

IA 006

LISTA DE EXERCÍCIOS 1

Roger Danilo Figlie

RA 189957

23/09/2019

Sumário

1	EXERCÍCIO TEÓRICOS	3
1.1	Exercício 1	3
1.2	Exercício 2	4
1.3	Exercício 3	4
2	EXERCÍCIO COMPUTACIONAIS	7
2.1	Exercício 1	7
2.1.1	Letra a	7
2.1.2	Letra b	10
2.2	Exercício 2	11
2.2.1	Letra a	12
2.2.2	Letra b	13
2.2.3	Letra c	13
	ANEXOS	15
	ANEXO A – CÓDIGO PARA EXERCÍCIO 1	16
	ANEXO B – CÓDIGO PARA EXERCÍCIO 2	19

1 Exercício Teóricos

1.1 Exercício 1

1. ~~X~~Y

	Y=0	Y=1
X=0	$\frac{1}{6}$	$\frac{3}{8}$
X=1	$\frac{1}{8}$	$\frac{1}{3}$

a) $P(X=0) = \frac{1}{6} + \frac{3}{8} = \frac{4+9}{24} = \frac{13}{24} //$

$P(X=1) = \frac{1}{8} + \frac{1}{3} = \frac{3+8}{24} = \frac{11}{24} //$

$P(Y=0) = \frac{1}{6} + \frac{1}{8} = \frac{4+3}{24} = \frac{7}{24} //$

$P(Y=1) = \frac{3}{8} + \frac{1}{3} = \frac{9+8}{24} = \frac{17}{24} //$

b) $P(X=0|Y=0) = \frac{P(X=0 \cap Y=0)}{P(Y=0)} = \frac{\frac{1}{6}}{\frac{7}{24}} = \frac{1}{6} \cdot \frac{24}{7} = \frac{4}{7}$

c) $E[X] = \sum_k x_k \cdot P_X(x_k)$
 "Valor média Esperada"

$E[X] = 0 \cdot \frac{13}{24} + 1 \cdot \frac{11}{24} = \frac{11}{24} //$

$E[Y] = 0 \cdot \frac{7}{24} + 1 \cdot \frac{17}{24} = \frac{17}{24} //$

d) Pelo teorema de Bayes temos:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Se $P(A|B) = P(A)$ p/ todos os valores de A e B então dizemos que A é independente de B.

1.2 Exercício 2

1.1

② $X \setminus Y$ $Y=0$ $Y=1$

$x=0$	$1/4$	$1/4$	$P(x=0) = 1/4$
$x=1$	$3/8$	$3/8$	$P(x=1) = 3/4$
	$P(y=0) = 1/8$	$P(y=1) = 5/8$	

a) $H(X) = -\sum p_x(x_i) \log_2(p_x(x_i)) =$

$$= -1/4 \cdot \log_2(1/4) - 3/4 \cdot \log_2(3/4) =$$

$$H(X) = -1/4 \cdot -2 - 3/4 \cdot -0,41504 = 0,8113 \text{ bits}$$

$H(Y) = -\sum p_y(y_i) \log_2(p_y(y_i)) =$

$$= -3/8 \cdot \log_2(3/8) - 5/8 \cdot \log_2(5/8) =$$

$$= -3/8 \cdot -1,4150 - 5/8 \cdot -0,67807 = 0,9544 \text{ bits}$$

$H(X,Y) = \sum_i \sum_j p_{xy}(x_i, y_j) \log_2(p_{xy}(x_i, y_j)) =$

Dois desconsiderados

$$= -0 \cdot \log_2(0) - 1/4 \cdot \log_2(1/4)$$

$$- 3/8 \cdot \log_2(3/8) - 3/8 \cdot \log_2(3/8)$$

$$= 0,23.2$$

$$0,23 \cdot 1,4151 + 0,23 \cdot 1,4151$$

$$= 0,5 + 0,53064 + 0,53064 = 1,5613 \text{ bits}$$

b) $H(X|Y) = H(X,Y) - H(Y) = 1,5613 - 0,9544 = 0,6069 \text{ bits}$

$H(Y|X) = H(X,Y) - H(X) = 1,5613 - 0,8113 = 0,75 \text{ bits}$

c) $I(X,Y) = H(X) - H(X|Y) = 0,8113 - 0,6069 = 0,2044 \text{ bits}$

1.3 Exercício 3

(3) a) Para o critério de ML, o classificador decide por $x \in C_1$ se $p(x|C_1) > p(x|C_2)$

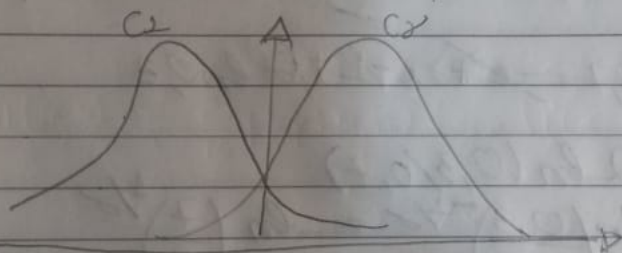
Logo precisamos achar o ponto onde $P(x|C_1) = P(x|C_2)$

$$\bullet P(x|C_1) = P(x|C_2) \Rightarrow N(x, -1, 1) = N(x, 1, 1)$$

$$\Rightarrow \frac{1}{\sqrt{2\pi} \cdot 1} \cdot e^{\frac{-(x+1)^2}{2}} = \frac{1}{\sqrt{2\pi} \cdot 1} \cdot e^{\frac{-(x-1)^2}{2}}$$

$$(x+1)^2 = (x-1)^2$$

$$4x = 0 \Rightarrow x = 0$$



Logo se $x \leq 0$, dizemos que $x \in C_1$ e se $x > 0$ dizemos que $x \in C_2$.

b) Já a MAP encontra a máxima a posteriori logo queremos saber o ponto:

$$P(C_1|x) = P(C_2|x) \Rightarrow \frac{P(x|C_1)P(C_1)}{P(x)} = \frac{P(x|C_2)P(C_2)}{P(x)}$$

$$\frac{1}{\sqrt{2\pi}} e^{-\frac{(x+1)^2}{2}} \cdot 0,4 = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2}} \cdot 0,3$$

$$\ln\left(e^{-\frac{(x+1)^2}{2}}\right) + \ln 0,4 = \ln\left(e^{-\frac{(x-1)^2}{2}}\right) + \ln 0,3$$

$$-\frac{(x+1)^2}{2} \ln e + \ln 0,4 = -\frac{(x-1)^2}{2} \ln e + \ln 0,3$$

$$-x^2 + 2x - 1 + 2\ln 0,4 = -x^2 + 2x - 1 + 2\ln 0,3$$

$$-4x = 2\ln(0,3/0,4)$$

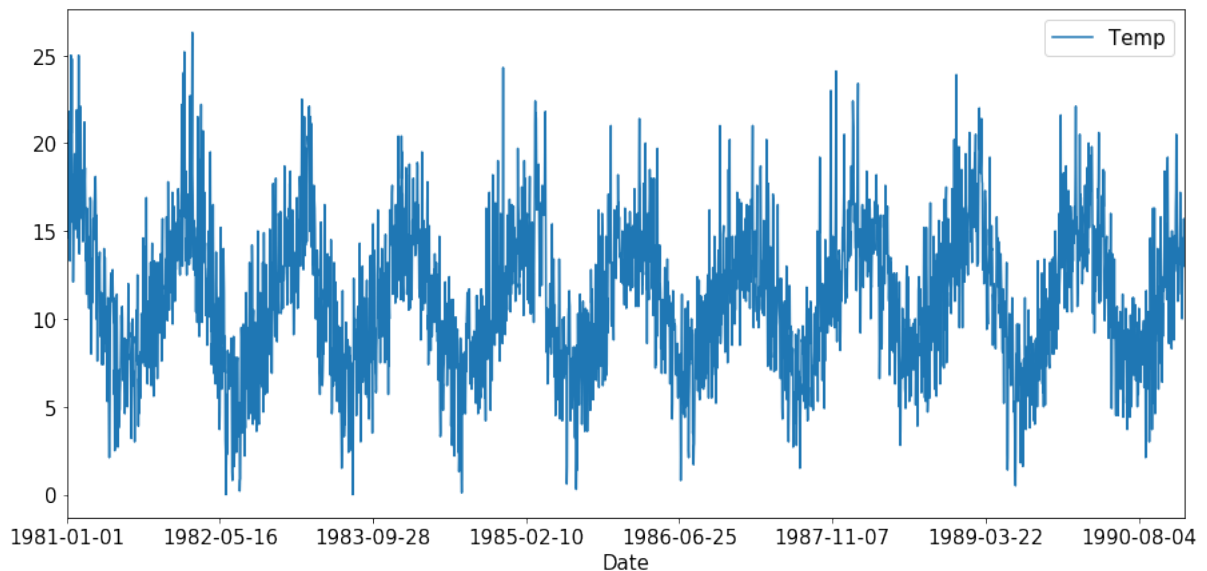
$$x = -\ln\left(\frac{0,3}{0,4}\right)^{1/2} = \ln\left(\frac{4}{3}\right)^{1/2}$$

Logo se $x < \ln(4/3)^{1/2}$ então $x \in C_1$ e se
 $x > \ln(4/3)^{1/2}$ então $x \in C_2$.

2 Exercício Computacionais

A linguagem utilizada para a realização dos exercícios foi o Python, com a ferramenta chamada Jupyter Notebook. Os dados da série temporal são constituídos pelas temperaturas diárias mínimas em uma determinada cidade desde 1981 até 1990 (um total de 3650 linhas de dados) e têm as seguintes características:

Temperatura Máxima	26, 30°C
Temperatura Mínima	0°C
Tempratura Média	11, 17°C



2.1 Exercício 1

2.1.1 Letra a

Esta série temporal será tratada como um problema de aprendizado de máquina supervisionado. Para isto iremos utilizar como saída a temperatura no dia e como variáveis de entrada as temperaturas nos K dias anteriores.

Desta forma foram gerados 30 colunas novas com os dados das temperaturas dos 30 dias anteriores. Como para os 30 primeiros dias não era possível saber a temperatura em todos os 30 dias anteriores, esses dados foram eliminados. Também foi adicionada uma coluna com valor constante igual a 1, esta coluna será utilizada para estimar o termo independente da regressão linear.

Os dados foram então separados em dois conjuntos: Treino e Teste. Os dados do ano de 1991 compuseram o conjunto de Teste e os dados dos demais anos fizeram parte do conjunto de Treino. Para esta parte do exercício, temos que separar nossos dados de treino em k -folds e fazer a regressão linear utilizando desde $K = 1$ atraso até 30 atrasos como variáveis de entrada.

Os dados de treino foram separados em k pastas através da adição de uma coluna que contém um número inteiro aleatório com distribuição uniforme entre 1 e k . Para cada k separamos as linhas com esse valor de k para Validação e as linhas com valores diferentes de k para Treino (note uma mudança no que queremos dizer por dados de treino).

Para cada um desses folders e para cada número de atrasos de entrada entre 1 e 30 (selecionando apenas as colunas correspondentes de variáveis, já que geramos as 30) foram geradas regressões lineares, através do critério do erro quadrático médio. Para cada k e cada K , avaliamos a raiz quadrada do erro quadrático médio de treino e validação. O valor final de erro de treino e validação para cada K foi definido como sendo a média em k dos erros. Para verificar se a implementação estava correta, os algoritmos implementados para cálculo da regressão linear e do erro quadrático médio foram comparados com os métodos prontos do pacote Scikit.

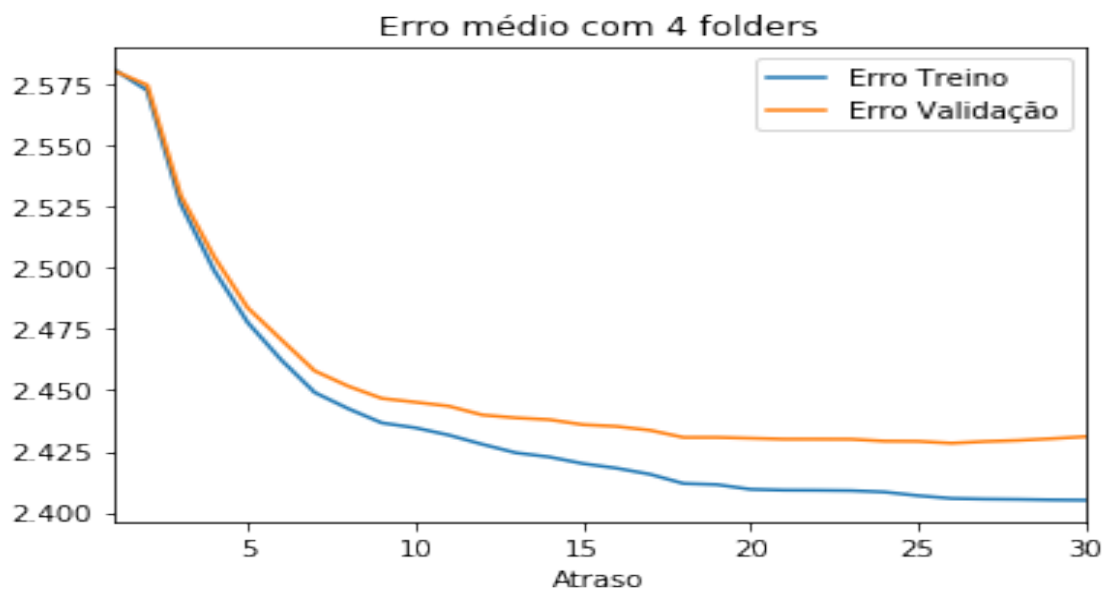
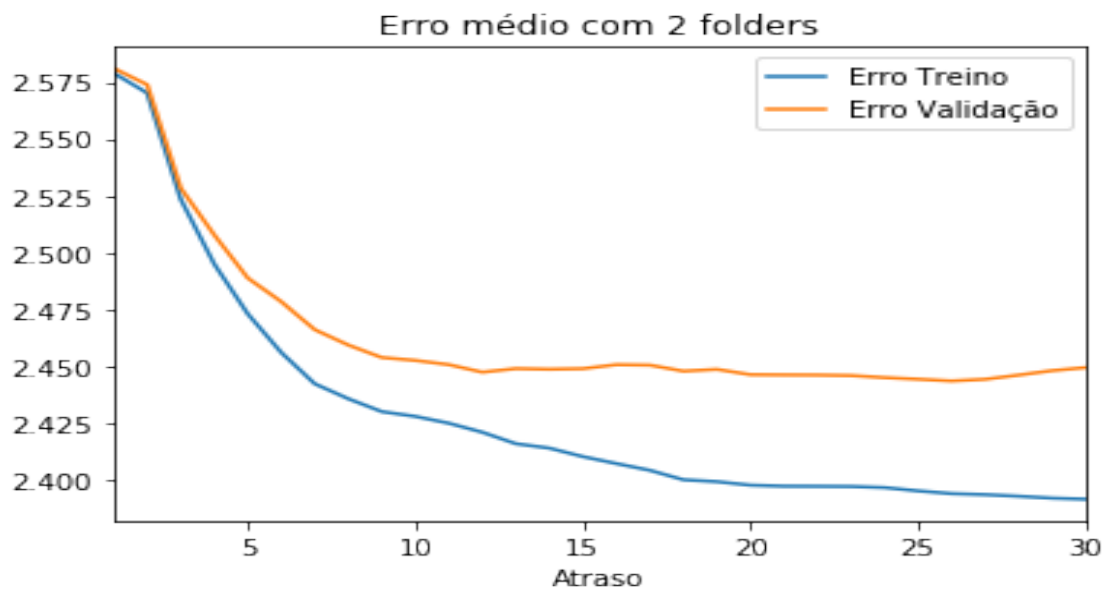
Os dados foram treinados com $k \in \{2, 4, 8\}$, para cada k o valor do K ótimo foi escolhido como o que gerava o menor erro de validação médio nos folders.

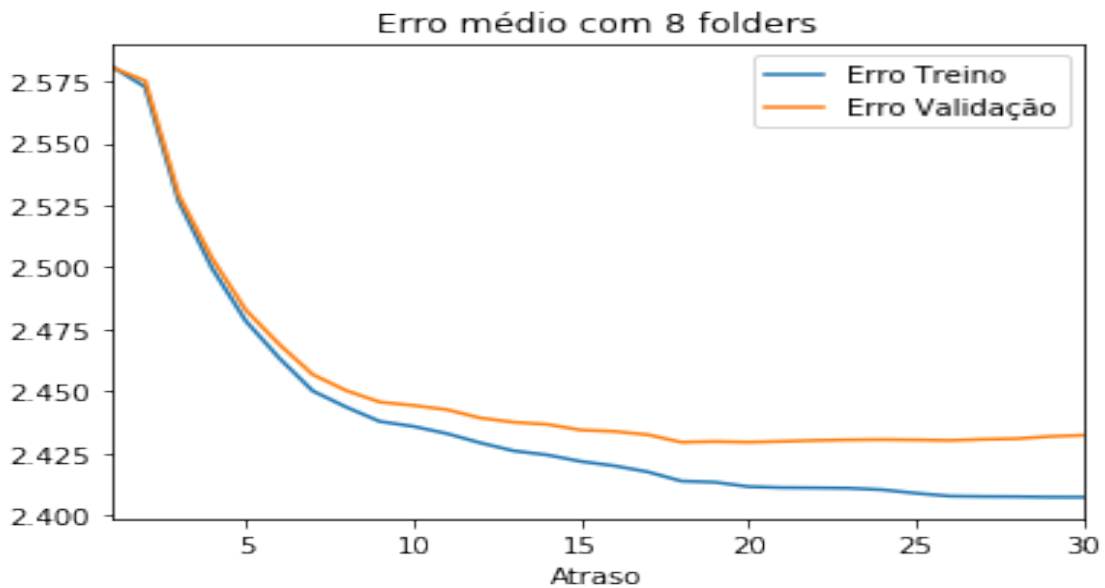
Em uma aplicação real não deveríamos utilizar os dados de teste para fazer escolhas relacionadas ao modelo. Entretanto neste exercício foi feita uma verificação de como dois métodos diferentes para estimar os dados de teste a partir de um treino com k -fold se comportavam.

A partir do K ótimo para cada k , foram gerados dois modelos diferentes para estimar os dados de teste. O primeiro deles utiliza como saída a média dos k modelos e o segundo deles cria um modelo linear novo utilizando como conjunto de treino os dados de todos os folders. Desta forma obtivemos os seguintes resultados:

Folders	"K"Ótimo	Erro Treino	Erro Validação	Erro Teste Método 1	Erro Teste Método 2
2	26	2,3939	2,4435	2,2643	2,2645
4	26	2,4058	2,4283	2,2626	2,2645
8	18	2,4237	2,4239	2,2726	2,2725

E os seguintes gráficos com os erros de teste e validação para os folders:

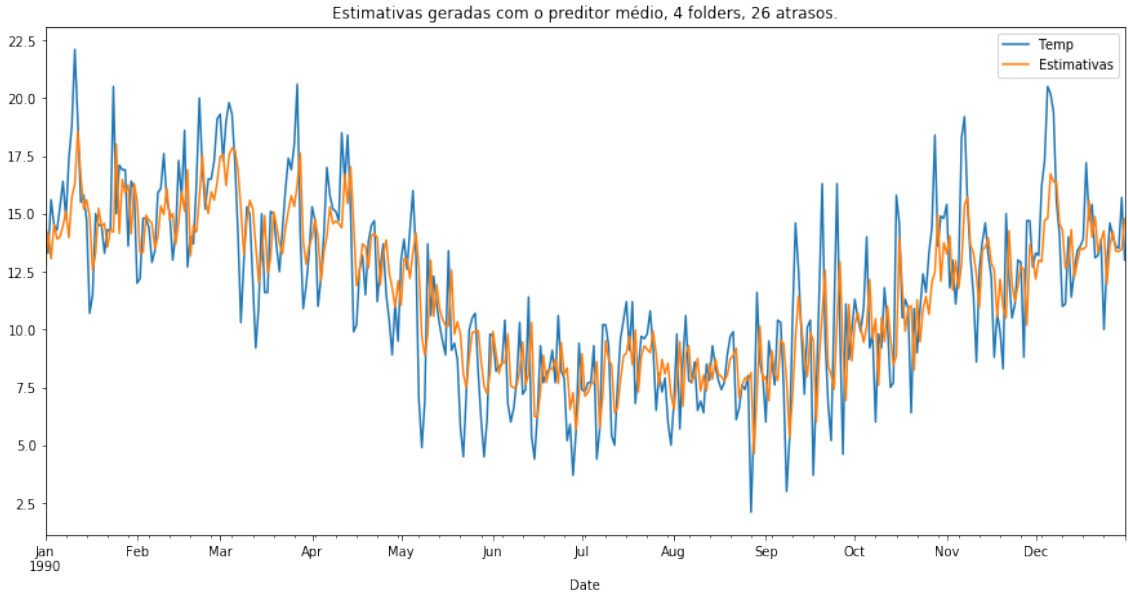




A primeira coisa a se notar nesses dados é que tivemos uma situação peculiar em que o erro de teste foi menor que os próprios erros de treino e validação. Não há também uma variação grande nos erros de teste entre os dois métodos logo não parece haver, neste caso, diferença em qual método usar. Por fim apesar de sutil, ao aumentarmos o número de folders nosso erro de teste aumentou um pouco. Talvez devido ao fato de aumentar o número de folders termos mais dados de treino e menos dados de validação, mesmo gerando mais modelos diferentes devido ao número de nosso k .

2.1.2 Letra b

Como obtivemos um melhor erro de teste para o valor de $k = 4$, utilizamos esse modelo que teve $K = 26$ como ótimo e utilizando como saída o método 1(média dos preditores) para gerar os resultados:



2.2 Exercício 2

Para este exercício deveríamos partir de um conjunto de dados com 5 atrasos, e gerar novas variáveis. Cada uma dessas variáveis novas é gerada pela combinação linear dessas 5 variáveis de atraso com pesos aleatórios e com a aplicação da tangente hiperbólica.

Como a tangente hiperbólica para valores muito grandes em módulo apresenta uma derivada muito pequena, os dados foram normalizados de forma que ao serem aplicados à tangente hiperbólica estivessem no intervalo $[-2,2]$. O processo de tratamento dos dados foi o seguinte. Dado um conjunto de variáveis de entrada com $K=5$ atrasos, e n amostras:

$$\begin{bmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(n) \end{bmatrix} = \begin{bmatrix} x_1(1) & x_2(1) & x_3(1) & x_4(1) & x_5(1) \\ x_1(2) & x_2(2) & x_3(2) & x_4(2) & x_5(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(n) & x_2(n) & x_3(n) & x_4(n) & x_5(n) \end{bmatrix}$$

Como esses dados de entrada são oriundos da temperatura, a primeira transformação aplicada sobre eles foi:

$$x_i(n) \leftarrow \frac{(x_i(n) - T_{min})}{(T_{max} - T_{min})} * 4 - 2 \quad (2.1)$$

Onde T_{min} e T_{max} são as temperaturas máxima e mínima da série, desta forma os valores $x_i(n)$ pertencem ao intervalo $[-2,2]$.

Em seguida gerou-se $T = 1, \dots, T = 100$ vetores $\mathbf{w}_T = (w_{T,1}, w_{T,2}, w_{T,3}, w_{T,4}, w_{T,5})^T$, com valores aleatórios em uma distribuição uniforme com valores no intervalo $[0,1]$. O

valor 1 foi escolhido ao acaso e poderia ser substituído por qualquer valor pois iremos aplicar a seguinte transformação sobre cada um desses vetores:

$$\mathbf{w}_T \leftarrow \frac{\mathbf{w}_T}{\sum_{k=1}^5 w_{T,k}} \quad (2.2)$$

Assim prosseguiu-se a geração das variáveis para nosso modelo da seguinte forma:

$$\begin{aligned} \begin{bmatrix} z_1(1) & z_2(1) & \cdots & z_{100}(1) \\ z_1(2) & z_2(2) & \cdots & z_{100}(2) \\ \vdots & \vdots & \ddots & \vdots \\ z_1(n) & z_2(n) & \cdots & z_{100}(n) \end{bmatrix} &= \quad (2.3) \\ &= \begin{bmatrix} x_1(1) & x_2(1) & x_3(1) & x_4(1) & x_5(1) \\ x_1(2) & x_2(2) & x_3(2) & x_4(2) & x_5(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(n) & x_2(n) & x_3(n) & x_4(n) & x_5(n) \end{bmatrix} \begin{bmatrix} w_{1,1} & w_{2,1} & \cdots & w_{100,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{100,2} \\ w_{1,3} & w_{2,3} & \cdots & w_{100,3} \\ w_{1,4} & w_{2,4} & \cdots & w_{100,4} \\ w_{1,5} & w_{2,5} & \cdots & w_{100,5} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^5 x_i(1)w_{1,i} & \sum_{i=1}^5 x_i(1)w_{1,2} & \cdots & \sum_{i=1}^5 x_i(1)w_{100,2} \\ \sum_{i=1}^5 x_i(2)w_{1,i} & \sum_{i=1}^5 x_i(2)w_{1,2} & \cdots & \sum_{i=1}^5 x_i(2)w_{100,2} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^5 x_i(n)w_{1,i} & \sum_{i=1}^5 x_i(n)w_{1,2} & \cdots & \sum_{i=1}^5 x_i(n)w_{100,2} \end{bmatrix} \end{aligned}$$

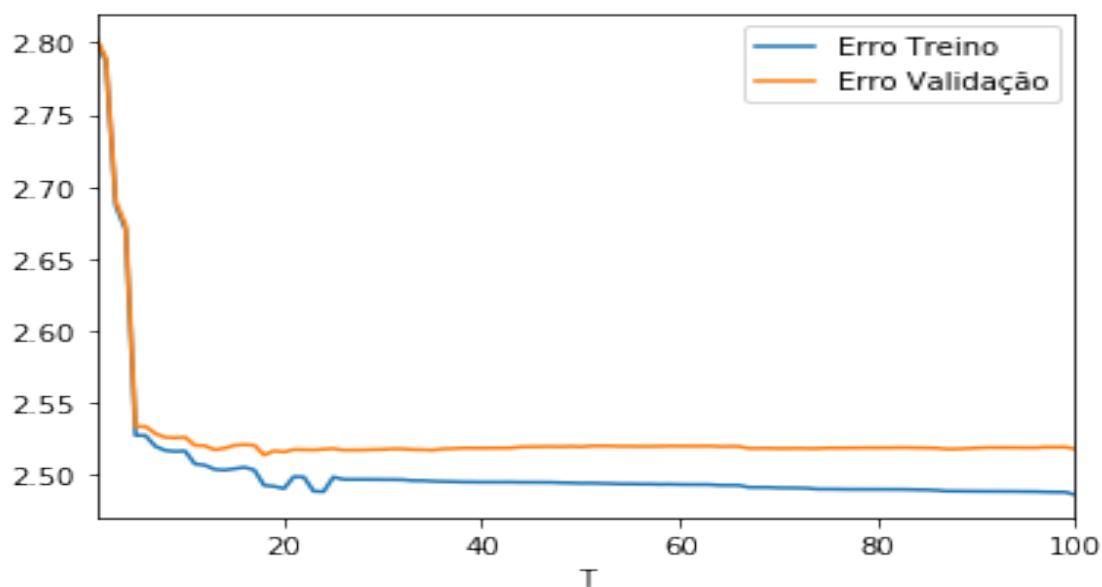
Através das normalizações 2.1 e 2.2 garante-se que nossos $z_i(n)$ pertencem ao intervalo $[-2, 2]$, por fim aplicou-se a tangente hiperbólica a estes valores:

$$z_i(n) \leftarrow \tanh(z_i(n))$$

O modelo então foi treinado utilizando a técnica de Ridge Regression tendo como critério o erro quadrático médio e com $k = 4$ folds. Para $T = 1, \dots, 100$ e $\lambda = 0, 2^{-10}, 2^{-9}, \dots, 2^{10}$, para cada T e λ o erro de validação considerado foi o erro médio da raiz do erro quadrático médio entre as pastas do fold.

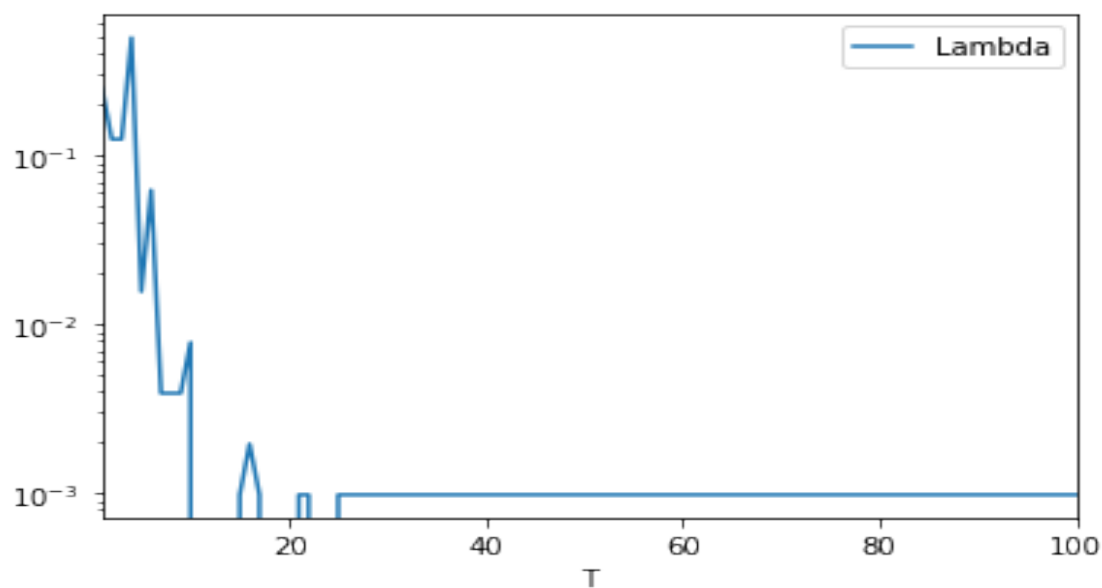
2.2.1 Letra a

Para cada T foi escolhido o λ que gera o menor erro, assim obtêm-se um gráfico do erro de treino e validação em função de T :



2.2.2 Letra b

E o gráfico do λ que gerou o menor erro de validação para cada T :



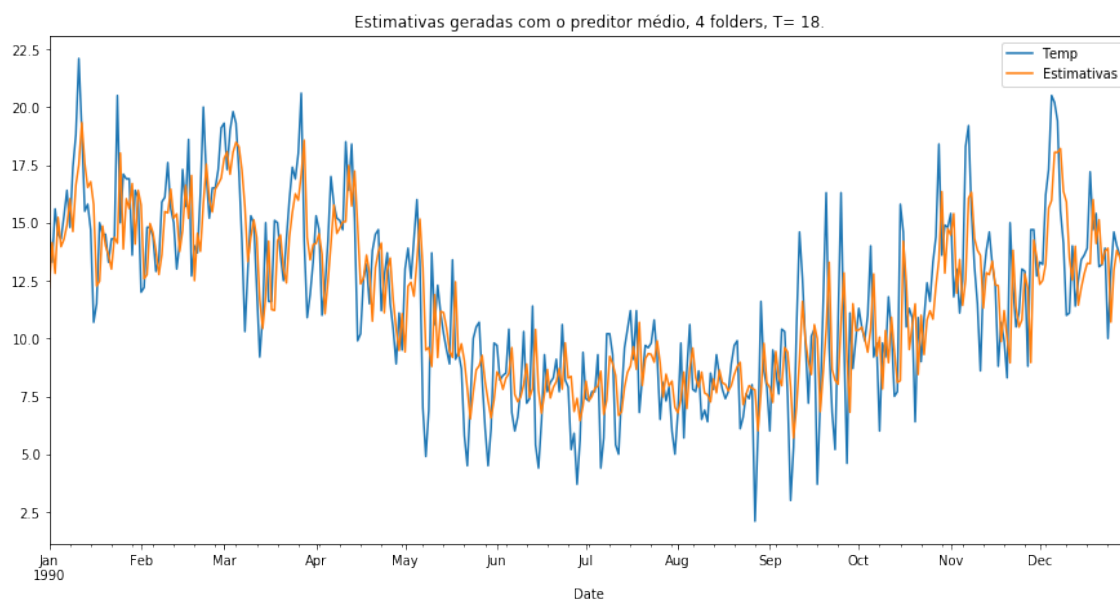
Neste gráfico vemos que os valores de λ que geraram o menor erro foram em sua maioria iguais as zeros.

2.2.3 Letra c

A partir dos testes realizados o menor erro de validação foi obtido com $T = 18$ e $\lambda = 0$:

Erro	Erro
Treino	Validação
2.4929	2.5141

Os 4 preditores para $T = 18$ e $\lambda = 0$, foram combinados através de uma média para gerar a seguinte previsão com erro de teste igual a 2,3268.



Anexos

ANEXO A – Código para exercício 1

```

1 # pacotes e configuracoes gerais
2 import pandas as pd
3 import matplotlib as ml
4 import numpy as np
5 import math as mt
6 from sklearn.linear_model import LinearRegression
7 from sklearn.metrics import mean_squared_error
8 pd.set_option('display.max_rows',500)
9 pd.set_option('display.max_columns',20)
10
11 # Importacao dos dados e verificacao dos dados
12 data=pd.read_csv("C:/Users/Roger/OneDrive/ESPECIAL/IA006/
    python/daily-minimum-temperatures.csv")
13 data.dtypes
14 data=data.sort_values('Date')
15 data['Date'] = pd.to_datetime(data['Date'])
16 data['Year']=data['Date'].dt.year
17 data=data[['Year','Date','Temp']]
18
19 #Transforma os dados em um problema supervisionado
20 data_windowed=data.copy()
21 data_windowed['C']=1
22 for k in range(1,31) :
23     colum_name='T-'+str(k)
24     data_windowed[colum_name]=data['Temp'].shift(k)
25 #elimina os 30 primeiros dias q nao ha como prever
26 data_windowed = data_windowed.iloc[30:]
27 data_windowed.describe()
28 data_windowed.head()
29
30 #configuracoes do método
31 ano_teste=1990
32 atrasos=30
33 lista_folders=[2]
34 #gera os dados de treino e teste

```

```
35 data_train=data_windowed[data_windowed['Year']!=ano_teste].  
    copy()  
36 data_test=data_windowed[data_windowed['Year']==ano_teste].  
    copy()  
37 #inicia os testes  
38 for folders in lista_folders:  
39     #gerando uma coluna com o numero de folds a partir de uma  
        distribuiÃ§Ã£o normal e inteiros  
40     np.random.seed(9001)  
41     data_train['fold'] = np.random.randint(1,folders+1,size=  
        len(data_train))  
42     for f in range(1,folders+1):  
43         lenght=len(data_train[data_train['fold']==f])  
44         linha=0  
45         resultados=pd.DataFrame(columns=['Atraso','Folder','Dado  
            Treino','Dado ValidaÃ§Ã£o','Matrix','Erro Treino','  
            Erro ValidaÃ§Ã£o','Erro Teste'])  
46         y_tf=data_test.iloc[:,2:3].copy()  
47         for k in range(1,atrasos+1):  
48             #dados treino globais  
49             x_tf=data_test.iloc[:,3:k+4].copy()  
50             for f in range(1,folders+1):  
51                 linha+=1  
52                 resultados.loc[linha,'Atraso']=k  
53                 resultados.loc[linha,'Folder']=f  
54                 #Dados treino no folder  
55                 x_train=data_train[data_train['fold']!=f]  
56                 x_train=x_train.iloc[:,3:k+4].copy()  
57                 y_train=data_train[data_train['fold']!=f]  
58                 y_train=y_train.iloc[:,2:3].copy()  
59                 resultados.loc[linha,'Dado Treino']=len(x_train)  
60                 #Dados validaÃ§Ã£o no folder  
61                 x_valid=data_train[data_train['fold']==f]  
62                 x_valid=x_valid.iloc[:,3:k+4].copy()  
63                 y_valid=data_train[data_train['fold']==f]  
64                 y_valid=y_valid.iloc[:,2:3].copy()  
65                 resultados.loc[linha,'Dado ValidaÃ§Ã£o']=len(  
                    x_valid)  
66                 #calcula da inversa
```

```
67         xtx=(x_train.transpose()).dot(x_train)
68         rank=np.linalg.matrix_rank(xtx.values)
69         if rank==(k+1):
70             message="Sim"
71         else:
72             message="Não"
73         resultados.loc[linha,'Matrix']=message
74         #cálculo do vetor w para regressão
75         xtxinv=pd.DataFrame(np.linalg.pinv(xtx.values),
76                             xtx.columns, xtx.index)
76         w=(xtxinv.dot(x_train.transpose())).dot(y_train)
77         #erro treino
78         yh_train=x_train.dot(w)
79         error_train=mt.sqrt(((yh_train-y_train)**2).mean
80                             ())
80         resultados.loc[linha,'Erro Treino']=error_train
81         #erro validação
82         yh_valid=x_valid.dot(w)
83         error_valid=mt.sqrt(((yh_valid-y_valid)**2).mean
84                             ())
84         resultados.loc[linha,'Erro Validação']=
85             error_valid
85         #erro teste
86         yh_tf=x_tf.dot(w)
87         error_tf=mt.sqrt(((yh_tf-y_tf)**2).mean())
88         resultados.loc[linha,'Erro Teste']=error_tf
```

ANEXO B – Código para exercício 2

```
1 import pandas as pd
2 import matplotlib as ml
3 import numpy as np
4 import math as mt
5 pd.set_option('display.max_rows',500)
6 pd.set_option('display.max_columns',50)
7
8 data=pd.read_csv("C:/Users/Roger/OneDrive/ESPECIAL/IA006/
   python/daily-minimum-temperatures.csv")
9 # data.head(20)
10 #data.tail(20)
11 # data.isna()
12 #data.dtypes
13 data=data.sort_values('Date')
14 data['Date'] = pd.to_datetime(data['Date'])
15 data['Year']=data['Date'].dt.year
16 data=data[['Year','Date','Temp']]
17
18 #normalizando a temperatura para ficar dentro do range [-2,2]
19 tmin=data.Temp.min()
20 tmax=data.Temp.max()
21 data_windowed=data.copy()
22 data_windowed['TempNorm']=(data['Temp']-tmin)/(tmax-tmin)*4-2
23
24 #gerando os 5 atrasos
25 for k in range(1,6) :
26     colum_name='T-'+str(k)
27     data_windowed[colum_name]=data_windowed['TempNorm'].
        shift(k)
28 data_windowed=data_windowed.iloc[5:].copy()
29 #criando os dados temporarios
30 x_temp=data_windowed.iloc[:,4:9]
31 y_temp=data_windowed.iloc[:,2:3]
32 date_temp=data_windowed.iloc[:,0:2]
33 #resetando o index para facilitar mais abaixo
```

```
34 x_temp.index = pd.RangeIndex(len(x_temp.index))
35 y_temp.index = pd.RangeIndex(len(y_temp.index))
36 date_temp.index = pd.RangeIndex(len(date_temp.index))
37 print(x_temp.describe())
38 print(y_temp.head())
39 print(date_temp.head())
40
41 #Criando o vetor w
42 w_temp=pd.DataFrame(columns=['w'])
43 w=pd.DataFrame(columns=['w1'])
44 variable_list=[]
45 np.random.seed(230482)
46 for t in range(0,100):
47     #criando o vetor w
48     for l in range(0,5):
49         w_temp.loc[l,'w']=np.random.uniform(0, 1)
50     #Normalizando o vetor w para que sua soma seja 1
51     w_temp['w']=w_temp['w']/w_temp['w'].sum()
52     colum_name='w'+str(t+1)
53     variable_name='xl_'+str(t+1)
54     variable_list.append(variable_name)
55     w[colum_name]=w_temp['w']
56 # vetor normalizado entre -2 e 2
57 x_a=pd.DataFrame(np.float_(np.dot(x_temp,w)),columns=
    variable_list)
58 print(x_a.describe())
59 x=pd.DataFrame(np.tanh(np.float_(np.dot(x_temp,w))),columns=
    variable_list)
60
61 modified_data=date_temp.copy()
62 modified_data['Temp']=y_temp['Temp']
63 modified_data['C']=1
64 for h in list(x):
65     modified_data[h]=x[h]
66
67 ano_teste=1990
68 atrasos=100
69 lista_folders=[4]
70 # lista_lambda=[0,]
```

```

71 lista_lambda
    =[0,0.000976563,0.001953125,0.00390625,0.0078125,0.015625,0.03125,0.0625,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,65536,131072,262144,524288,1048576,2097152,4194304,8388608,16777216,33554432,67108864,134217728,268435456,536870912,1073741824,2147483648,4294967296,8589934592,17179869184,34359738368,68719476736,137438953472,274877906944,549755813888,1099511627776,2199023255552,4398046511104,8796093022208,17592186044416,35184372088832,70368744177664,140737488355328,281474976710656,562949953421312,1125899906842624,2251799813685248,4503599627370496,9007199254740992,18014398509481984,36028797018963968,72057594037927936,144115188075855872,288230376151711744,576460752303423488,1152921504606846976,2305843009213693952,4611686018427387904,9223372036854775808,18446744073709551616,36893488147419103232,73786976294838206464,147573952589676412928,295147905179352825856,590295810358705651712,1180591620717411303424,2361183241434822606848,4722366482869645213696,9444732965739290427392,18889465931478580854784,37778931862957161709568,75557863725914323419136,151115727451828646838272,302231454903657293676544,604462909807314587353088,1208925819614629174706176,2417851639229258349412352,4835703278458516698824704,9671406556917033397649408,19342813113834066795298816,38685626227668133590597632,77371252455336267181195264,154742504910672534362390528,309485009821345068724781056,618970019642690137449562112,1237940039285380274899124224,2475880078570760549798248448,4951760157141521099596496896,9903520314283042199192993792,19807040628566084398385987584,39614081257132168796771975168,79228162514264337593543950336,158456325028528675187087900672,316912650057057350374175801344,633825300114114700748351602688,1267650600228229401496703205376,2535301200456458802993406410752,5070602400912917605986812821504,10141204801825835211973625643008,20282409603651670423947251286016,40564819207303340847894502572032,81129638414606681695789005144064,162259276829213363391578010288128,324518553658426726783156020576256,649037107316853453566312041152512,1298074214633706907132624082305024,2596148429267413814265248164610048,5192296858534827628530496329220096,10384593717069655257060992658440192,20769187434139310514121985316880384,41538374868278621028243970633760768,83076749736557242056487941267521536,166153499473114484112975882535043072,332306998946228968225951765070086144,664613997892457936451903530140172288,1329227995784915872903807060280344576,2658455991569831745807614120560689152,5316911983139663491615228241121378304,10633823966279326983230456482242756608,21267647932558653966460912964485513216,42535295865117307932921825928971026432,85070591730234615865843651857942052864,170141183460469231731687303715884105728,340282366920938463463374607431768211456,680564733841876926926749214863536422912,1361129467683753853853498429727072845824,2722258935367507707706996859454145691648,5444517870735015415413993718908291383296,10889035741470030830827987437816582766592,21778071482940061661655974875633165533184,43556142965880123323311949751266331066368,87112285931760246646623899502532662132736,174224571863520493293247799005065324265472,348449143727040986586495598010130648530944,696898287454081973172991196020261297061888,1393796574908163946345982392040522594123776,2787593149816327892691964784081045188247552,5575186299632655785383929568162090376495104,11150372599265311570767859136324180752990208,22300745198530623141535718272648361505980416,44601490397061246283071436545296723011960832,89202980794122492566142873090593446023921664,178405961588244985132285746181186892047843328,356811923176489970264571492362373784095686656,713623846352979940529142984724747568191373312,1427247692705959881058285969449495136382746624,2854495385411919762116571938898990272765493248,5708990770823839524233143877797980545530986496,11417981541647679048466287755595961091061972992,22835963083295358096932575511191922182123945984,45671926166590716193865151022383844364247891968,91343852333181432387730302044767688728495783936,182687704666362864775460604089535377456991567872,365375409332725729550921208179070754913983135744,730750818665451459101842416358141509827966271488,1461501637330902918203684832716283019655932542976,2923003274661805836407369665432566039311865085952,5846006549323611672814739330865132078623730171904,11692013098647223345629478661730264157247460343808,23384026197294446691258957323460528314494920687616,46768052394588893382517914646921056628989841375232,93536104789177786765035829293842113257979682750464,187072209578355573530071658587684226515959365500928,374144419156711147060143317175368453031918731001856,748288838313422294120286634350736906063837462003712,1496577676626844588240573268701473812127674924007424,2993155353253689176481146537402947624255349848014848,5986310706507378352962293074805895248510699696029696,11972621413014756705924586149611790497021399392059392,23945242826029513411849172299223580994042798784118784,47890485652059026823698344598447161988085597568237568,95780971304118053647396689196894323976171195136475136,191561942608236107294793378393788647952342390272950272,383123885216472214589586756787577295904684780545900544,766247770432944429179173513575154591809369561091801088,1532495540865888858358347027150309183618739122183602176,3064991081731777716716694054300618367237478244367204352,6129982163463555433433388108601236734474956488734408704,12259964326927110866866776217202473468949912977468817408,24519928653854221733733552434404946937899825954937634816,49039857307708443467467104868809893875799651909875269632,98079714615416886934934209737619787751599303819750539264,196159429230833773869868419475239575503198607639501078528,392318858461667547739736838950479151006397215279002157056,784637716923335095479473677900958302012794430558004314112,1569275433846670190958947355801916604025588861116008628224,3138550867693340381917894711603833208051177722232017256448,6277101735386680763835789423207666416102355444464034512896,12554203470773361527671578846415332832204710888928069025792,25108406941546723055343157692830665664409421777856138051584,50216813883093446110686315385661331328818843555712276103168,100433627766186892221372630771322662657637687111424552206336,200867255532373784442745261542645325315275374222849104412672,401734511064747568885490523085290650630550748445698208825344,803469022129495137770981046170581301261101496891396417650688,1606938044258990275541962092341162602522202993782792835301376,3213876088517980551083924184682325205044405987565585670602752,6427752177035961102167848369364650410088811975131171341205504,12855504354071922204335696738729300820177623950262342682411008,25711008708143844408671393477458601640355247900524685364822016,51422017416287688817342786954917203280710495801049370729644032,102844034832575377634685573909834406561420991602098741459288064,205688069665150755269371147819668813122841983204197482918576128,411376139330301510538742295639337626245683966408394965837152256,822752278660603021077484591278675252491367932816789931674304512,1645504557321206042154969182557350504982735865633579863348609024,3291009114642412084309938365114701009965471731267159726697218048,6582018229284824168619876730229402019930943462534319453394436096,13164036458569648337239753460458804039861886925068638906788872192,26328072917139296674479506920917608079723773850137277813577744384,52656145834278593348959013841835216159447547700274555627155488768,105312291668557186697918027683670432318895095400549111254310977536,210624583337114373395836055367340864637790190801098222508621955072,421249166674228746791672110734681729275580381602196445017243910144,842498333348457493583344221469363458551160763204392890034487820288,1684996666696914987166688442938726917102321526408785780068975640576,3369993333393829974333376885877453834204643052817571560137951281152,6739986666787659948666753771754907668409286105635143120275902562304,13479973333575319897333507543509815336818572211270286240551805124608,26959946667150639794667015087019630673637144422540572481103610249216,53919893334301279589334030174039261347274288845081144962207220498432,107839786668602559178668060348078522694548577690162289924414440996864,215679573337205118357336120696157045389097155380324579848828881993728,431359146674410236714672241392314090778194310760649159697657763987456,862718293348820473429344482784628181556388621521298319395315527974912,1725436586697640946858688965569256363112777243042596638790631055949824,3450873173395281893717377931138512726225554486085193277581262111899648,6901746346790563787434755862277025452451108972170386555162524223799296,13803492693581127574869511724554050904902217944340773110325048447598592,27606985387162255149739023449108101809804435888681546220650096895197184,55213970774324510299478046898216203619608871777363092441300193790394368,110427941548649020598956093796432407239217743554726184882600387580788736,220855883097298041197912187592864814478435487109452369765200775161577472,441711766194596082395824375185729628956870974218904739530401550323154944,883423532389192164791648750371459257913741948437809479060803100646309888,1766847064778384329583297500742918515827483896875618958121606201292619776,3533694129556768659166595001485837031654967793751237916243212402585239552,7067388259113537318333190002971674063309935587502475832486424805170479104,14134776518227074636666380005943348126619871175004951664972849610340958208,28269553036454149273332760011886696253239742350009903329945699220681916416,56539106072908298546665520023773392506479484700019806659891398441363832832,113078212145816597093331040047546785012958969400039613319782796882727665664,226156424291633194186662080095093570025917938800079226639565593765455331328,452312848583266388373324160190187140051835877600158453279131187530910662656,904625697166532776746648320380374280103671755200316906558262375061821325312,1809251394333065553493296640760748560207343510400633813116524750123642650624,3618502788666131106986593281521497120414687020801267626233049500247285301248,7237005577332262213973186563042994240829374041602535252466099000494570602496,14474011154664524427946373126085988481658748083205070504932198000989141204992,28948022309329048855892746252171976963317496166410141009864396001978282409984,57896044618658097711785492504343953926634992332820282019728792003956564819968,115792089237316195423570985008687907853269984665640564039457584007913129639936,231584178474632390847141970017375815706539969331281128078915168015826259279872,463168356949264781694283940034751631413079938662562256157830336031652518559744,926336713898529563388567880069503262826159877325124512315660672063305037119488,1852673427797059126777135760139006525652319754650249024631321344126610074238976,3705346855594118253554271520278013051304639509300498049262642688253220148477952,7410693711188236507108543040556026102609279018600996098525285376506440296955904,14821387422376473014217086081112052205218558037201992197050570753012880593911808,29642774844752946028434172162224104410437116074403984394101141506025761187823616,59285549689505892056868344324448208820874232148807968788202283012051522375647232,118571099379011784113736688648896417641748464297615937576404566024103044751294464,237142198758023568227473377297792835283496928595231875152809132048206089502588928,474284397516047136454946754595585670566993857190463750305618264096412179005177856,948568795032094272909893509191171341133987714380927500611236528192824358010355712,1897137590064188545819787018382342682267975428761855001222473056385648716020711424,3794275180128377091639574036764685364535950857523710002444946112771297432041422848,7588550360256754183279148073529370729071901715047420004889892225542594864082845696,15177100720513508366558296147058741458143803430094840009779784451085189728165691392,30354201441027016733116592294117482916287606860189680019559568902170379456331382784,60708402882054033466233184588234965832575213720379360039119137804340758912662765568,121416805764108066932466369176469931665150427440758720078238275608681517825325531136,242833611528216133864932738352939863330300854881517440156476551217363035650651062272,485667223056432267729865476705879726660601709763034880312953102434726071301302124544,971334446112864535459730953411759453321203419526069760625906204869452142602604249088,1942668892
```

```

102     y_train=data_train[data_train['fold']!=f]
103     y_train=y_train.iloc[:,2:3].copy()
104     resultados.loc[linha,'Dado Treino']=len(
105         x_train)
106     #Dados validaÃ§Ã£o
107     x_valid=data_train[data_train['fold']==f]
108     x_valid=x_valid.iloc[:,3:k+4].copy()
109     y_valid=data_train[data_train['fold']==f]
110     y_valid=y_valid.iloc[:,2:3].copy()
111     resultados.loc[linha,'Dado ValidaÃ§Ã£o']=len(
112         x_valid)
113     #criaÃ§Ã£o da matrix identidade
114     i=np.identity(k+1)
115     i[0,0]=0
116     #calcula da inversa
117     xtx=(x_train.transpose()).dot(x_train)
118     rank=np.linalg.matrix_rank(xtx.values)
119     if rank==(k+1):
120         message="Sim"
121     else:
122         message="NÃ£o"
123     resultados.loc[linha,'Matrix']=message
124     #cÃ;lculo do vetor b para regressÃ£o
125     xtxinv=pd.DataFrame(np.linalg.pinv(xtx.values
126         +la*i), xtx.columns, xtx.index)
127     b=(xtxinv.dot(x_train.transpose())) dot(
128         y_train)
129     #erro treino
130     yh_train=x_train.dot(b)
131     error_train=mt.sqrt(((yh_train-y_train)**2).
132         mean())
133     resultados.loc[linha,'Erro Treino']=
134         error_train
135     #erro validaÃ§Ã£o
136     yh_valid=x_valid.dot(b)
137     error_valid=mt.sqrt(((yh_valid-y_valid)**2).
138         mean())
139     resultados.loc[linha,'Erro ValidaÃ§Ã£o']=
140         error_valid

```



```
133         #erro teste
134         yh_test=x_test.dot(b)
135         error_tf=mt.sqrt(((yh_test-y_test)**2).mean()
136         )
136         resultados.loc[linha,'Erro Teste']=error_tf
```