uOttawa

ELG 5142 Ubiquitous Sensing for Smart Cities
Assignment 3

Setup required Libraries

```
!pip install pycaret[full]
```

```
!pip install markupsafe==2.0.1
```

Load Data

1. Load the dataset : Dataset_to_be_used_in_anomaly_detection

	Follower_measure_x_follower	Follower_measure_y_follower	Leader_measure_x_leader	Leader_measure_y_leader
0	-1.042570	-0.241098	-1.267957	0.414568
1	-1.056986	-0.245590	-1.165454	0.411869
2	-1.071858	-0.256787	-1.028780	0.407472
3	-1.084518	-0.257502	-0.850609	0.367564
4	-0.974811	-0.105985	-0.625045	0.236174
5	-0.808289	-0.008651	-0.417019	0.035897
6	-0.732102	-0.051811	-0.258204	-0.238741
7	-0.499133	-0.205854	-0.178043	-0.506508
8	-0.372178	-0.405159	-0.193983	-0.766137
9	-0.345284	-0.627297	-0.318578	-1.035780

2. Load the second dataset : Dataset_to_be_used_in_performance_comparison

	Follower_measure_x_follower	Follower_measure_y_follower	Leader_measure_x_leader	Leader_measure_y_leader
0	-1.042570	-0.241098	-1.267957	0.414568
1	-1.056986	-0.245590	-1.165454	0.411869
2	-1.071858	-0.256787	-1.028780	0.407472
3	-1.084518	-0.257502	-0.850609	0.367564
4	-0.974811	-0.105985	-0.625045	0.236174
5	-0.808289	-0.008651	-0.417019	0.035897
6	-0.732102	-0.051811	-0.258204	-0.238741
7	-0.499133	-0.205854	-0.178043	-0.506508
8	-0.372178	-0.405159	-0.193983	-0.766137
9	-0.345284	-0.627297	-0.318578	-1.035780

3. Get the Labels for comparison

```
anomalyTestingLabels[0:10]  # get the first ten labels
```

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
9    0
Name: labels, dtype: int64
```

4. Algorithms implementation

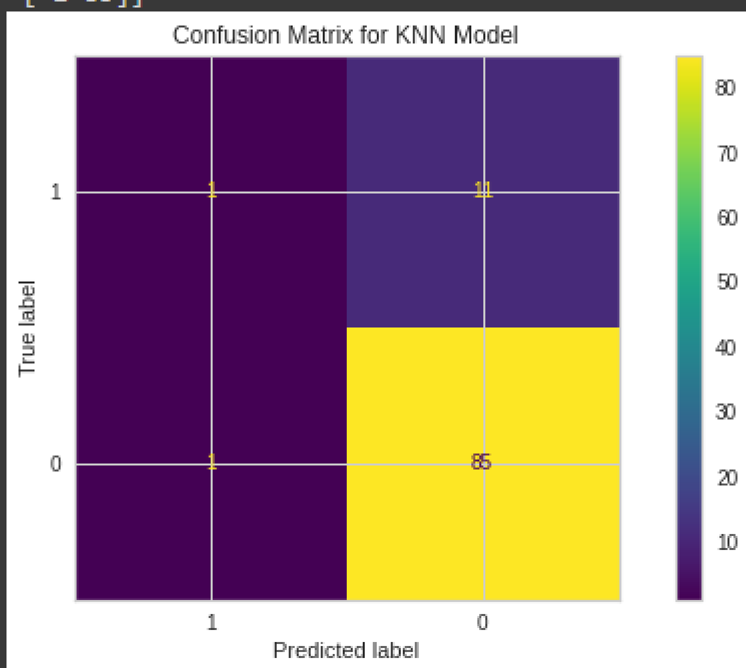
4.1 KNN

implement the KNN

```
# implementing the algorithm using Pychart
exp_name = setup(data = anomalyTrainingData,session_id=123)
knn = create_model('knn')
# return the predicted labels
knn_predictions = predict_model(model = knn, data = anomalyTestingData)
```

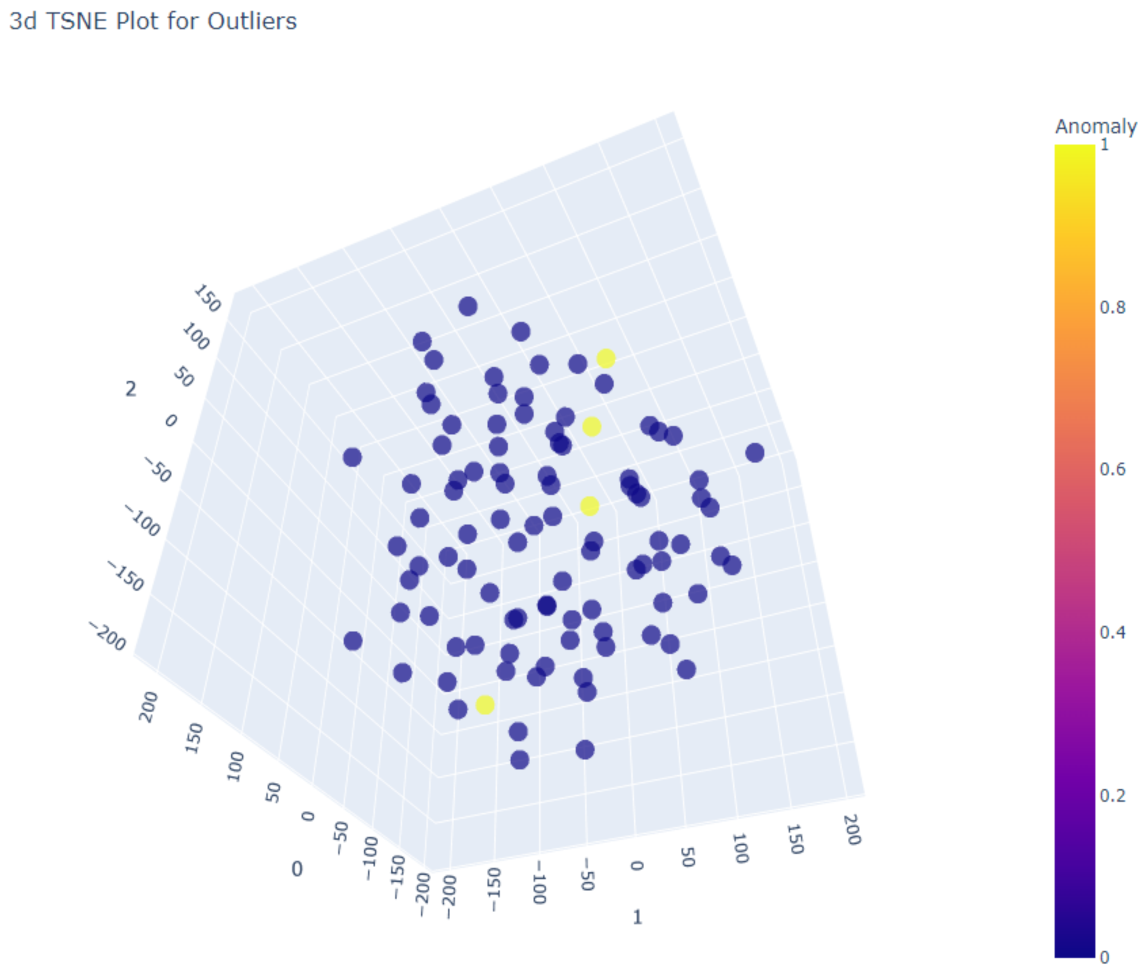
plot the confusion matrix and classification report for both anomaly and normal instances

```
[[ 1 11]
 [ 1 85]]
```

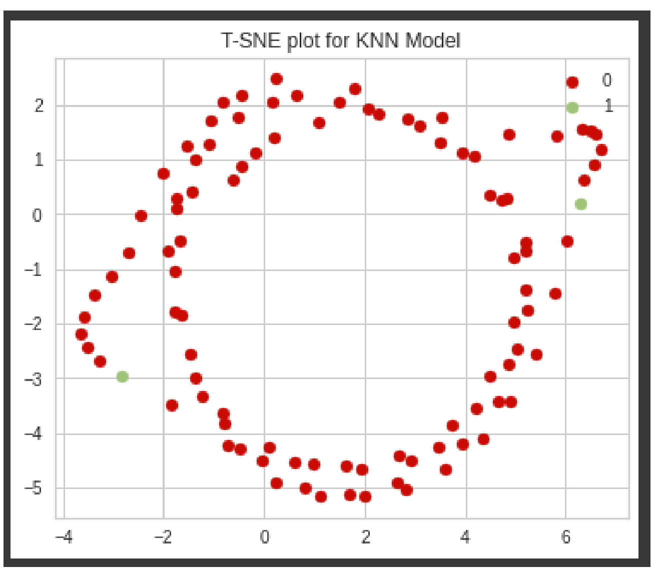



```
Anomly Detection Report for KNN Model
              precision    recall  f1-score   support

           0       0.89      0.99      0.93        86
           1       0.50      0.08      0.14        12

    accuracy                           0.88        98
   macro avg       0.69      0.54      0.54        98
weighted avg       0.84      0.88      0.84        98
```

plot 3D TSN so we can see both anomaly and normal instances



plot the 2D TSNE to see both anomaly and normal instances



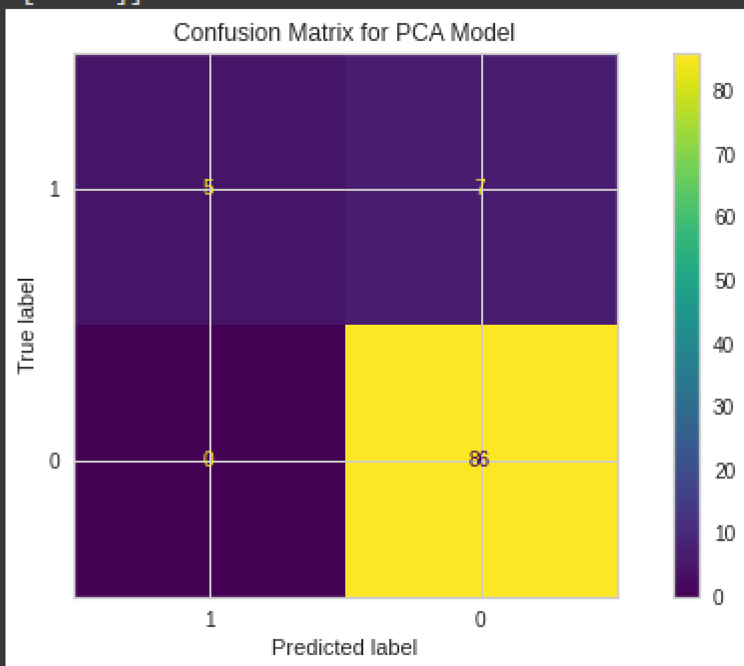
4.2 PCA

implement the PCA

```
exp_name = setup(data = anomalyTrainingData)
pca = create_model('pca')
pca_predictions = predict_model(model = pca, data = anomalyTestingData)
```

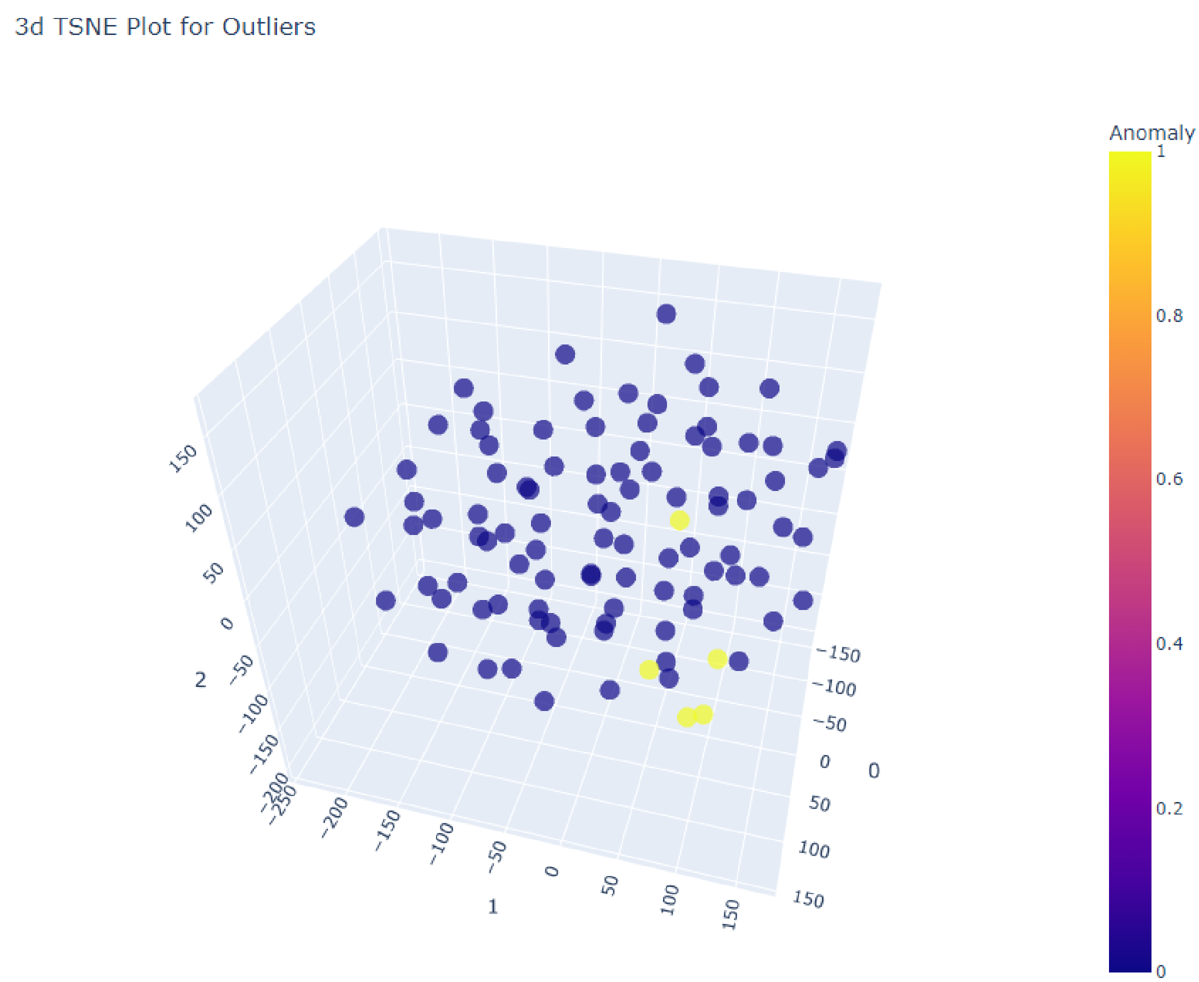
plot the confusion matrix and classification report for both anomaly and normal instances

```
[[ 5  7]
 [ 0 86]]
```

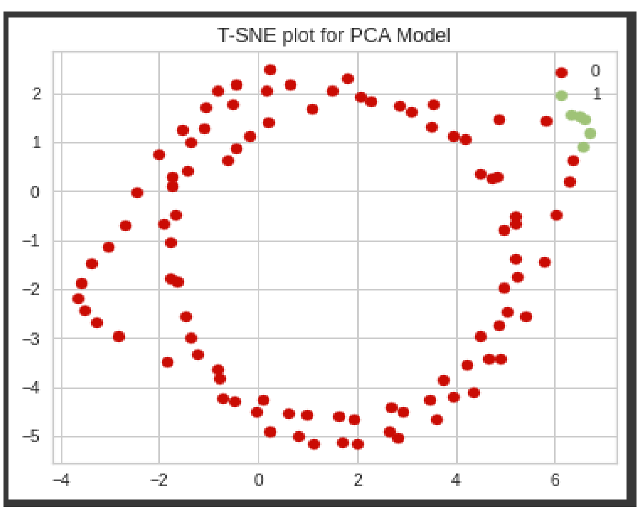



```
Anomly Detection Report for PCA Model
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        86
           1       1.00      0.42      0.59        12

    accuracy                           0.93        98
   macro avg       0.96      0.71      0.77        98
weighted avg       0.93      0.93      0.92        98
```

plot 3D TSN so we can see both anomaly and normal instances



plot the 2D TSNE to see both anomaly and normal instances



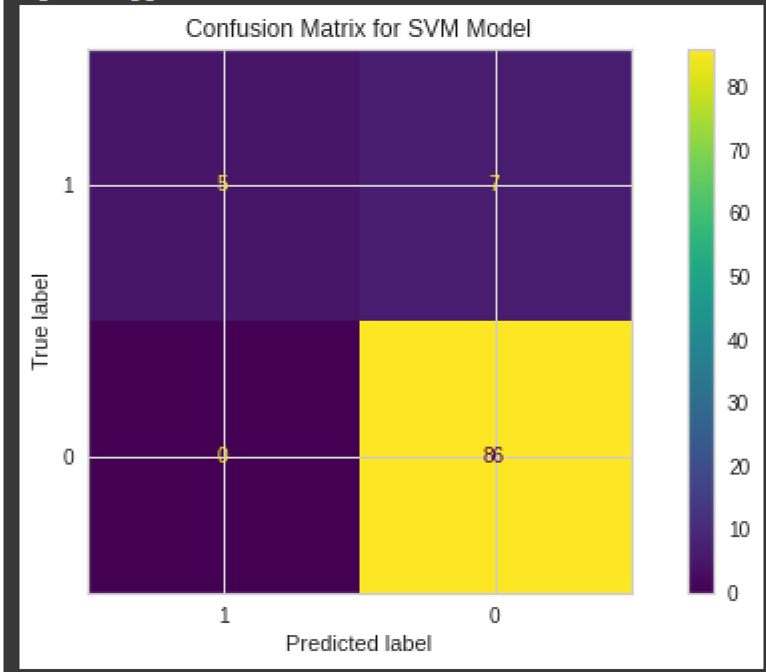
4.3 SVM

implement SVM

```
exp_name = setup(data = anomalyTrainingData)
svm = create_model('svm')
svm_predictions = predict_model(model = svm, data = anomalyTestingData)
```

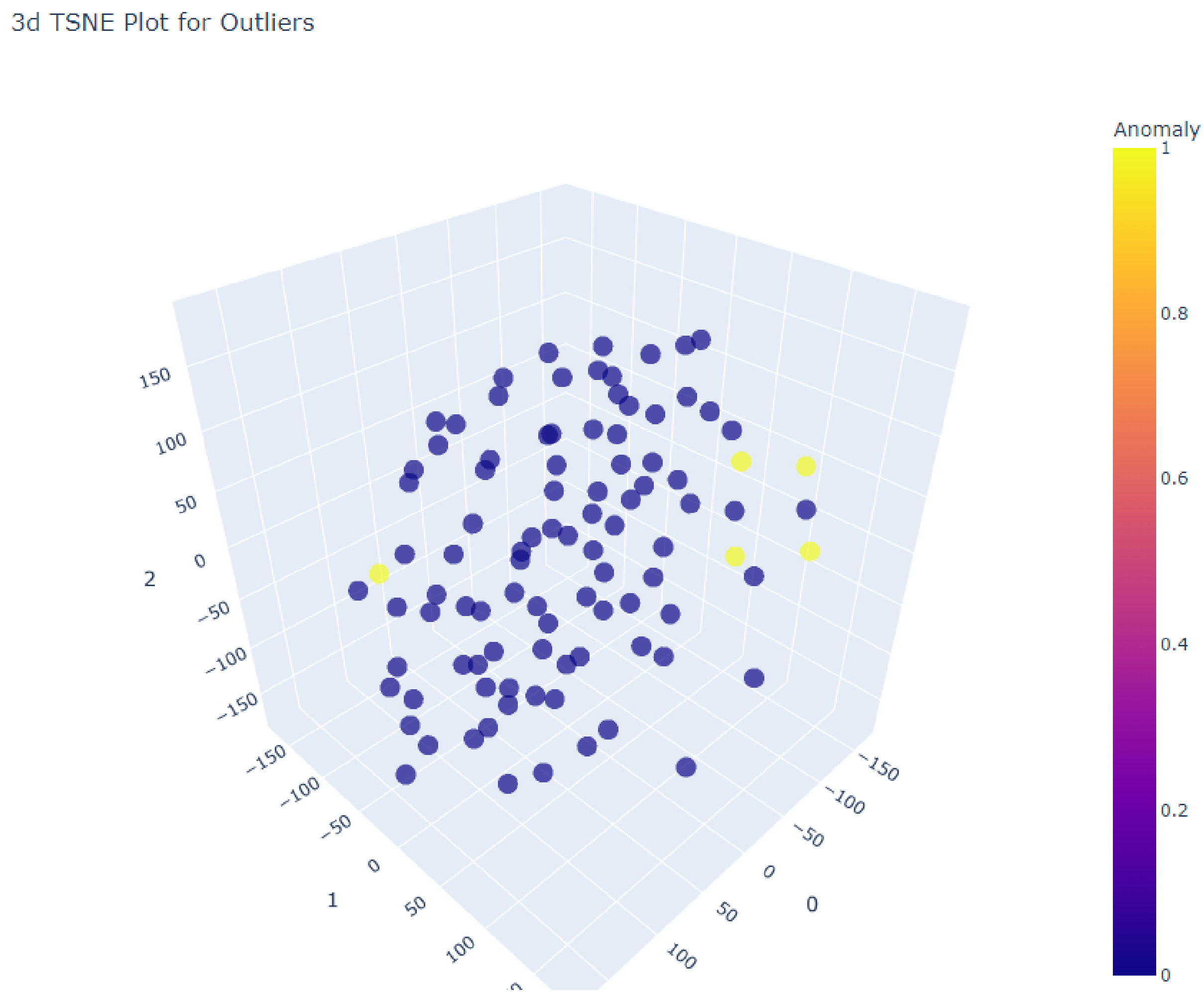
plot the confusion matrix and classification report for both anomaly and normal instances

```
[[ 5  7]
 [ 0 86]]
```

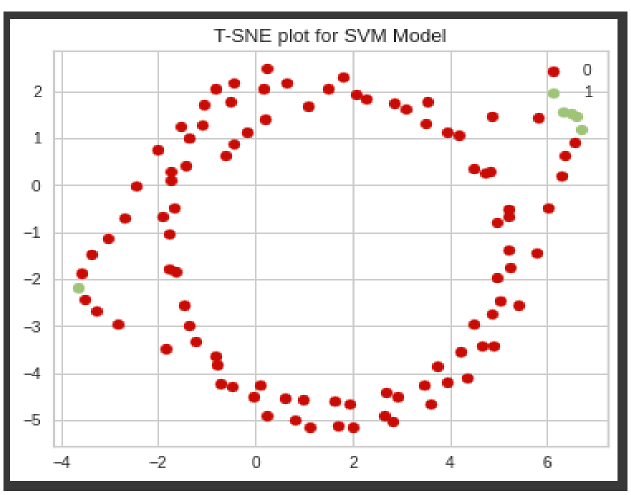



```
Anomly Detection Report for SVM Model
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        86
           1       1.00      0.42      0.59        12

    accuracy                           0.93        98
   macro avg       0.96      0.71      0.77        98
weighted avg       0.93      0.93      0.92        98
```

plot 3D TSN so we can see both anomaly and normal instances



plot the 2D TSNE to see both anomaly and normal instances



4.4 DBSCAN

implement DBSCAN using sklearn Library

Grid search to find the best parameter

```
[22] from math import nan

     max_acc=0
     best_epsilon=0.3
     best_min_samples=2
     #0.3 & 7
     parameters = {'eps':[0.3,0.4,0.5,0.6,0.7], 'min_samples':[2, 15]}
     for eps in tqdm(np.arange(0.3, 0.71,0.01)):
       for ms in range(2, 16):
         model = DBSCAN(eps=eps, min_samples=ms)
         predLabels = model.fit_predict(anomalyTrainingData)
         DBscananomalypredLabels=[]
         for val in predLabels :
           if(val!=-1):
             DBscananomalypredLabels.append('0')
           else:
             DBscananomalypredLabels.append('1')

         DBscananomalypredLabels =  [int(i) for i in DBscananomalypredLabels ]
         Dbscan_acc = accuracy_score(anomalyTestingLabels, DBscananomalypredLabels)*100
         if max_acc<Dbscan_acc:

           max_acc=Dbscan_acc
           best_epsilon=eps
           best_min_samples=ms
     print(max_acc)
     print(best_epsilon)
     print(best_min_samples)
```

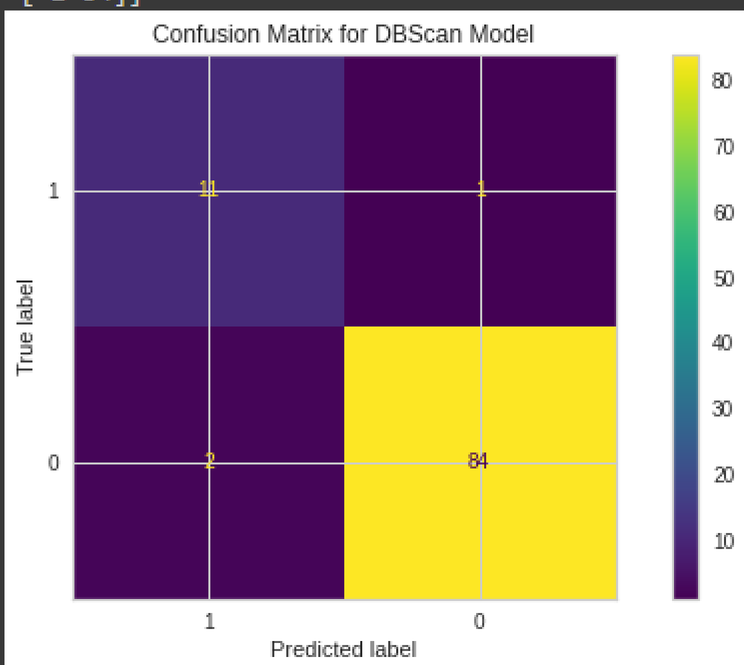
```
100%|██████████| 41/41 [00:02<00:00, 17.49it/s]96.93877551020408
0.6100000000000003
10
```

Building the model with the best parameters

```
from math import nan

model = DBSCAN(eps=best_epsilon, min_samples=best_min_samples)
predLabels = model.fit_predict(anomalyTrainingData)
DBscananomalypredLabels=[]
for val in predLabels :
  if(val!=-1):
    DBscananomalypredLabels.append('0')
  else:
    DBscananomalypredLabels.append('1')

DBscananomalypredLabels =  [int(i) for i in DBscananomalypredLabels ]
```

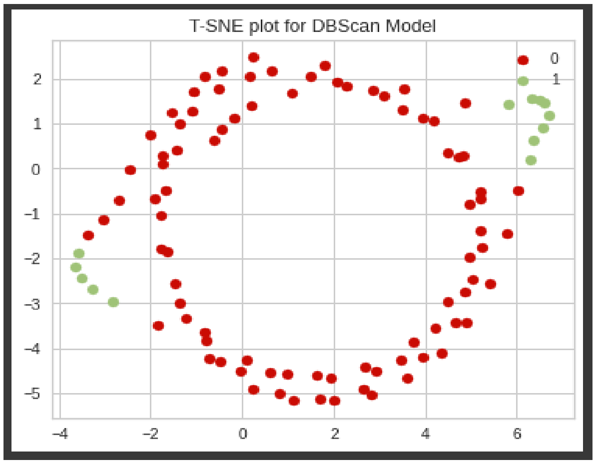
plot the confusion matrix and classification report for both anomaly and normal instances

```
[[11  1]
 [ 2 84]]
```



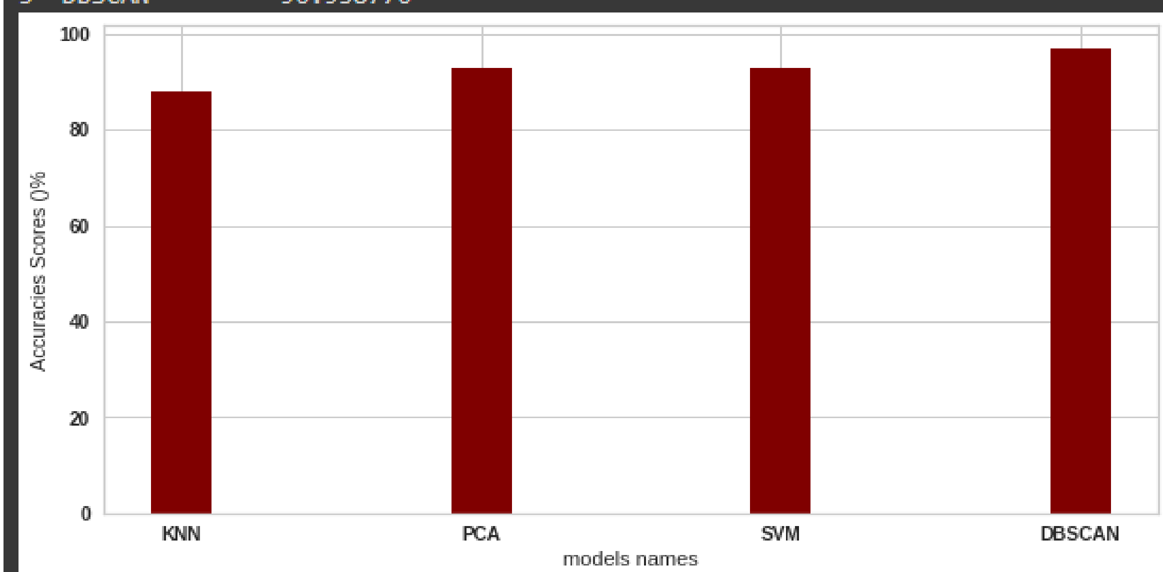

```
Anomly Detection Report for DBScan Model
              precision    recall  f1-score   support

           0       0.99      0.98      0.98        86
           1       0.85      0.92      0.88        12

    accuracy                           0.97        98
   macro avg       0.92      0.95      0.93        98
weighted avg       0.97      0.97      0.97        98
```

plot 2D TSN so we can see both anomaly and normal instances



Performance Evaluation

```
    names  Accuracies_Score
0     KNN         87.755102
1     PCA         92.857143
2     SVM         92.857143
3  DBSCAN         96.938776
```



We got a high accuracy by using DBSCAN algorithm

PCA and SVM give us the same accuracy but not as good as DBSCAN.

KNN is the worst accuracy regarding to the other accuracies.

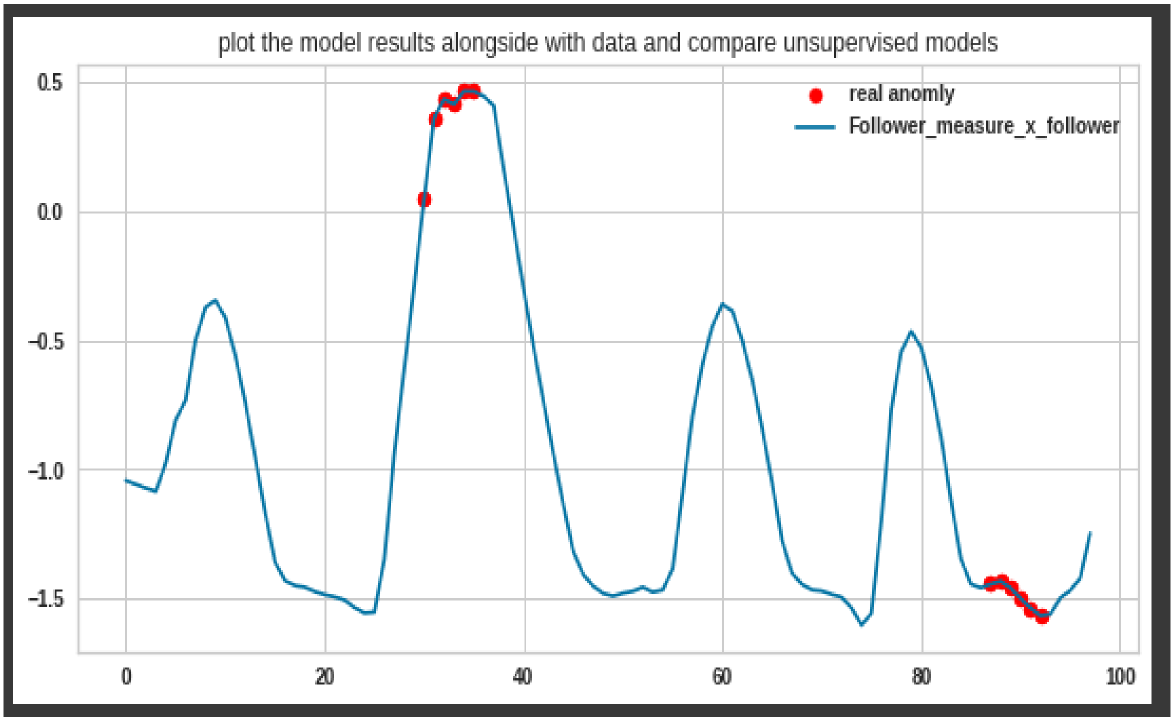
Conclusion

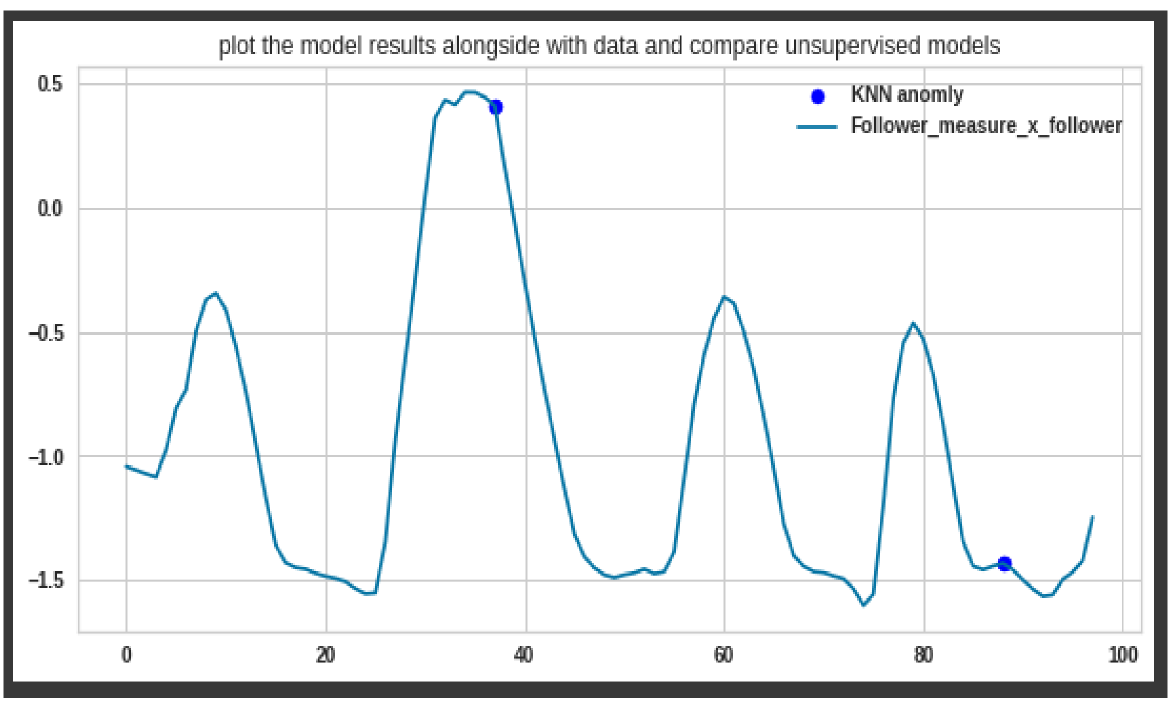
To see the effect of the model that can detect anomlies we ploted the data using one feature which is "Follower_measure_x_follower"

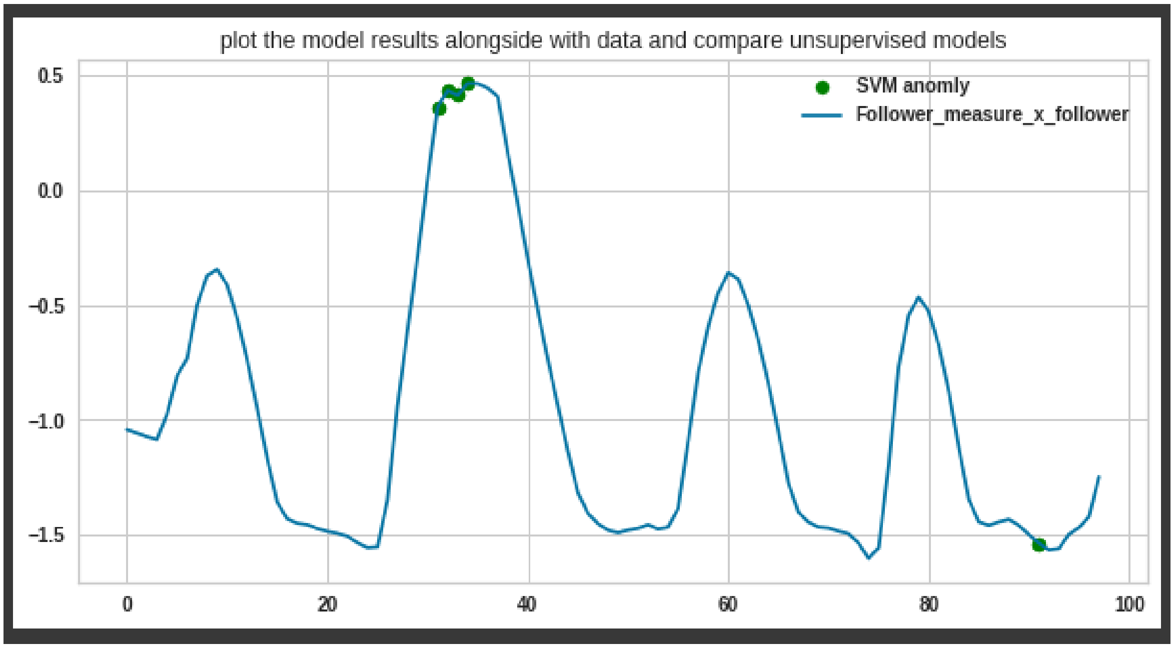
At the beginning we ploted the real anomlies regarding to our dataset.

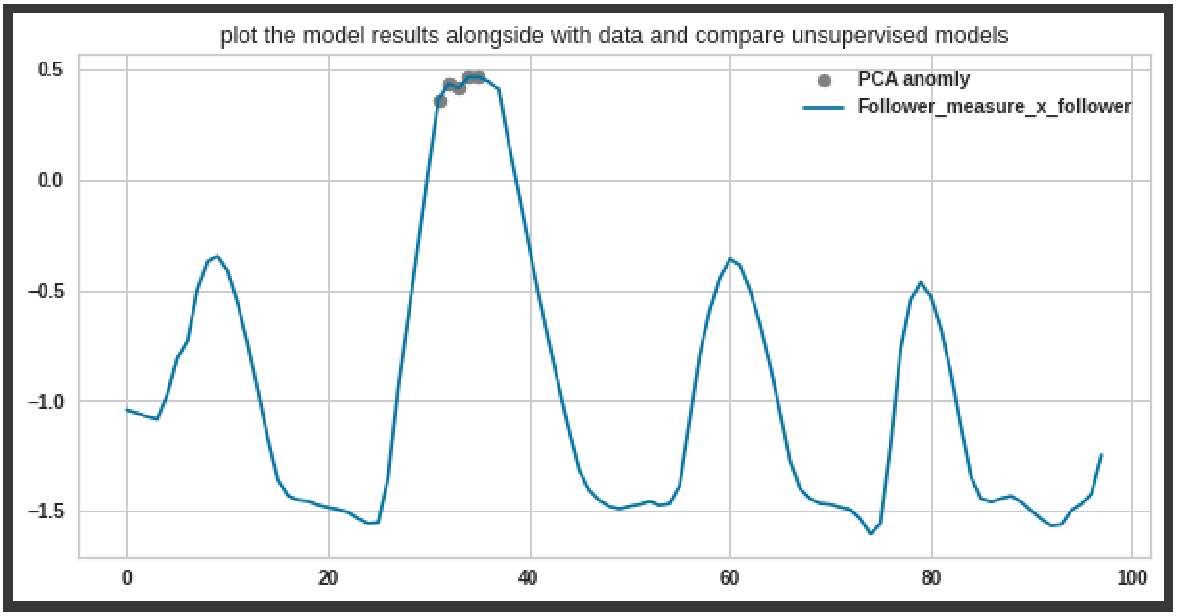
Then we ploted the anomlies regarding to what each model detect .

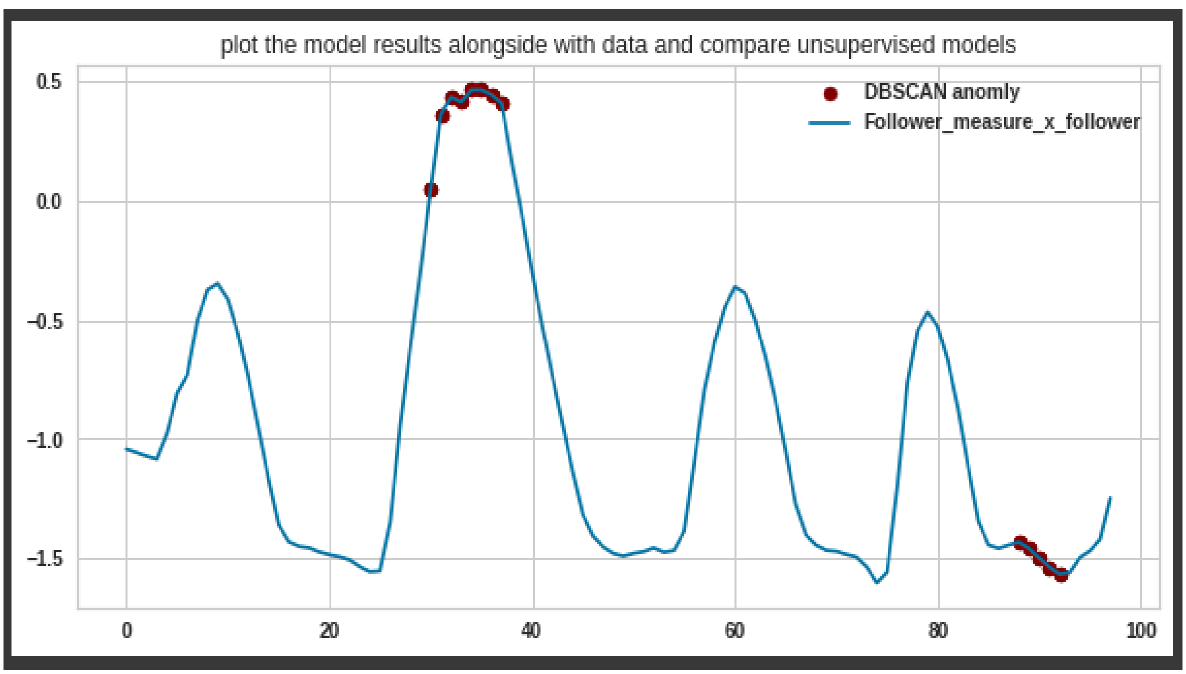
The DBSCAN model detect more than the other models .






plot the model results alongside with data and compare unsupervised models
SVM anomly
Follower_measure_x_follower


plot the model results alongside with data and compare unsupervised models
PCA anomly
Follower_measure_x_follower


plot the model results alongside with data and compare unsupervised models
DBSCAN anomly
Follower_measure_x_follower