uOttawa

Faculté de génie
Faculty of Engineering

REPORT

ELG 5142 Ubiquitous Sensing for Smart Cities

MOSTAFA MAHMOUD ABDELWAHAB NOFAL
NADA ABDELLATEF SHAKER SEDIK
HADEER MAMDOUH ADBELFATTAH MOHAMMED
GRUOP | 11

Important packages:

First, we import used packages.

```python
#Import important packages
import pandas as pd
import numpy as np
import sklearn as sk
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import jinja2
from pycaret.anomaly import *
from sklearn.metrics import classification_report,ConfusionMatrixDisplay,confusion_matrix
%matplotlib inline
from sklearn.cluster import DBSCAN
from matplotlib import pyplot as plt
from sklearn.manifold import TSNE
from sklearn.metrics import plot_confusion_matrix
```

Read data:

```python
data = pd.read_csv('/content/Dataset_to_be_used_in_anomaly_detection.csv')
performance = pd.read_csv('/content/Dataset_to_be_used_in_performance_comparison.csv')
```

Data Description:

```python
data
```

	Unnamed: 0	Follower_measure_x_follower	Follower_measure_y_follower	Leader_measure_x_leader	Leader_measure_y_leader
0	9	-1.042570	-0.241098	-1.267957	0.414568
1	10	-1.056986	-0.245590	-1.165454	0.411869
2	11	-1.071858	-0.256787	-1.028780	0.407472
3	12	-1.084518	-0.257502	-0.850609	0.367564
4	13	-0.974811	-0.105985	-0.625045	0.236174
...	...	...	...	...	...
93	102	-1.559131	0.440215	-2.325538	0.295837
94	103	-1.496434	0.357878	-2.105013	0.098846
95	104	-1.467606	0.253125	-1.857816	0.058397
96	105	-1.420551	0.223617	-1.606946	0.202749
97	106	-1.246517	0.129141	-1.390368	0.402667

98 rows × 5 columns

```python
ano_data = setup(data, session_id=123, log_experiment=True, experiment_name='anomaly1')
```

	Description	Value
0	session_id	123
1	Original Data	(98, 5)
2	Missing Values	False
3	Numeric Features	5
4	Categorical Features	0
5	Ordinal Features	False
6	High Cardinality Features	False
7	High Cardinality Method	None
8	Transformed Data	(98, 4)
9	CPU Jobs	-1
10	Use GPU	False
11	Log Experiment	True
12	Experiment Name	anomaly1
13	USI	4357
14	Imputation Type	simple
15	Iterative Imputation Iteration	None
16	Numeric Imputer	mean
17	Iterative Imputation Numeric Model	None
18	Categorical Imputer	mode
19	Iterative Imputation Categorical Model	None
20	Unknown Categoricals Handling	least_frequent
21	Normalize	False
22	Normalize Method	None
23	Transformation	False
24	Transformation Method	None

Second, Apply four different unsupervised machine learning models over the data:

Models:

SVM-Model:

SVM-Model

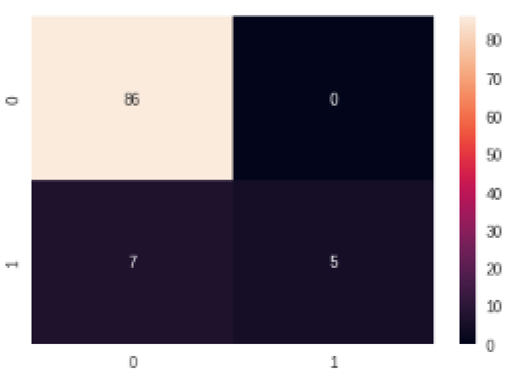
```
svm = create_model('svm')
predict_svm = predict_model(svm, data = X_test)
predict_svm
```

Classification report:

```
[64]  print(classification_report(y_label, svm_label))

              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
```

Confusion matrix:

```
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_label,svm_label)
f = sns.heatmap(cm, annot=True, fmt='d')
```

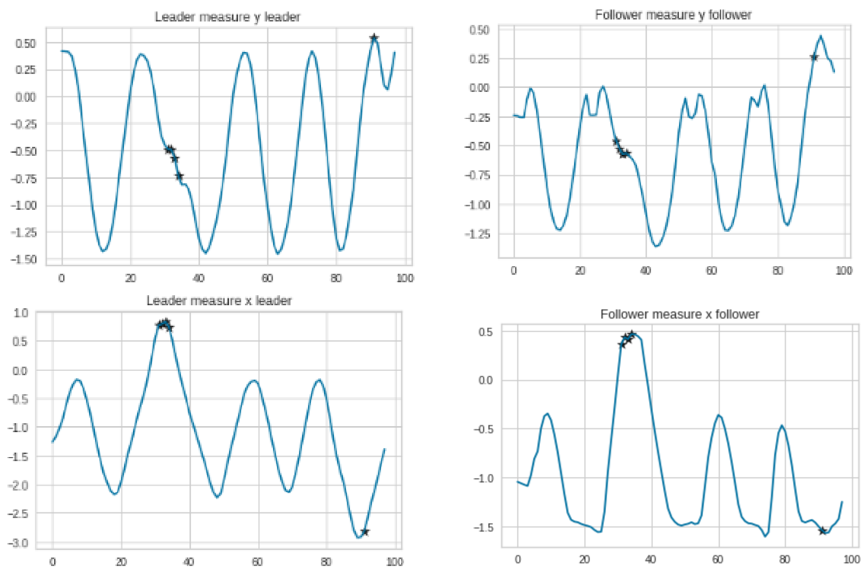



```
anomaly_svm = predict_svm[predict_svm['Anomaly'] != 0].drop('Anomaly',axis=1)
anomaly_svm.head()
```

	Follower_measure_x_follower	Follower_measure_y_follower	Leader_measure_x_leader	Leader_measure_y_leader	Anomaly_Score
31	0.360847	-0.459538	0.767382	-0.489506	8.172418
32	0.432356	-0.526272	0.795491	-0.493113	8.637892
33	0.412900	-0.574391	0.827058	-0.568237	8.695015
34	0.463492	-0.562305	0.734001	-0.737032	8.338360
91	-1.535425	0.260158	-2.816451	0.535399	7.785698

Plot SVM model:

```
for column in data.columns:
    plt.plot(predict_svm[column])
    plt.scatter(anomaly_svm.index,anomaly_svm[column],c='k',marker='*',s=90,alpha=1)
    plt.title(" ".join(column.split('_')))
    plt.show()
```



Plot 3D- tsne:

```
plot_model(knn, plot = 'tsne')
```

KNN Model:

```
knn = create_model('knn')
predict_knn = predict_model(knn, data = X_test)
predict_knn.head()
```

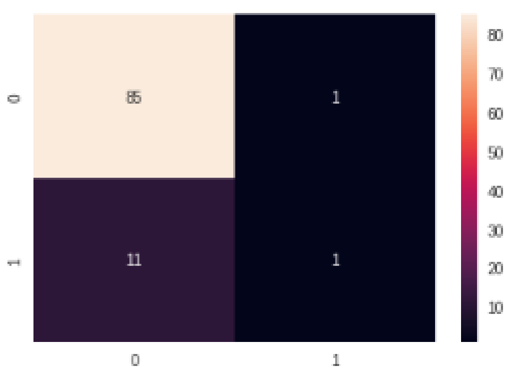
Classification report:

```
print(classification_report(y_label, knn_label))
```

```
              precision    recall  f1-score   support

         0.0       0.89      0.99      0.93        86
         1.0       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
```

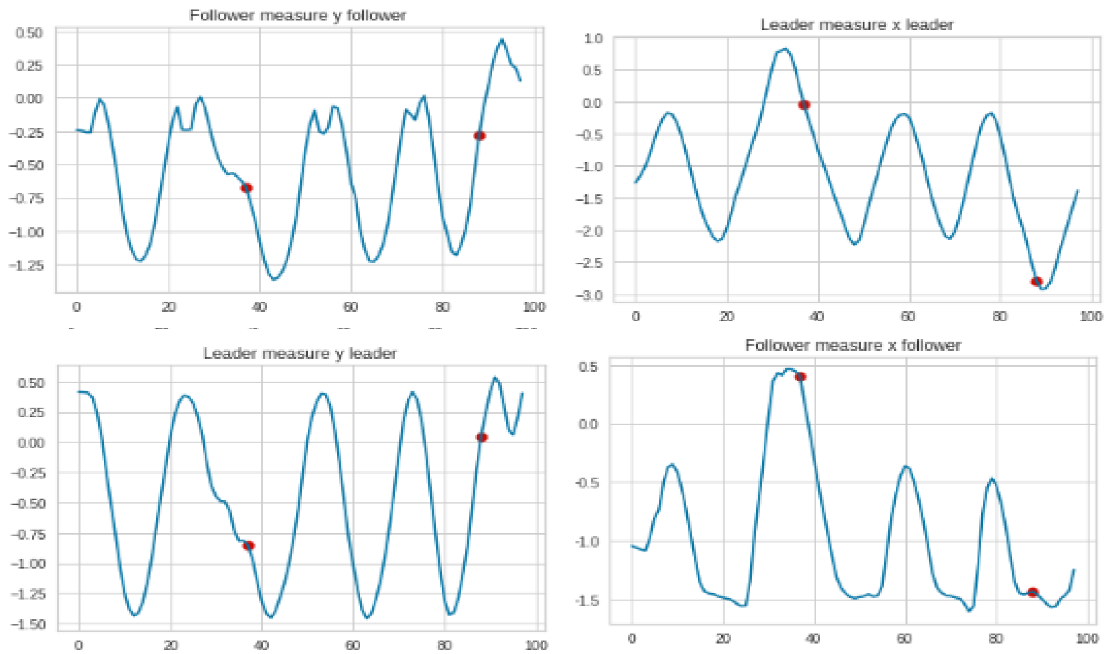
Confusion matrix:

```
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm_Knn = confusion_matrix(y_label,knn_label)
f = sns.heatmap(cm_Knn, annot=True, fmt='d')
```



Plot Knn model:

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



Plot 3d-TSNE:

```
1 plot_model(knn, plot = 'tsne')
```

PCA Model:

```
pca = create_model('pca')
predict_pca = predict_model(pca, data = X_test)
predict_pca.head()

```

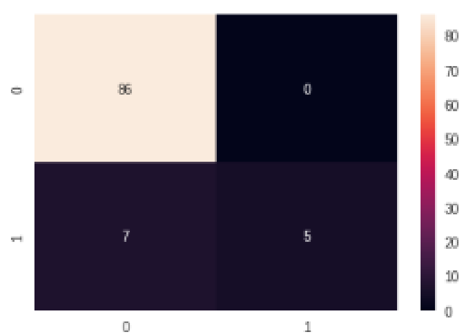
Classification report:

```
print(classification_report(y_label, pca_labels))
```

```
              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
```

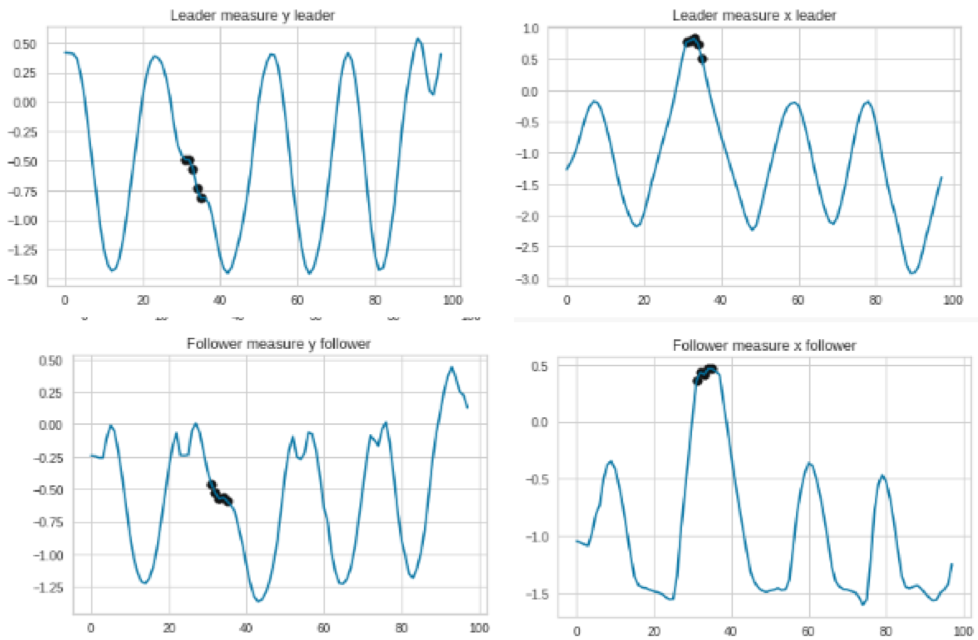
Confusion matrix:

```
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm_pca= confusion_matrix(y_label,pca_labels)
f = sns.heatmap(cm_pca, annot=True, fmt='d')

```



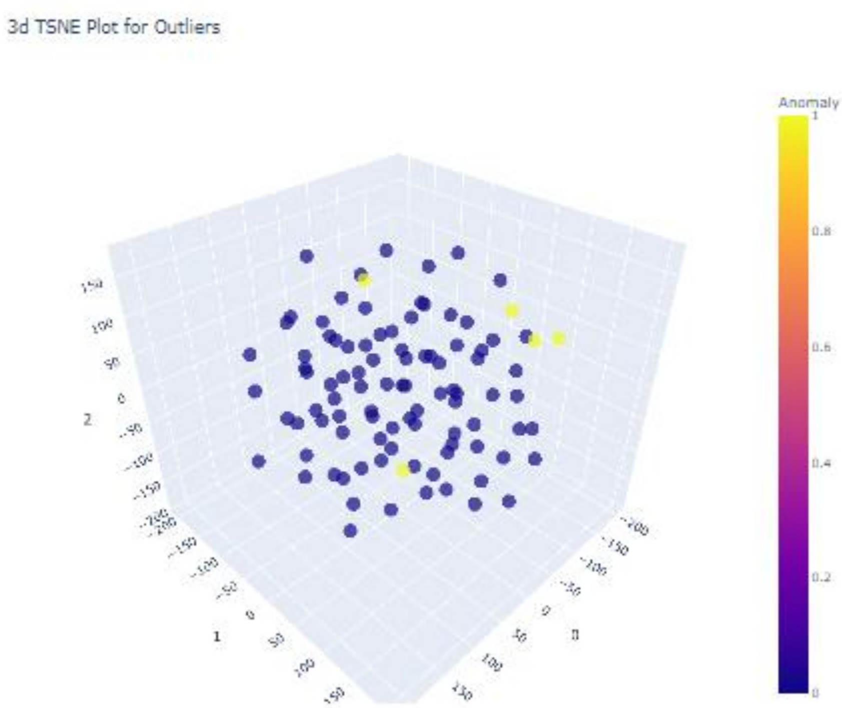
Plot PCA Model:

```
for column in data.columns:
    plt.plot(predict_pca[column])
    plt.scatter(anomaly_pca.index,anomaly_pca[column],c='k',marker='o',s=60,alpha=1)
    plt.title(" ".join(column.split('_')))
    plt.show()
```



-Plot 3d-TSNE:

```
1 plot_model(pca, plot = 'tsne')
```



DBSCAN Model:

```
clustering_DBSCAN = DBSCAN(eps=0.5, min_samples=7)
model=clustering_DBSCAN.fit(data)
label=model.labels_
```

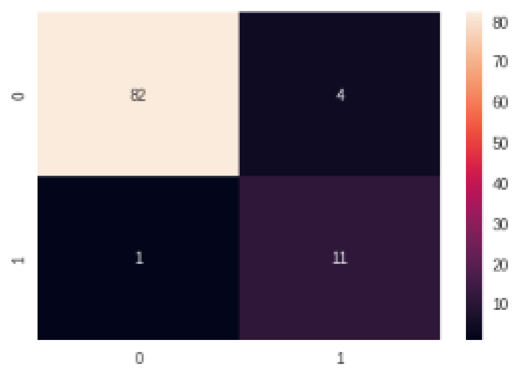
Classification report:

```
target_names = ['cluster 1', 'cluster 2']
print(classification_report(y_label,label,  target_names=target_names))
```

```
              precision    recall  f1-score   support

   cluster 1       0.99      0.95      0.97        86
   cluster 2       0.73      0.92      0.81        12

    accuracy                           0.95        98
   macro avg       0.86      0.94      0.89        98
weighted avg       0.96      0.95      0.95        98
```

Confusion matrix:

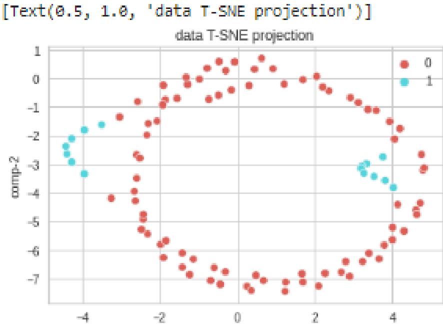
```
cm_DBSCAN = confusion_matrix(y_label,label)
f_DBSCAN = sns.heatmap(cm_DBSCAN, annot=True, fmt='d')
```



Plot 2d-TSNE:

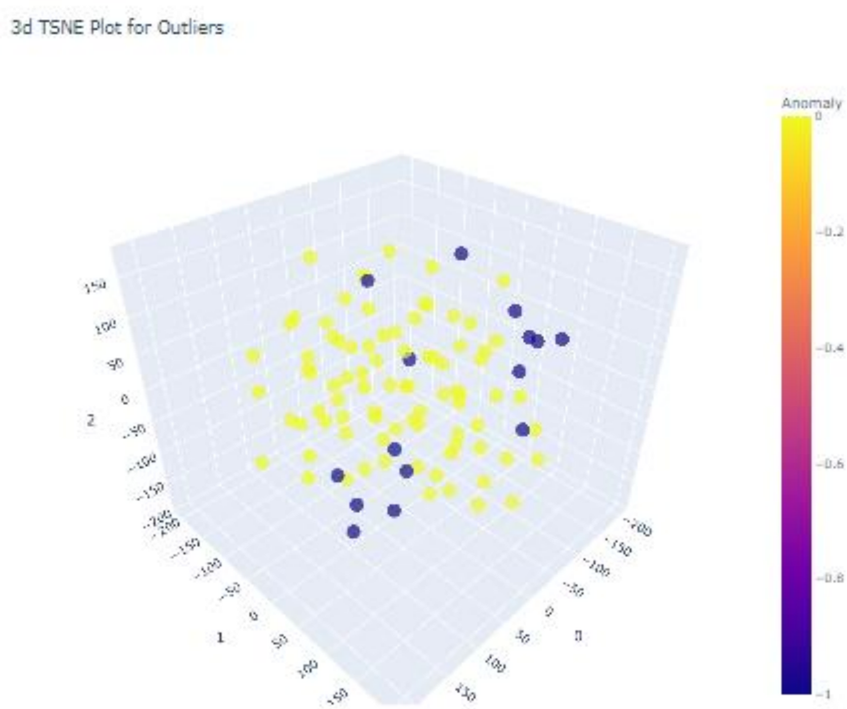
```
[94]
#plot the DBSCAN data  IN 2d T-SNE

tsne = TSNE(n_components=2, verbose=1, random_state=123)
z = tsne.fit_transform(data)
df = pd.DataFrame()
df["y"] = label
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", 2),
                data=df).set(title="data T-SNE projection")
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 98 samples in 0.000s...
[t-SNE] Computed neighbors for 98 samples in 0.002s...
[t-SNE] Computed conditional probabilities for sample 98 / 98
[t-SNE] Mean sigma: 0.817191
[t-SNE] KL divergence after 250 iterations with early exaggeration: 54.613007
[t-SNE] KL divergence after 850 iterations: 0.151545
[Text(0.5, 1.0, 'data T-SNE projection')]
```



Plot 3d-TSNE:

```
plot_model(clustering_DBSCAN, plot = 'tsne')
```



Performance Evaluation:

After implementing 4 different models (SVM, DBSCAN, KNN and PCA) to predict and detect anomalies there was a difference in accuracy it appears that DBSCAN algorithm shows the highest accuracy with 94.8% and the highest score in (precession, recall, score). After DBSCAN, SVM and PCA came in the second place with accuracy 92.8%. At the last KNN shows the lowest performance with accuracy 87.7%.

Conclusion:

We worked on the Dataset from a 2-minute experiment. 4 attributes from the dataset was extracted: “follower x data”, “follower y data”, “leader x data” and “leader y data” (‘x’ and ‘y’ refers to coordinate). We implement and compare the performance of different machine learning algorithms and to detect those anomalies under the two demo scenarios.

An implementation of 4 different unsupervised machine learning models was done:

1. SVM (binary)

2. KNN (unsupervised implementation)

3. PCA

4. DBSCAN

After implementation a comparison of 4 different unsupervised machine learning models was done and it shows that DBSCAN has the highest accuracy (94.8%) after it PCA and SVM came in the second place with accuracy 92.8. At the end KNN takes the third place with accuracy(87.7%)