

## **EXPERIMENT 8**

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

### **Theory:**

#### **What is Clustering in Data Mining?**

In clustering, a group of different data objects is classified as similar objects. One group means a cluster of data. Data sets are divided into different groups in the cluster analysis, which is based on the similarity of the data. After the classification of data into various groups, a label is assigned to the group. It helps in adapting to the changes by doing the classification.

So if we were to define clustering in data mining, then we can say that the process of clustering in data mining is basically a set of abstract objects into groups of similar objects. The process of dividing and storing them in these groups is known as cluster analysis.

#### **What is Cluster Analysis in Data Mining?**

Cluster Analysis in Data Mining means that to find out the group of objects which are similar to each other in the group but are different from the object in other groups. In the process of clustering in data analytics, the sets of data are divided into groups or classes based on data similarity. Then each of these classes is labeled according to their data types. Going through clustering in data mining examples can help you understand the analysis more extensively.

#### **Applications of Data Mining Cluster Analysis**

There are many uses of Data clustering analysis such as image processing, data analysis, pattern recognition, market research and many more. Using Data clustering, companies can discover new groups in the database of customers. They can also classify the existing customer base into distinct groups depending on the patterns of their purchases. Classification of data can also be done based on patterns of purchasing.

Taxonomy or the classification of animals with the help of cluster analysis is very common in the field of biology. Clustering can help identify and group species with similar genetic features and functionalities and also give us an understanding of some of the most commonly found inherent structures of specific populations or species. Areas are identified using the clustering in data mining. In the database of earth observation, lands are identified which are similar to each other.

Also read: Free data structures and algorithm course!

Based on geographic location, value and house type, a group of houses are defined in the city. Clustering in data mining helps in the discovery of information by classifying the files on the internet. It is also used in detection applications. Fraud in a credit card can be easily detected using clustering in data mining which analyzes the pattern of deception. Read more about the applications of data science in the finance industry.

If someone wants to observe the characteristics of each data cluster, then cluster analysis can act as the tool to help them gain insight into the data clusters.

It helps in understanding each cluster and its characteristics. One can understand how the data is distributed, and it works as a tool in the function of data mining.

### **Requirements of Clustering in Data Mining:**

#### **Interpretability:**

The result of clustering should be usable, understandable and interpretable. The main aim of clustering in data analytics is to make sure haphazard data is stored in groups based on their characteristics similarity.

#### **Helps in dealing with messed up data:**

Usually, the data is messed up and unstructured. It cannot be analyzed quickly, and that is why the clustering of information is so significant in data mining. Grouping can give some structure to the data by organizing it into groups of similar data objects.

It becomes more comfortable for the data expert in processing the data and also discover new things. Analyzing data that has already been classified and labelled through clustering is much easier than analyzing unstructured data. It also leaves less room for error.

#### **High Dimensional:**

Data clustering is also able to handle the data of high dimension along with the data of small size. The clustering algorithms in data mining need to be able to handle any dimension of data.

#### **Attribute shape clusters are discovered:**

Clustering algorithms in data mining should be able to detect arbitrary shaped clusters. These algorithms should not be limited by only being able to find smaller, spherical clusters.

### **Dealing with Erroneous Data:**

Usually, databases carry a lot of erroneous, noisy or absent data. If the algorithm being used during clustering is very sensitive to this type of anomaly, then it can lead to low-quality clusters. That is why it is very important that your clustering algorithm can handle this type of data without problems.

### **Algorithm Usability with multiple data kind:**

Many different kinds of data can be used with algorithms of clustering. The data can be like binary data, categorical and interval-based data.

### **Clustering Scalability:**

The database usually is enormous to deal with. The algorithm should be scalable to handle an extensive database, so it needs to be scalable.

## **Data Mining Clustering Methods:**

### **1. Partitioning Clustering Method**

In this method, let us say that the “m” partition is done on the “p” objects of the database. A cluster will be represented by each partition and  $m < p$ . K is the number of groups after the classification of objects. There are some requirements which need to be satisfied with this Partitioning Clustering Method and they are: –

One objective should only belong to only one group.

There should be no group without even a single purpose.

There are some points which should be remembered in this type of Partitioning Clustering Method which are:

There will be an initial partitioning if we already give no. of a partition (say m).

There is one technique called iterative relocation, which means the object will be moved from one group to another to improve the partitioning.

## **2. Hierarchical Clustering Methods**

Among the many different types of clustering in data mining, In this hierarchical clustering method, the given set of an object of data is created into a kind of hierarchical decomposition. The formation of hierarchical decomposition will decide the purposes of classification. There are two types of approaches for the creation of hierarchical decomposition, which are: –

### **a. Divisive Approach**

Another name for the Divisive approach is a top-down approach. At the beginning of this method, all the data objects are kept in the same cluster. Smaller clusters are created by splitting the group by using the continuous iteration. The constant iteration method will keep on going until the condition of termination is met. One cannot undo after the group is split or merged, and that is why this method is not so flexible.

### **b. Agglomerative Approach**

Another name for this approach is the bottom-up approach. All the groups are separated in the beginning. Then it keeps on merging until all the groups are merged, or condition of termination is met.

There are two approaches which can be used to improve the Hierarchical Clustering Quality in Data Mining which are: –

One should carefully analyze the linkages of the object at every partitioning of hierarchical clustering.

One can use a hierarchical agglomerative algorithm for the integration of hierarchical agglomeration. In this approach, first, the objects are grouped into micro-clusters. After grouping data objects into microclusters, macro clustering is performed on the microcluster.

## **3. Density-Based Clustering Method**

In this method of clustering in Data Mining, density is the main focus. The notion of mass is used as the basis for this clustering method. In this clustering method, the cluster will keep on growing continuously. At least one number of points should be there in the radius of the group for each point of data.

#### **4. Grid-Based Clustering Method**

In this type of Grid-Based Clustering Method, a grid is formed using the object together. A Grid Structure is formed by quantifying the object space into a finite number of cells.

Advantage of Grid-based clustering method: –

Faster time of processing: The processing time of this method is much quicker than another way, and thus it can save time.

This method depends on the no. of cells in the space of quantized each dimension.

#### **5. Model-Based Clustering Methods**

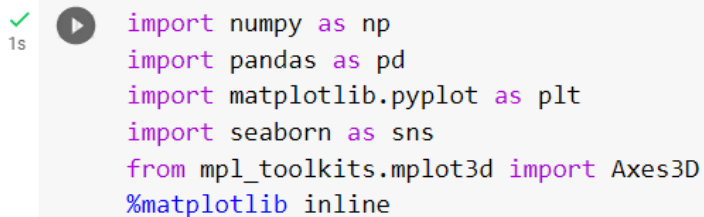
In this type of clustering method, every cluster is hypothesized so that it can find the data which is best suited for the model. The density function is clustered to locate the group in this method.

#### **6. Constraint-Based Clustering Method**

Application or user-oriented constraints are incorporated to perform the clustering. The expectation of the user is referred to as the constraint. In this process of grouping, communication is very interactive, which is provided by the restrictions.

### **Screenshots of implementation:**

#### **K-Means:(inPython)**



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

1. Import the csv file insurance.csv.

✓  
0s

```
[2] data=pd.read_csv("/insurance_policy.csv")
```

✓ [3] data.head()

	ID	City_Code	Region_Code	Accomodation_Type	Reco_Insurance_Type	Upper_Age	Lower_Age	Is_Spouse	Health_Indicator	Holding_Policy_Duration	Holding_Policy_Type	Reco_Poli
0	1	C3	3213	Rented	Individual	36	36	No	X1	14+		3.0
1	2	C5	1117	Owned	Joint	75	22	No	X2	NaN		NaN
2	3	C5	3732	Owned	Individual	32	32	No	NaN	1		1.0
3	4	C24	4378	Owned	Joint	52	48	No	X1	14+		3.0
4	5	C8	2190	Rented	Individual	44	44	No	X2	3		1.0



✓ [4] data.corr()

	ID	Region_Code	Upper_Age	Lower_Age	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response
ID	1.000000	-0.000465	-0.001725	0.001101	0.006649	-0.011922	-0.002350	0.005159
Region_Code	-0.000465	1.000000	-0.005649	-0.005928	0.011599	-0.064616	-0.010797	0.001121
Upper_Age	-0.001725	-0.005649	1.000000	0.921392	0.141890	0.024469	0.792689	0.002772
Lower_Age	0.001101	-0.005928	0.921392	1.000000	0.118028	0.020488	0.615739	-0.002099
Holding_Policy_Type	0.006649	0.011599	0.141890	0.118028	1.000000	0.079773	0.121342	0.009297
Reco_Policy_Cat	-0.011922	-0.064616	0.024469	0.020488	0.079773	1.000000	0.060349	0.113717
Reco_Policy_Premium	-0.002350	-0.010797	0.792689	0.615739	0.121342	0.060349	1.000000	0.007943
Response	0.005159	0.001121	0.002772	-0.002099	0.009297	0.113717	0.007943	1.000000

✓  
1s

```
[5] plt.figure(figsize=(10,6))
sns.set(style = 'whitegrid')
sns.distplot(data['Reco_Policy_Cat'])
plt.title('Distribution of Reco_Policy_Cat ', fontsize=20)
plt.xlabel('Range of Reco_Policy_Cat')
plt.ylabel('Count')
```

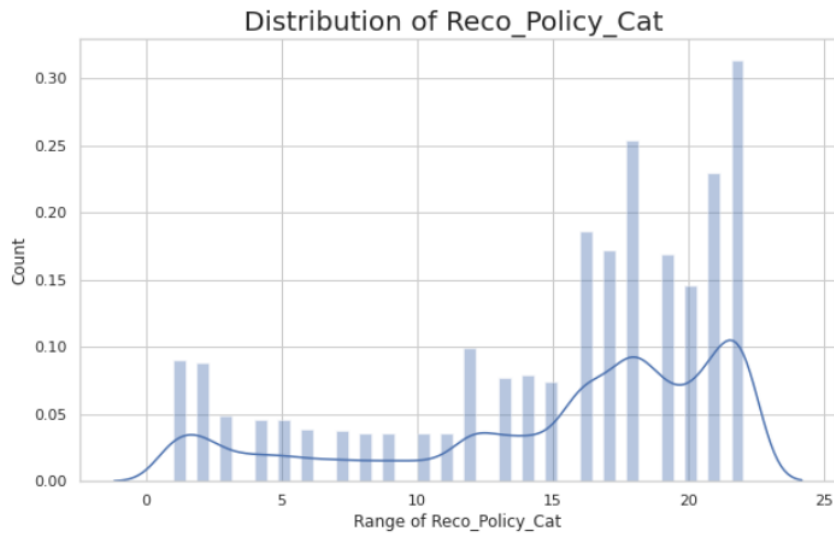
✓ [5] <ipython-input-5-7558f6486193>:3: UserWarning:

1a `distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data['Reco_Policy_Cat'])
Text(0, 0.5, 'Count')
```



✓ [6] `plt.figure(figsize = (10,6))`  
1a `sns.set(style = 'whitegrid')`  
`sns.distplot(data ['Upper_Age'])`  
`plt.title('Distribution of Upper_Age', fontsize = 20)`  
`plt.xlabel('Range of Upper_Age')`  
`plt.ylabel('Count')`

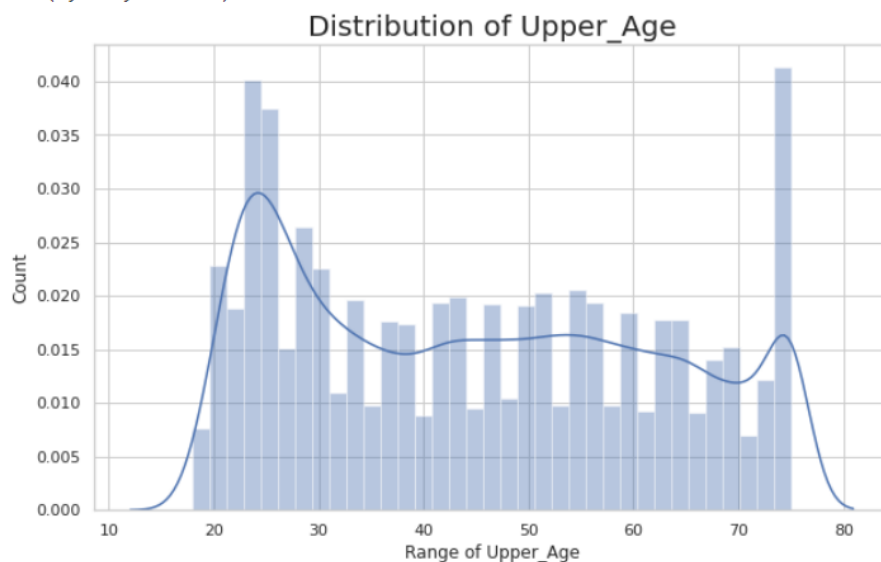
✓ 1s <ipython-input-6-90e1a9d4135f>:3: UserWarning:

🔗 `distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data ['Upper_Age'])
Text(0, 0.5, 'Count')
```



✓ 1s [7] 

```
plt.figure(figsize = (10,6))
sns.set(style = 'whitegrid')
sns.distplot(data['Reco_Policy_Premium'])
plt.title('Distribution of Reco_Policy_Premium')
plt.xlabel('Range of Reco_Policy_Premium')
plt.ylabel('Count')
```



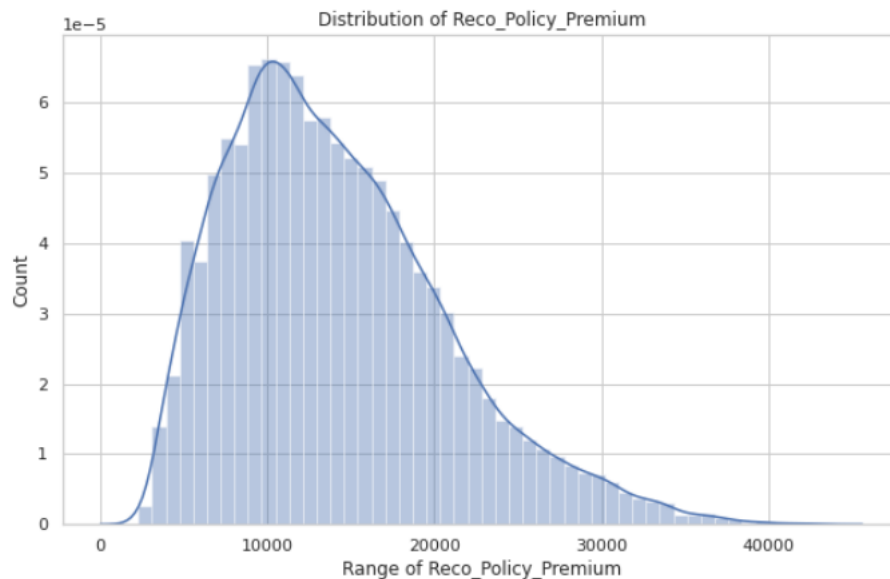
✓ 1s <ipython-input-7-b67b6c9073fa>:3: UserWarning:

↳ ``distplot` is a deprecated function and will be removed in seaborn v0.14.0.`

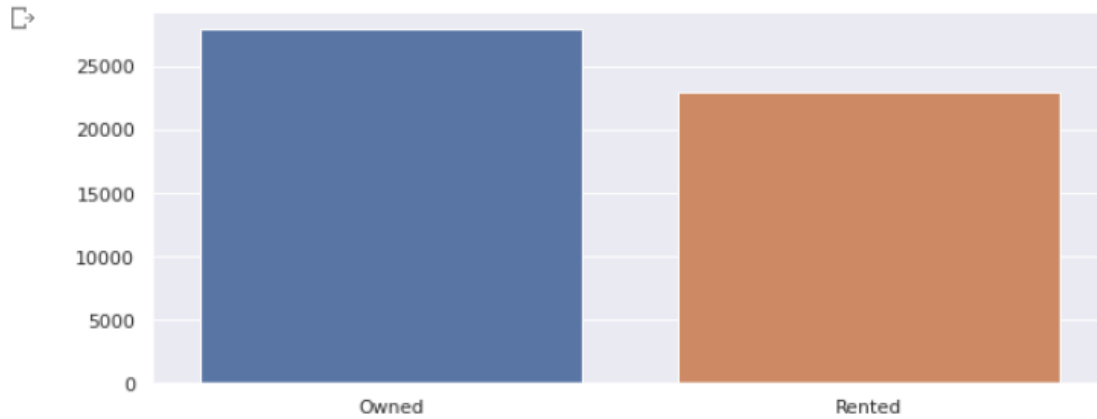
Please adapt your code to use either ``displot`` (a figure-level function with similar flexibility) or ``histplot`` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data['Reco_Policy_Premium'])
Text(0, 0.5, 'Count')
```



```
✓ 2s [8] accomodation = data.Accomodation_Type.value_counts()
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.barplot(x=accomodation.index, y=accomodation.values)
plt.show()
```

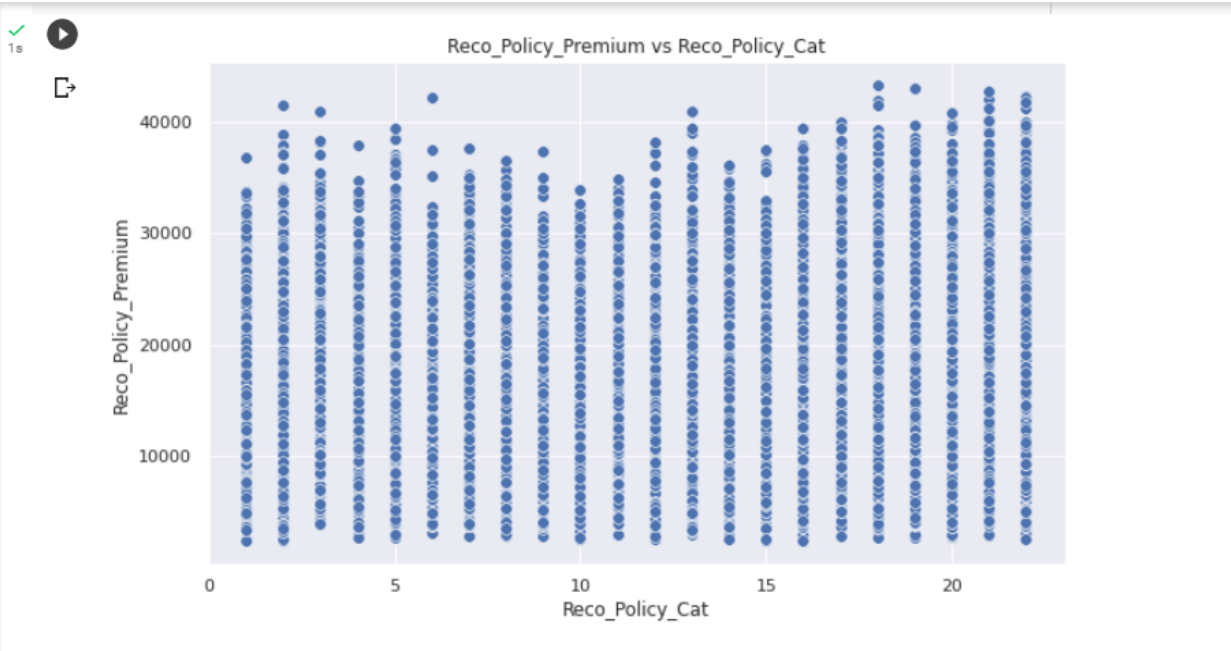


```
df1 = data[["ID", "City_Code", "Region_Code", "Accommodation_Type", "Reco_Insurance_Type", "Upper_Age", "Lower_Age", "Is_Spouse",  
            "Health_Indicator", "Holding_Policy_Duration", "Holding_Policy_Type", "Reco_Policy_Cat", "Reco_Policy_Premium", "Response"]]  
X = df1[["Reco_Policy_Cat", "Reco_Policy_Premium"]]
```

✓ [10] X.head()  
0s

	Reco_Policy_Cat	Reco_Policy_Premium
0	22	11628.0
1	22	30510.0
2	22	7450.0
3	22	17780.0
4	22	10404.0

```
✓ [10] 1s plt.figure(figsize=(10,6))  
sns.scatterplot(x = 'Reco_Policy_Cat' , y = 'Reco_Policy_Premium' , data = X, s=60)  
plt.xlabel('Reco_Policy_Cat')  
plt.ylabel('Reco_Policy_Premium')  
plt.title('Reco_Policy_Premium vs Reco_Policy_Cat')  
plt.show()
```



0s [12] `from sklearn.cluster import KMeans`

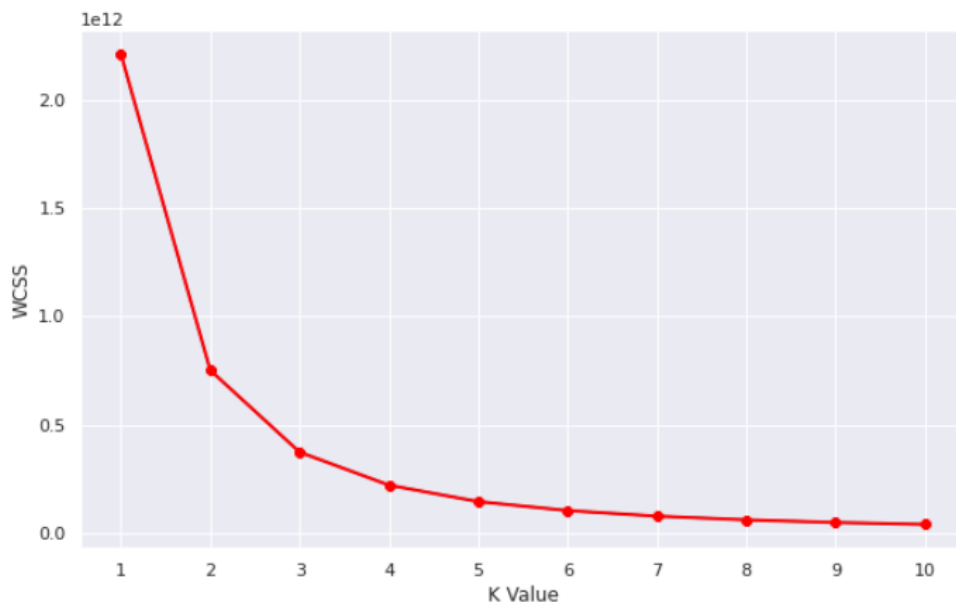
12s

```
wcss=[]
for i in range(1,11):
    km = KMeans(n_clusters = i)
    km.fit(X)
    wcss.append(km.inertia_)
```

`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`  
`/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warnings.warn(`

1s [14] `plt.figure(figsize = (10,6))`  
`plt.plot(range(1,11),wcss)`  
`plt.plot(range(1,11),wcss, linewidth=2, color="red", marker="8")`  
`plt.xlabel("K Value")`  
`plt.xticks(np.arange(1,11,1))`  
`plt.ylabel("WCSS")`  
`plt.show()`

✓ [14]  
1s



✓ [15] km1 = KMeans(n\_clusters = 5)  
0s

✓ [17] km1.fit(X)  
1s

/usr/local/lib/python3.9/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of 'n\_init' will change from 10 to 'auto' in 1.4. Set the value of 'n\_init' explicitly to suppress the warnings.warn(  
KMeans  
KMeans(n\_clusters=5)

✓ [18] y = km1.predict(X)  
0s

✓ [19] df1["label"] = y  
1s


✓ df1.head()  
0s

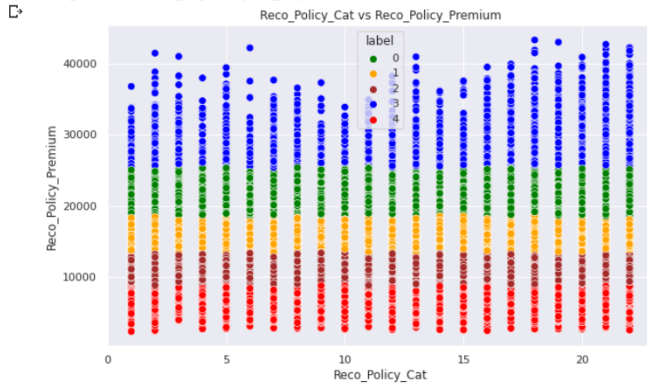
	ID	City_Code	Region_Code	Accommodation_Type	Reco_Insurance_Type	Upper_Age	Lower_Age	Is_Spouse	Health_Indicator	Holding_Policy_Duration	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response
0	1	C3	3213	Rented	Individual	36	36	No	X1	14+	3.0	22	11628.0	0
1	2	C5	1117	Owned	Joint	75	22	No	X2	NaN	NaN	22	30510.0	0
2	3	C5	3732	Owned	Individual	32	32	No	NaN	1	1.0	22	7450.0	1
3	4	C24	4378	Owned	Joint	52	48	No	X1	14+	3.0	22	17780.0	0
4	5	C8	2190	Rented	Individual	44	44	No	X2	3	1.0	22	10404.0	0

```

[21] plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Reco_Policy_Cat',y = 'Reco_Policy_Premium',hue="label",
                palette=['green','orange','brown','blue','red'], legend='full',data = df1,s = 60 )
plt.xlabel('Reco_Policy_Cat')
plt.ylabel('Reco_Policy_Premium')
plt.title('Reco_Policy_Cat vs Reco_Policy_Premium')
plt.show()

```


 /usr/local/lib/python3.9/dist-packages/IPython/core/pylabtools.py:128: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
fig.canvas.print\_figure(bytes\_io, \*\*kw)



```

[22] df2=data[["ID", "City_Code", "Region_Code", "Accommodation_Type", "Reco_Insurance_Type", "Upper_Age", "Lower_Age", "Is_Spouse", "Health_Indicator", "Holding_Policy_Duration", "Holding_Policy_Type", "
X2 = df2[["Upper_Age", "Reco_Policy_Cat", "Reco_Policy_Premium"]]
km2 = KMeans(n_clusters = 5)
y2 = km2.fit_predict(X2)
df2["label"] = y2
df2.head()

```

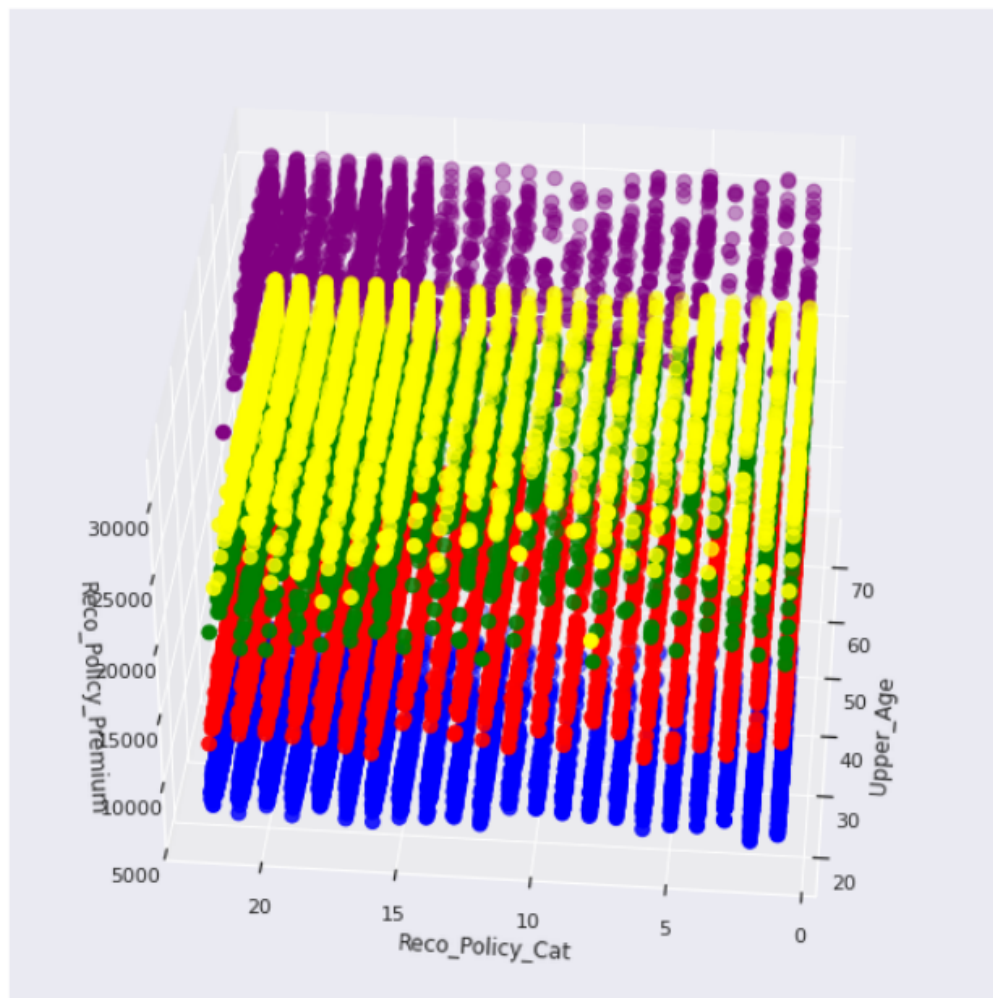
 /usr/local/lib/python3.9/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of 'n\_init' will change from 10 to 'auto' in 1.4. Set the value of 'n\_init' explicitly to suppress the
warnings.warn(

	ID	City_Code	Region_Code	Accommodation_Type	Reco_Insurance_Type	Upper_Age	Lower_Age	Is_Spouse	Health_Indicator	Holding_Policy_Duration	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response
0	1	C3	3213	Rented	Individual	36	36	No	X1	14+	3.0	22	11628.0	0
1	2	C5	1117	Owned	Joint	75	22	No	X2	NaN	NaN	22	30510.0	0
2	3	C5	3732	Owned	Individual	32	32	No	NaN	1	1.0	22	7450.0	1
3	4	C24	4378	Owned	Joint	52	48	No	X1	14+	3.0	22	17780.0	0
4	5	C8	2190	Rented	Individual	44	44	No	X2	3	1.0	22	10404.0	0

```

[23] fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection= "3d")
ax.scatter(df2.Upper_Age[df2.label == 0], df2["Reco_Policy_Cat"][df2.label == 0], df2["Reco_Policy_Premium"][df2.label == 0], c='purple', s=60)
ax.scatter(df2.Upper_Age[df2.label == 1], df2["Reco_Policy_Cat"][df2.label == 1], df2["Reco_Policy_Premium"][df2.label == 1], c='red', s=60)
ax.scatter(df2.Upper_Age[df2.label == 2], df2["Reco_Policy_Cat"][df2.label == 2], df2["Reco_Policy_Premium"][df2.label == 2], c='blue', s=60)
ax.scatter(df2.Upper_Age[df2.label == 3], df2["Reco_Policy_Cat"][df2.label == 3], df2["Reco_Policy_Premium"][df2.label == 3], c='green', s=60)
ax.scatter(df2.Upper_Age[df2.label == 4], df2["Reco_Policy_Cat"][df2.label == 4], df2["Reco_Policy_Premium"][df2.label == 4], c='yellow', s=60)
ax.view_init(35, 185)
plt.xlabel("Upper_Age")
plt.ylabel("Reco_Policy_Cat")
ax.set_zlabel("Reco_Policy_Premium")
plt.show()

```



✓  
0s



```
age1=df2[df2["label"]==1]
print('Number of customer in 1st age group=', len(age1))
print('They are -', age1["Upper_Age"].values)
print("-----")
age2=df2[df2["label"]==2]
print('Number of customer in 2nd age group=', len(age2))
print('They are -', age2["Upper_Age"].values)
print("-----")
age3=df2[df2["label"]==0]
print('Number of customer in 3rd age group=', len(age3))
print('They are -', age3["Upper_Age"].values)
print("-----")
age4=df2[df2["label"]==3]
print('Number of customer in 4th age group=', len(age4))
print('They are -', age4["Upper_Age"].values)
print("-----")
age5=df2[df2["label"]==4]
print('Number of customer in 5th age group=', len(age5))
print('They are -', age5["Upper_Age"].values)
print("-----")
```

✓  
0s

[31]

Number of customer in 1st age group= 7505  
They are - [36 55 32 ... 37 63 24]



-----  
Number of customer in 2nd age group= 6869  
They are - [32 20 34 ... 20 26 22]

-----  
Number of customer in 3rd age group= 2016  
They are - [75 75 52 ... 63 66 71]

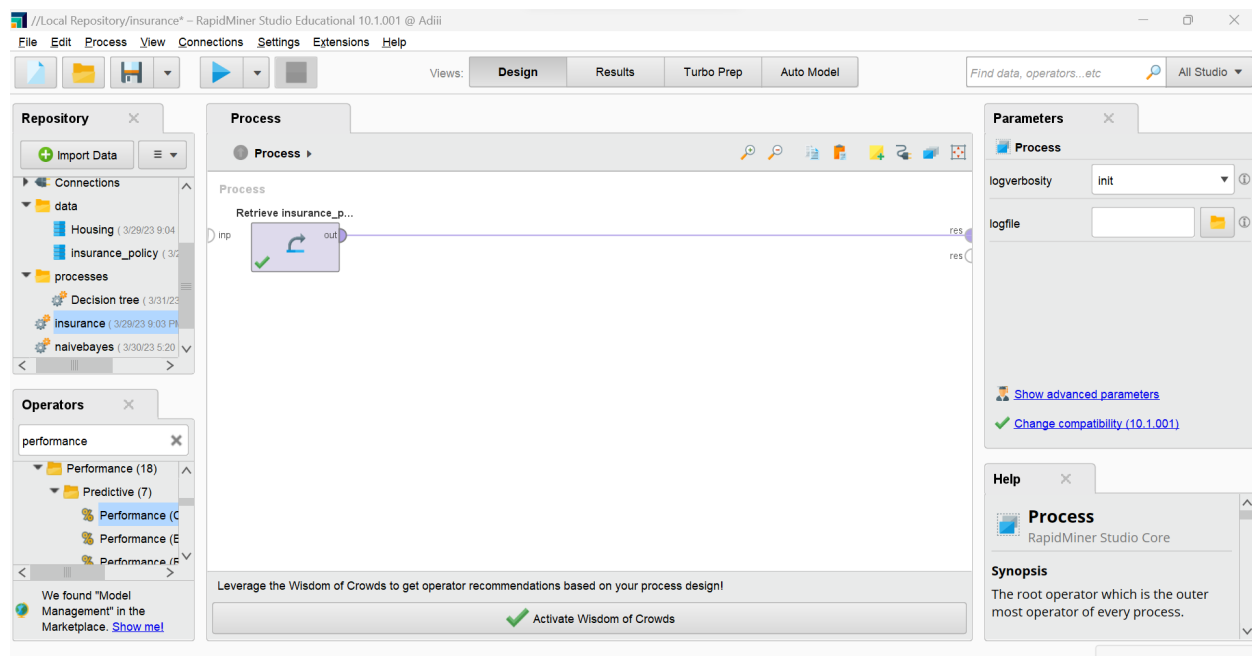
-----  
Number of customer in 4th age group= 6210  
They are - [52 66 42 ... 70 75 72]

-----  
Number of customer in 5th age group= 4911  
They are - [59 40 55 ... 75 61 65]

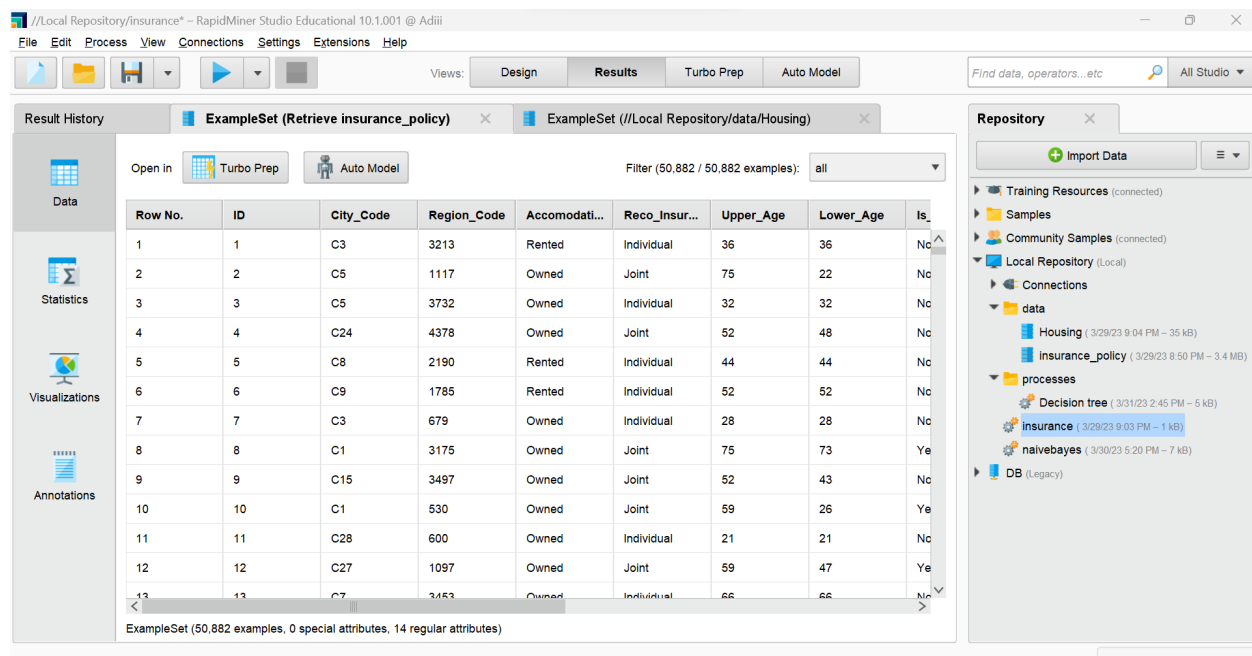
-----

# K-Means(RapidMiner)

## 1. Import the dataset insurance.csv.

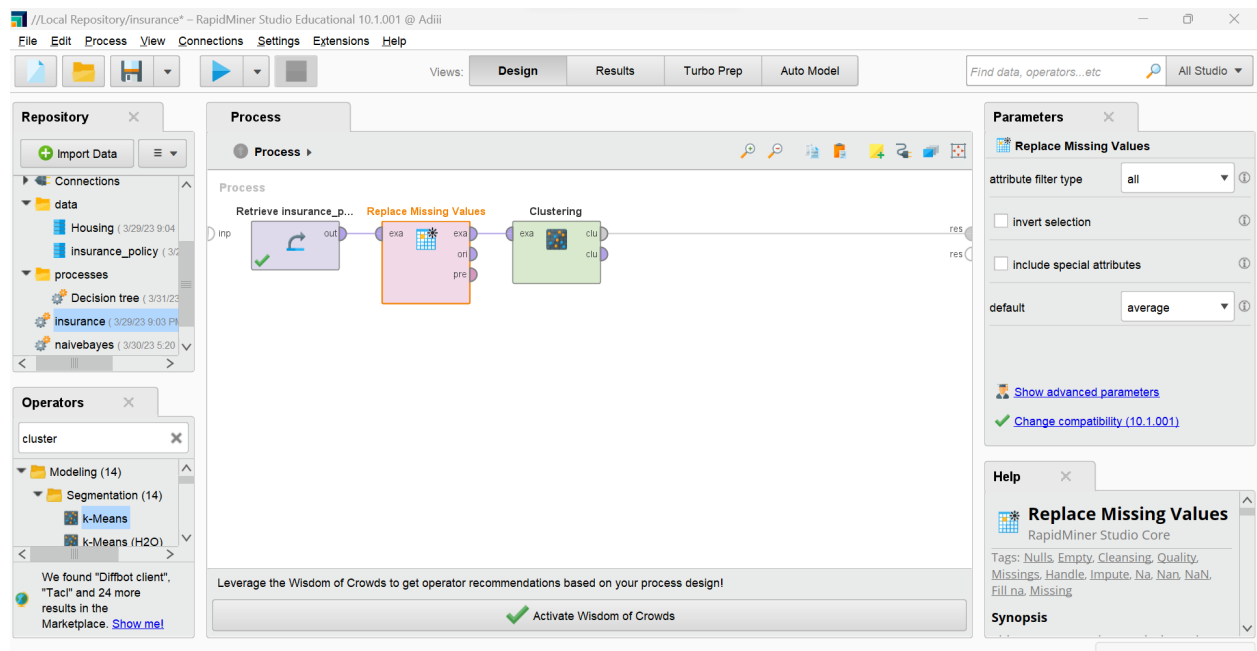


## 2. Run to see the columns in the dataset.

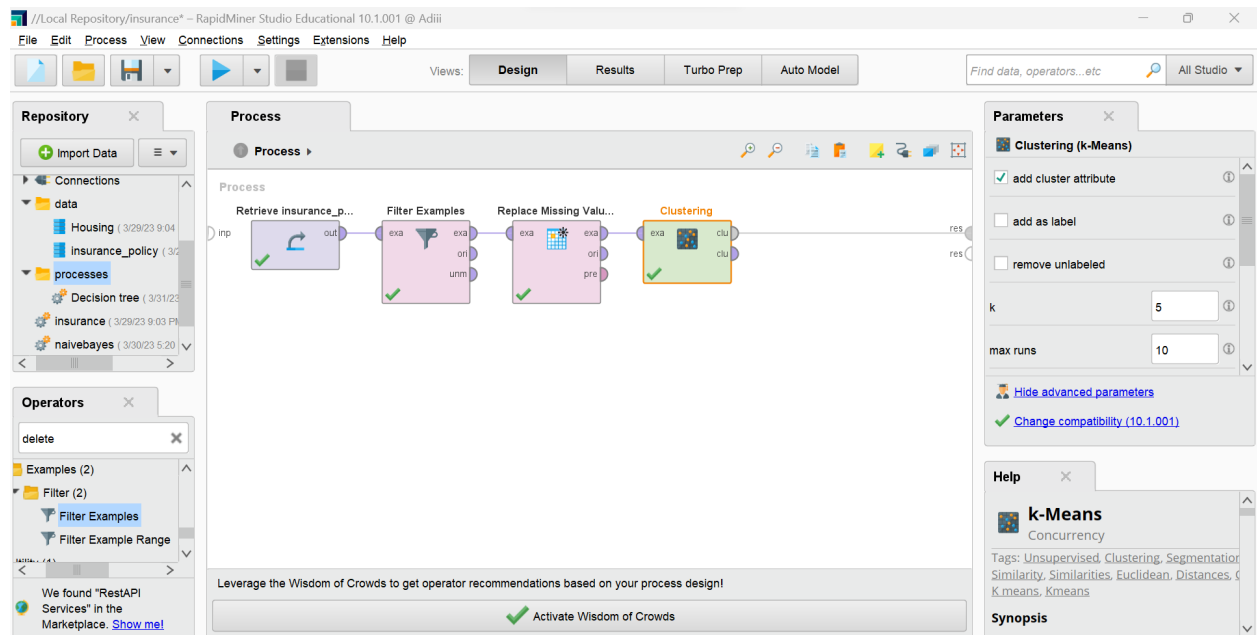


## 3. Replace the missing values and also add the clustering function.

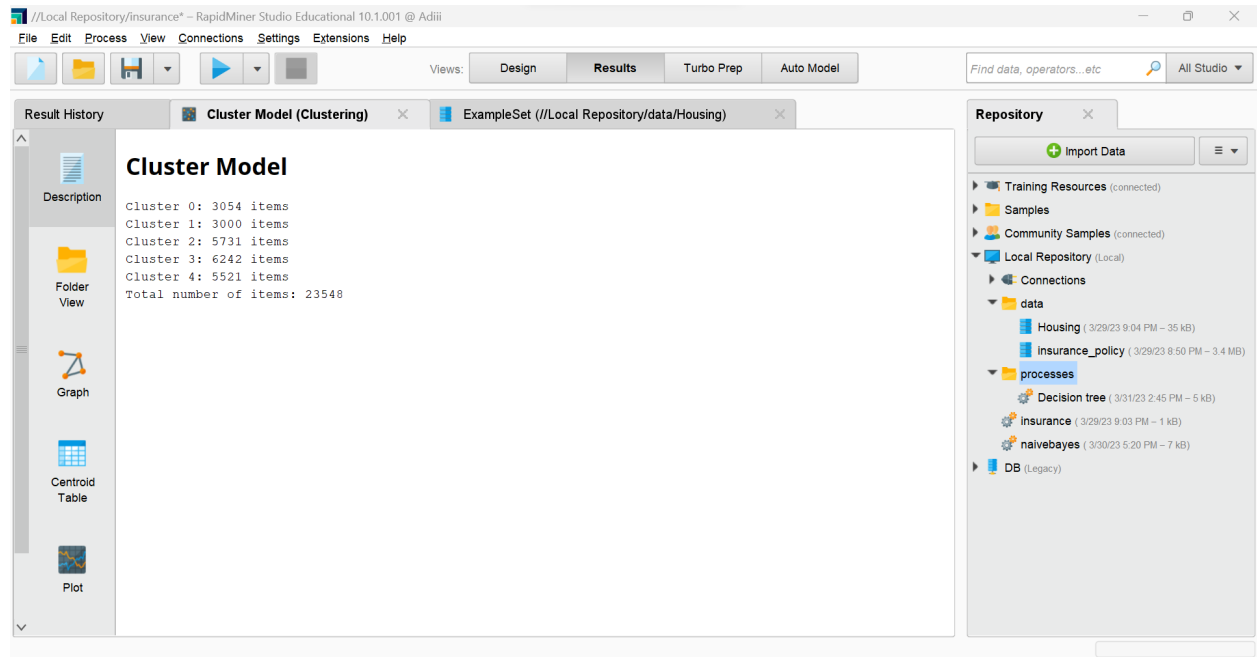




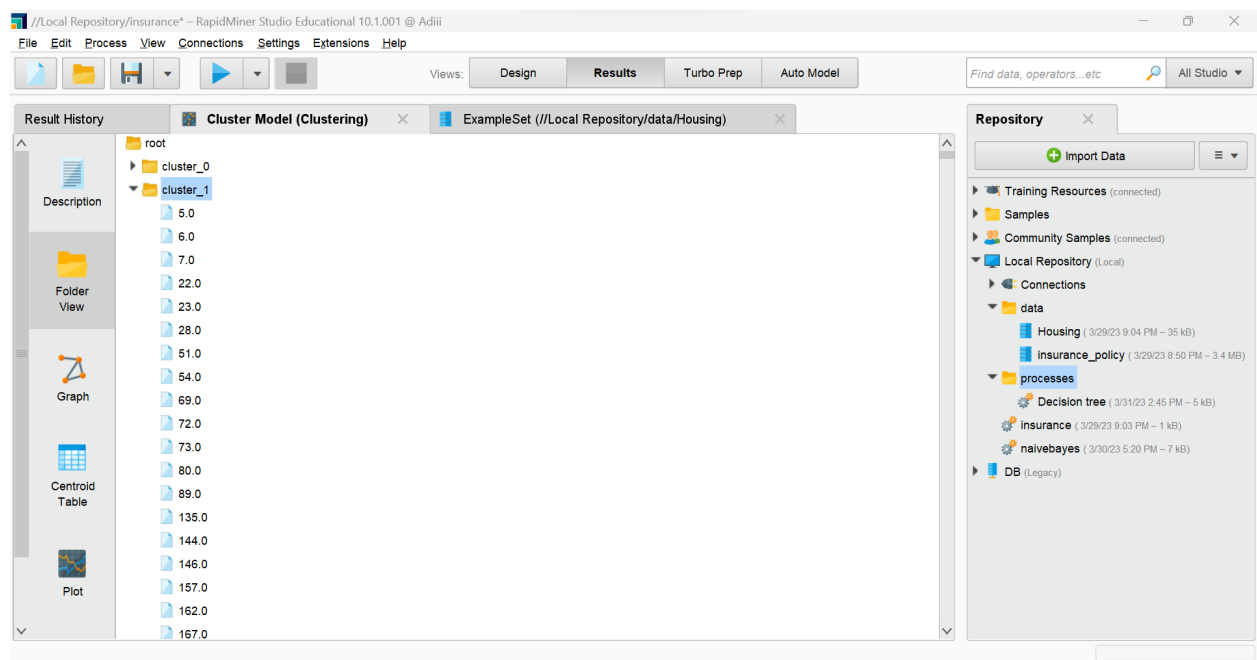
4. Add the required filters also as the K-Means algorithm cannot take in non-numerical values.



5. Run the model to see the clusters formed.



- You can also check out the folders of clusters. Expand the folder to see the individual values present in the cluster.



7. By clicking on the individual data entry you can also see the attribute and value of the particular point in the cluster.

The screenshot shows the RapidMiner Studio interface. On the left, the 'Result History' pane displays a 'Cluster Model (Clustering)' with a tree structure showing 'root', 'cluster\_0', and 'cluster\_1'. A list of data points is shown under 'cluster\_1', with values ranging from 5.0 to 167.0. A dialog box titled 'Example 5' is open, displaying a table of attributes and values for a specific data point.

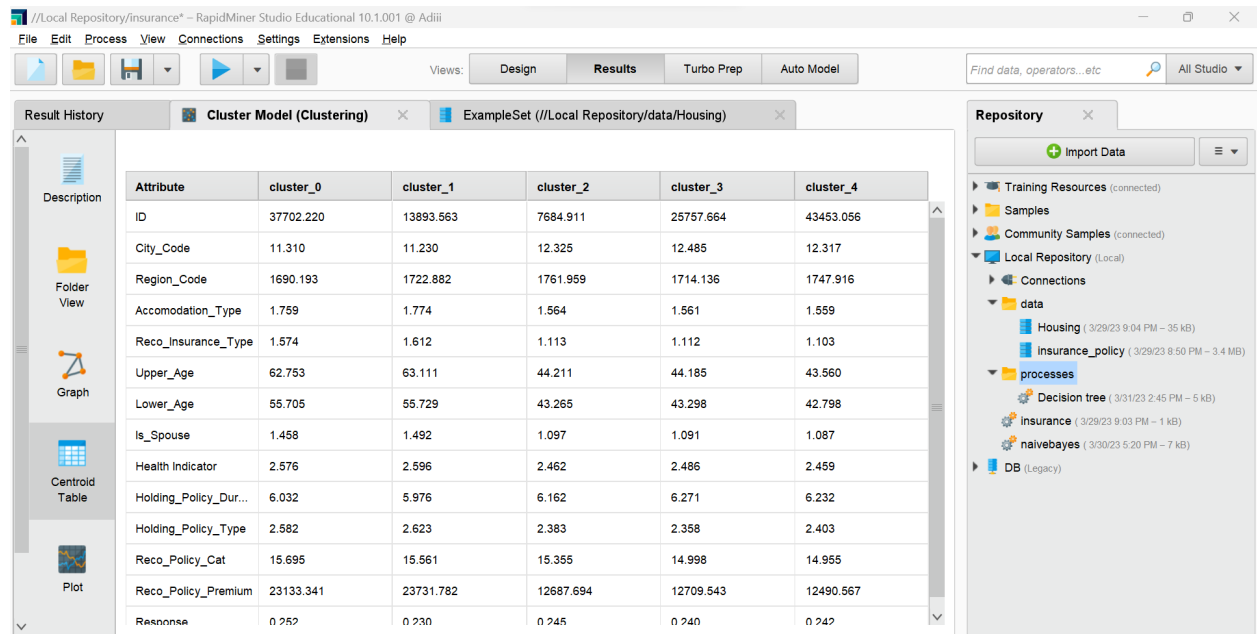
Attribute	Value
ID	8
City_Code	C1
Region_Code	3175
Accommodation_Type	Owned
Reco_Insurance_Type	Joint
Upper_Age	75
Lower_Age	73
Is_Spouse	Yes
Health_Indicator	X4
Holding_Policy_Duration	9
Holding_Policy_Type	4

The right pane shows the 'Repository' with various data sources and processes.

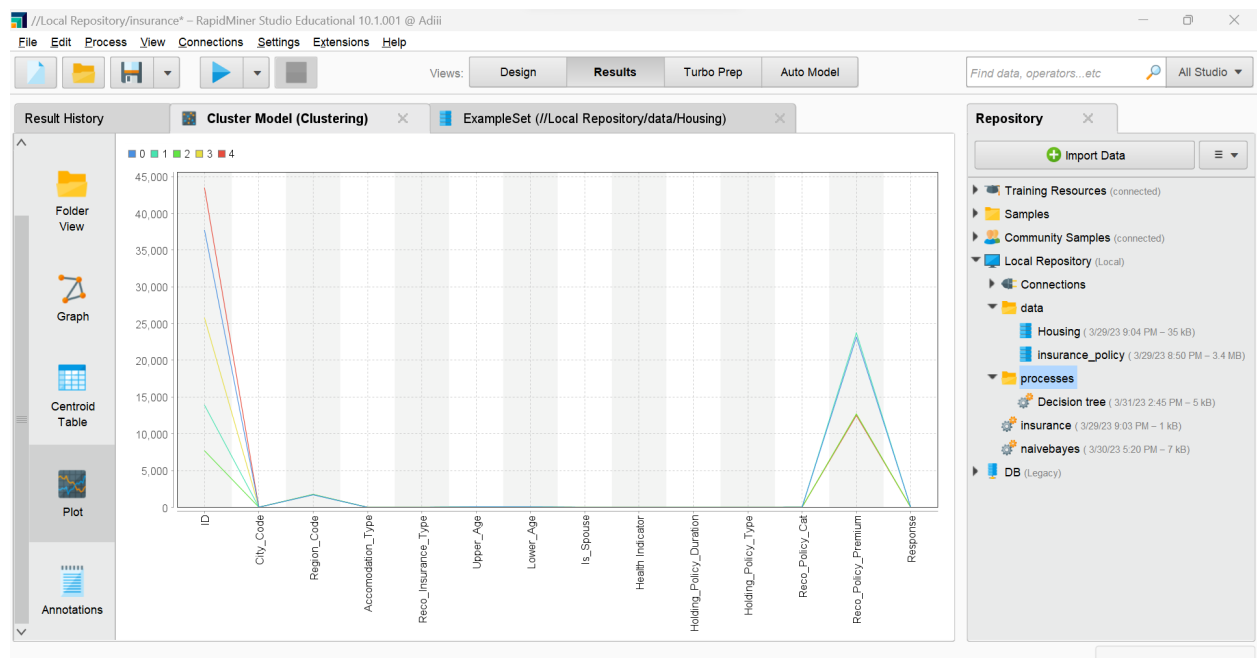
8. Click on the graph in the left hand side menu to look at the clusters in the graph mode.

The screenshot shows the RapidMiner Studio interface with the 'Cluster Model (Clustering)' selected. The 'Graph' view is active, displaying a hierarchical tree structure. The root node is labeled 'root set', and it branches into five child nodes labeled 0, 1, 2, 3, and 4. The left pane shows the 'Result History' with the 'Cluster Model (Clustering)' and 'ExampleSet (/Local Repository/data/Housing)'. The right pane shows the 'Repository' with various data sources and processes.

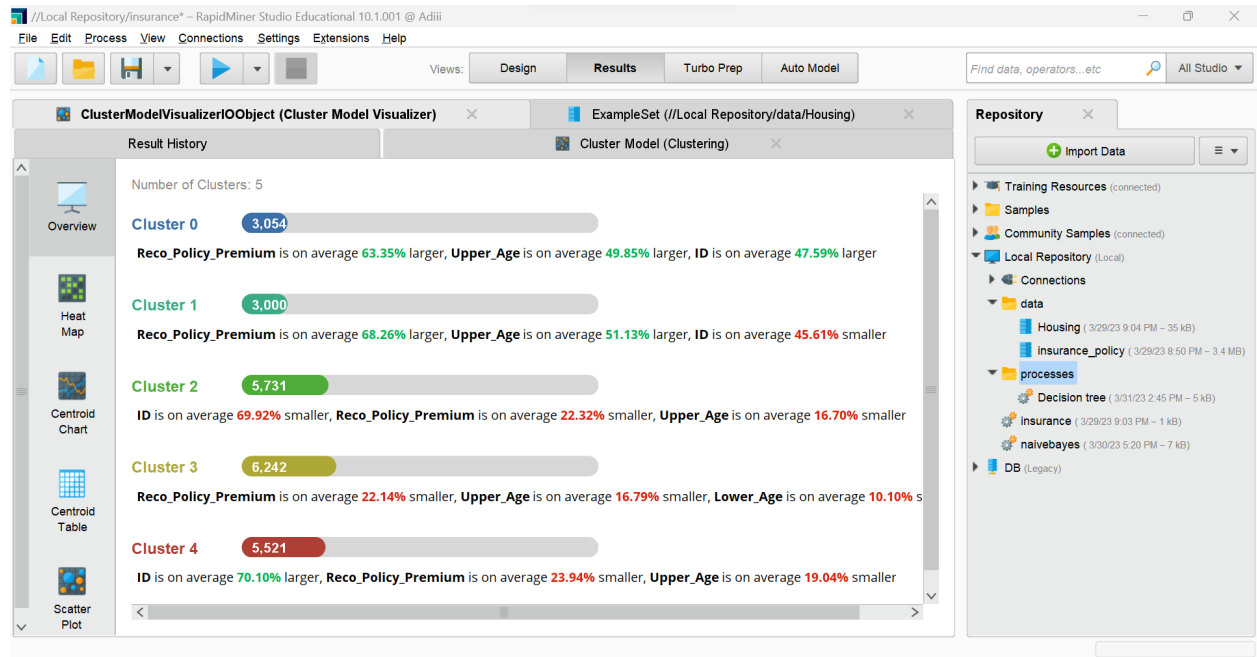
9. The centroid table format could also be viewed in the RapidMiner.



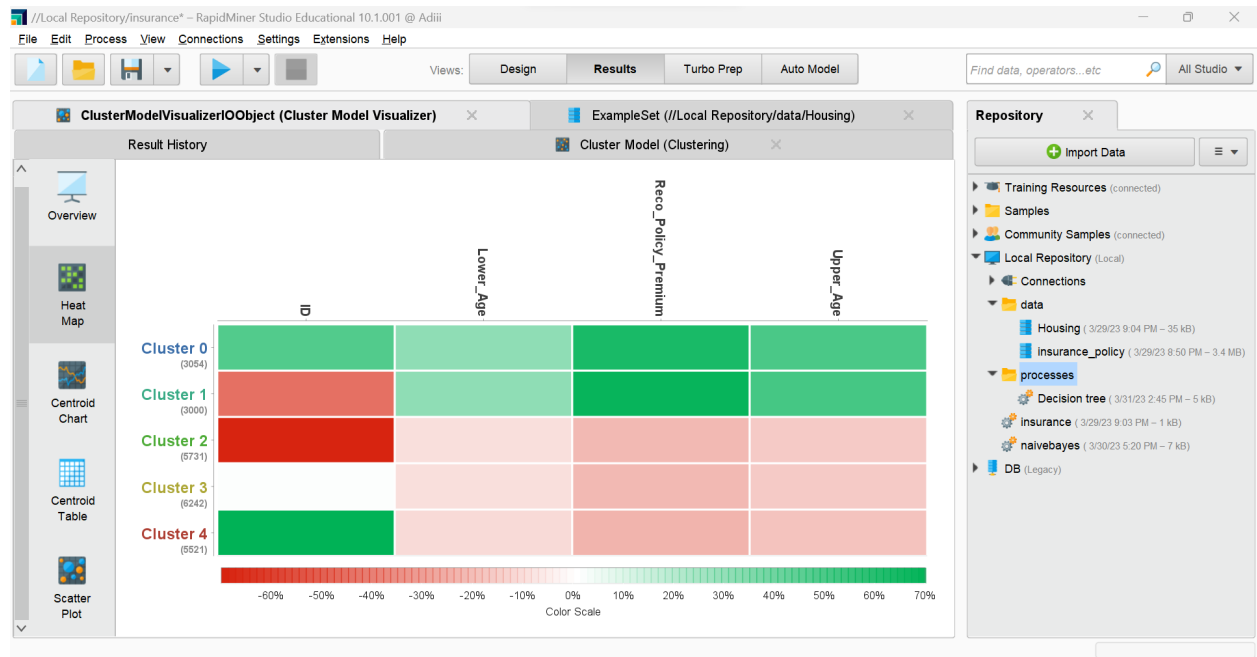
10. Similarly, a plot view is also depicted of the clusters formed.



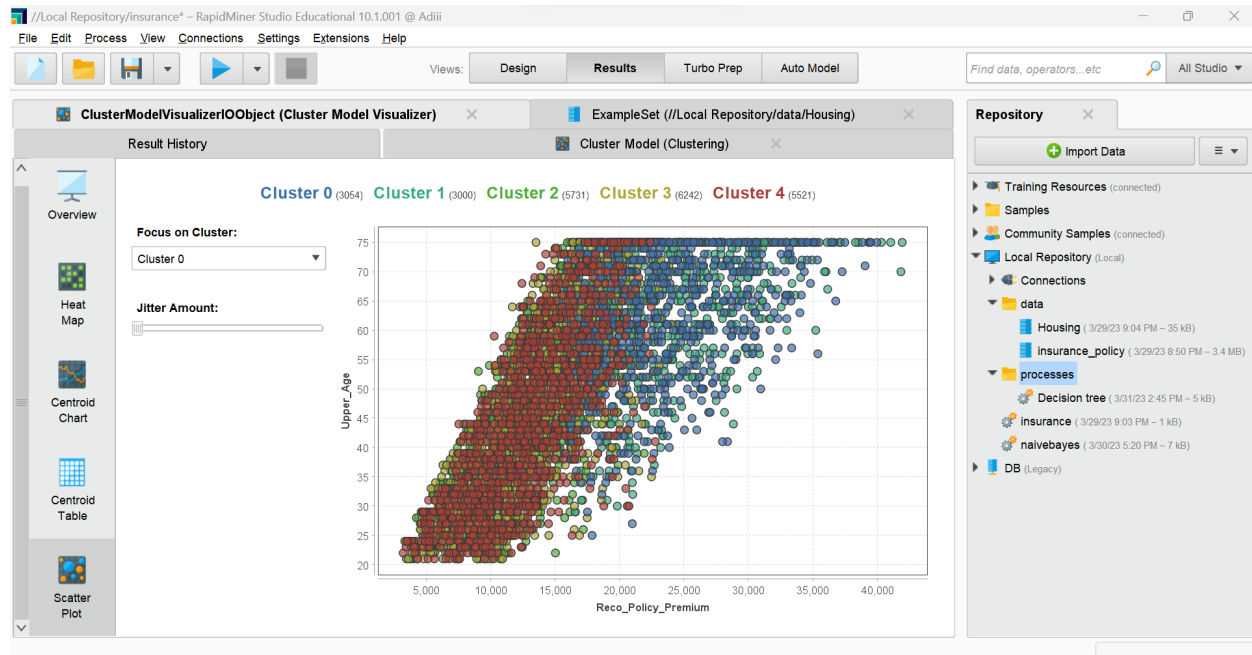
11. Visualization :



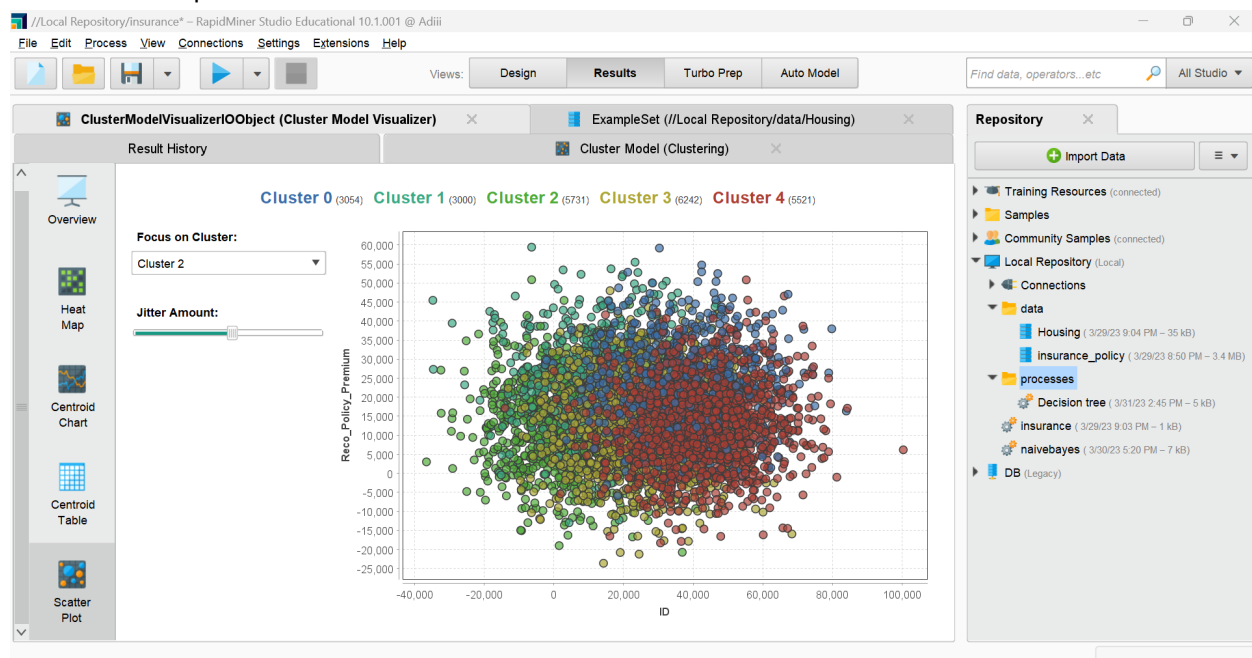
## 12. Heat Map visualization.



## 13. Scatter plot of cluster 0.



#### 14. Scatter plot of cluster 1.



## Conclusion:

We successfully implemented any one of the clustering algorithms(K-Means Clustering Algorithm) using Python and Rapidminer respectively.