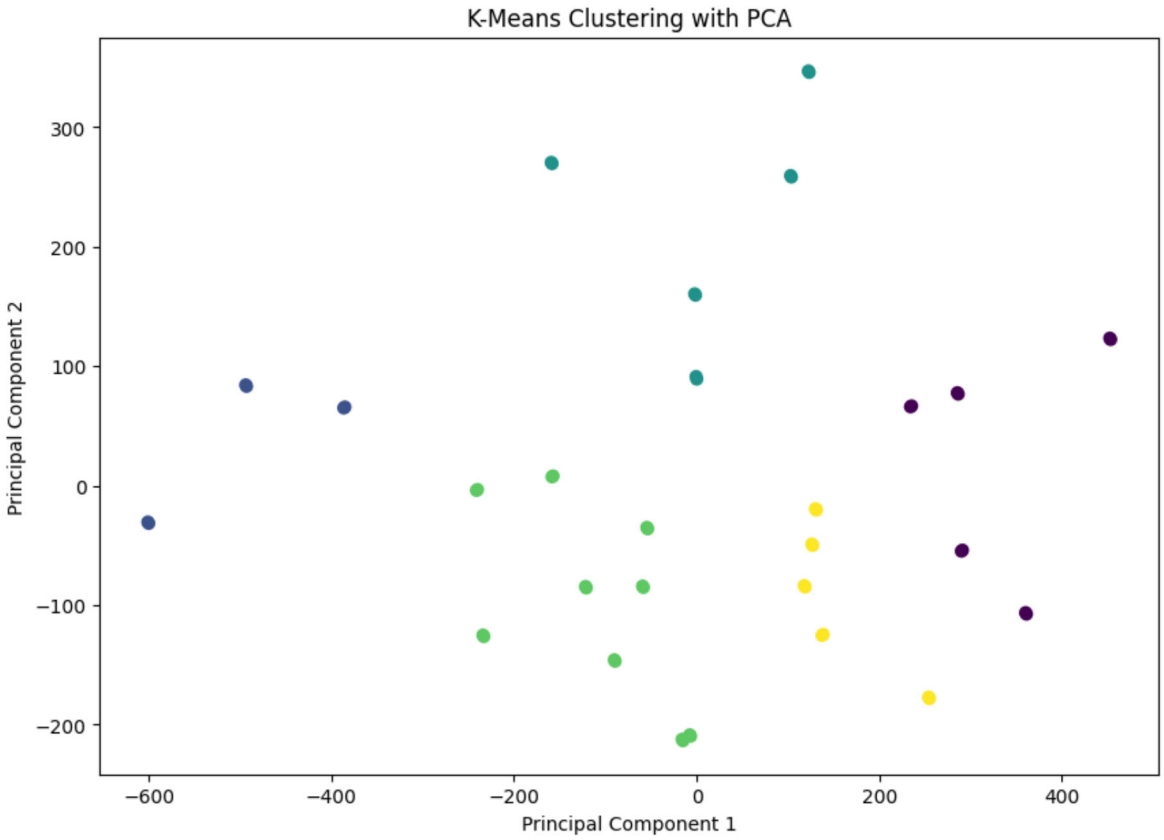
Train the model with this number of clusters:

In [405…
```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(clustering_human)
```

Visualization of the results

Visualization of the clusters:

In [49]:
```
labels = kmeans.labels_

plt.figure(figsize=(10, 7))
plt.scatter(principalDf['principal component 1'], principalDf['principal compone
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-Means Clustering with PCA')
plt.show()
```



In [50]:
```
silhouette_avg = silhouette_score(clustering_human, labels)
print("Silhouette Score:", silhouette_avg)
```

```
Silhouette Score: 0.4247941586141835
```

Distribution of the values for each cluster:

In [407…
```
# Group the data by cluster and calculate the mean of each feature
clustering_human.loc[:, 'cluster'] = y_kmeans
df_grouped = clustering_human.groupby('cluster').mean()

#Filter the DataFrame for each cluster
cluster1_data = clustering_human[clustering_human['cluster'] == 0]
cluster2_data = clustering_human[clustering_human['cluster'] == 1]
cluster3_data = clustering_human[clustering_human['cluster'] == 2]
cluster4_data = clustering_human[clustering_human['cluster'] == 3]
cluster5_data = clustering_human[clustering_human['cluster'] == 4]

#Calculate the mean of each feature for each cluster
cluster1_mean = cluster1_data.mean()
cluster2_mean = cluster2_data.mean()
cluster3_mean = cluster3_data.mean()
cluster4_mean = cluster4_data.mean()
cluster5_mean = cluster5_data.mean()

# Create a Markdown table string
table_md1 = f"""
| Feature | Mean value for cluster 1 | Mean value for cluster 2 | Mean value for
|---|---|---|---|---|---|
| Ratio of patient information | {cluster1_mean['AmountPatientRatio']:.2f} | **{
| Time of patient information (in seconds) | {cluster1_mean['TimePatientSeconds'
| Q-type: Technical question | {cluster1_mean['TechnicalQuestion']:.2f} | {clust
| Q-type: Request | **{cluster1_mean['Request']:.2f}** | {cluster2_mean['Request
| Q-type: Statement | {cluster1_mean['Statement']:.2f} | {cluster2_mean['Stateme
```

```
| Q-type: Differentiate | {cluster1_mean['Differentiate']:.2f} | {cluster2_mean[
| Q-type: Exclude | **{cluster1_mean['Exclude']:.2f}** | {cluster2_mean['Exclude
| First question was self-formulated (and not copied) | {cluster1_mean['FirstQue
| Amount of mentioned diagnoses | **{cluster1_mean['AmountDiagnosesMentioned']:.
| Duration of the total chat interaction (in seconds) | {cluster1_mean['Duration
| Amount of interactions in the chat | {cluster1_mean['AmountInteraction']:.2f}
"""

# Display Table
display(Markdown(table_md1))
```

Feature	Mean value for cluster 1	Mean value for cluster 2	Mean value for cluster 3	Mean value for cluster 4	Mean value for cluster 5
Ratio of patient information	0.50	**0.33**	**0.87**	0.61	0.39
Time of patient information (in seconds)	156.71	**104.56**	**441.60**	348.67	190.33
Q-type: Technical question	1.00	1.00	**0.40**	**1.67**	0.00
Q-type: Request	**1.43**	1.00	0.20	1.33	0.67
Q-type: Statement	2.14	1.78	0.60	**2.33**	0.33
Q-type: Differentiate	0.29	0.11	**0.40**	0.00	0.00
Q-type: Exclude	**3.29**	2.11	0.20	1.00	0.67
First question was self-formulated (and not copied)	0.71	0.78	1.00	1.00	1.00
Amount of mentioned diagnoses	**7.29**	5.44	3.20	5.33	**2.00**
Duration of the total chat interaction (in seconds)	1100.43	793.67	871.80	**1198.00**	**394.33**
Amount of interactions in the chat	31.14	**31.44**	12.20	27.67	**5.00**

Interpretation

- Cluster 1 - Requester and Excluder
  - The group that requested the most differential diagnoses (highest value for Q-type: Request) to the human and at the same time also asked them questions to exclude diagnoses (highest value for Q-type: Exclude)
  - This group is similar to cluster 3 of the ChatGPT model and similar to cluster 2 of the overall model
- Cluster 2 - Low informer but high interacter

    - The group that informed themselves the least about the patient case prior to initiating the chat interaction (lowest values for ratio and time of patient information acquisition prior to entering the chat)
    - But they then wrote the most messages to the human expert (highest amount of chat interactions among all clusters)
- Cluster 3 - High informer, experts and differentiater
    - The group that contains users who did inform themselves thoroughly before interacting in the chat (higest values for patient information acquisition variables)
    - They did not only know the most about the patient case, but also about the medical field in general - since they asked the least technical questions in the chat. This could be an indication for a group of people who like to internalize knowledge, being it on the patient case or on medical knowledge in general.
    - With their expert knowledge, they mostly asked questions in the chat that helped them to differentiate existing ideas and diagnoses they generated themselves (highest value for Q-type: Differentiate)
- Cluster 4 - Encyclopedia
    - People in this group asked mostly technical questions to close their knowledge gap (highest value for Q-type: Technical question). They also make the most statements (highest value for Q-type: Statement). This is probably because of additional information or the interaction for the technical questions.
    - Connected to this, people in this cluster had the longest chat interaction (longest duration of chat interaction in seconds), even tough they did not write the most messages back and forth (not the highest amount of chat interactions). This makes sense, since asking technical questions (as well as reading/ understanding the answers) and giving information for them requires some time.
    - This cluster is similar to cluster 3 of the overall model.
- Cluster 5 - Low performer
    - This group shows the lowest values for the chat interaction variables (i.e., duration and amount of chat interactions) and in addition has the lowest amount of generated diagnoses.
    - Therefore, people in this group did use the chat the least.
    - With this, the cluster is similar to cluster 1 of the ChatGPT model.

Conclusion

- Some clusters regarding the human expert are similar to ChatGPT (e.g., cluster "Please do everything for me"), but some others are different (e.g., cluster 2 - 4).
- There seems to be some influence of the chat partner when it comes to different groups of chat users. But at the same time, some groups of users are consistent, even with differing chat partner.
- Further data can help to support this assumption. That additional data will be available and provided in the near future.

Further models

For the final project, further models, additional to the K-means models, will be tested. To find the right models to cluster the final data sample, the following illustration can be used as a support. The illustration shows how different techniques and models create clusters differently. With this, the optimal models to implement can be chosen, according to the distribution of the data points after the PCA.

Possible further models for our data could for example be the DBSCAN or the Gaussian Mixture.