

Aim: Experiment to implement any one of the clustering algorithms using Rapid Miner and Python.

Theory:

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

Clustering Algorithms

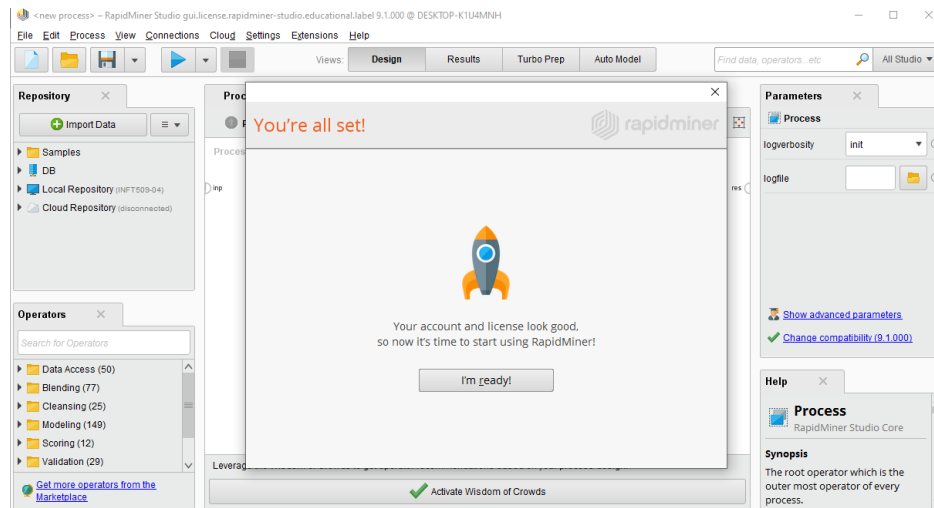
The Clustering algorithms can be divided based on their models that are explained above. There are different types of clustering algorithms published, but only a few are commonly used. The clustering algorithm is based on the kind of data that we are using.

Here we are discussing mainly popular Clustering algorithms that are widely used in machine learning:

1. **K-Means algorithm:** The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of $O(n)$.
2. **Mean-shift algorithm:** Mean-shift algorithm tries to find the dense areas in the smooth density of data points. It is an example of a centroid-based model that works on updating the candidates for centroid to be the center of the points within a given region.
3. **DBSCAN Algorithm:** It stands for Density-Based Spatial Clustering of Applications with Noise. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.

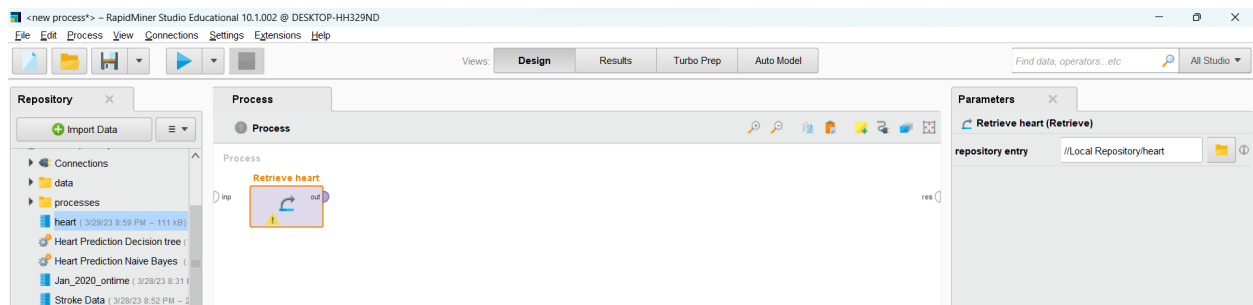
Implementation using K-Means algorithm:

After the successful installation of RapidMiner:

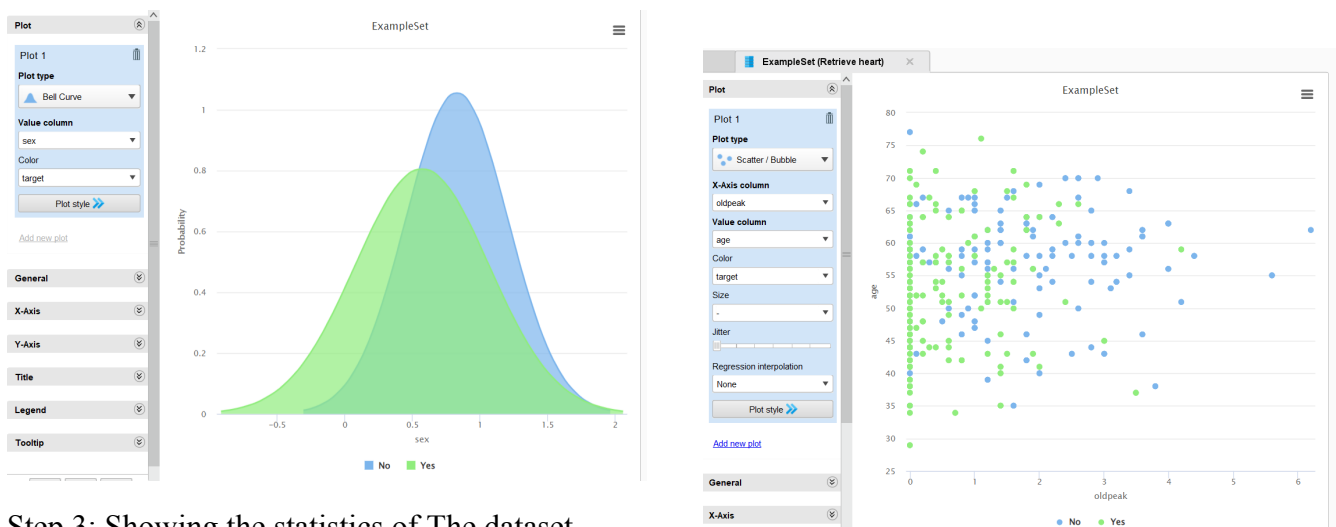


Step 1: Select a blank process

Dataset is imported



Step 2 : Visualizing the given Dataset in scatterplot and other kinds of Visualization techniques.

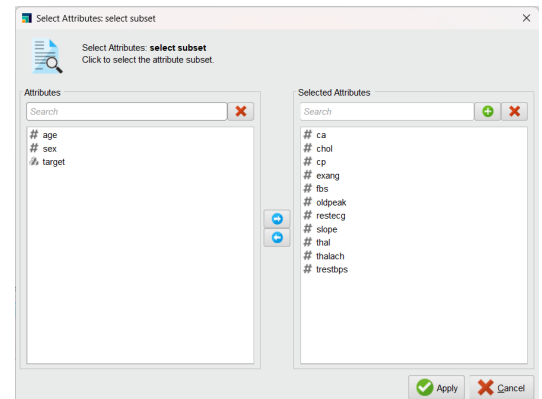
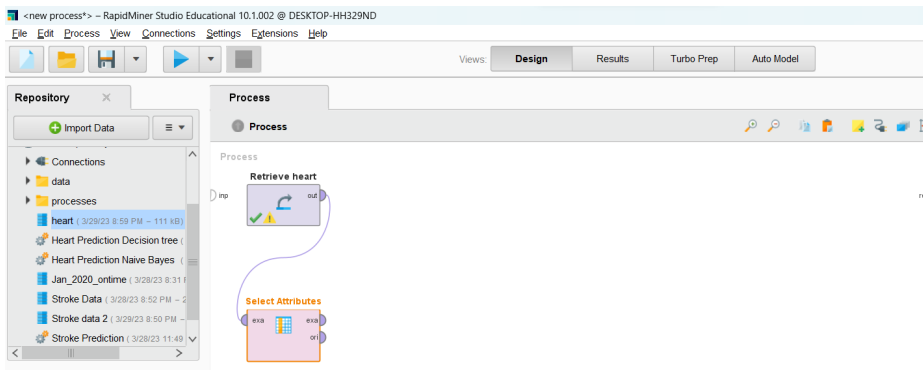


Step 3: Showing the statistics of The dataset

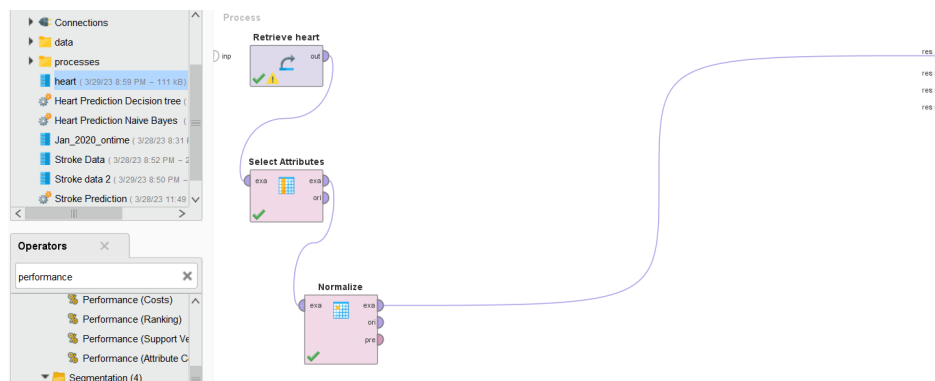
Name	Type	Missing	Min	Max
age	Integer	0	29	77
sex	Integer	0	0	1
cp	Integer	0	0	3
trestbps	Integer	0	94	200
chol	Integer	0	126	564
fbs	Integer	0	0	1
restecg	Integer	0	0	2
thalach	Integer	0	71	202
exang	Integer	0	0	1

Showing attributes 1 - 14 Examples: 1,025 Special Attributes: 0 Regular Attributes: 14

Step 4: Adding Select attribute operator and connect it with the Dataset
Then we will select the required attributes from our dataset as follows.



Step 5: Now import normalize operator as shown below
We will connect as shown in below



After running it will show:

The screenshot shows the Orange3 interface. On the left, a data table is displayed with 17 rows and 9 columns. The columns are: Row No., age, sex, cp, trestbps, chol, fbs, restecg, and thalac. The data is as follows:

Row No.	age	sex	cp	trestbps	chol	fbs	restecg	thalac
1	-0.268	0.661	-0.915	-0.377	-0.659	-0.419	0.891	0.821
2	-0.158	0.661	-0.915	0.479	-0.833	2.386	-1.004	0.256
3	1.716	0.661	-0.915	0.764	-1.396	-0.419	0.891	-1.046
4	0.724	0.661	-0.915	0.936	-0.833	-0.419	0.891	0.517
5	0.834	-1.511	-0.915	0.365	0.930	2.386	0.891	-1.874
6	0.393	-1.511	-0.915	-1.805	0.039	-0.419	-1.004	-1.176
7	0.393	0.661	-0.915	-1.005	1.396	-0.419	2.785	-0.396
8	0.062	0.661	-0.915	1.621	0.833	-0.419	-1.004	-0.176
9	-0.930	0.661	-0.915	-0.663	0.058	-0.419	-1.004	-0.222
10	-0.048	0.661	-0.915	-0.549	0.775	-0.419	-1.004	-1.436
11	1.826	-1.511	-0.915	-1.120	-1.880	-0.419	0.891	-1.046
12	-1.260	-1.511	-0.915	0.022	1.841	2.386	-1.004	-0.576
13	-2.252	-1.511	0.056	-0.777	-0.698	-0.419	0.891	1.864
14	-0.379	0.661	-0.915	0.479	1.008	-0.419	0.891	-1.176
15	-0.268	0.661	-0.915	-0.206	-0.814	2.386	0.891	0.299
16	-2.252	-1.511	0.056	-0.777	-0.698	-0.419	0.891	1.864
17	-0.379	-1.511	1.027	0.479	1.202	-0.419	-1.004	-0.306

On the right, the 'Import Data' panel shows a list of training resources, including 'heart', 'Heart Prediction Decision tree', 'Heart Prediction Naive Bayes', 'Jan_2020_ontime', 'Stroke Data', 'Stroke data 2', and 'Stroke Prediction'.

Step 6: Now drag K Means from Clustering

Step 7: Select respecting k value that you want

The screenshot shows the Orange3 interface with a workflow for K-Means clustering. The workflow consists of three widgets: 'Retrieve heart', 'Select Attributes', and 'Normalize', followed by a 'Clustering' widget. The 'Clustering' widget is configured with 'k' set to 4, 'max runs' set to 10, and 'measure types' set to 'BregmanDivergences'. The 'add cluster attribute' checkbox is checked. The 'Operators' panel on the left shows the 'kmeans' widget selected. The 'Help' panel on the right shows the 'k-Means' widget documentation.

After running, the following output will be displayed.

The screenshot shows the Orange3 interface with the output of the K-Means clustering workflow. The 'ExampleSet (Clustering)' widget displays a table with 17 rows and 9 columns. The columns are: Row No., id, cluster, cp, trestbps, chol, fbs, restecg, and thalac. The data is as follows:

Row No.	id	cluster	cp	trestbps	chol	fbs	restecg	thalac
1	1	cluster_2	-0.915	-0.377	-0.659	-0.419	0.891	0.821
2	2	cluster_1	-0.915	0.479	-0.833	2.386	-1.004	0.256
3	3	cluster_3	-0.915	0.764	-1.396	-0.419	0.891	-1.046
4	4	cluster_2	-0.915	0.936	-0.833	-0.419	0.891	0.517
5	5	cluster_0	-0.915	0.365	0.930	2.386	0.891	-1.874
6	6	cluster_3	-0.915	-1.805	0.039	-0.419	-1.004	-1.176
7	7	cluster_1	-0.915	-1.005	1.396	-0.419	2.785	-0.396
8	8	cluster_3	-0.915	1.621	0.833	-0.419	-1.004	-0.176
9	9	cluster_2	-0.915	-0.663	0.058	-0.419	-1.004	-0.222
10	10	cluster_3	-0.915	-0.549	0.775	-0.419	-1.004	-1.436
11	11	cluster_3	-0.915	-1.120	-1.880	-0.419	0.891	-1.046
12	12	cluster_1	-0.915	0.022	1.841	2.386	-1.004	-0.576
13	13	cluster_2	0.056	-0.777	-0.698	-0.419	0.891	1.864
14	14	cluster_1	-0.915	0.479	1.008	-0.419	0.891	-1.176
15	15	cluster_0	-0.915	-0.206	-0.814	2.386	0.891	0.299
16	16	cluster_2	0.056	-0.777	-0.698	-0.419	0.891	1.864
17	17	cluster_2	1.027	0.479	1.202	-0.419	-1.004	-0.306

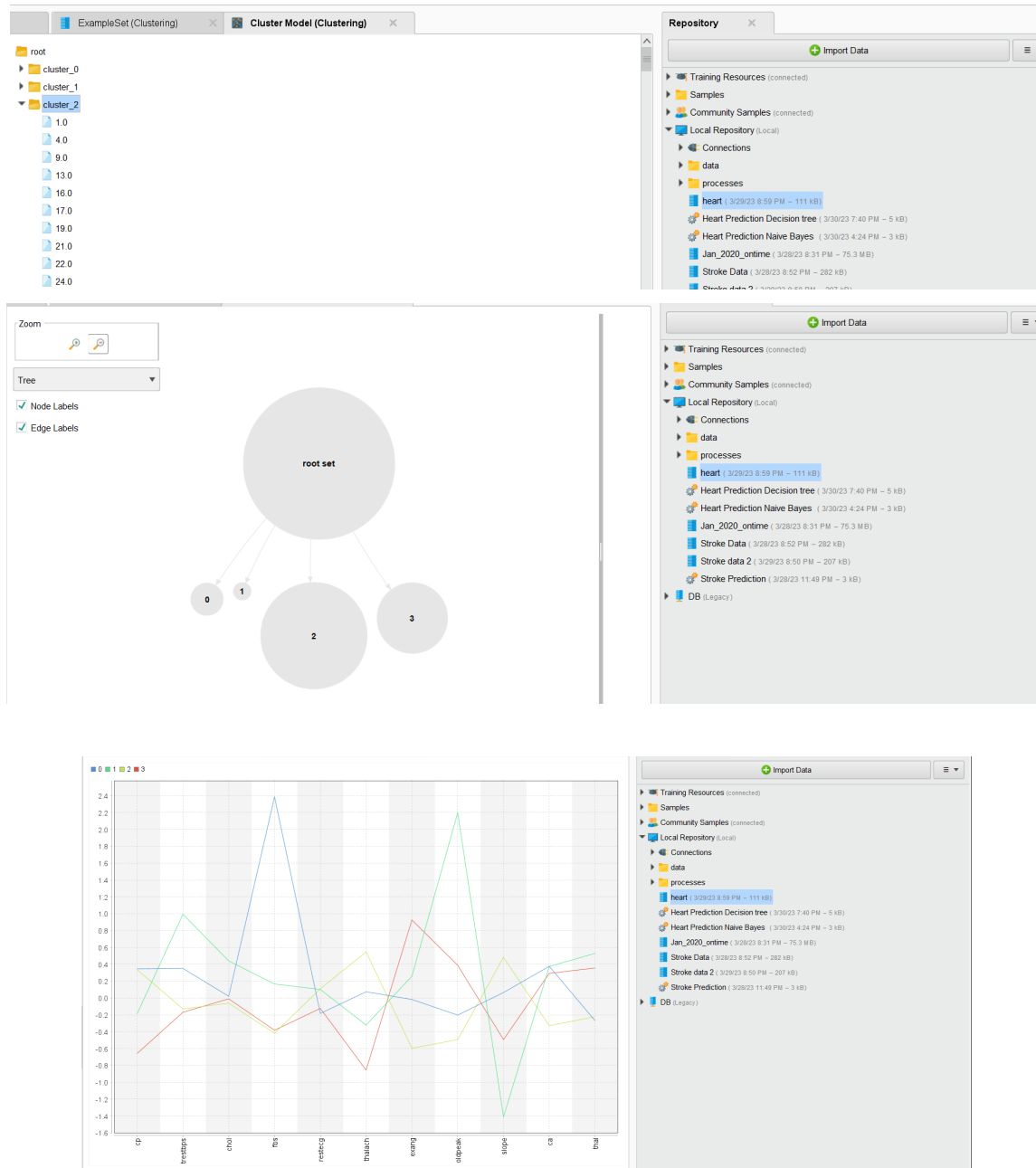
On the right, the 'Repository' panel shows a list of training resources, including 'heart', 'Heart Prediction Decision tree', 'Heart Prediction Naive Bayes', 'Jan_2020_ontime', 'Stroke Data', 'Stroke data 2', and 'Stroke Prediction'.

Summary of Clustering Model

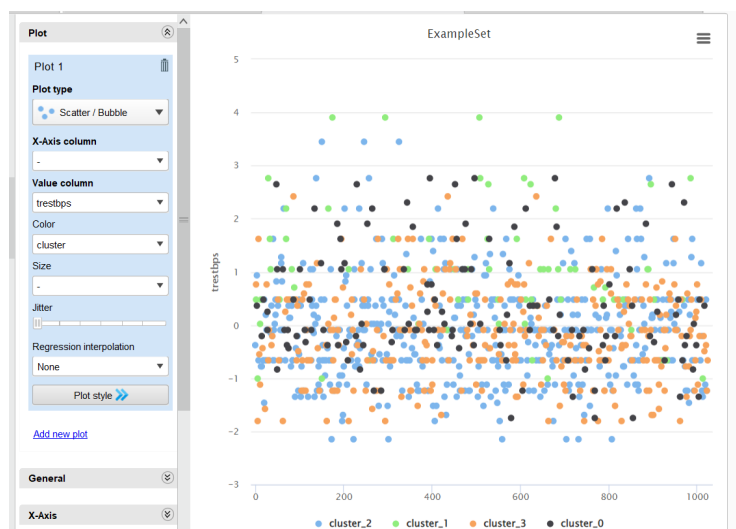
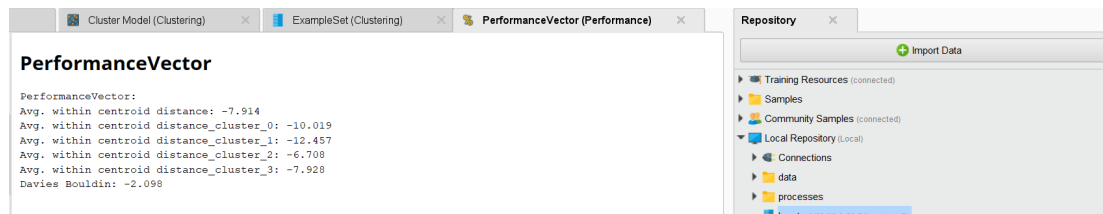
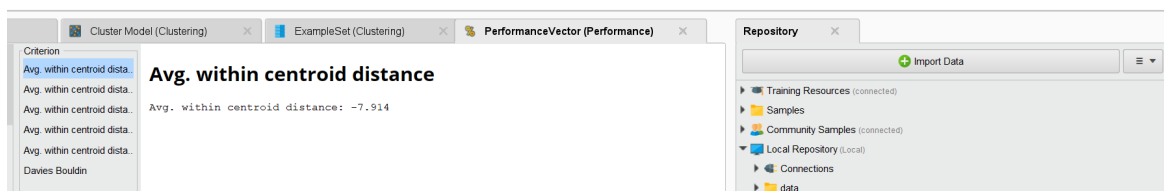
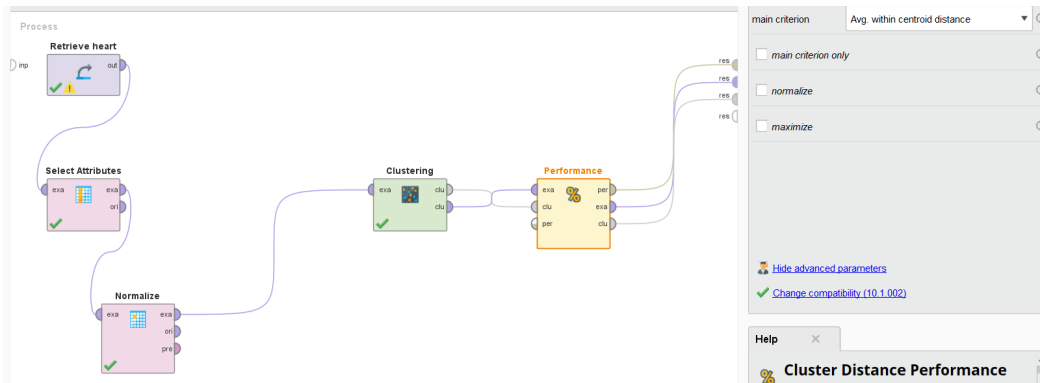
Cluster Model

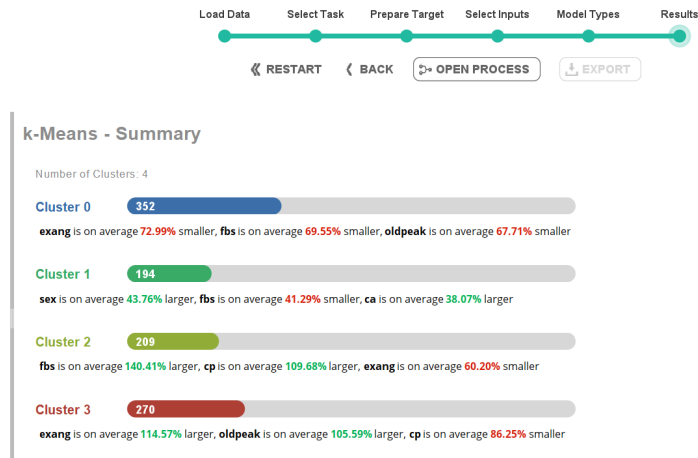
Cluster 0: 134 items
Cluster 1: 72 items
Cluster 2: 509 items
Cluster 3: 310 items
Total number of items: 1025

This the folder view of Clusters



Step 7: Search performance distance and then arrange as shown in below





K Means Clustering using Python

- Importing dataset

```
import numpy as np # to handle numeric data
import matplotlib.pyplot as plt # for visualization
import pandas as pd # for handling dataframe
#Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

```
[2] ourData = pd.read_csv('heart.csv') # read the data
ourData.head() # print the
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

- Importing K means from sklearn

```
wcss= []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

```
[3] from sklearn.cluster import KMeans
```

```
X = ourData[['cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']].copy()
```

```
[5] for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=0)
    kmeans.fit(X)
```

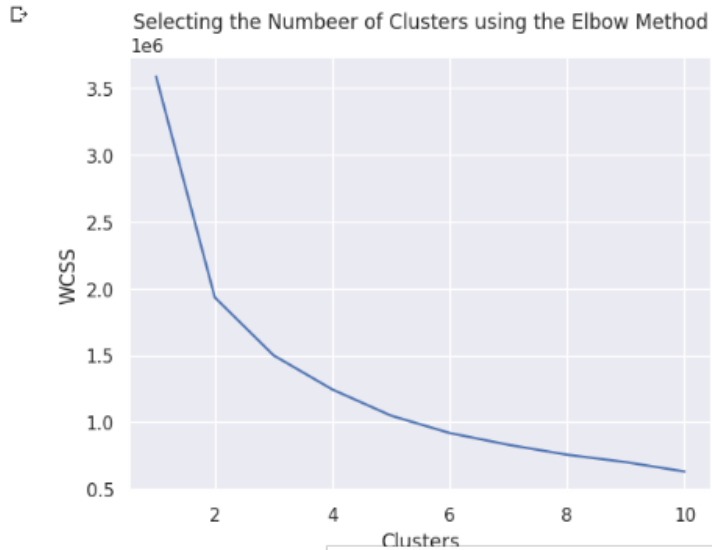
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[8] sns.set()
```

- Selecting number of clusters using wcss

```
[8] sns.set()
```

```
plt.plot(range(1, 11), wcss)
plt.title('Selecting the Number of Clusters using the Elbow Method')
plt.xlabel('Clusters')
plt.ylabel('WCSS')
plt.show()
```



- Using Gaussian Mixture to display the clusters

```
[10] from sklearn.mixture import GaussianMixture
```

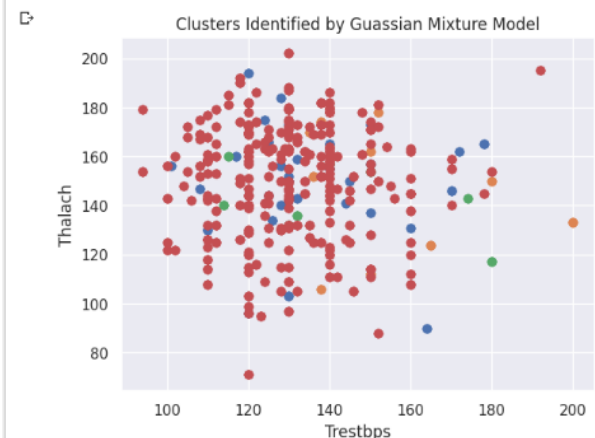
```
[15] from sklearn.mixture import GaussianMixture
n_clusters = 4
gmm_model = GaussianMixture(n_components=n_clusters)
gmm_model.fit(X)
```

```
GaussianMixture
GaussianMixture(n_components=4)
```

```
cluster_labels = gmm_model.predict(X)
X = pd.DataFrame(X)
X['cluster'] = cluster_labels
```

```
for k in range(0, n_clusters):
    data = X[X["cluster"]==k]
    plt.scatter(data["trestbps"], data["thalach"])

plt.title("Clusters Identified by Gaussian Mixture Model")
plt.ylabel("Thalach")
plt.xlabel("Trestbps")
plt.show()
```



Conclusion: We have successfully implemented K Means clustering algorithm using Rapid Miner and Python and compared their results.