Jay Goyal

C017

Btech EXTC

Isolation forest for anomaly detection

```
#Part A: Implementing on a randomly generated data
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
X= 0.3*np.random.randn(100,2)
X.shape
```

```
(100, 2)
```

```
X_train_normal = np.r_[X+2, X-2]     # r_ concatinates the samples alomg the firs
```

```
print(X.shape,X_train_normal.shape)
```

```
(100, 2) (200, 2)
```

```
#generate the data sample for testing
X= 0.3*np.random.randn(20,2)
X_train_normal = np.r_[X+2, X-2]
print(X.shape,X_train_normal.shape)
```

```
(20, 2) (40, 2)
```

```
#Generate data for testing
X= 0.3*np.random.randn(20,2)
X_test_normal=np.r_[X+2,X-2]
print(X.shape,X_test_normal.shape)
```

```
(20, 2) (40, 2)
```

```
#generating outliers using uniform distribution
```

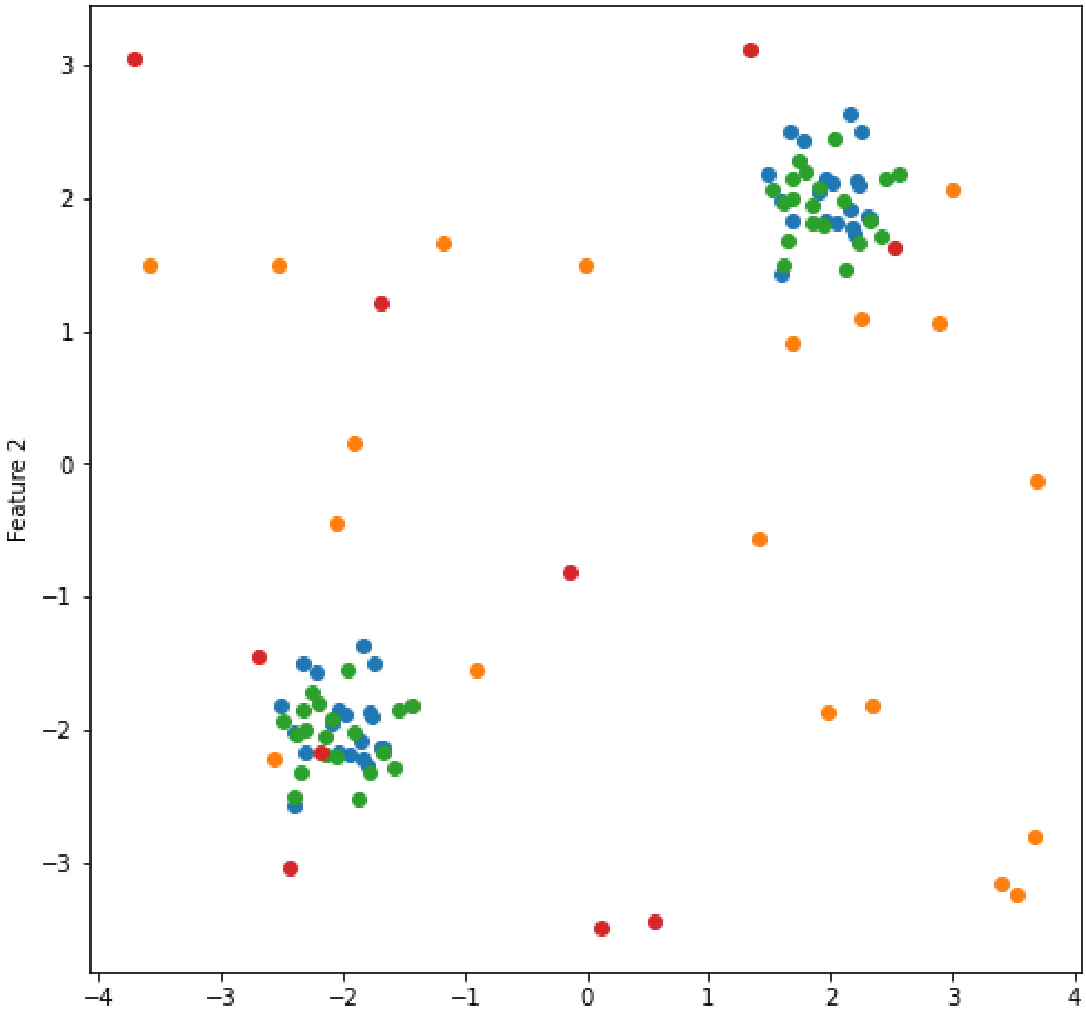
```
X_train_outliers = np.random.uniform(low=-4, high=4, size=(20,2))
X_test_outliers = np.random.uniform(low=-4, high=4, size=(10,2))
```

```
print(X_train_outliers.shape,X_test_outliers.shape)
```

```
(20, 2) (10, 2)
```

```
#plotting the datpoints
plt.figure(figsize=(8,8))
plt.scatter(X_train_normal[:,0],X_train_normal[:,1], label='X_train_normal')
plt.scatter(X_train_outliers[:,0],X_train_outliers[:,1], label='X_train_outliers
plt.scatter(X_test_normal[:,0],X_test_normal[:,1], label='X_test_normal')
plt.scatter(X_test_outliers[:,0],X_test_outliers[:,1], label='X_test_outliers')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
```

```
Text(0, 0.5, 'Feature 2')
```




```
X_train = np.append(X_train_normal,X_train_outliers,axis=0)
X_test = np.append(X_test_normal,X_test_outliers,axis=0)
```

```
from sklearn.ensemble import IsolationForest
```

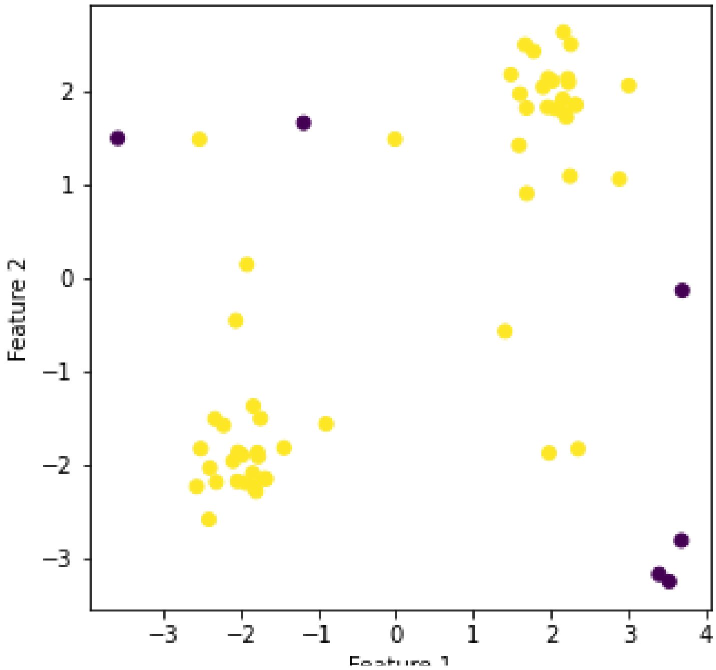
```
model=IsolationForest(random_state=0, contamination=0.1)
model.fit(X_train)
```

```
IsolationForest(behaviour='deprecated', bootstrap=False, contamination=0.1,
                max_features=1.0, max_samples='auto', n_estimators=100,
                n_jobs=None, random_state=0, verbose=0, warm_start=False)
```

```
#prediction
y_train = model.predict(X_train)
y_test = model.predict(X_test)
```

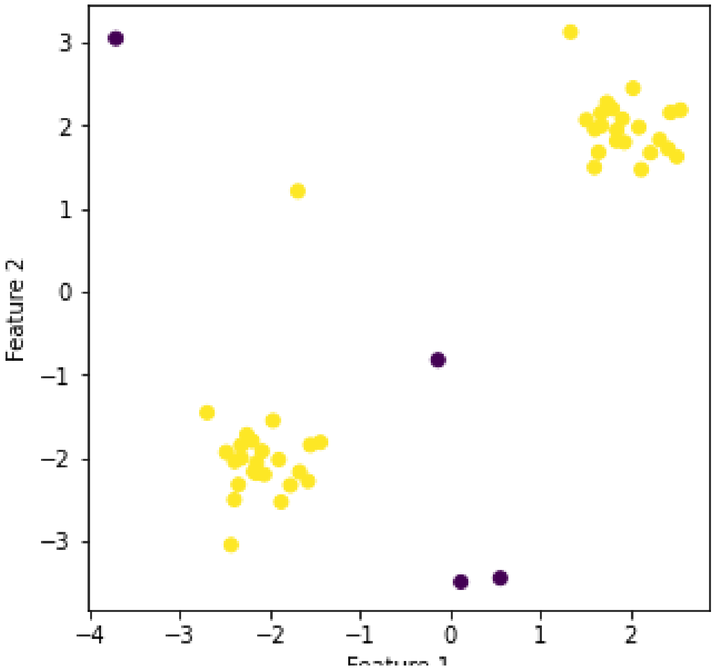
```
#Visualising the result
#plotting the datpoints
plt.figure(figsize=(5,5))
plt.scatter(X_train[:,0],X_train[:,1], c=y_train)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
```

```
Text(0, 0.5, 'Feature 2')
```




```
#Visualising the result
#plotting the datpoints
plt.figure(figsize=(5,5))
plt.scatter(X_test[:,0],X_test[:,1], c=y_test)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
```

```
Text(0, 0.5, 'Feature 2')
```




```
#Part B: Isolation on Credit Card Dataset
df=pd.read_csv("/content/creditcard (1).csv")
print(df.shape)
```

```
(284807, 31)
```

```
df.describe()
```

	Time	V1	V2	V3	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+

```
normal = df[df['Class']==0]
fraud = df[df['Class']==1]
print(normal.shape,fraud.shape)
```

```
(284315, 31) (492, 31)
```

```
data = df.sample(frac=0.2,random_state=1)
```

```
normal_frac = data[data['Class']==0]
fraud_frac = data[data['Class']==1]
print(normal_frac.shape,fraud_frac.shape)
```

```
(56874, 31) (87, 31)
```

```
anomoly_fraction = len(fraud_frac)/float(len(data))
```

```
model = IsolationForest(random_state=1, contamination=anomoly_fraction)
model.fit(data[['Class']])
```

```
IsolationForest(behaviour='deprecated', bootstrap=False,
                contamination=0.0015273608258281983, max_features=1.0,
                max_samples='auto', n_estimators=100, n_jobs=None,
                random_state=1, verbose=0, warm_start=False)
```

```
#Decesion boundary for class 0 or 1
data['scores'] = model.decision_function(data[['Class']])
data['anomaly_scores'] = model.predict(data[['Class']])
```

```
anomaly_count = data[data['Class']==1]
anomaly_count = anomaly_count.shape[0]
```

```
anomaly_count
```

```
87
```

```
accuracy = 100*list(data['anomaly_scores']).count(-1)/(anomaly_count)
accuracy
```

```
100.0
```

✓ 0s completed at 5:37 PM