Curso Big Data e Python
Aula 14

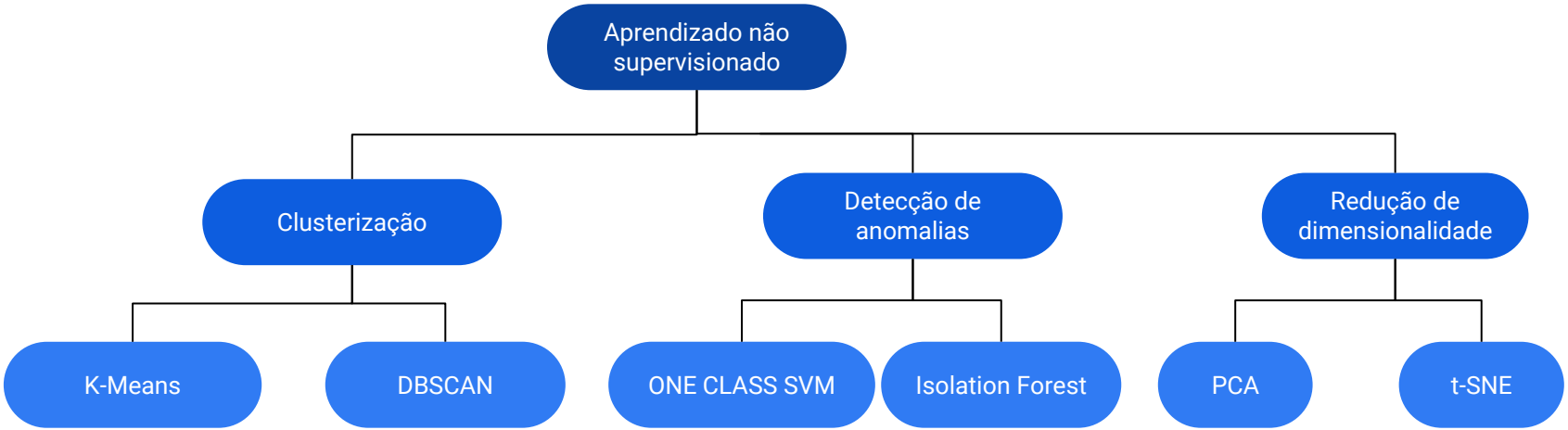
Prof. Me Daniel Vieira

SENAI

Agenda

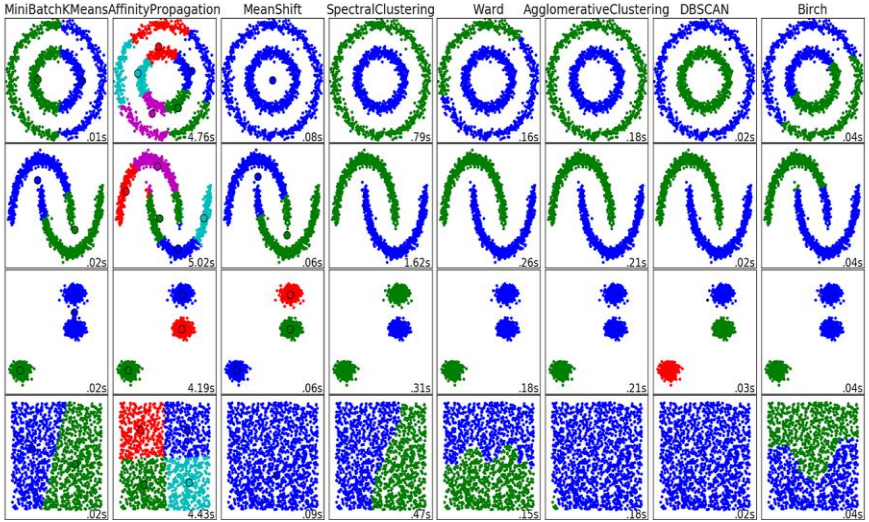
1 Aprendizado não supervisionado
2 Algoritmos utilizados no aprendizado de máquina não supervisionado;
3 Exemplo DBSCAN
4 Exercícios

Algoritmos utilizados no aprendizado de máquina não supervisionado



Algoritmos utilizados no aprendizado de máquina não supervisionado

DBSCAN- Density Based Spatial Clustering Of Applications with Noise



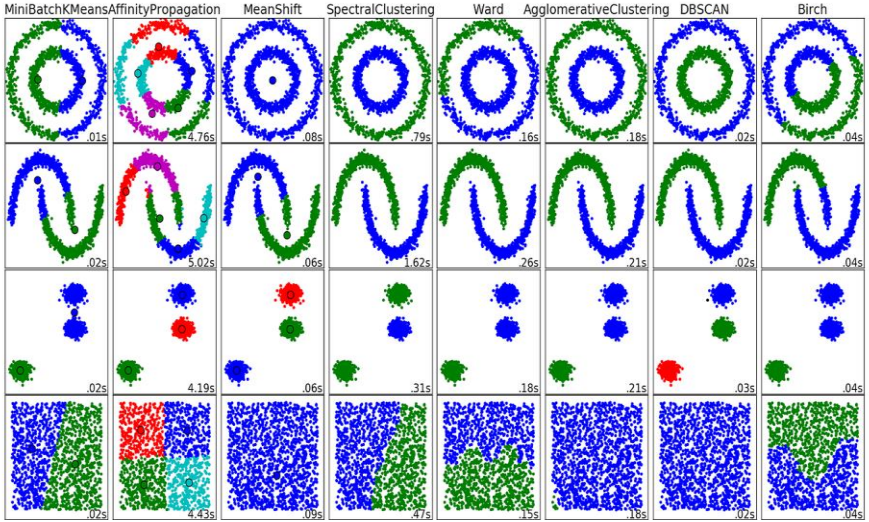
Caso um determinado ponto não obedeça critérios de densidade ou critérios dos limites de distância, este não pode ser classificado em um cluster.

Os dois parâmetros necessários para aplicação do modelo DBSCAN são:

Eps: Raio máximo em que dois pontos podem estar para serem considerados do mesmo cluster.
MinPts: Número mínimo de pontos em uma região necessários para garantir uma densidade desejada.

Algoritmos utilizados no aprendizado de máquina não supervisionado

DBSCAN- Density Based Spatial Clustering Of Applications with Noise



Caso um determinado ponto não obedeça critérios de densidade ou critérios dos limites de distância, este não pode ser classificado em um cluster.

A escolha dos parâmetros EPS e Minpts é crucial para o bom funcionamento do algoritmo.
Eps pequeno levará a muitos ruídos, já que não será possível classificar os pontos mais distantes. Porém escolher um valor alto pode levar um cluster muito generalizado.

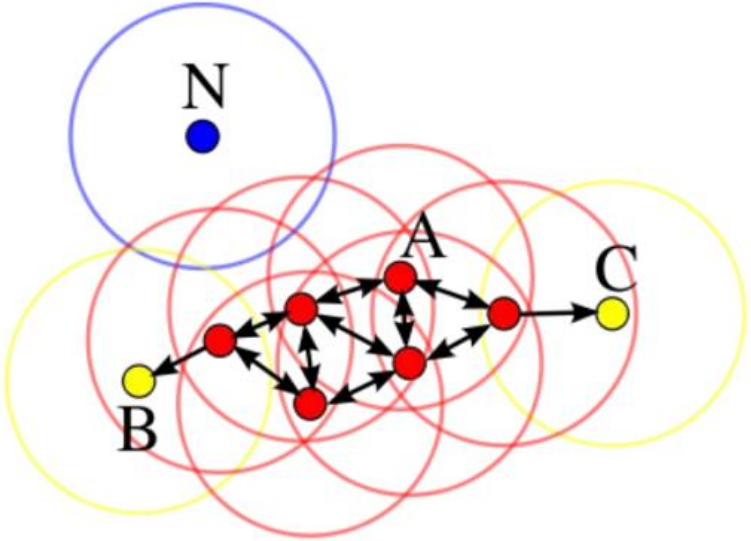
Algoritmos utilizados no aprendizado de máquina não supervisionado

Três tipos de pontos é possível encontrar em um dataset

Núcleo Um ponto é considerado núcleo quando possui MinPts pontos ou mais dentro de um raio de Eps de distância.

Bordas Um ponto é considerado borda quando não possui MinPts dentro de um raio Eps de distância, porém está inserido dentro de um raio Eps de um outro ponto no qual é núcleo.

Ruídos ou outliers Um ponto é considerado um ruído quando este não está presente no raio Eps de um núcleo e nem possui MinPts dentro do seu raio Eps.



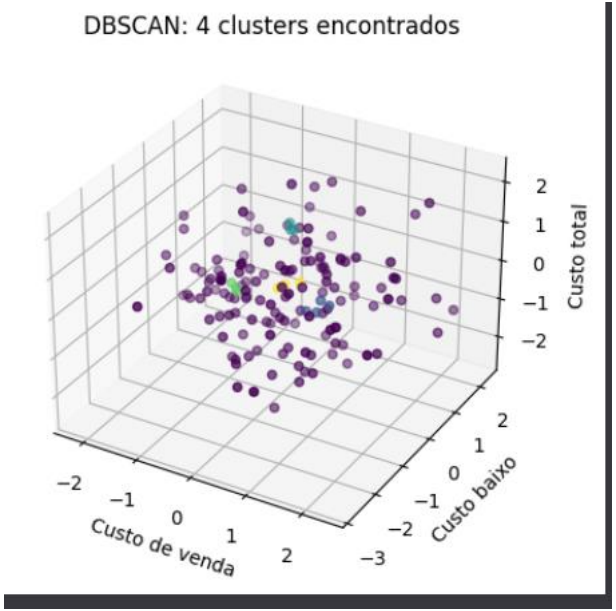
Pontos vermelhos são núcleos, amarelos bordas e azuis ruídos

Exemplo algoritmo DBSCAN

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import DBSCAN
# Gerando dados de exemplo
np.random.seed(0)
n_samples = 150
X = np.random.randn(n_samples, 3)
# Executando o algoritmo DBSCAN
dbscan = DBSCAN(eps=0.3, min_samples=3)
dbscan.fit(X)
# Obtendo os rótulos dos clusters
labels = dbscan.labels_
# Obtendo a quantidade de clusters encontrados
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
n_noise = list(labels).count(-1)
# Criando o gráfico em 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=labels)
```

Exemplo algoritmo DBSCAN

```
ax.set_title('DBSCAN: %d clusters encontrados' % n_clusters)
ax.set_xlabel('Custo de venda')
ax.set_ylabel('Custo baixo')
ax.set_zlabel('Custo total')

plt.show()
```



Exemplo algoritmo DBSCAN

```
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Passo 2: Carregar os dados dos filmes (exemplo hipotético)
dados_filmes = pd.DataFrame({
    'Gênero': ['Ação', 'Aventura', 'Comédia', 'Drama', 'Romance',
'Terror', 'Suspense'],
    'Duração': [120, 150, 90, 120, 110, 100, 130],
    'Avaliação': [4.5, 4.2, 3.9, 4.1, 3.8, 4.3, 3.7]
})
```

Exemplo algoritmo DBSCAN

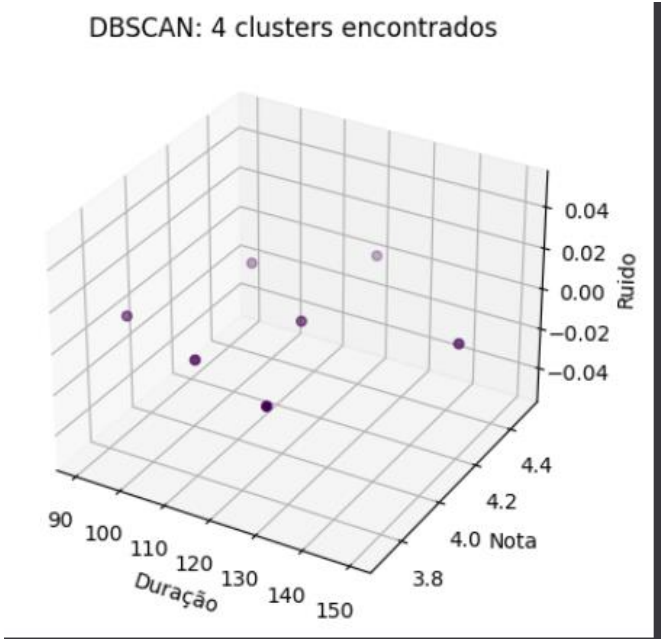
```
# Passo 3: Visualizar os dados
print(dados_filmes)

# Passo 4: Pré-processamento dos dados (opcional)
# Neste exemplo hipotético, não há pré-processamento necessário.

# Passo 5: Aplicar o algoritmo DBSCAN
dbscan = DBSCAN(eps=10, min_samples=10)  # Definindo os
parâmetros eps e min_samples
clusters = dbscan.fit_predict(dados_filmes[['Duração', 'Avaliação']])
```

Exemplo algoritmo DBSCAN

```
# Passo 6: Avaliar o desempenho do modelo DBSCAN
numero_clusters = len(np.unique(clusters)) - 1  # Desconsiderando o
cluster de ruído (-1)
print("Número de clusters:", numero_clusters)
# Passo 7: Visualizar os resultados do agrupamento
plt.scatter(dados_filmes['Duração'], dados_filmes['Avaliação'],
c=clusters)
plt.xlabel('Duração')
plt.ylabel('Avaliação')
plt.title('Agrupamento de Filmes')
plt.show()
# Passo 8: Analisar os grupos resultantes
dados_filmes['Cluster'] = clusters
grouped_data = dados_filmes.groupby('Cluster')
print(grouped_data)
```

Exemplo algoritmo DBSCAN

```
# Criando o gráfico em 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
#ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=labels)
ax.scatter(dados_filmes['Duração'], dados_filmes['Avaliação'],
c=clusters)

ax.set_title('DBSCAN: %d clusters encontrados' % n_clusters)
ax.set_xlabel('Duração')
ax.set_ylabel('Nota')
ax.set_zlabel('Ruido')

plt.show()
```



Exercício

1)Você trabalha em uma imobiliária e recebeu um conjunto de dados contendo informações sobre imóveis, incluindo a área (em metros quadrados) e a distância até a praia (em metros). Seu objetivo é utilizar o algoritmo DBSCAN para identificar grupos de imóveis com características semelhantes e, em seguida, analisar o valor médio dos imóveis em cada grupo.

Os dados serão gerados utilizando arrays numpy , com as seguintes colunas: "area", "distancia_praia" e "valor".

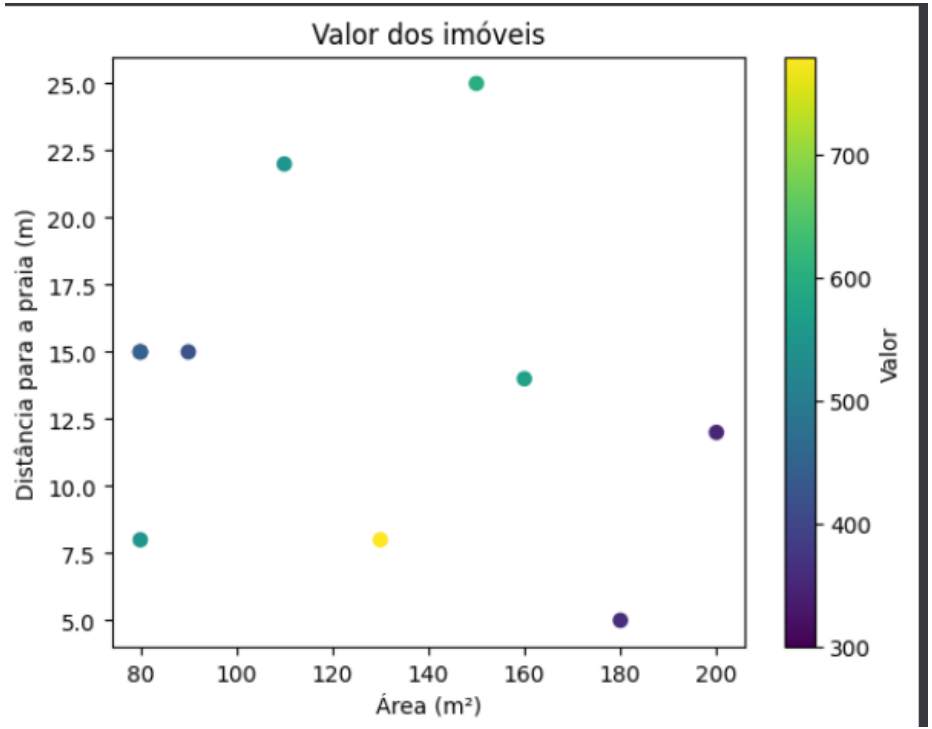
Exercício

```
import pandas as pd

imoveis ={
        'Area':[80,80,80,150,200,90,110,130,180,160],
        'Valor':[300,450,550,600,350,420,550,780,360,575],
        'Dist_praia':[15,15,8,25,12,15,22,8,5,14]

}
Label =['1º','2º','3º','4º','5º','6º','7º','8º','9º','10º']
imoveis_df=pd.DataFrame(imoveis, index= Label)
imoveis_df.head()
```

Exercício

```
import matplotlib.pyplot as plt

# Criar o gráfico de dispersão
plt.scatter(imoveis_df['Area'], imoveis_df['Dist_praia'],
c=imoveis_df['Valor'])
plt.xlabel('Área (m²)')
plt.ylabel('Distância para a praia (m)')
plt.title('Valor dos imóveis')
plt.colorbar(label='Valor')
plt.show()
```

Exercício


Valor dos imóveis
Distância para a praia (m)
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
80
100
120
140
160
180
200
Área (m²)
Valor
700
600
500
400
300

Exercício

```
from sklearn.preprocessing import StandardScaler

# Escalar as variáveis
scaler = StandardScaler()
scaled_data = scaler.fit_transform(imoveis_df[['Area', 'Dist_praia']])

from sklearn.cluster import DBSCAN

# Executar o algoritmo DBSCAN
dbscan = DBSCAN(eps=30, min_samples=2)
dbscan.fit(scaled_data)

# Obter os rótulos dos clusters
labels = dbscan.labels_
```

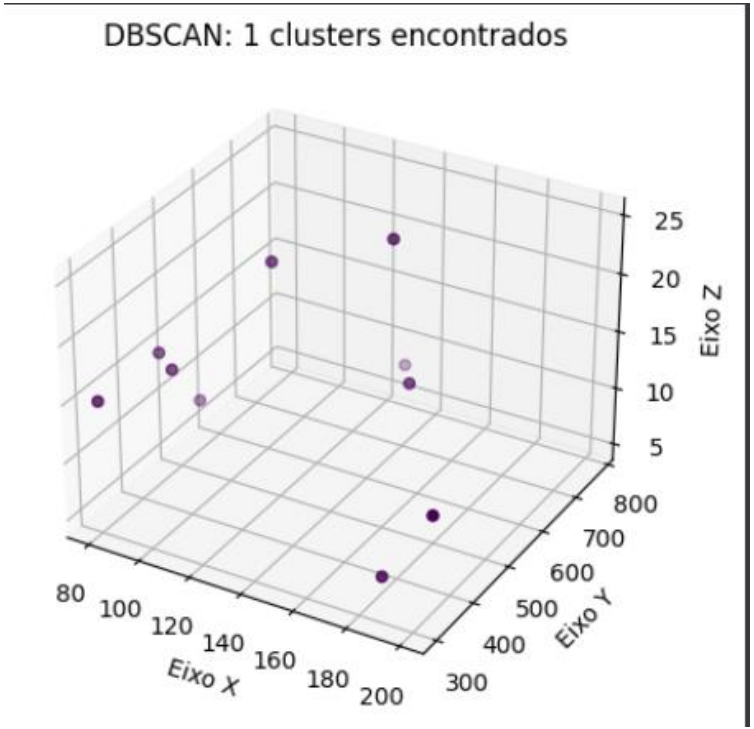
Exercício

```
# Adicionar os rótulos dos clusters ao DataFrame
imoveis_df['cluster'] = labels

# Calcular o valor médio dos imóveis em cada cluster
mean_values =imoveis_df.groupby('cluster')['Valor'].mean()

# Exibir os valores médios dos imóveis em cada cluster
print(mean_values)
# Número de clusters encontrados
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
print(f"Número de clusters encontrados: {n_clusters}")

# Gráfico de dispersão com os clusters
plt.scatter(imoveis_df['Area'], imoveis_df['Dist_praia'], c=imoveis_df['cluster'])
plt.xlabel('Área (m²)')
plt.ylabel('Distância para a praia (m)')
plt.title('Clusters de imóveis')
plt.show()
```

Exercício

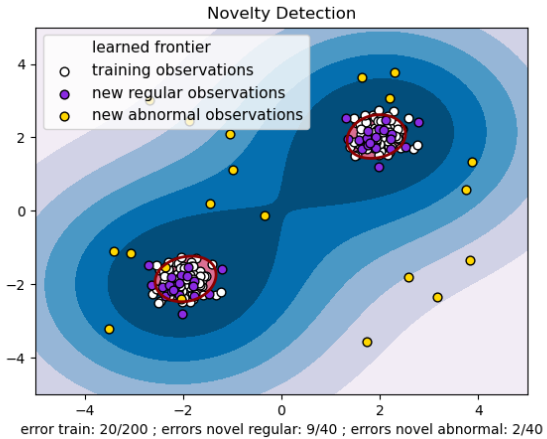
```
from mpl_toolkits.mplot3d import Axes3D
# Criando o gráfico em 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(imoveis_df['Area'], imoveis_df['Valor'], imoveis_df['Dist_praia'],
c=imoveis_df['cluster'])


ax.set_title('DBSCAN: %d clusters encontrados' % n_clusters)
ax.set_xlabel('Eixo X')
ax.set_ylabel('Eixo Y')
ax.set_zlabel('Eixo Z')

plt.show()
```

Exercício

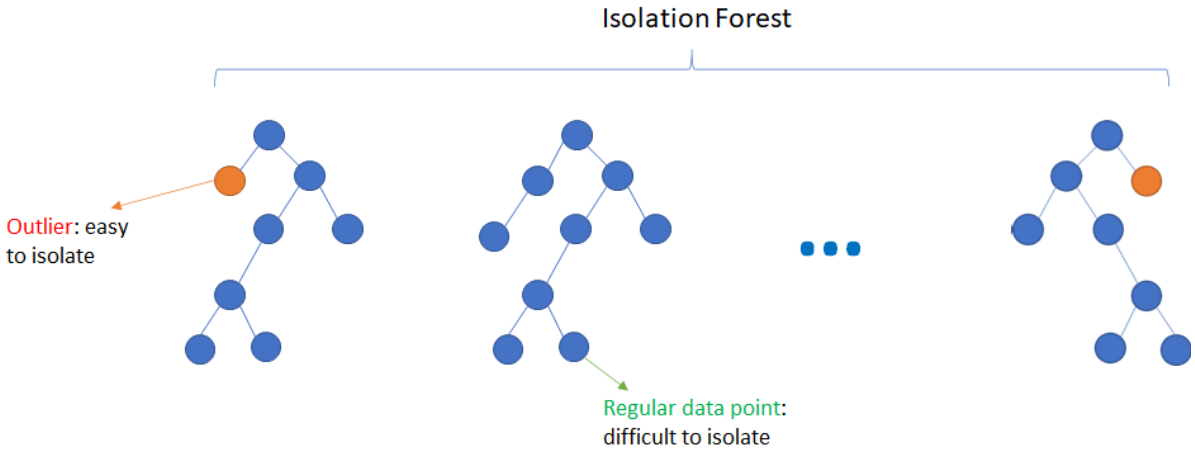

DBSCAN: 1 clusters encontrados
25
20
15
10
5
Eixo Z
800
700
600
500
400
300
Eixo Y
80
100
120
140
160
180
200
Eixo X

Algoritmos utilizados no aprendizado de máquina não supervisionado

One Class SVM



Isolation Forest



Exemplo DBSCAN

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import DBSCAN

# Gerar dados simulados
np.random.seed(0)
n_samples = 1000
tempo = np.random.uniform(low=0, high=10, size=n_samples)
temperatura = np.random.normal(loc=25, scale=5, size=n_samples)
vibracao = np.random.normal(loc=0, scale=2, size=n_samples)
corrente = np.random.normal(loc=10, scale=2, size=n_samples)
# Criar o array NumPy com os dados
dados_maquina = np.column_stack((tempo, temperatura, vibracao, corrente));
```

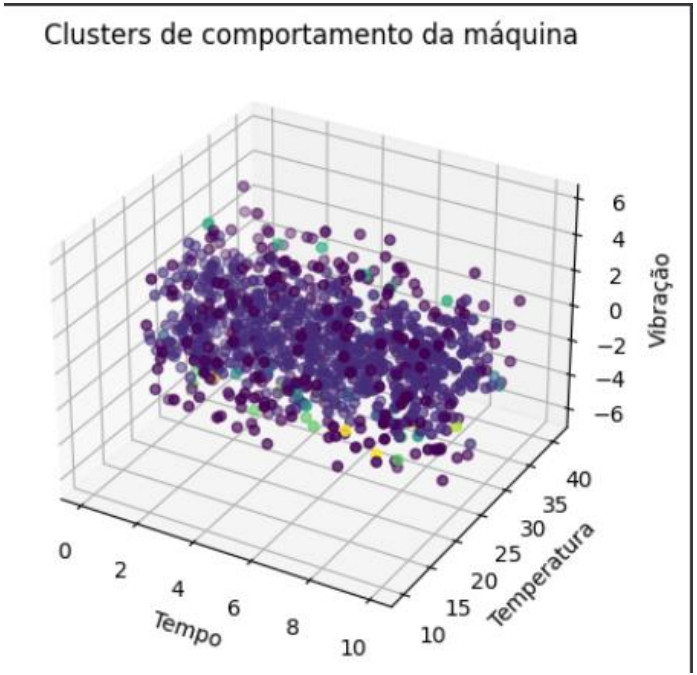
Exemplo DBSCAN

```
# Aplicar o algoritmo DBSCAN
dbscan = DBSCAN(eps=1, min_samples=5)
dbscan.fit(dados_maquina[:, 1:])  # Excluindo a coluna de tempo

# Obter os rótulos dos clusters
labels = dbscan.labels_

# Exibir os resultados
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
print(f"Número de clusters encontrados: {n_clusters}")
```

Exemplo DBSCAN

```
# Gráfico 3D dos clusters
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(tempo, temperatura, vibracao, c=labels)
ax.set_xlabel('Tempo')
ax.set_ylabel('Temperatura')
ax.set_zlabel('Vibração')
plt.title('Clusters de comportamento da máquina')
plt.show()
```

Exemplo DBSCAN



Exemplo Isolation Forest

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest

# Gerar dados simulados
np.random.seed(0)
n_samples = 1000
tempo = np.random.uniform(low=0, high=10, size=n_samples)
temperatura = np.random.normal(loc=25, scale=5, size=n_samples)
vibracao = np.random.normal(loc=0, scale=2, size=n_samples)
corrente = np.random.normal(loc=10, scale=2, size=n_samples)
```

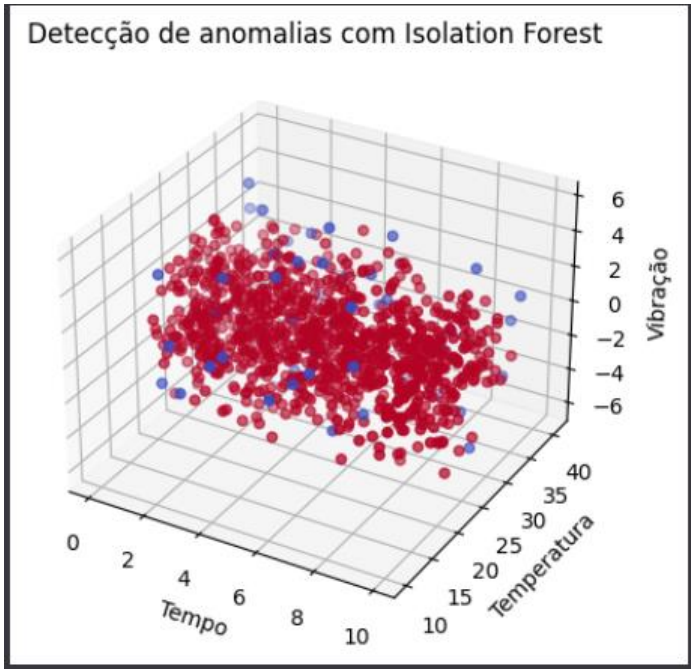
Exemplo Isolation Forest

```
# Criar o array NumPy com os dados
dados_maquina = np.column_stack((tempo, temperatura, vibracao,
corrente))
# Aplicar o algoritmo Isolation Forest
isolation_forest = IsolationForest(contamination=0.05,
random_state=0)
isolation_forest.fit(dados_maquina[:, 1:])  # Excluindo a coluna de
tempo
# Obter os rótulos de anomalia (-1 para anomalias, 1 para dados
normais)
labels = isolation_forest.predict(dados_maquina[:, 1:])
# Exibir os resultados
n_anomalias = np.sum(labels == -1)
print(f"Número de anomalias detectadas: {n_anomalias}")
```

Exemplo Isolation Forest

```
# Gráfico 3D com anomalias destacadas
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(tempo, temperatura, vibracao, c=labels, cmap='coolwarm')
ax.set_xlabel('Tempo')
ax.set_ylabel('Temperatura')
ax.set_zlabel('Vibração')
plt.title('Detecção de anomalias com Isolation Forest')
plt.show()
```

Exemplo Isolation Forest



Exercícios

1) Você trabalha em uma empresa de manutenção industrial e recebeu um conjunto de dados contendo medições de corrente elétrica ao longo do tempo de máquinas. Seu objetivo é utilizar o algoritmo Isolation Forest para identificar anomalias nos padrões de corrente elétrica das máquinas, que podem indicar problemas ou falhas iminentes.
   corrente elétrica = np.array([5, 10, 14, 2, 1.5, 6])
   tempo = np.array([1,2,4,6,7,10])

2) Você é um cientista de dados em um laboratório médico e recebeu um conjunto de dados contendo os resultados de exames de sangue de hemograma de diferentes pacientes. Seu objetivo é utilizar o algoritmo DBSCAN para identificar grupos de pacientes com perfis de hemograma semelhantes.

Leucócitos = 2000,4000,5000,6500,

Plaquetas = 100000,20000,80000,145000,

Linfócitos = 2.3,4.5,6,5,4.4

Exercícios

3) Você é um cientista de dados em um laboratório médico e recebeu um conjunto de dados contendo os resultados de exames de sangue de hemograma de diferentes pacientes. Seu objetivo é utilizar o algoritmo Isolation Forest para identificar grupos de pacientes com anomalias

Leucócitos = 2000,4000,5000,6500,
Plaquetas = 100000,20000,80000,145000,
Linfócitos = 2.3,4.5,6,5,4.4

Link Forms Aprendizado não supervisionado Isolation Forest e DBScan
https://docs.google.com/forms/d/1UiRxNW4JUR5ewl_Haj1kL5me0h1JrkSuEQ3iXuPLnW4/edit

Obrigado!

Prof. Me Daniel Vieira

Email: danielvieira2006@gmail.com

Linkedin: Daniel Vieira

Instagram: Prof daniel.vieira95