주제 : Anomaly Detection

(논문 : Isolation-based Anomaly Detection)

1. Isolation Forest?

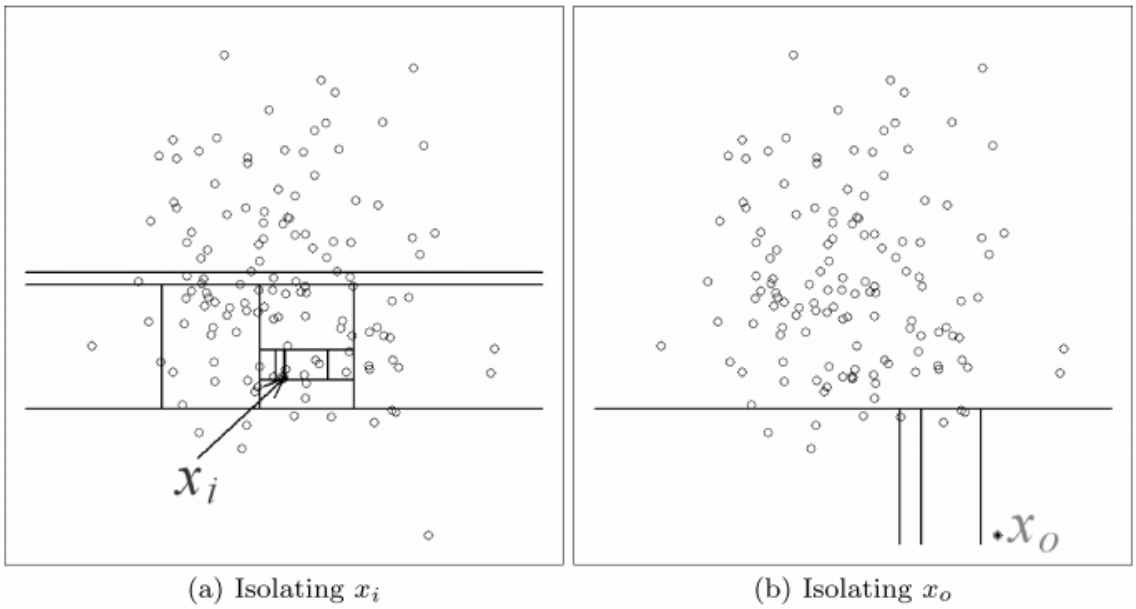
Unsupervised Learning
Regression Decision Tree 기반
Anomaly(=Novelty) Detection 알고리즘

① Anomaly Detection?

대다수의 정상 데이터들과 다른 양상을 보이는 소수의 비정상 데이터를 판별하는 것을 목적으로 하는 머신러닝

② 이진분할 기반 이상탐지

대다수의 정상 데이터들은 더 많은 재귀 이진분할이 수행되고, 소수의 비정상 데이터는 정상 데이터에 비해 상대적으로 적은 이진 분할을 수행함.
→ 해당 개념에서 착안하여 Tree로부터 Anomaly를 판단
→ Tree의 Depth가 깊을수록 정상 데이터 가능성 높음
→ Tree의 Depth가 얕을수록 비정상 데이터 가능성 높음



(a) Isolating x_i (b) Isolating x_o

위의 그림은 논문의 그림을 가져온 것이다.
(a)가 (b)보다 더 많은 이진 분할이 수행되었고, 이는 정상 데이터의 가능성이 높다는 의미이다. 또한 이는 Tree의 Root Node와 멀다는 것을 의미한다.
반대로 오른쪽 그림 (b)는 상대적으로 적은 이진 분할을 수행하였으며, 이는 비정상 데이터의 가능성이 높다. 또한 이는 Tree의 Root Node와 가깝다는 것을 의미한다.

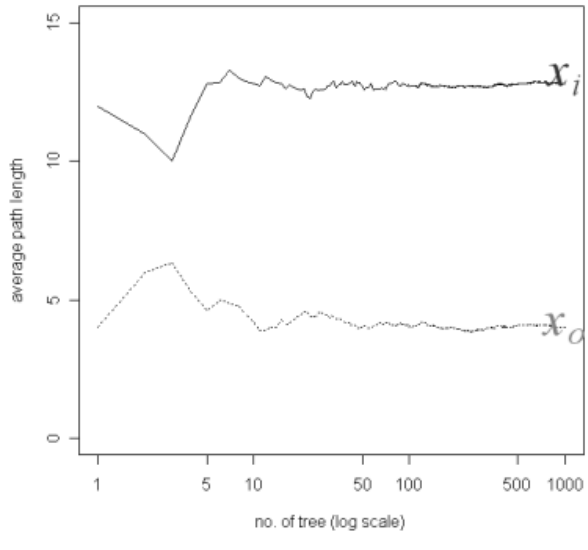
③ Isolation Forest Score

Score는 0과 1 사이 값으로 표현
Score가 1에 가까우면 비정상, 0.5이하면 정상 데이터로 판

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

$$\mathrm{C(n)} = \ 2\mathrm{H(n-1)} - (2(\mathrm{n}-1)/\mathrm{n})$$

- n : the number of data points
- h(x) : x의 Path Length
- c(n) : h(x)를 normalize 하기 위해 사용된 상수



(c) Average path lengths converge

위 그림은 논문에서 가져온 것이다. x_0는 비정상 데이터에 대한 평균 path length를 나타내고, x_i는 정상 데이터의 대한 평균 path length를 도식화한 것이다. 앞서 설명한 것처럼 비정상 데이터에 대한 Tree의 path length는 얕고, 정상 데이터에 대한 path length는 깊음을 확인 할 수 있다.

④ Isolation Forest 특징

- Swamping : 비정상 데이터가 정상 데이터와 가까이 위치한 경우 오분류 되는 현상을 말함.
- Masking : 비정상 데이터도 군집화 되어 있어서 정상 데이터로 오분류 되는 현상을 말함.
- Sub Sampling : Swamping, Masking 현상을 해결하기 위해 전체 데이터를 sampling한 데이터로 모델 훈련 수행(bootstrapping)

  ※ sub-sampling 개수가 늘어남에 따라 average path length가 수렴하기 때문에 앙상블에 의한 모델의 강건함 생김.

⑤ Isolation Forest 알고리즘

Algorithm 2 : $iTree(X')$

Inputs: X' - input data
Output: an iTree

1: **if** X' cannot be divided **then**
2: return $exNode\{Size \leftarrow |X'|\}$
3: **else**
4: let Q be a list of attributes in X'
5: randomly select an attribute $q \in Q$
6: randomly select a split point p between the max and min values of attribute q in X'
7: $X_l \leftarrow filter(X', q < p)$
8: $X_r \leftarrow filter(X', q \geq p)$
9: return $inNode\{Left \leftarrow iTree(X_l),$
10: $Right \leftarrow iTree(X_r),$
11: $SplitAtt \leftarrow q,$
12: $SplitValue \leftarrow p\}$
13: **end if**

위의 코드는 논문에서 발췌한 것인다.
1번째 줄을 보면 " X' cannot be divided " 이라는 부분을 확인할 수 있을 것이다. 해당 방식은 모든 데이터가 나누어지지 않을 때까지 split을 수행한다는 것이다. 그 이유는 ③에서도 확인하였 듯이 평균 path length를 통해 정상/비정상을 판단하기 위함이다.

line1. sampling 된 X'이 Isolaiton 된다면 exNode로 Return
line4. X'의 변수들을 Q에 list로 저장
line5. 리스트 Q의 원소 중 하나 q를 랜덤하게 선택
line6. split에 사용될 criterion 값 p를 랜덤 하게 선택.
line7. q<p를 만족하는 x들을 X_l로 할당
line8. q≤p를 만족하는 x들을 X_r로 할당
line9. 모든 데이터 포인트가 isolation 될 때까지 iTree 반복.
split 될 때마다 그 정보 (q,p) 저장.

2. Code 구현 및 실험

```
import numpy as np
import pandas as pd
import scipy.io
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.preprocessing import MinMaxScaler
from os.path import dirname, join as pjoin
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
```

우선 IsolationForest는 sklearn.ensemble 라이브러리에서 지원하는 모듈이다. 따라서 해당 모듈을 선언하고, 추가적으로 실험을 위한 값들을 선언하였다.

```
data_dir = pjoin(dirname(scipy.io.__file__), 'matlab', 'tests', 'data')
pima_path = pjoin(data_dir, 'pima.mat')
shuttle_path = pjoin(data_dir, 'shuttle.mat')
pima = scipy.io.loadmat(pima_path)
shuttle = scipy.io.loadmat(shuttle_path)

x_pima = pd.DataFrame(pima['X'])
y_pima = pd.Series([x[0] for x in pima['y']])
x_shuttle = pd.DataFrame(shuttle['X'])
y_shuttle = pd.Series([x[0] for x in shuttle['y']])

Counter(y_pima)
Counter(y_shuttle)
```

```
In [484]: Counter(y_pima)
Out[484]: Counter({1: 268, 0: 500})

In [485]: Counter(y_shuttle)
Out[485]: Counter({1: 3511, 0: 45586})
```

논문에서 사용하 터를 가져오는 부분이다. Pima data set은 정상(500개)/비정상(268개)로 구성되어 있으며, shuttle data set은 정상(45586개)/비정상(3511개)로 구성되어 있음을 확인할 수 있다.

```
anomalies_ratio = Counter(y_shuttle)[1]/Counter(y_shuttle)[0]
anomalies_ratio = Counter(y_pima)[1]/Counter(y_pima)[0]

if_sk = IsolationForest(n_estimators = 50,
                        max_samples = 50,
                        contamination = anomalies_ratio,
                        random_state = np.random.RandomState(100))
```

해당 부분이 main 알고리즘 code에 해당 된다. If_sk는 sklearn에서 제공하는 Isolation Forest를 의미한다. Pima, Shuttle Data Set의 anomalies ratio에 대해 구해주고, max_samples=50, n_estimators=50으로 설정하고 실험을 진행하였다.

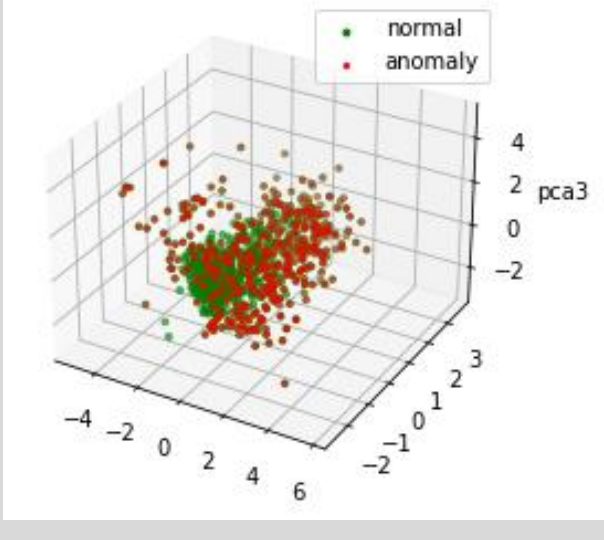
```
#%% PIMA
if_sk.fit(x_pima)
y_pima_pred = if_sk.predict(x_pima)
y_pima_pred = [1 if x == -1 else 0 for x in y_pima_pred]
Counter(y_pima_pred)
y_pima_pred = pd.DataFrame(y_pima_pred, columns=['y'])
confusion_matrix(y_pima, y_pima_pred)
f1_score(y_pima, y_pima_pred, zero_division=1)
plt.hist(if_sk.score_samples(x_pima))
Counter(plt.hist(if_sk.score_samples(x_pima)))
plt.hist(x_pima[3])
```

```
#%% Shuttle
if_sk.fit(x_shuttle)
y_shuttle_pred = if_sk.predict(x_shuttle)
y_shuttle_pred = [1 if x == -1 else 0 for x in y_shuttle_pred]
Counter(y_shuttle_pred)
y_shuttle_pred = pd.DataFrame(y_shuttle_pred, columns=['y'])
confusion_matrix(y_shuttle, y_shuttle_pred)
f1_score(y_shuttle, y_shuttle_pred, zero_division=1)
```

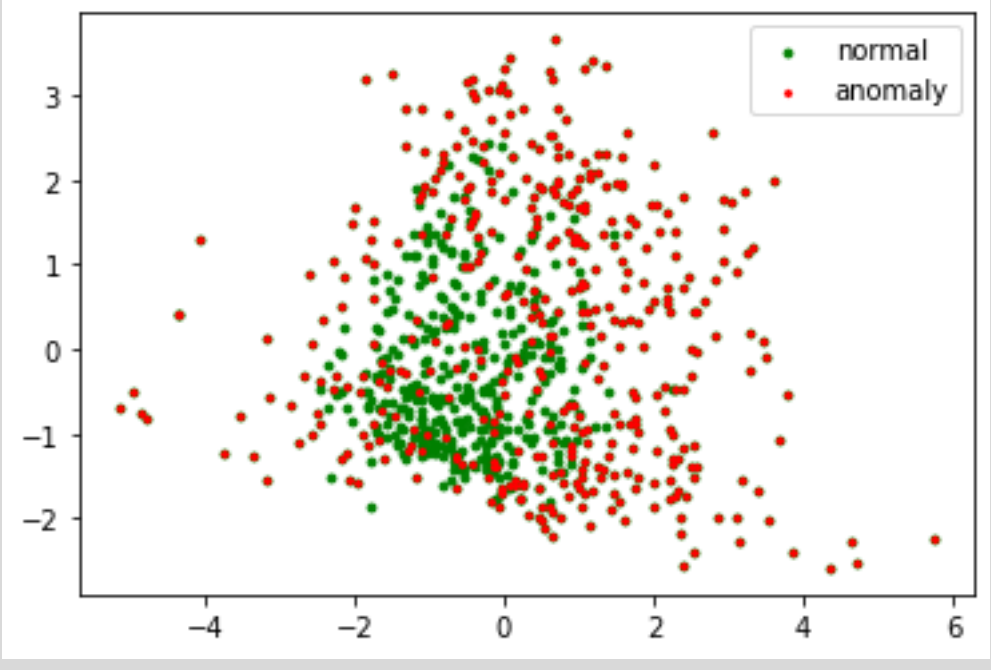
If_sk.fit 을 통해 학습을 시킬수 잇으며, predic을 통해 구분된 값에 대해 예측을 수행할 수 있다. 비정상 데이터에 대해서는 -1로 확보되어 해당 값을 0으로 변경해주었다.

평가 지표는 f1 Score를 대표로 비교해볼 것이다.

```
anormaly = y_pima_pred[y_pima_pred['y']==1]
anormaly_index = list(anormaly.index)

import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

pca = PCA(n_components=3)
scaler = StandardScaler()
X = scaler.fit_transform(x_pima)
X_reduce = pca.fit_transform(X)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("pca3")
ax.scatter(X_reduce[:, 0], X_reduce[:, 1], zs=X_reduce[:, 2], s=8, lw=1, label="normal",c="green")
ax.scatter(X_reduce[anormaly_index,0],X_reduce[anormaly_index,1], X_reduce[anormaly_index,2],
           lw=2, s=4, marker="x", c="red", label="anomaly")

ax.legend()
plt.show()
```



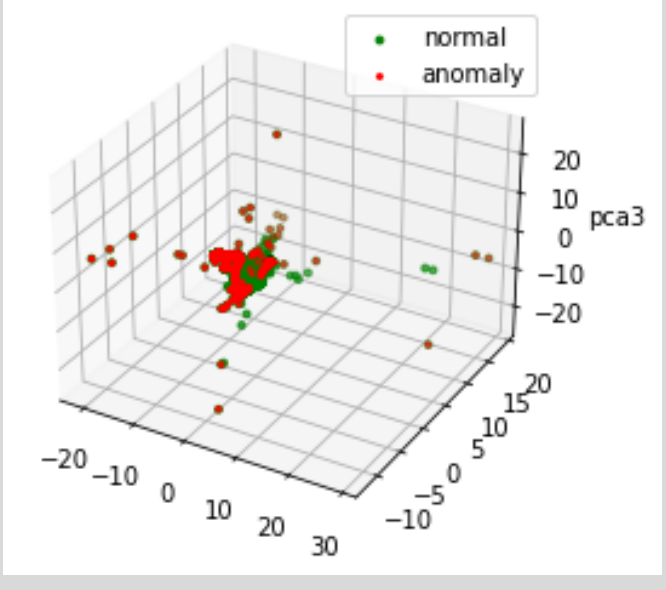
위의 code는 pima data set이 8개의 변수로 구성된 data set이라서 pca를 통해 3차원으로 표현해본 것이다. 초록색은 정상 데이터를 나타낸 것이고, 빨간색은 비정상 데이터를 pca를 통한 3차원 축에 표현한 것이다. 구분이 어느정도 잘 되는지 보일 것이라고 생각했지만, 3차원으로 식별이 어려워서 2차원으로도 표현해 보았다.



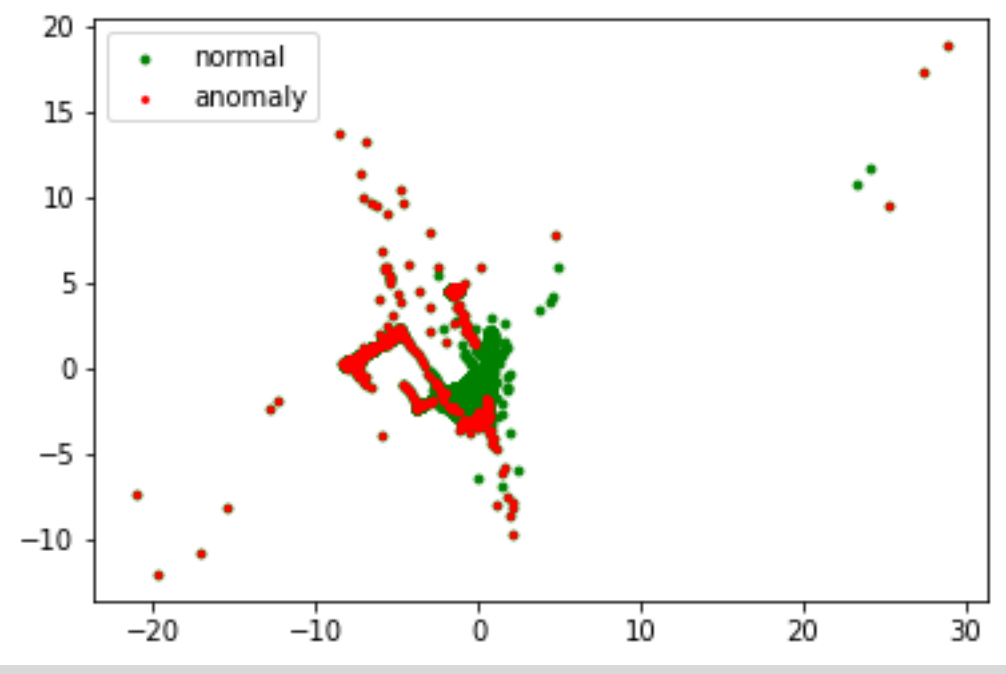
하지만 명백한 차이가 보이지 않았다.

그래서 앞서 확인한 f1 Score를 확인해보았는데 , 0.55로 낮게 확인되고 있었다. 논문에서도 해당 data set은 AUC 가 전체중에 가장 낮은 0.67로 확인되고 있었다. 데이터 자체가 Isolation Forest로 잘 구분되지 않는 특성을 가지고 있는 것으로 추정된다.

이번에는 논문상의 AUC가 1에 가까웠던 data set인 shuttle으로 실험을 진행하였다. F1 Score가 0.93으로 높게 확인되었다.



Pima data set보다는 잘 구분되는 것으로 보인다. 하지만 3차원에서는 역시 구분이 잘 안되어서 2차원으로 다시 구해보았다.



이번에는 pima data set보다는 명백히 구분되고 있음을 알 수 있었다. 초록색 normal 부분에 대해 Isolation Forest에 의해 이분 분할이 많이 이루어지고 빨간색 부분의 개수는 상대적으로 작은 것으로 확인된다.

3. 결론

AUC	Pima	shuttle
논문	0.67	1
실험	0.62	1

Unsupervised Learning 기반의 Anomaly Detection 알고리즘인 Isolation Forest에 대해 Sklearn 라이브러리를 통해 코드 구현과 논문상의 데이터 셋을 사용하여 실험하여 AUC 결과에 대해 확인 해 보았다.

Isolation Forest는 공간상에 분할을 통해 비정상에 대한 path length의 차이를 통해 정상과 비정상을 구분하는 알고리즘이라는 것을 확인했다.

해당 실험에서는 확인해보지는 못하였지만, 다른 기법의 알고리즘은 변수간의 거리나 밀도를 모두 계산해야 하지만 해당 알고리즘인 Isolation Forest는 Tree기반의 구분만을 수행하다 보니 모델 훈련 시간이 적게 걸린다는 장점도 가지고 있다.