ML: Clusterização

Ivan Sendin

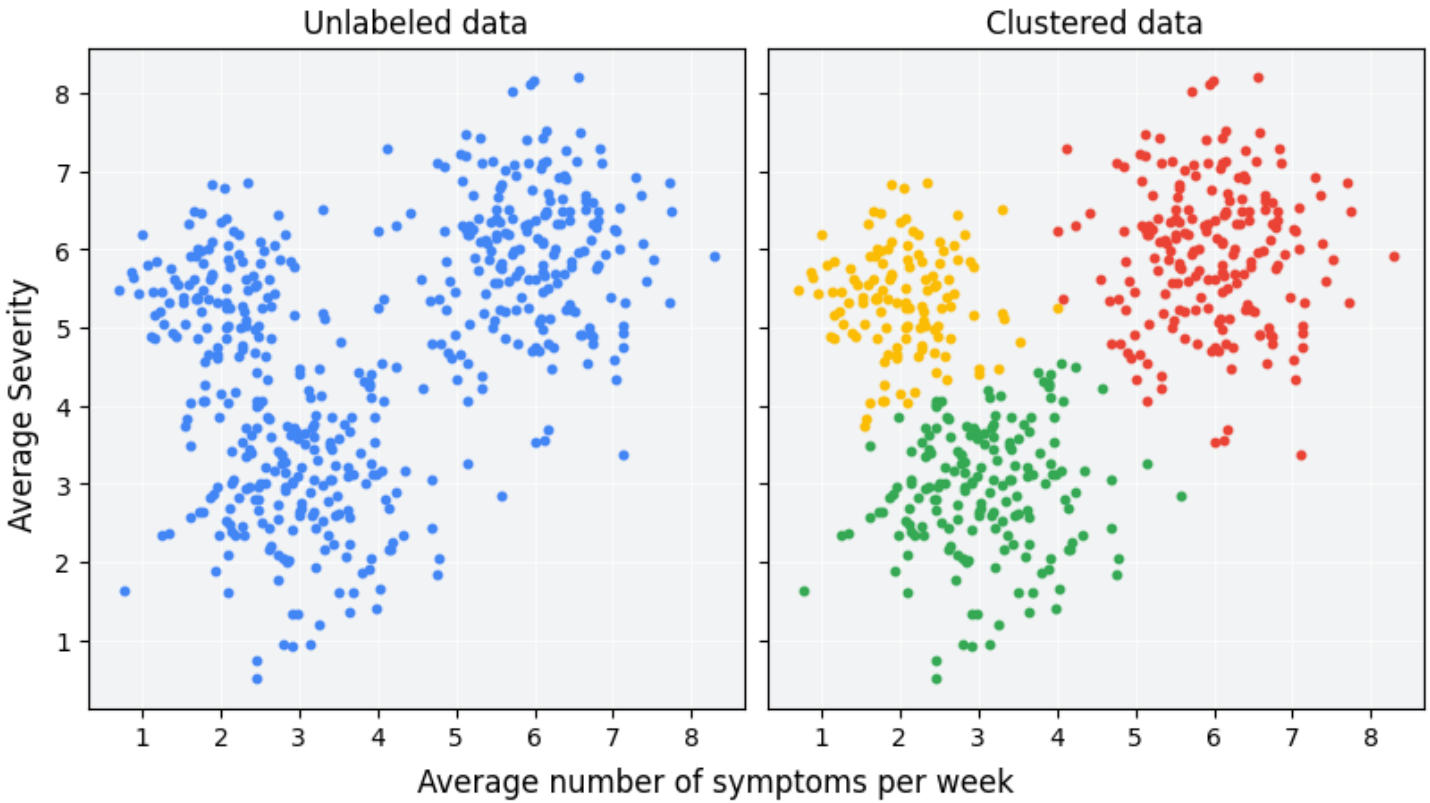
FACOM - Universidade Federal de Uberlândia
ivansendin@yahoo.com,sendin@ufu.br

3 de setembro de 2024

- Aplicação de ML na detecção de transações ilícitas
- Supervisionada
  Preciso ter uma base de dados classificada/rotulada...nem sempre isso é fácil
- Regressão Logística
- Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics
- IBM, Elliptic, MIT (Charles Leiserson)
- Graph Convolutional Networks
- Parte das features
- "Time Step"
  Componente conexa dentro de uma janela de 3 horas
  Achei que dificulta a nossa vida: transferencia de

- Misturar com Deep/Dark Web ?
  Sim..mas não sei extamente como
- K-Means na base de dados
  Deve funcionar..o Pedro deve rodar um experimento
  Mas ja temos os rótulos
- E se tirar os Não-Classificados
- E se homogenizar os tamanhos ??
  Também será testado

- Clustering é uma técnica não supervisionada de Machine Learning que visa **agrupar** entradas não rotuladas baseadas na suas similaridades
- O **cenário** é mais simples.
- Nem sempre os rótulos existem



Fonte: Google Developers.What is clustering?

- A entrada poderia ser peso/altura e o algoritmo dividiria a população em 3 grupos
  Exemplo: 3 esportes diferentes
  Depois de "aprendido" podemos inferir o esporte de um indivíduo "colocando ele no grafico"
- Renda-Escolaridade, X-Y,
- X-Y-Z-W,...
- O Algoritmo trabalha no espaço n-dimensional e determina os mais similares/proximos
- Apresentado em 3D e 2D

- K-Means
- (A Comprehensive Survey of Clustering Algorithms)
- Algoritmo “padrão”
- Escalabilidade
- Simplicidade
  Dados e o valor k: número de clusters

- Identificar Mixer Wasabi
- Exemplo.

Valores Únicos de Saída O processo de *mixing*, que esconde a relação de origem e destino dos valores, só tem efeito se houverem valores repetidos;

Razão entre entrada e saída O CoinJoin do Wasabi une n entidades, e cada entidade deve produzir duas saídas: uma do mixer e outra de troco. Ainda uma última saída para remuneração do operador do mixer. Assim para n entradas espera-se $2n + 1$ saídas;

Reuso de entrada O reuso de endereços de entrada aponta uma falta de preocupação com privacidade;

SegWit O serviço de mixer usa endereços do tipo SegWit. ($bc1...$)

```
[44.        0.63333333  0.          1.
[47.        0.61654135  0.          1.
[18.        0.70731707  0.          1.
[39.        0.64705882  0.          1.
[33.        0.75789474  0.          1.
[44.        0.56730769  1.          1.
[39.        0.54477612  1.          1.
[51.        0.62179487  0.          1.
[45.        0.66666667  1.          1.
[33.        0.70114943  0.          1.
```

O Experimento

- Transações sorteadas da blockchain
- Transações Wasabi
  Towards Understanding and Demystifying Bitcoin Mixing Services
- De fato temos os dados rotulados....mas vamos fingir que não por motivos didaticos

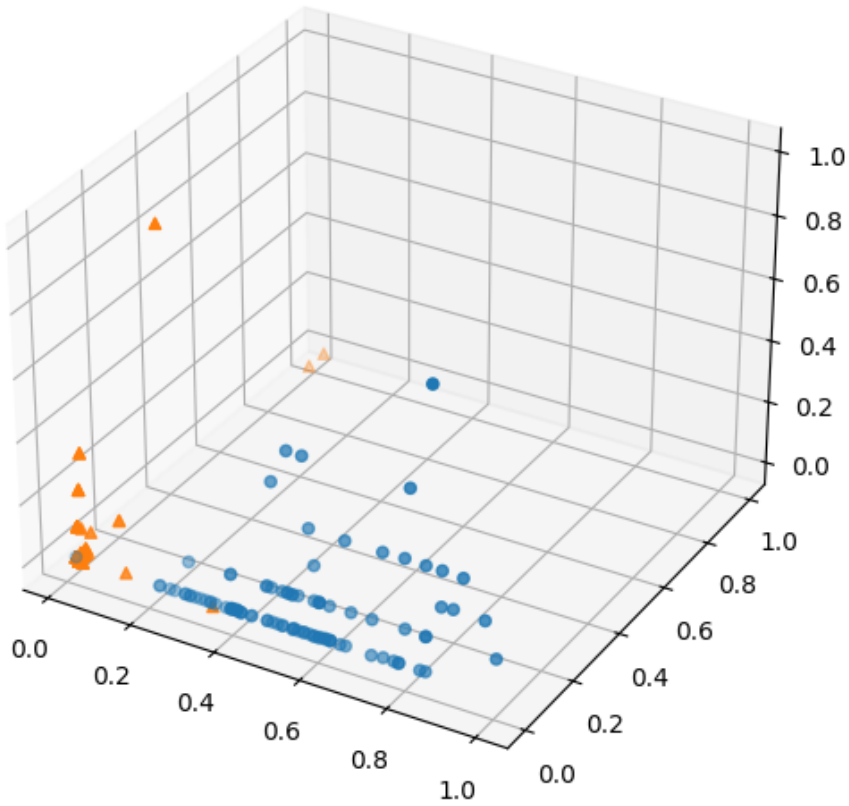
```
import glob
import json
import requests
import numpy as np
import random

import matplotlib.pyplot as plt
from numpy import unique
from numpy import where
from sklearn.cluster import kmeans_plusplus
from sklearn.cluster import KMeans

from matplotlib import pyplot
```

```
def isSegWit(l):
    for a in l:
        if not a[0:3] == "bc1":
            return False
    return True


def features(tx):
    values = [o['value'] for o in tx['out']]

    addrin = [a['prev_out']['addr'] for a in tx['inputs']]
    addrout = [a['addr'] for a in tx['out']]

    reuse =0
    for a in addrin:
        if a in addrout:
            reuse+=1
    return [len(set(values)),len(addrin)/len(addrout),
           reuse,isSegWit(addrin) and isSegWit(addrout)]
```

```
TXS = []
tempa = [a for a in  glob.glob("./txsw/*")]
for a in tempa[:100]:
    with open(a, 'r', encoding='utf-8') as f:
        data = json.load(f)
        TXS.append( features(data) )


tempa = [a for a in  glob.glob("./txnotsw/*")]
for a in tempa[:100]:
    with open(a, 'r', encoding='utf-8') as f:
        data = json.load(f)
        TXS.append( features(data) )

print(len(TXS))

TXS = np.array(TXS)
model = KMeans(n_clusters=2)
model.fit(TXS)
yhat = model.predict(TXS)
clusters = unique(yhat)
```

```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')


m = ['o','^']
for cluster in clusters:
    row_ix = where(yhat == cluster)
    print(row_ix)
    ax.scatter(TXS[row_ix, 0], TXS[row_ix, 1],TXS[row_ix,2],marker=m[cluster])
plt.show()
```

5 minutos de CPU.


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

- Acertou quase tudo
- Lembrando que não “identifica” mixer...
- ...apenas separa em dois grupos