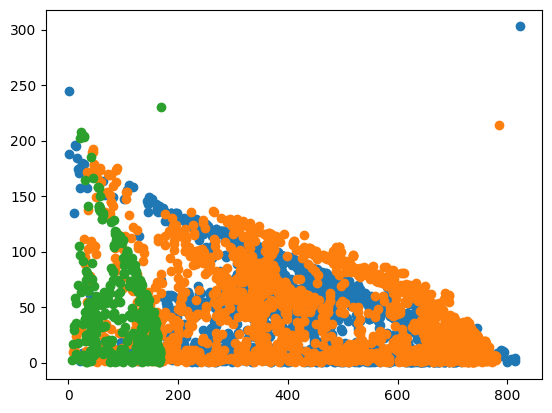
Procedure

* Load the dataset
* Preprocessing
  1. Implement **PCA f**or the dataset (n\_componenets=2)
     + Result: two columns (dfnew1, dfnew2) ->dfnew
  2. Implement **K-means++**
     + Create an elbow curve to see the best splitting value
     + We got 3 as best splitting value so split them
       - Result: clusters (contains three clusters)
* Main function (RdpOD)
  1. Calculate the **Centroids** of the three clusters.
  2. Using distance\_from\_centroid function, calculate the **distances** of every data points in three clusters.
  3. Calculate **mean** of distances of every cluster.
  4. Calculate **standard deviation** for the clusters.
  5. Using mean and standard deviation calculate **threshold** value for every cluster.
  6. Calculate **average cutoff distance.**
  7. Calculate **local density** and **max distance** of each cluster.
  8. Calculate **closest distance** to highest density points for each point of the clusters.
  9. Plot local density and distance values of each cluster.



* Split data points, local\_densities and closest\_distance values into cluster 0,1,2

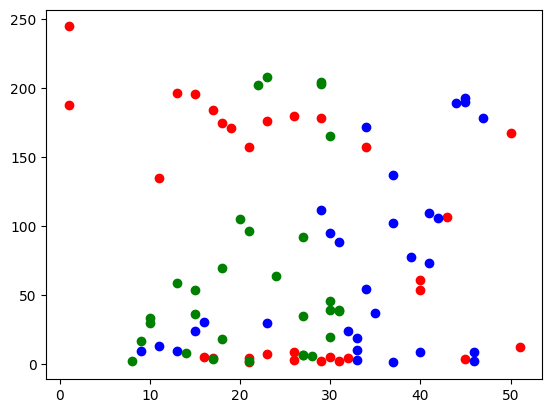
1. Visual illustration:

Cluster0[ [data points] [local density values] [closest distance values] ]

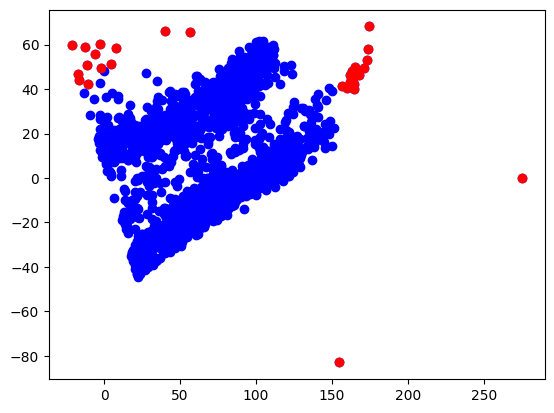
* **Sort** the local density values and also sort data points and closest distance values with local density as reference.
* Take **input** from the user for **probable outliers** for each cluster (ex:30) from each cluster.
* Using sorted local density values get the values using slicing method in python

Result: outliers\_in\_cluster0 [ [data points] [local density] [closest distance] ]

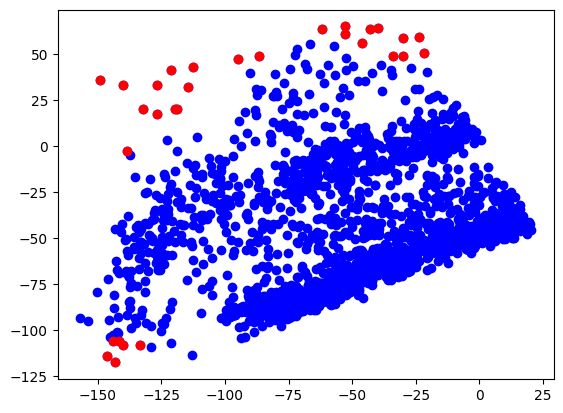
* Plot the graph for probable outliers



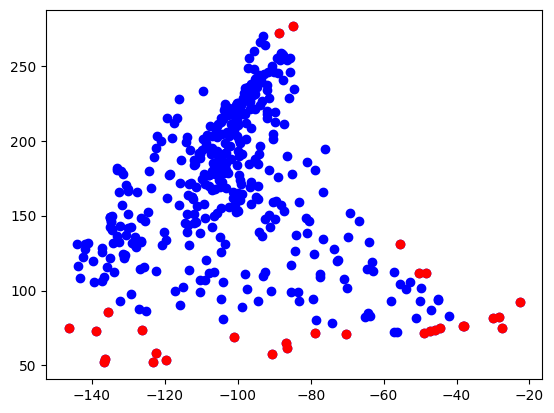
* Plot **probable outliers** for each cluster.
  + Cluster 1:



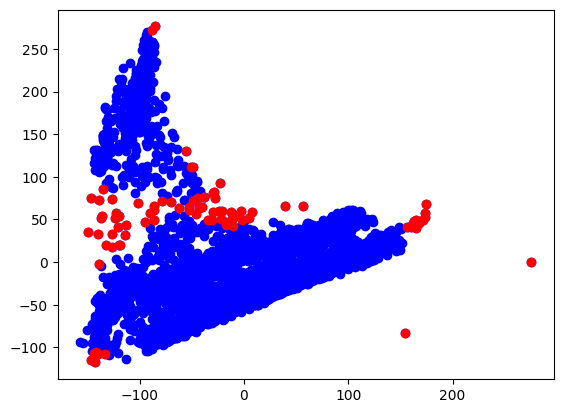
* + Cluster 2:



* + Cluster 3:



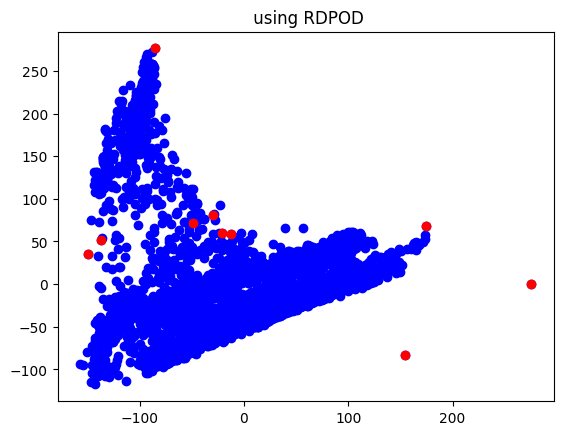
* **Merge** each cluster probable outliers in to one list
* Again sort the outliers based on local densities remember also sort data points and closest distance values.
* **Merged probable outliers** i.e 90 data points for entire dataset.



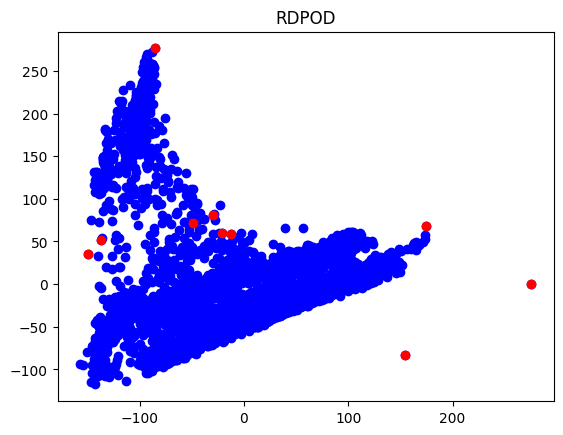
* Perform **ODF** (outlier detection factor)
  + Result: 90 values calculated using 90 local densities and closest distance values.
* **Merging** outlier factor into the cluster list of lists

Result: Cluster0[ [data points] [local densities] [closest distance] [outlier factor] ]

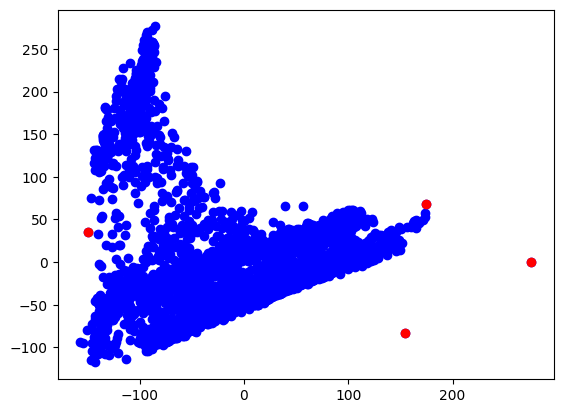
* **Sorting** the array with reference to outlier factor values
* Take **input** from the user for **global outliers**. (ex.10)
* Take first N data points (i.e global outliers) using **slicing** method.
* Plot those global outliers data points.



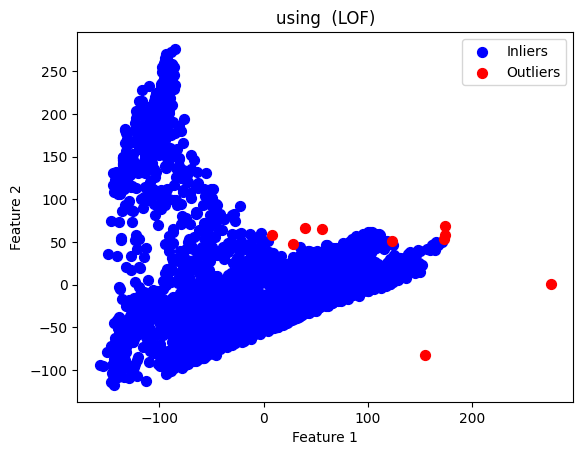
* Implement **EXTERME LEARNING MACINE** (ELM) method to the final global outliers.
  + That takes global outliers data points and returns individual score.
  + Using threshold value which is calculated using np.percentile method get the final outliers.
* Result analysis:
  + Using RdpOD



* + Using ELM:



* + Using LOF (local outlier factor):



* + Using **Relative density algorithm**

