IA 006

LISTA DE EXERCÍCIOS 1

Roger Danilo Figlie

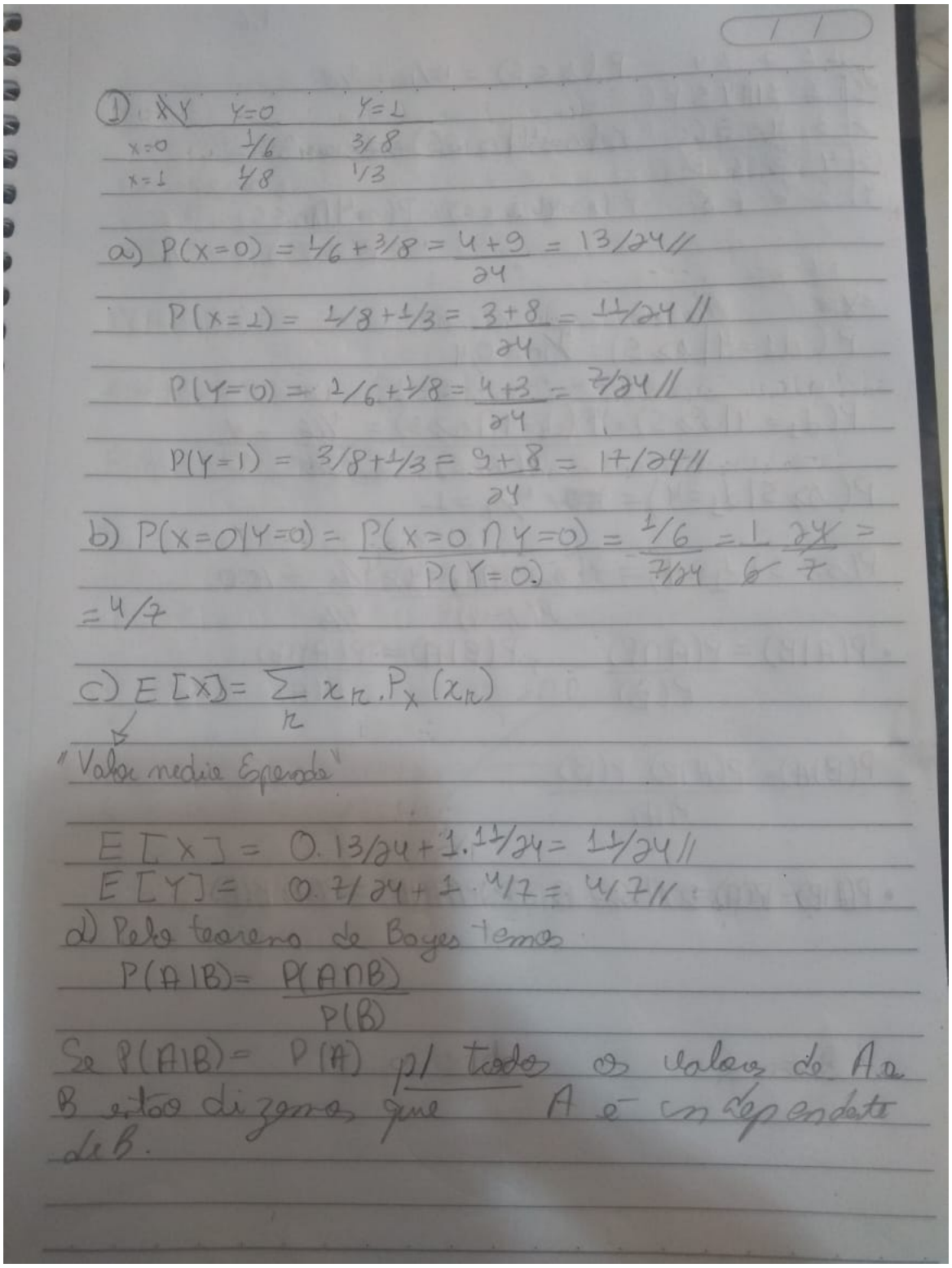
RA 189957

23/09/2019

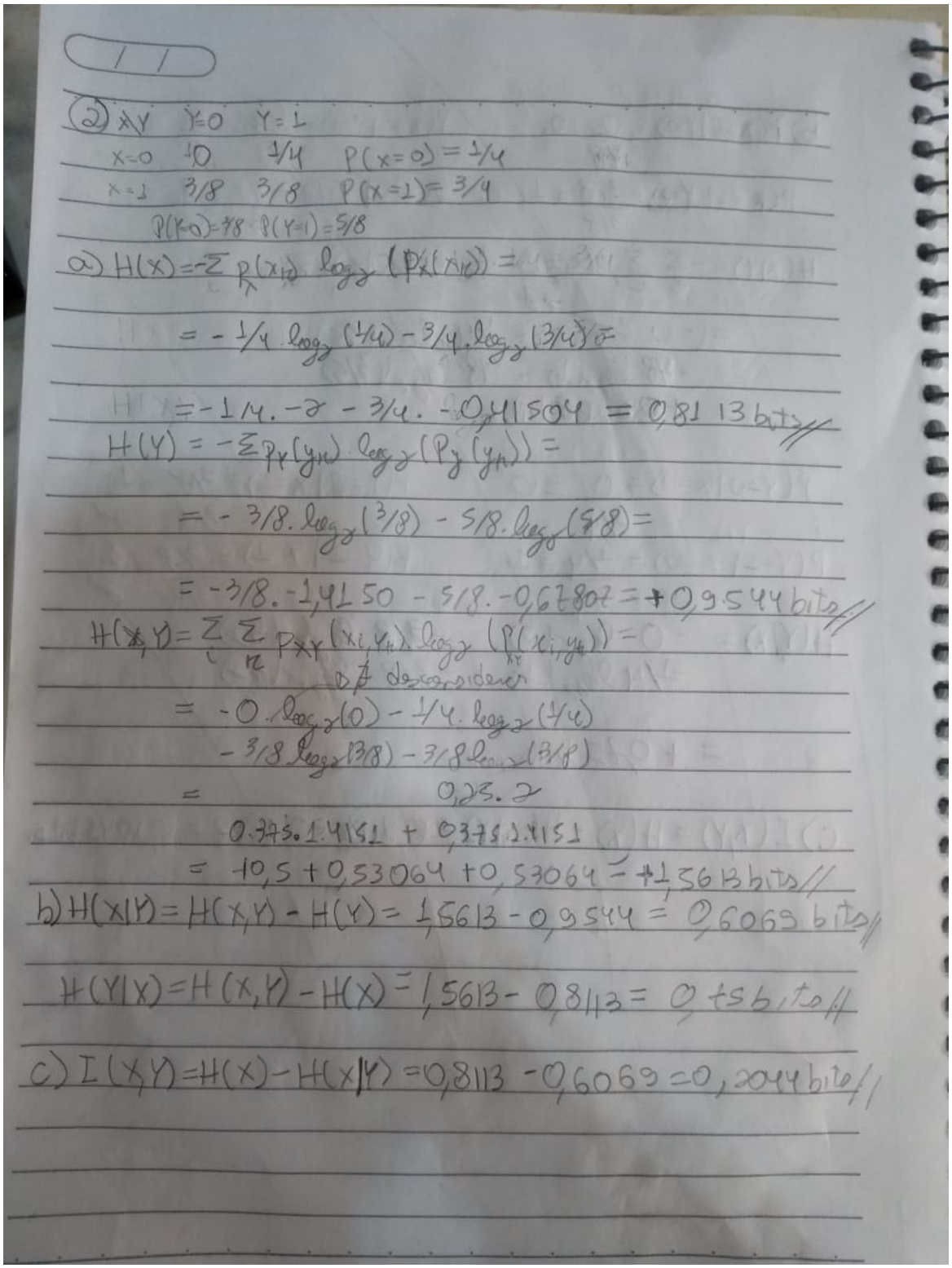
Sumário

1 Exercício Teóricos

1.1 Exercício 1



1.2 Exercício 2



1.3 Exercício 3

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

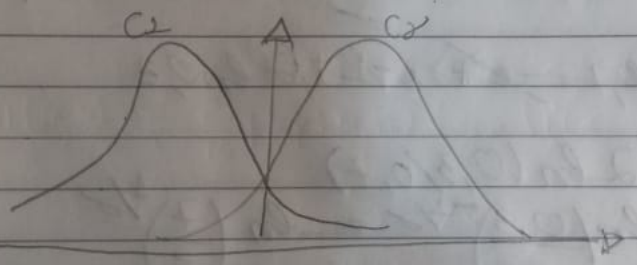
Logo precisamos achar o ponto em que $P(x|C_1) = P(x|C_2)$

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

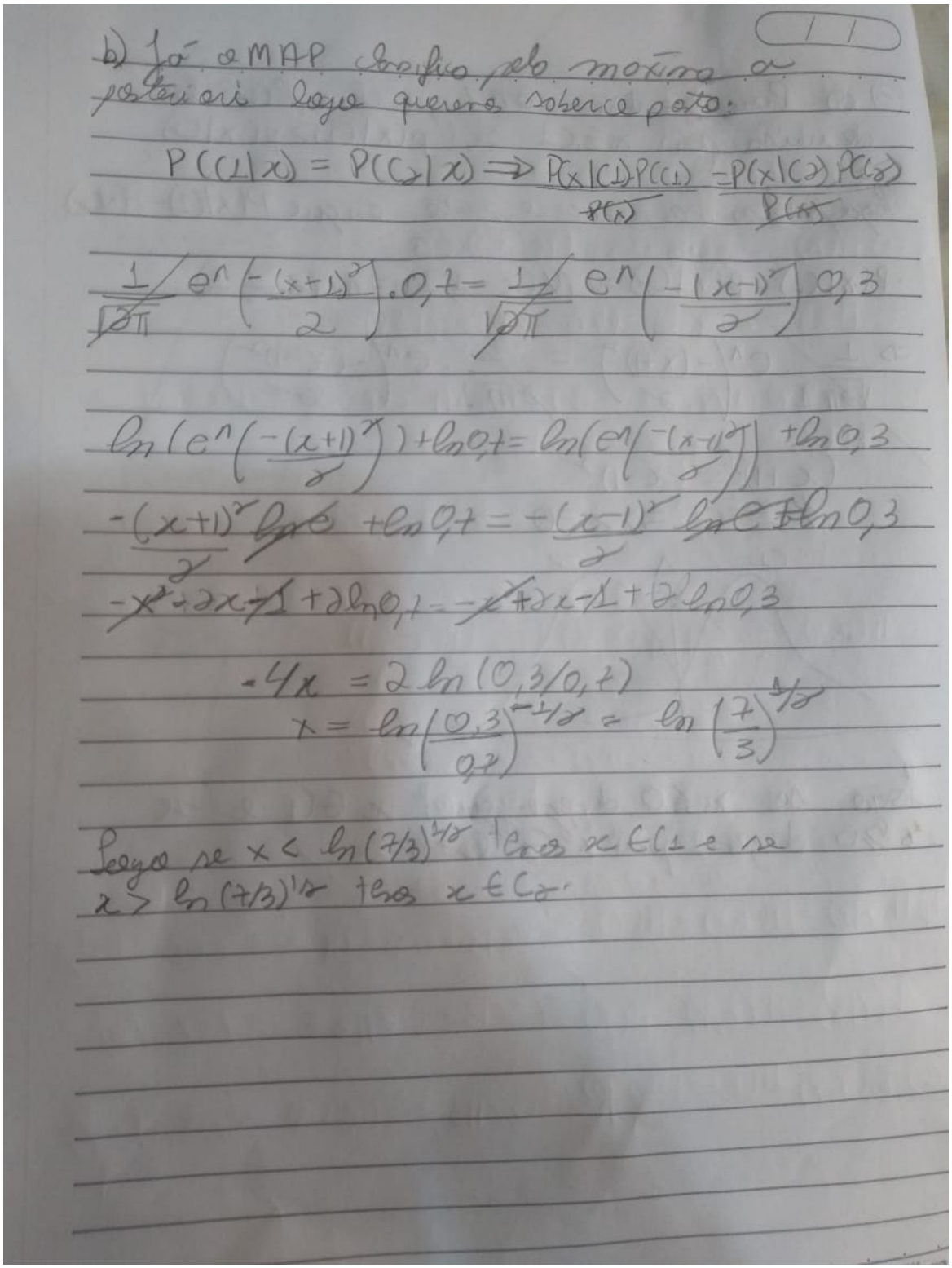
$$\Rightarrow \frac{1}{\sqrt{2\pi}\cdot 1} \cdot e^{\wedge}\left(\frac{-(x+1)^2}{2}\right) = \frac{1}{\sqrt{2\pi}\,1} \cdot e^{\wedge}\left(\frac{-(x-1)^2}{2}\right)$$

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

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



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



b) Já o MAP baseia pelo máximo a posteriori logo queremos saber o ponto:

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

$$\frac{1}{\sqrt{2\pi}} e^{\left(-\frac{(x+1)^2}{2}\right)} \cdot 0,7 = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{(x-1)^2}{2}\right)} 0,3$$

$$\ln\left(e^{\left(-\frac{(x+1)^2}{2}\right)}\right) + \ln 0,7 = \ln\left(e^{\left(-\frac{(x-1)^2}{2}\right)}\right) + \ln 0,3$$

$$-\frac{(x+1)^2}{2} \ln e + \ln 0,7 = -\frac{(x-1)^2}{2} \ln e + \ln 0,3$$

$$-x^2 - 2x - 1 + 2\ln 0,7 = -x^2 + 2x - 1 + 2\ln 0,3$$

$$-4x = 2\ln(0,3/0,7)$$

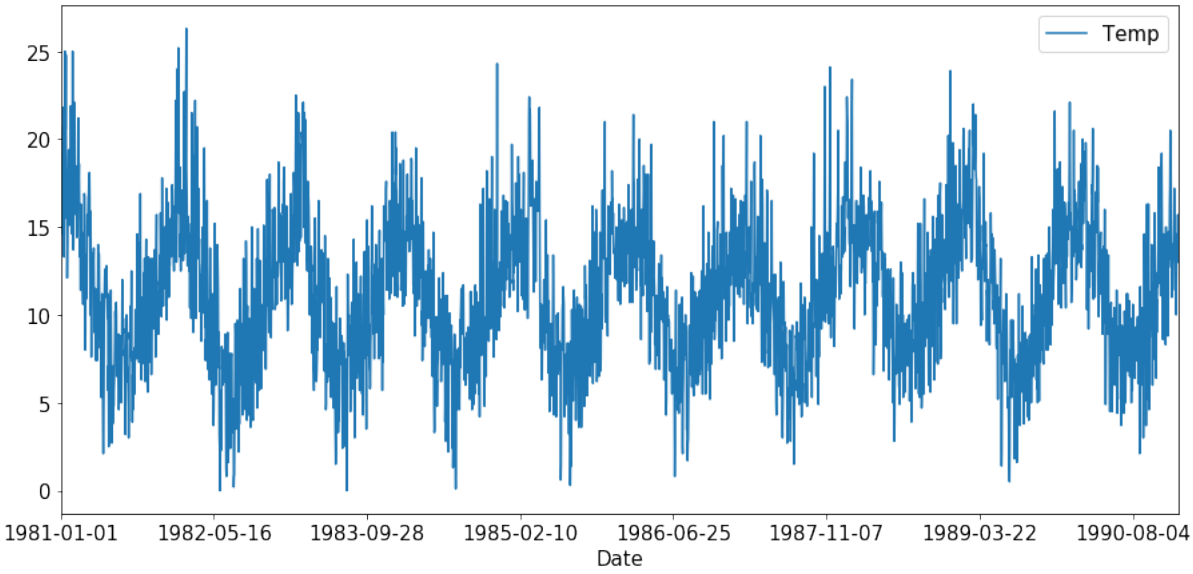
$$x = \ln\left(\frac{0,3}{0,7}\right)^{-1/2} = \ln\left(\frac{7}{3}\right)^{1/2}$$

Logo se $x < \ln(7/3)^{1/2}$ temos $x \in C_1$ e se $x > \ln(7/3)^{1/2}$ temos $x \in C_2$.

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 minímas 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 Minima	$0°C$
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 contêm 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.

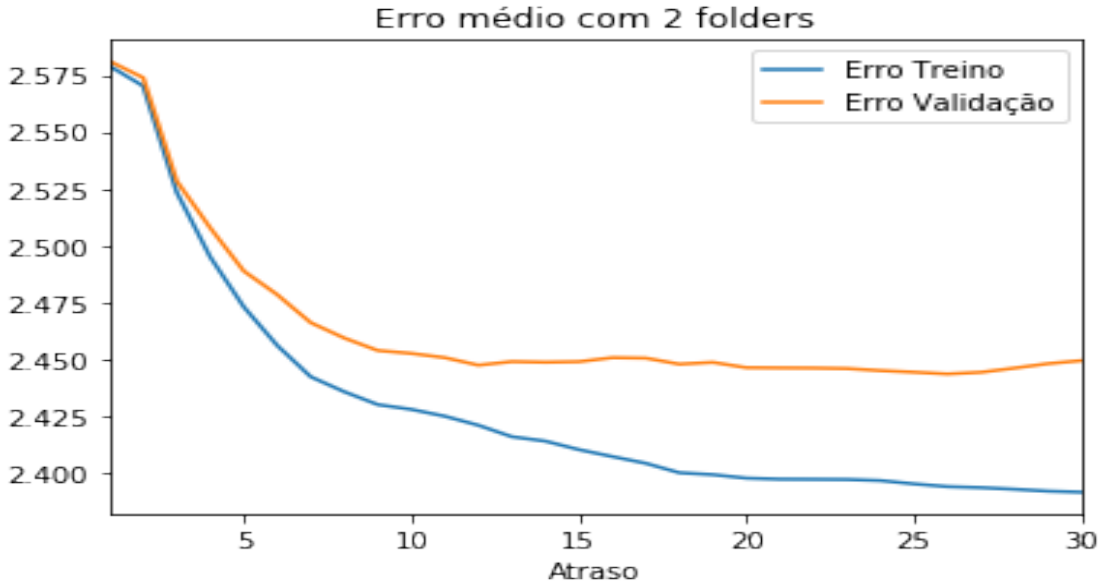
Os dados foram treinados com $k \in \{2, 4, 8\}$, para cada k o valor do K ótimo foi escolhido como o que gerava o menor erro de validação médio nos folders.

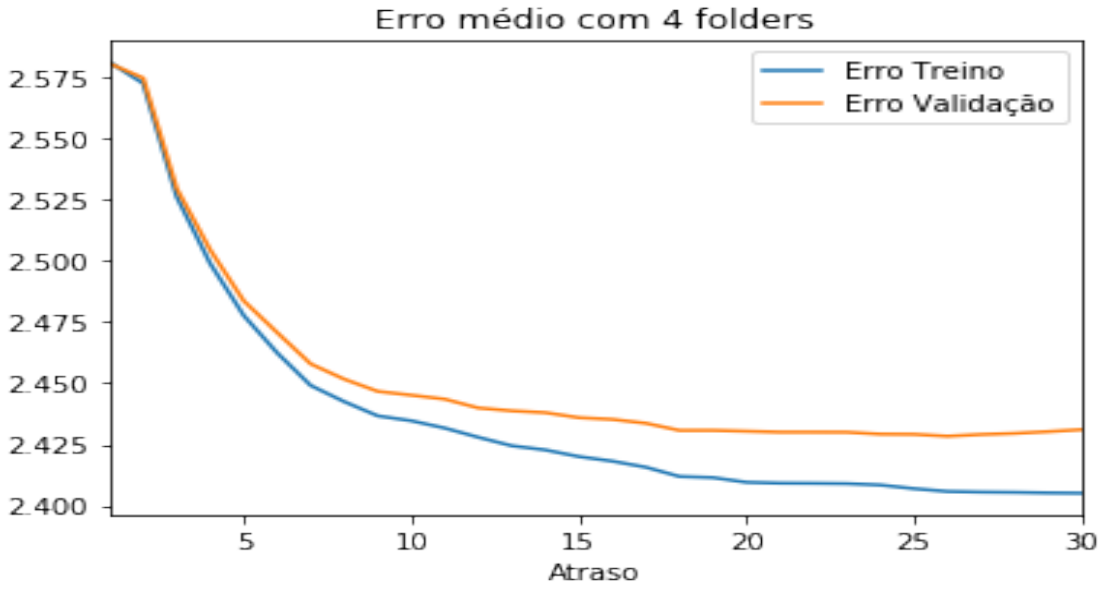
Em uma aplicação real não deveríamos utilizar os dados de teste para fazer escolhas relacionadas ao modelo. Entretanto neste exercício foi feita uma verificação de como dois métodos diferentes para estimar os dados de teste a partir de um treino com k-fold se comportavam.

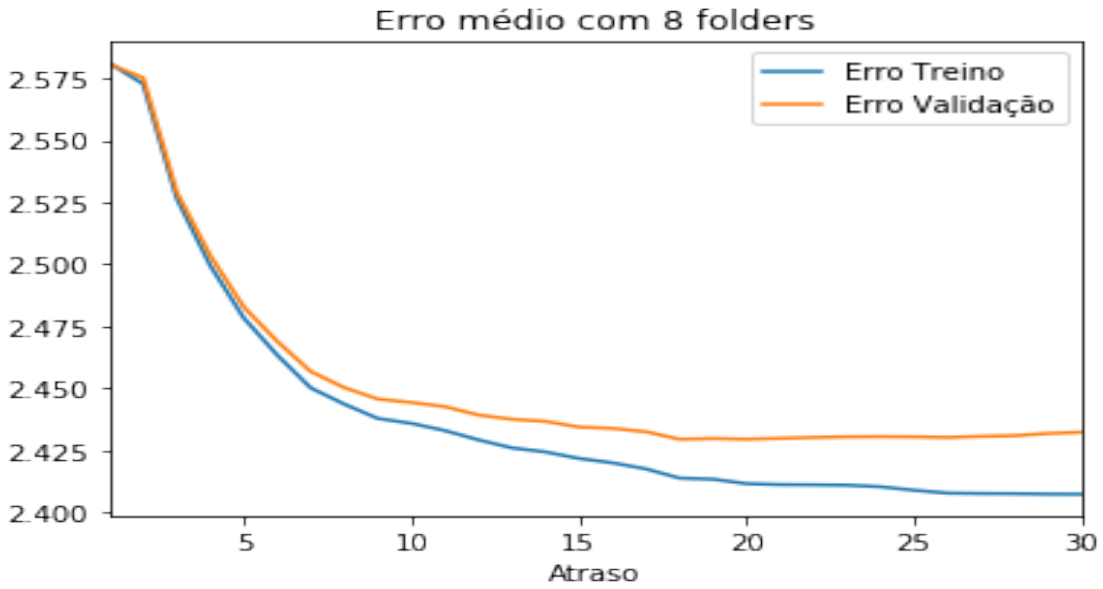
A partir do K ótimo para cada k, foram gerados dois modelos diferentes para estimar os dados de teste. O primeiro deles utiliza como saída a média dos k modelos e o o segundo deles cria um modelo linear novo utilizando como conjunto de treino os dados de todos os folders. Desta forma obtivemos os seguintes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


Erro médio com 2 folders
Erro Treino
Erro Validação
2.575
2.550
2.525
2.500
2.475
2.450
2.425
2.400
5
10
15
20
25
30
Atraso

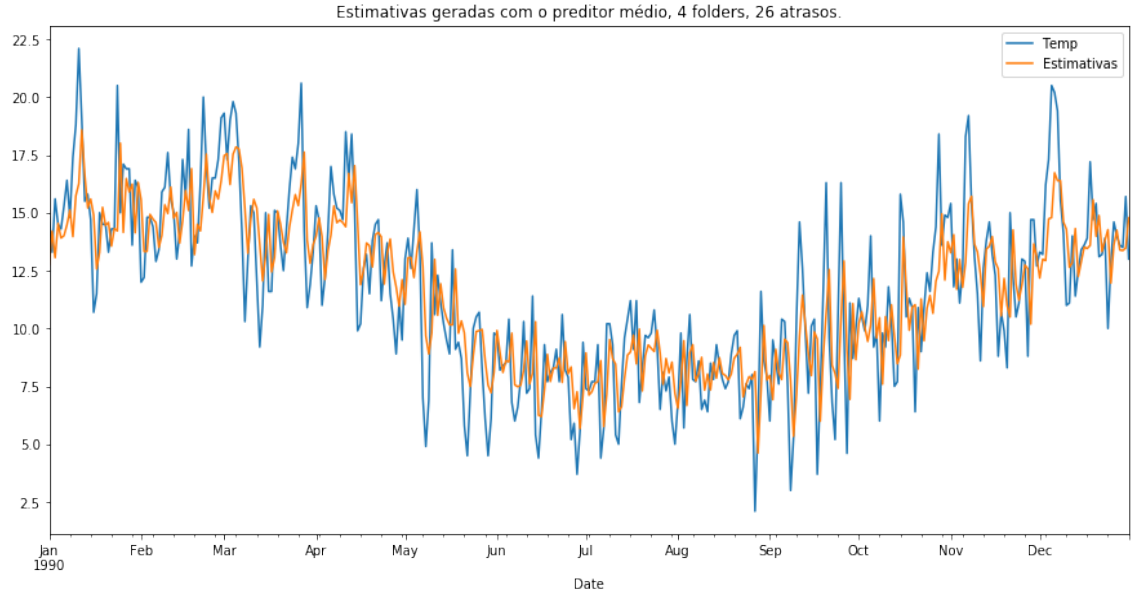

Erro médio com 4 folders
Erro Treino
Erro Validação
2.575
2.550
2.525
2.500
2.475
2.450
2.425
2.400
5
10
15
20
25
30
Atraso



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:

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

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

Em seguida gerou-se $T = 1, ..., T = 100$ 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:

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

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

$$\begin{bmatrix} z_1(1) & z_2(1) & \cdots & z_{100}(1) \\ z_1(2) & z_2(2) & \cdots & z_{100}(2) \\ \vdots & \vdots & \ddots & \vdots \\ z_1(n) & z_2(n) & \cdots & z_{100}(n) \end{bmatrix} = \tag{2.3}$$

$$= \begin{bmatrix} x_1(1) & x_2(1) & x_3(1) & x_4(1) & x_5(1) \\ x_1(2) & x_2(2) & x_3(2) & x_4(2) & x_5(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(n) & x_2(n) & x_3(n) & x_4(n) & x_5(n) \end{bmatrix} \begin{bmatrix} w_{1,1} & w_{2,1} & \cdots & w_{100,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{100,2} \\ w_{1,3} & w_{2,3} & \cdots & w_{100,3} \\ w_{1,4} & w_{2,4} & \cdots & w_{100,4} \\ w_{1,5} & w_{2,5} & \cdots & w_{100,5} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{5} x_i(1)w_{1,i} & \sum_{i=1}^{5} x_i(1)w_{1,2} & \cdots & \sum_{i=1}^{5} x_i(1)w_{100,2} \\ \sum_{i=1}^{5} x_i(2)w_{1,i} & \sum_{i=1}^{5} x_i(2)w_{1,2} & \cdots & \sum_{i=1}^{5} x_i(2)w_{100,2} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{5} x_i(n)w_{1,i} & \sum_{i=1}^{5} x_i(n)w_{1,2} & \cdots & \sum_{i=1}^{5} x_i(n)w_{100,2} \end{bmatrix}$$

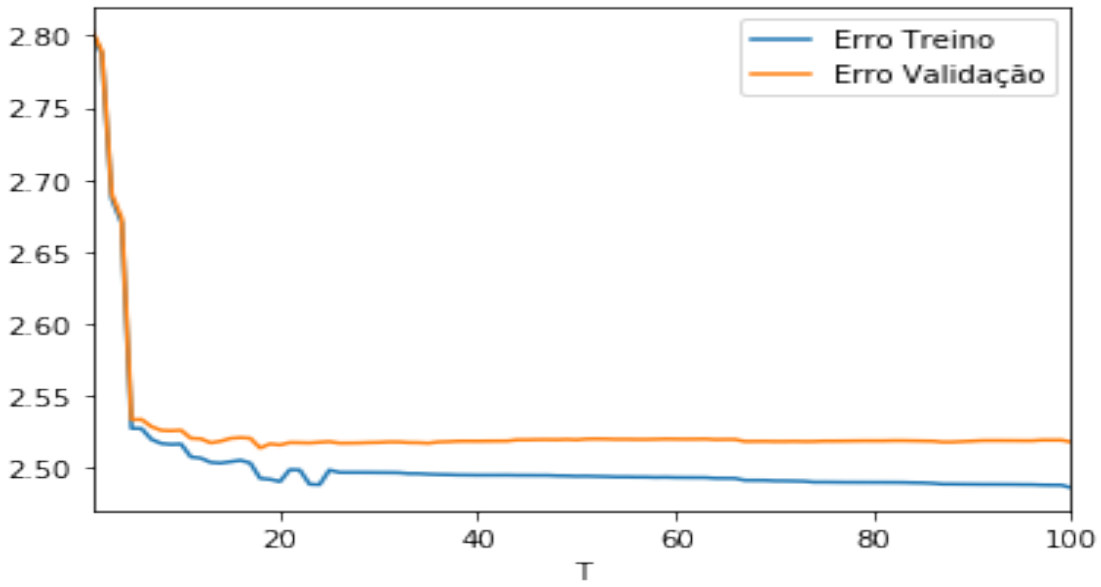
Através das normalizações 2.1 e 2.2 garante-se que nossos $z_i(n)$ pertencem ao intervalo $[-2, 2]$, por fim aplicou-se a tangente hiperbólica a estes valores:

$$z_i(n) \leftarrow tangh(z_i(n))$$

O modelo então foi treinado utilizando a técnica de Ridge Regression tendo como critério o erro quadrático médio e com $k = 4$ folds. Para $T = 1, ...100$ e $\lambda = 0, 2^{-10}, 2^{-9}, ..., 2^{10}$, 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.

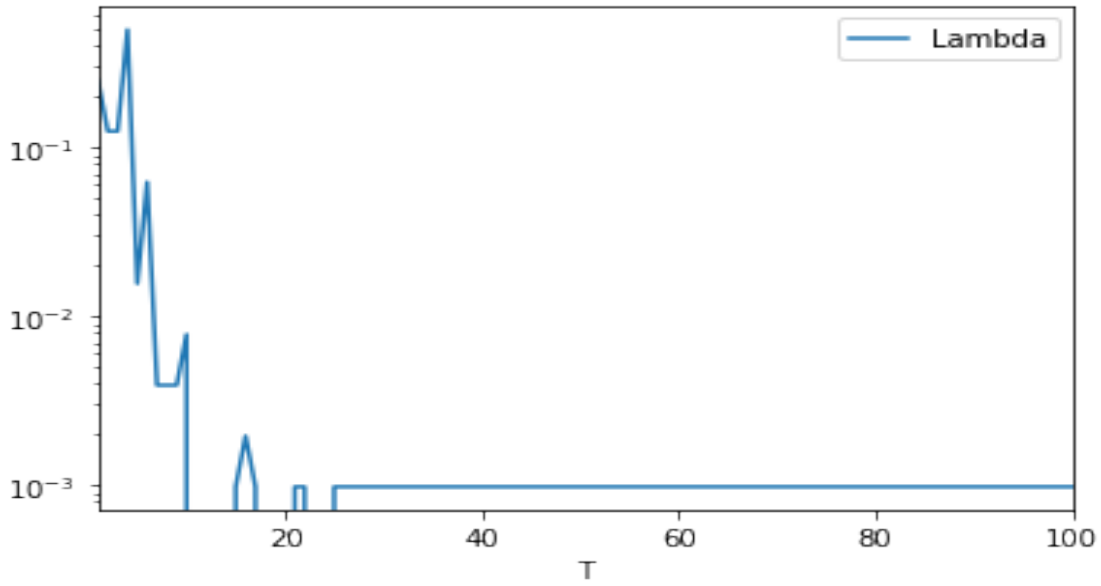
2.2.1 Letra a

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:



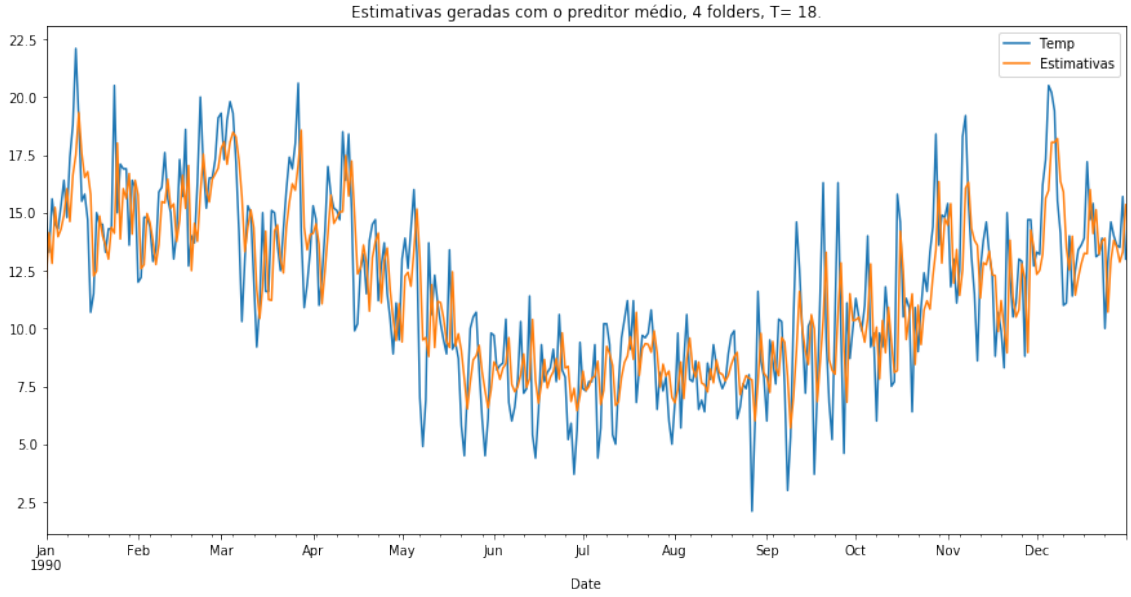
Neste gráfico vemos que os valores de λ que geraram o menor erro foram em sua maioria iguais as zeros.

2.2.3 Letra c

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

Erro Treino	Erro 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

```
# pacotes e configuracoes gerais
import pandas as pd
import matplotlib as ml
import numpy as np
import math as mt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
pd.set_option('display.max_rows',500)
pd.set_option('display.max_columns',20)

# Importacao dos dados e verificaÃ§Ã£o dos dados
data=pd.read_csv("C:/Users/Roger/OneDrive/ESPECIAL/IA006/
   python/daily-minimum-temperatures.csv")
data.dtypes
data=data.sort_values('Date')
data['Date'] = pd.to_datetime(data['Date'])
data['Year']=data['Date'].dt.year
data=data[['Year','Date','Temp']]

#Transforma os dados em um problema supervisionado
data_windowed=data.copy()
data_windowed['C']=1
for k in range(1,31) :
     colum_name='T-'+str(k)
     data_windowed[colum_name]=data['Temp'].shift(k)
#elimina os 30 primeiros dias q nao ha como prever
data_windowed = data_windowed.iloc[30:]
data_windowed.describe()
data_windowed.head()

#configuracoesos do mÃ©todo
ano_teste=1990
atrasos=30
lista_folders=[2]
#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             #calculo 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="NÃ£o"
73              resultados.loc[linha,'Matrix']=message
74              #cÃ¡lculo do vetor w para regressÃ£o
75              xtxinv=pd.DataFrame(np.linalg.pinv(xtx.values),
                   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              resultados.loc[linha,'Erro Treino']=error_train
81              #erro validaÃ§Ã£o
82              yh_valid=x_valid.dot(w)
83              error_valid=mt.sqrt(((yh_valid-y_valid)**2).mean
                   ())
84              resultados.loc[linha,'Erro ValidaÃ§Ã£o']=
                   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

```
import pandas as pd
import matplotlib as ml
import numpy as np
import math as mt
pd.set_option('display.max_rows',500)
pd.set_option('display.max_columns',50)

data=pd.read_csv("C:/Users/Roger/OneDrive/ESPECIAL/IA006/
   python/daily-minimum-temperatures.csv")
# data.head(20)
#data.tail(20)
# data.isna()
#data.dtypes
data=data.sort_values('Date')
data['Date'] = pd.to_datetime(data['Date'])
data['Year']=data['Date'].dt.year
data=data[['Year','Date','Temp']]

#normalizando a temperatura para ficar dentro do range [-2,2]
tmin=data.Temp.min()
tmax=data.Temp.max()
data_windowed=data.copy()
data_windowed['TempNorm']=(data['Temp']-tmin)/(tmax-tmin)*4-2

#gerando os 5 atrasos
for k in range(1,6) :
     colum_name='T-'+str(k)
     data_windowed[colum_name]=data_windowed['TempNorm'].
        shift(k)
data_windowed=data_windowed.iloc[5:].copy()
#criando os dados temporarios
x_temp=data_windowed.iloc[:,4:9]
y_temp=data_windowed.iloc[:,2:3]
date_temp=data_windowed.iloc[:,0:2]
#resetando o index para facilitar mais abaixo
```

```
x_temp.index = pd.RangeIndex(len(x_temp.index))
y_temp.index = pd.RangeIndex(len(y_temp.index))
date_temp.index = pd.RangeIndex(len(date_temp.index))
print(x_temp.describe())
print(y_temp.head())
print(date_temp.head())

#Criando o vetor w
w_temp=pd.DataFrame(columns=['w'])
w=pd.DataFrame(columns=['w1'])
variable_list=[]
np.random.seed(230482)
for t in range(0,100):
    #criando o vetor w
    for l in range(0,5):
        w_temp.loc[l,'w']=np.random.uniform(0, 1)
    #Normalizando o vetor w para que sua soma seja 1
    w_temp['w']=w_temp['w']/w_temp['w'].sum()
    colum_name='w'+str(t+1)
    variable_name='xl_'+str(t+1)
    variable_list.append(variable_name)
    w[colum_name]=w_temp['w']
# vetor normalizado entre -2 e 2
x_a=pd.DataFrame(np.float_(np.dot(x_temp,w)),columns=
   variable_list)
print(x_a.describe())
x=pd.DataFrame(np.tanh(np.float_(np.dot(x_temp,w))),columns=
   variable_list)

modified_data=date_temp.copy()
modified_data['Temp']=y_temp['Temp']
modified_data['C']=1
for h in list(x):
     modified_data[h]=x[h]

ano_teste=1990
atrasos=100
lista_folders=[4]
# lista_lambda=[0,]
```

```
71 lista_lambda
      =[0,0.000976563,0.001953125,0.00390625,0.0078125,0.015625,0.03125,0
72 #gera os dados de treino e teste
73 data_train=modified_data[modified_data['Year']!=ano_teste].
      copy()
74 # print(data_train.head())
75 data_test=modified_data[modified_data['Year']==ano_teste].
      copy()
76 #inicia os testes
77 resultados=pd.DataFrame(columns=['T','Lambda','Folder','Dado
      Treino','Dado ValidaÃ§Ã£o','Matrix','Erro Treino','Erro
      ValidaÃ§Ã£o','Erro Teste'])
78 linha=0
79 for folders in lista_folders:
80     #gerando uma coluna com o nÃºmero de folds a partir de
         uma distribuiÃ§Ã£o normal e inteiros
81     np.random.seed(9001)
82     data_train['fold'] = np.random.randint(1,folders+1,size=
         len(data_train))
83     for la in lista_lambda:
84         for f in range(1,folders+1):
85             lenght=len(data_train[data_train['fold']==f])
86         y_test=data_test.iloc[:,2:3].copy()
87         for k in range(1,atrasos+1):
88             print("RegressÃ£o:Lambda {} K {}".format(la,k))
89             #dados treino globais
90             x_test=data_test.iloc[:,3:k+4].copy()
91
92 #             print(x_test.head(2))
93             for f in range(1,folders+1):
94 #                 print(k)
95                 linha+=1
96                 resultados.loc[linha,'T']=k
97                 resultados.loc[linha,'Lambda']=la
98                 resultados.loc[linha,'Folder']=f
99                 #Dados treino
100                x_train=data_train[data_train['fold']!=f]
101                x_train=x_train.iloc[:,3:k+4].copy()
```

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

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