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.

0.3 Bibliotecas

As bibliotecas utilizadas para construir o código em Python foram:

- Numpy
- Scipy

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: $x^T \cdot x \cdot w = x^T \cdot y$, 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:
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:
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:
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:
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:
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:
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:
U =  [[-0.80618188  0.37625828 -0.45661854]
 [-0.54157827 -0.78003762  0.31342349]
 [-0.23825146  0.49997102  0.8326255 ]]

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

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


COM O TERMO INDEPENDENTE:
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 7.73257486e-01 5.31947766e-01 2.79352229e-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:
U =  [[-0.76039163  0.64759888 -0.0491961 ]
 [-0.35223975 -0.47485674 -0.80649751]
 [-0.54564799 -0.59592513  0.58918716]]

Σ =  [9.59570396e+02 1.33162776e+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:
U =  [[-0.7545525   0.64474223  0.03167953 -0.11812875]
 [-0.34954066 -0.46962259  0.80248968 -0.11526607]
 [-0.54144512 -0.60234554 -0.58637025  0.01367345]
 [-0.1237297   0.030691    0.1057197   0.98619084]]
```

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

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

4.1.3 Iris-Virginica

```
Iris-Virginica

R = UΣV^(T)

SEM O TERMO INDEPENDENTE:
U =  [[-0.72592866  0.2645951  -0.63483621]
 [-0.32772296 -0.94458385 -0.01894783]
 [-0.60466953  0.19429563  0.77241438]]

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

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


COM O TERMO INDEPENDENTE:
U =  [[-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]]

Σ =  [1.29740071e+03 1.08243546e+00 5.82311582e-01 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]]
```

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
from scipy.linalg import svd


############################## 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

    for i in range (0,15):
        dados[i][3] = 1

    #achar (x^T) e x
    x = np.array(dados)
    x_transposta = np.transpose(x) #faz a transposta de x: (x^T)
    #achar R
```

```
    R1= np.dot(x_transposta,x) #multiplica (x^T) por x
    #achar p
    p1 = np.dot(x_transposta,y)

    return R,p,R1,p1

# ------------------------------------------------------------
def PLU(R,p):
    indice_coluna = 0
    indice_linha = 1
    indice_L_1s = 0
    tamanho_R = len(R)

    while (indice_coluna < tamanho_R):
        #constroi a matriz L, triangular inferior
        if (R[indice_coluna][indice_coluna] != 0):
            L = np.zeros((tamanho_R, tamanho_R))

            indice_coluna_aux = indice_coluna + 1
            while (indice_coluna_aux < tamanho_R):
                L[indice_coluna_aux][indice_coluna] = - R[
                   indice_coluna_aux][indice_coluna]/R[indice_coluna][
                   indice_coluna]
                indice_linha += 1
                indice_coluna_aux += 1

            while (indice_L_1s < tamanho_R):
                L[indice_L_1s][indice_L_1s] = 1
                indice_L_1s += 1

            indice_L_1s = 0
            indice_linha = 1

            #multiplica a matriz R por L
            R = np.dot (L,R)
            p = np.dot (L,p)


        #se o pivo for zero, necessario multiplicar por uma matriz P de
           permutacao
        if (R[indice_coluna][indice_coluna] == 0):
            P = np.zeros((tamanho_R, tamanho_R))
            P[indice_coluna][indice_coluna + 1] = 1
            P[indice_coluna + 1][indice_coluna] = 1

            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 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)
```

```
189
190     erro_setosa = estimativa_setosa[0] - amostra[3]
191     lista_erros.append(erro_setosa)
192
193     erro_versicolor = estimativa_versicolor[0] - amostra[3]
194     lista_erros.append(erro_versicolor)
195
196     erro_virginica = estimativa_virginica[0] - amostra[3]
197     lista_erros.append(erro_virginica)
198
199     while (indice_erros_modulo < len(lista_erros)):
200         lista_erros [indice_erros_modulo] = abs(lista_erros[
              indice_erros_modulo])
201         indice_erros_modulo += 1
202
203     if (min(lista_erros) == abs(erro_setosa)):
204         estimativa = "Iris-Setosa"
205
206         return estimativa
207
208     if (min(lista_erros) == abs(erro_versicolor)):
209         estimativa = "Iris-Versicolor"
210
211         return estimativa
212
213     if (min(lista_erros) == abs(erro_virginica)):
214         estimativa = "Iris-Virginica"
215
216         return estimativa
217
218 ################### Programa Principal ###################
219 def menu():
220     resultado = ""
221     tipo_iris = ""
222     coeficientes_sem_aux = []
223     coeficientes_com_aux = []
224     indice = 0
225
226     print()
227     print("Digite somente o numero da questao que voce deseja ver o
          resultado: ")
228     print("Questao 1")
229     print("Questao 2")
230     print("Questao 3")
231     print("Questao 4")
232     print()
233     print("PARA SAIR DIGITE 0")
234
235     while (resultado != '0'):
236         resultado = input()
237
```

```
238         if (resultado == '1'):
239             print("Digite qual especie voce deseja fazer a regressao
                    linear: ")
240             print("1 = Iris-Setosa")
241             print("2 = Iris-Versicolor")
242             print("3 = Iris Virginica")
243             print()
244             print("PARA SAIR DIGITE 0")
245 
246             while (tipo_iris != '0'):
247 
248                 tipo_iris = input()
249 
250                 if (tipo_iris == '0'):
251                     break
252 
253                 dados = pegarDados (tipo_iris)
254                 R,p,R1,p1 = construir_equacao_normal(dados)
255                 w = PLU(R,p)
256                 w1 = PLU(R1,p1)
257 
258                 print()
259                 if  (tipo_iris == '1'):
260                     print("Iris-Setosa\n")
261                 elif (tipo_iris == '2'):
262                     print("Iris-Versicolor\n")
263                 elif (tipo_iris == '3'):
264                     print("Iris-Virginica\n")
265 
266                 print("SEM O TERMO INDEPENDENTE: ")
267                 print("y = a*x1 + b*x2 + c*x3")
268                 print("[a b c] = ",end="")
269 
270 
271                 while (indice < len(w)):
272                     coeficientes_sem_aux.append(w[indice][0])
273                     coeficientes_sem = np.array(coeficientes_sem_aux)
274                     indice += 1
275 
276                 print(coeficientes_sem)
277                 coeficientes_sem_aux = []
278 
279 
280                 print()
281 
282                 print("COM O TERMO INDEPENDENTE: ")
283                 print("y = a*x1 + b*x2 + c*x3 + k")
284                 print("[a b c k] = ",end="")
285 
286                 indice = 0
287                 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



        if (resultado == '2'):
            print("Digite qual especie voce deseja fazer a decomposicao
               espectral: ")
            print("1 = Iris-Setosa")
            print("2 = Iris-Versicolor")
            print("3 = Iris Virginica")
            print()
            print("PARA SAIR DIGITE 0")

            while (tipo_iris != '0'):

                tipo_iris = input()

                if (tipo_iris == '0'):
                    break

                print()

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

                print()
```

```
                print()

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



        if (resultado == '3'):
            print("Digite qual especie voce deseja fazer o SVD: ")
            print("1 = Iris-Setosa")
            print("2 = Iris-Versicolor")
            print("3 = Iris Virginica")
            print()
            print("PARA SAIR DIGITE 0")

            while (tipo_iris != '0'):
                tipo_iris = input()

                if (tipo_iris == '0'):
                    break

                print()

                dados = pegarDados (tipo_iris)
                R,p,R1,p1 = construir_equacao_normal(dados)
                U, s, VT = svd(R)
                U1, s1, VT1 = 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("U = ", U)
                print()
                print('\u03A3 = ', s)
                print()
                print("V^T = ", VT)

                print()
                print()
```

```
389
390             print("COM O TERMO INDEPENDENTE: ")
391             print("U = ", U1)
392             print()
393             print('\u03A3 = ', s1)
394             print()
395             print("V^T = ", VT1)
396
397         if (resultado == '4'):
398             A = [5.0,2.3,3.3,1.0]
399             B = [4.6,3.2,1.4,0.2]
400             C = [5.0,3.3,1.4,0.2]
401             D = [6.1,3.0,4.6,1.4]
402             E = [5.9,3.0,5.1,1.8]
403
404             print("SEM O TERMO INDEPENDENTE: ")
405             dados = pegarDados ('1')
406             R,p,R1,p1 = construir_equacao_normal(dados)
407             w_setosa = PLU(R,p)
408
409             dados = pegarDados ('2')
410             R,p,R1,p1 = construir_equacao_normal(dados)
411             w_versicolor = PLU(R,p)
412
413             dados = pegarDados ('3')
414             R,p,R1,p1 = construir_equacao_normal(dados)
415             w_virginica = PLU(R,p)
416
417             c_independente = False
418             estimativa = estimar_amostras(A,w_setosa,w_versicolor,
                    w_virginica,c_independente)
419             print("A = ", estimativa)
420             estimativa = estimar_amostras(B,w_setosa,w_versicolor,
                    w_virginica,c_independente)
421             print("B = ", estimativa)
422             estimativa = estimar_amostras(C,w_setosa,w_versicolor,
                    w_virginica,c_independente)
423             print("C = ", estimativa)
424             estimativa = estimar_amostras(D,w_setosa,w_versicolor,
                    w_virginica,c_independente)
425             print("D = ", estimativa)
426             estimativa = estimar_amostras(E,w_setosa,w_versicolor,
                    w_virginica,c_independente)
427             print("E = ", estimativa)
428
429             print()
430             print("COM O TERMO INDEPENDENTE: ")
431             dados = pegarDados ('1')
432             R,p,R1,p1 = construir_equacao_normal(dados)
433             w1_setosa = PLU(R1,p1)
434
```

```
            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)

######## 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