Trabalho_3

November 29, 2021

1 Tarefa 03

- Andreza Aparecida dos Santos - RA 164213
- Gil Ribeiro de Carvalho - RA 225323
- Thamiris Coelho - RA 187506

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.neighbors import LocalOutlierFactor
from sklearn.ensemble import IsolationForest
from sklearn import svm
from sklearn.cluster import DBSCAN
```

1.1 Leitura dos Dados

Conjunto possui 900 dados "normais" e até 7 outliers.

[2]:
```
data = pd.read_csv('dados3.csv')
data.head()
```

[2]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	-2.97	1.020	-2.340	3.460	1.630	0.157	-2.660	0.559	-5.27	1.960
1	4.30	-0.817	1.410	-2.160	0.673	0.870	-1.220	1.620	3.43	-0.771
2	-2.62	0.378	-1.010	1.430	-0.278	-0.384	0.613	-0.880	-2.14	0.465
3	2.38	-0.356	0.731	-1.250	0.391	0.362	-0.817	1.000	1.85	-0.260
4	1.87	-0.568	0.440	-0.856	0.401	0.576	-0.568	0.793	1.55	-0.412

[3]:
```
data.shape
```

[3]:
```
(907, 10)
```

[4]:
```
# Variables

contamination = 0.01
```

Consideramos que existe uma contaminação de 1% nos dados, já que é um parâmetro configurável em alguns dos algorítmos implementados em Python.

1.2 Local Outlier Factor

Para a escolha dos parâmetros do Local Outlier Factor, foram testadas as variações de número de vizinhos (2, 30, 100 e 300) e contaminação de 1%. Para qualquer caso, obtivemos 10 outliers, muito próximo dos 9 esperados.

O fato do valor dar o mesmo para todo caso pode ser atribuido à restrição da taxa de contaminação.

[5]:
```
clf = LocalOutlierFactor(n_neighbors=100, contamination=contamination)
lof = clf.fit_predict(data)

lof_anomalies = np.where(lof == -1)[0]
unique, counts = np.unique(lof, return_counts=True)
lof_clusters = dict(zip(unique, counts))

print("Número de anomalias detectadas:", lof_clusters[-1])
```

```
Número de anomalias detectadas: 10
```

Considerando o LOF com número de vizinhos de 100 e índice de contaminação de 1%, obtivemos 10 outliers.

1.3 Isolation Forest

Para a escolha dos parâmetros do Isolation Forest, foram testadas as variações com número de estimadores (30, 100, 400 e 1000) e contaminação de 1%. Para todos os valores testados, obtivemos 10 outliers, muito próximo dos 9 esperados. O fato do valor dar o mesmo para todas as variações testadas pode ser atribuido à restrição devido a taxa de contaminação utilizada.

Por conta disso, foi utilizado o valor padrão de número de estimadores, que é 100.

[6]:
```
clf = IsolationForest(random_state=42, contamination=contamination).fit(data)
isolation = clf.predict(data)

isolation_anomalies = np.where(isolation == -1)[0]
unique, counts = np.unique(isolation, return_counts=True)
isolation_clusters = dict(zip(unique, counts))
```

```
/l/disk0/andrezasa/.local/lib/python3.8/site-packages/sklearn/base.py:441:
UserWarning: X does not have valid feature names, but IsolationForest was fitted
with feature names
  warnings.warn(
/l/disk0/andrezasa/.local/lib/python3.8/site-packages/sklearn/base.py:441:
UserWarning: X does not have valid feature names, but IsolationForest was fitted
with feature names
  warnings.warn(
```

[7]:
```
print("Número de anomalias detectadas:", isolation_clusters[-1])
```

```
Número de anomalias detectadas: 10
```

Considerando o default (100 árvores), com índice de contaminação de 1%, obtivemos 10 outliers.

1.4 One Class SVM

Para a escolha dos parâmetros do One Class SVM, foram testadas as variações de nu (0.001, 0.005, 0.01, 0.015 e 0.02). Encontramos variados números de outliers, observando uma queda inicial entre 0.001 e 0.005, chegando a 6 outliers, e voltando a subir já com 0.01, onde temos 7 outliers detectados.

Constatamos que para valores de nu maiores que 0.01, já no valor de 0.015, o número de outliers detectado dispara, chegando a 19 outliers detectados para nu = 0.02, passando do valor da contaminação esperada.

Consideraremos então o valor de nu de 0.01 que proprociona 7 outliers, estando dentro dos valores esperados de contaminação.

[8]:
```
clf = svm.OneClassSVM(nu=contamination, kernel="rbf").fit(data)
ol = clf.predict(data)

ol_anomalies = np.where(ol == -1)[0]
unique, counts = np.unique(ol, return_counts=True)
ol_clusters = dict(zip(unique, counts))
```

[9]:
```
print("Número de anomalias detectadas:", ol_clusters[-1])
```

```
Número de anomalias detectadas: 7
```

Inserindo o nu equivalente ao índice de contaminação anteriormente definido (1%), o resultado obtido foi de 7 outliers.

1.5 DBSCan

Para a escolha dos parâmetros para o DBSCan, foram testadas as variações de amostras vizinhas (2, 4, 8 e 16) e eps (0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5 e 5). Foi constatado que para os variados números de amostras vizinhas, dentro da faixa de eps de 1 ou 1.5 a 3, existe uma constância no valor de outliers detectados, no caso, 7, próximo dos 9 esperados com a contaminação de 1%.

Portanto consideraremos o valor de 2 amostras vizinhas e eps de 2, ficando com 7 outliers, pouco abaixo da contaminação esperada.

[10]:
```
outlier_detection = DBSCAN(min_samples = 2, eps = 3)
clusters = outlier_detection.fit_predict(data)

dbscan_anomalies = np.where(clusters == -1)[0]
unique, counts = np.unique(clusters, return_counts=True)
dbscan_clusters = dict(zip(unique, counts))

print("Número de anomalias detectadas:", dbscan_clusters[-1])
```

```
Número de anomalias detectadas: 7
```

1.6 Combinando os resultados

Por fim, os resultados dos três algorítmos foram combinados afim de descobrir quais pontos são considerados outliers por todos os algorítmos testados.

[11]:
```
total_anomalies = list(set(dbscan_anomalies) & set(ol_anomalies) &␣
↪set(isolation_anomalies) & set(lof_anomalies))
print("Número de anomalias combinando os resultados:", len(total_anomalies))
```

```
Número de anomalias combinando os resultados: 4
```

[12]:
```
print("Dados anomalos encontrados combinando os resultados de todas as técnicas␣
↪aplicadas")
data.loc[data.index.isin(total_anomalies)]
```

```
Dados anomalos encontrados combinando os resultados de todas as técnicas
aplicadas
```

[12]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	-2.97	1.020	-2.340	3.46	1.630	0.157	-2.660	0.559	-5.270	1.96
358	-1.06	-0.771	0.273	-1.49	-1.930	-0.709	3.400	-1.430	2.410	-1.02
554	-3.68	-1.890	-4.730	1.19	0.696	0.306	-0.464	1.630	-0.680	1.66
832	-2.15	0.469	-1.350	-1.12	-1.200	-2.070	0.909	0.224	0.527	1.67