Solução Lista 01

Nome: Caio Eduardo Curcio Valcazara
E-mail: caio.valcazara@aluno.ufabc.edu.br

26 fevereiro, 2025

Exercício 01: Faça uma pesquisa sobre aplicações práticas de aprendizado de máquinas e descreva aplicações de cada um dos problemas abaixo, incluindo a descrição dos vetores de características, rótulos ou respostas quando apropriado para cada um dos problemas:

a) Problema de classificação.

#Na classificação, o modelo prevê uma categoria ou classe baseada nos dados de entrada. Os rótulos são discretos.

#Uma aplicação prática do ml seria na área de cybersegurança com detecção de spams em email ou phishing.

#Vetores Característica: Palavras-chave ("promoção", "grátis", "clique aqui", "limitado"); Quantidade de anexo; Remetente; Formato do Email; Frequência de links.

#Rótulos ou classes de Saída: Spam (1); Não Spam(0)

b) Problema de regressão.

#Na regressão, o modelo prevê um valor numérico contínuo baseado nos dados de entrada.

#Uma aplicação prática para esse problema é da estimativa de life cycle de uma máquina ou equipamento eletrônicos como computadores e servidores.

#Vetores de Característica: Idade do dispositivo; Número de horas utilizado por dia; Temperatura mpedia da CPU e GPU; Taxa de uso dos componentes (disco, memória RAM); Histórico de sunstituição de componentes.

#Respostas: Tempo de Vida restante (em meses ou ano).

c) Problema de Agrupamento

#No agrupamento, o modelo identifica padrões ou grupos dentro dos dados sem rótulos prévios.

#Uma aplcaição desse problema está no ramo bancário com detecção de padrões incomuns em um conjunto de dados para identificar anomalias em transações fraudulentas.

#Vetores de Característica: Valores da transação; Localização geográfica; Horário da transação; Tipo de compra; Frequência de compra.

Exercício 02: Descreva com suas próprias palavras o que é a "maldição da dimensionalidade".

#A maldição da dimensionalidade nada mais é que o desafio de trabalhar com dados mais "complexos" o sentido de quantidade de suas variáveis ou características. Podemos traçar um paralelo com fractais, pois os fractais são padrões geométricos encontrados na natureza que se repetem infinitamente, logo cada vez que olhamos para um dado e queremos ser cada vez amis específico em analisá-lo para prevê-lo, caimos nessa maldição.

#A maldição nos exige amaior complexidade computacional, precisando de computadores mais potentes e códigos amis otimizados e quando chegamos no que chamamos de alta dimensão, a dificuldade de identificar relações aumenta.Essa seria a minha leitura e entendimento da maldição da dimensionalidade.

Exercício 03

```
#baixar o pacote dplyr com install.packages("dplyr")

library(dplyr)

knn_classify <- function(k, x, D) {
  # Calcula a distância euclidiana ao quadrado entre x e todos os pontos de D
  D2 <- D %>%
    mutate(dist = (x[1] - x_1)^2 + (x[2] - x_2)^2) %>%
    arrange(dist) %>%   # Ordena pelos vizinhos mais próximos
    head(k) %>%         # Seleciona os k vizinhos mais próximos
    count(y)            # Conta a frequência das classes

  # Retorna a classe mais frequente entre os vizinhos mais próximos
  return(D2$y[which.max(D2$n)])
}

# Testando a função
set.seed(42) # Para garantir reprodutibilidade
D <- tibble(
  x_1 = rnorm(100,1,1),
  x_2 = rnorm(100,-1,2),
  y = factor(sample(c("one", "two", "three"), 100, replace = TRUE))
)

x_test <- c(1, 2)
k_value <- 10

resultado <- knn_classify(k_value, x_test, D)
print(resultado)
```

```
## [1] three
## Levels: one three two
```

Exercício 04

```
library(tidyverse)
library(purrr)

# Função kNN otimizada (usa distância euclidiana real)
knn_classify <- function(k, x, D) {
  D %>%
    mutate(dist = sqrt((x[1] - x_1)^2 + (x[2] - x_2)^2)) %>%  # Distância euclidiana
    arrange(dist) %>%
    head(k) %>%
    count(y, sort = TRUE) %>%  # Já ordena por frequência da classe
    slice(1) %>%  # Pega a classe mais comum
    pull(y)
}

# Carregar e preparar o dataset iris
data("iris")
iris <- iris %>%
  as_tibble() %>%
  select(Petal.Length, Sepal.Length, Species) %>%
  rename(x_1 = Petal.Length, x_2 = Sepal.Length, y = Species)

# Função para calcular precisão do modelo kNN
test_knn_accuracy <- function(k, D) {
  sum(pmap_lgl(D, ~ knn_classify(k, c(..1, ..2), D) == ..3))  # Usa argumentos compactos
}

# Rodar testes para k = 10 e k = 1
accuracy_results <- tibble(
  k = c(10, 1),
  acertos = map_int(c(10, 1), ~ test_knn_accuracy(.x, iris))
)

# Exibir os resultados de forma bonita
print(accuracy_results)
```

```
## # A tibble: 2 x 2
##       k acertos
##   <dbl>   <int>
## 1    10     143
## 2     1     149
```