Datasets

- [wave_benchmarks.zip (https://ir.library.oregonstate.edu/concern/parent/47429f155/file_sets/jh343z59f)](https://ir.library.oregonstate.edu/concern/parent/47429f155/file_sets/jh343z59f)

要求

使用[Python Outlier Detection (PyOD) (https://github.com/yzhao062/pyod)](https://github.com/yzhao062/pyod)或其他已知的工具包来完成分析工作

提交的内容

- 完整的分析代码
- 分析报告：展示分析的思路，详细过程，结果及你的分析
- 所选择的数据集在README中说明，数据文件不要上传到Github中

In [1]:

```
import pandas as pd
import os
import time
import warnings
import numpy as np

warnings.filterwarnings('ignore')

# timekeeping
timekeeping = time.time()
```

In [2]:

```
PAGEB_ROOT = 'wave/benchmarks'
benchmark_list = os.listdir(PAGEB_ROOT)
print(len(benchmark_list))
```

```
1080
```

数据来源说明

根据论文[1]可知，数据集中会引入4种不同的层次的不相关特征（i.e., noise）。

要创建新的不相关特征，首先从原始母集中随机选择一个特征。 然后，对于原始数据集中的每个数据点，通过从原始数据点的值进行统一采样（替换）来为此特征选择一个值。 结果是新添加的特征与某些原始特征具有相同的边缘分布，但是其值不包含有关数据点异常状态的信息。这保留了真实数据的特质，同时允许引入噪声。

为了简化确定需要多少不相关特征的过程，如果数据集已经具有d维特征，而我们想评估d'维，即将成对平均距离增加一个因子α所需的维数，那么

$$d' = \left(\alpha\sqrt{d}\right)^2 \quad (1),$$

其中$\alpha \in \{1.0, 1.2, 1.5, 2.0\}$.

[1] Emmott A, Das S, Dietterich T G, et al. A Meta-Analysis of the Anomaly Detection Problem[J]. arXiv: Artificial Intelligence, 2015.

随机选取一个csv文件，确定该数据集的原始特征有哪些?

In [3]:
```
df = pd.read_csv(os.path.join(PAGEB_ROOT, benchmark_list[0]))
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3167 entries, 0 to 3166
Data columns (total 90 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   point.id        3167 non-null   object 
 1   motherset       3167 non-null   object 
 2   origin          3167 non-null   object 
 3   original.label  3167 non-null   int64  
 4   diff.score      3167 non-null   float64
 5   ground.truth    3167 non-null   object 
 6   V               3167 non-null   float64
 7   V.1             3167 non-null   float64
 8   V.2             3167 non-null   float64
 9   V.3             3167 non-null   float64
 10  V.4             3167 non-null   float64
 11  V.5             3167 non-null   float64
 12  V.6             3167 non-null   float64
 13  V.7             3167 non-null   float64
 14  V.8             3167 non-null   float64
 15  V.9             3167 non-null   float64
 16  V.10            3167 non-null   float64
 17  V.11            3167 non-null   float64
 18  V.12            3167 non-null   float64
 19  V.13            3167 non-null   float64
 20  V.14            3167 non-null   float64
 21  V.15            3167 non-null   float64
 22  V.16            3167 non-null   float64
 23  V.17            3167 non-null   float64
 24  V.18            3167 non-null   float64
 25  V.19            3167 non-null   float64
 26  V.20            3167 non-null   float64
 27  noise..1        3167 non-null   float64
 28  noise..2        3167 non-null   float64
 29  noise..3        3167 non-null   float64
 30  noise..4        3167 non-null   float64
 31  noise..5        3167 non-null   float64
 32  noise..6        3167 non-null   float64
 33  noise..7        3167 non-null   float64
 34  noise..8        3167 non-null   float64
 35  noise..9        3167 non-null   float64
 36  noise..10       3167 non-null   float64
 37  noise..11       3167 non-null   float64
 38  noise..12       3167 non-null   float64
 39  noise..13       3167 non-null   float64
 40  noise..14       3167 non-null   float64
 41  noise..15       3167 non-null   float64
 42  noise..16       3167 non-null   float64
 43  noise..17       3167 non-null   float64
 44  noise..18       3167 non-null   float64
 45  noise..19       3167 non-null   float64
 46  noise..20       3167 non-null   float64
 47  noise..21       3167 non-null   float64
 48  noise..22       3167 non-null   float64
 49  noise..23       3167 non-null   float64
 50  noise..24       3167 non-null   float64
 51  noise..25       3167 non-null   float64
```

```
 52  noise..26       3167 non-null   float64
 53  noise..27       3167 non-null   float64
 54  noise..28       3167 non-null   float64
 55  noise..29       3167 non-null   float64
 56  noise..30       3167 non-null   float64
 57  noise..31       3167 non-null   float64
 58  noise..32       3167 non-null   float64
 59  noise..33       3167 non-null   float64
 60  noise..34       3167 non-null   float64
 61  noise..35       3167 non-null   float64
 62  noise..36       3167 non-null   float64
 63  noise..37       3167 non-null   float64
 64  noise..38       3167 non-null   float64
 65  noise..39       3167 non-null   float64
 66  noise..40       3167 non-null   float64
 67  noise..41       3167 non-null   float64
 68  noise..42       3167 non-null   float64
 69  noise..43       3167 non-null   float64
 70  noise..44       3167 non-null   float64
 71  noise..45       3167 non-null   float64
 72  noise..46       3167 non-null   float64
 73  noise..47       3167 non-null   float64
 74  noise..48       3167 non-null   float64
 75  noise..49       3167 non-null   float64
 76  noise..50       3167 non-null   float64
 77  noise..51       3167 non-null   float64
 78  noise..52       3167 non-null   float64
 79  noise..53       3167 non-null   float64
 80  noise..54       3167 non-null   float64
 81  noise..55       3167 non-null   float64
 82  noise..56       3167 non-null   float64
 83  noise..57       3167 non-null   float64
 84  noise..58       3167 non-null   float64
 85  noise..59       3167 non-null   float64
 86  noise..60       3167 non-null   float64
 87  noise..61       3167 non-null   float64
 88  noise..62       3167 non-null   float64
 89  noise..63       3167 non-null   float64
dtypes: float64(85), int64(1), object(4)
memory usage: 2.2+ MB
```

Out[3]:

	point.id	motherset	origin	original.label	diff.score	ground.truth	V	
0	wave_point_2031	wave	multiclass	2	0.000419	nominal	0.242400	-0.7
1	wave_point_2344	wave	multiclass	0	0.133717	anomaly	0.875982	-0.2
2	wave_point_0849	wave	multiclass	2	0.001321	nominal	-0.094190	0.2
3	wave_point_4662	wave	multiclass	0	0.248145	anomaly	0.658188	0.0
4	wave_point_1214	wave	multiclass	1	0.042073	nominal	0.064206	1.8

5 rows × 90 columns

根据以上的信息我们可以确定，pageb这个数据集的原始特征维度$d = 21$(`V`，`V.1` ~ `V.20`)。因此，由等式（1）可知，所有csv文件所包含的列数可能为$27 = \left(1.0 \times \sqrt{21}\right)^2 + 6, 36 = \left(1.2 \times \sqrt{21}\right)^2 + 6, 53 = \left(1.5 \times \sqrt{21}\right)^2 + 6, 90 = \left(2.0 \times \sqrt{21}\right)^2 + 6.$

下面我们遍历所有csv文件，验证一下。

In [4]:
```
d_set = set()
d_count = 0
for i in range(len(benchmark_list)):
    df = pd.read_csv(os.path.join(PAGEB_ROOT, benchmark_list[i]))
    d_set.add(len(df.columns))
    d_count += len(df)
print('Possible columns of all csv files:', d_set)
print('Total amount:', d_count)
```

```
Possible columns of all csv files: {90, 27, 36, 53}
Total amount: 2632953
```

数据特征选择

为了充分利用所提供的数据集完成离群点分析与异常检测，将提取所有csv文件共同的特征（即原始特征, `V`，`V.1` ~ `V.20` ）作为算法或模型的输入，用于检测该条数据是否属于异常点。

In [5]:
```
ORIGIN_FEATURES = ['V', 'V.1', 'V.2', 'V.3', 'V.4', 'V.5', 'V.6', 'V.7', 'V.8', 'V.9','V.10',
                   'V.11', 'V.12', 'V.13', 'V.14', 'V.15', 'V.16', 'V.17', 'V.18', 'V.19', 'V.20',
                   'ground.truth']
def feature_section(benchmark_list):
    concat_data = pd.DataFrame()
    for i in benchmark_list:
        df = pd.read_csv(os.path.join(PAGEB_ROOT, i))
        concat_data = concat_data.append(df[ORIGIN_FEATURES])
    return concat_data
```

In [6]:
```
concat_data = feature_section(benchmark_list=benchmark_list)
concat_data.info()
concat_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2632953 entries, 0 to 3009
Data columns (total 22 columns):
 #   Column        Dtype
---  ------        -----
 0   V             float64
 1   V.1           float64
 2   V.2           float64
 3   V.3           float64
 4   V.4           float64
 5   V.5           float64
 6   V.6           float64
 7   V.7           float64
 8   V.8           float64
 9   V.9           float64
 10  V.10          float64
 11  V.11          float64
 12  V.12          float64
 13  V.13          float64
 14  V.14          float64
 15  V.15          float64
 16  V.16          float64
 17  V.17          float64
 18  V.18          float64
 19  V.19          float64
 20  V.20          float64
 21  ground.truth  object
dtypes: float64(21), object(1)
memory usage: 462.0+ MB
```

Out[6]:

	V	V.1	V.2	V.3	V.4	V.5	V.6	V.7	
0	0.242400	-0.739089	0.460923	0.345094	-1.376929	-0.483581	-0.566436	-0.715453	1.549
1	0.875982	-0.255060	-1.525660	-0.891447	0.011388	0.227481	-0.447375	-0.961720	-1.089
2	-0.094190	0.247950	0.090543	-0.863183	-0.423578	-1.172594	-0.348157	-0.566546	-0.764
3	0.658188	0.086607	-0.624964	-0.304973	0.762151	-0.825332	-1.494119	-0.629545	-1.275
4	0.064206	1.851888	0.317821	1.793612	0.791943	1.335416	1.194675	1.374960	-0.079

5 rows × 22 columns

数据集划分

train set : test set = 8 : 2

In [7]:
```
from sklearn.model_selection import train_test_split

train, test = train_test_split(concat_data, test_size=0.2, random_sta
te=2020)

def data_label_split(data, label_column='ground.truth'):
    x = data.drop(label_column, axis=1)
    y = []
    for i in data[label_column].values:
        if i == 'nominal':
            y.append(0)
        else:
            y.append(1)
    y = np.array(y)
    return x, y

X_train, y_train = data_label_split(train)
X_test, y_test = data_label_split(test)
```

In [8]:
```
from sklearn.utils.multiclass import type_of_target
type_of_target(y_train)
```

Out[8]:
```
'binary'
```

t-SNE降维，用于可视化

In [9]:
```
from sklearn.manifold import TSNE
# T-SNE Implementation
t0 = time.time()
X_train_reduced_tsne = TSNE(n_components=2, random_state=2020, init=
'pca', n_iter=2000).fit_transform(X_train.values)
X_test_reduced_tsne = TSNE(n_components=2, random_state=2020, init='p
ca', n_iter=2000).fit_transform(X_test.values)
t1 = time.time()
print("T-SNE took {:.2} s".format(t1 - t0))
```

```
T-SNE took 1.4e+04 s
```

模型比较

单一模型

- KNN
- PCA
- LOF

组合模型

- **Average**: average scores of all detectors
- **Maximization**: maximum score across all detectors.
- **Average of Maximum (AOM)**
- **Maximum of Average (MOA)**

ref: [https://github.com/yzhao062/pyod/tree/master/examples (https://github.com/yzhao062/pyod/tree/master/examples)](https://github.com/yzhao062/pyod/tree/master/examples)

kNN

初始化一个 `pyod.models.knn.KNN` 检测器, 模型拟合, 然后给出预测。

In [10]:
```
# train the KNN detector
from pyod.models.knn import KNN

clf_name = 'KNN'
clf = KNN()
clf.fit(X_train)

# get the prediction labels and outlier scores of the training data
y_train_pred = clf.labels_  # binary labels (0: inliers, 1: outliers)
y_train_scores = clf.decision_scores_  # raw outlier scores

# get the prediction on the test data
y_test_pred = clf.predict(X_test)  # outlier labels (0 or 1)
y_test_scores = clf.decision_function(X_test)  # outlier scores
```

利用 `ROC` 和 `Precision @ Rank` 评估预测。

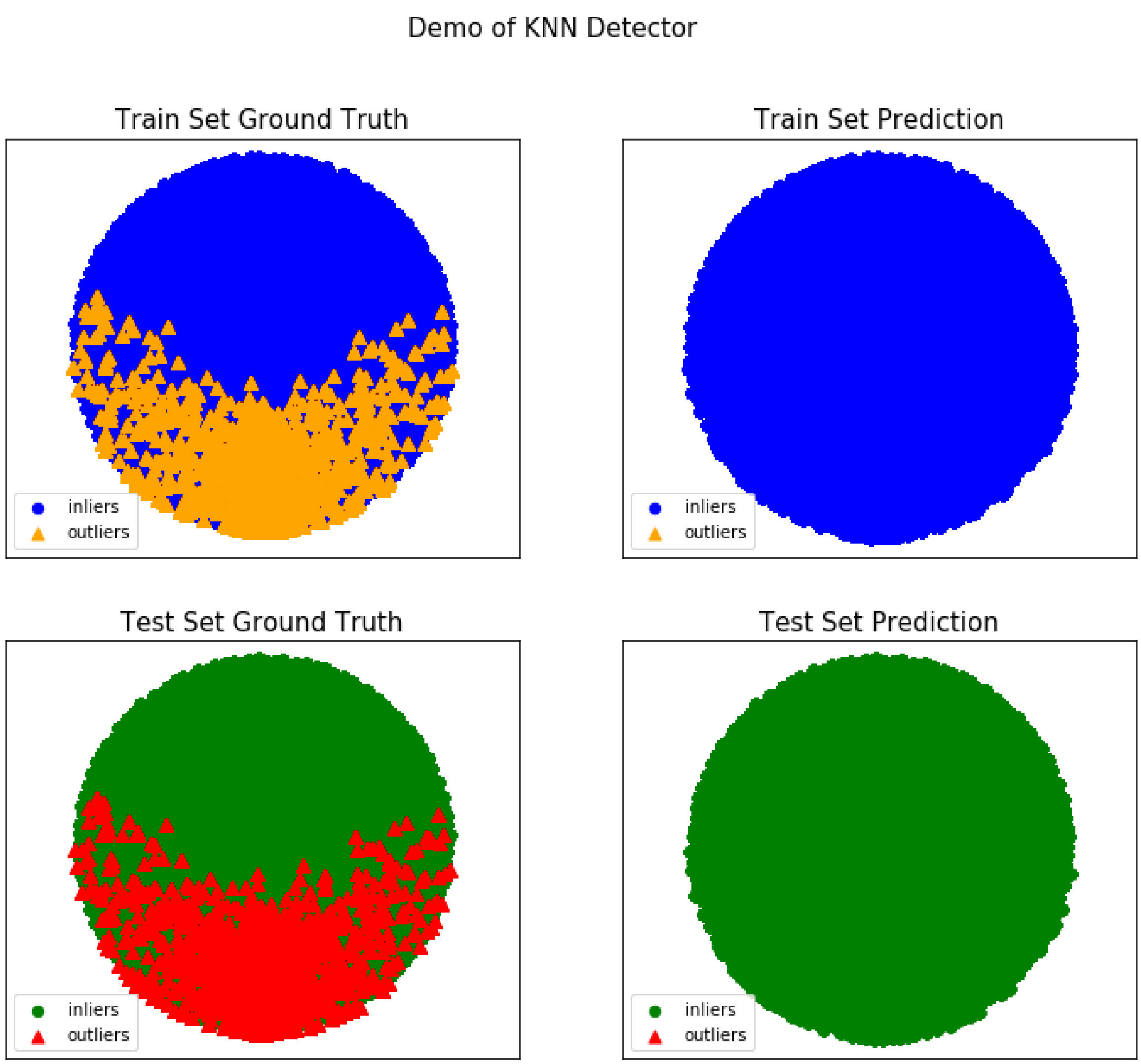
In [11]:
```python
from pyod.utils.data import evaluate_print
# evaluate and print the results
print("\nOn Training Data:")
evaluate_print(clf_name, y_train, y_train_scores)
print("\nOn Test Data:")
evaluate_print(clf_name, y_test, y_test_scores)
```

```
On Training Data:
KNN ROC:0.5, precision @ rank n:0.0

On Test Data:
KNN ROC:0.5, precision @ rank n:0.0
```

可视化 KNN 的结果

In [12]:
```python
from pyod.utils.example import visualize
visualize(clf_name, X_train_reduced_tsne, y_train, X_test_reduced_tsn
e, y_test, y_train_pred,
          y_test_pred, show_figure=True, save_figure=False)
```



PCA

初始化一个 `pyod.models.pca.PCA` 检测器, 模型拟合, 然后给出预测。

In [13]:
```
# train PCA detector
from pyod.models.pca import PCA

clf_name = 'PCA'
clf = PCA(n_components=3)
clf.fit(X_train)

# get the prediction labels and outlier scores of the training data
y_train_pred = clf.labels_  # binary labels (0: inliers, 1: outliers)
y_train_scores = clf.decision_scores_  # raw outlier scores

# get the prediction on the test data
y_test_pred = clf.predict(X_test)  # outlier labels (0 or 1)
y_test_scores = clf.decision_function(X_test)  # outlier scores
```

利用 ROC 和 Precision @ Rank 评估预测

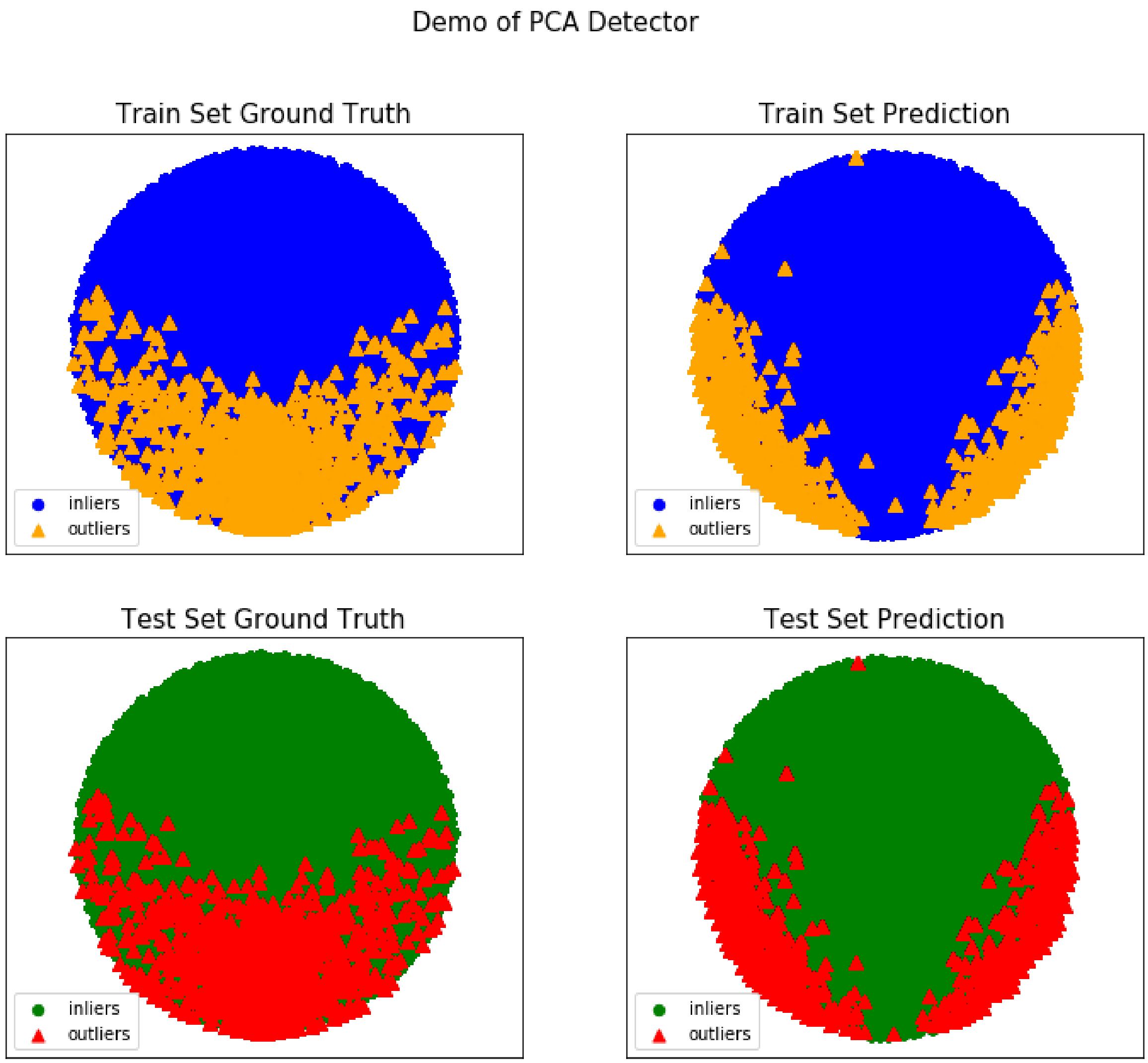
In [14]:
```
# evaluate and print the results
print("\nOn Training Data:")
evaluate_print(clf_name, y_train, y_train_scores)
print("\nOn Test Data:")
evaluate_print(clf_name, y_test, y_test_scores)
```

```
On Training Data:
PCA ROC:0.6463, precision @ rank n:0.1699

On Test Data:
PCA ROC:0.6472, precision @ rank n:0.1726
```

可视化 PCA 的结果

```
In [15]: visualize(clf_name, X_train_reduced_tsne, y_train, X_test_reduced_tsn
e, y_test, y_train_pred,
          y_test_pred, show_figure=True, save_figure=False)
```



LOF

初始化一个 `pyod.models.lof.LOF` 检测器, 模型拟合, 然后给出预测。

```
In [16]: # train LOF detector
from pyod.models.lof import LOF
clf_name = 'LOF'
clf = LOF()
clf.fit(X_train)

# get the prediction labels and outlier scores of the training data
y_train_pred = clf.labels_  # binary labels (0: inliers, 1: outliers)
y_train_scores = clf.decision_scores_  # raw outlier scores

# get the prediction on the test data
y_test_pred = clf.predict(X_test)  # outlier labels (0 or 1)
y_test_scores = clf.decision_function(X_test)  # outlier scores
```

利用 ROC 和 Precision @ Rank 评估预测

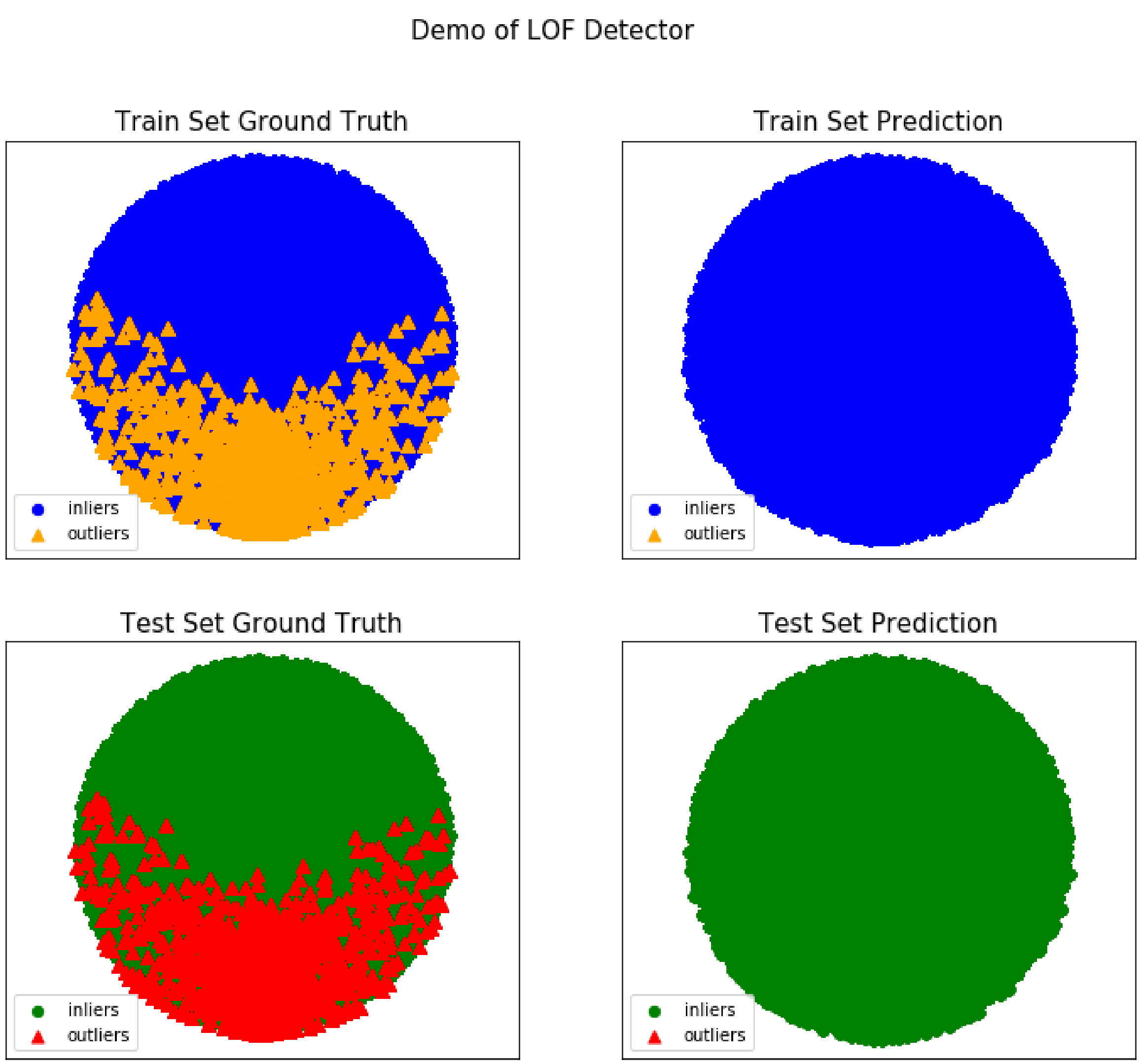
In [17]:
```
# evaluate and print the results
print("\nOn Training Data:")
evaluate_print(clf_name, y_train, y_train_scores)
print("\nOn Test Data:")
evaluate_print(clf_name, y_test, y_test_scores)
```

```
On Training Data:
LOF ROC:0.5, precision @ rank n:0.0

On Test Data:
LOF ROC:0.5, precision @ rank n:0.0
```

可视化 LOF 的结果

In [18]:
```
# visualize the results
visualize(clf_name, X_train_reduced_tsne, y_train, X_test_reduced_tsn
e, y_test, y_train_pred,
          y_test_pred, show_figure=True, save_figure=False)
```



Model Combination

用不同的k(10 ~ 200)初始化20个 kNN 离群点检测器，然后得到所有的离群点的分数。

In [19]:
```
from pyod.models.knn import KNN  # kNN detector
from pyod.models.combination import aom, moa, average, maximization
from pyod.utils.utility import standardizer
```

In [20]:
```
# standardizing data for processing
X_train_norm, X_test_norm = standardizer(X_train, X_test)

n_clf = 20  # number of base detectors

# initialize 20 base detectors for combination
k_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140
, 150, 160, 170, 180, 190, 200]

train_scores = np.zeros([X_train.shape[0], n_clf])
test_scores = np.zeros([X_test.shape[0], n_clf])

print('Combining {n_clf} kNN detectors'.format(n_clf=n_clf))

for i in range(n_clf):
    k = k_list[i]

    clf = KNN(n_neighbors=k, method='largest')
    clf.fit(X_train_norm)

    train_scores[:, i] = clf.decision_scores_
    test_scores[:, i] = clf.decision_function(X_test_norm)
```

```
Combining 20 kNN detectors
```

In [21]:
```
# Decision scores have to be normalized before combination
train_scores_norm, test_scores_norm = standardizer(train_scores, test
_scores)

# Combination by average
y_by_average = average(test_scores_norm)
# Combination by max
y_by_maximization = maximization(test_scores_norm)
# Combination by aom
y_by_aom = aom(test_scores_norm, n_buckets=5)
# Combination by moa
y_by_moa = moa(test_scores_norm, n_buckets=5)
```

In [22]:

```
print("\nOn Test Data:")
evaluate_print('Combination by Average', y_test, y_by_average)
evaluate_print('Combination by Maximization', y_test, y_by_maximizati
on)
evaluate_print('Combination by AOM', y_test, y_by_aom)
evaluate_print('Combination by MOA', y_test, y_by_moa)
```

```
On Test Data:
Combination by Average ROC:0.8717, precision @ rank n:1.0
Combination by Maximization ROC:0.8717, precision @ rank n:1.0
Combination by AOM ROC:0.8717, precision @ rank n:1.0
Combination by MOA ROC:0.8717, precision @ rank n:1.0
```

In [23]:

```
m, s = divmod(time.time()-timekeeping, 60)
h, m = divmod(m, 60)
print ('run time: %02d:%02d:%02d' % (h, m, s))
```

```
run time: 06:26:56
```