In [5]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv("E:\\varnit\\data.csv")
target = pd.read_csv("E:\\varnit\\target_1.csv")

data_encoded = pd.get_dummies(data)

label_encoder = LabelEncoder()
target_encoded = label_encoder.fit_transform(target)


X = data_encoded
y = target_encoded

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

from sklearn.linear_model import Perceptron
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

#CLASSIFIERS
clf1=Perceptron(alpha=0,l1_ratio=0.15,max_iter=100)
clf2=SVC(C=1.0,kernel="rbf")
clf3=DecisionTreeClassifier(criterion="gini",splitter="best", max_depth=5)
clf4=KNeighborsClassifier(n_neighbors=5,metric="minkowski")
clf5=GaussianNB(priors=None)

clf=[clf1,clf2,clf3,clf4,clf5]
clf_name=["perceptron","svc","decisiontree","kneighbors","gaussionNB"]


from sklearn.metrics import accuracy_score
accuracy={}

for model,model_name in zip(clf,clf_name):
    model.fit(X_train,y_train)
    prediction=model.predict(X_test)
    accuracy[model_name]=accuracy_score(y_test,prediction)
print("CLASSIFICATION ACCURACY\n")


for i,j in accuracy.items():
    print(i,":-",j,)
```

```
CLASSIFICATION ACCURACY

perceptron :- 0.6112852664576802
svc :- 0.6112852664576802
decisiontree :- 0.6112852664576802
kneighbors :- 0.6112852664576802
gaussionNB :- 0.3887147335423197
```

In [7]:

```
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.mixture import GaussianMixture
```

```python
from sklearn.cluster import Birch

clustering_algorithms = {
    'K-Means': KMeans(n_clusters=5),
    'Agglomerative': AgglomerativeClustering(n_clusters=5),
    'DBSCAN': DBSCAN(eps=0.5, min_samples=5),
    'GMM': GaussianMixture(n_components=5),
    'K-Means++': KMeans(n_clusters=5, init='k-means++'),
}

data = data_encoded

for algorithm_name, algorithm in clustering_algorithms.items():
    labels = algorithm.fit_predict(data)
    print(f"Algorithm: {algorithm_name}")
    print("Cluster Labels:")
    print(labels)
    print("------")
```

```
Algorithm: K-Means
Cluster Labels:
[0 0 0 ... 0 0 0]
------
Algorithm: Agglomerative
Cluster Labels:
[0 0 0 ... 2 0 0]
------
Algorithm: DBSCAN
Cluster Labels:
[-1 -1 -1 ... -1 -1 -1]
------
Algorithm: GMM
Cluster Labels:
[0 0 0 ... 0 0 0]
------
Algorithm: K-Means++
Cluster Labels:
[0 0 0 ... 0 0 0]
------
```

In [ ]: