Solução Lista 01

Nome: Caio Stoduto Ervilha
E-mail: caio.stoduto@aluno.ufabc.edu.br
Nome: Nicolas Gomes Greco
E-mail: nicolas.greco@aluno.ufabc.edu.br

25 February, 2025

Exercício 01

a) Uma aplicação clássica na área de classificação é a de 'análise de crédito', onde o agente do empréstimo utiliza os dados do cliente para assim calcular a viabilidade de concessão de crédito. Um exemplo prático seria um dataset que possui inúmeros vetores, que representam os clientes, e cada componente desse vetor seria uma característica que ajude o algoritmo a tomar a decisão entre conceder ou não o crédito àquele usuário, como: histórico de pagamento, renda e outras informações relevantes de sua vida financeira e cada um desses vetores é rotulado como sendo de clientes inadimplentes ou não. Finalmente o algoritmo é devidamente treinado e dado novos vetores, pode-se estimar se o novo cliente pagará suas dívidas devidamente ou não.

b) Um problema de regressão é o cálculo de preços de imóveis numa determinada região, usa-se essa estimativa para dar uma previsão do preço de um imóvel com base nos imóveis de característica mais parecida. O método começa com vetores que são os imóveis do dataset de treinamento, cada vetor possui componentes que são as caractéristicas de cada imóvel, por exemplo, quantidade de quartos, metros quadrados, quantidade de banheiros entre outros. Cada vetor desse também possui seu valor funcional (análogo ao rótulo para problemas de classificação), nesse contexto, é o preço do imóvel. Finalmente, o modelo treinado consegue calcular o valor funcional de um novo vetor, dado os vetores presentes em seu dataset, o que retorna finalmente seu preço estimado.

c) As técnicas de agrupamento são normalmente utilizadas para resolver problemas em que há a necessidade de associações entre dados que não são facilmente vistas sem uma análise estatística completa, dessa forma a máquina fornece insights valiosos a respeito de um conjunto de dados não rotulado. Como por exemplo, no uso de técnicas de agrupamento por exmpresas de vendas online, que usam suas grandes quantidades de dados para melhorar seu marketing, descobrindo associações interessantes, como a relação de um certo perfil de consumidor com um produto específico.

Exercício 02

A Maldição da Dimensionalidade é um fenômeno observado na área de Aprendizado de Máquina, no qual o aumento do número de dimensões (ou variáveis de entrada) em um conjunto de dados ocasiona o aumento exponencial da distância entre os pontos do espaço, causando sua dispersão. Esse fenômeno é prejudicial ao aprendizado de máquina, pois pode diminuir sua performance, aumentar o tempo de treinamento, dificultar a identificação de padrões e até causar o *overfitting* — quando um modelo se ajusta excessivamente aos dados de treinamento e não generaliza bem para novos dados.

Uma abordagem comum para mitigar esse problema é a aplicação de métodos de redução de dimensionalidade, que utilizam diferentes técnicas para diminuir o número de variáveis mantendo a maior quantidade possível de informação relevante.

Exercício 03

```
library(tibble)
library(reticulate)

D <- tibble( x_1 = rnorm(100,1,1),
x_2 = rnorm(100,-1,2),
y
= factor(sample(c("one","two","three"),100,replace = T)))
head(D)
```

```
## # A tibble: 6 x 3
##      x_1    x_2 y
##    <dbl>  <dbl> <fct>
## 1  0.779  0.250 three
## 2  1.61  -0.995 two
## 3  1.38   1.25  three
## 4  1.80   0.946 three
## 5 -0.300 -2.92  three
## 6  2.47   1.92  one
```

```
import pandas as pd
import math

df = pd.DataFrame(r.D)

def knn(k: int, x: tuple, D: pd.DataFrame) -> str:
  filteredDf = df.drop(['y'], axis=1, inplace=False)
  distancias_rank = []

  for i in range(len(df)):
    distancia = 0
    for index, j in enumerate(filteredDf.iloc[i]):
      distancia+=(x[index]-j)**2
    distanciaFinal = math.sqrt(distancia)
    distancias_rank.append((i, distanciaFinal))

  distancias_rank.sort(key=lambda e: e[1])

  vizinhos = []

  for i in range(k):
    vizinhos.append(distancias_rank[i][0])

  contagem = {"one": 0, "two": 0, "three": 0}

  for index in vizinhos:
    rotulo = df.iloc[index]['y']
    contagem[rotulo]+=1

  def rotulo_resultante(contagem):
    return max(contagem, key=contagem.get)
```

```
  return rotulo_resultante(contagem)

print(knn(10, (2,0), df))
```

```
## three
```

Exercício 04

Exercício 05 (Opcional)

Exercício 06 (Opcional)

Notas adicionais

- Todos os códigos desta lista foram realizados em Python 3.12.2;