Universidade Federal do Rio de Janeiro
Escola Politécnica
Departamento de Engenharia Eletrônica e de Computação
Engenharia Eletrônica e de Computação
Álgebra Linear 2

TRABALHO DE ÁLGEBRA LINEAR 2

Aluna: Karen dos Anjos Arcoverde

Professor: Marcello Luiz Rodrigues de Campos

Rio de Janeiro
2021

Sumário

0 Introdução

0.1 Conteúdo

O relatório contém os resultados encontrados para cada questão passada pelo professor e o código final em linguagem Python.

0.2 Software e linguagem

O software usado para programação foi o Spyder e a linguagem foi o Python 3.8.5.

0.3 Biblioteca

A biblioteca utilizada para construir o código em Python foi:

- Numpy

0.4 Base de dados

O conjunto de dados "Iris"selecionado para o trabalho foi:

ESPÉCIES	DE	PARA
Iris-setosa	25	39
Iris-versicolor	75	89
Iris-virginica	125	139

Tabela 1: Base de dados "Iris"selecionada

1 Observações

A variável y da coluna PetalWidthCm foi escrita em função das outras variáveis x_1,x_2,x_3 das colunas SepalLengthCm,SepalWidthCm,PetalLengthCm, respectivamente. De forma que $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3$ sem o termo independente. Com o termo independente: $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3 + k$.

Além disso, a equação normal é: $x^T \cdot x \cdot w = x^T \cdot y$ ($R \cdot w = p$, $R = x^T \cdot x$ e $p = x^T \cdot y$), onde w é o vetor de coeficientes, o vetor y é a coluna PetalWidthCm e a matriz x é as colunas SepalLengthCm,SepalWidthCm,PetalLengthCm (se for com o termo independente, possui uma coluna a mais que só contém valores 1).

2 Questão 1

2.1 Resultados

2.1.1 Iris-Setosa

```
Iris-Setosa

SEM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3
[a b c] = [ 0.07455282 -0.06602361  0.03673264]

```

```
COM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3 + k
[a b c k] = [ 0.13497209 -0.07886596  0.10365958 -0.36144997]
```

2.1.2 Iris-Versicolor

```
Iris-Versicolor

SEM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3
[a b c] = [-0.14382683  0.17051618  0.40397714]

COM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3 + k
[a b c k] = [-0.06826027  0.24149193  0.39877688 -0.63863516]
```

2.1.3 Iris-Virginica

```
Iris-Virginica

SEM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3
[a b c] = [-0.11329721  0.3990124   0.25976859]

COM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3 + k
[a b c k] = [-0.11686468  0.35346044  0.22325742  0.36734017]
```

3 Questão 2

3.1 Resultados

3.1.1 Iris-Setosa

```
Iris-Setosa

R = VΛV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57]
 [255.73 172.5   75.51]
 [112.57  75.51  33.84]]

V =  [[-0.80618188 -0.45661854  0.37625828]
 [-0.54157827  0.31342349 -0.78003762]
 [-0.23825146  0.8326255   0.49997102]]

Λ =  [[5.86392634e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.29776824e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 7.47589571e-01]]
```

```

V^T =  [[-0.80618188 -0.54157827 -0.23825146]
 [-0.45661854  0.31342349  0.8326255 ]
 [ 0.37625828 -0.78003762  0.49997102]]


COM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57  75.5 ]
 [255.73 172.5   75.51  50.6 ]
 [112.57  75.51  33.84  22.4 ]
 [ 75.5   50.6   22.4   15.  ]]

V =  [[-0.79610972  0.1628056  -0.47792329 -0.33360603]
 [-0.53478061 -0.03864172  0.33762985  0.77364243]
 [-0.23529777  0.18572124  0.80860497 -0.50626137]
 [-0.15765144 -0.96825037  0.06126507 -0.18407563]]

Λ =  [[6.01336860e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 2.79352229e-02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.31947766e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 7.73257486e-01]]

V^T =  [[-0.79610972 -0.53478061 -0.23529777 -0.15765144]
 [ 0.1628056  -0.03864172  0.18572124 -0.96825037]
 [-0.47792329  0.33762985  0.80860497  0.06126507]
 [-0.33360603  0.77364243 -0.50626137 -0.18407563]]
```

3.1.2 Iris-Versicolor

```
Iris-Versicolor

R = VΛV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[555.38 256.64 397.59]
 [256.64 119.98 184.35]
 [397.59 184.35 286.5 ]]

V =  [[ 0.76039163  0.64759888 -0.0491961 ]
 [ 0.35223975 -0.47485674 -0.80649751]
 [ 0.54564799 -0.59592513  0.58918716]]

Λ =  [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.33162776e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 9.57976568e-01]]

V^T =  [[ 0.76039163  0.35223975  0.54564799]
 [ 0.64759888 -0.47485674 -0.59592513]
 [-0.0491961  -0.80649751  0.58918716]]


COM O TERMO INDEPENDENTE:
```

```
R =  [[555.38 256.64 397.59  91.  ]
 [256.64 119.98 184.35  42.2 ]
 [397.59 184.35 286.5   65.2 ]
 [ 91.    42.2   65.2   15.  ]]

V =  [[-0.7545525   0.11812875  0.64474223  0.03167953]
 [-0.34954066  0.11526607 -0.46962259  0.80248968]
 [-0.54144512 -0.01367345 -0.60234554 -0.58637025]
 [-0.1237297  -0.98619084  0.030691    0.1057197 ]]

Λ =  [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 7.14132919e-02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.33280153e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]

V^T =  [[-0.7545525  -0.34954066 -0.54144512 -0.1237297 ]
 [ 0.11812875  0.11526607 -0.01367345 -0.98619084]
 [ 0.64474223 -0.46962259 -0.60234554  0.030691  ]
 [ 0.03167953  0.80248968 -0.58637025  0.1057197 ]]
```

3.1.3 Iris-Virginica

```
Iris-Virginica

R = VΛV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72]
 [304.85 138.7  253.94]
 [562.72 253.94 469.3 ]]

V =  [[-0.72592866 -0.63483621  0.2645951 ]
 [-0.32772296 -0.01894783 -0.94458385]
 [-0.60466953  0.77241438  0.19429563]]

Λ =  [[1.28249881e+03 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.79253032e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.07193335e+00]]

V^T =  [[-0.72592866 -0.32772296 -0.60466953]
 [-0.63483621 -0.01894783  0.77241438]
 [ 0.2645951  -0.94458385  0.19429563]]


COM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72 100.3 ]
 [304.85 138.7  253.94  45.4 ]
 [562.72 253.94 469.3   83.6 ]
 [100.3   45.4   83.6   15.  ]]

V =  [[-7.21743001e-01 -2.76297507e-01  6.34623278e-01 -1.50933108e-04]
 [-3.25843186e-01  9.35769444e-01  3.68031853e-02 -1.29642937e-01]
```

```
 [-6.01187515e-01 -1.93722166e-01 -7.68083597e-01 -1.05322749e-01]
 [-1.07176625e-01  1.02308151e-01 -7.71129602e-02  9.85951218e-01]]

Λ =  [[1.29740071e+03 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.08243546e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.82311582e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 8.45463191e-02]]

V^T =  [[-7.21743001e-01 -3.25843186e-01 -6.01187515e-01 -1.07176625e-01]
 [-2.76297507e-01  9.35769444e-01 -1.93722166e-01  1.02308151e-01]
 [ 6.34623278e-01  3.68031853e-02 -7.68083597e-01 -7.71129602e-02]
 [-1.50933108e-04 -1.29642937e-01 -1.05322749e-01  9.85951218e-01]]
```

4 Questão 3

4.1 Resultados

4.1.1 Iris-Setosa

```
Iris-Setosa

R = UΣV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57]
 [255.73 172.5   75.51]
 [112.57  75.51  33.84]]

U =  [[ 0.80618188 -0.45661854 -0.37625828]
 [ 0.54157827  0.31342349  0.78003762]
 [ 0.23825146  0.8326255  -0.49997102]]

Σ =  [[5.86392634e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.29776824e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 7.47589571e-01]]

V^T =  [[ 0.80618188  0.54157827  0.23825146]
 [-0.45661854  0.31342349  0.8326255 ]
 [-0.37625828  0.78003762 -0.49997102]]


COM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57  75.5 ]
 [255.73 172.5   75.51  50.6 ]
 [112.57  75.51  33.84  22.4 ]
 [ 75.5   50.6   22.4   15.  ]]

U =  [[ 0.79610972 -0.33360603 -0.47792329 -0.1628056 ]
 [ 0.53478061  0.77364243  0.33762985  0.03864172]
 [ 0.23529777 -0.50626137  0.80860497 -0.18572124]
 [ 0.15765144 -0.18407563  0.06126507  0.96825037]]
```

```

Σ =  [[6.01336860e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 7.73257486e-01 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.31947766e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 2.79352230e-02]]

V^T =  [[ 0.79610972  0.53478061  0.23529777  0.15765144]
 [-0.33360603  0.77364243 -0.50626137 -0.18407563]
 [-0.47792329  0.33762985  0.80860497  0.06126507]
 [-0.1628056   0.03864172 -0.18572124  0.96825037]]
```

4.1.2 Iris-Versicolor

```
Iris-Versicolor

R = UΣV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[555.38 256.64 397.59]
 [256.64 119.98 184.35]
 [397.59 184.35 286.5 ]]

U =  [[ 0.76039163  0.64759888 -0.0491961 ]
 [ 0.35223975 -0.47485674 -0.80649751]
 [ 0.54564799 -0.59592513  0.58918716]]

Σ =  [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.33162776e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 9.57976568e-01]]

V^T =  [[ 0.76039163  0.35223975  0.54564799]
 [ 0.64759888 -0.47485674 -0.59592513]
 [-0.0491961  -0.80649751  0.58918716]]


COM O TERMO INDEPENDENTE:
R =  [[555.38 256.64 397.59  91.  ]
 [256.64 119.98 184.35  42.2 ]
 [397.59 184.35 286.5   65.2 ]
 [ 91.    42.2   65.2   15.  ]]

U =  [[ 0.7545525  -0.64474223 -0.11812875  0.03167953]
 [ 0.34954066  0.46962259 -0.11526607  0.80248968]
 [ 0.54144512  0.60234554  0.01367345 -0.58637025]
 [ 0.1237297  -0.030691    0.98619084  0.1057197 ]]

Σ =  [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.33280153e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 7.14132919e-02 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]

V^T =  [[ 0.7545525   0.34954066  0.54144512  0.1237297 ]
```

```
 [-0.64474223  0.46962259  0.60234554 -0.030691  ]
 [-0.11812875 -0.11526607  0.01367345  0.98619084]
 [ 0.03167953  0.80248968 -0.58637025  0.1057197 ]]
```

4.1.3 Iris-Virginica

```
Iris-Virginica

R = UΣV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72]
 [304.85 138.7  253.94]
 [562.72 253.94 469.3 ]]

U =  [[ 0.72592866 -0.63483621 -0.2645951 ]
 [ 0.32772296 -0.01894783  0.94458385]
 [ 0.60466953  0.77241438 -0.19429563]]

Σ =  [[1.28249881e+03 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.79253032e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.07193335e+00]]

V^T =  [[ 0.72592866  0.32772296  0.60466953]
 [-0.63483621 -0.01894783  0.77241438]
 [-0.2645951   0.94458385 -0.19429563]]


COM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72 100.3 ]
 [304.85 138.7  253.94  45.4 ]
 [562.72 253.94 469.3   83.6 ]
 [100.3   45.4   83.6   15.  ]]

U =  [[ 7.21743001e-01 -2.76297507e-01 -6.34623278e-01 -1.50933128e-04]
 [ 3.25843186e-01  9.35769444e-01 -3.68031855e-02 -1.29642937e-01]
 [ 6.01187515e-01 -1.93722166e-01  7.68083597e-01 -1.05322749e-01]
 [ 1.07176625e-01  1.02308151e-01  7.71129601e-02  9.85951218e-01]]

Σ =  [[1.29740071e+03 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.08243546e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.82311582e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 8.45463191e-02]]

V^T =  [[ 7.21743001e-01  3.25843186e-01  6.01187515e-01  1.07176625e-01]
 [-2.76297507e-01  9.35769444e-01 -1.93722166e-01  1.02308151e-01]
 [-6.34623278e-01 -3.68031855e-02  7.68083597e-01  7.71129601e-02]
 [-1.50933128e-04 -1.29642937e-01 -1.05322749e-01  9.85951218e-01]]
```

5 Questão 4

5.1 Resultados

```
SEM O TERMO INDEPENDENTE:
A =  Iris-Versicolor
B =  Iris-Setosa
C =  Iris-Setosa
D =  Iris-Versicolor
E =  Iris-Virginica

COM O TERMO INDEPENDENTE:
A =  Iris-Versicolor
B =  Iris-Setosa
C =  Iris-Setosa
D =  Iris-Versicolor
E =  Iris-Virginica
```

6 Código

```
# Programa codigo.py
# Autora: Karen dos Anjos Arcoverde
# Data: 06/02/2021
#


import numpy as np
import sys



############################## Funcoes ##############################
def pegarDados(tipo_iris):

    # tipo_iris = 1 Setosa  , tipo_iris = 2  Versicolor,
    # tipo_iris = 3 Virginica
    dados = []
    IDs = []

    Setosa = range(25,39+1)
    Versicolor = range(75,89+1)
    Virginica = range (125,139+1)

    arquivo= open("dados_13.csv",'r')
    arquivo.readline() # ignora a primeira linha

    if (tipo_iris == 1):
        IDs = Setosa
    if (tipo_iris == 2):
        IDs = Versicolor
    if (tipo_iris == 3):
```

```
        IDs = Virginica

    for i in range(1,46): # percorre todo o banco de dados 1-45
        linha = (arquivo.readline()).split(',') #separa os dados por virgula


        if  (int(linha[0]) in IDs):# percorre os ids selecionados
            linha.pop(0) # retira o ID dos dados
            linha.pop(-1) # retira a especie dos dados
            for j in range(4): #para cada dado
                linha[j] = float(linha[j]) #transforma em numero

            dados.append(linha) #adiciona na lista de dados

    arquivo.close()
    return dados

# ------------------------------------------------------------
def construir_equacao_normal (dados):

    #equacao normal - minimos quadrados:
    # (x^T).x.w = (x^T).y
    #(x_transposta).x.w =(x_transposta).y
    # R.w = p
    # R = (x^T).x
    # p = (x^T).y

    ###### sem termo independente #####
    # y = a*x1 +b*x2 +c*x3
    x = []
    y = []

    #achar (x^T) e x
    # x - colunas SepalLengthCm,SepalWidthCm,PetalLengthCm
    x = np.array(dados)
    x = np.delete(x.reshape(15,4),3,1) #deleta a ultima coluna de x
    x_transposta = np.transpose(x) #faz a transposta de x: (x^T)

    #achar R
    R = np.dot(x_transposta,x) #multiplica (x^T) por x

    #achar y - coluna PetalWidthCm
    y = np.array(dados)
    y = np.delete(y.reshape(15,4),0,1) #deleta a primeira coluna de y
    y = np.delete(y.reshape(15,3),0,1) #deleta a segunda coluna de y
    y = np.delete(y.reshape(15,2),0,1) #deleta a terceira coluna de y

    #achar p
    p = np.dot(x_transposta,y)

    ###### com termo independente #####
    # y = a*x1 +b*x2 +c*x3 +k
```

```
84
85      for i in range (0,15):
86          dados[i][3] = 1
87
88      #achar (x^T) e x
89      x = np.array(dados)
90      x_transposta = np.transpose(x) #faz a transposta de x: (x^T)
91      #achar R
92      R1= np.dot(x_transposta,x) #multiplica (x^T) por x
93      #achar p
94      p1 = np.dot(x_transposta,y)
95
96      return R,p,R1,p1
97
98  # ------------------------------------------------------------
99  def PLU(R,p):
100     indice_coluna = 0
101     indice_linha = 1
102     indice_L_1s = 0
103     tamanho_R = len(R)
104
105     while (indice_coluna < tamanho_R):
106         #constroi a matriz L, triangular inferior
107         if (R[indice_coluna][indice_coluna] != 0):
108             L = np.zeros((tamanho_R, tamanho_R))
109
110             indice_coluna_aux = indice_coluna + 1
111             while (indice_coluna_aux < tamanho_R):
112                 L[indice_coluna_aux][indice_coluna] = - R[indice_coluna_aux][
                        indice_coluna]/R[indice_coluna][indice_coluna]
113                 indice_linha += 1
114                 indice_coluna_aux += 1
115
116             while (indice_L_1s < tamanho_R):
117                 L[indice_L_1s][indice_L_1s] = 1
118                 indice_L_1s += 1
119
120             indice_L_1s = 0
121             indice_linha = 1
122
123             #multiplica a matriz R por L
124             R = np.dot (L,R)
125             p = np.dot (L,p)
126
127
128         #se o pivo for zero, necessario multiplicar por uma matriz P de
               permutacao
129         if (R[indice_coluna][indice_coluna] == 0):
130             P = np.zeros((tamanho_R, tamanho_R))
131             P[indice_coluna][indice_coluna + 1] = 1
132             P[indice_coluna + 1][indice_coluna] = 1
133
```

```
            i = 0
            j = 0
            guarda_1 = False
            while (i < tamanho_R):
                while (j < tamanho_R):
                    if (P[i][j] == 1):
                        guarda_1 = True
                        j += 1
                if (guarda_1 == False):
                    P[i][i] = 1
                i += 1

            R = np.dot(P,R)
            p = np.dot (P,p)

        indice_coluna += 1

    #backsubstitution
    w = np.linalg.solve(R, p)

    return w

# ------------------------------------------------------------
def decomposicao_espectral(R):

    # R = VDV^(T)
    #determinando autovalores e autovetores
    autovalores, autovetores = np.linalg.eig(R)
    # matriz diagonal de autovalores
    matrizDiagonal = np.diag(autovalores)

    return  autovetores, matrizDiagonal

# ------------------------------------------------------------
def contruir_svd (R):
    s = []
    i = 0
    j = 0

    # R = U.s.VT
    #determinando autovalores e autovetores
    ### U
    ## autovetores de R.(R^T)
    autovaloresU, autovetoresU = np.linalg.eig(np.dot(R,np.transpose(R)))

    ### V
    ## autovetores de (R^T).R
    autovaloresV, autovetoresV = np.linalg.eig(np.dot(np.transpose(R),R))

    tamanho_autovalores_U = autovaloresU.shape
    tamanho_autovalores_V = autovaloresV.shape

```

```
    while (i < tamanho_autovalores_U [0]):
        while (j < tamanho_autovalores_V [0]):
            if (autovaloresU [i] == autovaloresV [j]):
                s.append(np.sqrt(autovaloresU [i]))
            elif (autovaloresV [j] == 0):
                s.append(0)
            j += 1

        if (autovaloresU [i] == 0):
            s.append(0)

        j = 0
        i += 1

    return  autovetoresU, np.array(s),np.transpose(autovetoresV)

# -------------------------------------------------------------
def estimar_amostras(amostra, w_setosa,w_versicolor,w_virginica,c_independente
    ):

    lista_erros = []
    estimativa = ""
    indice = 0
    amostra_x = []
    indice_erros_modulo = 0

    while (indice < 3):
        amostra_x.append (amostra[indice])
        indice += 1

    if (c_independente == True):
        amostra_x.append(1)

    amostra_x = np.array(amostra_x)

    #produto interno <x,y> = (x^T).y

    estimativa_setosa = np.dot(amostra_x,w_setosa)
    estimativa_versicolor = np.dot(amostra_x,w_versicolor)
    estimativa_virginica = np.dot(amostra_x,w_virginica)

    erro_setosa = estimativa_setosa[0] - amostra[3]
    lista_erros.append(erro_setosa)

    erro_versicolor = estimativa_versicolor[0] - amostra[3]
    lista_erros.append(erro_versicolor)

    erro_virginica = estimativa_virginica[0] - amostra[3]
    lista_erros.append(erro_virginica)

    while (indice_erros_modulo < len(lista_erros)):
        lista_erros [indice_erros_modulo] = abs(lista_erros[indice_erros_modulo
```

```
          ])
        indice_erros_modulo += 1

    if (min(lista_erros) == abs(erro_setosa)):
        estimativa = "Iris-Setosa"

        return estimativa

    if (min(lista_erros) == abs(erro_versicolor)):
        estimativa = "Iris-Versicolor"

        return estimativa

    if (min(lista_erros) == abs(erro_virginica)):
        estimativa = "Iris-Virginica"

        return estimativa

#################### Programa Principal ####################
def menu():
    resultado = 0
    tipo_iris = 0
    coeficientes_sem_aux = []
    coeficientes_com_aux = []
    indice = 0


    while (resultado != 5):
        print()
        print('###################### MENU PRINCIPAL
           ##############################')
        print("Digite somente o numero da questao que voce deseja ver o
           resultado: ")
        print("1 = Questao 1")
        print("2 = Questao 2")
        print("3 = Questao 3")
        print("4 = Questao 4")
        print("5 = SAIR")
        print('
           ######################################################################
           ')
        print()

        resultado = int(input())

        if (resultado == 5):
            sys.exit(0)

        if (resultado == 1):

            while (tipo_iris != 4):
                print('###################### MENU IRIS
```

```
                ##########################')
284             print("Digite qual especie voce deseja fazer a regressao
                linear: ")
285             print("1 = Iris-Setosa")
286             print("2 = Iris-Versicolor")
287             print("3 = Iris Virginica")
288             print("4 = VOLTAR AO MENU PRINCIPAL")
289             print('
                ############################################################')
290             print()
291
292             tipo_iris = int(input())
293
294             if (tipo_iris == 4):
295                 menu()
296
297             else:
298                 dados = pegarDados (tipo_iris)
299                 R,p,R1,p1 = construir_equacao_normal(dados)
300                 w = PLU(R,p)
301                 w1 = PLU(R1,p1)
302
303                 print()
304                 if  (tipo_iris == 1):
305                     print("Iris-Setosa\n")
306                 elif (tipo_iris == 2):
307                     print("Iris-Versicolor\n")
308                 elif (tipo_iris == 3):
309                     print("Iris-Virginica\n")
310
311                 print("SEM O TERMO INDEPENDENTE: ")
312                 print("y = a*x1 + b*x2 + c*x3")
313                 print("[a b c] = ",end="")
314
315
316                 while (indice < len(w)):
317                     coeficientes_sem_aux.append(w[indice][0])
318                     coeficientes_sem = np.array(coeficientes_sem_aux)
319                     indice += 1
320
321                 print(coeficientes_sem)
322                 coeficientes_sem_aux = []
323
324
325                 print()
326
327                 print("COM O TERMO INDEPENDENTE: ")
328                 print("y = a*x1 + b*x2 + c*x3 + k")
329                 print("[a b c k] = ",end="")
330
331                 indice = 0
332                 while (indice < len(w1)):
```

```
                        coeficientes_com_aux.append(w1[indice][0])
                        coeficientes_com = np.array(coeficientes_com_aux)
                        indice += 1

                    print(coeficientes_com)
                    coeficientes_com_aux = []
                    indice = 0

                    print()

        if (resultado == 2):

            while (tipo_iris != 4):
                print('###################### MENU IRIS
                    ##############################')
                print("Digite qual especie voce deseja fazer a decomposicao
                    espectral: ")
                print("1 = Iris-Setosa")
                print("2 = Iris-Versicolor")
                print("3 = Iris Virginica")
                print("4 = VOLTAR AO MENU PRINCIPAL")
                print('
                    ############################################################
                    ')
                print()


                tipo_iris = int(input())

                if (tipo_iris == 4):
                    menu()

                else:
                    dados = pegarDados (tipo_iris)
                    R,p,R1,p1 = construir_equacao_normal(dados)
                    autovetores, matrizDiagonal = decomposicao_espectral(R)
                    autovetores1, matrizDiagonal1 = decomposicao_espectral(R1)

                    if  (tipo_iris == 1):
                        print("Iris-Setosa\n")
                    elif (tipo_iris == 2):
                        print("Iris-Versicolor\n")
                    elif (tipo_iris == 3):
                        print("Iris-Virginica\n")

                    print("R = V\u039BV^(T)")
                    print()


                    print("SEM O TERMO INDEPENDENTE: ")
                    print("R = ", R)
                    print()
```

```
                    print("V = ",autovetores)
                    print()
                    print('\u039B = ',matrizDiagonal)
                    print()
                    print("V^T = ",np.transpose(autovetores))

                    print()
                    print()

                    print("COM O TERMO INDEPENDENTE: ")
                    print("R = ", R1)
                    print()
                    print("V = ", autovetores1)
                    print()
                    print('\u039B = ', matrizDiagonal1)
                    print()
                    print("V^T = ", np.transpose(autovetores1))

                    print()

        if (resultado == 3):

            while (tipo_iris != 4):
                print('###################### MENU IRIS #############')
                print("Digite qual especie voce deseja fazer o SVD: ")
                print("1 = Iris-Setosa")
                print("2 = Iris-Versicolor")
                print("3 = Iris Virginica")
                print("4 = VOLTAR AO MENU PRINCIPAL")
                print('##############################################')
                print()

                tipo_iris = int(input())

                if (tipo_iris == 4):
                    menu()

                else:

                    dados = pegarDados (tipo_iris)
                    R,p,R1,p1 = construir_equacao_normal(dados)
                    U, s, VT = contruir_svd(R)
                    U1, s1, VT1 = contruir_svd(R1)

                    if  (tipo_iris == 1):
                         print("Iris-Setosa\n")
                    elif (tipo_iris == 2):
                         print("Iris-Versicolor\n")
                    elif (tipo_iris == 3):
                         print("Iris-Virginica\n")

                    print("R = U\u03A3V^(T)")
```

```
                print()

                print("SEM O TERMO INDEPENDENTE: ")
                print("R = ", R)
                print()
                print("U = ", U)
                print()
                print('\u03A3 = ', np.diag(s))
                print()
                print("V^T = ", VT)
                print()
                print()

                print("COM O TERMO INDEPENDENTE: ")
                print("R = ", R1)
                print()
                print("U = ", U1)
                print()
                print('\u03A3 = ', np.diag(s1))
                print()
                print("V^T = ", VT1)

                print()



    if (resultado == 4):
        A = [5.0,2.3,3.3,1.0]
        B = [4.6,3.2,1.4,0.2]
        C = [5.0,3.3,1.4,0.2]
        D = [6.1,3.0,4.6,1.4]
        E = [5.9,3.0,5.1,1.8]

        print("SEM O TERMO INDEPENDENTE: ")
        dados = pegarDados (1)
        R,p,R1,p1 = construir_equacao_normal(dados)
        w_setosa = PLU(R,p)

        dados = pegarDados (2)
        R,p,R1,p1 = construir_equacao_normal(dados)
        w_versicolor = PLU(R,p)

        dados = pegarDados (3)
        R,p,R1,p1 = construir_equacao_normal(dados)
        w_virginica = PLU(R,p)

        c_independente = False
        estimativa = estimar_amostras(A,w_setosa,w_versicolor,w_virginica,
            c_independente)
        print("A = ", estimativa)
        estimativa = estimar_amostras(B,w_setosa,w_versicolor,w_virginica,
            c_independente)
```

```
            print("B = ", estimativa)
            estimativa = estimar_amostras(C,w_setosa,w_versicolor,w_virginica,
               c_independente)
            print("C = ", estimativa)
            estimativa = estimar_amostras(D,w_setosa,w_versicolor,w_virginica,
               c_independente)
            print("D = ", estimativa)
            estimativa = estimar_amostras(E,w_setosa,w_versicolor,w_virginica,
               c_independente)
            print("E = ", estimativa)

            print()
            print("COM O TERMO INDEPENDENTE: ")
            dados = pegarDados (1)
            R,p,R1,p1 = construir_equacao_normal(dados)
            w1_setosa = PLU(R1,p1)

            dados = pegarDados (2)
            R,p,R1,p1 = construir_equacao_normal(dados)
            w1_versicolor = PLU(R1,p1)

            dados = pegarDados (3)
            R,p,R1,p1 = construir_equacao_normal(dados)
            w1_virginica = PLU(R1,p1)

            c_independente = True
            estimativa1 = estimar_amostras(A,w1_setosa,w1_versicolor,
               w1_virginica,c_independente)
            print("A = ", estimativa1)
            estimativa1 = estimar_amostras(B,w1_setosa,w1_versicolor,
               w1_virginica,c_independente)
            print("B = ", estimativa1)
            estimativa1 = estimar_amostras(C,w1_setosa,w1_versicolor,
               w1_virginica,c_independente)
            print("C = ", estimativa1)
            estimativa1 = estimar_amostras(D,w1_setosa,w1_versicolor,
               w1_virginica,c_independente)
            print("D = ", estimativa1)
            estimativa1 = estimar_amostras(E,w1_setosa,w1_versicolor,
               w1_virginica,c_independente)
            print("E = ", estimativa1)

            menu()

######## chamada ao menu
menu()
```

7 Bibliografia

- https://algebralinearufcg.github.io/jup-not/prog02-learning-numpy.html

- https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/
- https://pt.coredump.biz/questions/34007632/how-to-remove-a-column-in-a-numpy-array
- https://pythonforundergradengineers.com/unicode-characters-in-python.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.svd.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.diagsvd.html#scipy.linalg.diagsvd
- https://www.geeksforgeeks.org/python-exit-commands-quit-exit-sys-exit-and-os-_exit/