UNSUPERVISED MACHINE LEARNING MODELS TO DETECT ROUTINE QUALITY

PATTERNS THAT INFLUENCE SLEEP

Vania Oliveira (10607174)

INTRODUCTION

Sleep disorders have become increasingly common in today's society. There is much speculation about the excessive use of screens (computers, cell phones, TV), the consumption of substances such as caffeine or even a possible deregulation of melatonin levels in the human body.

In fact, what worries doctors, researchers and people in general most are the serious consequences of poor sleep. Lack of sleep can lead to Cardiovascular Diseases, Diabetes, Obesity, Gastrointestinal Problems, Depression, Anxiety, Cognitive Disorders, Mood Disorders, Premature Aging, among others.

For these reasons, sleep quality care is gaining more and more attention not only in the health context, but within society and in general.

SOME CURRENT ALTERNATIVES

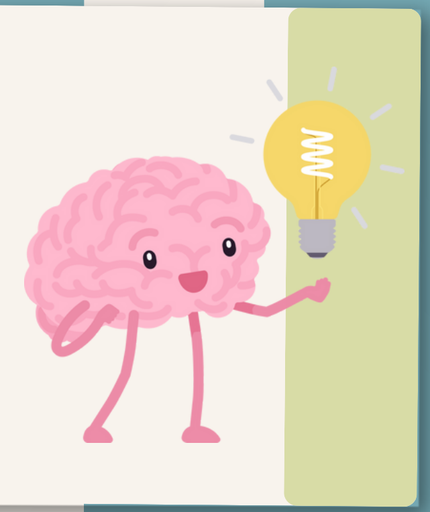
Medicines that promise to help with sleep. However, they are not accepted everywhere and are not recommended for all ages, in addition to the risk of causing addiction.

Physical activities can certainly contribute to some aspects of sleep, but more research is needed in this area to prove the effectiveness of exercise on sleep quality, as well as which types of exercise are most effective.

Natural therapies such as acumulture, homeopathy, hypnosis that until now have no scientific proof of their effects or relationship with sleep quality, but promote a state of relaxation that can help with the sleep process

THE RESEARCH

The purpose of our research is to test unsupervised machine learning models to identify clusters that group sleep and routine data. This way we can use this database for comparisons, as well as to suggest changes in habits seeking to positively implement our routine with the aim of improving sleep quality. For this we used two dimension reduction techniques PCA and UMAP and two clustering techniques K-means and DBSCAN.



OBJECTIVES

Provides a cheap, practical and natural alternative for improving sleep quality, through comparisons and recommendations made based on the clusters created by the models.

RESEARCH GOALS

1. Test dimension reduction models and observe their performance, considering their advantages and limitations.

2. Apply unsupervised machine learning algorithms to create clusters and labels based on routine, lifestyle and sleep quality data.

3. Evaluate the significance of the results as well as the performance of the models, analyzing performance, functionality and usefulness.

UNDERSTANDING THE DATA

We start by downloading the necessary libraries, printing the first data information, so we have an idea of what type of data we have and plan the next steps.

```
] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Person ID                374 non-null    int64
 1   Gender                   374 non-null    object
 2   Age                      374 non-null    int64
 3   Occupation               374 non-null    object
 4   Sleep Duration           374 non-null    float64
 5   Quality of Sleep         374 non-null    int64
 6   Physical Activity Level  374 non-null    int64
 7   Stress Level             374 non-null    int64
 8   BMI Category             374 non-null    object
 9   Blood Pressure           374 non-null    object
 10  Heart Rate               374 non-null    int64
 11  Daily Steps              374 non-null    int64
 12  Sleep Disorder           155 non-null    object
dtypes: float64(1), int64(7), object(5)
memory usage: 38.1+ KB
```

```
[ ] df.head()
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	NaN
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	NaN
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	NaN
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea

DATA CLEANING

```
null_counts = df.isnull().sum()
null_counts = null_counts[null_counts > 0]  # Filter to only columns with missing values
print(null_counts)
```

```
Sleep Disorder    219
dtype: int64
```

```
df['Sleep Disorder'].value_counts()
```

```
Sleep Disorder
Sleep Apnea    78
Insomnia       77
Name: count, dtype: int64
```

```
df['Sleep Disorder'].fillna('None', inplace=True)
df['Sleep Disorder'].value_counts()
```

```
Sleep Disorder
None           219
Sleep Apnea     78
Insomnia        77
Name: count, dtype: int64
```

```
df['BMI Category'].value_counts()
```

```
BMI Category
Normal            195
Overweight        148
Normal Weight      21
Obese              10
Name: count, dtype: int64
```

```
df['BMI Category'] = df['BMI Category'].replace('Normal Weight', 'Normal')
df['BMI Category'].value_counts()
```

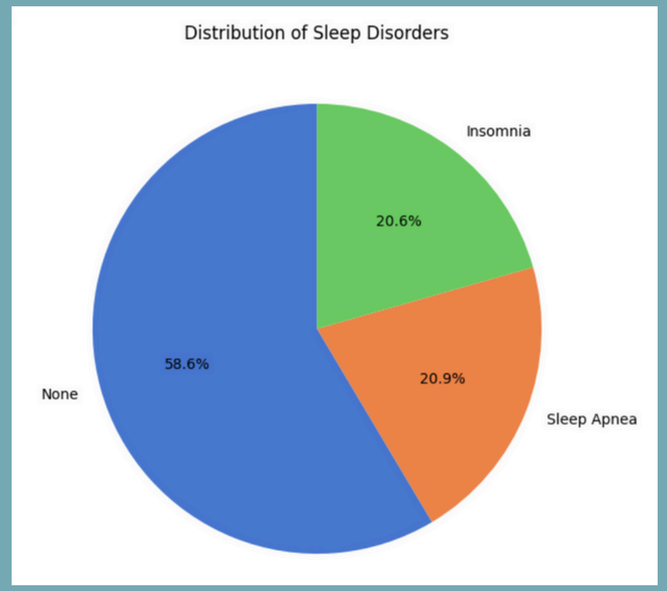
```
BMI Category
Normal        216
Overweight    148
Obese          10
Name: count, dtype: int64
```

```
print("Number of duplicate rows: ", df.duplicated().sum())
```

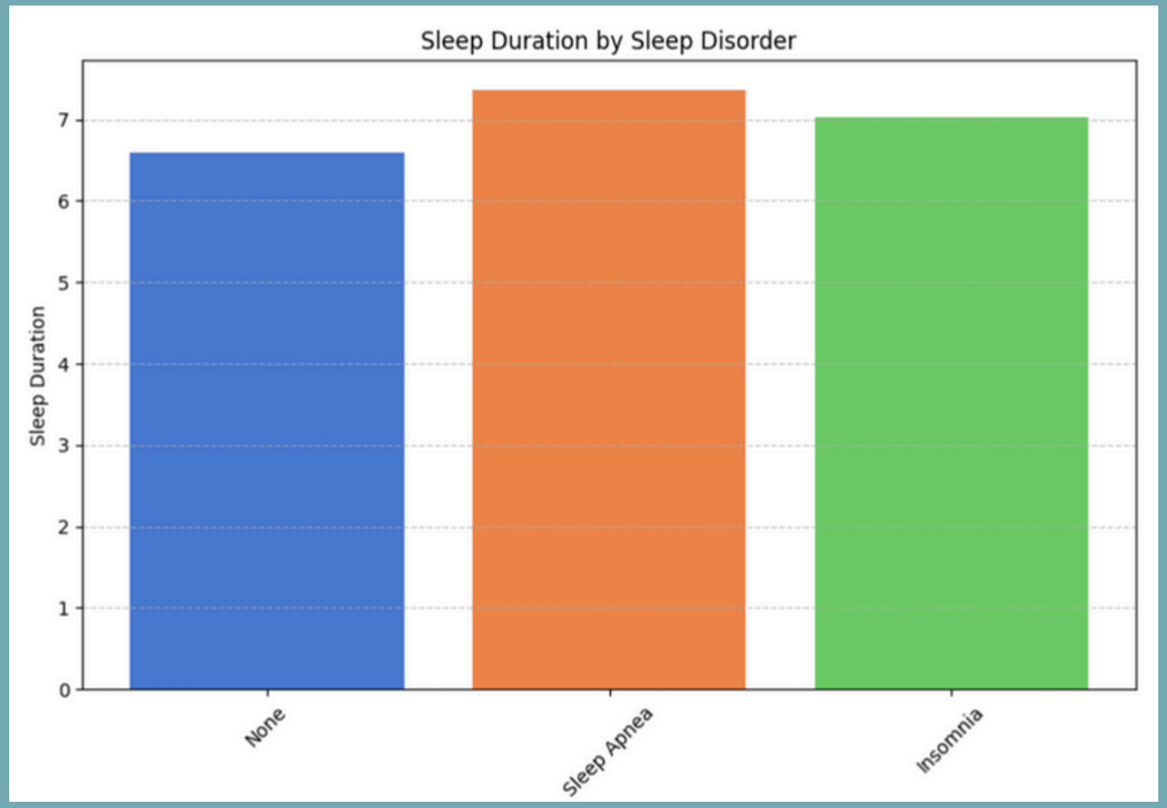
```
Number of duplicate rows:  0
```

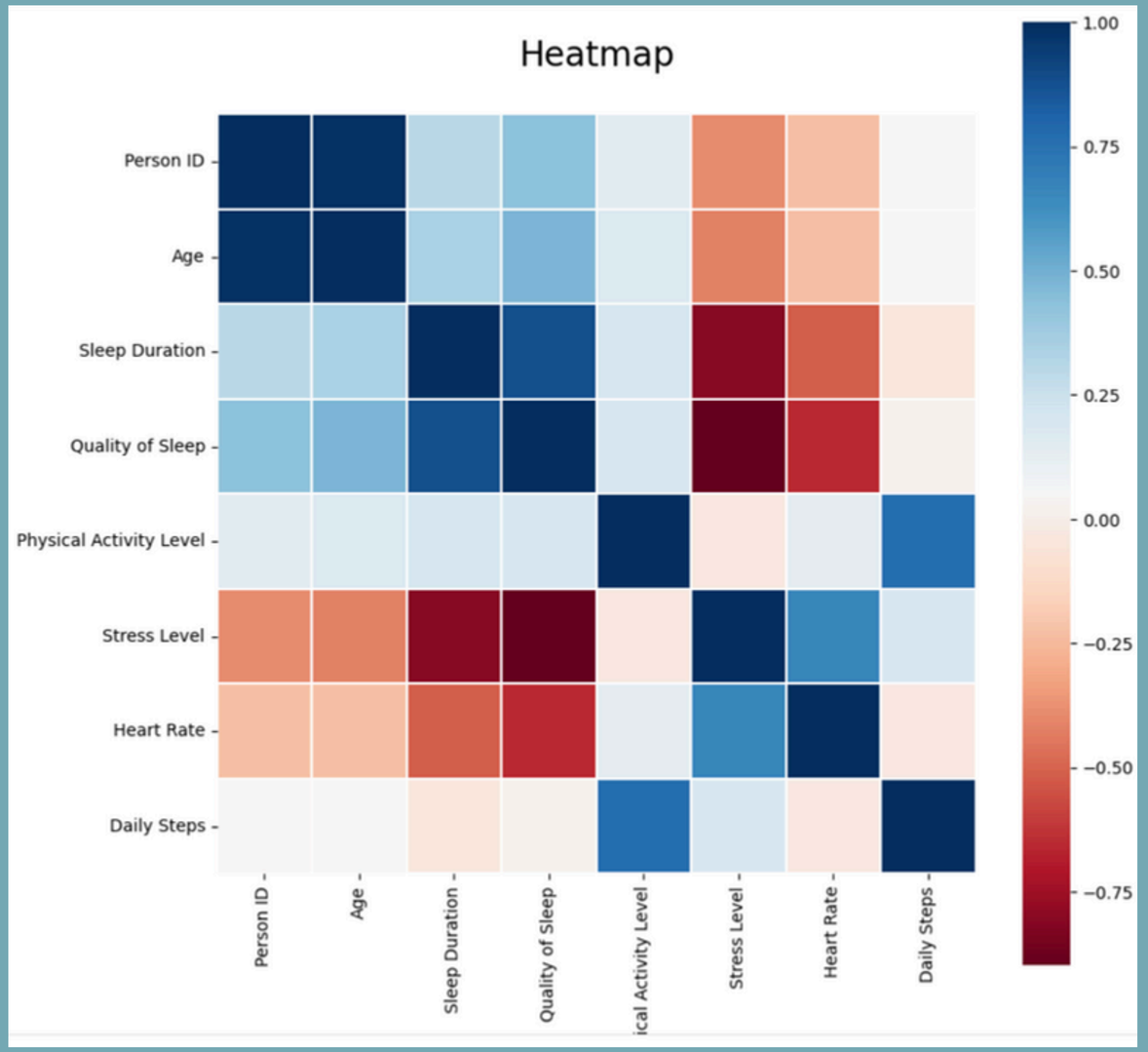
We deal with NaN in the 'Sleep Disorder' feature, duplicate information about 'Normal' and 'Normal Weight' in 'BMI Category' and check duplicate rows.

DATA VISUALIZATION



DATA VISUALIZATION





DATA CONSISTENCY

Transforming categorical data into numeric data

```
df['Sleep Disorder'].value_counts()
```

```
Sleep Disorder
None           219
Sleep Apnea     78
Insomnia        77
Name: count, dtype: int64
```

```
df['Sleep Disorder'] = le.fit_transform(df['Sleep Disorder'])
df['Sleep Disorder'].value_counts()
```

```
Sleep Disorder
1    219
2     78
0     77
Name: count, dtype: int64
```

```
df['Gender'].value_counts()
```

```
Gender
Male      189
Female    185
Name: count, dtype: int64
```

```
from sklearn.preprocessing import LabelEncode

le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
df['Gender'].value_counts()
```

```
Gender
1    189
0    185
Name: count, dtype: int64
```

```
df['Occupation'].value_counts()
```

```
Occupation
Nurse                   73
Doctor                  71
Engineer                63
Lawyer                  47
Teacher                 40
Accountant              37
Salesperson             32
Software Engineer        4
Scientist                4
Sales Representative     2
Manager                  1
Name: count, dtype: int64
```

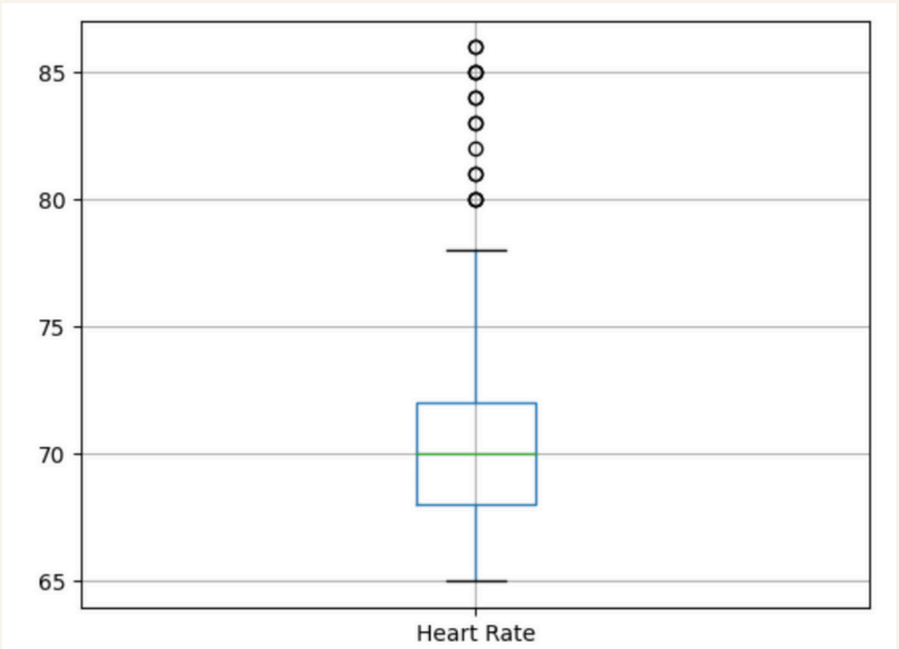
```
df['Occupation'] = le.fit_transform(df['Occupation'])
df['Occupation'].unique()
```

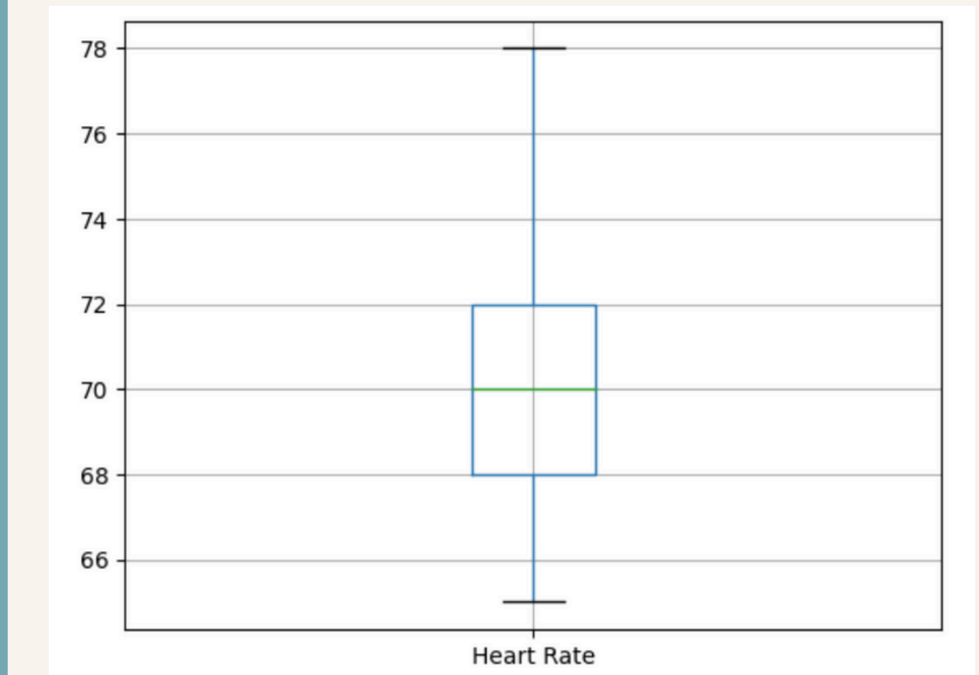
```
array([ 9,  1,  6, 10,  5,  2,  0,  8,  3,  7,  4])
```

```
df['Occupation'].value_counts()
```

```
Occupation
5     73
1     71
2     63
3     47
10    40
0     37
7     32
9      4
8      4
6      2
4      1
Name: count, dtype: int64
```

OUTLIERS






```
#dealing with outliers using the IQR technique
Q1 = df['Heart Rate'].quantile(0.25)
Q3 = df['Heart Rate'].quantile(0.75)
IQR = Q3 - Q1

filter = (df['Heart Rate'] >= Q1 - 1.5 * IQR) & (df['Heart Rate'] <= Q3 + 1.5 *IQR)
df = df.loc[filter]  #original DataFrame is updated

df.boxplot(column='Heart Rate')
```

DESCRIPTIVE STATISTICS

Calculation of basic statistical measures: mean, min, standard deviation, IQR. Correlation Matrix

```
df.describe()
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	Sleep Disorder	Systolic_bp	Diastolic_bp
count	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000	359.000000
mean	190.610028	0.498607	42.428969	3.738162	7.149582	7.376045	59.598886	5.348189	0.807799	69.629526	6945.961003	0.994429	128.214485	84.467967
std	107.055270	0.500696	8.609781	3.051416	0.790936	1.126415	20.799941	1.766760	0.982725	3.231188	1513.894349	0.625324	7.680143	6.209423
min	1.000000	0.000000	27.000000	0.000000	5.900000	5.000000	30.000000	3.000000	0.000000	65.000000	4100.000000	0.000000	115.000000	75.000000
25%	99.500000	0.000000	36.000000	1.000000	6.450000	6.000000	45.000000	4.000000	0.000000	68.000000	6000.000000	1.000000	125.000000	80.000000
50%	191.000000	0.000000	43.000000	3.000000	7.200000	7.000000	60.000000	5.000000	0.000000	70.000000	7000.000000	1.000000	130.000000	85.000000
75%	284.500000	1.000000	50.000000	5.000000	7.800000	8.000000	75.000000	7.000000	2.000000	72.000000	8000.000000	1.000000	135.000000	90.000000
max	374.000000	1.000000	59.000000	10.000000	8.500000	9.000000	90.000000	8.000000	2.000000	78.000000	10000.000000	2.000000	140.000000	95.000000

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	Sleep Disorder	Systolic_bp	Diastolic_bp
Person ID	1.000000	-0.616335	0.990037	0.326567	0.263134	0.390001	0.109790	-0.366938	0.566197	-0.191564	-0.013341	0.197831	0.674325	0.631099
Gender	-0.616335	1.000000	-0.627738	-0.221461	-0.131019	-0.323473	NaN	0.418940	-0.355351	0.218090	0.023853	-0.258748	-0.252345	-0.297175
Age	0.990037	-0.627738	1.000000	0.283383	0.315749	0.439260	0.142077	-0.397676	0.546573	-0.193177	NaN	0.263489	0.666214	0.632673
Occupation	0.326567	-0.221461	0.283383	1.000000	-0.300616	-0.245145	-0.071944	-0.016720	0.709742	0.018465	-0.097703	-0.185218	0.537932	0.537502
Sleep Duration	0.263134	-0.131019	0.315749	-0.300616	1.000000	0.889815	0.174076	-0.801368	-0.379420	-0.612842	-0.086963	0.193711	-0.172908	-0.163647
Quality of Sleep	0.390001	-0.323473	0.439260	-0.245145	0.889815	1.000000	0.127943	-0.908132	-0.310512	-0.721403	-0.093540	0.217127	-0.086196	-0.091125
Physical Activity Level	0.109790	NaN	0.142077	-0.071944	0.174076	0.127943	1.000000	0.012020	0.086683	0.256545	0.820730	0.466927	0.295088	0.404592
Stress Level	-0.366938	0.418940	-0.397676	-0.016720	-0.801368	-0.908132	0.012020	1.000000	0.159313	0.823647	0.249656	-0.038693	0.092676	0.086953
BMI Category	0.566197	-0.355351	0.546573	0.709742	-0.379420	-0.310512	0.086683	0.159313	1.000000	0.284520	0.035056	NaN	0.742340	0.769162
Heart Rate	-0.191564	0.218090	-0.193177	0.018465	-0.612842	-0.721403	0.256545	0.823647	0.284520	1.000000	0.318414	0.214638	0.202218	0.231000
Daily Steps	-0.013341	0.023853	NaN	-0.097703	-0.086963	-0.093540	0.820730	0.249656	0.035056	0.318414	1.000000	0.415721	0.217339	0.333808
Sleep Disorder	0.197831	-0.258748	0.263489	-0.185218	0.193711	0.217127	0.466927	-0.038693	NaN	0.214638	0.415721	1.000000	0.256746	0.326554
Systolic_bp	0.674325	-0.252345	0.666214	0.537932	-0.172908	-0.086196	0.295088	0.092676	0.742340	0.202218	0.217339	0.256746	1.000000	0.975412
Diastolic_bp	0.631099	-0.297175	0.632673	0.537502	-0.163647	-0.091125	0.404592	0.086953	0.769162	0.231000	0.333808	0.326554	0.975412	1.000000

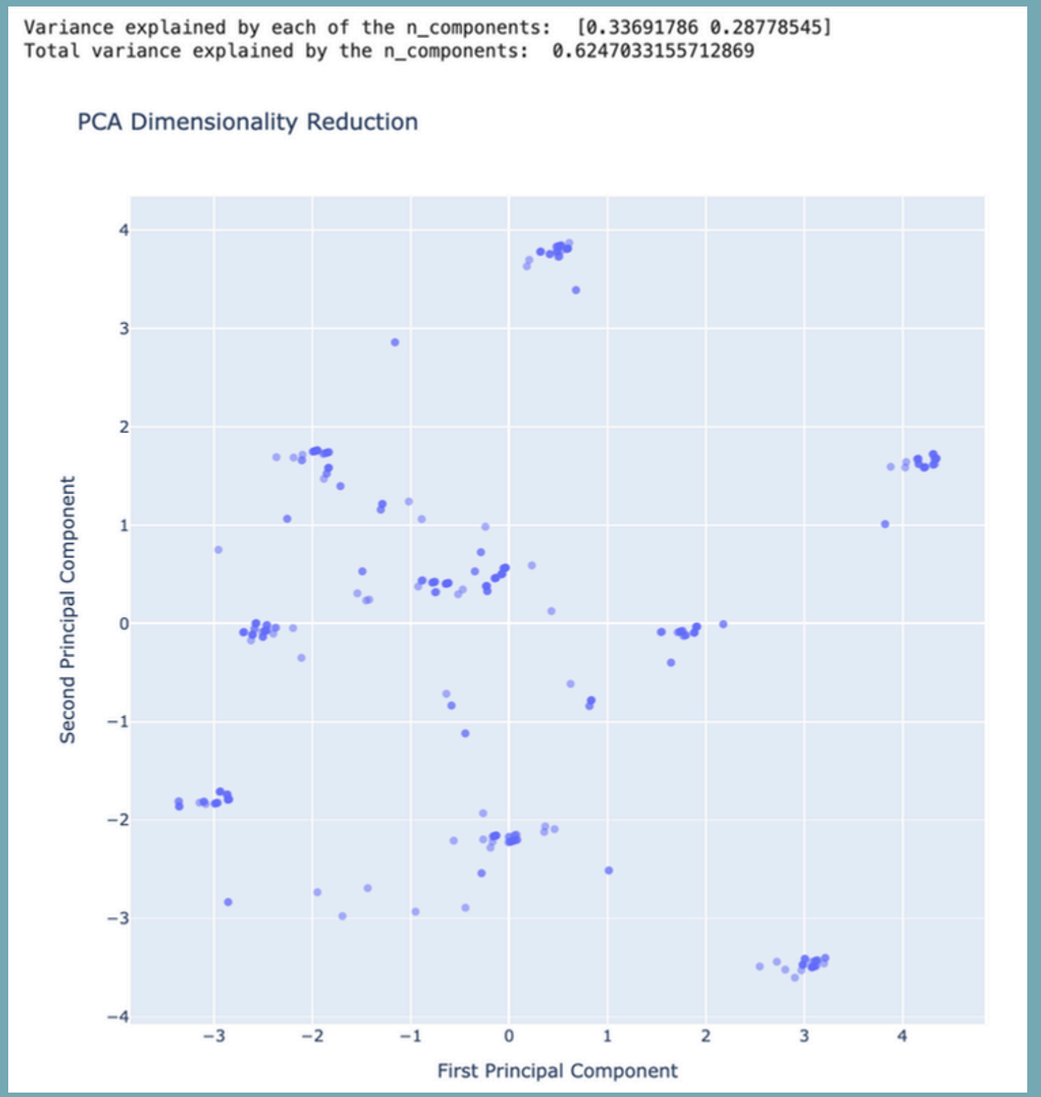
PRINCIPAL COMPONENTS ANALYSIS (PCA)

```
# Normalizing numerical features so that each feature has mean 0 and variance 1
feature_scaler = StandardScaler()
df_scaled = feature_scaler.fit_transform(df)

print(type(df))
print(df.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
(359, 14)
```

```
# Implementing PCA to visualize dataset
pca = PCA(n_components = 2)
pca.fit(df_scaled)
x_pca = pca.transform(df_scaled)
print("Variance explained by each of the n_components: ",pca.explained_variance_ratio_)
print("Total variance explained by the n_components: ",sum(pca.explained_variance_ratio_))

Quality_of_Sleep_pca=list(df['Quality of Sleep'])
data_pca = [go.Scatter(x=x_pca[:,0],
                       y=x_pca[:,1],
                       mode='markers',
                       marker = dict(color=None, colorscale='Rainbow', opacity=0.5),
                       text=[f'Quality of Sleep: {a}' for a in Quality_of_Sleep_pca],
                       hoverinfo='text')]

layout = go.Layout(title = 'PCA Dimensionality Reduction', width = 800, height = 800,
                   xaxis = dict(title='First Principal Component'),
                   yaxis = dict(title='Second Principal Component'))

fig = go.Figure(data=data_pca, layout=layout)
fig.show()
```

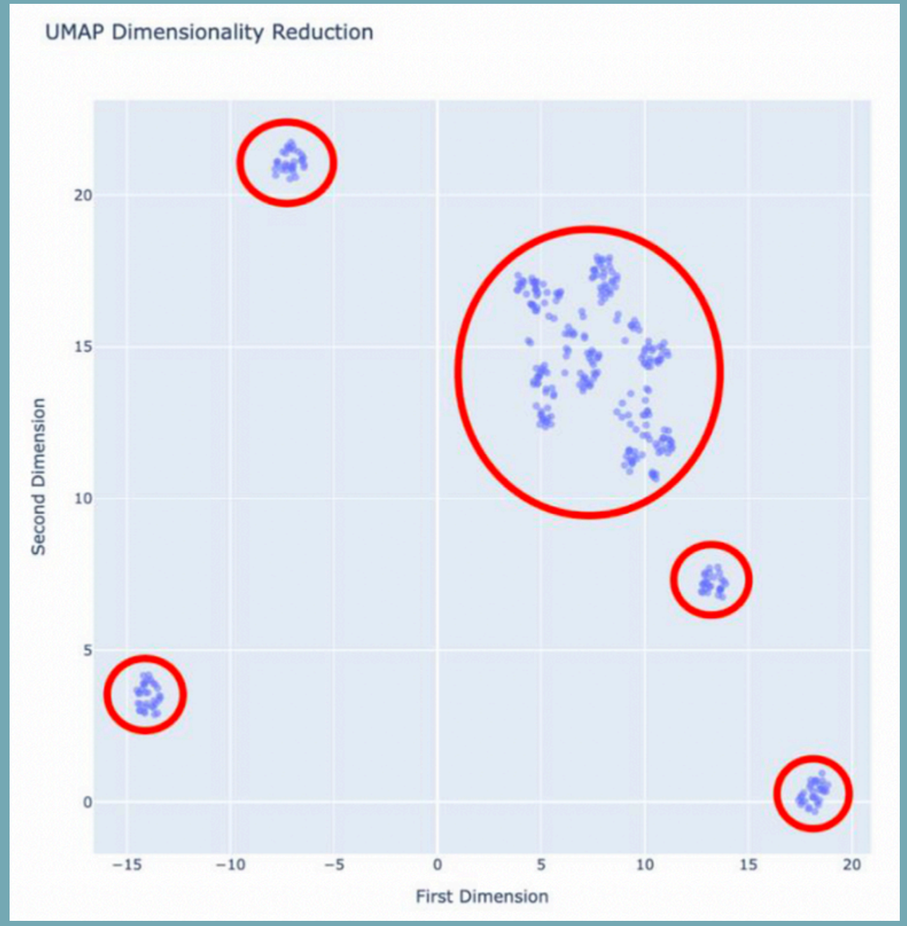
```
Variance explained by each of the n_components:  [0.33691786 0.28778545]
Total variance explained by the n_components:  0.6247033155712869
```



UNIFORM MANIFOLD APPROXIMATION AND PROJECTION (UMAP)

```
#libraries needed to implement UMAP
!pip install umap-learn
import umap.umap_ as umap
```

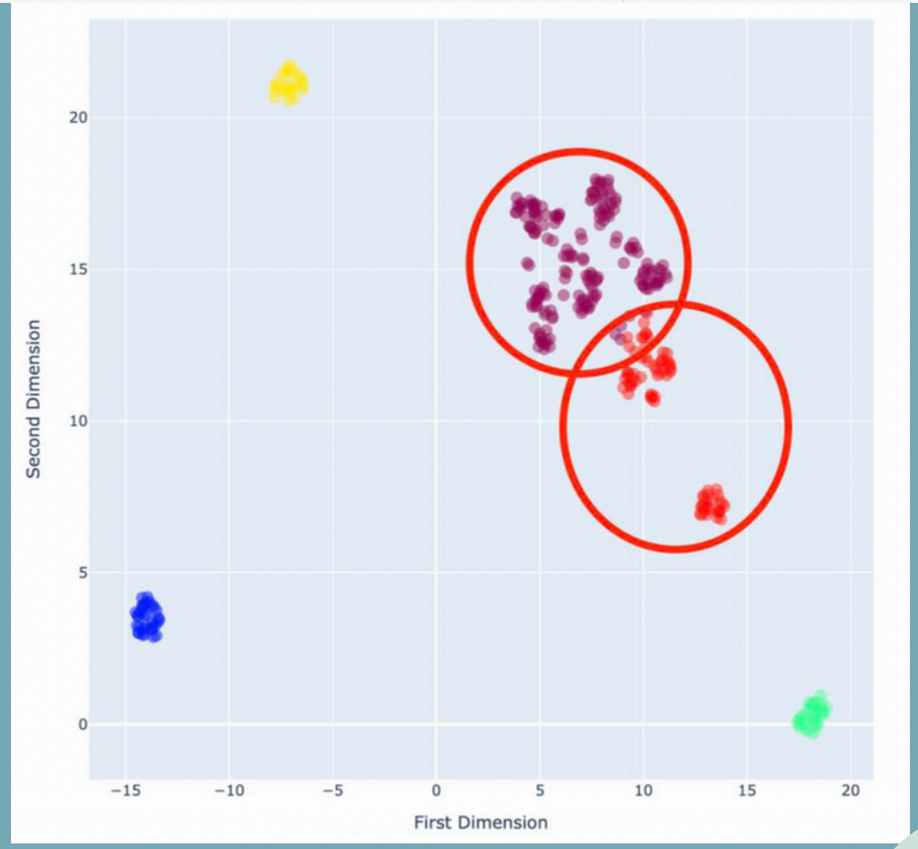
```
# Implementing UMAP to visualize dataset
u = umap.UMAP(n_components = 2, n_neighbors=120, min_dist=0.4, random_state=42)
u.fit(df_scaled)
x_umap = u.transform(df_scaled)

Quality_of_Sleep_umap=list(df['Quality of Sleep'])
data_umap = [go.Scatter(x=x_umap[:,0],
                        y=x_umap[:,1],
                        mode='markers',
                        marker = dict(color=None, colorscale='Rainbow', opacity=0.5),
                        text=[f'Quality of Sleep: {a}' for a in Quality_of_Sleep_umap],
                        hoverinfo='text')]

layout = go.Layout(title = 'UMAP Dimensionality Reduction',
                   width = 800, height = 800,
                   xaxis = dict(title='First Dimension'),
                   yaxis = dict(title='Second Dimension'))

fig = go.Figure(data=data_umap, layout=layout)
fig.show()
```



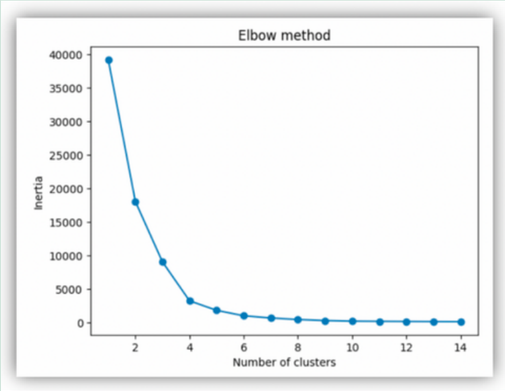
K-MEANS

K-means showed the labels for two clusters that should be identified in a different, more precise way.

```
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(x_umap)

labels_k = list(kmeans.labels_)
data_umap_kmeans = [go.Scatter(x=x_umap[:, 0],
                    y=x_umap[:, 1],
                    mode='markers',
                    marker=dict(color=kmeans.labels_, colorscale='Rainbow', opacity=0.5, size=10),
                    text=[f'Quality of Sleep: {a}<br>Label: {b}' for a,b in list(zip(Quality_of_Sleep_umap,labels_k))],
                    hoverinfo='text')]

layout = go.Layout(title='UMAP Dimensionality Reduction + K-means Cluster',
                   width=800,
                   height=800,
                   xaxis=dict(title='First Dimension'),
                   yaxis=dict(title='Second Dimension'))

fig = go.Figure(data=data_umap_kmeans, layout=layout)
fig.show()

df['Label'] = kmeans.labels_
df.to_csv("/content/drive/My Drive/Cluster_Sleep_Kmeans.csv", index=False)
print(df.Label.value_counts())
```



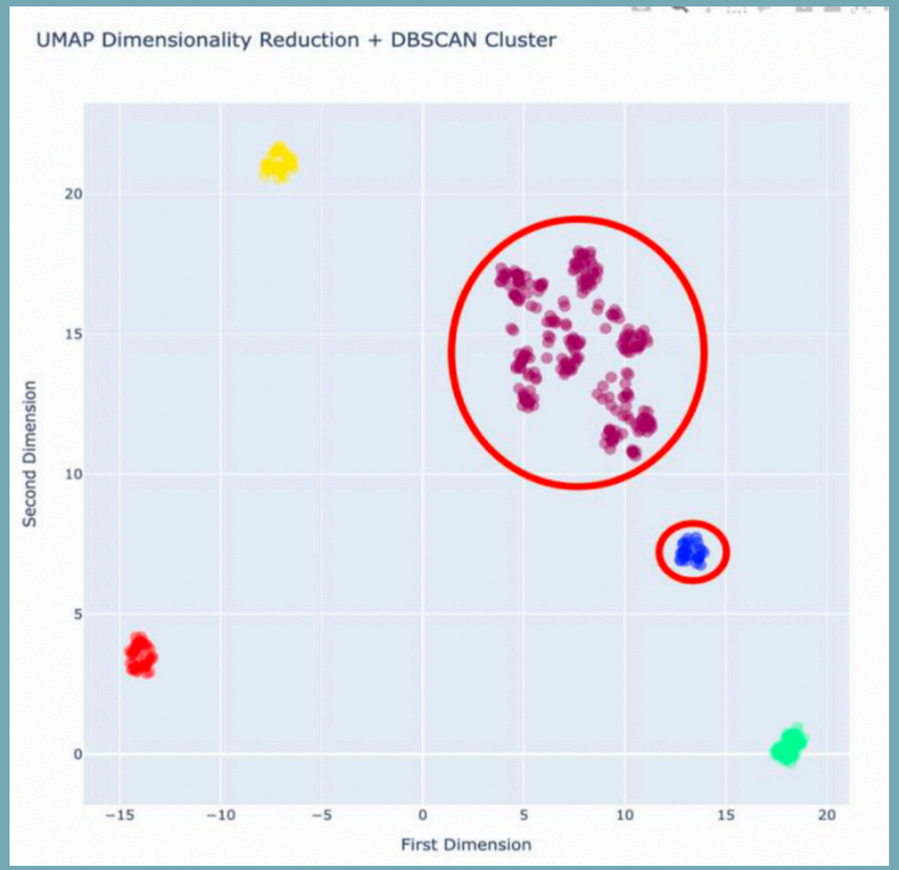
Elbow technique to get a sense of the cluster number for K-means



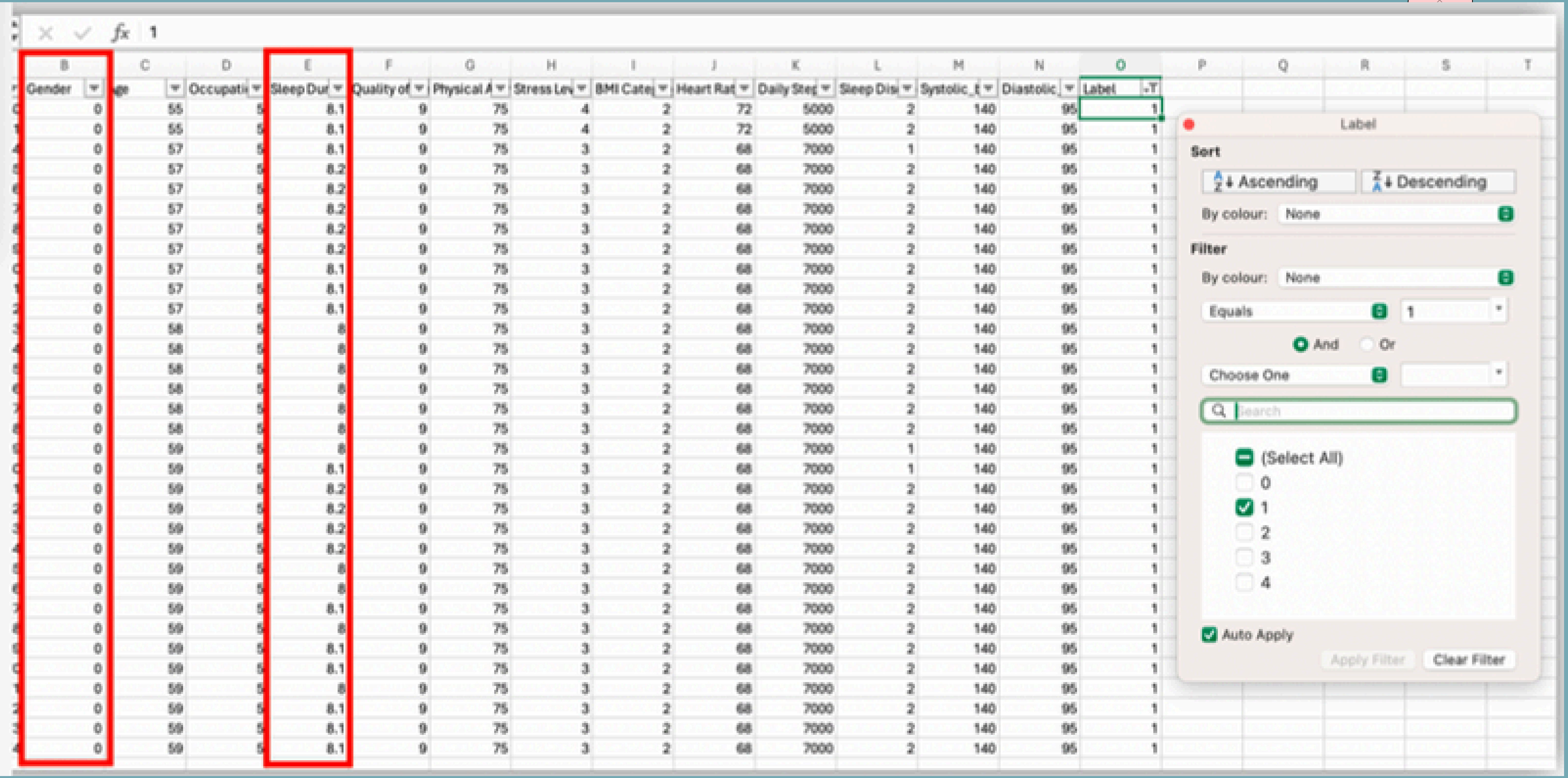
DENSITY-BASED SPATIAL CLUSTERING OF APPLICATION WITH NOISE (DBSCAN)

DBSCAN was able to correctly identify the clusters, effectively assigning the labels

```
db = DBSCAN(eps=1.5, min_samples=5)
db.fit(x_umap)

labels_db = db.labels_
data_umap_db = [go.Scatter(x=x_umap[:, 0],
                    y=x_umap[:, 1],
                    mode='markers',
                    marker=dict(color=db.labels_, colorscale='Rainbow', opacity=0.5, size=10),
                    text=[f'Quality of Sleep: {a}<br>Label: {b}' for a,b in list(zip(Quality_of_Sleep_umap,labels_db))],
                    hoverinfo='text')]

layout = go.Layout(title='UMAP Dimensionality Reduction + DBSCAN Cluster',
                    width=800,
                    height=800,
                    xaxis=dict(title='First Dimension'),
                    yaxis=dict(title='Second Dimension'))

fig = go.Figure(data=data_umap_db, layout=layout)
fig.show()

df['Label'] = db.labels_
df.to_csv("/content/drive/My Drive/Cluster_Sleep_DBSCAN.csv", index=False)
print(df.Label.value_counts())
```



WE HAVE SOME PROBLEMS...



Gender	ge	Occupati	Sleep Dur	Quality of	Physical A	Stress Lev	BMI Cate	Heart Rat	Daily Step	Sleep Dis	Systolic_t	Diastolic_	Label
0	55	5	8.1	9	75	4	2	72	5000	2	140	95	1
0	55	5	8.1	9	75	4	2	72	5000	2	140	95	1
0	57	5	8.1	9	75	3	2	68	7000	1	140	95	1
0	57	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	57	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	57	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	57	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	57	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	57	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	57	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	57	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	58	5	8	9	75	3	2	68	7000	2	140	95	1
0	58	5	8	9	75	3	2	68	7000	2	140	95	1
0	58	5	8	9	75	3	2	68	7000	2	140	95	1
0	58	5	8	9	75	3	2	68	7000	2	140	95	1
0	58	5	8	9	75	3	2	68	7000	2	140	95	1
0	58	5	8	9	75	3	2	68	7000	2	140	95	1
0	59	5	8	9	75	3	2	68	7000	1	140	95	1
0	59	5	8.1	9	75	3	2	68	7000	1	140	95	1
0	59	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.2	9	75	3	2	68	7000	2	140	95	1
0	59	5	8	9	75	3	2	68	7000	2	140	95	1
0	59	5	8	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	59	5	8	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	59	5	8	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.1	9	75	3	2	68	7000	2	140	95	1
0	59	5	8.1	9	75	3	2	68	7000	2	140	95	1

MODIFYING SOME POINTS IN SEARCH OF BETTER RESULTS

```
[55] df['Sleep Duration'].value_counts()

Sleep Duration
7.2    36
6.0    31
7.8    28
6.1    25
7.7    24
6.5    23
6.6    20
7.1    19
8.4    14
8.0    13
8.1    13
8.5    13
6.3    12
7.3    12
6.2    12
8.2    11
7.6    10
6.4     9
7.9     7
7.5     5
6.8     5
8.3     5
6.7     5
6.9     3
7.4     3
5.9     1
Name: count, dtype: int64
```

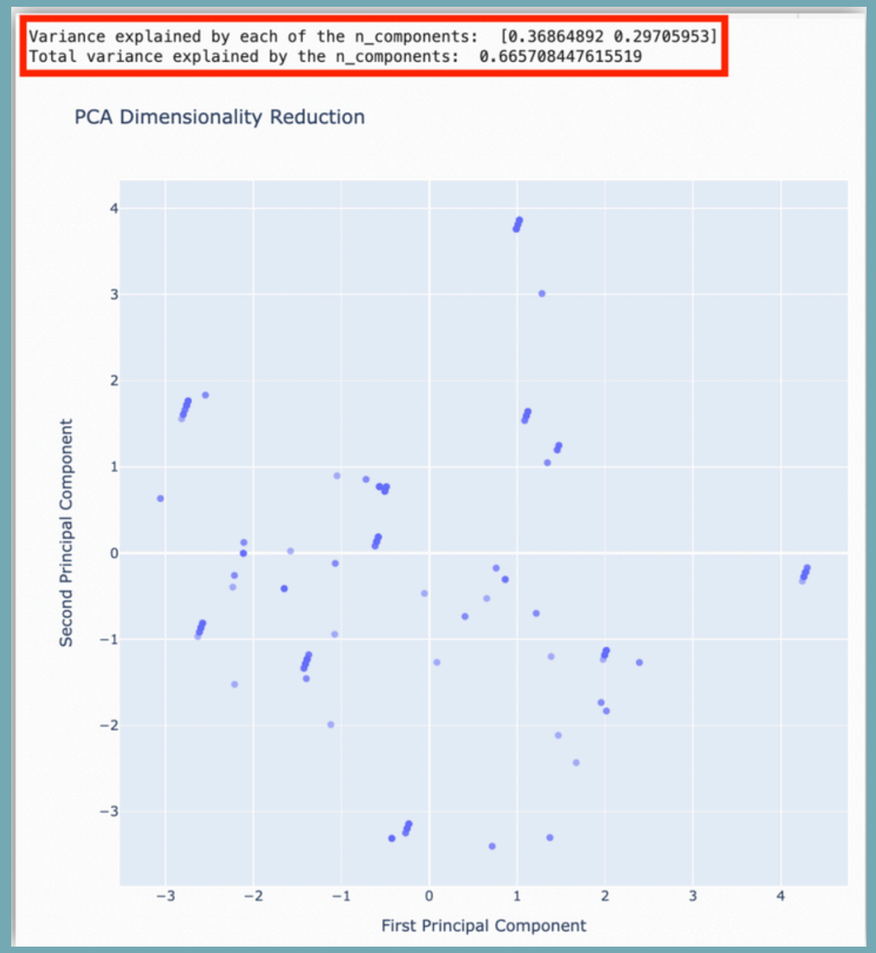
```
[56] df['Sleep Duration'] = df['Sleep Duration'].round().astype(int)
     df['Sleep Duration'].value_counts()

Sleep Duration
8    143
6    113
7    103
Name: count, dtype: int64
```

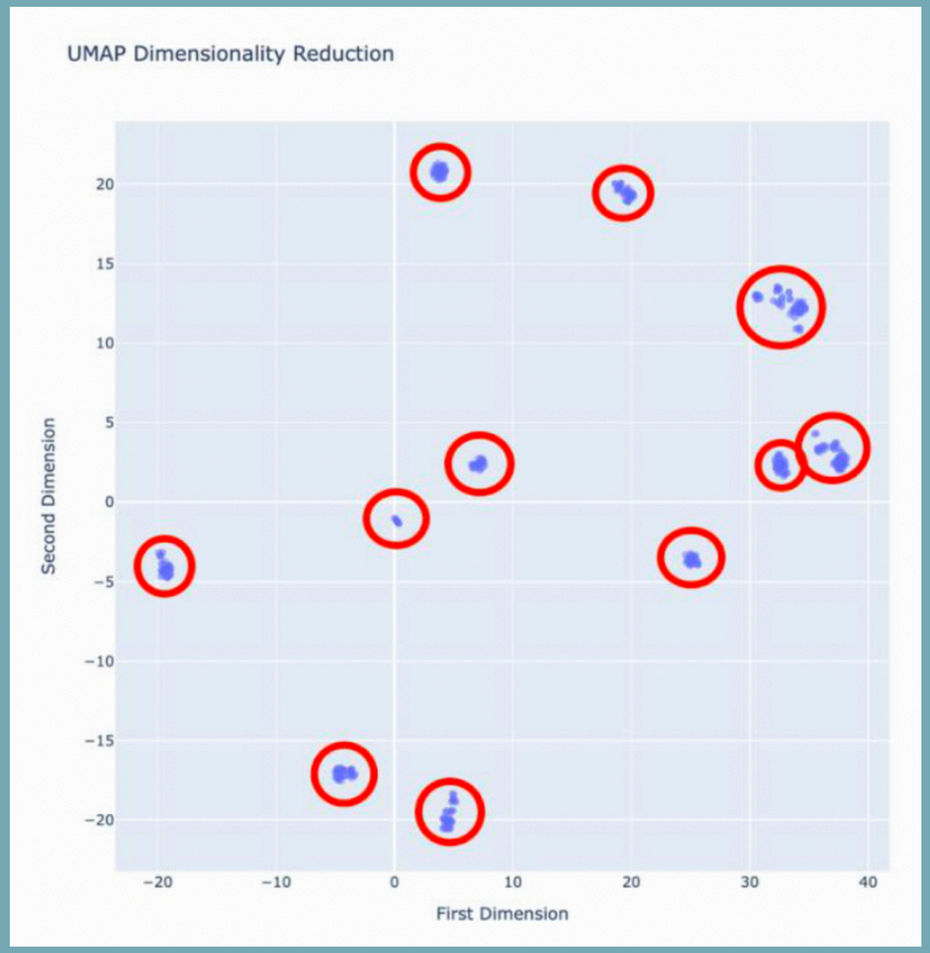
We round the sleep duration to achieve more accurate results

	Feature	Importance
6	Stress Level	0.173716
4	Sleep Duration	0.135934
0	Person ID	0.126047
13	Label	0.104709
2	Age	0.102862
8	Heart Rate	0.091381
5	Physical Activity Level	0.061527
3	Occupation	0.039441
9	Daily Steps	0.038264
11	Systolic_bp	0.037666
7	BMI Category	0.036159
12	Diastolic_bp	0.033915
1	Gender	0.010608
10	Sleep Disorder	0.007771

Dropped 'Labels', 'Person IS', 'Gender' a 'Sleep Disorder'

NEW ATTEMPTS PCA AND UMAP

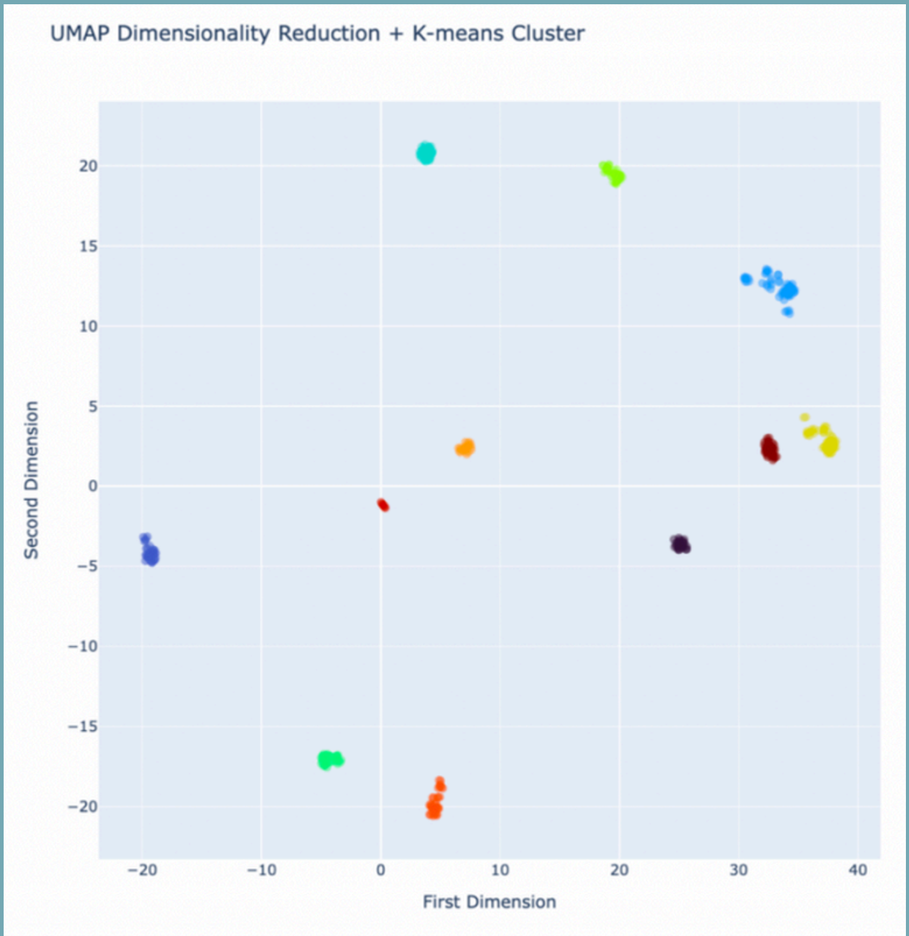


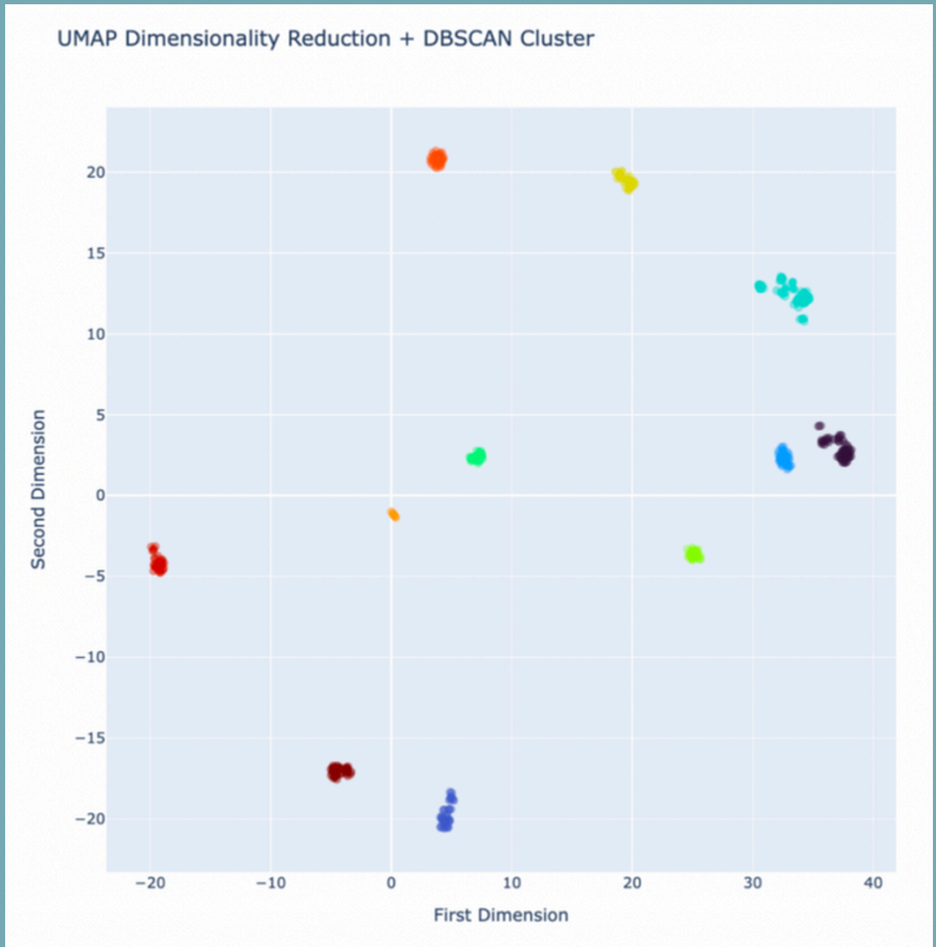


increase in the variance value, but not enough to adequately identify the clusters.

significant improvement in the division of clusters, presenting better divided groups

NEW ATTEMPTS K-MEANS AND DBSCAN





Both models adequately assigned the new labels to the clusters.

RESULTS

PCA

PCA was not able to separate the data into clusters, presenting visualizations without clarity or meaning. It is important to emphasize that we should not proceed with the use of PCA when we have less than 75% variance capture. In this case, PCA did not work, although it is a very good algorithm in other situations.

UMAP

UMAP separated the clusters properly from the first attempt, delivering even better results after the modifications made. This is an algorithm that performs very well in numerous cases, including large datasets, making it a good option in more complex cases.

RESULTS

K-means

K-means is very popular for clustering, however its limitations make it a more difficult algorithm to deal with. In our first attempt K-means failed, however after improving the input data and reviewing the values for K, we had good results.

DBSCAN

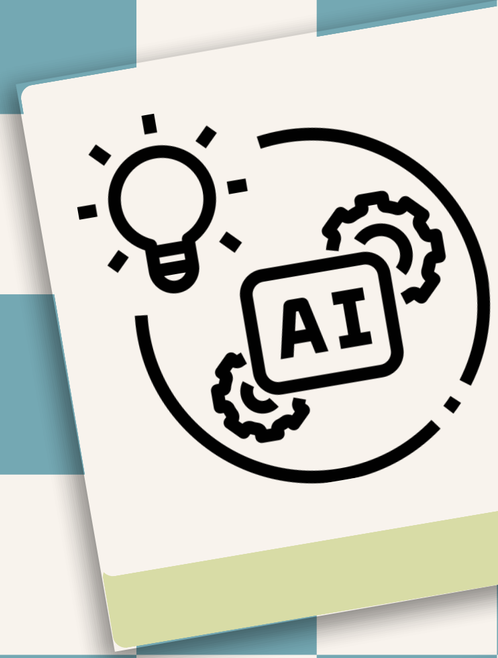
DBSCAN is a more versatile and dynamic algorithm that works well on different data sets. We were able to use the same parameters from start to finish, making the code building process less complex. The labels were correctly assigned to the clusters, enabling good interpretation of the results.

CONCLUSIONS

1. Although PCA is a popularly known technique for dimension reduction, in the case of clustering this algorithm did not perform as well as the recently discovered UMAP which delivered good results in data clustering.

2. Even though it delivers good results in the end, K-means has limitations that make it perform less well in certain types of datasets. While DBSCAN adapts well and did its job through our research,

3. It is also concluded that feature selection plays an important role even with the reduction of dimensions, as the better the inputs, the better the results



THANK YOU