Weekend
SPECIAL

COMPLETE ANOMALY DETECTION

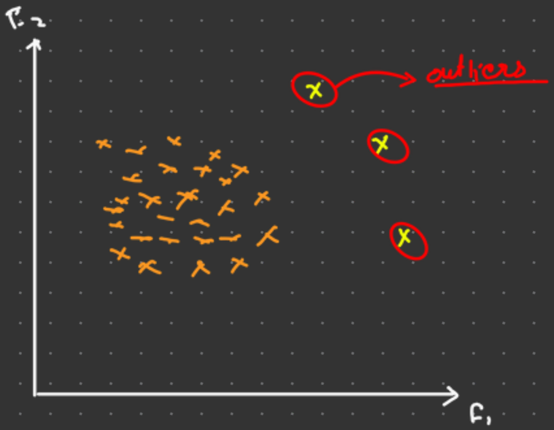
- Isolation Forest
- DBSCAN Anomaly detection
- Local Outlier Factor

Anomaly Detection

Anomaly Detection is use to detect outliers that are very Important for a problem statement.

↳ In any business use case, if we want to find a specific Outlier, Which is very important for the business problem Statement then we use Anomaly detection for that.

↳ If any third party person tries to login to our account we got the massage of authontication approval because Anomaly detection algorithm is implemented in it.



Types of Anomaly detection

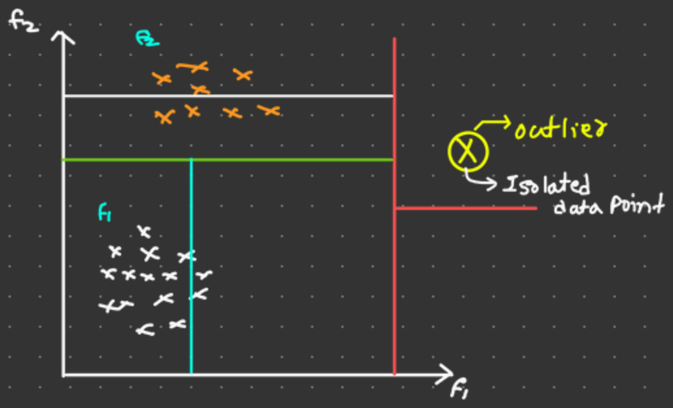
① Isolation forest [Decision tree] :-

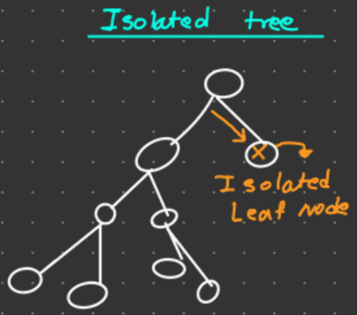
we will create internally isolated trees. inside the Isolation Forest. Inside the isolation tree, we will take any feature from the dataset and keep splitting them and the splitting will continue until a simple single node of our entrie data is found. it means we want to get leaf node from every node that are present here. And by doing this we are creating isolated leaf node here.

↳ ex

F1	F2	F3	F4
–	–	–	–
–	–	–	–
–	–	–	–
–	–	–	–
–	–	–	–

/ end goal to find outlier.





How to determind outlier in between normal points

The isolated leaf node get the leaf node very Quickly. A leaf node that has less depth has a higher possibility of becaming a outlier.
For this we calculate Anomaly Score.

Anomaly Score.

To find the posibilities of which leaf node can became outlier, for that we calculate Anomaly Score.

Mathematical Formula :-

Compute anomaly score for a new point.

$$\boxed{S(x,m) = 2^{\frac{-E(h(x))}{C(m)}}}$$

This will help to find out who is outlier OR who is normal data point

Here :

m = Total No. of data point.
x = Isolated data point
$E(h(x))$ = Average search depth for x from the isolated tree.
$C(m)$ = Average depth of all the data point.

let, threshold ≥ 0.5

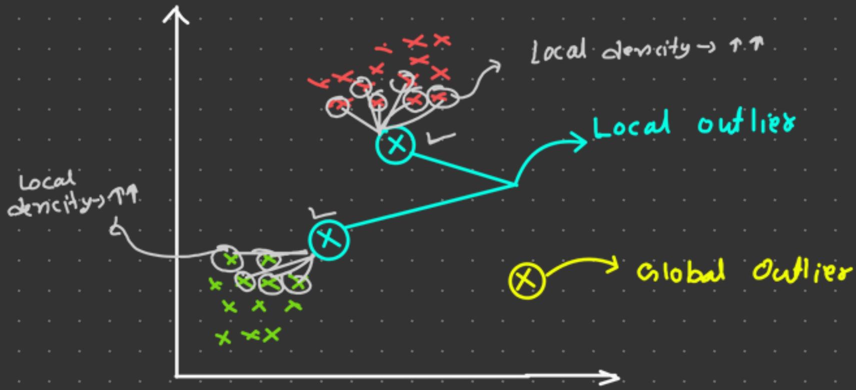
- $E(h(x)) << C(m) \Rightarrow S(x,m) \approxeq 1 \Rightarrow$ Anomaly score $\Rightarrow$ Outlier

- $E(h(x)) >> C(m) \Rightarrow S(x,m) \approxeq 0.5 \Rightarrow$ Normal data point.

② DBSCAN - Anomaly Detection $\longrightarrow$ Theory same.

③ Local Outlier factor Anomaly detection (Lof)

Lof use to detect outlier from the dataset.



➤ Global Outlier :-

It is completly different and far from the other points.
Global outliers can be calculate by DBSCAN and Isolation forest anomaly detection.

➤ Local Outlier :-

LOF is a method used in anomaly detection to identify data points that are outliers in their local neighbourhoods.

- **Local Neighborhood** → LOF measures how isolated a data point within its local neighbourhood, compared to its neighbours. It calculates the density of data points in the vicinity of each point.

- **Calculation** → For each data point, LOF compare the density of its local neighbourhood to the density of its neighbour. If a point has a significantly lower density than its neighbours, it is considered an outlier.

- **Scoring** → LOF assign a score to each data point, where a higher LOF score indicates a higher likelihood of being an outlier.

f1dc51vgd

October 20, 2023

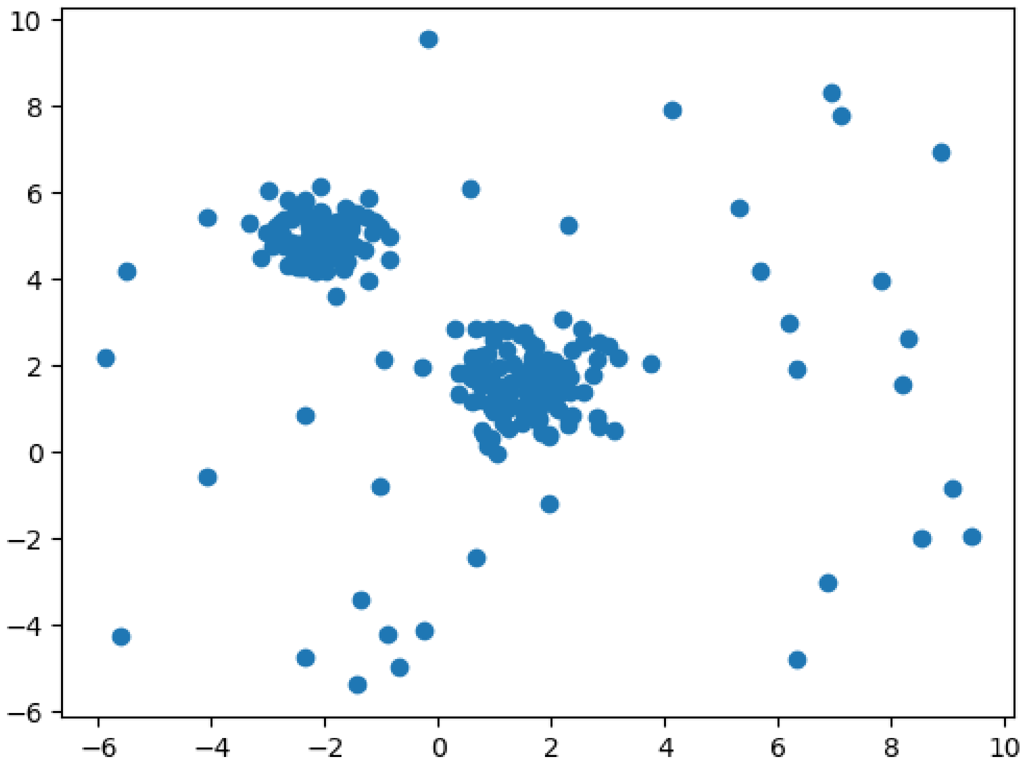
1 Isolation Forest :(Anomaly Detection)

[1]:
```
# importing the dataset
import pandas as pd
df = pd.read_csv('healthcare.csv')
df.head(3)
```

[1]:
```
          0         1
0  1.616671  1.944522
1  1.256461  1.609444
2 -2.343919  4.392961
```

[2]:
```
# now plot the data points into the scater plot
import matplotlib.pyplot as plt
plt.scatter(df.iloc[:,0], df.iloc[:,1])
```

[2]:
```
<matplotlib.collections.PathCollection at 0x7f99a3c91c60>
```



[3]:
```
# importing the Isolation forest from sklearn
import warnings
warnings.filterwarnings('ignore')
from sklearn.ensemble import IsolationForest
```

[4]:
```
clf = IsolationForest(contamination= 'auto')      #contamination == threshold␣
 ↪value (go for 'auto' so that it give best result)
clf.fit(df) ## applying fit on the df

prediction = clf.predict(df)
```

[5]:
```
prediction     # +1 == Normal data points , -1 == Outliers
```

[5]:
```
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1, -1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1,  1,  1,
        1,  1, -1,  1, -1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
        1,  1,  1,  1, -1,  1, -1,  1,  1,  1, -1,  1,  1,  1,  1, -1,  1,
       -1,  1, -1,  1,  1,  1,  1, -1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
       -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1,  1,  1,
        1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,
       -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1, -1,  1,  1, -1,
       -1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1, -1,  1,  1,  1,  1, -1, -1,  1])
```

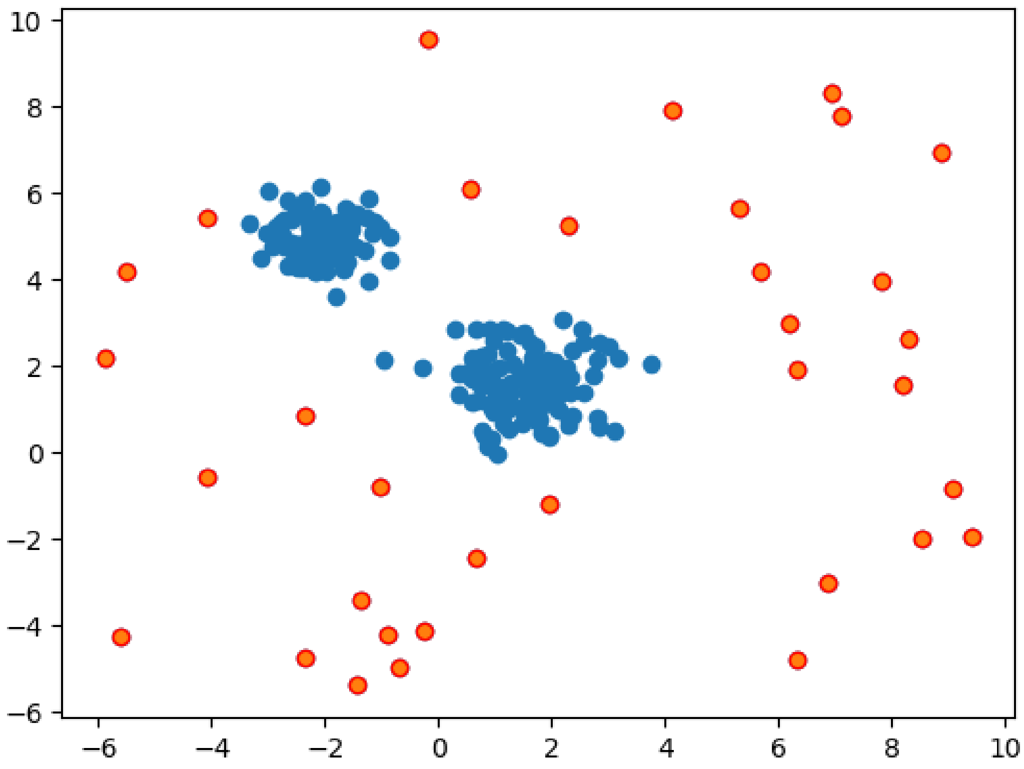
[6]:
```
# geting the outlier values into a index
import numpy as np
index = np.where(prediction < 0 )
index
```

[6]:
```
(array([ 20,  24,  45,  48,  53,  55,  63,  72,  74,  78,  83,  85,  87,
         92,  97, 108, 114, 119, 130, 133, 141, 151, 167, 179, 187, 199,
        212, 217, 220, 221, 227, 242, 247, 248]),)
```

[7]:
```
# converting into array
x = df.values
```

[8]:
```
index = np.where(prediction < 0)
plt.scatter(df.iloc[:,0], df.iloc[:,1])
plt.scatter(x[index,0], x[index,1], edgecolors= 'r')
```

[8]:
```
<matplotlib.collections.PathCollection at 0x7f9997924040>
```



2 Observation:

The data points in red color are actually Outliers that we get by Anomaly detection technique. this red points here represent those persons who having disease in our dataset.

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

ilgztsjnl

October 20, 2023

1 DBSCAN Anomaly Detection

[1]:
```
# importing imp. libraries
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.datasets import make_circles ## non linear dataset
```

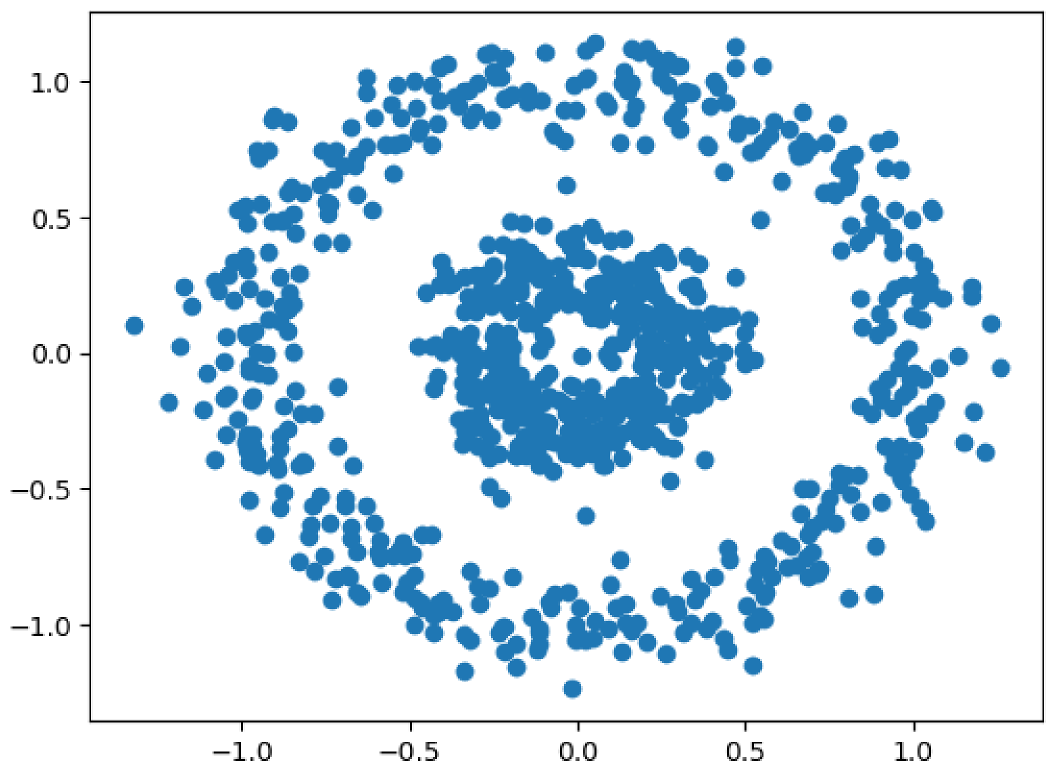
[2]:
```
x, y = make_circles(n_samples= 900, factor= 0.3, noise= 0.1 )  # factor range␣
 ↪between 0 --> 1
```

[3]:
```
x
```

[3]:
```
array([[ 0.26553989,  0.08844725],
       [-0.22191213,  0.39664314],
       [ 0.03629262, -0.31765026],
       …,
       [ 0.70015927, -0.63973118],
       [-0.62642404,  0.75961353],
       [-0.13891447, -0.14012556]])
```

[4]:
```
'''We Know that Anomaly detection is also a Unsupervised machine learning␣
 ↪algorithm and Unsupervised ML
algorithm has no use of y'''

#ploting the data
plt.scatter(x[:,0], x[:,1])
```

[4]:
```
<matplotlib.collections.PathCollection at 0x7fcd294579a0>
```



[5]:
```
dbcan = DBSCAN(eps= 0.10)
```

[6]:
```
dbcan.fit_predict(x)        # -1 == anomaly value
```

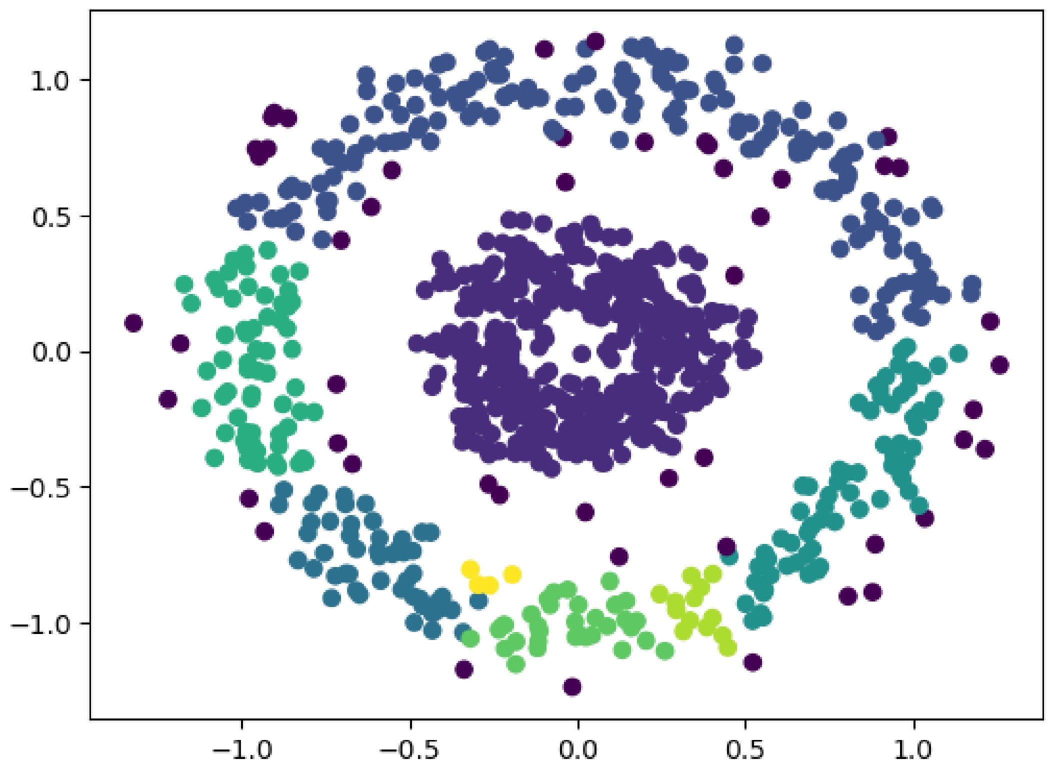
[6]:
```
array([ 0,  0,  0,  0,  0,  1,  0,  0,  1,  1,  2,  0,  0,  0,  0,  0,  1,
        3,  3,  0,  2,  0,  0,  0, -1,  0,  0,  0,  3,  0,  1,  4,  1,  3,
        0,  0,  3,  3,  3,  1,  3, -1,  0,  0,  3,  1,  1,  4,  0,  0,  0,
        3,  0,  1,  0,  0, -1,  0,  1,  0,  0,  1,  3,  5,  0,  0,  0,  0,
        0,  4,  0, -1,  1,  0,  2,  0,  0,  0,  4,  0,  6,  3,  0,  4,  1,
        4,  1,  0,  3,  0,  1,  0,  3,  0,  0,  1,  0,  1,  0,  1,  0,  1,
        0,  1,  1,  0,  0,  0,  0,  2,  0, -1,  1,  0,  4,  0,  3,  3,  3,
        0,  1,  3,  5,  1,  4, -1,  0,  0,  1,  0,  0,  3,  1,  1,  0,  2,
       -1,  1,  0,  2,  0,  0,  0,  0, -1,  5,  1,  0,  0,  1,  5,  0,  5,
        1,  0,  0,  0,  0,  0,  1,  0,  0, -1,  0,  0,  0,  1,  0,  0,  0,
        2,  1,  0, -1,  0,  0,  2,  0, -1,  0,  0,  2,  0,  0,  6,  0,  0,
        1,  1,  0,  1,  0,  6,  0,  0,  0,  2,  0,  0,  1,  0,  0,  0,  1,
        0,  0,  3,  1,  0,  1,  0,  1,  0,  0,  2,  3,  5,  0,  5,  0,  0,
        1,  1,  0,  1,  0,  2,  0,  0,  1,  6,  5,  0,  4,  5,  3,  0,  0,
        1,  0,  0,  6, -1,  0,  0,  0,  6,  0,  0,  3,  0,  1,  0,  5,  0,
        2,  4,  1,  0,  3,  0,  0,  3,  3,  0,  0,  2,  4,  0,  2,  0,  0,
        0,  4,  0,  5,  3, -1,  0,  0,  4,  4,  2,  5,  0,  0,  1,  0,  0,
        0,  2,  3,  0,  0,  0,  0,  0,  0,  1,  0,  0,  1,  3,  0,  0,  4,
        4,  5,  4,  0,  0,  0,  0, -1, -1,  0,  0,  1,  4,  0,  0,  4,  0,
        0,  0,  1,  4,  0,  0,  1,  0,  4,  0,  5,  0, -1,  0,  0,  0,  0,
        0,  2,  0,  4,  2,  0,  3,  0,  1,  7,  0, -1,  1,  5,  1,  1,  1,
        0,  1,  2, -1,  0,  5,  1,  0,  0,  0,  0, -1,  0,  0,  0,  1,  1,
        4,  3,  6,  0,  6,  0,  0,  1,  0,  0,  1,  3,  1,  2,  3,  0,  0,
        3,  1,  0,  0,  1, -1,  3,  0,  3,  0,  0,  0, -1,  1,  1,  4,  5,
        0,  1,  1,  0,  1,  1,  1,  1,  0,  0,  2,  3,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  4, -1,  0,  1,  0,  0,  0,  1,  2,  0,  1,
        0,  1,  0,  1,  0,  0,  0,  0,  0, -1,  2,  0,  1,  4,  2,  0,  6,
        1,  0,  6,  0,  4,  3,  1,  0,  1,  0,  0,  0,  1, -1,  0,  1,  1,
        0,  1,  0,  0,  1,  4,  0,  1,  0,  3,  0,  0,  0,  0,  1,  5,  0,
        0, -1,  1,  3, -1,  3,  4,  0,  0,  0, -1,  1,  3,  0, -1,  3,  0,
       -1,  0, -1,  0,  1,  6,  0,  0,  1,  3,  0,  0,  2,  0,  0, -1,  1,
        0, -1,  0,  3,  1,  0,  0,  0,  2,  2,  0,  1,  1,  4, -1,  4,  1,
       -1,  1,  0,  0,  0,  4,  4, -1,  6,  1,  0,  0,  1,  0,  3,  0,  0,
        2,  0, -1,  0,  5,  3,  2,  0,  0,  0,  0,  0,  3,  0,  0,  4,  0,
        4,  5,  0,  5,  3,  0,  1,  2,  0,  4,  2,  1,  0,  0,  1,  7,  0,
        2,  1,  0,  1,  2,  0,  3,  4,  0,  0,  0,  0, -1,  1,  0,  0, -1,
        0,  1,  0,  0,  1,  3,  2,  0,  1,  0,  0,  3,  0,  3,  0,  1,  0,
        0,  1,  0,  1,  0,  5,  5,  0,  1,  4,  0,  0,  0,  4,  0,  0,  4,
        0,  4,  0,  0,  1,  0,  2,  3,  0,  1,  3,  1,  1,  3,  0,  0,  4,
        0,  1,  6,  3,  0,  3,  0,  0,  2,  0,  0,  0,  0, -1,  0,  1,  0,
        2,  0,  1,  4,  0,  0, -1,  1,  0,  3,  0,  1,  3,  1,  1,  4,  1,
        1,  0,  0,  2,  3,  0,  1,  0,  3,  0,  0,  0, -1,  1,  5,  0,  0,
        7, -1,  1,  0,  0,  5,  0,  0,  4,  0,  0,  0,  0,  1,  0,  0,  0,
        0,  1,  4,  0,  1,  0,  2,  1,  3,  1,  1,  0,  1,  0,  0,  0,  5,
        0,  0,  0,  0,  0,  2,  0,  0,  0,  2,  1,  4,  1,  2,  2,  0,  1,
        0,  1,  1,  5,  1,  0,  1,  0,  3,  4,  0, -1, -1,  0,  0, -1, -1,
        1,  1,  0,  1,  1,  1,  0,  1,  0,  5,  0,  0,  0,  0,  1,  5,  0,
        0,  0,  7,  2,  0,  0,  3,  3,  0,  0,  3,  3,  1,  4,  0,  0, -1,
        1,  0,  4,  5,  0, -1,  0,  0,  1,  1,  2,  0,  0,  0,  0,  3,  1,
        0,  0,  0,  0,  0,  1,  3,  4,  0,  0,  1,  0,  0,  0,  3,  1,  0,
        3,  4,  0,  4,  5,  4,  4,  0,  0,  4,  0,  0,  4,  1,  0,  0,  4,
        3,  1,  2,  1,  0,  0,  0,  0,  1, -1,  0,  3,  4,  4,  2,  0,  5,
        1,  0,  0,  4,  5,  0,  4,  1,  0,  0, -1,  0,  2,  3,  1,  0])
```

[7]:
```
dbcan.labels_
```

[7]:
```
array([ 0,  0,  0,  0,  0,  1,  0,  0,  1,  1,  2,  0,  0,  0,  0,  0,  1,
        3,  3,  0,  2,  0,  0,  0, -1,  0,  0,  0,  3,  0,  1,  4,  1,  3,
        0,  0,  3,  3,  3,  1,  3, -1,  0,  0,  3,  1,  1,  4,  0,  0,  0,
        3,  0,  1,  0,  0, -1,  0,  1,  0,  0,  1,  3,  5,  0,  0,  0,  0,
        0,  4,  0, -1,  1,  0,  2,  0,  0,  0,  4,  0,  6,  3,  0,  4,  1,
        4,  1,  0,  3,  0,  1,  0,  3,  0,  0,  1,  0,  1,  0,  1,  0,  1,
        0,  1,  1,  0,  0,  0,  0,  2,  0, -1,  1,  0,  4,  0,  3,  3,  3,
        0,  1,  3,  5,  1,  4, -1,  0,  0,  1,  0,  0,  3,  1,  1,  0,  2,
       -1,  1,  0,  2,  0,  0,  0,  0, -1,  5,  1,  0,  0,  1,  5,  0,  5,
        1,  0,  0,  0,  0,  0,  1,  0,  0, -1,  0,  0,  0,  1,  0,  0,  0,
        2,  1,  0, -1,  0,  0,  2,  0, -1,  0,  0,  2,  0,  0,  6,  0,  0,
        1,  1,  0,  1,  0,  6,  0,  0,  0,  2,  0,  0,  1,  0,  0,  0,  1,
        0,  0,  3,  1,  0,  1,  0,  1,  0,  0,  2,  3,  5,  0,  5,  0,  0,
        1,  1,  0,  1,  0,  2,  0,  0,  1,  6,  5,  0,  4,  5,  3,  0,  0,
        1,  0,  0,  6, -1,  0,  0,  0,  6,  0,  0,  3,  0,  1,  0,  5,  0,
        2,  4,  1,  0,  3,  0,  0,  3,  3,  0,  0,  2,  4,  0,  2,  0,  0,
        0,  4,  0,  5,  3, -1,  0,  0,  4,  4,  2,  5,  0,  0,  1,  0,  0,
        0,  2,  3,  0,  0,  0,  0,  0,  0,  1,  0,  0,  1,  3,  0,  0,  4,
        4,  5,  4,  0,  0,  0,  0, -1, -1,  0,  0,  1,  4,  0,  0,  4,  0,
        0,  0,  1,  4,  0,  0,  1,  0,  4,  0,  5,  0, -1,  0,  0,  0,  0,
        0,  2,  0,  4,  2,  0,  3,  0,  1,  7,  0, -1,  1,  5,  1,  1,  1,
        0,  1,  2, -1,  0,  5,  1,  0,  0,  0,  0, -1,  0,  0,  0,  1,  1,
        4,  3,  6,  0,  6,  0,  0,  1,  0,  0,  1,  3,  1,  2,  3,  0,  0,
        3,  1,  0,  0,  1, -1,  3,  0,  3,  0,  0,  0, -1,  1,  1,  4,  5,
        0,  1,  1,  0,  1,  1,  1,  1,  0,  0,  2,  3,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  4, -1,  0,  1,  0,  0,  0,  1,  2,  0,  1,
        0,  1,  0,  1,  0,  0,  0,  0,  0, -1,  2,  0,  1,  4,  2,  0,  6,
        1,  0,  6,  0,  4,  3,  1,  0,  1,  0,  0,  0,  1, -1,  0,  1,  1,
        0,  1,  0,  0,  1,  4,  0,  1,  0,  3,  0,  0,  0,  0,  1,  5,  0,
        0, -1,  1,  3, -1,  3,  4,  0,  0,  0, -1,  1,  3,  0, -1,  3,  0,
       -1,  0, -1,  0,  1,  6,  0,  0,  1,  3,  0,  0,  2,  0,  0, -1,  1,
        0, -1,  0,  3,  1,  0,  0,  0,  2,  2,  0,  1,  1,  4, -1,  4,  1,
       -1,  1,  0,  0,  0,  4,  4, -1,  6,  1,  0,  0,  1,  0,  3,  0,  0,
        2,  0, -1,  0,  5,  3,  2,  0,  0,  0,  0,  0,  3,  0,  0,  4,  0,
        4,  5,  0,  5,  3,  0,  1,  2,  0,  4,  2,  1,  0,  0,  1,  7,  0,
        2,  1,  0,  1,  2,  0,  3,  4,  0,  0,  0,  0, -1,  1,  0,  0, -1,
        0,  1,  0,  0,  1,  3,  2,  0,  1,  0,  0,  3,  0,  3,  0,  1,  0,
        0,  1,  0,  1,  0,  5,  5,  0,  1,  4,  0,  0,  0,  4,  0,  0,  4,
        0,  4,  0,  0,  1,  0,  2,  3,  0,  1,  3,  1,  1,  3,  0,  0,  4,
        0,  1,  6,  3,  0,  3,  0,  0,  2,  0,  0,  0,  0, -1,  0,  1,  0,
        2,  0,  1,  4,  0,  0, -1,  1,  0,  3,  0,  1,  3,  1,  1,  4,  1,
        1,  0,  0,  2,  3,  0,  1,  0,  3,  0,  0,  0, -1,  1,  5,  0,  0,
        7, -1,  1,  0,  0,  5,  0,  0,  4,  0,  0,  0,  0,  1,  0,  0,  0,
        0,  1,  4,  0,  1,  0,  2,  1,  3,  1,  1,  0,  1,  0,  0,  0,  5,
        0,  0,  0,  0,  0,  2,  0,  0,  0,  2,  1,  4,  1,  2,  2,  0,  1,
        0,  1,  1,  5,  1,  0,  1,  0,  3,  4,  0, -1, -1,  0,  0, -1, -1,
        1,  1,  0,  1,  1,  1,  0,  1,  0,  5,  0,  0,  0,  0,  1,  5,  0,
        0,  0,  7,  2,  0,  0,  3,  3,  0,  0,  3,  3,  1,  4,  0,  0, -1,
        1,  0,  4,  5,  0, -1,  0,  0,  1,  1,  2,  0,  0,  0,  0,  3,  1,
        0,  0,  0,  0,  0,  1,  3,  4,  0,  0,  1,  0,  0,  0,  3,  1,  0,
        3,  4,  0,  4,  5,  4,  4,  0,  0,  4,  0,  0,  4,  1,  0,  0,  4,
        3,  1,  2,  1,  0,  0,  0,  0,  1, -1,  0,  3,  4,  4,  2,  0,  5,
        1,  0,  0,  4,  5,  0,  4,  1,  0,  0, -1,  0,  2,  3,  1,  0])
```

[8]:
```
plt.scatter(x[:,0], x[:,1], c = dbcan.labels_)
```

[8]:
```
<matplotlib.collections.PathCollection at 0x7fcf72fe6cb0>
```



[ ]:
```

```

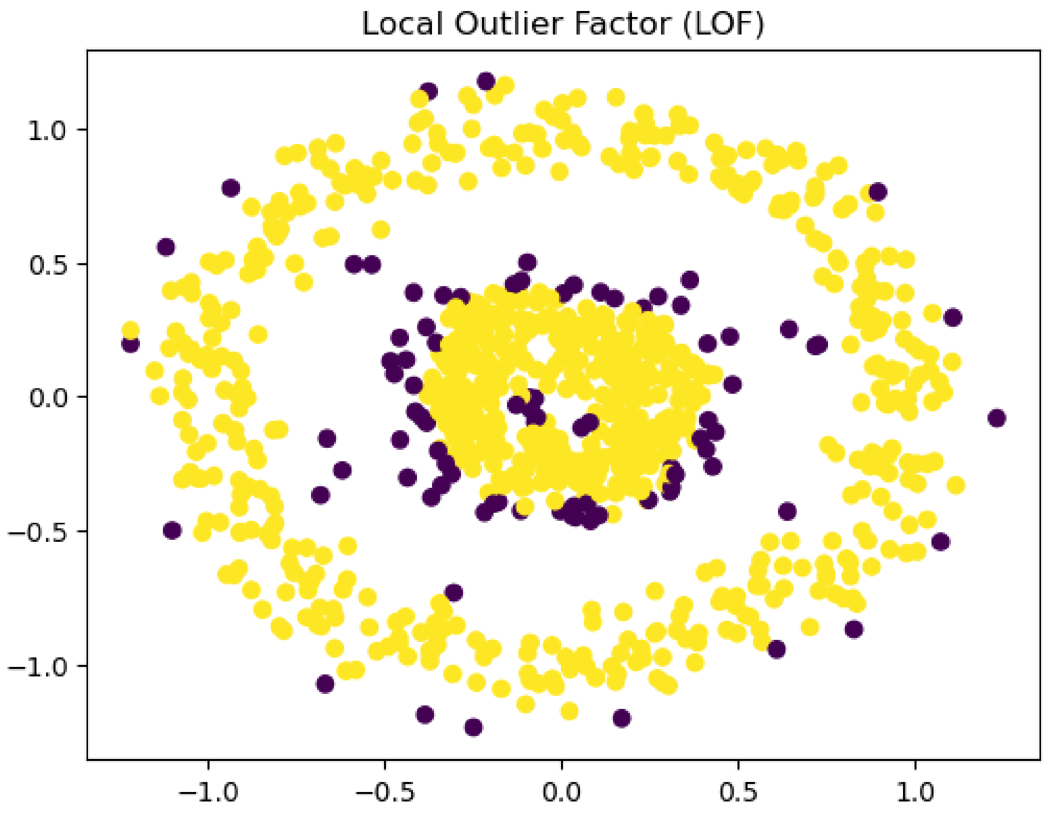
1.1 Local Outlier Factor Anomaly Detection

[9]:
```
# importing the Lof form the same data
from sklearn.neighbors import LocalOutlierFactor

x,y = make_circles(n_samples= 900, factor= 0.3, noise= 0.1)
```

[10]:
```
#creating the instance of lof
lof = LocalOutlierFactor(n_neighbors= 20, contamination=0.1)
y_pred = lof.fit_predict(x)
```

[11]:
```
#getting the lof score for each data points
lof_score = lof.negative_outlier_factor_

# As we know The more negative the LOF score, the more likely the data point is␣
↪an outlier.
```

```
[12]: #plots the outliers
      plt.scatter(x[:,0], x[:,1], c = y_pred)
      plt.title('Local Outlier Factor (LOF)')
      plt.show()
```



```
[13]: import numpy as np
      outlier_indices = np.where(y_pred == -1)[0]
      outlier_indices
```

```
[13]: array([  8,  12,  14,  15,  24,  35,  41,  70, 102, 112, 114, 116, 117,
             131, 140, 142, 149, 156, 193, 197, 203, 220, 223, 228, 231, 233,
             234, 249, 295, 313, 317, 329, 338, 345, 349, 357, 362, 363, 377,
             397, 401, 412, 417, 429, 445, 447, 449, 466, 471, 480, 496, 510,
             511, 524, 542, 551, 569, 576, 589, 591, 593, 596, 600, 602, 609,
             623, 630, 674, 678, 680, 684, 699, 700, 705, 731, 735, 751, 754,
             755, 759, 769, 777, 782, 795, 809, 812, 817, 834, 836, 899])
```

```
[14]: len(outlier_indices)
```

```
[14]: 90
```

2 Observatiom:

So there are 90 data points which is consider as outlier here.

```
[ ]:
```