MO433

Alunos:

- Gian Franco Joel Condori Luna (RA 234826)
- Jarol Vijay Butron Soria (RA 234833)
- Walter Augusto Perez Casas (RA 187990)

Problema

Os dados em dados3.csv contem 900 dados “normais” e até 7 outliers.

Use pelo menos 3 algoritmos/técnicas diferentes para detectar os outliers.

Leitura de dados

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

import seaborn as sns

from sklearn.decomposition import PCA
```

In [2]:
```
df = pd.read_csv('https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/dados3.
```

In [3]:
```
df.head()
```

Out[3]:

	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

Algoritmo 1: DBSCAN

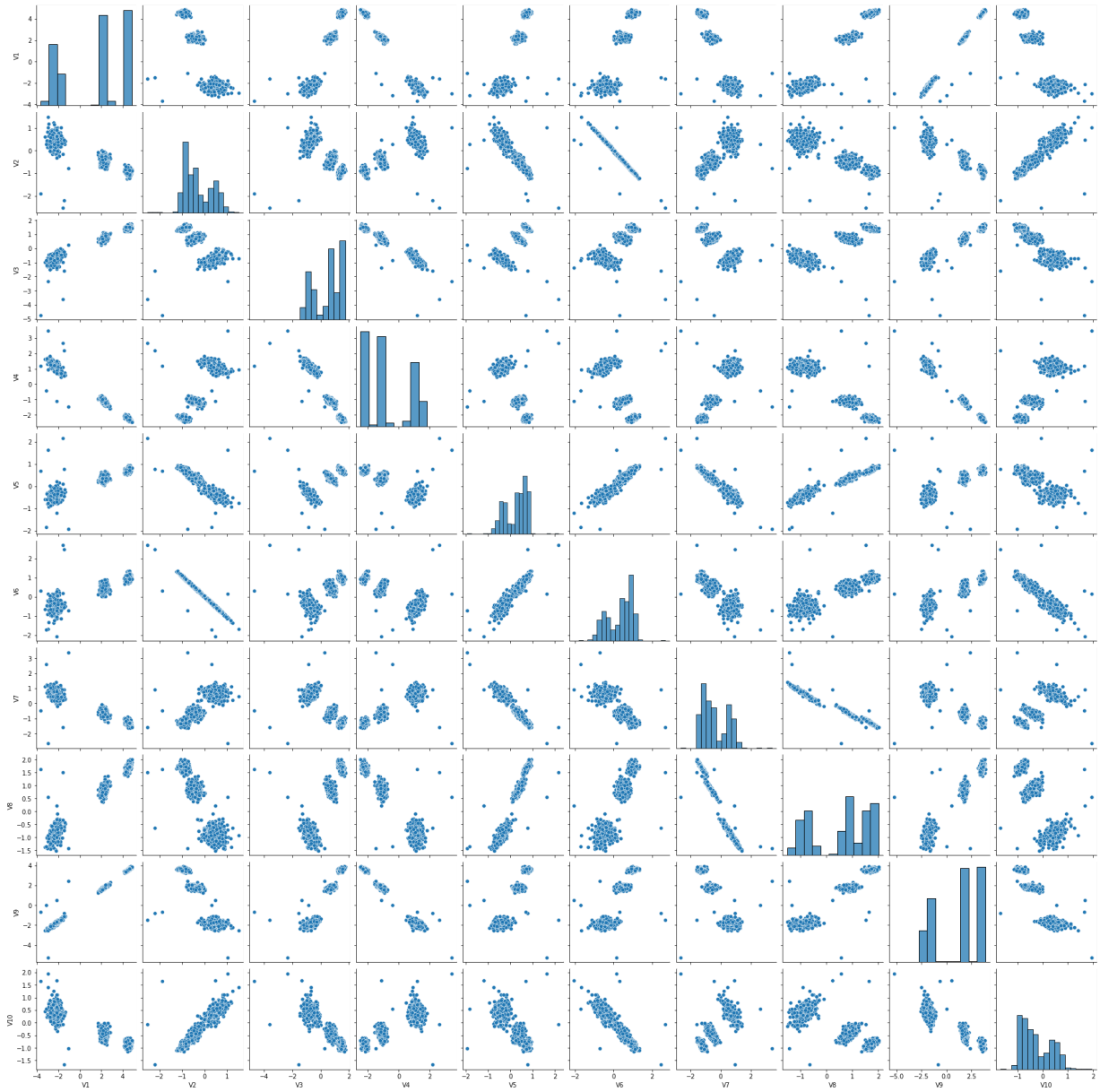
In [4]:
```
df.describe()
```

Out[4]:

	V1	V2	V3	V4	V5	V6	V7	
count	907.000000	907.000000	907.000000	907.000000	907.000000	907.000000	907.000000	907.0
mean	1.457850	-0.307068	0.449542	-0.718637	0.247354	0.324393	-0.425596	0.5
std	2.847931	0.604886	0.976490	1.423629	0.488978	0.657623	0.837753	1.0

	V1	V2	V3	V4	V5	V6	V7	
min	-3.680000	-2.540000	-4.730000	-2.480000	-1.930000	-2.070000	-2.660000	-1.5
25%	-2.075000	-0.830000	-0.581500	-2.170000	-0.219000	-0.283500	-1.200000	-0.6
50%	2.240000	-0.471000	0.699000	-1.100000	0.375000	0.501000	-0.645000	0.8
75%	4.420000	0.263500	1.385000	0.979500	0.678500	0.890500	0.446000	1.6
max	4.860000	1.500000	1.750000	3.460000	2.160000	2.690000	3.400000	2.0

In [5]:
```
sns.pairplot(df)
plt.show()
```



Do gráfico anterior é possível observar os pontos que seriam os outliers

O algoritmo de DBSCAN é baseado em densidades. Os valores que não pertencem a um cluster são considerados como outliers.

In [6]:
```
from sklearn.cluster import DBSCAN
```

In [7]:
```
samples_list = [7, 8, 9, 10, 11, 12]
```

```
eps_list = [0.8, 0.85, 0.9, 0.95, 1, 1.05]

for eps, sample in zip(eps_list, samples_list):
    print(f"eps: {eps} - minSample: {sample}")
    model = DBSCAN(eps=eps, min_samples=sample).fit(df)

    labels = model.labels_
    labels_df = pd.DataFrame(labels, columns=['cluster'])
    quantityOutliers = labels_df['cluster'].value_counts()[-1]

    print("\tQuantidade de Outliers:", quantityOutliers)
```

```
eps: 0.8 - minSample: 7
        Quantidade de Outliers: 8
eps: 0.85 - minSample: 8
        Quantidade de Outliers: 8
eps: 0.9 - minSample: 9
        Quantidade de Outliers: 8
eps: 0.95 - minSample: 10
        Quantidade de Outliers: 7
eps: 1 - minSample: 11
        Quantidade de Outliers: 7
eps: 1.05 - minSample: 12
        Quantidade de Outliers: 7
```

Testamos diferentes combinações de raio e número mínimo de amostras. A quantidade de outliers fica entre 8 e 7.

Agora escolhemos um dos casos testados para olhar os outliers que achou.

In [8]:
```
model = DBSCAN(eps=0.95, min_samples=9).fit(df)
model
```

Out[8]:
```
DBSCAN(eps=0.95, min_samples=9)
```

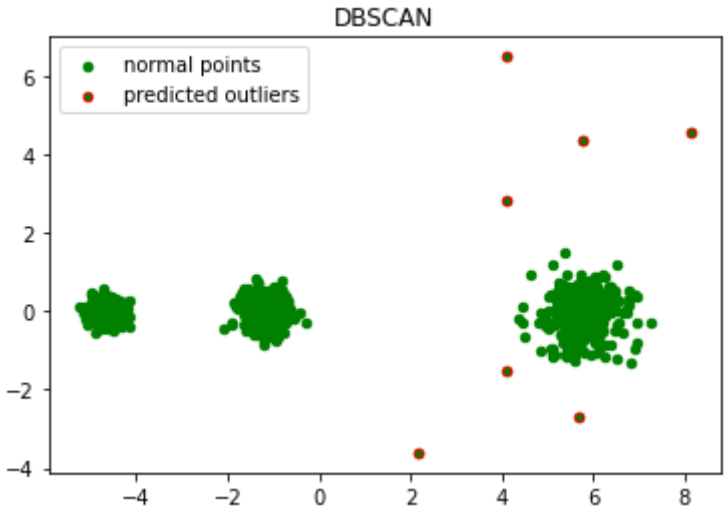
In [9]:
```
labels = model.labels_
labels_df = pd.DataFrame(labels, columns=['cluster'])
labels_df['cluster'].value_counts()
```

Out[9]:
```
 1    300
 2    300
 0    300
-1      7
Name: cluster, dtype: int64
```

Finalmente, o algoritmo indetificou 3 clusters e os outliers, definido com o label "-1", onde é possível observar que têm 7 outliers.

Faremos uma redução de dimensionalidade para mostrar os outliers.

In [11]:
```
outliersIndex = labels_df[labels_df['cluster'] == -1].index

pca = PCA(2)
pca.fit(df)
df_2d = pd.DataFrame(pca.transform(df))

plt.title("DBSCAN")
plt.scatter(df_2d[0], df_2d[1], c='green',s=20,label="normal points")
plt.scatter(df_2d.iloc[outliersIndex,0],df_2d.iloc[outliersIndex,1], c='greer
plt.legend(loc="upper left")
plt.show()
```



Algoritmo 2: Isolation forest

Isolation forest é um algoritmo de aprendizagem não supervisionado que identifica anomalias isolando outliers nos dados. Utilizando árvores binárias identifica quão isolado é um ponto de dados do resto dos dados.

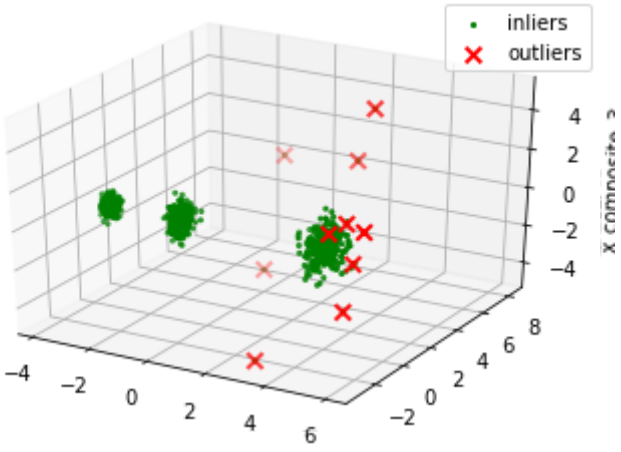
In [12]:
```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest

data = df.copy()
```

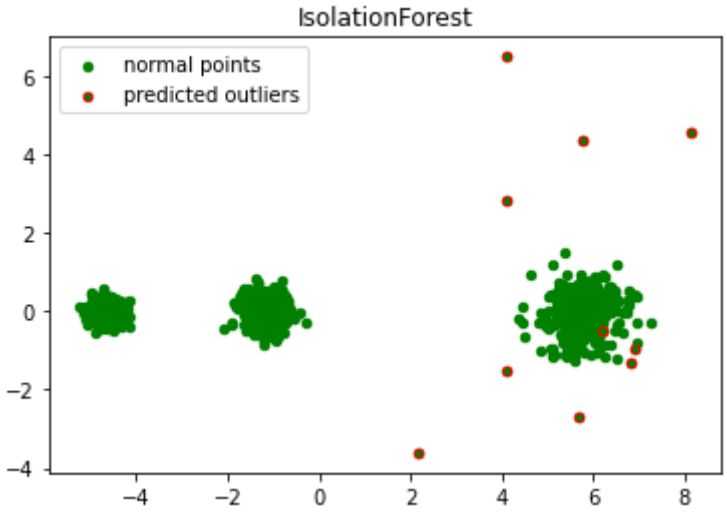
Para a criação do modelo lhe enviamos como parâmetros: **n_estimators** que é o número de estimadores de base ou árvores no conjunto, **max_samples** que é o número de amostras que serão retiradas para treinar cada estimador de base, **contamination** que é a proporção esperada de outliers no conjunto de dados e o **max_features** que é o número de características a extrair das características totais para treinar cada estimador de base ou árvore.

In [13]:
```
# Criação do modelo
model=IsolationForest(n_estimators=50, max_samples='auto', contamination=floa
model.fit(data[['V1','V2','V3','V4','V5','V6','V7','V8','V9','V10']])

# Agregação de colunas de pontuações e anomalias
data['scores']=model.decision_function(data[['V1','V2','V3','V4','V5','V6','V
data['anomaly']=model.predict(data[['V1','V2','V3','V4','V5','V6','V7','V8','

# Análise das anomalias previstas
outliers=data.loc[data['anomaly']==-1]
outlier_index=list(outliers.index)

print(data['anomaly'].value_counts())

# Visualização de os resultados - 3D
pca = PCA(n_components=3)
scaler = StandardScaler()
X = scaler.fit_transform((data[['V1','V2','V3','V4','V5','V6','V7','V8','V9',
X_reduce = pca.fit_transform(X)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("x_composite_3")
```

```
ax.scatter(X_reduce[:, 0], X_reduce[:, 1], zs=X_reduce[:, 2], s=4, lw=1, labe
ax.scatter(X_reduce[outlier_index,0],X_reduce[outlier_index,1], X_reduce[outl
ax.legend()
plt.show()


# Visualização de os resultados - 2D
pca = PCA(2)
pca.fit(data[['V1','V2','V3','V4','V5','V6','V7','V8','V9','V10']])
res=pd.DataFrame(pca.transform(data[['V1','V2','V3','V4','V5','V6','V7','V8',
Z = np.array(res)
plt.title("IsolationForest")
b1 = plt.scatter(res[0], res[1], c='green',s=20,label="normal points")
b1 =plt.scatter(res.iloc[outlier_index,0],res.iloc[outlier_index,1], c='greer
plt.legend(loc="upper left")
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning: X do
es not have valid feature names, but IsolationForest was fitted with feature
names
  "X does not have valid feature names, but"
 1    897
-1     10
Name: anomaly, dtype: int64
```





Algoritmo 3: LOF

O Fator de Anomalia Local é um algoritmo para detectar anomalias em dados observacionais. Medir a pontuação da densidade local de cada amostra e ponderar suas pontuações é o principal conceito do algoritmo. Ao comparar a pontuação da amostra com seus vizinhos, o

algoritmo define elementos de menor densidade como anomalias nos dados. O Fator de Anomalia Local é um algoritmo para detectar anomalias em dados observacionais. Medir a pontuação da densidade local de cada amostra e ponderar suas pontuações é o principal conceito do algoritmo. Ao comparar a pontuação da amostra com seus vizinhos, o algoritmo define elementos de menor densidade como anomalias nos dados.

In [14]:
```
from sklearn.neighbors import LocalOutlierFactor
```

In [15]:
```
df_lof = df.copy()
```

In [16]:
```
range = [5, 10, 15, 20, 25]
for i in range:
  clf = LocalOutlierFactor(n_neighbors=i, contamination=0.01)
  outlier_lof = clf.fit_predict(df_lof)
  print((outlier_lof == -1).sum())#Outlier
```

```
10
10
10
10
10
```

Ao testar com números diferentes de vizinhos [5, 10, 15, 20, 25] vemos que o resultado não varia, portanto, neste caso, usaremos 20 vizinhos.

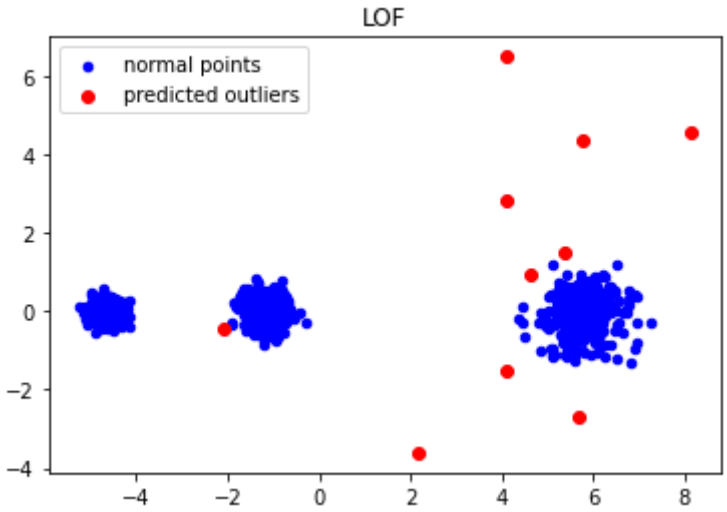
In [17]:
```
clf = LocalOutlierFactor(n_neighbors=20, contamination=0.01)
outlier_lof = clf.fit_predict(df_lof)
print((outlier_lof == -1).sum())#Outlier
```

```
10
```

In [18]:
```
labels_df = pd.DataFrame(outlier_lof, columns=['Result'])
labels_df['Result'].value_counts()
```

Out[18]:
```
 1    897
-1     10
Name: Result, dtype: int64
```

In [19]:
```
# Visualização de os resultados - 2D
pca_lof = PCA(2)
pca_lof.fit(df_lof[['V1','V2','V3','V4','V5','V6','V7','V8','V9','V10']])
res_lof = pd.DataFrame(pca_lof.transform(df_lof[['V1','V2','V3','V4','V5','V6
y_pred = clf.fit_predict(res_lof)
lofs_index = np.where(y_pred==-1)
plt.title("LOF")
b2 = plt.scatter(res_lof[0], res_lof[1], c='blue', s=20, label="normal points
b2 = plt.scatter(res_lof.iloc[lofs_index][0],res_lof.iloc[lofs_index][1], c='
plt.legend(loc="upper left")
plt.show()
```



O gráfico mostra os 10 outliers vermelhos.