IA048 – Aprendizado de Máquina

Exercícios de Fixação de Conceitos (EFC) 1 – 2s2020

Aluno: Rodrigo Santos Gonçalves **RA:** 265306

Parte 1 – Atividades teóricas

Exercício 1 – Considere duas variáveis aleatórias binárias X e Y , com valores possíveis iguais a 0 e 1. A distribuição conjunta dessas variáveis, $P(X, Y)$, é apresentada na tabela a seguir.

	Y = 0	**Y = 1**
X = 0	**0,5**	**0,05**
X = 1	**0,3**	**0,15**

a) Obtenha $P(X)$ e $P(Y)$.

Solução:

Calculando a probabilidade marginal para cada valor da variável aleatória X, temos:

$$P(X = x_i) = \sum_{j=1}^{M} P(x_i, y_j) \tag{1}$$

Logo a partir de (1), calculando as probabilidades marginais $P(X = 0)$ e $P(X = 1)$, temos:

$$P(X = 0) = \sum_{j=0}^{1} P(0, y_j) = P(0,0) + P(0,1) = 0{,}5 + 0{,}05 = 0{,}55$$

$$\therefore P(X = 0) = 0{,}55$$

$$P(X = 1) = \sum_{j=0}^{1} P(1, y_j) = P(1,0) + P(1,1) = 0{,}3 + 0{,}15 = 0{,}45$$

$$\therefore P(X = 1) = 0{,}45$$

Para a variável aleatória Y, temos que:

$$P(Y = y_j) = \sum_{i=1}^{N} P(x_i, y_j) \tag{2}$$

$$P(Y = 0) = \sum_{i=0}^{1} P(x_i, 0) = P(0,0) + P(1,0) = 0{,}5 + 0{,}3 = 0{,}8$$

$$\therefore P(Y = 0) = 0{,}8$$

$$P(Y = 1) = \sum_{i=0}^{1} P(x_i, 0) = P(0,1) + P(1,1) = 0{,}05 + 0{,}15 = 0{,}2$$

$$\therefore P(Y = 1) = 0{,}2$$

Os resultados obtidos para as probabilidades marginais também pode ser representa através de tabela:

	Y = 0	Y = 1	P(x)
X = 0	0,5	0,05	0,55
X = 1	0,3	0,15	0,45
P(y)	0,8	0,2	1,0

b) Calcule $P(X = 1|Y = 1)$.

Solução:

Seja y_j um valor de Y, tal que $P(Y = y_j) = P(y_j) > 0$, então a probabilidade expressa por (3)

$$P(X = x_i|Y = y_j) = \frac{P(X = x_i, Y = y_j)}{P(Y = y_j)} \quad (3)$$

é a probabilidade condicional de $X = x_i$ dado que $Y = y_j$, logo calculando a probabilidade $P(X = 1|Y = 1)$, temos:

$$P(X = 1|\, Y = 1) = \frac{P(X = 1, Y = 1)}{P(Y = 1)} = \frac{0{,}15}{0{,}20} = 0{,}75$$

$\therefore P(X = 1|Y = 1) = 0{,}75$

c) As variáveis são descorrelacionadas? Por quê?

Solução:

Uma maneira de verificar se as variáveis são descorrelacionadas é calculando a covariância entre elas, se $cov(X, Y) = 0$ as variáveis são descorrelacionadas, portando calculando a covariância que é dada por:

$$cov(X, Y) = E\big[(X - E(X))(Y - E(Y))\big] = \sum_{i=1}^{N}\sum_{j=1}^{M} x_i y_j\, P(X = x_i, Y = y_j) - E(X)E(Y)$$

calculando $E(X)$ e $E(Y)$ temos que:

$$E(X) = \sum_{i=0}^{1} x_i P(X = x_i) = 0.P(X = 0) + 1.P(X = 1) = 0.0{,}55 + 1.0{,}45 = 0{,}45$$

$$E(Y) = \sum_{j=0}^{1} y_j P(Y = y_j) = 0.P(Y = 0) + 1.P(Y = 1) = 0.0{,}8 + 1.0{,}2 = 0{,}2$$

Determinando a covariância, temos:

$$\begin{aligned} cov(X, Y) &= \sum_{i=1}^{1}\sum_{j=1}^{1} x_i y_j\, P(X = x_i, Y = y_j) - E(X)E(Y) \\ &= 1.1.0{,}15 - 0{,}45.0{,}2 \\ &= 0{,}06 \end{aligned}$$

$\therefore cov(X, Y) = 0{,}06 > 0$

Portanto as variáveis não são descorrelacionadas, pois a covariância não é nula.

d) As variáveis são estatisticamente independentes? Por quê?

Solução:

Por definição para as variáveis aleatórias discretas bidimensionais (X, Y), dizemos que X e Y são independentes se, e somente se, para todo par de valores (x_i, y_j) de X e Y, temos:

$$P\big(X = x_i, Y = y_j\big) = P(X = x_i).P\big(Y = y_j\big) \tag{4}$$

logo se a condição não existir para apenas um par (x_i, y_j), as variáveis aleatórias X e Y não são independentes. Logo apenas um contra exemplo não satisfazendo (4) é o suficinte para provar que X e Y não são independentes, assim:

$$P(X = 1, Y = 1) = 0{,}15 \neq P(X = 1).P(Y = 1) = 0{,}09$$

portanto como $P(X = 1, Y = 1) \neq P(X = 1).P(Y = 1)$, temos que X e Y não são estatisticamente independentes.

e) Calcule $H(X, Y)$, $H(X)$, $H(Y)$, $H(X|Y)$ e $H(Y|X)$.

Solução:

A entropia conjunta de duas variáveis aleatórias X e Y é dada por:

$$H(X, Y) = -\sum_{i=0}^{N}\sum_{j=0}^{M} P\big(X = x_i, Y = y_j\big)\log_2\big[P(X = x_i, Y = y_j)\big] \tag{5}$$

calculando a entropia conjunta:

$$H(X, Y) = -\sum_{i=0}^{1}[P(x_i, y_0)\log_2 P(x_i, y_0) + P(x_i, y_1)\log_2 P(x_i, y_1)]$$
$$= -[P(x_0, y_0)\log_2 P(x_0, y_0) + P(x_1, y_0)\log_2 P(x_1, y_0) + P(x_0, y_1)\log_2 P(x_0, y_1) + P(x_1, y_1)\log_2 P(x_1, y_1)]$$
$$= -[0{,}5\log_2 0{,}5 + 0{,}3\log_2 0{,}3 + 0{,}05\log_2 0{,}05 + 0{,}15\log_2 0{,}15]$$
$$\simeq 1{,}64773\ bits$$
$$\therefore H(X, Y) \simeq 1{,}64773\ bits$$

Calculando $H(X)$ e $H(Y)$, obtemos

$$H(X) = -\sum_{i=0}^{1} P(X = x_i)\log_2[P(X = x_i)]$$

$$= -[P(X = x_0)\log_2 P(X = x_0) + P(X = x_1)\log_2 P(X = x_1)]$$
$$= -[0{,}55\log_2 0{,}55 + 0{,}45\log_2 0{,}45]$$
$$\simeq 0{,}992774\ bits$$

$$\therefore H(X) \simeq 0{,}992774\ bits$$

$$H(Y) = -\sum_{j=0}^{1} P(Y = y_j)\log_2[P(Y = y_j)]$$
$$= -[P(Y = y_0)\log_2 P(Y = y_0) + P(Y = y_1)\log_2 P(Y = y_1)]$$
$$= -[0{,}8\log_2 0{,}8 + 0{,}2\log_2 0{,}2]$$
$$\simeq 0{,}721928\ bits$$

$\therefore H(Y) \simeq 0{,}721928\ bits$

Calculando a entropia condicional de $H(X|Y)$ e $H(Y|X)$, temos:

$$H(X|Y) = -\sum_{i=0}^{N}\sum_{j=0}^{M} P(X = x_i, Y = y_j)\log_2[P(X = x_i|Y = y_j)]$$
$$= -[P(x_0, y_0)\log_2 P(x_0|y_0) + P(x_1, y_0)\log_2 P(x_1|y_0) + P(x_0, y_1)\log_2 P(x_0|y_1) + P(x_1, y_1)\log_2 P(x_1|y_1)]$$
$$= -[0{,}5\log_2 0{,}625 + 0{,}3\log_2 0{,}375 + 0{,}05\log_2 0{,}25 + 0{,}15\log_2 0{,}75]$$
$$\simeq 0{,}925803\ bits$$

$\therefore H(X|Y) \simeq 0{,}925803\ bits$

Como temos que a entropia conjunta pode ser escrita em função da entropia condicional:

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

logo, se tem que $H(Y|X) = H(X, Y) - H(X) = H(Y) - H(X) + H(X|Y)$, portanto

$$H(Y|X) = H(X, Y) - H(X)$$
$$= 1{,}64773\ bits - 0{,}992774\ bits$$
$$= 1{,}64773\ bits - 0{,}992774\ bits$$
$$= 0{,}654956\ bits$$

$\therefore H(Y|X) \simeq 0{,}654956\ bits$

f) Calcule $I(X, Y)$.

Calculando a informação mútua $I(X, Y)$, dada por:

$$I(X, Y) = H(X) - H(X|Y)$$
$$= 0{,}992774\ bits - 0{,}925803\ bits$$
$$= 0{,}992774\ bits - 0{,}925803\ bits$$
$$= 0{,}066971\ bits$$

$\therefore I(X, Y) = 0{,}066971\ bits$

Parte 2 – Atividade computacional

Nesta atividade, vamos abordar uma instância do problema de regressão de grande interesse prático e com uma extensa literatura: a **predição de séries temporais**. A fim de se prever o valor futuro de uma série de medidas de uma determinada grandeza, um procedimento típico consiste em construir um modelo matemático de estimação baseado na hipótese de que os valores passados da própria série podem explicar o seu comportamento futuro.

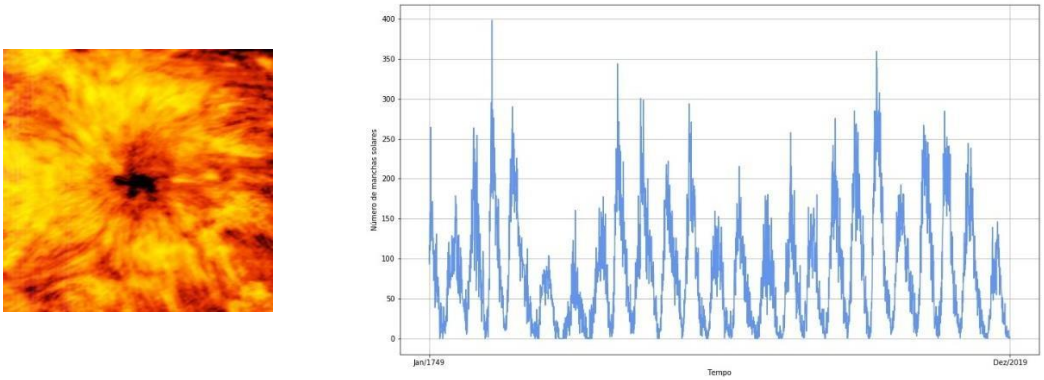
Seja $x(n)$ o valor da série temporal no instante (discreto) n. Então, o modelo construído deve realizar um mapeamento do vetor de entradas $\mathbf{x}(n) \in \mathbb{R}^{K\times 1}$, o qual é formado por um subconjunto de K amostras passadas, *i.e.*,

$$\mathbf{x}(n) = [x(n-1) \ldots x(n-K)]^{\mathrm{T}},$$

para uma saída $\hat{x}(n)$, que representa uma estimativa do valor futuro da série $x(n)$[*].

Neste exercício, vamos trabalhar com a famosa série histórica de medidas do número de manchas solares (*sunspots*). No caso, dispomos das leituras mensais desde 1749 a 2019, totalizando 3252 amostras. A Figura 1 exibe um registro de mancha solar juntamente com o gráfico da série completa.

[*] Esta modelagem está pressupondo o caso em que desejamos prever o valor da série um passo à frente.



(a) (b)

Figura 1. Em (a), temos um exemplo de uma mancha solar observada pelo Atacama Large Millimeter / Submillimeter Array (ALMA). Em (b), os valores mensais do número de manchas solares desde 1749 a 2019.

Exercício 1

Inicialmente, vamos explorar um modelo linear para a previsão, tal que:

$$\hat{y}(n) = \mathbf{w}^T\mathbf{x}(n) + w_0$$

Para o projeto do preditor linear, separe os dados disponíveis em dois conjuntos, um para treinamento e outro para teste. No caso, reserve as amostras referentes aos últimos dez anos (2010-2019) em seu conjunto de teste. Além disso, utilize um esquema de validação cruzada do tipo *k-fold* para selecionar o melhor valor do hiperparâmetro K.

Faça a análise de desempenho do preditor linear ótimo, no sentido de quadrados mínimos irrestrito, considerando:

1. A progressão do valor médio da raiz quadrada do erro quadrático médio (RMSE, do inglês *root mean squared error*), junto aos dados de validação, em função do número de entradas (K) do preditor (desde $K = 1$ a $K = 24$).

Resultados e Discussões - Exercício 1.1

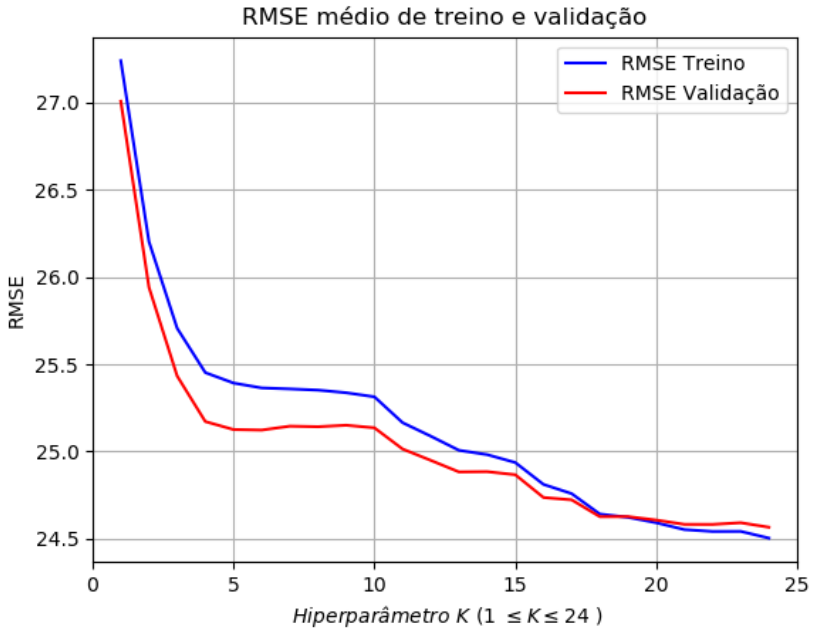


Figura 1.1 – Curva dos erros RMSE médios de treino e validação, a regressão linear obtém seu RMSE mínimo em $K = 24$.

Na figura 1.1 temos as curvas de erros RMSE médios obtidos em uma validação cruzada k-fold para cada novo modelo obtido com seu hiperparâmetro K modificado e considerando a utilização de 10 pastas no k-fold, observa-se que não se pode estabelecer um paralelo com as curvas de erros estudas pois em geral se tratam de curvas de erro em função do número de iterações, aqui no caso temos uma curva de erro de uma regressão linear múltipla de uma série temporal em função do número de entradas, que são as k-ésimas amostras passadas da série, utilizadas para estimar a série em um determinado instante n. O algoritmo desenvolvido para implementar esses resultados, basicamente

possui um primeiro loop mais externo onde é feita a variação do número de parâmetros de entrada utilizado na regressão linear múltipla, e um loop interno onde é feita a validação cruzada para cada modelo de regressão linear múltipla criado com uma determinada quantidade de variáveis de entrada. Analisando a curva RMSE da figura 1.1, observa-se uma evidente tendência de que a medida que se aumenta o número de entradas da regressão ou seja o número de amostras anteriores utilizadas da série, menor é o erro do preditor obtido, e também ao longo das iterações do número de hiperpâmetros o RMSE de validação se manteve abaixo do valor de treino até $K < 18$, os RMSE mínimos de validação e treino aproximados são respectivamente $24{,}566$ para validação e $24{,}504$ em treino, em ambos o valor mínimo ocorre para $K = 24$.

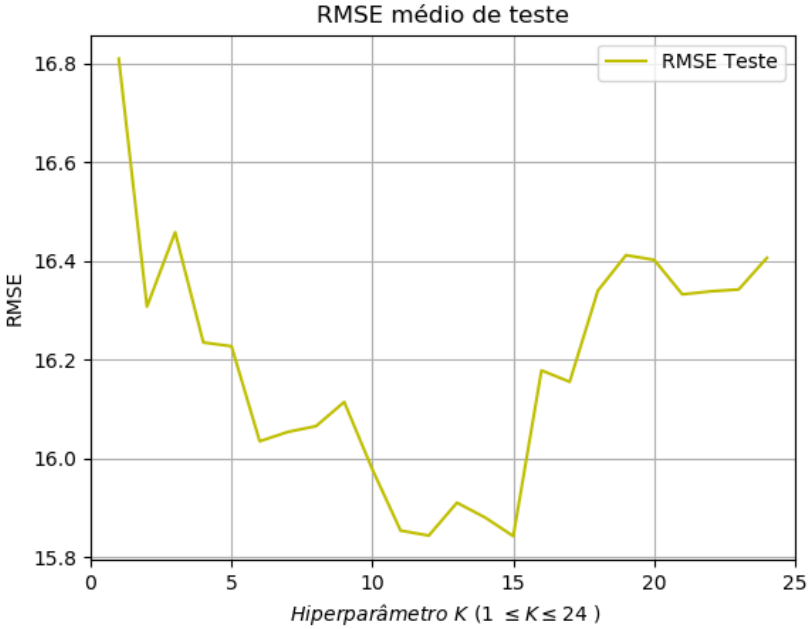


Figura 1.2 – Curva dos erros RMSE médios com o conjunto de dados de teste, a regressão linear obtém seu RMSE mínimo em $K = 15$.

Como curiosidade se analisou também o desempenho de cada modelo anteriormente criado com o conjunto de amostras da série deixados para teste, na figura 1.2 temos o RMSE médios obtidos com os conjuntos de teste, pode se observar que o RMSE mínimo da curva ocorreu para o modelo com hiperparâmetro $K = 15$ e seu RMSE mínimo foi de $15{,}843$. Além disso, outro resultado interessante obtido foi avaliando o que aconteceria se fosse aumentado o número de pastas da validação cruzada, pode ser visto na figura 1.3, que aumento do número de pastas acabou reduzindo consideravelmente o erro de validação, outro fato é que a diferenças entre os erros RMS validação e treino se tornaram mais acentuados, entretanto agora a validação permanece abaixo de treino para qualquer valor de K.

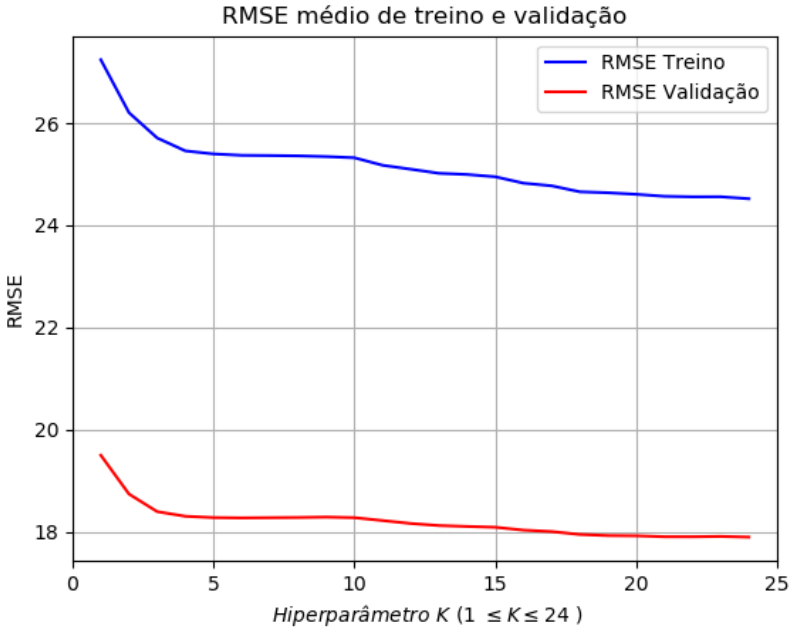


Figura 1.3 – Curva dos erros RMSE médios de treino e validação, feita considerando validação cruzada *leave one out validation*, a curva de validação obtém seu RMSE mínimo em $K = 24$ com valor RMSE igual a 17,892, entretanto.

2. O gráfico com as amostras de teste da série temporal e com as respectivas estimativas geradas pela melhor versão do preditor (*i.e.*, usando o valor de K que levou ao mínimo erro de validação).
   Observação: Neste exercício, não é necessário utilizar regularização, nem efetuar normalizações nos dados.

Resultados e Discussões - Exercício 1.2

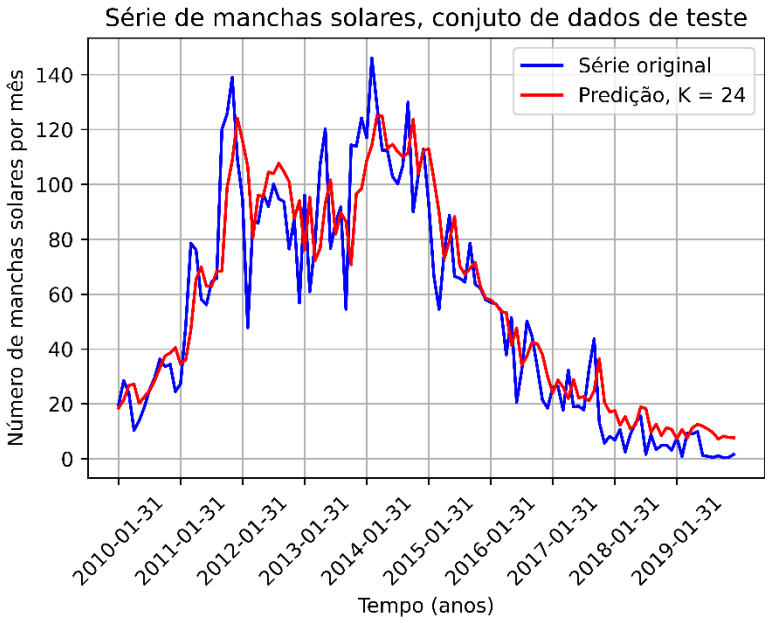


Figura 1.4 – Série de manchas solares, no gráfico temos a série original a partir do ano de 2010, e a série obtida pelo preditor com menor erro na validação.

Agora nas figuras 1.4 e 1.5 temos os resultados do preditores obtidos, na figura 1.4 temos exposto o desempenho do preditor com o menor erro do conjunto de validação que utiliza 24 parâmetros de entrada e foi obtido na validação cruzada com 10 pastas, e na figura 1.5 é representando o preditor com o menor RMSE no conjunto de teste, os resultados dos preditores obtidos em validação cruzada do tipo *leave one out validation*, não são apresentados graficamente devido a pequena diferença percentual do RMSE encontrado no desempenho com relação as instâncias de teste, as diferenças percentuais são de 0,32% no preditor de validação e de 0,25% no de teste. Além disso outro aspecto relevante é a diferença percentual entre os desempenhos dos dois melhores preditores, enquanto o preditor com melhor resultado com dados de teste utiliza 15 hiperparâmetros e tem um RMSE de 15,758, o preditor com melhor resultado na validação tem 24 hiperparâmetros e seu RMSE é 16,447 o que implica em uma diferença percentual de 4,19%, olhando pela perspectiva da diferença no número de parâmetros entre um preditor e outro a diferença de desempenhos é razoavelmente baixa.

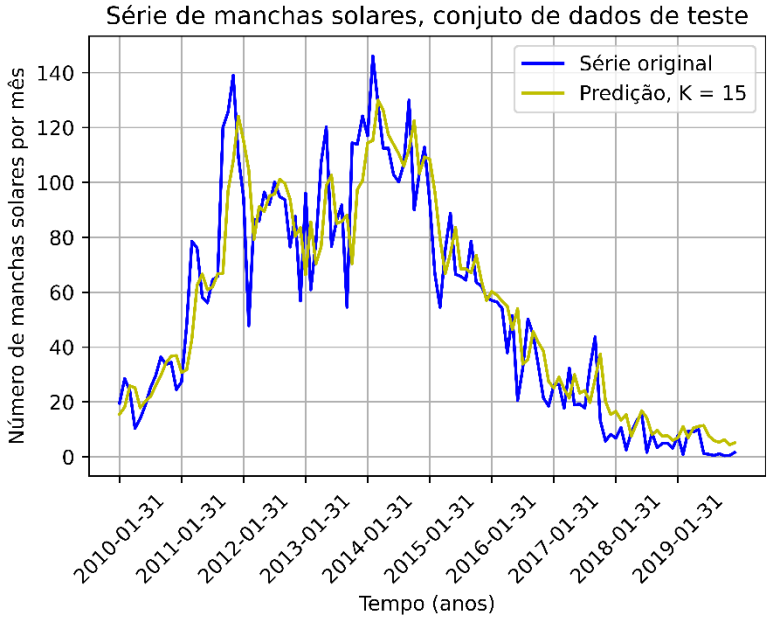


Figura 1.5 – Série de manchas solares, no gráfico temos a série original a partir do ano de 2010, e a série obtida pelo preditor com menor erro com relação ao conjunto de testes.

Exercício 2

Agora, vamos explorar um modelo de predição linear que utiliza como entrada valores obtidos a partir de transformações não-lineares do vetor $\mathbf{x}(n)$. Em outras palavras, os atributos que efetivamente são combinados linearmente na predição resultam de mapeamentos não-lineares dos atrasos da série presentes no vetor original $\mathbf{x}(n)$. No caso, vamos gerar T atributos transformados da seguinte forma:

$$x'_k(n) = \tanh(\mathbf{w}_k{}^T \mathbf{x}(n)),$$

para $k = 1, \ldots, T$, $n = 1, \ldots, N$. Os vetores $\mathbf{w}_k$ tem seus elementos gerados aleatoriamente de acordo com uma distribuição uniforme.

Curiosidade: a estrutura explorada neste exercício corresponde, na realidade, a uma rede neural conhecida como *extreme learning machine* (ELM).

- Huang, G.-B., Zhu, Q.-Y., e Siew, C.-K. (2006). *Extreme learning machine: theory and applications*. Neurocomputing, 70, 489–501.

Utilizando um esquema de validação cruzada do tipo *k-fold*, juntamente com a técnica *ridge regression* para a regularização do modelo:

a) Apresente o gráfico com a média dos valores de RMSE do preditor em função do número de atributos (T) utilizados, desde $T = 1$ a $T = 100$. Neste caso, considere $K = 8$ (número de atrasos presentes no vetor $\mathbf{x}(n)$).

Resultados e Discussões - Exercício 2.a)

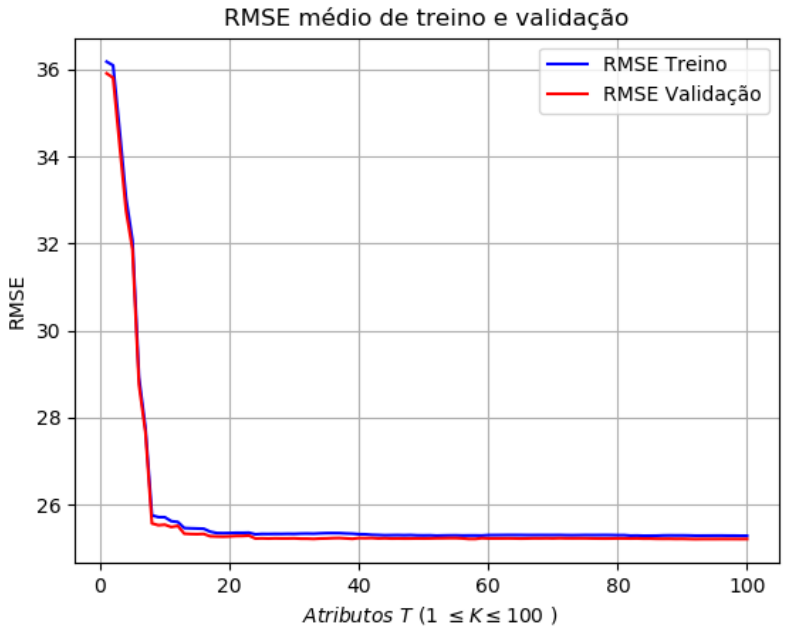


Figura 2.1 – Curva dos erros RMSE médios de treino e validação, a regressão linear de Ridge obtém seu RMSE mínimo em $T = 92$.

Na figura 2.1 temos as curvas de erros RMSE médios de treino e validação, obtidos em uma validação cruzada k-fold, para cada novo modelo obtido com sua nova quantidade de atributos T, é evidente que as curvas tanto de treino como validação se

sobrepõem e valores RMSE são muito próximos. Na implementação deste treino o código é praticamente igual sendo apenas diferente nas manipulações necessárias para a transformação não linear do domínio das entradas, para evitar a saturação da tangente hiperbólica os elementos da matriz resultante foram divididos pelo elemento de maior valor absoluto da matriz. O melhor resultado obtido na validação foi para um modelo com a quantidade de atributos $T = 92$ e um RMSE de validação igual a $25{,}205$, que é aproximadamente 2,54% maior do que o resultado do exercício 1, na figura 2.2 temos a curva do RMSE médio para o conjunto de teste dos modelos à medida que variado o T e obtemos RMSE mínimo de $16{,}049$, que é 1,28% maior do que o resultado no gráfico 1.2 da questão 1. Outro aspecto relevante é que matriz aleatória $\boldsymbol{W}$ introduz uma aleatoriedade elevada nos resultados entre uma simulação e outra, ficando evidente uma forte dependência dos resultados em função da matriz $\boldsymbol{W}$ gerada, logo é ideal um valor fixo de semente antes de gerar a matriz aleatória $\boldsymbol{W}$ para evitar resultados distintos entre simulações, aqui nesta simulação foi adotado como semente o valor 3 para a função seed() receber.

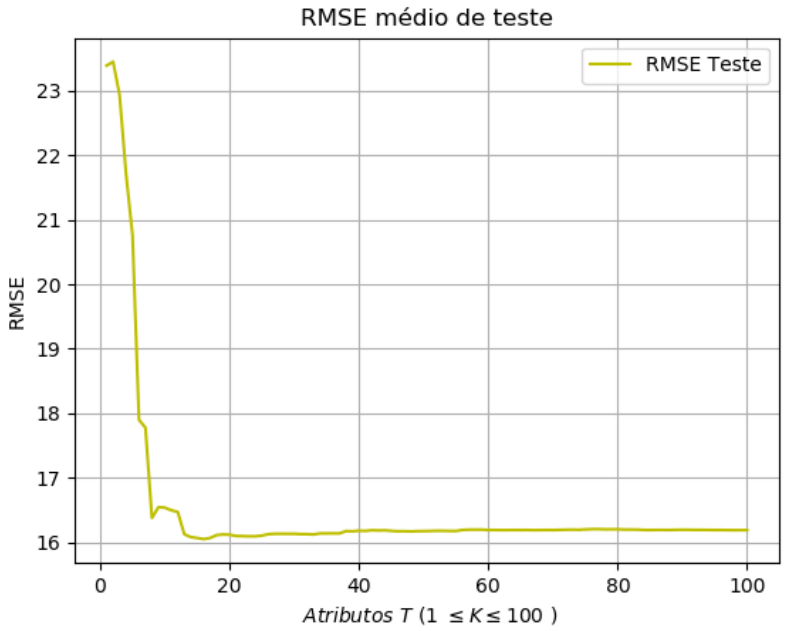


Figura 2.2 – Curva dos erros RMSE médios com o conjunto de dados de teste, a regressão linear de Ridge obtém seu RMSE mínimo em $T = 16$.

b) Apresente o melhor valor do parâmetro de regularização obtido para cada valor de T.

Resultados e Discussões - Exercício 2.b)

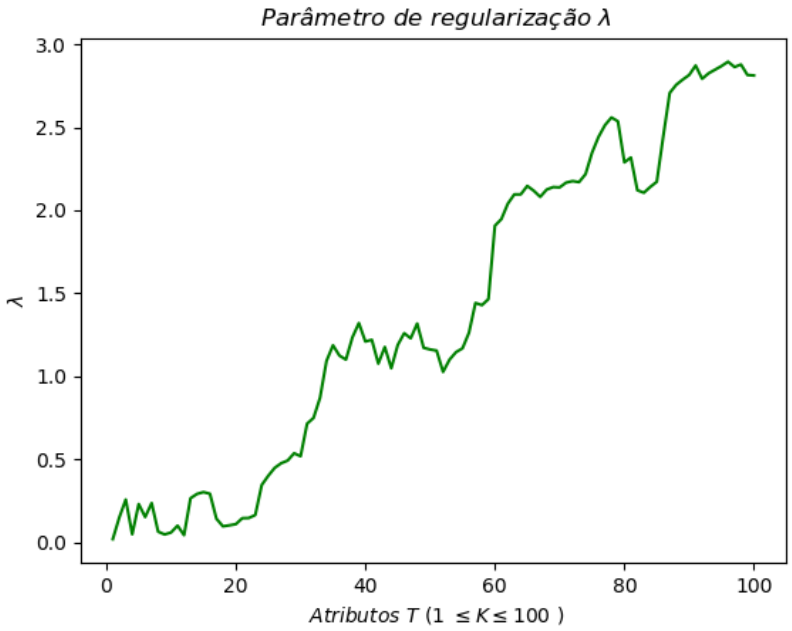


Figura 2.3 – Curva com os melhores valores de λ obtidos para cada valor do atributo T.

Na figura 2.3 temos os melhores valores médios do parâmetro de regularização obtidos para cada quantidade de atributo T utilizada pelo modelo de regressão, para o melhor modelo utilizado na validação com $T = 92$, temos $\lambda = 2{,}792$, enquanto que para o preditor com melhor resultado no conjunto de testes e com $T = 16$, o parâmetro de regularização é $\lambda = 0{,}293$. Observe que o parâmetro de regularização cresce à medida que o número de atributos do modelo de regressão cresce, ficando evidente uma relação diretamente proporcional entre as grandezas.

c) Por fim, aplique o modelo com os melhores valores de λ (regularização) e de T aos dados de teste. Meça o desempenho em termos de RMSE e mostre o gráfico com as amostras de teste da série temporal e as respectivas estimativas geradas pela melhor versão do preditor.

Observação: neste exercício, é preciso levar em consideração a escala dos valores da série ao se pensar no intervalo admissível para os coeficientes aleatórios das projeções. Também é possível tratar esta questão através de normalizações. Contudo, os valores de RMSE e a exibição da série de teste estimada devem ser referentes ao domínio original do problema.

Considerações gerais:

- Sejam criteriosos na escolha de todos os parâmetros e justifiquem todas as opções relevantes feitas. Além disso, analisem e comentem todos os resultados obtidos.

Resultados e Discussões - Exercício 2.c)

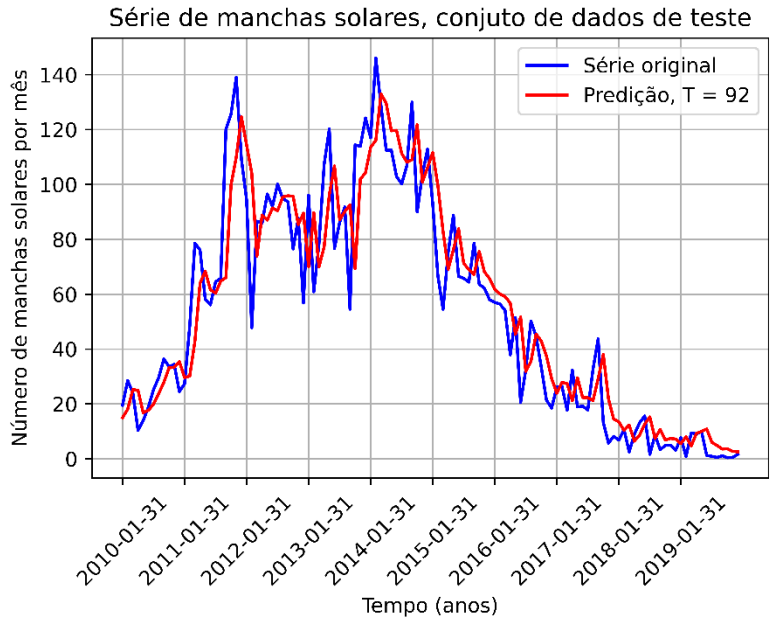


Figura 2.4 – Série de manchas solares, no gráfico temos a série original a partir do ano de 2010, e a série obtida pelo preditor com menor erro na validação, utilizando 92 atributos, possui uma diferença percentual de 0,02% com relação ao modelo da figura 1.4.

Nas figura 2.4 e 2.5 temos os preditores com melhores resultados obtidos no conjunto de validação e no conjunto de teste. O preditor com melhor desempenho na validação que está representado no gráfico da figura 2.4 obteve um RMSE no conjunto de teste de 16,151, e o preditor com melhor desempenho no conjunto de teste, tem RMSE de 15,973, implicando em uma diferença percentual de 1,1% entre os dois modelos, novamente assim como no exercício 1 temos uma grande variação no número de parâmetros de entrada utilizados em ambos os modelos, mas uma diferença percentual insignificante nos desempenhos.

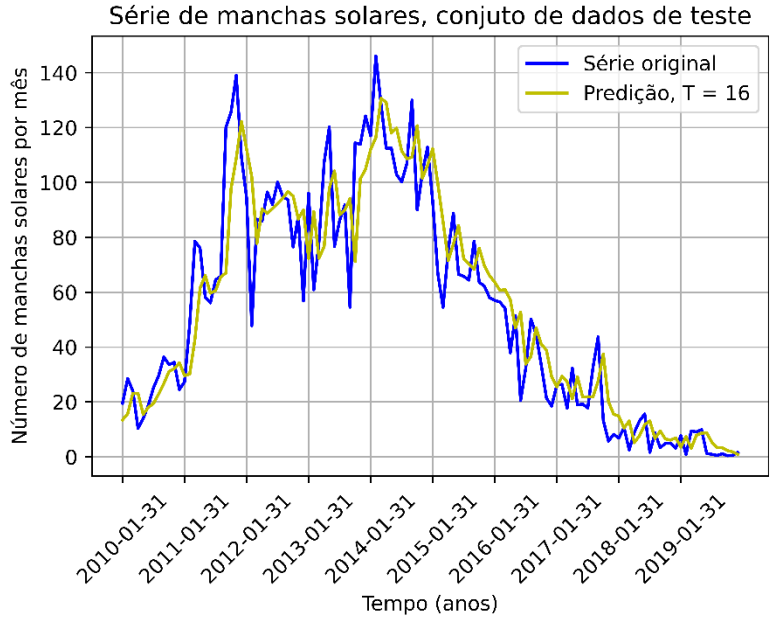


Figura 2.5 – Série de manchas solares, no gráfico temos a série original a partir do ano de 2010, e a série

obtida pelo preditor com menor erro no conjunto de teste, utilizando 16 atributos, possui uma diferença percentual de 1,35% com relação ao modelo da figura 1.5.

Referências:

[1] Attux, Romis. Boccato, Levy. Notas de aula: Fundamentos de Probabilidade. Unicamp – FEEC – DCA, 2020.

[2] Attux, Romis. Boccato, Levy. Notas de aula: Fundamentos de Teoria da Informação. Unicamp – FEEC – DCA, 2020.

[3] Attux, Romis. Boccato, Levy. Notas de aula: Regressão Linear. Unicamp – FEEC – DCA, 2020.

Anexos:

Link para emular o script na plataforma da Google Colab Research:

Exercício 1

https://colab.research.google.com/drive/1CL67jZRw9Lvq2dQFtvh3J2mS-lDsQmkx?authuser=1#scrollTo=DQkf_2PphgMD

Exercício 2

https://colab.research.google.com/drive/1gvR16997bCNaKxxZk5309bP3mYV6PIB-?usp=sharing

Anexos:

Exercício 1

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold


manchas_solares = pd.read_csv('/content/drive/My Drive/Colab Noteboo
ks/monthly-sunspots.csv')

for k in np.arange(1, 25):
  manchas_solares['x[n-
'+str(k)+']'] = manchas_solares['Monthly Mean Total Sunspot Number']
  manchas_solares['x[n-'+str(k)+']'] = manchas_solares['x[n-
'+str(k)+']'].shift(k)

rmse_validation_medium_array = []
rmse_train_medium_array = []
rmse_test_medium_array = []
linear_regression_array = []
rmse_test_minimum = 1000000
rmse_validation_minimum = 1000000
datas = np.array(manchas_solares.iloc[3132:, 1:2]).reshape(1, -1)
datas = datas[0]
print(datas)
for K in np.arange(1, 25):
  linha_maxima_kfold = 3132
  hiperparametro_K = K

  x = manchas_solares.iloc[hiperparametro_K:linha_maxima_kfold, 3:hi
perparametro_K+3]
  y = manchas_solares.iloc[hiperparametro_K:linha_maxima_kfold, 2:3]
  x_test = manchas_solares.iloc[linha_maxima_kfold:, 3:hiperparametr
o_K+3]
  y_test = manchas_solares.iloc[linha_maxima_kfold:, 2:3]
  x = np.asarray(x)
  y = np.asarray(y)
  x_test = np.asarray(x_test)
  y_test = np.asarray(y_test)

  kf = KFold(n_splits = len(x))
  rmse_validation_array = []
  rmse_train_array = []
  rmse_test_array = []
```

```
  for train_index, validation_index in kf.split(x):
    x_train, x_validation = x[train_index], x[validation_index]
    y_train, y_validation = y[train_index], y[validation_index]

    linear_regression = LinearRegression()
    linear_regression.fit(x_train, y_train)

    prediction_validation = linear_regression.predict(x_validation)
    predicition_train = linear_regression.predict(x_train)
    prediction_test = linear_regression.predict(x_test)

    rmse_validation = np.sqrt(mean_squared_error(y_validation, predi
ction_validation))
    rmse_train = np.sqrt(mean_squared_error(y_train, predicition_tra
in))
    rmse_test = np.sqrt(mean_squared_error(y_test, prediction_test))

    rmse_validation_array.append(rmse_validation)
    rmse_train_array.append(rmse_train)
    rmse_test_array.append(rmse_test)

    if K == 24:
      if (rmse_validation_minimum > rmse_validation):
        rmse_validation_minimum = rmse_validation
        prediction_validation_best = linear_regression.predict(x_tes
t)
        Melhor_K_validacao = K

    if (rmse_test_minimum > rmse_test):
      rmse_test_minimum = rmse_test
      prediction_test_best = prediction_test
      Melhor_K_teste = K

    print("Hiperparâmetro K = " + str(K))
    print("Erro RMS de treino da interação = ", rmse_train)
    print("Erro RMS de validação da interação = ", rmse_validation)
    print("Erro RMS de teste da interação = ", rmse_test)

  rmse_validation_medium = np.mean(rmse_validation_array)
  rmse_train_medium = np.mean(rmse_train_array)
  rmse_test_medium = np.mean(rmse_test_array)

  rmse_validation_medium_array.append(rmse_validation_medium)
  rmse_train_medium_array.append(rmse_train_medium)
  rmse_test_medium_array.append(rmse_test_medium)

print('valor minimo rmse treino:',min(rmse_train_medium_array))
print('valor minimo rmse validação:',min(rmse_validation_medium_arra
y))
print('valor minimo rmse teste:',min(rmse_test_medium_array))
```

```
print('Melhor hiperparâmetro K no treino:',rmse_train_medium_array.i
ndex(min(rmse_train_medium_array))+1)
print('Melhor hiperparâmetro K na validação:',rmse_validation_medium
_array.index(min(rmse_validation_medium_array))+1)
print('Melhor hiperparâmetro K no teste:',rmse_test_medium_array.ind
ex(min(rmse_test_medium_array))+1)

print(rmse_train_medium_array)
print(rmse_validation_medium_array)
print(rmse_test_medium_array)

hiperparametros = np.arange(1, 25)

plt.title('RMSE médio de treino e validação')
plt.ylabel('RMSE')
plt.xlabel(r'$\ Hiperparâmetro \ K  \ (1\  \leq K \leq 24 \ ) $')
plt.grid(True)
plt.plot(hiperparametros, rmse_train_medium_array, color = 'b', labe
l='RMSE Treino')
plt.plot(hiperparametros, rmse_validation_medium_array, color = 'r',
 label='RMSE Validação')
plt.legend()
plt.show()

plt.title('RMSE médio de teste')
plt.ylabel('RMSE')
plt.xlabel(r'$\ Hiperparâmetro \ K  \ (1\  \leq K \leq 24 \ ) $')
plt.grid(True)
plt.plot(hiperparametros, rmse_test_medium_array, color = 'y', label
='RMSE Teste')
plt.legend()
plt.show()

print('RMSE de validação mínimo : ',np.sqrt(mean_squared_error(y_tes
t, prediction_validation_best)))
plt.title('Série de manchas solares, conjuto de dados de teste')
plt.ylabel('Número de manchas solares por mês')
plt.xlabel('Tempo (anos)')
plt.grid(True)
plt.plot(datas,y_test, color = 'b', label='Série original')
plt.plot(datas,prediction_validation_best, color = 'r', label='Predi
ção, K = 24')
plt.xticks(rotation=45)
plt.xticks(datas[np.arange(0,len(datas),12)])
plt.legend(loc='best')
plt.savefig('figura_manchassolares1.png', dpi=1080, facecolor='w', e
dgecolor='w', orientation='landscape', papertype=None, format=None,t
ransparent=False, bboxinches=None, padinches=0.1,frameon=None, bbox_
inches='tight')
plt.show()

print('RMSE de teste mínimo : ',rmse_test_minimum)
plt.title('Série de manchas solares, conjuto de dados de teste')
```

```
plt.ylabel('Número de manchas solares por mês')
plt.xlabel('Tempo (anos)')
plt.grid(True)
plt.plot(datas,y_test, color = 'b', label='Série original')
plt.plot(datas,prediction_test_best, color = 'y', label='Predição, K
 = 15')
plt.xticks(rotation=45)
plt.xticks(datas[np.arange(0,len(datas),12)])
plt.legend(loc='best')
plt.savefig('figura_manchassolares2.png', dpi=1080, facecolor='w', e
dgecolor='w', orientation='landscape', papertype=None, format=None,t
ransparent=False, bboxinches=None, padinches=0.1,frameon=None, bbox_
inches='tight')
plt.show()

print("Melhor K validação: ", Melhor_K_validacao)
print("Melhor K teste:", Melhor_K_teste)
```

Exercício 2

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression, RidgeCV
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Norm
alizer, PowerTransformer

manchas_solares = pd.read_csv('/content/drive/My Drive/Colab Noteboo
ks/monthly-sunspots.csv')

for k in np.arange(1, 9):
  manchas_solares['x[n-
'+str(k)+']'] = manchas_solares['Monthly Mean Total Sunspot Number']
  manchas_solares['x[n-'+str(k)+']'] = manchas_solares['x[n-
'+str(k)+']'].shift(k)

datas = np.array(manchas_solares.iloc[3132:, 1:2]).reshape(1, -1)
datas = datas[0]

manchas_solares_x = manchas_solares.iloc[8:, 3:11]
manchas_solares_y = manchas_solares.iloc[8:, 2:3]

x = np.asarray(manchas_solares_x)
y = np.asarray(manchas_solares_y)

np.random.seed(3)
W = np.random.uniform(low=-0.01, high= 0.01, size=(8,100))

WT_x_X = np.zeros((len(x),100))
for T in np.arange(0, len(x)):
```

```
  WT_x_X[T,:] = np.dot(W.T, x[T,:].reshape(-1, 1)).reshape(1, -1)

WtX = (2.7 / np.max(np.abs(WT_x_X))) * WT_x_X

X = np.tanh(WtX)

rmse_validation_medium_array = []
rmse_train_medium_array = []
rmse_test_medium_array = []
alpha_best_medium_array = []
rmse_test_minimum = 1000000
rmse_validation_minimum = 1000000

for T in np.arange(1, 101):
  linha_maxima_kfold = 3124

  Xt = X[:linha_maxima_kfold,0:T]
  _y = y[:linha_maxima_kfold,:]
  Xt_teste = X[linha_maxima_kfold:,0:T]
  _y_teste = y[linha_maxima_kfold:,:]

  kf = KFold(n_splits = 10)

  rmse_validation_array = []
  rmse_train_array = []
  rmse_test_array = []
  alpha_best_array = []

  for train_index, validation_index in kf.split(Xt):
    X_train, X_validation = Xt[train_index], Xt[validation_index]
    y_train, y_validation = _y[train_index], _y[validation_index]

    linear_regression = RidgeCV(alphas=list(np.arange(0.01, 10.1, 0.
01)))
    linear_regression.fit(X_train, y_train)

    print('Melhor alpha: ',linear_regression.alpha_)
    alpha_best = linear_regression.alpha_
    alpha_best_array.append(alpha_best)

    prediction_validation = linear_regression.predict(X_validation)
    predicition_train = linear_regression.predict(X_train)
    prediction_test = linear_regression.predict(Xt_teste)

    rmse_validation = np.sqrt(mean_squared_error(y_validation,predic
tion_validation))
    rmse_train = np.sqrt(mean_squared_error(y_train, predicition_tra
in))
    rmse_test = np.sqrt(mean_squared_error(_y_teste, prediction_test
))

    rmse_validation_array.append(rmse_validation)
    rmse_train_array.append(rmse_train)
```

```
    rmse_test_array.append(rmse_test)

    if T == 92:
      if (rmse_validation_minimum > rmse_validation):
        rmse_validation_minimum = rmse_validation
        prediction_validation_best = linear_regression.predict(Xt_te
ste)
        best_all_alpha_validation = alpha_best
        print("Melhor lambda de validação:",best_all_alpha_validatio
n)
        Melhor_T_validacao = T

    if T == 16:
      if (rmse_test_minimum > rmse_test):
        rmse_test_minimum = rmse_test
        prediction_test_best = prediction_test
        best_all_alpha_test = alpha_best
        Melhor_T_teste = T

    print("Número de atributos T = " + str(T))
    print("Erro RMS de treino da interação = ", rmse_train)
    print("Erro RMS de validação da iteração = ", rmse_validation)
    print("Erro RMS de teste da iteração = ", rmse_test)

  rmse_validation_medium = np.mean(rmse_validation_array)
  rmse_train_medium = np.mean(rmse_train_array)
  rmse_test_medium = np.mean(rmse_test_array)
  alpha_best_medium = np.mean(alpha_best_array)

  rmse_validation_medium_array.append(rmse_validation_medium)
  rmse_train_medium_array.append(rmse_train_medium)
  rmse_test_medium_array.append(rmse_test_medium)
  alpha_best_medium_array.append(alpha_best_medium)

print('valor minimo rmse treino:',min(rmse_train_medium_array))
print('Valor minimo rmse validação:',min(rmse_validation_medium_arra
y))
print('Valor minimo rmse teste:',min(rmse_test_medium_array))

print('Melhor quantidade de parâmetros T no treino:',rmse_train_medi
um_array.index(min(rmse_train_medium_array))+1)
print('Melhor quantidade de parâmetros T na validação:',rmse_validat
ion_medium_array.index(min(rmse_validation_medium_array))+1)
print('Melhor quantidade de parâmetros T no teste:',rmse_test_medium
_array.index(min(rmse_test_medium_array))+1)

print('Melhor lambda validação:', best_all_alpha_validation)
print('Melhor lambda teste:', best_all_alpha_test)

print(rmse_train_medium_array)
print(rmse_validation_medium_array)
print(rmse_test_medium_array)
```

```
parametros_T = np.arange(1, 101)

plt.title('RMSE médio de treino e validação')
plt.ylabel('RMSE')
plt.xlabel(r'$\ Atributos \ T  \ (1\  \leq K \leq 100 \ ) $')
plt.grid(True)
plt.plot(parametros_T, rmse_train_medium_array, color = 'b', label='
RMSE Treino')
plt.plot(parametros_T, rmse_validation_medium_array, color = 'r', la
bel='RMSE Validação')
plt.legend()
plt.show()

plt.title('RMSE médio de teste')
plt.ylabel('RMSE')
plt.xlabel(r'$\ Atributos \ T  \ (1\  \leq K \leq 100 \ ) $')
plt.grid(True)
plt.plot(parametros_T, rmse_test_medium_array, color = 'y', label='R
MSE Teste')
plt.legend()
plt.show()

plt.title(r'$\ Parâmetro \ de \ regularização \ \lambda$')
plt.ylabel(r'$ \lambda $')
plt.xlabel(r'$\ Atributos \ T  \ (1\  \leq K \leq 100 \ ) $')
plt.plot(parametros_T, alpha_best_medium_array, color = 'g')
plt.show()

#print(np.sqrt(mean_squared_error(_y_teste, prediction_test_best)))
#plt.plot(_y_teste, color = 'b')
#plt.plot(prediction_test_best, color = 'r')
#plt.show()

print('RMSE de validação mínimo : ',np.sqrt(mean_squared_error(_y_te
ste, prediction_validation_best)))
plt.title('Série de manchas solares, conjuto de dados de teste')
plt.ylabel('Número de manchas solares por mês')
plt.xlabel('Tempo (anos)')
plt.grid(True)
plt.plot(datas,_y_teste, color = 'b', label='Série original')
plt.plot(datas,prediction_validation_best, color = 'r', label='Predi
ção, T = 92')
plt.xticks(rotation=45)
plt.xticks(datas[np.arange(0,len(datas),12)])
plt.legend(loc='best')
plt.savefig('figura_manchassolares1.png', dpi=1080, facecolor='w', e
dgecolor='w', orientation='landscape', papertype=None, format=None,t
ransparent=False, bboxinches=None, padinches=0.1,frameon=None, bbox_
inches='tight')
plt.show()

print('RMSE de teste mínimo : ',np.sqrt(mean_squared_error(_y_teste,
 prediction_test_best)))
```

```
plt.title('Série de manchas solares, conjuto de dados de teste')
plt.ylabel('Número de manchas solares por mês')
plt.xlabel('Tempo (anos)')
plt.grid(True)
plt.plot(datas,_y_teste, color = 'b', label='Série original')
plt.plot(datas,prediction_test_best, color = 'y', label='Predição, T
 = 16')
plt.xticks(rotation=45)
plt.xticks(datas[np.arange(0,len(datas),12)])
plt.legend(loc='best')
plt.savefig('figura_manchassolares2.png', dpi=1080, facecolor='w', e
dgecolor='w', orientation='landscape', papertype=None, format=None,t
ransparent=False, bboxinches=None, padinches=0.1,frameon=None, bbox_
inches='tight')
plt.show()
print("Melhor T validação: ", Melhor_T_validacao)
print("Melhor lambda validação: ",best_all_alpha_validation)
print("Melhor T teste:", Melhor_T_teste)
print("Melhor lambda teste: ",best_all_alpha_test)
```