TABLE OF CONTENTS

TASK 3 (CLUSTERING)

Title

The use of clustering algorithms to find groups in a User Knowledge Modeling dataset: Comparing the classes discovered to the 'knowledge level of user' classes already established.

Introduction

User knowledge is centered around an individual who accesses files stored in a knowledge repository and applies the knowledge to make more refined decisions while completing tasks.

For instance, when students prepare for an examination or a project, there is a need to explore and study materials for an appreciable length of time. The degree of study time for both the goal and related object, and the user exam performance, are all attributes that can point to the knowledge level of the user. (Knowledge user engagement, 2016)

There is a need to be able to separate the dataset into groups, so that the characteristics of each group can be studied.

This report aims to group the dataset by applying the K-Means and Hierarchical clustering algorithm, and also test its efficiency by comparing the results with the already established user knowledge levels.

Datasets

This dataset was created by Hamdi Tolga Kahraman from the Karadeniz Technical University, Trabzon, Turkey. It contains information about students' level of knowledge of Electrical DC Machines.

It was designed to find groups or clusters, assess the attributes of such clusters, and draw information from the data presented.

It can be found at https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling and was downloaded in the '.xls' format.

Explanation and preparation of datasets

The data was downloaded as an excel workbook with 3 sheets. The second and third sheets contained the test and train data respectively so they were joined and columns to the analysis were dropped. See the code and result below:

```
In [68]: workbook = pd.ExcelFile('C:\\Users\C\Downloads\Data_User_Modeling_Dataset_Hamdi Tolga KAHRAMAN.xls')
sheet1 = pd.read_excel(workbook, 1)
sheet2 = pd.read_excel(workbook, 2)
dataset = pd.concat([sheet1,sheet2], ignore_index=True)
dataset.drop(dataset.columns[[6,7,8]], inplace=True, axis=1)
dataset[' UNS'] = dataset[' UNS'].replace(['very_low'],'Very Low', inplace=True)
dataset.drop(dataset.columns[[6]], inplace=True, axis=1)
dataset.columns = dataset.columns.str.replace(' UNS', 'UNS')
dataset
```

Out[68]:

	STG	SCG	STR	LPR	PEG	UNS
0	0.00	0.00	0.00	0.00	0.00	Very Low
1	0.08	0.08	0.10	0.24	0.90	High
2	0.06	0.06	0.05	0.25	0.33	Low
3	0.10	0.10	0.15	0.65	0.30	Middle
4	0.08	0.08	0.08	0.98	0.24	Low
...	...	...	...	...	...	...
398	0.90	0.78	0.62	0.32	0.89	High
399	0.85	0.82	0.66	0.83	0.83	High
400	0.56	0.60	0.77	0.13	0.32	Low
401	0.66	0.68	0.81	0.57	0.57	Middle
402	0.68	0.64	0.79	0.97	0.24	Middle

403 rows × 6 columns

The figure below shows that the dataset has 403 rows and 6 columns. The first 5 columns are the features and are all of ‘float’ datatypes except the last column which is the class and is an ‘object’ datatype.

```
In [29]: dataset.shape
Out[29]: (403, 6)

In [28]: dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 403 entries, 0 to 402
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   STG     403 non-null    float64
 1   SCG     403 non-null    float64
 2   STR     403 non-null    float64
 3   LPR     403 non-null    float64
 4   PEG     403 non-null    float64
 5    UNS    403 non-null    object 
dtypes: float64(5), object(1)
memory usage: 19.0+ KB
```

The figure above shows the non-null count in all columns equal to the number of rows, this means there are no missing values. To verify further, see below:

```
In [38]: dataset.isnull().sum()
Out[38]: STG     0
SCG     0
STR     0
LPR     0
PEG     0
 UNS    0
dtype: int64
```

The meaning of each column is as shown below:

	Columns	Full Column Name
0	STG	The degree of study time for goal object materails
1	SCG	The degree of repetition number of user for goal object materails
2	STR	The degree of study time of user for related objects with goal object
3	LPR	The exam performance of user for related objects with goal object
4	PEG	The exam performance of user for goal objects
5	UNS	The knowledge level of user

Exploratory Data Analysis

For a quick overview of the data, '.describe' was used, and a chart to show the distribution amongst the previously given classes was plotted.

```
In [39]: dataset.describe()
```

Out[39]:

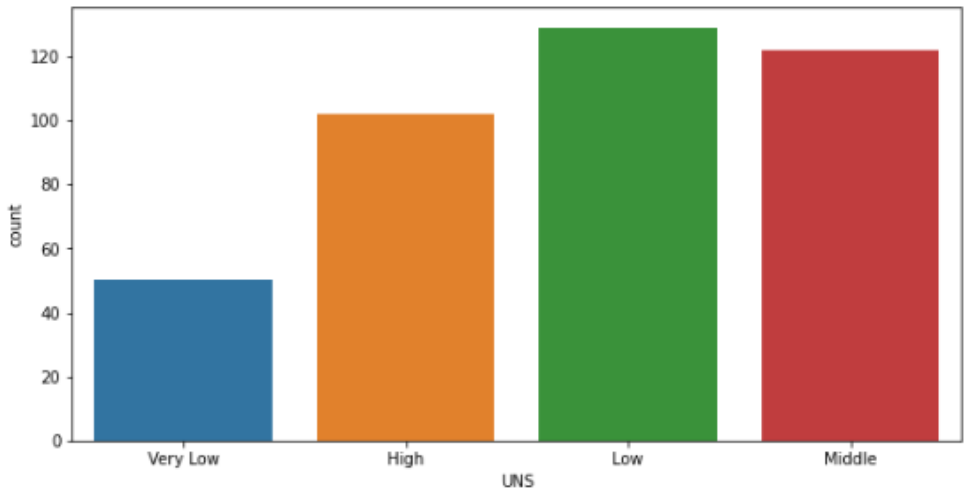
	STG	SCG	STR	LPR	PEG
count	403.000000	403.000000	403.000000	403.000000	403.000000
mean	0.353141	0.355940	0.457655	0.431342	0.456360
std	0.212018	0.215531	0.246684	0.257545	0.266775
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.200000	0.200000	0.265000	0.250000	0.250000
50%	0.300000	0.300000	0.440000	0.330000	0.400000
75%	0.480000	0.510000	0.680000	0.650000	0.660000
max	0.990000	0.900000	0.950000	0.990000	0.990000

The graph showing the count of the class variable is shown below

```
In [85]: plt.figure(figsize = (10 , 5))
         sns.countplot(data=dataset, x='UNS')
```

Out[85]:

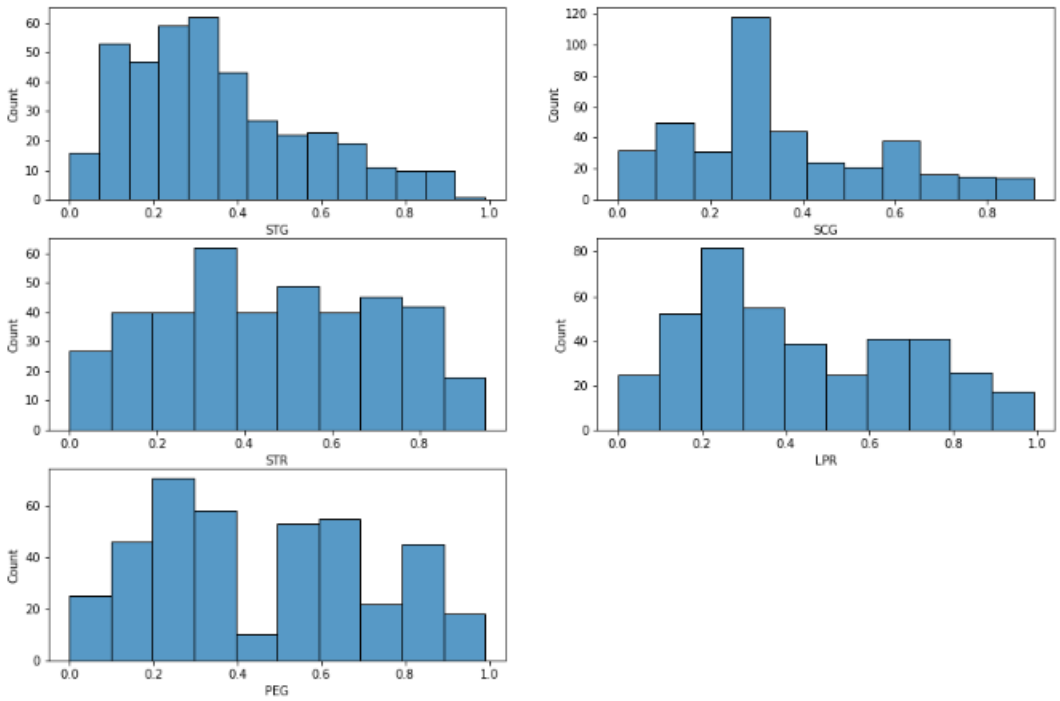
```
<AxesSubplot:xlabel='UNS', ylabel='count'>
```



It can be observed that the majority of the students have a knowledge level in the Low and Middle classes.

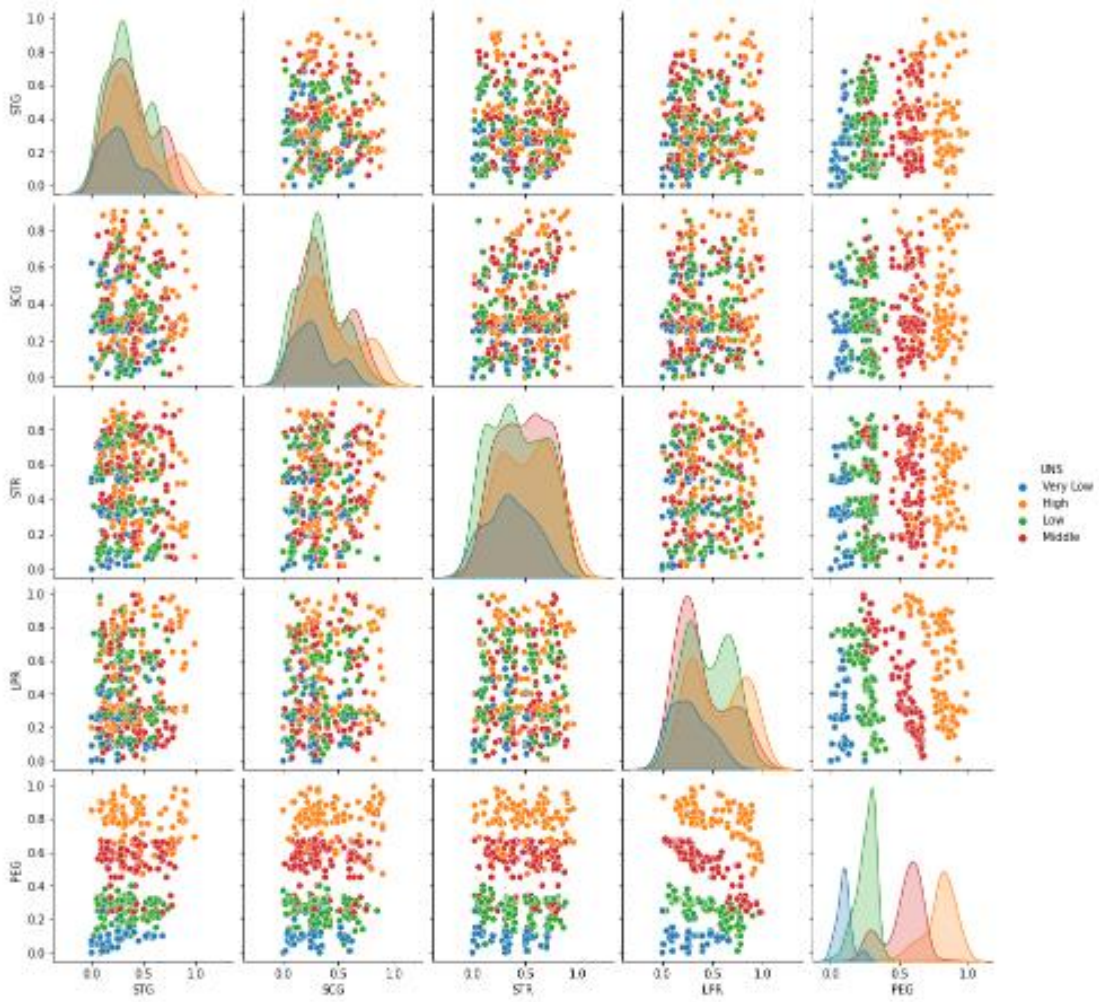
To get a view of the distribution of all the attributes of all the students in one glance

```
In [91]: plt.figure(figsize=(15,10))
         for i in range(5):
             plt.subplot(3, 2, i+1)
             sns.histplot(data=dataset, x=dataset.columns[i])
```



To view the relationships between attributes and the class variable distribution, we can plot a scatterplot matrix:

```
In [95]: sns.pairplot(dataset, hue='UNS')
         #sns.pairplot(dataset.iloc[:,[0,1,2,3,4]])
Out[95]: <seaborn.axisgrid.PairGrid at 0x1d2b163c5b0>
```



From the graph above, since exams are a means of testing User-knowledge, the PEG which is the Exam Performance for Goal Objects, shows a positive correlation with user knowledge. The class level clusters can already be seen for all PEG axis, so it suggests that as the exam performance increases, so does the ranking in User Knowledge.

Scaling the X variables

This step was done to make the columns have a mean of zero and a variance of 1, i.e scale the data.

```
In [98]: X = dataset.drop('UNS',axis=1)
         sc_X = StandardScaler()
         X = sc_X.fit_transform(X)
```

Reducing the dimensions

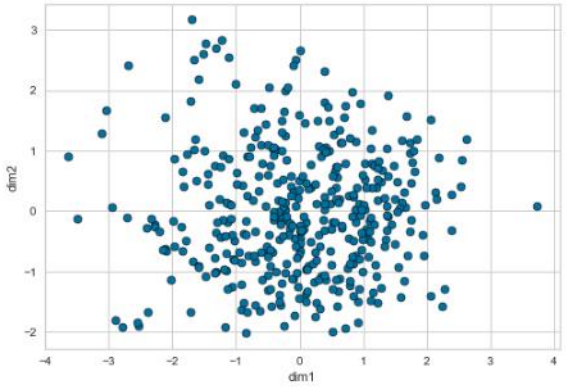
There are 5 columns to plot in the dataset on a 2-Dimensional chart, which is not physically possible, so there is the need to reduce the 5 dimensions into 2. This is done using the PCA function. The code and the top 5 rows of the result of the reduction and its chart are shown below:

```
In [46]: dim = PCA(n_components=2)
         dimmed = dim.fit_transform(X)
         dimensions = pd.DataFrame(dimmed, columns=['dim1','dim2'])
         dimensions.head()
```

Out[46]:

	dim1	dim2
0	3.726245	0.091266
1	1.019882	0.512718
2	2.393311	0.275547
3	1.664575	0.072429
4	1.607760	0.147666

```
In [45]: sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, edgecolor='black')
Out[45]: <AxesSubplot:xlabel='dim1', ylabel='dim2'>
```



Implementation in Python

Brief description of the algorithms used

k-Means Clustering Algorithm

This method of clustering works by starting with k number of means and partitioning each observation in the dataset based on its proximity to the nearest mean defined in the initial process.

It repeats this process until the means (cluster centroids) are no longer changing significantly.

The distance between two points on a plane is the Euclidean Distance, while the initial points (means) identified are collectively called the Cluster Centroids.

Hierarchical Clustering Algorithm

Like the k-means, this method is also a distance-based clustering method. It starts with each object as a single cluster, then it merges the closest pair of clusters into one, this process is repeated until there's only one cluster left.

This approach is different from the k-means because it works from the bottom-up, starting from the total number of data points as the number of clusters, while k-means uses the top-bottom approach, which means defining the number of clusters and working down to individual data points.

Application & Explanation of Techniques

K-Means Clustering

The Elbow Method

To get the optimum number of clusters, 3 different evaluation metrics were run, and k-means clustering was applied using the results of each. A comparison of that result and the true values is eventually shown.

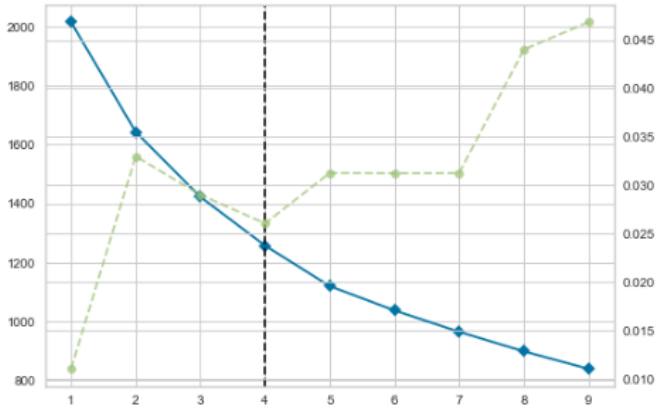
- Distortion score elbow
- Silhouette
- Calinski-Harabasz

Distortion Score

This evaluation gives the optimum number of k as 4:

```
In [31]: model = KMeans()
         visualizer = KElbowVisualizer(model, k=(1,10))
         visualizer.fit(X)
         visualizer.show
```

```
Out[31]: <bound method Visualizer.show of KElbowVisualizer(ax=<AxesSubplot:>, estimator=KMeans(n_clusters=9), k=(1, 10))>
```

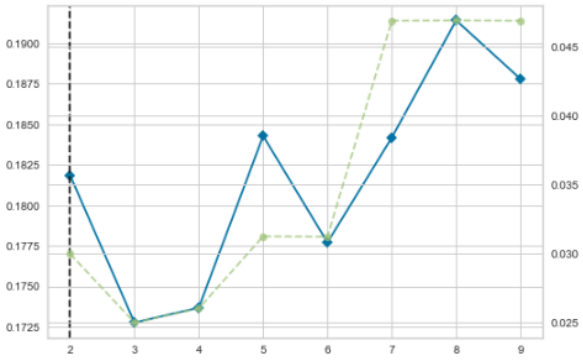


Silhouette Score

This evaluation gives the optimum number of k as 2:

```
In [33]: model = KMeans()
         visualizer = KElbowVisualizer(model, k=(2,10), metric = 'silhouette')
         visualizer.fit(X)
         visualizer.show
```

```
Out[33]: <bound method Visualizer.show of KElbowVisualizer(ax=<AxesSubplot:>, estimator=KMeans(n_clusters=9), k=(2, 10),
                    metric='silhouette')>
```

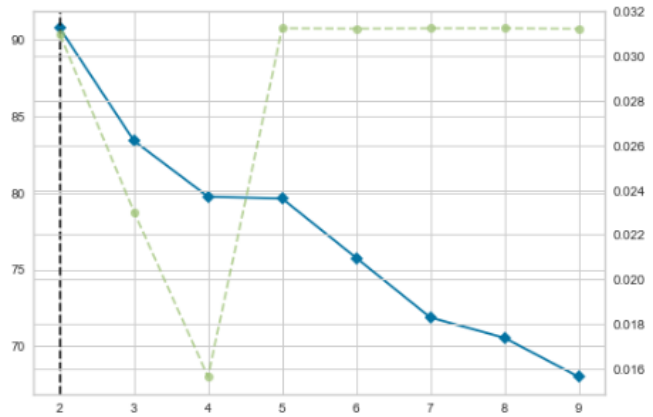


Calinski-Harabasz Score

This evaluation gives the ideal number of k as 2:

```
In [35]: model = KMeans()
         visualizer = KElbowVisualizer(model, k=(2,10), metric='calinski_harabasz')
         visualizer.fit(X)
         visualizer.show
```

```
Out[35]: <bound method Visualizer.show of KElbowVisualizer(ax=<AxesSubplot:>, estimator=KMeans(n_clusters=9), k=(2, 10),
                    metric='calinski_harabasz')>
```

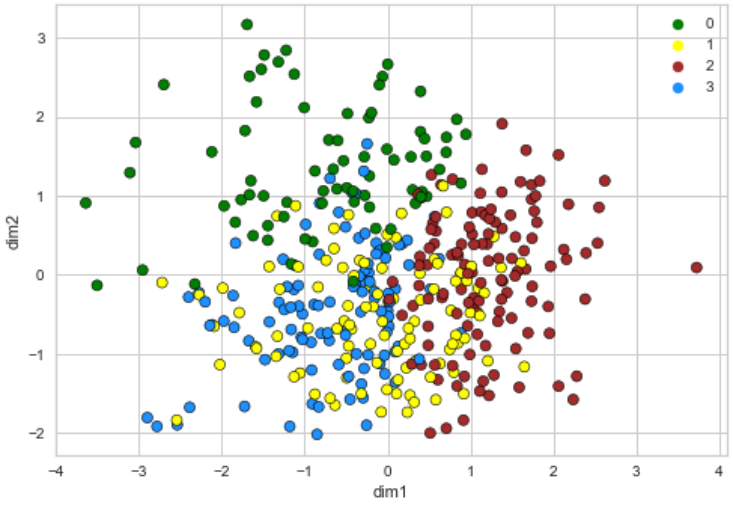


K-Means Cluster Predict Results Visualisation

The Distortion method above shows 4 as the ideal number of k, while the Silhouette and the Calinski-Halabasz method give 2. Plotting the K-Means with this number of k gives the results as shown below:

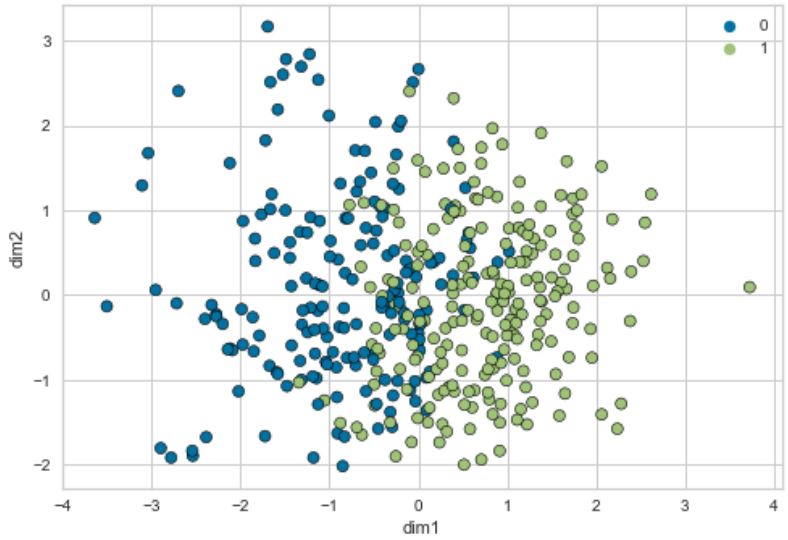
For k=4:

```
In [42]: kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)
         clusters = kmeans.fit_predict(X)

In [47]: sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, hue=clusters, edgecolor='black',
                         palette=['green','yellow','brown','dodgerblue'])

Out[47]: <AxesSubplot:xlabel='dim1', ylabel='dim2'>
```



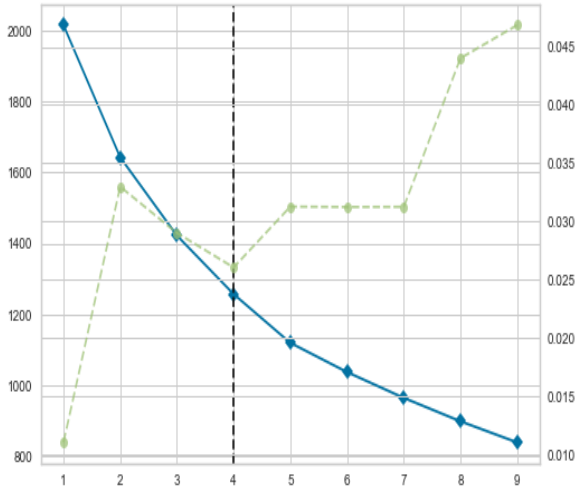
For k=2:

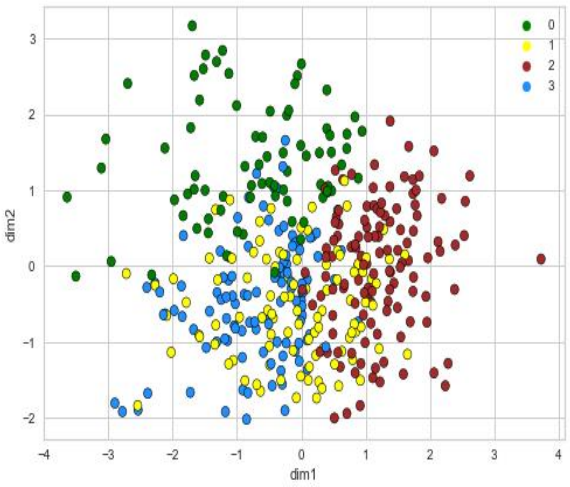
```
In [38]: kmeans = KMeans(n_clusters=2, init='k-means++', random_state=42)
         clusters = kmeans.fit_predict(X)

In [39]: sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, hue=clusters, edgecolor='black')

Out[39]: <AxesSubplot:xlabel='dim1', ylabel='dim2'>
```



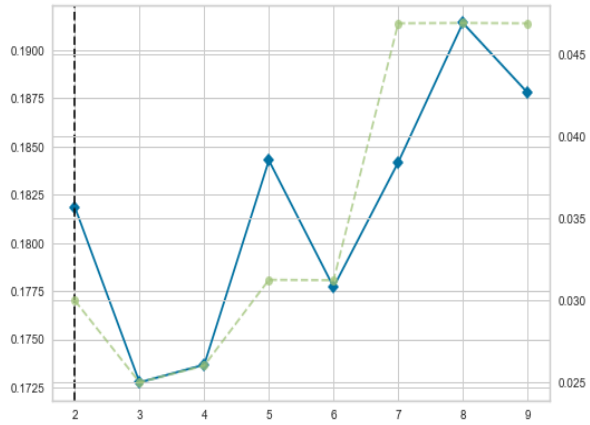
Visualisation of the Results

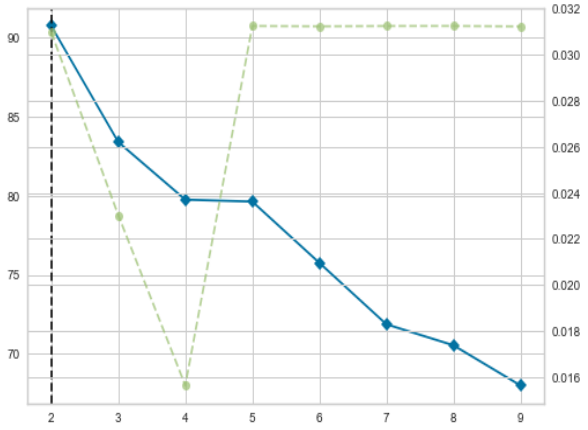
For k=4, gotten using the distortion evaluation metric, the elbow plot and the cluster scatterplot are shown side-by-side below:

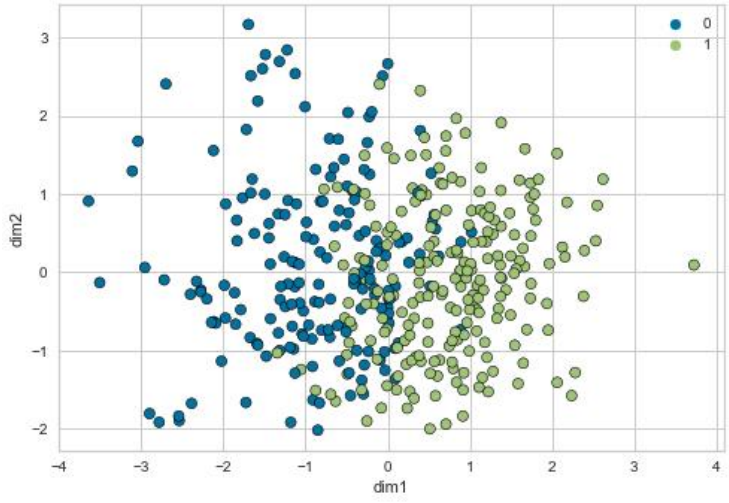




For k=2, gotten using the Silhouette and the Calinski-Halabasz evaluation metric, the elbow plot and the cluster scatterplot are as shown below:





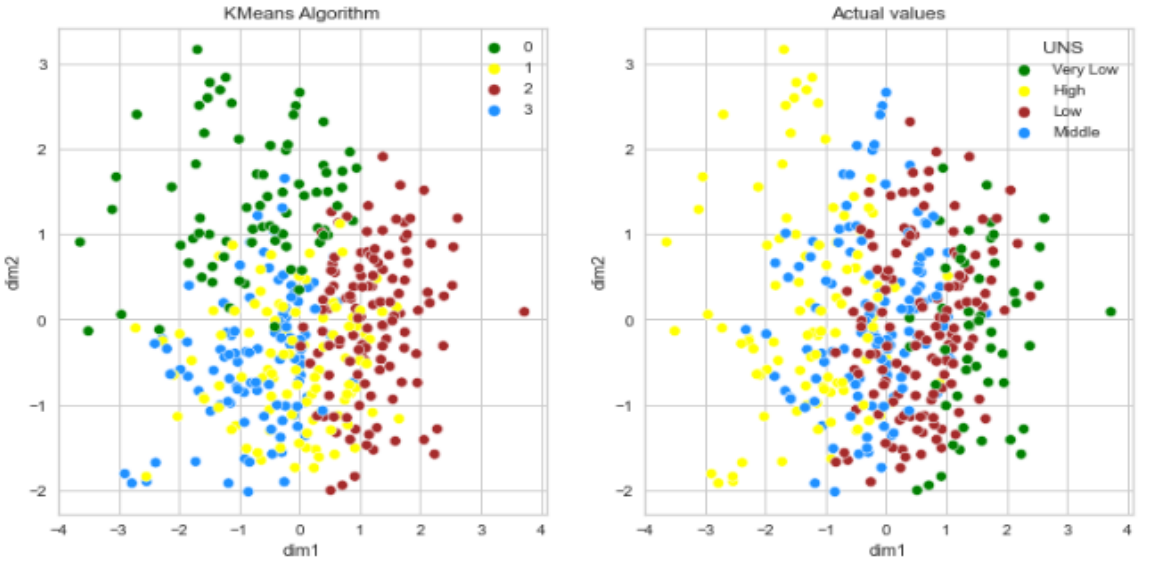


Comparison of results with already established classes

To compare the performance of both numbers of k, a scatterplot of the true classes is plotted and compared for both values.

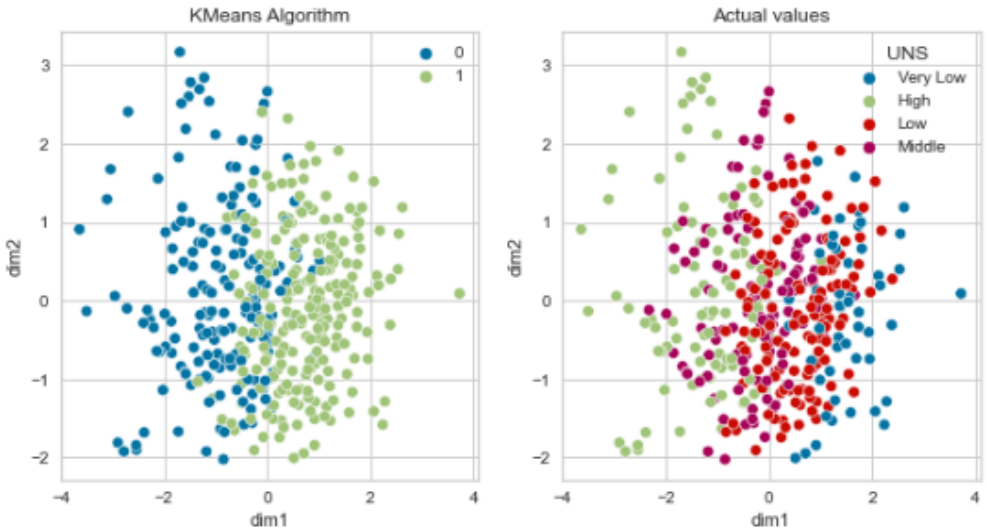
For k=4

```
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, hue=clusters4, palette = pal)
plt.title('KMeans Algorithm')
plt.subplot(1,2,2)
sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, hue=dataset.UNS, palette = pal)
plt.title('Actual values')
plt.show()
```



For k=2:

In [52]:
```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, hue=clusters2)
plt.title('KMeans Algorithm')
plt.subplot(1,2,2)
sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, hue=dataset.UNS)
plt.title('Actual values')
plt.show()
```



From the results shown above, it can be seen that the clustering when k=4 shows a closer resemblance to the scatterplot of the already established classes than the clustering when k=2. Therefore k=4 is the ideal number for this report.

Relevant Literature

k-Means is the most popular clustering algorithm out there but it seems to have a glaring flaw; the initial seed selection (centroids) around which the means of the nearest data point attributes are calculated.

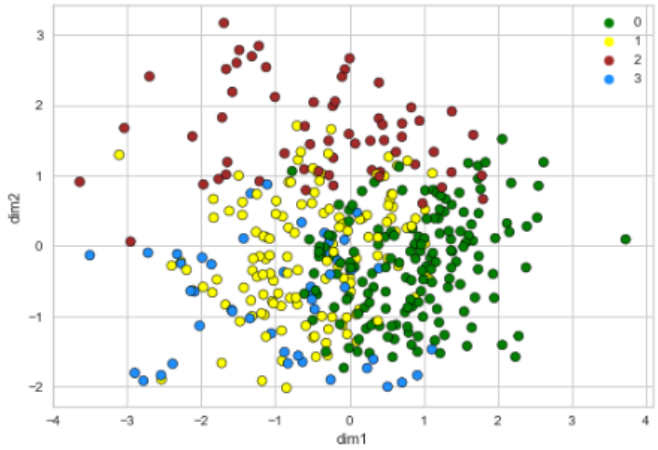
Furthermore, it is said to suffer from high computational complexity when dealing with datasets with a high number of observations. (Qi, Yu, Wang, Liu, & Wang, 2017)

The results of the experiment support the above statements, as it can be seen that because the initial clusters were overlapping, the centroids determined are fuzzy, and this leads to not-so-ideal results.

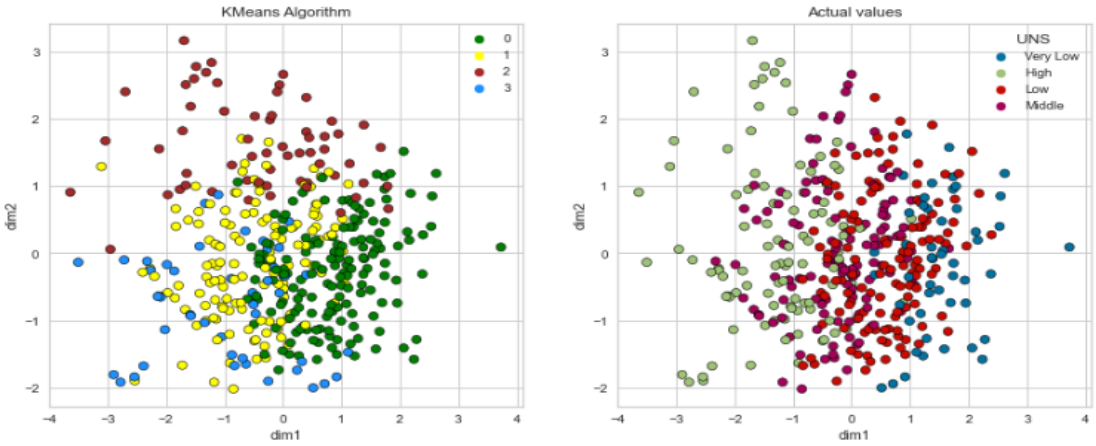
Hierarchical Clustering

The agglomerative function was initialized, and the clustering was predicted on our dataset with the parameters set as shown in the code and results below:

```
In [100]: hierarchical = AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='ward')
          clusters_H = hierarchical.fit_predict(X)

In [101]: np.unique(clusters_H, return_counts=True)

Out[101]: (array([0, 1, 2, 3], dtype=int64), array([160, 135,  60,  48], dtype=int64))

In [103]: sns.scatterplot(x=dimensions.dim1, y=dimensions.dim2, hue=clusters_H, edgecolor='black', palette = pal)

Out[103]: <AxesSubplot:xlabel='dim1', ylabel='dim2'>
```



A scatterplot of this clustering and that of the class in the original dataset is shown side-by-side for comparison. See below:



Relevant Literature

(Firdaus & Uddin, 2015) and the results of this experiment seem to agree when it was said that Hierarchical clustering is more advantageous because the optimal quantity of clusters can be obtained intrinsically from the model itself, but its drawback is that it is not suitable for a large amount of data owing to its large computational time and complexity.

Results analysis and discussion

Performance metrics Used:

At the end of this experiment, 3 performance metrics were used to evaluate the performance of the clustering algorithms.

Silhouette Score: This ranges from -1 to 1, and is measured using the average intra-cluster distance (a) and the average nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). For clarification, b is the distance between a sample and the closest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if the number of labels is 2 <= n_labels <= n_samples - 1.

Homogeneity Score: This score ranges from 0 to 1 and measures if the clustering outcome contains only data points that are members of a single class.

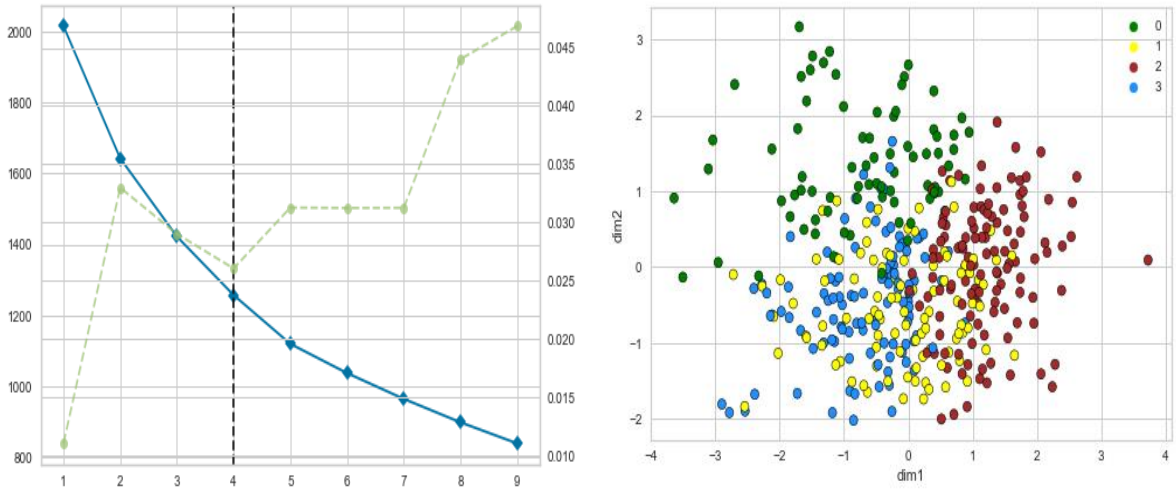
Completeness Score: This score computes the completeness metric of a cluster labeling given ground truth. It is 100% complete if all the observations that are members of a given class are identified as elements of the same cluster. It has a range from 0 to 1

Physical evaluation of the Scatterplot: Even though this is not a standard metric, it was used to gauge, at first glance, the performance of the clustering algorithms.

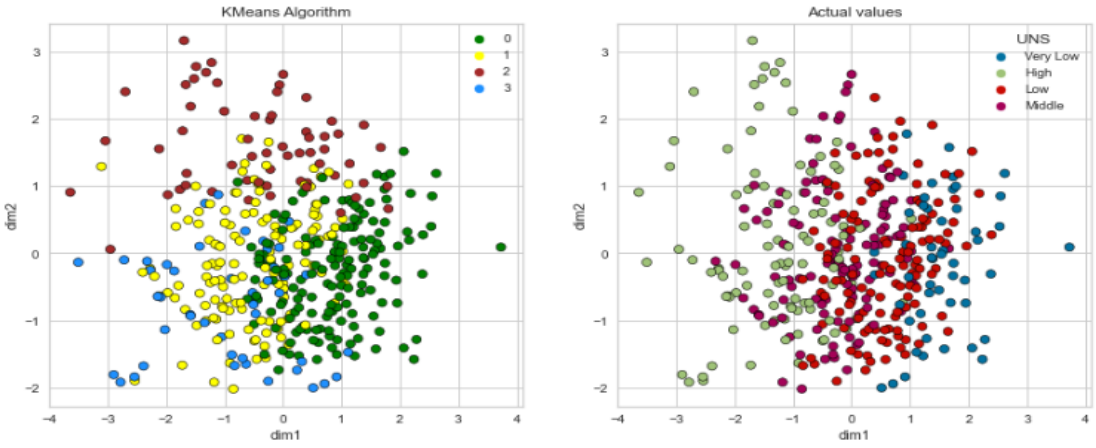
These metrics were used because they give a quantified standard measure of the performance of the models.

Results Presentation

The results of the **k-means** clustering are as shown below:



While that of the hierarchical model is as below:



Models Performance Comparison

Below is a comparison of the models' performance.

	K-Means Score	Hierarchical Score
Silhouette Score	0.173730	0.147698
Homogeneity Score	0.212611	0.256008
Completeness Score	0.206558	0.268694

It can be seen that the K-Means model performed better in the Silhouette score, but the Hierarchical model showed a better Homogeneity and completeness score. As a result, from this experiment, it can be said that the Hierarchical model performed better overall.

The K-means might be said to be somewhat struggling because the classes already established were not perfectly separated (they were nested) initially. As a result, it becomes difficult for the k-Means model to separate the classes effectively. As the shape is not perfectly spherical, it is difficult for the k-means to determine the accurate centroid and hence clusters.

The hierarchical model performed better because it uses the Ward linkage which minimizes the variance within each cluster and projects a clearer plot, but it has the disadvantage of requiring the computation and storage of (n x n, where n is the number of observations) distance matrix. If the data is large, the process will be slow and very expensive.

Ethical, legal, and professional considerations

Some considerations made in the

1. The dataset poses no risks to individuals or organisations as names and personal information have been excluded.
2. The data involved was collected from legitimate, publicly available means.
3. All parties involved in the collection of the dataset were duly listed

Conclusion

The objective of the experiment was to compare the results obtained by both clustering models and compare them with the class already established in the dataset. This objective has been met and it can be seen, from the performance metrics that the hierarchical model performed better than the k-means model.

Clustering is important in everyday life processes as the need to categorize elements into clusters helps in making characterizations and assigning attributes, with which appropriate actions can be taken.