Rohan Nyati

500075940

R177219148

Batch-5(AI&ML)

Assignment -2

Q: Use the above provided dataset to perform outlier detection.

State the algorithm used and explain the reason for use of the same.

Ans: Here I used DBSCAN

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Importing the dataset
dataset = pd.read_csv("d.csv")

#removing infinite values or nan values
dataset = dataset.reset_index()
dataset.replace([np.inf, -
np.inf], np.nan, inplace=True)

from sklearn.cluster import DBSCAN


dataset.isna().sum()
dbscan=DBSCAN(eps=5,min_samples=1000)


#features


X = dataset.iloc[:,[3,]].values

# Fitting the model
```

```
model=dbscan.fit(X)

labels=model.labels_
print(labels)
no_clusters = len(np.unique(labels))
no_noise = np.sum(np.array(labels) == -1, axis=0)
 #                                blue(8-
10)               (black - noise)
colors = list(map(lambda x: '#3b4cc0' if x == 1 else(
"#02030a"  if x == -1  else '#b40426'), labels))

#identifying the points which makes up our core point
s
sample_cores=np.zeros_like(labels,dtype=bool)

sample_cores[dbscan.core_sample_indices_]=True

#Calculating the number of clusters
print("number of clusters = ",no_clusters)
print("noise points = ",no_noise)
#plt.scatter(X[:,0], X[:,1], c=colors, marker="o", pi
cker=True)
#plt.legend(["red 8-
10","blue firther"],loc ="lower right")
#plt.title(label=f"clusters = {no_clusters}")
#plt.xlabel('X>>>')
#plt.ylabel('Y>>>',labelpad=5)
#plt.show()
from scipy import stats
print("z score = ", stats.zscore(X,axis=0))
#blue 8-10
#black -1 noise
#red 4-8
```

```
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
number of clusters =  1
noise points =  475
```

```
z score =  [[ 0.18110968]
 [ 0.18110968]
 [ 0.18110968]
 [ 0.18110968]
 [ 0.18110968]
 [ 0.18110968]
 [ 0.18110968]
 [ 0.070535  ]
 [ 0.29168435]
 [ 0.18110968]
 [ 0.12582234]
 [ 0.34697169]
 [ 0.12582234]
```

DBSCAN algorithm is a density based algorithm. It looks at the *density* of data points in a neighbourhood to decide whether they belong to the same cluster or not. If a point is too far from all other points then it is considered an outlier and is assigned a label of -1, as it did in above output.

As it doesn't require us to give prior number of clusters to be formed, we have an advantage of using the potential of the system on its own.

And it can work on arbitrary clusters.