IA048 - Exercícios de Fixação de Conceitos 1

Kaleb Roncatti de Souza **RA: 171360**
Gabriel Teixeira Callado **RA: 168172**

Parte 1 - Atividades Teóricas

1. Duas variáveis aleatórias binárias X e Y nos foram fornecidas conjuntamente.

   (a) Para obtermos as probabilidades associadas a cada variável separadamente (P(X) e P(Y)) somaremos sobre as demais variáveis da seguinte forma:

   $$P_X(x) = \sum_y P_{XY}(x, y)$$

   Assim sendo,

   $$P_X(X = 0) = P_{XY}(X = 0, Y = 0) + P_{XY}(X = 0, Y = 1) = 0.5 + 0.05 = 0.55$$

   $$P_X(X = 1) = P_{XY}(X = 1, Y = 0) + P_{XY}(X = 1, Y = 1) = 0.3 + 0.15 = 0.45$$

   $$P_X(X) = \begin{cases} 0.55 & \text{if X} = 0 \\ 0.45 & \text{if X} = 1 \end{cases}$$

   Analogamente para Y:

   $$P_Y(Y = 0) = P_{XY}(X = 0, Y = 0) + P_{XY}(X = 1, Y = 0) = 0.5 + 0.3 = 0.8$$

   $$P_Y(Y = 1) = P_{XY}(X = 0, Y = 1) + P_{XY}(X = 1, Y = 1) = 0.05 + 0.15 = 0.45$$

   $$P_Y(Y) = \begin{cases} 0.8 & \text{if Y} = 0 \\ 0.2 & \text{if Y} = 1 \end{cases}$$

   Como esperado, ambas as somas das probabilidades sobre todos os casos de cada variável em X e Y resultam em 1.

   (b) Para uma probabilidade condicional basta utilizarmos que:

   $$P_{X|Y}(X = 1|Y = 1) = \frac{P_{XY}(X = 1, Y = 1)}{P_Y(Y = 1)} = \frac{0.15}{0.2} = 0.75$$

   (c) Para determinar se duas variáveis são descorrelacionadas ou não podemos usar uma ferramenta chamada **covariância** entre duas variáveis. A covariância pode ser calculada em função da esperança matemática da seguinte forma:

   $$cov(X, Y) = E[XY] - E[X]E[Y]$$

   Para o cálculo da esperança do produto, sabemos que a esperança de XY só será não nula quando ambas as variáveis forem 1. Com relação ao produto das esperanças, a única ocasião em que tal produto não se anula é quando ambas variáveis são também iguais a 1 resultando em

   $$cov(X, Y) = 1 \times 1 \times 0.15 - 1 \times 0.45 \times 1 \times 0.2 = 0.06$$

   Assim sendo, não podemos afirmar que as variáveis são descorrelacionadas.

(d) Podemos afirmar que as variáveis **NÃO** são estatisticamente independentes. Pois através da teoria de probabilidades para variáveis acopladas podemos realizar a seguinte afirmação:

$$\text{Se X e Y são independentes} \Rightarrow cov(X,Y) = 0$$

Se escrevermos a contrapositiva da expressão lógica acima teremos que,

$$\text{Se } cov(X,Y) \neq 0 \Rightarrow \text{X e Y NÃO são independentes}$$

Sabemos do item anterior que $cov(X,Y) \neq 0$, portanto X e Y não são independentes.

(e) Para calcular a entropia conjunta (H(X, Y)) podemos utilizar diretamente a definição:

$$H(X,Y) = -\sum_x \sum_y p(x,y) log_2[p(x,y)]$$

$$H(X,Y) = -[0.5log_2(0.5) + 0.05log_2(0.05) + 0.3log_2(0.3) + 0.15log_2(0.15)] = 1.6477309$$

Para a entropia associada a cada uma das variáveis também podemos utilizar diretamente a definição:

$$H(X) = -\sum_x p(x) log_2[p(x)] = -[0.55log_2(0.55) + 0.45log_2(0.45)] = 0.992774$$

$$H(Y) = -\sum_y p(y) log_2[p(y)] = -[0.8log_2(0.8) + 0.2log_2(0.2)] = 0.721928$$

Finalmente, calcularemos as entropias condicionais mais uma vez através da definição:

$$H(X|Y) = -\sum_x \sum_y p(x,y) log_2[p(x|y)] =$$

$$= -[p(x=0,y=0) log_2 \frac{p(x=0,y=0)}{p(y=0)} + p(x=1,y=1) log_2 \frac{p(x=1,y=1)}{p(y=1)} +$$

$$+ p(x=1,y=0) log_2 \frac{p(x=1,y=0)}{p(y=0)} + p(x=0,y=1) log_2 \frac{p(x=0,y=1)}{p(y=1}] =$$

$$= -[0.5log_2 \frac{0.5}{0.8} + 0.15log_2 \frac{0.15}{0.2} + 0.3log_2 \frac{0.3}{0.8} + 0.05log_2 \frac{0.05}{0.2}] = 0.925803$$

Analogamente,

$$H(Y|X) = -\sum_x \sum_y p(x,y) log_2[p(y|x)] =$$

$$= -[p(x=0,y=0) log_2 \frac{p(x=0,y=0)}{p(x=0)} + p(x=1,y=1) log_2 \frac{p(x=1,y=1)}{p(x=1)} +$$

$$+ p(x=1,y=0) log_2 \frac{p(x=1,y=0)}{p(x=1)} + p(x=0,y=1) log_2 \frac{p(x=0,y=1)}{p(x=0)}] = 0.654956$$

Vale ressaltar que os valores calculados estão de acordo com o esperado. A entropia H(X,Y) deve ser a maior de todas pois, como tratamos ambas as variáveis ao mesmo tempo, ela é a que traz o maior nível de incerteza. Quando tratamos as variáveis separadamente, a entropia diminui. Notamos ainda que H(X) ¿ H(Y) pois a distribuição de probabilidades de X é mais esparsa que Y. Por fim, a entropia H(Y—X) é a menor de todas pois é a que carrega a menor quantidade de informação: ao conhecermos o valor da variável X, torna-se mais previsível a variável Y.

(f) Para o cálculo da informação mútua, podemos usar que:

$$I(X,Y) = H(X) - H(X|Y) = 0.9927 - 0.9258 = 0.0669$$

que pode também ser calculado/verificado usando-se:

$$I(X,Y) = H(Y) - H(Y|X) = \sum_x \sum_y p(x,y) log_2 \frac{p(x,y)}{p(x)p(y)} = 0.0669$$

Parte 2 – Atividade computacional

Nesta atividade computacional, trabalhamos com a predição de séries temporais com base no modelo mais simples de **regressão linear** visto em curso. A atividade foi realizada em *Python* através da biblioteca *sklearn*, a qual é comumente utilizada em trabalhos com aprendizado de máquina.

Para construir um modelo de previsão linear, normalmente partimos da seguinte expressão:

$$\hat{y}(n) = \sum w_j x_j(n) + w_0$$

Sendo $\hat{y}$(n) o valor à ser previsto e os pesos w_j e coeficiente linear w_0 elementos à serem determinados.

Normalmente, quando trabalhamos computacionalmente, é comum utilizarmos a notação matricial para apresentar o modelo acima, assim como comprimir o termo w_0 na própria matriz de pesos e adicionar uma coluna/linha de elementos unitários em x. Deste modo, podemos escrever a forma matricial:

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

Para uma previsão, sempre é preciso definir uma função que caracteriza a qualidade dos valores previstos ($\hat{y}_i$) com relação aos valores reais aos quais temos acesso (y_i). Através da abordagem de máxima verossimilhança que define o "ruído"do dataset como Gaussiano, chegamos numa função *Loss* quadrática que é convexa e possue solução fechada em sua minimização da seguinte forma:

$$\hat{\mathbf{y}} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$$

Neste exercício em específico a técnica descrita acima será usada ao pé da letra, no entanto, quando o tamanho do *dataset* fornecido for muito grande, nem sempre aplicar as inversões matriciais é a melhor opção pela complexidade da solução apresentar $\mathcal{O}(n^3)$. Neste caso, recorreríamos a técnicas iterativas tal como o gradiente descente.

Para a realização de todos os exercícios, os dados a serem utilizados foram separados em duas pastas de diferentes tamanhos, uma pasta de **treinamento** representando a maior porção do *dataset* e outra pasta de **teste** referente aos últimos dez anos (2010-2019) de informações a respeito de manchas solares.

1. **Exercício 1**

   Para a previsão de uma série temporal usando regressão linear, algumas informações precisam ser pré-definidas e os dados precisam estar dispostos de uma maneira bem particular.

   Considere que o vetor ($\mathbf{v}$) de tamanho N fornecido para as manchas solares em função do tempo possuiua uma forma estritamente linear, e como o tempo entre cada coleta era de um mês, o primeiro passo foi desconsiderar o elemento temporal e trabalhar em cima do vetor $\mathbf{v}$.

   A segunda informação de extrema importância para a montagem do modelo é o número de ***K* amostras passadas** dados de input ao modelo para a previsão do ponto subsequente. Suponha que utilizaremos $K = 5$ pontos para a previsão do sexto ponto por motivos de simplificação. Desta forma, precisamos tratar o vetor $\mathbf{v}$ de forma a predispor os dados numa maneira da qual seja possível aplicar o modelo de regressão linear.

Assim sendo, considerando-se $\mathbf{v}$ da forma

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 & ... & v_N \end{bmatrix}$$

podemos construir um modelo linear que une as informações relativas aos 5 primeiros pontos como sendo algo da seguinte forma em sua primeira linha:

$$\hat{y}_1 = w_{11}x_1^1 + w_{12}x_1^2 + ... + w_{15}x_1^5 + w_{10}1$$

que quando comparado com $\mathbf{v}$ tería-se $x_1^1 = v_1, ..., x_1^5 = v_5$ e portanto $\hat{y}_1$ a previsão do elemento v_6.

Analogamente para a segunda linha do modelo, iríamos *shiftar* em uma posição para a direita no vetor $\mathbf{v}$ para realizarmos a próxima predição:

$$\hat{y}_2 = w_{21}x_2^1 + w_{22}x_2^2 + ... + w_{25}x_2^5 + w_{20}1$$

que quando comparado com $\mathbf{v}$ tería-se $x_2^1 = v_2, ..., x_2^5 = v_6$ e portanto $\hat{y}_2$ a previsão do elemento v_7, e assim sucessivamente para as próximas linhas.

Teríamos então N-5 sequências/linhas de 5 pontos (não podemos usar os 5 últimos pontos como entrada pois não conseguiríamos prever o seguinte), cada uma delas *shiftada* de um elemento para a direita no vetor $\mathbf{v}$, e com isso, cada um desses 5 pontos poderia ser enxergado como uma *feature/dimensão* diferente do modelo de regressão sendo x_i^j o i-ésimo ponto da j-ésima dimensão. As matrizes $\mathbf{x}$ e $\mathbf{y}$ ficariam disposta da seguinte maneira:

$$\mathbf{x} = \begin{bmatrix} x_1^1 & x_1^2 & ... & x_1^5 & 1 \\ x_2^1 & x_2^2 & ... & x_2^5 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-5}^1 & x_{N-5}^2 & ... & x_{N-5}^5 & 1 \end{bmatrix}_{(N-5)\times(6)}$$

$$\mathbf{y} = \begin{bmatrix} y_1 = v_6 \\ \vdots \\ y_N = v_N \end{bmatrix}_{(N-5)\times(1)}$$

Treinamos então vários modelos variando $\boldsymbol{K}$ entre 1 e 24 como determinado pelo enunciado. Para a escolha do melhor modelo escolhemos o critério conhecido como *Root Mean Squared Error* que pode ser descrito da seguinte forma:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y - \hat{y})^2}$$

Além do mais, utilizamos uma técnica conhecida como **k-fold Cross Validation** para a validação dos dados, técnica recorrente nos trabalhos científicos no campo de aprendizado de máquina. Nesta técnica, separa-se os dados de treinamento em **k** pastas e realiza-se k treinamentos diferentes. Em cada um dos treinamentos, $k - 1$ pastas são usadas para efetivamente treinar o modelo e a pasta restante é usada para a validação, sendo estas pastas permutadas a CADA treinamento.

No nosso caso, tiramos a média dentre os erros *RMS* dos **k** treinamentos da técnica k-Fold e, por fim, utilizamos o modelo que apresenta o menor erro *RMS* num conjunto de dados de **teste** que o modelo NUNCA tinha visto anteriormente. A ideia da técnica k-fold é justamente impedir que o modelo sobreajuste esses dados vistos e tenha dificuldade em prever dados não vistos - fenômeno conhecido como ***overfitting***.

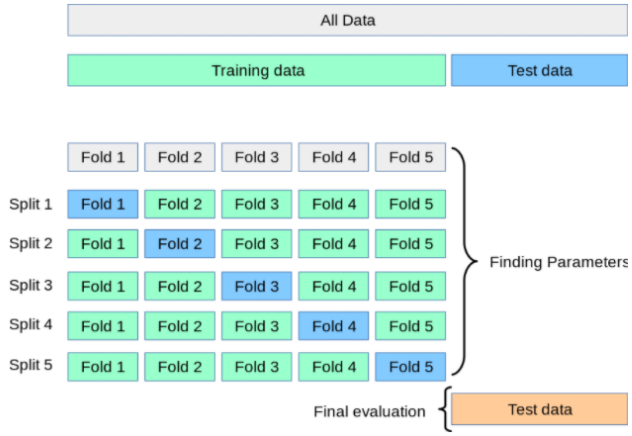

Figura 1: Modelo de *k-fold Cross-Validation* extraído da documentação da biblioteca *sklearn*

Assim sendo, exploramos a princípio como a média dos erros RMS evolue em função do número de pastas. Para tal tarefa, fixamos o valor K de atrasos e apenas variamos o número de pastas (k) da técnica k-fold, o resultado é mostrado na Figura 2.

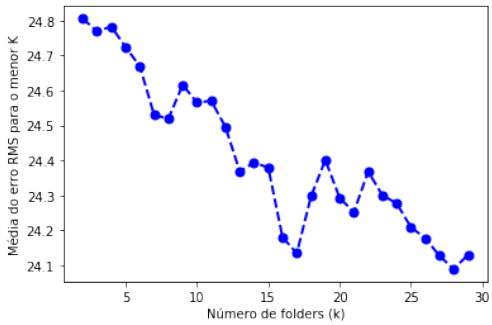

Figura 2: Média dos erros RMS com K fixado em função do número de pastas (k).

Notamos que, quanto maior o número de pastas, menor aparenta ser o erro da nossa predição. Discutindo-se intuitivamente, quanto maior o número de pastas, maior será o conjunto de dados dado ao modelo para treinar - e, ainda, menor é o número de dados de validação. Tendo mais dados de treino e menos dados de validação, o erro RMS diminui pois o modelo se generaliza melhor com mais dados. No entanto, o aumento no número de pastas implica numa maior variância, o que possivelmente pode levar os dados à condição de overfitting, visto que ele "aprenderá"cada vez melhor os dados de treino. Além do mais, o tempo de execução computacional aumenta conforme aumentamos o número de pastas. Assim sendo, prosseguimos com o valor 10 pastas para as próximas etapas do exercício.

Tendo fixado o valor de k-fold em 10 pastas, aplicamos a técnica e observamos a média do erro RMS variar em função do número de atrasos K como mostrado na Figura 3. Chegamos à conclusão de que o valor de K *sunspots* anteriores que minimiza o erro RMS para a previsão do próximo sunspot (K+1) é de K = 24.

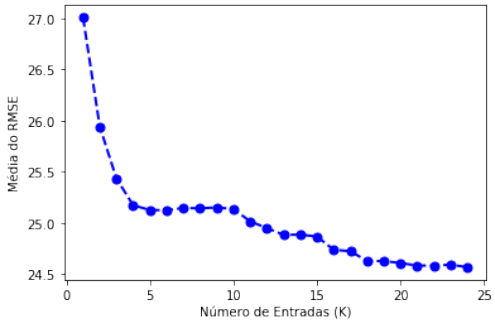

Figura 3: Média dos erros RMS em função do número de atrasos (K).

Após a obtenção do K-optimal, retreinamos o modelo com TODO o conjunto de treinamento (ou seja, não separamos mais em k-pastas) e por fim aplicamos o modelo para previsão dos dados de teste. Em seguida, comparamos os valores previstos com os verdadeiros para avaliarmos se o modelo performou de maneira adequada na previsão das manchas solares em função do tempo. O resultado é mostrado na Figura 4.

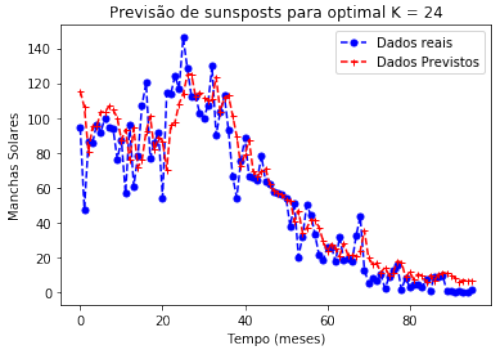

Figura 4: Previsão das manchas solares ($\hat{y}$) e dados reais de mancha solar (y) em função do tempo.

O modelo observado na figura anterior apresentou um erro RMS de aproximadamente 16.24 quando performado no *dataset* de teste. Observando-se este valor e comparando-o com o valor médio das manchas solares (em torno de 60,43), podemos concluir que o erro relativo é em torno de 26% e que modelo performou de maneira razoável. Podemos ainda ir um pouco além e verificar o erro RMS para CADA ponto previsto do conjunto de teste como mostrado na Figura 5.

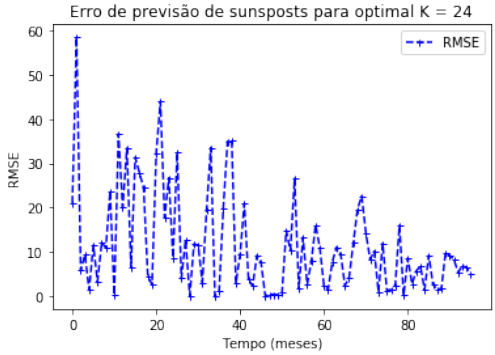

Figura 5: Erro RMS de cada previsão no conjunto de teste em função do tempo.

Observando-se o erro RMS variar no tempo, pode-se concluir que, aparentemente, o erro diminuiu próximo ao fim do *dataset*. Isso se deve ao comportamento das manchas solares nos últimos meses do conjunto de teste. Notamos claramente que o comportamento das manchas solares na Figura 4 é muito mais estável no final quando comparado com os primeiros meses - de modo que a previsão no final seja mais simples para nosso modelo.

2. **Exercício 2**

Para tal atividade, continuamos explorando o modelo de regressão linear, porém, utilizamos como entradas transformações não-lineares do vetor **x**. Anteriormente, o modelo de predição utilizado era puramente linear da forma $\hat{y} = \mathrm{w}^{\mathrm{T}}\mathrm{x}$, com **x** de tamanho (N,K+1). Para esta atividade, inverteremos a matriz **x** para que as dimensões se encaixem harmonicamente e não iremos inserir a coluna unitária direto na matriz **x**, mas sim, após a aplicação da não-linearidade.

Assim sendo, nesta atividade utilizamos **x** de tamanho (K,N) e agora cada variável que anteriormente era representada por x_k na matriz de variáveis será substituída por x'_k da forma:

$$x'_k(n) = tanh(w_k^T x(n))$$

sendo w_k um vetor de pesos com uma distribuição uniforme.

Por conseguinte, w^{T} tem dimensão (T,K) e x(n) tem dimensão (K,N) de modo que podemos construir uma matriz x′(n) que terá dimensões de (T,N), reduzindo o problema novamente à uma regressão linear, salvo que agora as entradas do problema não serão diretamente os valores das manchas solares, e sim não-linearidades (tangente hiperbólica) aplicadas à estas manchas. Então, T pode ser visto como o número de atributos/features/variáveis do nosso novo modelo. Neste item em específico, fixamos $K = 8$ e iremos variar o valor de T.

Note que nosso novo problema está **transposto** com relação ao item anterior onde tínhamos uma **x** de dimensões (N,T). Por fim, adiciona-se uma linha de 1's na matriz x′(n) para diluir o termo do coeficiente linear do modelo dentro da própria matriz.

Quando construímos o modelo e aplicamos a não-linearidade, logo percebemos que a função tangente hiperbólica possui um caráter bem particular e que deve ser levado em conta. Quando aplicamos a tangente hiperbólica à um valor muito acima de 3, a tendência é nos retornar algo cada vez mais próximo de 1 pois $\lim_{\mathrm{x}\to\infty} \tanh(\mathrm{x}) = 1$. Da mesma forma, quando aplicamos a tangente hiperbólica à um valor muito abaixo de -3, a tendência da função é nos retornar algo cada vez mais próximo de -1 pois $\lim_{\mathrm{x}\to-\infty} \tanh(\mathrm{x}) = -1$. Ou seja, ao aplicarmos diretamente a não-linearidade sem nenhuma normalização, percebemos que os dados eram concentrados em torno de -1, gerando uma perda de informações significável que seria um fator determinante na qualidade do modelo treinado. Além disso, devemos tomar cuidado com a normalização pois, se escolhermos um range pequeno (por exemplo, entre -0.2 e 0.2), estaríamos com os valores concentrados em uma parte da função tanh que ela tem

comportamento quase linear - perdendo, assim, o sentido de utilizar uma função não linear no nosso modelo.

Após alguns testes, percebemos que o valor mínimo e máximo da normalização era um fator preponderante para o desempenho do modelo. Devido à isso, fizemos um código na qual variamos o valor mínimo e máximo da normalização e, logo em seguida, encontramos o valor de erro para o melhor T e alfa (da rigde regression) do problema. Por fim, plotamos o gráfico abaixo para visualizar o erro em função do valor da normalização:

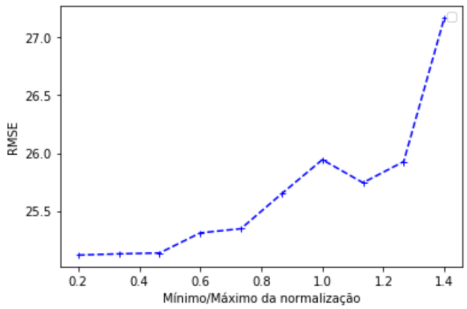

Figura 6: Erro do modelo em função da normalização dos dados

Observando o gráfico exposto, notamos que o erro do modelo aumenta substancialmente conforme aumentamos o valor do mínimo e máximo da normalização. Com tal imagem em mãos, o range de (-0.5,0.5) foi escolhido para a normalização dos dados.

Após a normalização, realizamos o mesmo procedimento do exercício anterior aplicando a técnica k-fold Cross-Validation escolhendo novamente k = 10 pastas. Porém, nesta atividade, utilizamos uma técnica de regularização conhecida como Ridge Regression. A ideia geral da técnica é adicionar um termo de penalização na função *Loss* quadrática (termo proporcional à um α, parâmetro de "enrijecimento"do modelo) utilizada previamente na regressão linear, este termo de penalização irá "enrijecer"o modelo e auxiliar na generalização do modelo.

Variamos então o número de atributos não-lineares T e medimos a média dos erros RMS para cada um dos valores de T com um valor fixo do coeficiente ridge $\alpha = 0.1$ como mostrado na Figura 7. Observe que o modelo atinge rapidamente uma aparente saturação em torno de 40 atributos. A escolha inicial do parâmetro α foi arbitrária para motivos de visualização da performance do modelo.

Note que, para a apresentação dos erros RMS, foi necessário voltar à distribuição original dos dados/sunspots através de uma "desnormalização", visto que tínhamos realizado uma normalização no começo.

Como próxima etapa, estudamos a variação do parâmetro de regularização-ridge (α) para cada atributo

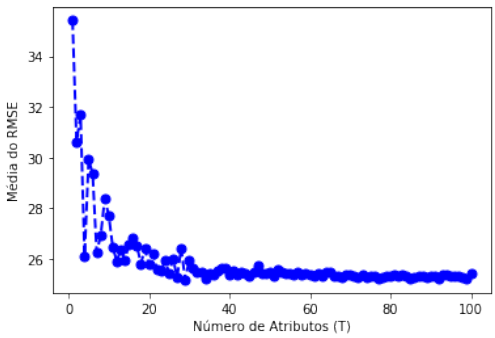

Figura 7: Média dos erros RMS em função do número de atributos (T). Coeficiente-ridge fixo $\alpha = 0.1$

T, assim sendo, escolhemos uma gama de variação para α entre 0 e 2 pois é uma escolha típica e escolhemos o resultado que apresentou menor média RMS para cada um dos T atributos, como mostrado na Figura 8.

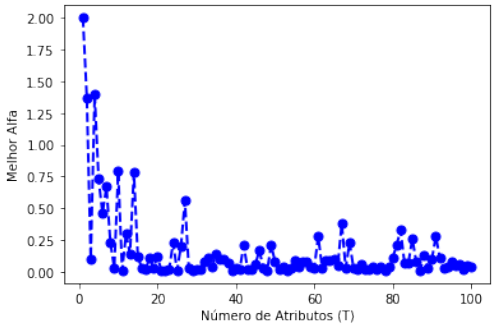

Figura 8: Alpha com menor RMS em função do número de atributos (T)

Assim sendo, fomos capazes de escolher os melhores hiperparâmetros dentro do *range* estimado, obtendo a mínima média de RMSE encontrada para o conjunto de treinamento de aproximadamente 25.19 para o valor T = 95 com parâmetro de regularização $\alpha = 0.08$.

Por fim, assim como no Exercício 1, retreinamos um modelo com os melhores hiperparâmetros encontrados desta vez usando TODO o conjunto de treinamento e realizamos a previsão dos dados de teste do novo modelo, o resultado final é apresentado na Figura 9 e o erro RMS para todos os meses do dataset de teste é mostrado na Figura 10.

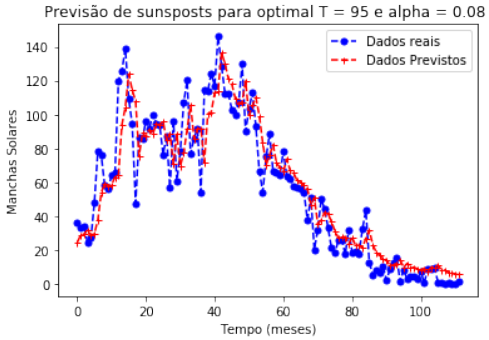

Figura 9: Alpha com menor RMS em função do número de atributos (T)

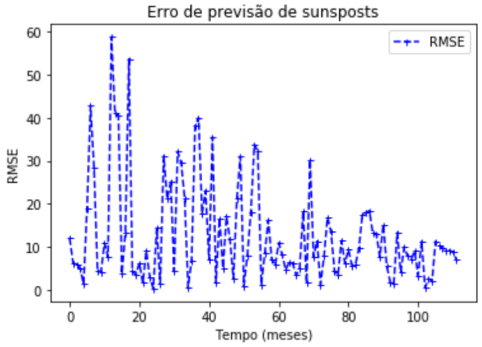

Figura 10: Erro de previsão em função do tempo para os dados de teste

Por fim, notamos que o erro do melhor modelo proposto no exercício 2 é maior que o erro do modelo do exercício 1. Há alguns pontos à serem considerados para entendermos o motivo da diferença de desempenho entre os modelos. No modelo do exercício 2, temos uma complexidade maior exigida do modelo pois fizemos uma transformação não-linear nos dados de entrada. Podemos ver que o modelo tem uma certa dificuldade em lidar com isso quando visualizamos a Figura 6: o modelo performa melhor quando a normalização é em um range pequeno, concentrando-se na região mais linear da função tanh. Um outro ponto relevante é com relação ao tipo de regressão proposto no exercício 2. Pela Figura 8, podemos inferir que a penalização do modelo pelo fator alfa passa a ser importante e necessária quando o modelo utilizado tem um número menor de atributos T como entrada. Entretanto, como vemos na Figura 7 e como visto no exercício 1, o modelo costuma performar melhor quando damos um valor grande de atributos T na entrada. Desta forma, com um valor T grande, o modelo não necessita de um valor de alfa relevante para performar bem.