uOttawa

ELG 5142: Ubiquitous Sensing / Smart Cities Assignment 3

Dr. Burak Kantarci

GROUP 14

Table of Contents

1. Models and model Results: 3
   A. SVM model: 3
   B. KNN model: 4
   C. PCA model: 5
   D. DBSCAN model: 6
2. TSNE plots: 7
   A. SVM model: 7
   B. KNN model: 8
   C. PCA model: 8
   D. DBSCAN model: 8
3. Performance evaluation results: 9
   A. SVM model: 10
   B. KNN model: 10
   C. PCA model: 10
   D. DBSCAN model: 11
4. Conclusive discussions: 11

Table of Figures

Figure 1 Plot charts of SVM 3
Figure 2 Plot charts of KNN 4
Figure 3 Plot charts of PCA 5
Figure 4 Plot charts of DBSCAN 6
Figure 5 2D T-SNE of SVM 7
Figure 6 3D T-SNE of SVM 7
Figure 7 2D T-SNE of KNN 8
Figure 8 3D T-SNE of KNN 8
Figure 9 3D T-SNE of PCA 8
Figure 10 2D T-SNE of PCA 8
Figure 11 2D T-SNE of DBSCAN 8
Figure 12 Classification Report of SVM 10
Figure 13 Confusion matrix of SVM 10
Figure 14 Classification Report of KNN 10
Figure 15 Confusion matrix of KNN 10
Figure 16 Confusion matrix of PCA 10
Figure 17 Classification Report of PCA 10
Figure 18 Classification Report of DBSCAN 11
Figure 19 Confusion Matrix of DBSCAN 11

1. Models and model Results:

A. SVM model:

```
ML_Model_svm = create_model('svm')
anomalies_svm = assign_model(ML_Model_svm)
```

```
results = anomalies_svm.iloc[:,:-2]
anomalies = anomalies_svm[anomalies_svm['Anomaly'] == 1].iloc[:,:-2]
```

```
results = anomalies_svm.iloc[:,:-2]
anomalies = anomalies_svm[anomalies_svm['Anomaly'] == 1].iloc[:,:-2]
anomalies.head()
for column in results.columns[1:]:
    plt.plot(anomalies_svm[column])
    plt.scatter(anomalies.index,anomalies[column],c='r',marker='o',s=60,
alpha=1)
    plt.title(" ".join(column.split('_')))
    plt.show()
```

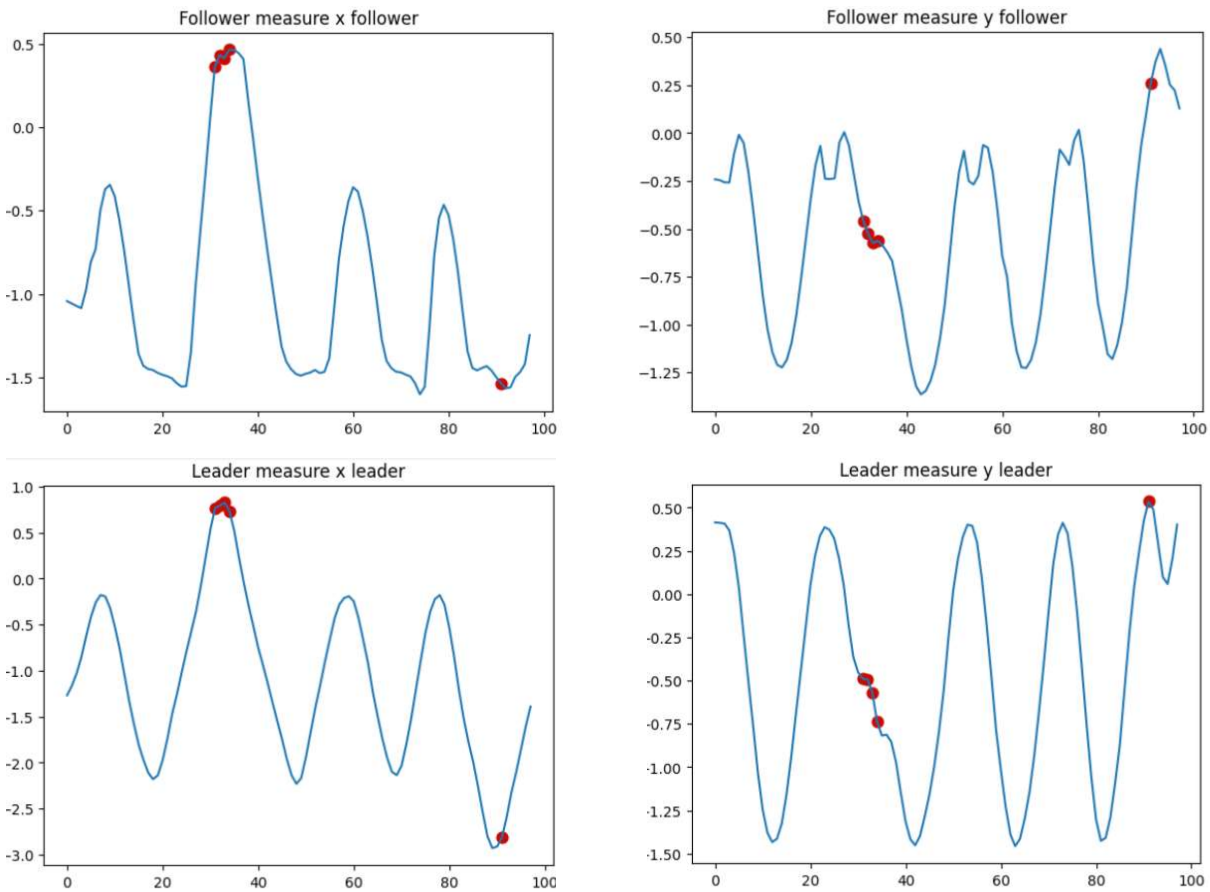

Figure 1 Plot charts of SVM

B. KNN model:

```
ML_Model_knn = create_model('knn')
anomalies_knn = assign_model(ML_Model_knn)
```

```
results = anomalies_knn.iloc[:,:-2]
anomalies = anomalies_knn[anomalies_knn['Anomaly'] == 1].iloc[:,:-2]
```

```
for column in results.columns[1:]:
    plt.plot(anomalies_knn[column])
    plt.scatter(anomalies.index,anomalies[column],c='r',marker='o',s=60,
alpha=1)
    plt.title(" ".join(column.split('_')))
    plt.show()
```

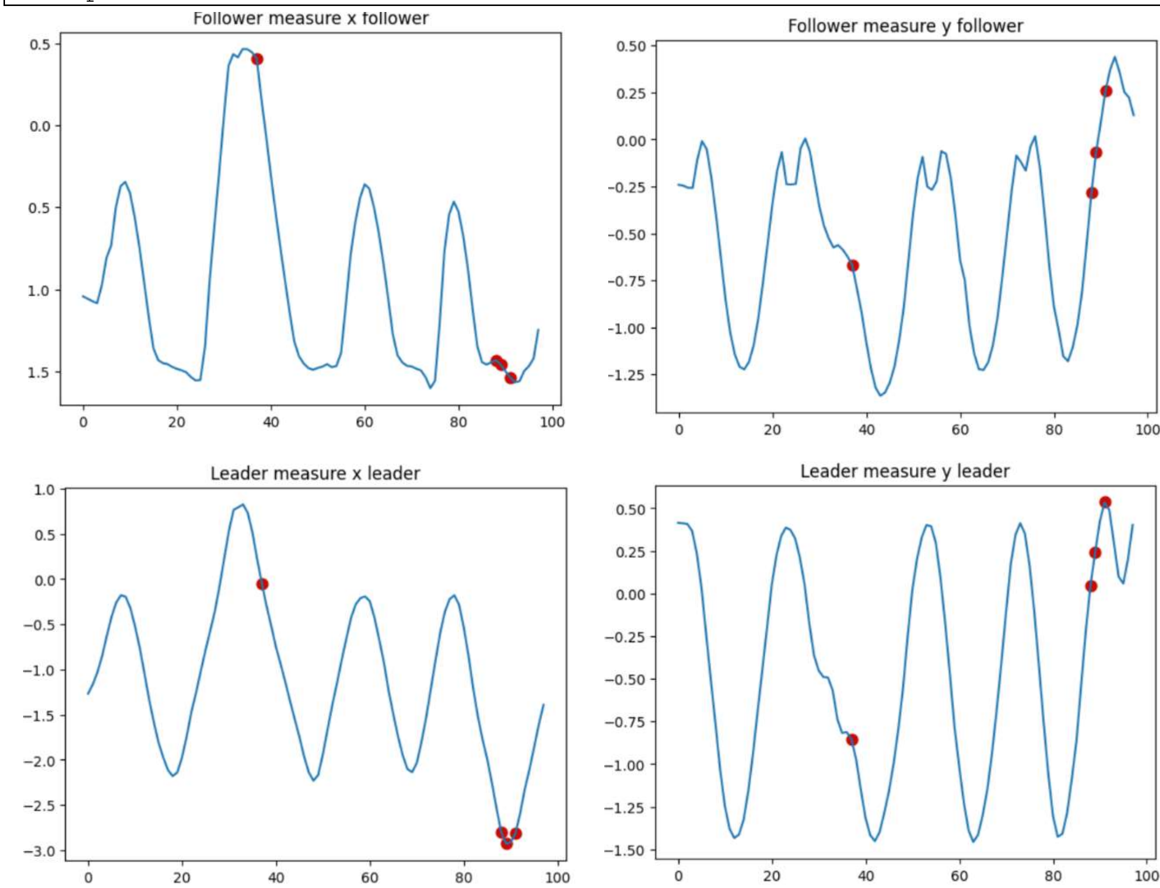

Figure 2 Plot charts of KNN

C. PCA model:

```
ML_Model_pca = create_model('pca')
anomalies_pca = assign_model(ML_Model_pca)
```

```
results = anomalies_pca.iloc[:,:-2]
anomalies = anomalies_pca[anomalies_pca['Anomaly'] == 1].iloc[:,:-2]
```

```
for column in results.columns[1:]:
    plt.plot(anomalies_pca[column])
    plt.scatter(anomalies.index,anomalies[column],c='r',marker='o',s=60,
alpha=1)
    plt.title(" ".join(column.split('_')))
    plt.show()
```

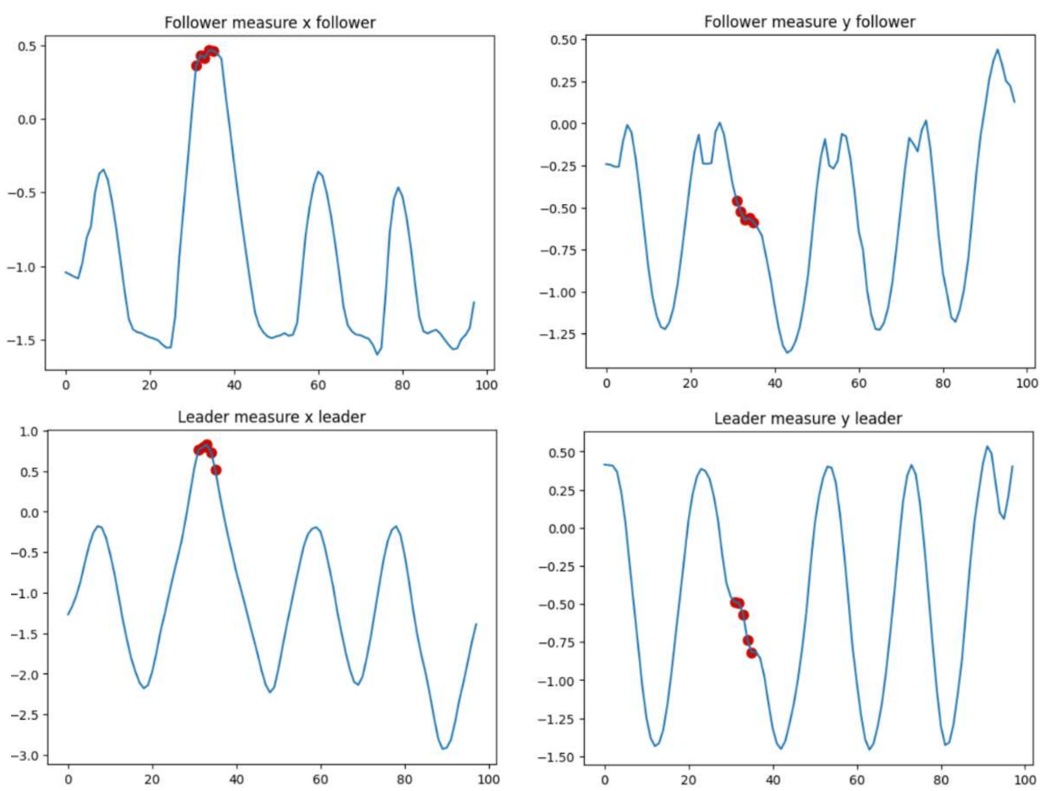

Figure 3 Plot charts of PCA

D. DBSCAN model:

To choose the values of epsilon and minpoints for DBSCAN model, a simple hyperparameter search is done to get the parameters that will make the model give the highest F1 score.

```
best_pred=None
best=0
for eps in np.arange(0.1, 0.75, 0.01):
  for ms in range(2, 16):
    db = DBSCAN(eps=eps, min_samples=ms)
    predLabels = db.fit_predict(X)
    predLabels[predLabels != -1]=0
    predLabels[predLabels ==-1]=1
    acc=f1_score(y,predLabels)
        if(best<=acc):
      best_pred=predLabels
      best=acc
```

```
df2=df2.iloc[:,:-1]
```

```
anomalies_DBSC=df2.iloc[:,:-1]
anomalies_DBSC['Anomaly'] = best_pred
```

```
for column in results.columns[1:]:
    plt.plot(anomalies_DBSC[column])
    plt.scatter(anomalies.index,anomalies[column],c='r',marker='o',s=60,
alpha=1)
    plt.title(" ".join(column.split('_')))
    plt.show()
```

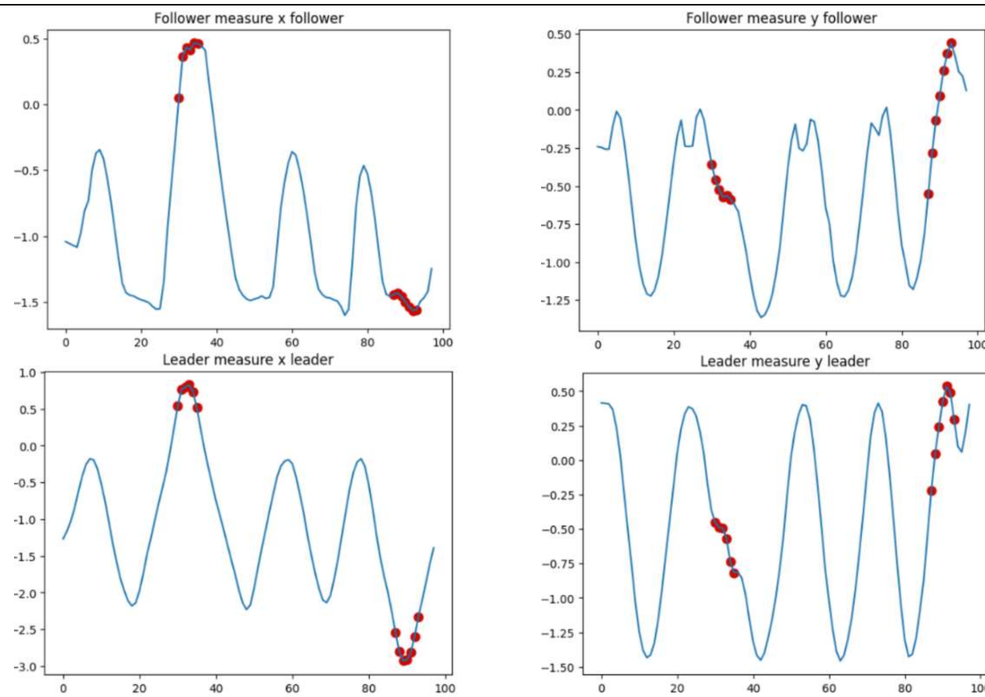

Figure 4 Plot charts of DBSCAN

2. TSNE plots:

```
def tsne(X,y,k,title):
  tsne = TSNE(n_components=2, verbose=1, random_state=0)
  z = tsne.fit_transform(X)
  df = pd.DataFrame()
  df["y"] = y
  df["comp-1"] = z[:,0]
  df["comp-2"] = z[:,1]

  sns.scatterplot(x="comp-1", y="comp-2", hue=df.y.tolist(),
                  palette=sns.color_palette("hls", k),
                  data=df).set(title=title)
  plt.show()
```

A. SVM model:

`plot_model(ML_Model_svm,'tsne')`	`tsne(X,anomalies_svm['Anomaly'],2,'DBSC Model')`

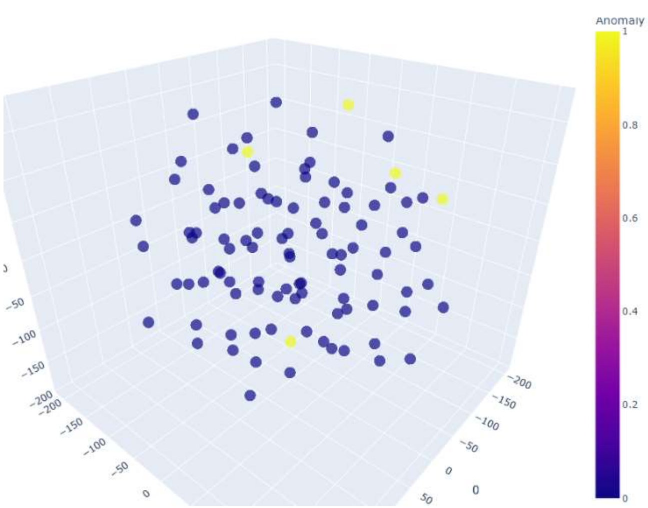

Figure 6 3D T-SNE of SVM

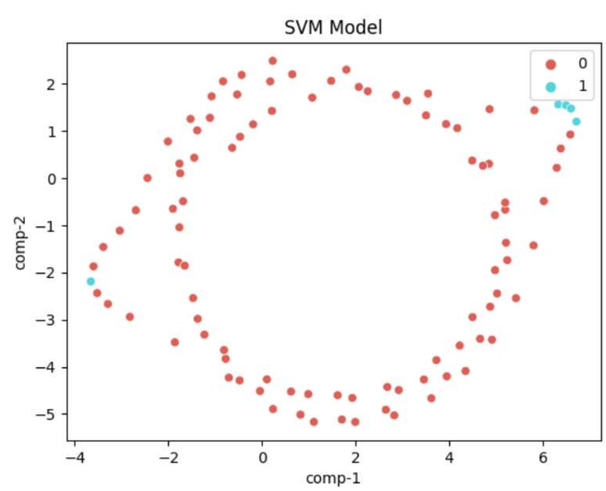

Figure 5 2D T-SNE of SVM

B. KNN model:

`plot_model(ML_Model_knn,'tsne')`	`tsne(X,anomalies_knn['Anomaly'],2,'KNN Model')`

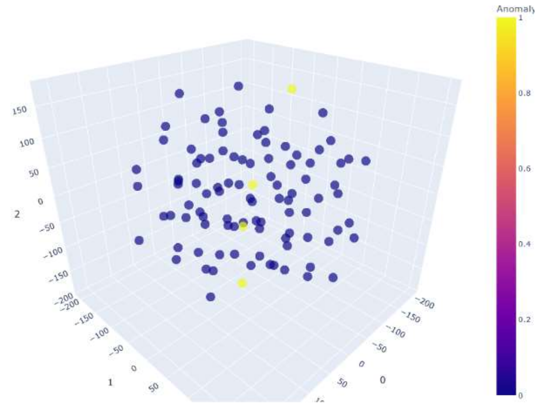

Figure 8 3D T-SNE of KNN

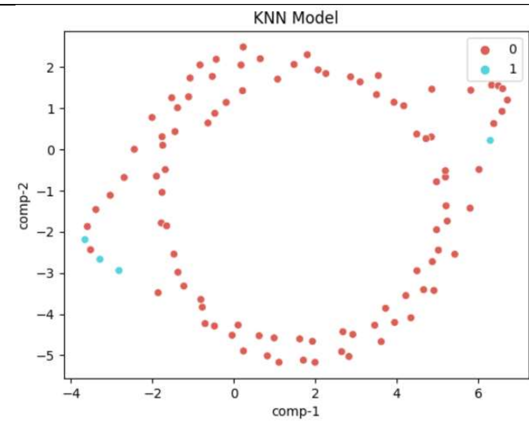

Figure 7 2D T-SNE of KNN

C. PCA model:

`plot_model(ML_Model_pca,'tsne')`	`tsne(X,anomalies_pca['Anomaly'],2,'PCA Model')`

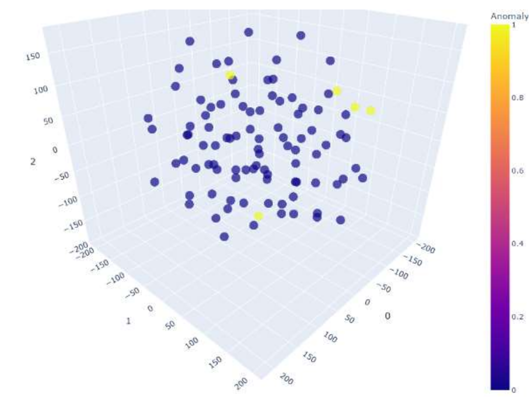

Figure 9 3D T-SNE of PCA

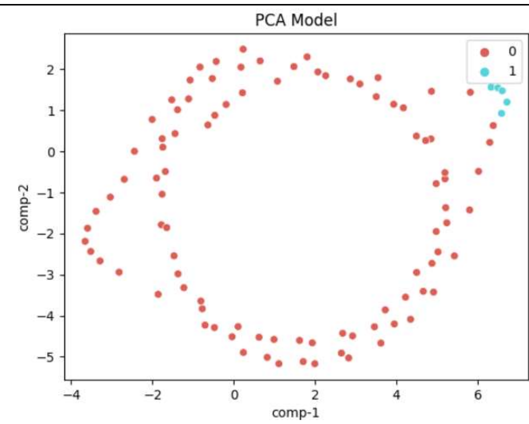

Figure 10 2D T-SNE of PCA

D. DBSCAN model:

```
tsne(X,anomalies_DBSC['Anomaly'],2,'DBSC Model')
```

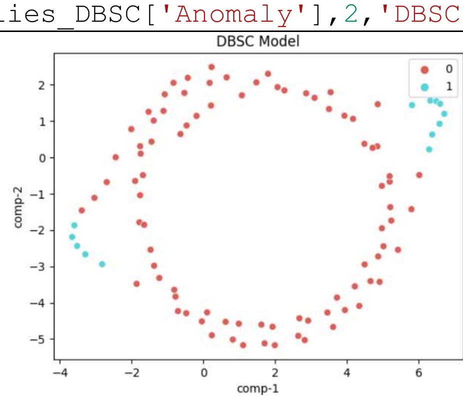

Figure 11 2D T-SNE of DBSCAN

3. Performance evaluation results:

```
def eval(y_test,y_pred):

  acc=accuracy_score(y_test,y_pred)
  print("_________________________________________________________
________________")
  print("The Test acceracy :",acc)
  print("_________________________________________________________
________________")
  print(classification_report(y_test,y_pred))
  print("_________________________________________________________
________________")

  pr,re,fc,c=precision_recall_fscore_support(y_test, y_pred, avera
ge='macro')
  pr1,re1,fc1,c=precision_recall_fscore_support(y_test, y_pred, av
erage='weighted')
  print("The macro precision on test data is  :",pr)
  print("The weighted precision on test data is  :",pr1)
  print("_________________________________________________________
________________")
  print("The macro recall on test data is  :",re)
  print("The weighted recall on test data is  :",re1)
  print("_________________________________________________________
________________")
  print("The macro f_score on test data is  :",fc)
  print("The weighted f_score on test data is  :",fc1)
  print("_________________________________________________________
________________")
  cm = confusion_matrix(y_test,y_pred)
  ConfusionMatrixDisplay(confusion_matrix=cm).plot()

  plt.show()
```

A. SVM model:

```
eval(df2['labels'],anomalies_svm['Anomaly'])
```

```
The Test accuracy : 0.9285714285714286
_____________________________________________________________
              precision    recall  f1-score   support

         0.0       0.92      1.00      0.96        86
         1.0       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

_____________________________________________________________
The macro precision on test data is   : 0.9623655913978495
The weighted precision on test data is  : 0.93394777265745
_____________________________________________________________
The macro recall on test data is   : 0.7083333333333334
The weighted recall on test data is   : 0.9285714285714286
_____________________________________________________________
The macro f_score on test data is   : 0.7745645744331251
The weighted f_score on test data is   : 0.9152621942631801
```

Figure 12 Classification Report of SVM

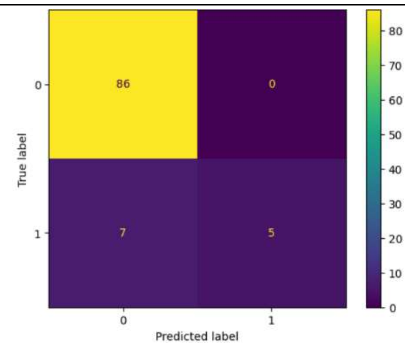

Figure 13 Confusion matrix of SVM

B. KNN model:

```
eval(df2['labels'],anomalies_knn['Anomaly'])
```

```
The Test accuracy : 0.8979591836734694
_____________________________________________________________
              precision    recall  f1-score   support

         0.0       0.90      0.99      0.94        86
         1.0       0.75      0.25      0.38        12

    accuracy                           0.90        98
   macro avg       0.83      0.62      0.66        98
weighted avg       0.89      0.90      0.87        98

_____________________________________________________________
The macro precision on test data is   : 0.8271276595744681
The weighted precision on test data is   : 0.8853669127225359
_____________________________________________________________
The macro recall on test data is   : 0.6191860465116279
The weighted recall on test data is   : 0.8979591836734694
_____________________________________________________________
The macro f_score on test data is   : 0.6597222222222222
The weighted f_score on test data is   : 0.8747165532879818
```

Figure 14 Classification Report of KNN

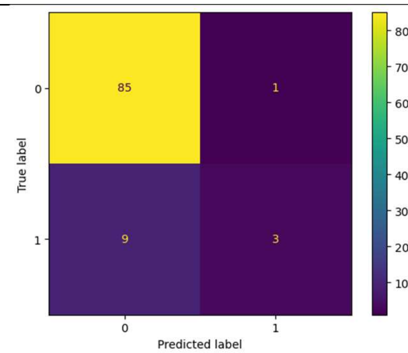

Figure 15 Confusion matrix of KNN

C. PCA model:

```
eval(df2['labels'],anomalies_pca['Anomaly'])
```

```
The Test accuracy : 0.9285714285714286
_____________________________________________________________
              precision    recall  f1-score   support

         0.0       0.92      1.00      0.96        86
         1.0       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

_____________________________________________________________
The macro precision on test data is   : 0.9623655913978495
The weighted precision on test data is   : 0.93394777265745
_____________________________________________________________
The macro recall on test data is   : 0.7083333333333334
The weighted recall on test data is   : 0.9285714285714286
_____________________________________________________________
The macro f_score on test data is   : 0.7745645744331251
The weighted f_score on test data is   : 0.9152621942631801
```

Figure 17 Classification Report of PCA

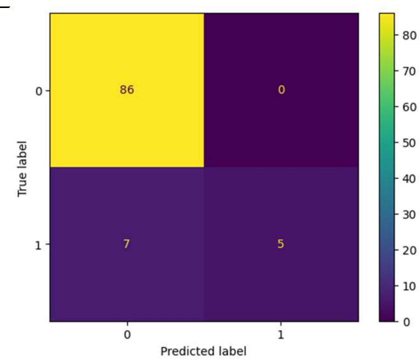

Figure 16 Confusion matrix of PCA

D. DBSCAN model:

```
eval(df2['labels'],anomalies_DBSC['Anomaly'])
```

```
______________________________________________________________
The Test accuracy : 0.9693877551020408
______________________________________________________________
              precision    recall  f1-score   support

         0.0       0.99      0.98      0.98        86
         1.0       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

______________________________________________________________
The macro precision on test data is  : 0.9171945701357467
The weighted precision on test data is  : 0.9708375657955491
______________________________________________________________
The macro recall on test data is  : 0.946705426356589
The weighted recall on test data is  : 0.9693877551020408
______________________________________________________________
The macro f_score on test data is  : 0.9312280701754385
The weighted f_score on test data is  : 0.9699104905119943
```

Figure 18 Classification Report of DBSCAN

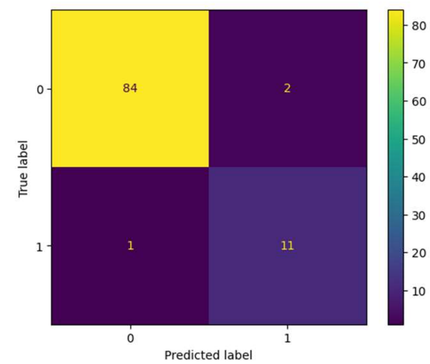

Figure 19 Confusion Matrix of DBSCAN

4. Conclusive discussions:

From the plots of model results alongside with the data, it's concluded that the outlier is detected based on the values of "leader x" feature and the DBSCAN model is the best model that managed to detect the outliers in both positive and negative extreme values.

Regarding the accuracy, DBSCAN model gave the highest accuracy (96.94%) among all models. KNN model gave the lowest accuracy (89.8%) while SVM and PCA models gave the same accuracy (92.86%).

Regarding the precision for anomaly instances, SVM and PCA models gave the highest precision (100%) among all models. KNN model gave the lowest precision (75%) while DBSCAN model gave a precision of 85%.

Regarding the precision for normal instances, DBSCAN model gave the highest precision (99%) among all models. KNN model gave the lowest precision (90%) while SVM and PCA models gave the same precision (92%).

Regarding the recall for anomaly instances, DBSCAN model gave the highest recall (92%) among all models. KNN model gave the lowest recall (25%) while SVM and PCA models gave the same recall (42%).

Regarding the recall for normal instances, SVM and PCA models gave the highest recall (100%) among all models. DBSCAN model gave the lowest accuracy (98%) while KNN model gave a recall of 99%.

Regarding the F1 score for anomaly instances, DBSCAN model gave the highest score (88%) among all models. KNN model gave the lowest score (38%) while SVM and PCA models gave the same score (59%).

Regarding the F1 score for normal instances, DBSCAN model gave the highest score (98%) among all models. KNN model gave the lowest score (94%) while SVM and PCA models gave the same score (96%).

From the above comparisons, it's concluded that DBSCAN model is the best model for anomaly detection among all models and this performance is predictable as DBSCAN is a density-based clustering algorithm so it's robust to outliers.