

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

<b>0</b>	<b>Introdução</b>	<b>3</b>
0.1	Conteúdo . . . . .	3
0.2	Software e linguagem . . . . .	3
0.3	Biblioteca/Módulo . . . . .	3
0.4	Base de dados . . . . .	3
<b>1</b>	<b>Explicação e Código</b>	<b>3</b>
1.1	Explicação do código . . . . .	3
1.1.1	Menu . . . . .	3
1.1.2	Funções construídas . . . . .	4
1.1.3	pegarDados . . . . .	4
1.1.4	construir_equacao_normal . . . . .	4
1.1.5	PLU . . . . .	4
1.1.6	decomposicao_espectral . . . . .	5
1.1.7	construir_svd . . . . .	5
1.1.8	estimar_amstras . . . . .	6
1.2	Código . . . . .	6
<b>2</b>	<b>Questão 1</b>	<b>17</b>
2.1	Funções utilizadas . . . . .	17
2.2	Resultados . . . . .	17
2.2.1	Iris-Setosa . . . . .	17
2.2.2	Iris-Versicolor . . . . .	17
2.2.3	Iris-Virginica . . . . .	17
<b>3</b>	<b>Questão 2</b>	<b>18</b>
3.1	Funções utilizadas . . . . .	18
3.2	Resultados . . . . .	18
3.2.1	Iris-Setosa . . . . .	18
3.2.2	Iris-Versicolor . . . . .	19
3.2.3	Iris-Virginica . . . . .	19
<b>4</b>	<b>Questão 3</b>	<b>20</b>
4.1	Funções utilizadas . . . . .	20
4.2	Resultados . . . . .	21
4.2.1	Iris-Setosa . . . . .	21
4.2.2	Iris-Versicolor . . . . .	21
4.2.3	Iris-Virginica . . . . .	22
<b>5</b>	<b>Questão 4</b>	<b>23</b>
5.1	Funções utilizadas . . . . .	23
5.2	Resultados . . . . .	23
<b>6</b>	<b>Bibliografia</b>	<b>24</b>

# 0 Introdução

## 0.1 Conteúdo

O relatório contém a explicação do código, o código final, as funções utilizadas e os resultados encontrados para cada questão passada pelo professor.

## 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/Módulo

A biblioteca e o módulo utilizados para construir o código em Python foi:

- Numpy - É usada para fazer as operações com as matrizes
- Sys - É um módulo que faz parte da biblioteca padrão do Python e foi usado o "sys.exit (0)" para sair do programa quando o usuário pedir.

## 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 Explicação e Código

## 1.1 Explicação do código

### 1.1.1 Menu

O código contém um menu principal onde o usuário pode escolher qual questão deseja saber o resultado. Caso o usuário tenha selecionado as questões 1,2 e 3, aparecerá qual tipo de Iris (Iris-Setosa, Iris-Versicolor, Iris-Virginica) deseja ver o resultado. Se tiver selecionado a questão 4, aparecerá o resultado em seguida, sem a opção de escolher o tipo de Iris. Se o usuário digitar 5 ("SAIR") no Menu Principal, o programa se encerra. Caso ele digite 4 ("VOLTAR AO MENU PRINCIPAL"), quando estiver no Menu Iris, o programa volta para o menu principal e o usuário poderá escolher novamente o número da questão ou poderá sair. Um exemplo da questão 1 de como aparece para o usuário:

```
1 ##### MENU PRINCIPAL #####
2 Digite somente o numero da questao que voce deseja ver o resultado:
3 1 = Questao 1
4 2 = Questao 2
5 3 = Questao 3
6 4 = Questao 4
7 5 = SAIR
8 #####
```

```

1 ##### MENU IRIS #####
2 Digite qual especie voce deseja fazer a regressao linear:
3 1 = Iris-Setosa
4 2 = Iris-Versicolor
5 3 = Iris Virginica
6 4 = VOLTAR AO MENU PRINCIPAL
7 #####

```

### 1.1.2 Funções construídas

O código também contém funções construídas e que são chamadas durante o menu principal, tais como: **pegarDados**, **construir\_equacao\_normal**, **PLU**, **decomposicao\_espectral**, **construir\_svd** e **estimar\_amstras**.

#### 1.1.3 pegarDados

A função **pegarDados** extrai os dados do arquivo ".csv", de acordo com o tipo de Iris selecionado pelo usuário, percorrendo todo o banco de dados e retorna os dados em forma de matriz para ser usada em outras funções. Nessa matriz, os valores estão em forma de matriz [15x4], onde cada linha é um ID, e cada coluna um dos dados das medições das flores. Por exemplo, se o usuário escolheu a Questão 1 e Iris-Versicolor, os dados guardados em uma matriz serão referentes à Iris-Versicolor.

#### 1.1.4 construir\_equacao\_normal

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" na matriz  $x$ ).

A função **construir\_equacao\_normal** utiliza a equação normal  $x^T \cdot x \cdot w = x^T \cdot y$  ( $R \cdot w = p$ ,  $R = x^T \cdot x$  e  $p = x^T \cdot y$ ) e retorna os termos  $R$  e  $p$  com e sem o termo independente, considerando a construção de uma equação normal para cada tipo de Iris. Sem o termo independente, é construída uma matriz  $x$  deletando-se a última coluna da matriz de dados, de forma que a última coluna (PetalWidthCm) seja a nossa estimativa posteriormente, formando uma matriz [15x3]. Caso seja com o termo independente, essa última coluna é substituída por uma coluna com valores 1, formando uma matriz [15x4]. Para construir a matriz  $y$ , são deletadas as três primeiras colunas (SepalLengthCm, SepalWidthCm, PetalLengthCm) da matriz de dados, sobrando só a última coluna (PetalWidthCm), formando uma matriz [15x1].

Por fim, para encontrar  $R$ , é feita a transposta de  $x$  multiplicada por  $x$ , formando uma matriz  $R$  [3x3] sem o termo independente e [4x4] com o termo independente. Também para encontrar  $p$ , é feita a transposta de  $x$  multiplicada por  $y$ , formando uma matriz [3x1] sem o termo independente e [4x1] com o termo independente. Essa função retornará dois  $R$  e  $p$  diferentes, sem o termo independente e com o termo independente, que poderá ser utilizada posteriormente para as outras funções.

#### 1.1.5 PLU

A função **PLU** tem como objetivo transformar  $R$  em uma matriz triangular superior. Primeiro é feito um loop para percorrer cada coluna da matriz  $R$ , depois um loop dentro desse para percorrer as linhas de  $R$ . Sendo assim, ele fixa uma coluna em  $R$  e percorre todas as linhas de  $R$  nessa mesma coluna,

depois passa para próxima coluna e faz o mesmo loop das linhas até terminar na penúltima coluna. Então, durante o loop das colunas, é verificado sempre se o pivô, que é o elemento da diagonal principal, é igual a zero. Caso não seja zero, começa a construir a matriz L (triangular inferior). Caso seja igual a zero, vai para uma condição, para criar a matriz de permutação (P). Cada vez que é construída a matriz L ou P, R e p são multiplicadas por L ou P.

A matriz L é construída de forma que a cada coluna que o loop das colunas da matriz R percorre, é criada uma matriz L de zeros do mesmo tamanho que R. Desse modo, a matriz L será preenchida de forma que os elementos abaixo do pivô da coluna fixada se tornem zeros na matriz R. Assim, o loop das linhas de R começa sempre do número da coluna fixada de R+1 de modo que o loop das linhas comece sempre na posição abaixo do pivô. E o valor na linha e na coluna que estão sendo analisadas na matriz R será colocado na mesma posição na matriz L, só que dividido pelo valor do pivô da coluna fixada na matriz R e multiplicando essa razão por -1. Depois de percorrer todas as linhas da coluna fixada, a matriz L é preenchida com valor 1 na diagonal principal. Quando for para próxima coluna de R, é criada uma nova matriz L de zeros do mesmo tamanho que R e o loop das linhas se repete.

A matriz P é construída criando uma matriz de zeros do mesmo tamanho que R. Na matriz R, temos o objetivo de trocar a linha em que o pivô de R tem valor zero por uma linha em que o valor na mesma coluna do pivô não tenha zero. Desse modo, é feito um loop para percorrer todas as linhas abaixo da coluna do pivô de R e ver qual elemento é diferente de zero. Se for achado o elemento, é feita a construção da matriz P para fazer as trocas das duas linhas e o loop é terminado.

Depois é feito um loop para percorrer cada linha de P e ver se tem algum elemento 1 na linha, caso já tenha, não é colocado 1. Se não tiver o valor 1 em qualquer elemento da linha, é colocado 1 na posição com a linha e coluna iguais ao número da linha analisada.

Posteriormente, o loop das colunas de R volta para essa mesma coluna em que as linhas foram trocadas e ela é novamente analisada para zerar os elementos abaixo do pivô da coluna fixada. Caso não tenha um valor diferente de zero abaixo do pivô nessa coluna, é passada para a próxima coluna da matriz R e ela será novamente analisada.

E assim o loop continua, indo para a próxima coluna da matriz R até ter percorrido a penúltima coluna. Quando tiver terminado de percorrer até a penúltima coluna, a matriz R já terá se tornado triangular superior, é feito o backsubstitution, em que é dado R e p e a função `np.linalg.solve` retorna w (os coeficientes).

### 1.1.6 decomposicao\_espectral

A função **decomposicao\_espectral** calcula os autovalores e autovetores de R. Posteriormente, retorna uma matriz diagonal de autovalores de R ( $\Lambda$ ) e uma matriz de autovetores de R (matriz V). No menu principal, quando selecionada a opção Questão 2, essa parte chamará a função `decomposicao_espectral` duas vezes, uma para calcular as matrizes sem o termo independente e outra com o termo independente. Essa parte usará essas matrizes encontradas e imprimirá R, V,  $\Lambda$  e a transposta de V ( $V^T$ ) com e sem o termo independente.

### 1.1.7 construir\_svd

A função **construir\_svd** foi considerada como: U os autovetores de  $R \cdot R^T$ , V os autovetores de  $R^T \cdot R$  e  $\Sigma$  a matriz diagonal formada com os valores singulares (raiz quadrada dos autovalores) em comuns em  $R^T \cdot R$  e  $R \cdot R^T$ . Assim:  $R = U \cdot \Sigma \cdot V^T$ .

Como R é a matriz da equação normal, formada assim:  $R = x^T \cdot x$ . Nesse sentido, R é simétrica, já que  $R^T = R$ , uma vez que  $(x^T \cdot x)^T = x^T \cdot x$ . Por consequência,  $R^T \cdot R = R \cdot R^T$ ,  $R^T \cdot R$  possuirá os mesmos autovalores e autovetores de  $R \cdot R^T$  e U será igual a V. Dessa forma, só foi calculado os autovetores e autovalores de  $R \cdot R^T$ .

Assim, essa função calcula os autovalores e autovetores (matriz  $U$ ) de  $R \cdot R^T$ , depois faz um loop, para descobrir os valores singulares, então percorre cada autovalor, tira a raiz quadrada e coloca em um vetor. Essa função retornará os autovetores de  $R \cdot R^T$  (matriz  $U$ ), um vetor com os valores singulares de  $R \cdot R^T$  e a transposta de  $U$  ( $U^T$ ). Já no menu principal, quando selecionada a opção Questão 3, essa parte chamará a função `construir_svd` duas vezes, uma para calcular as matrizes e os vetores sem o termo independente e outra com o termo independente. Essa parte usará essas matrizes e vetores encontrados e imprimirá  $R$ ,  $U$ , uma matriz diagonal de valores singulares ( $\Sigma$ ) e a transposta de  $U$  (Lembrando que a transposta de  $V$  é igual a transposta de  $U$ , já que  $U = V$ ) com e sem o termo independente.

### 1.1.8 estimar\_amstras

A função `estimar_amstras` terá o propósito de fazer o produto interno de cada vetor de amostras selecionadas pelo professor, excluindo a coluna `PetalWidthCm`, com cada vetor de coeficientes da Iris-Setosa, Iris-Versicolor e Iris-Virginica. Caso seja com o termo independente, terá mais um valor nos vetores de amostras selecionados, com o número 1. Uma vez que a coluna `PetalWidthCm` será a selecionada para estimar a classe da Iris.

Depois de fazer o produto interno, será calculada a diferença (erro) do resultado do produto interno com o valor da `PetalWidthCm` de cada amostra e colocada numa lista que posteriormente será percorrida para cada erro ser transformado em seu módulo.

Por fim, é usada uma função que pega o menor valor dessa lista. Esse menor valor da lista é comparado com o módulo de cada erro da Iris-Setosa, Iris-Versicolor e Iris-Virginica, quando a comparação der igual, é retornada uma string com o tipo de Iris que deu igual. Por exemplo, se a comparação deu igual na Iris-Virginica, retornará uma string "Iris-Virginica".

No Menu Principal, a função `estimar_amstras` é chamada duas vezes, uma para calcular sem o termo independente e imprimir a classe estimada e outra com o termo independente. Por isso existe uma variável booleana que é `True` quando quer calcular com o termo independente e `False` quando é sem o termo independente. De forma que se essa variável for `True`, ela adiciona o valor 1 nos vetores de amostras.

## 1.2 Código

```
1 # Programa codigo.py
2 # Autora: Karen dos Anjos Arcoverde
3 # Data: 06/02/2021
4 #
5
6
7 import numpy as np
8 import sys
9
10
11
12 ##### Funcoes #####
13 def pegarDados(tipo_iris):
14
15     # tipo_iris = 1 Setosa , tipo_iris = 2 Versicolor,
16     # tipo_iris = 3 Virginica
17     dados = []
18     IDs = []
19
20     ## definicao dos ids das especies selecionadas para o trabalho
```

```

21 Setosa = range(25,39+1)
22 Versicolor = range(75,89+1)
23 Virginica = range (125,139+1)
24
25 arquivo= open("dados_13.csv",'r')
26 arquivo.readline() # ignora a primeira linha
27
28 if (tipo_iris == 1):
29     IDs = Setosa
30 if (tipo_iris == 2):
31     IDs = Versicolor
32 if (tipo_iris == 3):
33     IDs = Virginica
34
35 for i in range(1,46): # percorre todo o banco de dados 1-45
36     linha = (arquivo.readline()).split(',') #separa os dados por virgula
37
38     if (int(linha[0]) in IDs):# percorre os ids selecionados
39         linha.pop(0) # retira o ID dos dados
40         linha.pop(-1) # retira a especie dos dados
41         for j in range(4): #para cada dado
42             linha[j] = float(linha[j]) #transforma em numero
43
44         dados.append(linha) #adiciona na lista de dados
45
46 arquivo.close()
47
48 return dados
49
50 # -----
51 def construir_equacao_normal (dados):
52
53     #equacao normal - minimos quadrados:
54     #  $(x^T).x.w = (x^T).y$ 
55     #  $(x\_transposta).x.w = (x\_transposta).y$ 
56     #  $R.w = p$ 
57     #  $R = (x^T).x$ 
58     #  $p = (x^T).y$ 
59
60     ##### sem termo independente #####
61     #  $y = a*x1 + b*x2 + c*x3$ 
62     x = []
63     y = []
64
65     #achar  $(x^T)$  e x
66     # x - colunas SepalLengthCm, SepalWidthCm, PetalLengthCm
67     x = np.array(dados)
68     x = np.delete(x.reshape(15,4),3,1) #deleta a ultima coluna de x
69     x_transposta = np.transpose(x) #faz a transposta de x:  $(x^T)$ 
70
71     #achar R
72     R = np.dot(x_transposta,x) #multiplica  $(x^T)$  por x

```

```

73
74 #achar y - coluna PetalWidthCm
75 y = np.array(dados)
76 y = np.delete(y.reshape(15,4),0,1) #deleta a primeira coluna de y
77 y = np.delete(y.reshape(15,3),0,1) #deleta a segunda coluna de y
78 y = np.delete(y.reshape(15,2),0,1) #deleta a terceira coluna de y
79
80 #achar p
81 p = np.dot(x_transposta,y)
82
83 ##### com termo independente #####
84 # y = a*x1 +b*x2 +c*x3 +k
85
86 for i in range (0,15):
87     dados[i][3] = 1
88
89 #achar (x^T) e x
90 x = np.array(dados)
91 x_transposta = np.transpose(x) #faz a transposta de x: (x^T)
92 #achar R
93 R1= np.dot(x_transposta,x) #multiplica (x^T) por x
94 #achar p
95 p1 = np.dot(x_transposta,y)
96
97 return R,p,R1,p1
98
99 # -----
100 def PLU(R,p):
101     indice_coluna = 0
102     indice_L_1s = 0
103     indice_escolher_linha = 0
104     tamanho_R = len(R)
105
106     while (indice_coluna < (tamanho_R-1)):
107         #constroi a matriz L, triangular inferior
108         if (R[indice_coluna][indice_coluna] != 0):
109             L = np.zeros((tamanho_R, tamanho_R))
110
111             indice_coluna_aux = indice_coluna + 1
112             while (indice_coluna_aux < tamanho_R):
113                 L[indice_coluna_aux][indice_coluna] = - R[indice_coluna_aux][
114                     indice_coluna]/R[indice_coluna][indice_coluna]
115                 indice_coluna_aux += 1
116
117             while (indice_L_1s < tamanho_R):
118                 L[indice_L_1s][indice_L_1s] = 1
119                 indice_L_1s += 1
120
121             indice_L_1s = 0
122
123             #multiplica a matriz R por L
124             R = np.dot (L,R)

```



```

124     p = np.dot (L,p)
125
126
127     #se o pivo for zero, necessario multiplicar por uma matriz P de
    permutacao
128     if (R[indice_coluna][indice_coluna] == 0):
129         P = np.zeros((tamanho_R, tamanho_R))
130         indice_escolher_linha = indice_coluna + 1
131         while (indice_escolher_linha < tamanho_R):
132             if (R[indice_escolher_linha][indice_coluna] != 0):
133                 P[indice_coluna][indice_escolher_linha] = 1
134                 P[indice_escolher_linha][indice_coluna] = 1
135
136                 indice_coluna -= 1
137                 break
138
139                 indice_escolher_linha += 1
140
141     i = 0
142     j = 0
143     guarda_1 = False
144     while (i < tamanho_R):
145         while (j < tamanho_R):
146             if (P[i][j] == 1):
147                 guarda_1 = True
148                 j += 1
149             if (guarda_1 == False):
150                 P[i][i] = 1
151             i += 1
152
153     R = np.dot(P,R)
154     p = np.dot (P,p)
155
156
157     indice_coluna += 1
158
159     #backsubstitution
160     w = np.linalg.solve(R, p)
161
162     return w
163
164 # -----
165 def decomposicao_espectral(R):
166
167     #  $R = VDV^T$ 
168     #determinando autovalores e autovetores
169     autovalores, autovetores = np.linalg.eig(R)
170     # matriz diagonal de autovalores
171     matrizDiagonal = np.diag(autovalores)
172
173     return autovetores, matrizDiagonal
174

```

```

175 # -----
176 def contruir_svd (R):
177     s = []
178     i = 0
179
180     # R = U.s.VT
181     #determinando autovalores e autovetores
182     ### U
183     ## autovetores de R.(R^T)
184     autovaloresU, autovetoresU = np.linalg.eig(np.dot(R,np.transpose(R)))
185
186     # como R eh simetrica e quadrada: (R^T).R = R.(R^T)
187     # entao autovetores e autovalores de U sao iguais aos de V
188
189     tamanho_autovalores_U = autovaloresU.shape
190
191     while (i < tamanho_autovalores_U [0]):
192         s.append(np.sqrt(autovaloresU [i]))
193         i += 1
194
195     return autovetoresU, np.array(s),np.transpose(autovetoresU)
196
197 # -----
198 def estimar_amstras(amostra, w_setosa,w_versicolor,w_virginica,c_independente
199 ):
200
201     lista_erros = []
202     estimativa = ""
203     indice = 0
204     amostra_x = []
205     indice_erros_modulo = 0
206
207     while (indice < 3):
208         amostra_x.append (amostra[indice])
209         indice += 1
210
211     if (c_independente == True):
212         amostra_x.append(1)
213
214     amostra_x = np.array(amostra_x)
215
216     #produto interno <x,y> = (x^T).y
217
218     estimativa_setosa = np.dot(amostra_x,w_setosa)
219     estimativa_versicolor = np.dot(amostra_x,w_versicolor)
220     estimativa_virginica = np.dot(amostra_x,w_virginica)
221
222     erro_setosa = estimativa_setosa[0] - amostra[3]
223     lista_erros.append(erro_setosa)
224
225     erro_versicolor = estimativa_versicolor[0] - amostra[3]
226     lista_erros.append(erro_versicolor)

```

```

226 erro_virginica = estimativa_virginica[0] - amostra[3]
227 lista_erros.append(erro_virginica)
228
229
230 while (indice_erros_modulo < len(lista_erros)):
231     lista_erros[indice_erros_modulo] = abs(lista_erros[indice_erros_modulo
232     ])
233     indice_erros_modulo += 1
234
235 if (min(lista_erros) == abs(erro_setosa)):
236     estimativa = "Iris-Setosa"
237
238     return estimativa
239
240 if (min(lista_erros) == abs(erro_versicolor)):
241     estimativa = "Iris-Versicolor"
242
243     return estimativa
244
245 if (min(lista_erros) == abs(erro_virginica)):
246     estimativa = "Iris-Virginica"
247
248     return estimativa
249
250 ##### Programa Principal #####
251 def menu():
252     resultado = 0
253     tipo_iris = 0
254     coeficientes_sem_aux = []
255     coeficientes_com_aux = []
256     indice = 0
257
258     while (resultado != 5):
259         print()
260         print('##### MENU PRINCIPAL
261         #####')
262         print("Digite somente o numero da questao que voce deseja ver o
263         resultado: ")
264         print("1 = Questao 1")
265         print("2 = Questao 2")
266         print("3 = Questao 3")
267         print("4 = Questao 4")
268         print("5 = SAIR")
269         print('
270         #####
271         ')
272         print()
273
274         resultado = int(input())
275
276         if (resultado == 5):

```

```

273     sys.exit(0)
274
275     if (resultado == 1):
276
277         while (tipo_iris != 4):
278             print('##### MENU IRIS
279             #####')
280             print("Digite qual especie voce deseja fazer a regressao
281             linear: ")
282             print("1 = Iris-Setosa")
283             print("2 = Iris-Versicolor")
284             print("3 = Iris Virginica")
285             print("4 = VOLTAR AO MENU PRINCIPAL")
286             print('
287             #####')
288             print()
289
290             tipo_iris = int(input())
291
292             if (tipo_iris == 4):
293                 menu()
294
295             else:
296                 dados = pegarDados (tipo_iris)
297                 R,p,R1,p1 = construir_equacao_normal(dados)
298                 w = PLU(R,p)
299                 w1 = PLU(R1,p1)
300
301                 print()
302                 if (tipo_iris == 1):
303                     print("Iris-Setosa\n")
304                 elif (tipo_iris == 2):
305                     print("Iris-Versicolor\n")
306                 elif (tipo_iris == 3):
307                     print("Iris-Virginica\n")
308
309                 print("SEM O TERMO INDEPENDENTE: ")
310                 print("y = a*x1 + b*x2 + c*x3")
311                 print("[a b c] = ",end="")
312
313                 while (indice < len(w)):
314                     coeficientes_sem_aux.append(w[indice][0])
315                     coeficientes_sem = np.array(coeficientes_sem_aux)
316                     indice += 1
317
318                 print(coeficientes_sem)
319                 coeficientes_sem_aux = []
320
321                 print()

```

```

322     print("COM O TERMO INDEPENDENTE: ")
323     print("y = a*x1 + b*x2 + c*x3 + k")
324     print("[a b c k] = ",end="")
325
326     indice = 0
327     while (indice < len(w1)):
328         coeficientes_com_aux.append(w1[indice][0])
329         coeficientes_com = np.array(coeficientes_com_aux)
330         indice += 1
331
332     print(coeficientes_com)
333     coeficientes_com_aux = []
334     indice = 0
335
336     print()
337
338 if (resultado == 2):
339
340     while (tipo_iris != 4):
341         print('##### MENU IRIS
342             #####')
343         print("Digite qual especie voce deseja fazer a decomposicao
344             espectral: ")
345         print("1 = Iris-Setosa")
346         print("2 = Iris-Versicolor")
347         print("3 = Iris Virginica")
348         print("4 = VOLTAR AO MENU PRINCIPAL")
349         print('
350             #####
351             ')
352         print()
353
354         tipo_iris = int(input())
355
356         if (tipo_iris == 4):
357             menu()
358
359         else:
360             dados = pegarDados (tipo_iris)
361             R,p,R1,p1 = construir_equacao_normal(dados)
362             autovetores, matrizDiagonal = decomposicao_espectral(R)
363             autovetores1, matrizDiagonal1 = decomposicao_espectral(R1)
364
365             if (tipo_iris == 1):
366                 print("Iris-Setosa\n")
367             elif (tipo_iris == 2):
368                 print("Iris-Versicolor\n")
369             elif (tipo_iris == 3):
370                 print("Iris-Virginica\n")
371
372             print("R = V\u039BV^(T)")

```

```

370         print()
371
372
373         print("SEM O TERMO INDEPENDENTE: ")
374         print("R = ", R)
375         print()
376         print("V = ", autovetores)
377         print()
378         print('\'u039B = ', matrizDiagonal)
379         print()
380         print("V^T = ", np.transpose(autovetores))
381
382         print()
383         print()
384
385         print("COM O TERMO INDEPENDENTE: ")
386         print("R = ", R1)
387         print()
388         print("V = ", autovetores1)
389         print()
390         print('\'u039B = ', matrizDiagonal1)
391         print()
392         print("V^T = ", np.transpose(autovetores1))
393
394         print()
395
396     if (resultado == 3):
397
398         while (tipo_iris != 4):
399             print('##### MENU IRIS #####')
400             print("Digite qual especie voce deseja fazer o SVD: ")
401             print("1 = Iris-Setosa")
402             print("2 = Iris-Versicolor")
403             print("3 = Iris Virginica")
404             print("4 = VOLTAR AO MENU PRINCIPAL")
405             print('#####')
406             print()
407
408             tipo_iris = int(input())
409
410             if (tipo_iris == 4):
411                 menu()
412
413             else:
414
415                 dados = pegarDados (tipo_iris)
416                 R,p,R1,p1 = construir_equacao_normal(dados)
417                 U, s, VT = contruir_svd(R)
418                 U1, s1, VT1 = contruir_svd(R1)
419
420                 if (tipo_iris == 1):
421                     print("Iris-Setosa\n")

```

```

422         elif (tipo_iris == 2):
423             print("Iris-Versicolor\n")
424         elif (tipo_iris == 3):
425             print("Iris-Virginica\n")
426
427     print("R =  $U \cdot V^T$ ")
428     print()
429
430     print("SEM O TERMO INDEPENDENTE: ")
431     print("R = ", R)
432     print()
433     print("U = ", U)
434     print()
435     print('  $\Lambda$  = ', np.diag(s))
436     print()
437     print("V^T = ", VT)
438     print()
439     print()
440
441     print("COM O TERMO INDEPENDENTE: ")
442     print("R = ", R1)
443     print()
444     print("U = ", U1)
445     print()
446     print('  $\Lambda$  = ', np.diag(s1))
447     print()
448     print("V^T = ", VT1)
449
450     print()
451
452
453
454 if (resultado == 4):
455     A = [5.0,2.3,3.3,1.0]
456     B = [4.6,3.2,1.4,0.2]
457     C = [5.0,3.3,1.4,0.2]
458     D = [6.1,3.0,4.6,1.4]
459     E = [5.9,3.0,5.1,1.8]
460
461     print("SEM O TERMO INDEPENDENTE: ")
462     dados = pegarDados (1)
463     R,p,R1,p1 = construir_equacao_normal(dados)
464     w_setosa = PLU(R,p)
465
466     dados = pegarDados (2)
467     R,p,R1,p1 = construir_equacao_normal(dados)
468     w_versicolor = PLU(R,p)
469
470     dados = pegarDados (3)
471     R,p,R1,p1 = construir_equacao_normal(dados)
472     w_virginica = PLU(R,p)
473

```

```

474     c_independente = False
475     estimativa = estimar_amostras(A,w_setosa,w_versicolor,w_virginica,
        c_independente)
476     print("A = ", estimativa)
477     estimativa = estimar_amostras(B,w_setosa,w_versicolor,w_virginica,
        c_independente)
478     print("B = ", estimativa)
479     estimativa = estimar_amostras(C,w_setosa,w_versicolor,w_virginica,
        c_independente)
480     print("C = ", estimativa)
481     estimativa = estimar_amostras(D,w_setosa,w_versicolor,w_virginica,
        c_independente)
482     print("D = ", estimativa)
483     estimativa = estimar_amostras(E,w_setosa,w_versicolor,w_virginica,
        c_independente)
484     print("E = ", estimativa)
485
486     print()
487     print("COM O TERMO INDEPENDENTE: ")
488     dados = pegarDados (1)
489     R,p,R1,p1 = construir_equacao_normal(dados)
490     w1_setosa = PLU(R1,p1)
491
492     dados = pegarDados (2)
493     R,p,R1,p1 = construir_equacao_normal(dados)
494     w1_versicolor = PLU(R1,p1)
495
496     dados = pegarDados (3)
497     R,p,R1,p1 = construir_equacao_normal(dados)
498     w1_virginica = PLU(R1,p1)
499
500     c_independente = True
501     estimativa1 = estimar_amostras(A,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
502     print("A = ", estimativa1)
503     estimativa1 = estimar_amostras(B,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
504     print("B = ", estimativa1)
505     estimativa1 = estimar_amostras(C,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
506     print("C = ", estimativa1)
507     estimativa1 = estimar_amostras(D,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
508     print("D = ", estimativa1)
509     estimativa1 = estimar_amostras(E,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
510     print("E = ", estimativa1)
511
512     menu()
513
514     ##### chamada ao menu
515     menu()

```



## 2 Questão 1

### 2.1 Funções utilizadas

Para essa parte foram utilizadas as funções: `menu`, `pegarDados`, `construir_equacao_normal` e `PLU`.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal e com o R e p retornados da equação normal, aplicar PLU e backsubstitution para descobrir os coeficientes da equação normal.

### 2.2 Resultados

#### 2.2.1 Iris-Setosa

```
1 Iris-Setosa
2
3 SEM O TERMO INDEPENDENTE:
4  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3$ 
5  $[a \ b \ c] = [0.07455282 \ -0.06602361 \ 0.03673264]$ 
6
7 COM O TERMO INDEPENDENTE:
8  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3 + k$ 
9  $[a \ b \ c \ k] = [0.13497209 \ -0.07886596 \ 0.10365958 \ -0.36144997]$ 
```

#### 2.2.2 Iris-Versicolor

```
1 Iris-Versicolor
2
3 SEM O TERMO INDEPENDENTE:
4  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3$ 
5  $[a \ b \ c] = [-0.14382683 \ 0.17051618 \ 0.40397714]$ 
6
7 COM O TERMO INDEPENDENTE:
8  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3 + k$ 
9  $[a \ b \ c \ k] = [-0.06826027 \ 0.24149193 \ 0.39877688 \ -0.63863516]$ 
```

#### 2.2.3 Iris-Virginica

```
1 Iris-Virginica
2
3 SEM O TERMO INDEPENDENTE:
4  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3$ 
5  $[a \ b \ c] = [-0.11329721 \ 0.3990124 \ 0.25976859]$ 
6
7 COM O TERMO INDEPENDENTE:
8  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3 + k$ 
9  $[a \ b \ c \ k] = [-0.11686468 \ 0.35346044 \ 0.22325742 \ 0.36734017]$ 
```

## 3 Questão 2

### 3.1 Funções utilizadas

Para essa parte foram utilizadas as funções: `menu`, `pegarDados`, `construir_equacao_normal` e `decomposicao_espectral`.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal e utilizar o R retornado da equação normal para fazer a decomposição espectral.

### 3.2 Resultados

#### 3.2.1 Iris-Setosa

```
1 Iris-Setosa
2
3 R = VAV^(T)
4
5 SEM O TERMO INDEPENDENTE:
6 R = [[381.33 255.73 112.57]
7      [255.73 172.5 75.51]
8      [112.57 75.51 33.84]]
9
10 V = [[-0.80618188 -0.45661854 0.37625828]
11      [-0.54157827 0.31342349 -0.78003762]
12      [-0.23825146 0.8326255 0.49997102]]
13
14 Λ = [[5.86392634e+02 0.00000000e+00 0.00000000e+00]
15      [0.00000000e+00 5.29776824e-01 0.00000000e+00]
16      [0.00000000e+00 0.00000000e+00 7.47589571e-01]]
17
18 V^T = [[-0.80618188 -0.54157827 -0.23825146]
19         [-0.45661854 0.31342349 0.8326255 ]
20         [ 0.37625828 -0.78003762 0.49997102]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[381.33 255.73 112.57 75.5 ]
25      [255.73 172.5 75.51 50.6 ]
26      [112.57 75.51 33.84 22.4 ]
27      [ 75.5 50.6 22.4 15. ]]]
28
29 V = [[-0.79610972 0.1628056 -0.47792329 -0.33360603]
30      [-0.53478061 -0.03864172 0.33762985 0.77364243]
31      [-0.23529777 0.18572124 0.80860497 -0.50626137]
32      [-0.15765144 -0.96825037 0.06126507 -0.18407563]]
33
34 Λ = [[6.01336860e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35      [0.00000000e+00 2.79352229e-02 0.00000000e+00 0.00000000e+00]
36      [0.00000000e+00 0.00000000e+00 5.31947766e-01 0.00000000e+00]
37      [0.00000000e+00 0.00000000e+00 0.00000000e+00 7.73257486e-01]]
38
39 V^T = [[-0.79610972 -0.53478061 -0.23529777 -0.15765144]
```

```

40 [ 0.1628056 -0.03864172 0.18572124 -0.96825037]
41 [-0.47792329 0.33762985 0.80860497 0.06126507]
42 [-0.33360603 0.77364243 -0.50626137 -0.18407563]]

```

### 3.2.2 Iris-Versicolor

```

1 Iris-Versicolor
2
3 R = VΛV^(T)
4
5 SEM 0 TERMO INDEPENDENTE:
6 R = [[555.38 256.64 397.59]
7      [256.64 119.98 184.35]
8      [397.59 184.35 286.5 ]]
9
10 V = [[ 0.76039163 0.64759888 -0.0491961 ]
11      [ 0.35223975 -0.47485674 -0.80649751]
12      [ 0.54564799 -0.59592513 0.58918716]]
13
14 Λ = [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
15      [0.00000000e+00 1.33162776e+00 0.00000000e+00]
16      [0.00000000e+00 0.00000000e+00 9.57976568e-01]]
17
18 V^T = [[ 0.76039163 0.35223975 0.54564799]
19        [ 0.64759888 -0.47485674 -0.59592513]
20        [-0.0491961 -0.80649751 0.58918716]]
21
22
23 COM 0 TERMO INDEPENDENTE:
24 R = [[555.38 256.64 397.59 91. ]
25      [256.64 119.98 184.35 42.2 ]
26      [397.59 184.35 286.5 65.2 ]
27      [ 91. 42.2 65.2 15. ]]
28
29 V = [[-0.7545525 0.11812875 0.64474223 0.03167953]
30      [-0.34954066 0.11526607 -0.46962259 0.80248968]
31      [-0.54144512 -0.01367345 -0.60234554 -0.58637025]
32      [-0.1237297 -0.98619084 0.030691 0.1057197 ]]
33
34 Λ = [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35      [0.00000000e+00 7.14132919e-02 0.00000000e+00 0.00000000e+00]
36      [0.00000000e+00 0.00000000e+00 1.33280153e+00 0.00000000e+00]
37      [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]
38
39 V^T = [[-0.7545525 -0.34954066 -0.54144512 -0.1237297 ]
40        [ 0.11812875 0.11526607 -0.01367345 -0.98619084]
41        [ 0.64474223 -0.46962259 -0.60234554 0.030691 ]
42        [ 0.03167953 0.80248968 -0.58637025 0.1057197 ]]

```

### 3.2.3 Iris-Virginica

```

1 Iris-Virginica
2
3 R = VAV^(T)
4
5 SEM O TERMO INDEPENDENTE:
6 R = [[676.15 304.85 562.72]
7       [304.85 138.7 253.94]
8       [562.72 253.94 469.3 ]]
9
10 V = [[-0.72592866 -0.63483621 0.2645951 ]
11       [-0.32772296 -0.01894783 -0.94458385]
12       [-0.60466953 0.77241438 0.19429563]]
13
14 Λ = [[1.28249881e+03 0.00000000e+00 0.00000000e+00]
15       [0.00000000e+00 5.79253032e-01 0.00000000e+00]
16       [0.00000000e+00 0.00000000e+00 1.07193335e+00]]
17
18 V^T = [[-0.72592866 -0.32772296 -0.60466953]
19         [-0.63483621 -0.01894783 0.77241438]
20         [0.2645951 -0.94458385 0.19429563]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[676.15 304.85 562.72 100.3 ]
25       [304.85 138.7 253.94 45.4 ]
26       [562.72 253.94 469.3 83.6 ]
27       [100.3 45.4 83.6 15. ]]
28
29 V = [[-7.21743001e-01 -2.76297507e-01 6.34623278e-01 -1.50933108e-04]
30       [-3.25843186e-01 9.35769444e-01 3.68031853e-02 -1.29642937e-01]
31       [-6.01187515e-01 -1.93722166e-01 -7.68083597e-01 -1.05322749e-01]
32       [-1.07176625e-01 1.02308151e-01 -7.71129602e-02 9.85951218e-01]]
33
34 Λ = [[1.29740071e+03 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35       [0.00000000e+00 1.08243546e+00 0.00000000e+00 0.00000000e+00]
36       [0.00000000e+00 0.00000000e+00 5.82311582e-01 0.00000000e+00]
37       [0.00000000e+00 0.00000000e+00 0.00000000e+00 8.45463191e-02]]
38
39 V^T = [[-7.21743001e-01 -3.25843186e-01 -6.01187515e-01 -1.07176625e-01]
40         [-2.76297507e-01 9.35769444e-01 -1.93722166e-01 1.02308151e-01]
41         [6.34623278e-01 3.68031853e-02 -7.68083597e-01 -7.71129602e-02]
42         [-1.50933108e-04 -1.29642937e-01 -1.05322749e-01 9.85951218e-01]]

```

## 4 Questão 3

### 4.1 Funções utilizadas

Para essa parte foram utilizadas as funções: menu, PegarDados, construir\_equacao\_normal e construir\_svd.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal e utilizar o R retornado da equação normal para fazer o SVD.

## 4.2 Resultados

### 4.2.1 Iris-Setosa

```
1 Iris-Setosa
2
3  $R = U\Sigma V^T$ 
4
5 SEM O TERMO INDEPENDENTE:
6  $R = \begin{bmatrix} 381.33 & 255.73 & 112.57 \\ 255.73 & 172.5 & 75.51 \\ 112.57 & 75.51 & 33.84 \end{bmatrix}$ 
7
8
9
10  $U = \begin{bmatrix} 0.80618188 & -0.45661854 & -0.37625828 \\ 0.54157827 & 0.31342349 & 0.78003762 \\ 0.23825146 & 0.8326255 & -0.49997102 \end{bmatrix}$ 
11
12
13
14  $\Sigma = \begin{bmatrix} 5.86392634e+02 & 0.00000000e+00 & 0.00000000e+00 \\ 0.00000000e+00 & 5.29776824e-01 & 0.00000000e+00 \\ 0.00000000e+00 & 0.00000000e+00 & 7.47589571e-01 \end{bmatrix}$ 
15
16
17
18  $V^T = \begin{bmatrix} 0.80618188 & 0.54157827 & 0.23825146 \\ -0.45661854 & 0.31342349 & 0.8326255 \\ -0.37625828 & 0.78003762 & -0.49997102 \end{bmatrix}$ 
19
20
21
22
23 COM O TERMO INDEPENDENTE:
24  $R = \begin{bmatrix} 381.33 & 255.73 & 112.57 & 75.5 \\ 255.73 & 172.5 & 75.51 & 50.6 \\ 112.57 & 75.51 & 33.84 & 22.4 \\ 75.5 & 50.6 & 22.4 & 15. \end{bmatrix}$ 
25
26
27
28
29  $U = \begin{bmatrix} 0.79610972 & -0.33360603 & -0.47792329 & -0.1628056 \\ 0.53478061 & 0.77364243 & 0.33762985 & 0.03864172 \\ 0.23529777 & -0.50626137 & 0.80860497 & -0.18572124 \\ 0.15765144 & -0.18407563 & 0.06126507 & 0.96825037 \end{bmatrix}$ 
30
31
32
33
34  $\Sigma = \begin{bmatrix} 6.01336860e+02 & 0.00000000e+00 & 0.00000000e+00 & 0.00000000e+00 \\ 0.00000000e+00 & 7.73257486e-01 & 0.00000000e+00 & 0.00000000e+00 \\ 0.00000000e+00 & 0.00000000e+00 & 5.31947766e-01 & 0.00000000e+00 \\ 0.00000000e+00 & 0.00000000e+00 & 0.00000000e+00 & 2.79352230e-02 \end{bmatrix}$ 
35
36
37
38
39  $V^T = \begin{bmatrix} 0.79610972 & 0.53478061 & 0.23529777 & 0.15765144 \\ -0.33360603 & 0.77364243 & -0.50626137 & -0.18407563 \\ -0.47792329 & 0.33762985 & 0.80860497 & 0.06126507 \\ -0.1628056 & 0.03864172 & -0.18572124 & 0.96825037 \end{bmatrix}$ 
40
41
42
```

### 4.2.2 Iris-Versicolor

```
1 Iris-Versicolor
2
3  $R = U\Sigma V^T$ 
```

```

4
5 SEM O TERMO INDEPENDENTE:
6 R = [[555.38 256.64 397.59]
7       [256.64 119.98 184.35]
8       [397.59 184.35 286.5 ]]
9
10 U = [[ 0.76039163  0.64759888 -0.0491961 ]
11        [ 0.35223975 -0.47485674 -0.80649751]
12        [ 0.54564799 -0.59592513  0.58918716]]
13
14 Σ = [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
15        [0.00000000e+00 1.33162776e+00 0.00000000e+00]
16        [0.00000000e+00 0.00000000e+00 9.57976568e-01]]
17
18 V^T = [[ 0.76039163  0.35223975  0.54564799]
19          [ 0.64759888 -0.47485674 -0.59592513]
20          [-0.0491961  -0.80649751  0.58918716]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[555.38 256.64 397.59  91.  ]
25        [256.64 119.98 184.35  42.2 ]
26        [397.59 184.35 286.5   65.2 ]
27        [ 91.    42.2   65.2   15.  ]]
28
29 U = [[ 0.7545525  -0.64474223 -0.11812875  0.03167953]
30        [ 0.34954066  0.46962259 -0.11526607  0.80248968]
31        [ 0.54144512  0.60234554  0.01367345 -0.58637025]
32        [ 0.1237297  -0.030691   0.98619084  0.1057197 ]]
33
34 Σ = [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35        [0.00000000e+00 1.33280153e+00 0.00000000e+00 0.00000000e+00]
36        [0.00000000e+00 0.00000000e+00 7.14132919e-02 0.00000000e+00]
37        [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]
38
39 V^T = [[ 0.7545525  0.34954066  0.54144512  0.1237297 ]
40          [-0.64474223  0.46962259  0.60234554 -0.030691 ]
41          [-0.11812875 -0.11526607  0.01367345  0.98619084]
42          [ 0.03167953  0.80248968 -0.58637025  0.1057197 ]]

```

### 4.2.3 Iris-Virginica

```

1 Iris-Virginica
2
3 R = UΣV^T)
4
5 SEM O TERMO INDEPENDENTE:
6 R = [[676.15 304.85 562.72]
7       [304.85 138.7  253.94]
8       [562.72 253.94 469.3 ]]
9
10 U = [[ 0.72592866 -0.63483621 -0.2645951 ]

```

```

11 [ 0.32772296 -0.01894783  0.94458385]
12 [ 0.60466953  0.77241438 -0.19429563]]
13
14 Σ = [[1.28249881e+03  0.00000000e+00  0.00000000e+00]
15 [0.00000000e+00  5.79253032e-01  0.00000000e+00]
16 [0.00000000e+00  0.00000000e+00  1.07193335e+00]]
17
18 V^T = [[ 0.72592866  0.32772296  0.60466953]
19 [-0.63483621 -0.01894783  0.77241438]
20 [-0.2645951  0.94458385 -0.19429563]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[676.15  304.85  562.72  100.3 ]
25 [304.85  138.7  253.94  45.4 ]
26 [562.72  253.94  469.3  83.6 ]
27 [100.3  45.4  83.6  15.  ]]
28
29 U = [[ 7.21743001e-01 -2.76297507e-01 -6.34623278e-01 -1.50933128e-04]
30 [ 3.25843186e-01  9.35769444e-01 -3.68031855e-02 -1.29642937e-01]
31 [ 6.01187515e-01 -1.93722166e-01  7.68083597e-01 -1.05322749e-01]
32 [ 1.07176625e-01  1.02308151e-01  7.71129601e-02  9.85951218e-01]]
33
34 Σ = [[1.29740071e+03  0.00000000e+00  0.00000000e+00  0.00000000e+00]
35 [0.00000000e+00  1.08243546e+00  0.00000000e+00  0.00000000e+00]
36 [0.00000000e+00  0.00000000e+00  5.82311582e-01  0.00000000e+00]
37 [0.00000000e+00  0.00000000e+00  0.00000000e+00  8.45463191e-02]]
38
39 V^T = [[ 7.21743001e-01  3.25843186e-01  6.01187515e-01  1.07176625e-01]
40 [-2.76297507e-01  9.35769444e-01 -1.93722166e-01  1.02308151e-01]
41 [-6.34623278e-01 -3.68031855e-02  7.68083597e-01  7.71129601e-02]
42 [-1.50933128e-04 -1.29642937e-01 -1.05322749e-01  9.85951218e-01]]

```

## 5 Questão 4

### 5.1 Funções utilizadas

Para essa parte foram utilizadas as funções: `menu`, `pegarDados`, `construir_equacao_normal`, `PLU` e `estimar_amstras`.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal, utilizar o `R` e `p` retornados da equação normal para aplicar `PLU` e `backsubstitution` para descobrir os coeficientes da equação normal e estimar as amostras fornecidas utilizando os coeficientes achados.

### 5.2 Resultados

```

1 SEM O TERMO INDEPENDENTE:
2 A =  Iris-Versicolor
3 B =  Iris-Setosa
4 C =  Iris-Setosa
5 D =  Iris-Versicolor

```

```
6 E = Iris-Virginica
7
8 COM O TERMO INDEPENDENTE:
9 A = Iris-Versicolor
10 B = Iris-Setosa
11 C = Iris-Setosa
12 D = Iris-Versicolor
13 E = Iris-Virginica
```

## 6 Bibliografia

- <https://algebra-linear.ufcg.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/](https://www.geeksforgeeks.org/python-exit-commands-quit-exit-sys-exit-and-os_exit/)