DBSCAN Clustering of Credit Card Data

By Dipansh Girdhar

Overview

While dealing with spatial clusters of different density, size, and shape, it could be challenging to detect the cluster of points. The task can be even more complicated if the data contains noise and outliers. To deal with large spatial databases, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was proposed. 3 main reasons for using the algorithm are

1. It requires minimum domain knowledge.
2. It can discover clusters of arbitrary shapes.
3. Efficient for large databases, i.e. sample size more than a few thousand.

The main concept of DBSCAN algorithm is to locate regions of high density that are separated from one another by regions of low density. So, how do we measure density of a region? Below are the 2 steps —

1. Density at a point P: Number of points within a circle of Radius Eps from point P.
2. Dense Region: For each point in the cluster, the circle with radius ϵ contains at least a minimum number of points (MinPts).

Steps in DBSCAN Algorithm

1. The algorithm starts with an arbitrary point which has not been visited and its neighborhood information is retrieved from the ϵ parameter.
2. If this point contains MinPts within ϵ neighborhood, cluster formation starts. Otherwise the point is labeled as noise. This point can be later found within the ϵ neighborhood of a different point and, thus can be made a part of the cluster. Concept of density reachable and density connected points are important here.
3. If a point is found to be a core point then the points within the ϵ neighborhood is also part of the cluster. So all the points found within ϵ neighborhood are added, along with their own ϵ neighborhood, if they are also core points.
4. The above process continues until the density-connected cluster is completely found.
5. The process restarts with a new point which can be a part of a new cluster or labeled as noise.

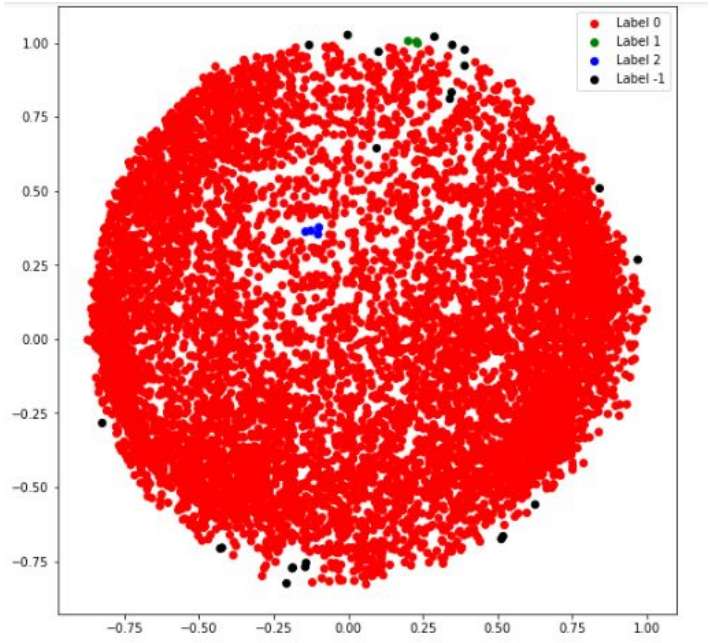
Analysis of Credit Card Data

In the given dataset, we are given a dataset with 18 columns. Here, the first column CUSTOMER_ID is unique to every customer. So, for the task of clustering, we will drop this column.
The clustering of data is done using the remaining 17 columns.
The correlation matrix for these 16 columns is shown below.

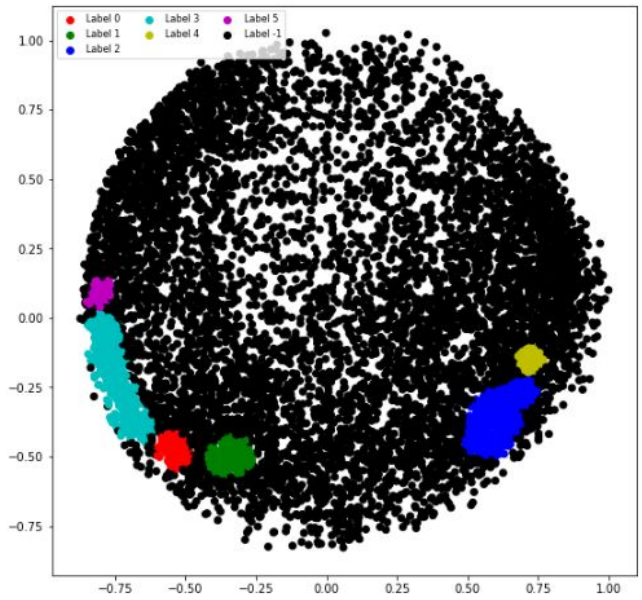
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	0.297441	0.114129	0.131865	0.0280014	0.496657	-0.113998	0.0708695	-0.112117	0.440013	0.416517	0.0771743	0.419697	0.299457	0.518118	-0.27528	0.0014289
1	0.297441	1	0.0488131	0.036884	0.0414811	0.0262937	0.125841	0.112356	0.0890189	0.13766	0.0893616	0.0989545	-0.0183666	-0.018212	0.127591	-0.202649	0.142185
2	0.114129	0.0488131	1	0.873444	0.674764	-0.0218075	0.514695	0.583926	0.379272	-0.100684	-0.0392269	0.732816	0.324502	0.463636	0.116448	0.231557	-0.0200071
3	0.131865	0.036884	0.873444	1	0.230114	-0.00113084	0.294589	0.669232	0.0737754	-0.050991	-0.00855909	0.53825	0.295071	0.407018	0.0726437	0.128888	-0.0278723
4	0.0280014	0.0414811	0.674764	0.230114	1	-0.0421107	0.582324	0.152783	0.646308	-0.123965	-0.0655227	0.649045	0.201283	0.309566	0.122564	0.267261	0.00215392
5	0.496657	0.0262937	-0.0218075	-0.00113084	-0.0421107	1	-0.205663	-0.0567097	-0.167939	0.651103	0.68141	-0.0522037	0.269436	0.43614	0.242361	-0.0740194	-0.14712
6	-0.113998	0.125841	0.514695	0.294589	0.582324	-0.205663	1	0.429281	0.879264	-0.284276	-0.203967	0.693622	0.110953	0.108816	0.00536169	0.299019	-0.00143314
7	0.0708695	0.112356	0.583926	0.669232	0.152783	-0.0567097	0.429281	1	0.0503238	-0.0726242	-0.0369174	0.547735	0.250981	0.240144	0.0161025	0.116398	-0.000728844
8	-0.112117	0.0890189	0.379272	0.0737754	0.646308	-0.167939	0.879264	0.0503238	1	-0.240253	-0.171338	0.616117	0.0498794	0.0743684	0.0223793	0.266386	0.0125452
9	0.440013	0.13766	-0.100684	-0.050991	-0.123965	0.651103	-0.284276	-0.0726242	-0.240253	1	0.860955	-0.109127	0.086915	0.232756	0.203051	-0.162218	-0.145521
10	0.416517	0.0893616	-0.0392269	-0.00855909	-0.0655227	0.68141	-0.203967	-0.0369174	-0.171338	0.860955	1	-0.0438854	0.124256	0.278555	0.233701	-0.105083	-0.110683
11	0.0771743	0.0989545	0.732816	0.53825	0.649045	-0.0522037	0.693622	0.547735	0.616117	-0.109127	-0.0438854	1	0.233113	0.313273	0.110708	0.222365	0.0159828
12	0.419697	-0.0183666	0.324502	0.295071	0.201283	0.269436	0.110953	0.250981	0.0498794	0.086915	0.124256	0.233113	1	0.364949	0.142176	0.120325	0.0292876
13	0.299457	-0.018212	0.463636	0.407018	0.309566	0.43614	0.108816	0.240144	0.0743684	0.232756	0.278555	0.313273	0.364949	1	0.209842	0.159858	0.00489795
14	0.518118	0.127591	0.116448	0.0726437	0.122564	0.242361	0.00536169	0.0161025	0.0223793	0.203051	0.233701	0.110708	0.142176	0.209842	1	-0.121653	-0.00486007
15	-0.27528	-0.202649	0.231557	0.128888	0.267261	-0.0740194	0.299019	0.116398	0.266386	-0.162218	-0.105083	0.222365	0.120325	0.159858	-0.121653	1	-0.0808209
16	0.0014289	0.142185	-0.0200071	-0.0278723	0.00215392	-0.14712	-0.00143314	-0.000728844	0.0125452	-0.145521	-0.110683	0.0159828	0.0292876	0.00489795	-0.00486007	-0.0808209	1

The next step is to reduce the number of features using principal component analysis. We use dimensionality reduction to get two features instead of 17 to make the clustering easier.

To cluster, we use the DBSCAN model present in the sklearn library.
When we consider the number of samples to be at least 3, we get 4 clusters in which the data is divided. The scatter plot of the same can be seen to visualize it.



From the scatter plot, we can see that most of the sample data belong to one particular cluster. When we increase the minimum sample size to 50, we get data clustered in 7 types.



We see better trends in the data this time. Although, there's still one class that occupies the major portion of the data, but we have a better picture and understanding of the minority classes.

The number of members in each cluster after this operation is observed as:

```
{-1: 7218, 0: 134, 1: 217, 2: 564, 3: 646, 4: 83, 5: 88}
```