In [1]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:

```python
# Part 1: Introduction
# - This project demonstrates the application of Principal Component Analysis (PCA) on a dataset.
# - PCA is used to reduce the dimensionality of the data while retaining most of the variance.
# - We will visualize the results of PCA in both 2D and 3D scatter plots.

# Teil 1: Einführung
# - Dieses Projekt zeigt die Anwendung der Hauptkomponentenanalyse (PCA) auf einem Datensatz.
# - PCA wird verwendet, um die Dimensionalität der Daten zu reduzieren und dabei den größten Teil der Varianz beizubehalten.
# - Wir werden die Ergebnisse von PCA sowohl in 2D- als auch in 3D-Streudiagrammen visualisieren.
```

In [3]:

```python
# Part 2: Data Loading
# - Load the dataset from a CSV file.

# Teil 2: Laden der Daten
# - Laden Sie den Datensatz aus einer CSV-Datei.

df = pd.read_csv("[redacted]")
```

In [4]:

```python
df.head()
```

Out[4]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMea
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205	

5 rows × 563 columns

In [5]:

```python
# Part 3: Feature and Target Separation
# - Separate features (`X`) from the target variable (`y`).

# Teil 3: Trennung von Merkmalen und Zielvariablen
# - Trennen Sie die Merkmale (`X`) von der Zielvariablen (`y`).


X = df.drop("subject", axis = 1).drop("Activity", axis = 1)
y = df["Activity"]
```

In [4]:

```python
X
```

Out[4]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-meanFreq()	fBodyBodyGyro. sk
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.074323	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	0.158075	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	0.414503	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	0.404573	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	0.087753	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7347	0.299665	-0.057193	-0.181233	-0.195387	0.039905	0.077078	-0.282301	0.043616	0.060410	0.210795	...	-0.070157	
7348	0.273853	-0.007749	-0.147468	-0.235309	0.004816	0.059280	-0.322552	-0.029456	0.080585	0.117440	...	0.165259	
7349	0.273387	-0.017011	-0.045022	-0.218218	-0.103822	0.274533	-0.304515	-0.098913	0.332584	0.043999	...	0.195034	
7350	0.289654	-0.018843	-0.158281	-0.219139	-0.111412	0.268893	-0.310487	-0.068200	0.319473	0.101702	...	0.013865	
7351	0.351503	-0.012423	-0.203867	-0.269270	-0.087212	0.177404	-0.377404	-0.038678	0.229430	0.269013	...	-0.058402	

7352 rows × 561 columns

In [6]:

```python
X = df.drop("subject", axis = 1).drop("Activity", axis = 1)
y = df["Activity"]

# Part 4: Data Standardization
# - Standardize the features to have zero mean and unit variance.

# Teil 4: Datenstandardisierung
# - Standardisieren Sie die Merkmale, um einen Mittelwert von null und eine Varianz von eins zu erreichen.

from sklearn.preprocessing import StandardScaler
```

```
s = StandardScaler()
X = s.fit_transform(X)
```

In [7]:

```
X
```

Out[7]:

```
array([[ 0.20064157, -0.0636826 , -0.41962845, ..., -0.68721921,
         0.40794614, -0.00756789],
       [ 0.05594788,  0.03148567, -0.25390836, ..., -0.694138  ,
         0.40911698,  0.00787517],
       [ 0.07351535, -0.04341648, -0.07629468, ..., -0.702239  ,
         0.4102883 ,  0.02650234],
       ...,
       [-0.01566765,  0.0167814 ,  1.13222107, ..., -0.56584847,
         0.64059683,  0.34870928],
       [ 0.21586648, -0.02812252, -0.86770988, ..., -0.57766781,
         0.63147758,  0.29327564],
       [ 1.09620157,  0.12919873, -1.67268082, ..., -0.57392691,
         0.63274259,  0.33396081]])
```

In [8]:

```
# Part 5: Applying PCA (2 Components)
# - Apply PCA to reduce the dimensionality to 2 components.

# Teil 5: Anwendung von PCA (2 Komponenten)
# - Wenden Sie PCA an, um die Dimensionalität auf 2 Komponenten zu reduzieren.

from sklearn.decomposition import PCA

p = PCA(n_components = 2)
p.fit(X)

X_transformed = p.transform(X)
```

In [9]:

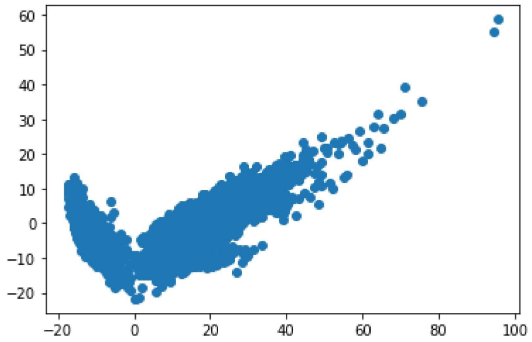
```
X_transformed
```

Out[9]:

```
array([[-16.13854371,   2.15202402],
       [-15.2961943 ,   1.38714381],
       [-15.13701861,   2.47335095],
       ...,
       [ 14.33343587, -12.26071186],
       [ 12.87601895, -14.07125583],
       [ 13.01610365, -12.2442612 ]])
```

In [10]:

```
# Part 6: Visualizing 2D PCA Results
# - Visualize the transformed data in a 2D scatter plot.

# Teil 6: Visualisierung der 2D-PCA-Ergebnisse
# - Visualisieren Sie die transformierten Daten in einem 2D-Streudiagramm.

%matplotlib inline

import matplotlib.pyplot as plt

plt.scatter(X_transformed[:, 0], X_transformed[:, 1])
plt.show()
```



In [14]:

```
y.unique()
```

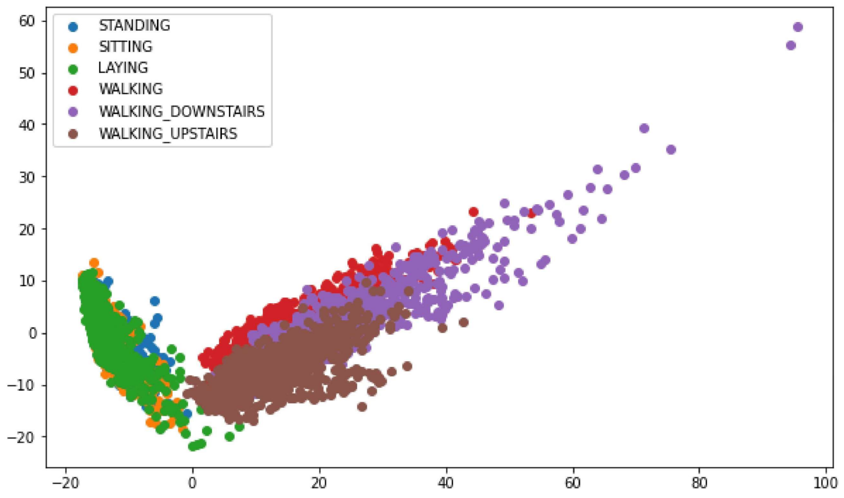
Out[14]:

```
array(['STANDING', 'SITTING', 'LAYING', 'WALKING', 'WALKING_DOWNSTAIRS',
       'WALKING_UPSTAIRS'], dtype=object)
```

In [11]:

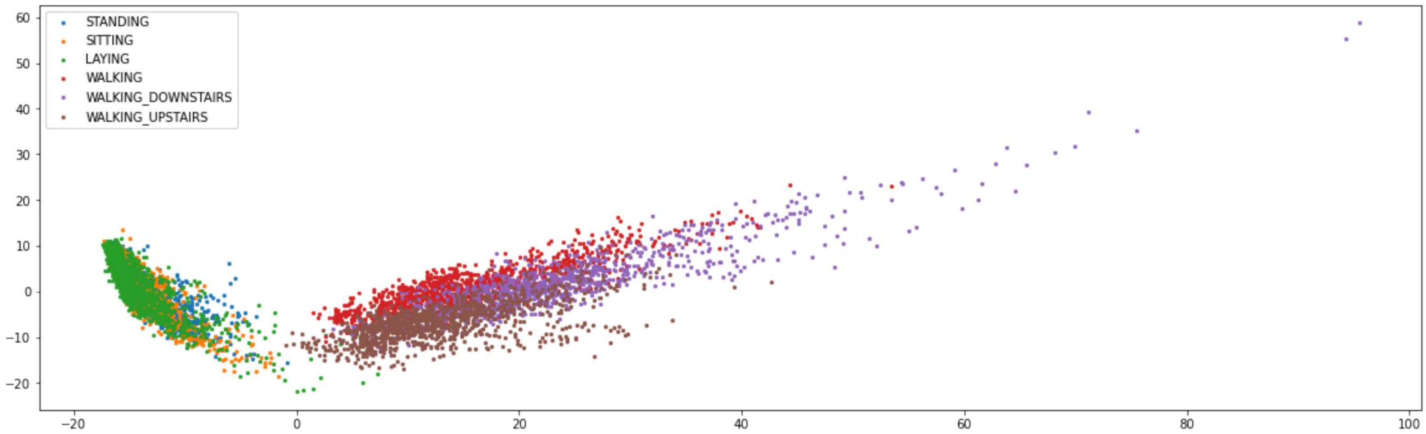
```
# Part 7: Enhanced 2D Visualization
# - Visualize the transformed data in a 2D scatter plot, separated by activity labels.

# Teil 7: Erweiterte 2D-Visualisierung
# - Visualisieren Sie die transformierten Daten in einem 2D-Streudiagramm, getrennt nach Aktivitätslabels.

%matplotlib inline

import matplotlib.pyplot as plt

plt.figure(figsize = (10, 6))

for activity in y.unique():
    X_transformed_filtered = X_transformed[y == activity, :]
    plt.scatter(X_transformed_filtered[:, 0], X_transformed_filtered[:, 1], label = activity)
```

```
plt.legend()
plt.show()
```



In [12]:

```
# Part 9: Applying PCA (3 Components)
# - Apply PCA to reduce the dimensionality to 3 components.

## Anwendung von PCA (3 Komponenten)
# - Wenden Sie PCA an, um die Dimensionalität auf 3 Komponenten zu reduzieren.

%matplotlib inline

import matplotlib.pyplot as plt

plt.figure(figsize = (20, 6))

for activity in y.unique():
    X_transformed_filtered = X_transformed[y == activity, :]
    plt.scatter(X_transformed_filtered[:, 0], X_transformed_filtered[:, 1], label = activity, s = 5)

plt.legend()
plt.show()
```



In [13]:

```
# Part 8: Applying PCA (3 Components)
# - Apply PCA to reduce the dimensionality to 3 components.

# Teil 8: Anwendung von PCA (3 Komponenten)
# - Wenden Sie PCA an, um die Dimensionalität auf 3 Komponenten zu reduzieren.


from sklearn.decomposition import PCA

p = PCA(n_components = 3)
p.fit(X)

X_transformed = p.transform(X)
```

In [14]:

```
X_transformed.shape
```

Out[14]: (7352, 3)

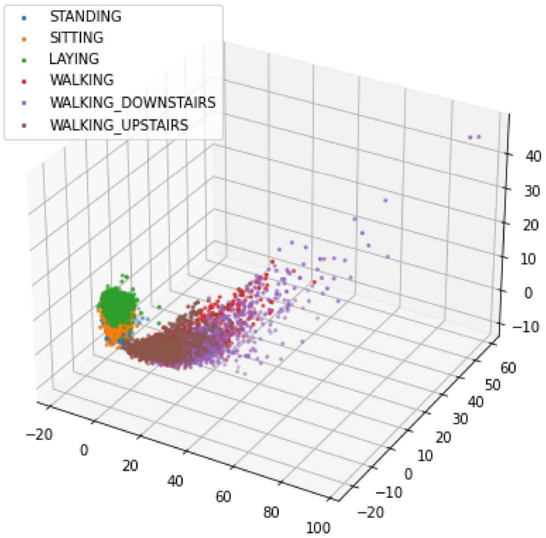
In [15]:

```
# Part 9: Visualizing 3D PCA Results
# - Visualize the transformed data in a 3D scatter plot, separated by activity labels.

# Teil 9: Visualisierung der 3D-PCA-Ergebnisse
# - Visualisieren Sie die transformierten Daten in einem 3D-Streudiagramm, getrennt nach Aktivitätslabels.

%matplotlib inline

import numpy as np
```

```python
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

In [16]:

```python
s = StandardScaler()
X = s.fit_transform(X)

p = PCA(n_components=3)
p.fit(X)
X_transformed_3d = p.transform(X)

fig = plt.figure(figsize=(20, 7))
ax = fig.add_subplot(111, projection='3d')

for activity in y.unique():
    X_transformed_filtered = X_transformed_3d[y == activity, :]
    ax.scatter(
        X_transformed_filtered[:, 0],
        X_transformed_filtered[:, 1],
        X_transformed_filtered[:, 2],
        label=activity,
        s=4
    )

ax.legend()
plt.show()
```



In [17]:

```python
# Part 10: Conclusion
# - PCA is an effective technique for dimensionality reduction and visualization.
# - It helps in understanding the structure of high-dimensional data.

# Teil 10: Fazit
# - PCA ist eine effektive Technik zur Dimensionsreduktion und Visualisierung.
# - Es hilft, die Struktur von hochdimensionalen Daten zu verstehen.
```

In [ ]: