Aprendizado de máquina para redução de viés na estimativa da temperatura máxima do dia seguinte

Vitor Mazal Krauss[1]

[1]Programa de Engenharia de Sistemas e Computação
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia
Universidade Federal do Rio de Janeiro

1. Introdução

A previsão das temperaturas mínima e máxima de um dia é de relevância por variados motivos. No caso da imprensa, por exemplo, estas quantidades são geralmente parte dos quadros diários de previsão do tempo e demonstram ser de interesse do leitor ou telespectador. No caso de lojas e serviços esta informação tem relevância no planejamento a curto prazo e operação do serviço. Por exemplo, certos restaurantes, hotéis e pontos turísticos podem esperar uma demanda maior em dias mais quentes e ensolarados e, portanto, se preparar devidamente para um dia serviço com base na previsão do tempo. Além disso, este tipo de previsão também pode auxiliar autoridades no planejamento e resposta a ondas de calor ou de frio. De fato, nas últimas décadas foram registradas diversas ocorrências de ondas de calor e frio em variados países de diferentes continentes e hemisférios [1, 2]. Efetivamente, a previsão das temperaturas mínima e máxima são essenciais para a prevenção e mitigação dos riscos ligados a estes tipos de fenômeno [3].

Na Coréia do Sul, o *Korea Metereological Administration* (KMA), órgão nacional de serviços metereológicos, opera um modelo numérico de previsão do tempo chamado de *LDAPS*, sigla que denota "local data assimilation and prediction system" [4]. O LDAPS, assim como outros modelo numéricos de previsão do tempo, proporciona estimativas para diferentes medidas metereológicas, como temperaturas mínima e máxima, humidade relativa, velocidade do vento, nebulosidade, precipitação, etc. Naturalmente, as estimativas do LDAPS não correspondem de maneira exata a realidade e por isso dizemos que o modelo tem viés. Levando-se em conta o viés do LDAPS e a importância de uma maior precisão nas previsãoes do tempo em comparação à realidade, torna-se relevante o problema de redução de viés do modelo. Historicamente, as duas técnicas mais comuns empregados para esta redução são *model output statistics* e *filtros de Kalman* [3]. Por outro lado, nos últimos anos foram publicados diferentes artigos relatando o uso de técnicas de aprendizado de máquina para a redução de viés em modelos numéricos de previsão do tempo. Alguns exemplos são [5, 6, 7] e uma discussão mais detalhada é apresentada em [3].

Neste trabalho, consideramos o problema de redução de viés do modelo LDAPS na previsão da temperatura máxima do dia seguinte. Para tal, empregaremos técnicas de aprendizado de máquina, mais especificamente de regressão. Os modelos considerados terão como dados de entrada as quantidades proporcionados pelo modelo LDAPS como previsão para o dia seguinte, além de algumas medições metereológicas realizadas no mesmo dia. E a variável de resposta é a temperatura máxima observada no dia seguinte. Em [3], os autores consideram o uso de aprendizado de máquina para redução de viés na previsão das temperaturas mínima e máxima. Novamente, neste trabalho consideramos apenas a temperatura máxima.

Na Seção 2 apresentamos o conjunto de dados. Na Seção 3 apresentamos a metodologia adotada para o tratamento dos dados, da análise exploratória dos dados e da engenharia de atributos. Na Seção 4 apresentamos as formulações matemáticas dos modelos de regressão considerados no trabalho. Na Seção 5 apresentamos os resultados dos modelos. E na Seção 6 apresentamos a conclusão. No Apêndice A apresentamos todas as

figuras, e no Apêndice B apresentamos as tabelas [1] .

2. Conjunto de Dados

O conjunto de dados considerada é apresentada em [3] e está disponível no *UCI Machine Learning Repository* [8] sob o nome de "Bias correction of numerical prediction model temperature forecast data set". Esta base contém dados coletados durantes os verões entre os anos de 2013 a 2018 na cidade de Seul, na Coréia do Sul. O conjunto de dados conta com 7752 registros e 25 atributos para cada registro. Na cidade de Seul estão situadas 25 estações metereológicas em localidades diferentes. Cada registro é respectivo a um dia na cidade de Seul e uma estação metereológica. Para cada registro, correspondem 25 atributos. Dois destes atributos são as temperaturas mínima e máxima medida naquela estação durante aquele dia. Quatorze destes atributos correspondem a previsões do LDAPS para o dia seguinte, incluindo temperatura, humidade relativa, precipitação, etc. Quatro destes atributos correspondem a latidude, longitude, elevação e o declive relativos à localização da estação. Dois destes atributos são as temperaturas mínima e máxima observadas naquela estação no dia seguinte. Em especial, a temperatura máxima observada naquela estação no dia seguinte será considerada como a variável resposta dos modelos de regressão considerados neste trabalho.

A seguir são apresentados os nomes de cada atributo no conjunto de dados original, além de uma breve descrição do atributo.

- station : denota o ID da estação. Cada uma das 25 estações é representada por um único ID, que é um número inteiro no conjunto $\{1, \ldots, 25\}$.
- Date : denota a data associada aquele registro.
- Present_Tmax : denota a temperatura máxima no dia do registro, em graus célcius.
- Present_Tmin : denota a temperatura máxima no dia do registro, em graus célcius.
- LDAPS_RHmax : denota a previsão do LDAPS para humidade relativa máxima no dia seguinte ao registro, em pontos percentuais.
- LDAPS_RHmin : denota a previsão do LDAPS para humidade relativa máxima no dia seguinte ao registro, em pontos percentuais.
- LDAPS_Tmax_lapse : denota a previsão do LDAPS para temperatura máxima no dia seguinte ao registro, em graus celcius.
- LDAPS_Tmin_lapse : denota a previsão do LDAPS para temperatura mínima no dia seguinte ao registro, em graus celcius.
- LDAPS_WS : denota a previsão do LDAPS para velocidade média de vento no dia seguinte ao registro, em metros por segundo.
- LDAPS_LH : denota a previsão do LDAPS para o calor latente no dia seguinte ao registro, watt por metro quadrado.
- LDAPS_CC1 : denota a previsão do LDAPS para nebulosidade percentual média durante o primeiro quarto do dia seguinte ao registro, em pontos percentuais.
- LDAPS_CC2 : denota a previsão do LDAPS para nebulosidade percentual média durante o segundo quarto do dia seguinte ao registro, em pontos percentuais.

[1] Neste relatório optamos por apresentar todas as figuras e tabelas no Apêndice A e B ao final do texto. Esta escolha foi feita para que as imagens pudessem ser exibidas em suas dimensões originais sem comprometer a continuidade e organização do texto. Uma sugestão para o leitor é abrir o relatório em duas janelas diferentes, uma para leitura do texto e outra para visualização das figuras.

- LDAPS_CC3 : denota a previsão do LDAPS para nebulosidade percentual média durante o terceiro quarto do dia seguinte ao registro, em pontos percentuais.
- LDAPS_CC4 : denota a previsão do LDAPS para nebulosidade percentual média durante o último quarto do dia seguinte ao registro, em pontos percentuais.
- LDAPS_PPT1 : denota a previsão do LDAPS para precipitação média durante o primeiro quarto do dia seguinte ao registro.
- LDAPS_PPT2 : denota a previsão do LDAPS para precipitação média durante o segundo quarto do dia seguinte ao registro.
- LDAPS_PPT3 : denota a previsão do LDAPS para precipitação média durante o terceiro quarto do dia seguinte ao registro.
- LDAPS_PPT4 : denota a previsão do LDAPS para precipitação média durante o último quarto do dia seguinte ao registro.
- lat : denota a latitude da estação na qual o registro foi coletado.
- lon : denota a longitude da estação na qual o registro foi coletado.
- DEM : denota a elevação da estação na qual o registro foi coletado, em metros.
- Slope : denota o declive do terreno no qual a estação está situada, em graus.
- Solar radiation : denota a irradiação solar durante o dia na estação na qual o registro foi coletado.
- Next_Tmax : denota a temperatura máxima observada no dia seguinte naquela estação.
- Next_Tmax : denota a temperatura mínima observada no dia seguinte naquela estação.

2.1. Tecnologia

Este trabalho foi feito com o uso da linguagem de programação *Python* [9] e de algumas das bibliotecas disponíveis para a linguagem. O uso da linguagem Python foi feito por meio do ambiente *Jupyter Notebook* por conta da iteratividade proporcionada por esta ferramenta. Em especial, foram utilizadas as seguintes bibliotecas:

- NumPy: para cálculos vetoriais e matriciais.
- Pandas: para leitura e tratamento dos dados.
- Matplotlib e Seaborn: para a ilustração de gráficos, histogramas, etc.
- Sklearn : para treinamento e avaliação dos modelos, padronização dos dados e particionamento do conjunto de dados em treino e teste e validação cruzada.

3. Metodologia

3.1. Tratamento dos Dados

Dos 7752 registros disponíveis no conjunto de dados, exatamente 164 registros estavam com pelo menos um atributo faltoso. Levando em conta que este número representa apenas $\frac{164}{7752} \approx 2.11\%$ do total dos dados, foi feita a opção pelo desconsideração destas 164 instâncias. Com isso, daqui em diante, entende-se como conjunto de dados o conjunto de dados original com exceção destas 164 instâncias com dados faltosos, totalizando 7588 instâncias.

3.2. Análise Exploratória de Dados

Nesta seção fazemos uma análise exploratória do conjunto de dados.

Começamos investigando a distribuição dos registros entre as diferentes estações. Esta distribuição é ilustrada na Figura 1. Como pode ser visto na Figura 1, para cada estação estão disponíveis em torno de 300 registros.

Na Figura 2 são apresentados os coeficientes de correlação de Pearson para cada par de atributos do conjunto de dados. Uma vez calculadas as correlações, se cada par de atributos seguir uma distribuição normal então poderíamos calcular p-valores e fazer testes de hipótese para a significância do coeficiente de correlação. Porém, o teste de Shapiro-Wilk para normalidade foi realizado para cada uma dos atributos numéricos e para todos estes atributos a hipótese de normalidade foi rejeitada. Uma vez que o cálculo de p-valores e o teste de significância do coeficiente de correlação tem como hipótese a normalidade dos dados, não faremos este tipo de análise. Sendo assim, a significância de cada coeficiente de correlação é feita com base no seu valor absoluto e em comparação com os valores dos coeficientes de correlação para os demais pares de atributos.

Neste momento, voltamos nossa atenção exclusivamente aos coeficientes de correlação pares pares incluindo a variável resposta, que é a temperatura máxima no dia seguinte, representada pela chave Next_Tmax. Isso é apresentado na Figura 3.

Na Figura 3, os atributos são apresentados em ordem decrescente do valor do coeficiente de correlação com Next_Tmax.

Levando em conta os valores absolutos de cada coeficiente de correlação e em comparação com os demais, consideramos que os seguintes atributos têm coeficiente de correlação significativo com a variável Next_Tmax: Present_Tmax, Present_Tmin, LDAPS_Tmax_lapse e LDAPS_Tmin_lapse, entre os valores positivos; LDAPS_RHmax, LDAPS_RHmin, LDAPS_CC1, LDAPS_CC2, LDAPS_CC3 e LDAPS_CC4, entre os valores negativos.

Note que o par de atributos Next_Tmax e Present_Tmax tem coeficiente de correlação positivo e relativamente significativo, com valor de 0.61. Na Figura 4 apresentamos o gráfico de dispersão entre estes dois atributos.

Note que o par de atributos Next_Tmax e Present_Tmin tem coeficiente de correlação positivo e relativamente significativo, com valor de 0.46. Na Figura 5 apresentamos o gráfico de dispersão entre estes dois atributos.

Note que o par de atributos Next_Tmax e LDAPS_Tmax_lapse tem coeficiente de correlação positivo e relativamente significativo, com valor de 0.84. Na Figura 6 apresentamos o gráfico de dispersão entre estes dois atributos.

Note que o par de atributos Next_Tmax e LDAPS_Tmin_lapse tem coeficiente de correlação positivo e relativamente significativo, com valor de 0.59. Na Figura 7 apresentamos o gráfico de dispersão entre estes dois atributos.

Note que o par de atributos Next_Tmax e LDAPS_RHmax tem coeficiente de correlação negativo e relativamente significativo, com valor de -0.29. Na Figura 8 apresentamos o gráfico de dispersão entre estes dois atributos.

Note que o par de atributos Next_Tmax e LDAPS_RHmin tem coeficiente de

correlação negativo e relativamente significativo, com valor de -0.45. Na Figura 9 apresentamos o gráfico de dispersão entre estes dois atributos.

Note que, para $i = 1, 2, 3, 4$ o par de atributos Next_Tmax e LDAPS_CCi tem coeficiente de correlação negativo e relativamente significativo, com valores de $-0.46, -0.50, -0.52, -0.45$, respectivamente. Na Figura 10 apresentamos o gráfico de dispersão entre cada par de atributos formado por Next_Tmax e LDAPS_CCi, para $i = 1, 2, 3, 4$.

Observamos que para $i = 1, 2, 3, 4$ o par de atributos Next_Tmax e LDAPS_PPTi não apresentou coeficiente de correlação significativo. Por outro lado, quando se observa o histograma dos atributos LDAPS_PPTi, para $i = 1, 2, 3, 4$, podemos notar que estes atributos têm distribuições assimétricas, com a maior parte dos registros com valor igual ou muito próximo de zero, i.e, com nenhuma ou pouca ocorrência de precipitação prevista pelo LDAPS. Por isso, optamos pela introdução de um novo atributo LDAPS_BINARY_RAINi, para cada $i = 1, 2, 3, 4$. Para cada $i = 1, 2, 3, 4$, LDAPS_BINARY_RAINi é binário e denota a ocorrência ou não de precipitação prevista pelo LDAPS para o i-ésimo quarto do dia seguinte. Na Figura 11 apresentamos, para $i = 1, 2, 3, 4$, os coeficientes de correlação para o par de atributos formado por Next_Tmax e LDAPS_BINARY_RAINi. Os valores destes coeficientes são $-0.28, -0.37, -0.32, -0.30$, respectivamente. Note que estes valores são maiores, em valor absoluto, do que os coeficientes de correlação para os pares de atributos formados por Next_Tmax e LDAPS_PPTi, para $i = 1, 2, 3, 4$.

Por outro lado, os atributos LDAPS_BINARY_RAINi, para $i = 1, 2, 3, 4$, apresentam coeficientes de correlação relativamente significativos com algumas das demais variáveis regressoras. Em especial, destacamos os coeficientes de correlação entre os atributos LDAPS_BINARY_RAINi e LDAPS_CCj para $i, j = 1, 2, 3, 4$. Estes valores são apresentados na Figura 12. E também destacamos os coeficientes de correlação entre os atributos LDAPS_BINARY_RAINi e LDAPS_RHmax e LDAPS_RHmin, para $i = 1, 2, 3, 4$. Estes valores são apresentados na Figura 13.

Também observamos que pares de atributos formados por LDAPS_CCi e LDAPS_CCj, para $i, j = 1, 2, 3, 4$, têm coeficiente de correlação positivo e relativamente significantes. Estes valores são apresentados na Figura 14.

E os pares de atributos formados por LDAPS_CCi, para $i = 1, 2, 3, 4$ e LDAPS_RHmax e LDAPS_RHmin também apresentaram coeficientes de correlação positivos e relativamente significativos. Estes valores são apresentados na Figura 15.

Observamos que os quatro atributos geográficos lat, lon, DEM e Slope apresentaram coeficiente de correlação com Next_Tmax relativamente pequenos, com valores de $-0.06, 0.0, -0.17, -0.1$, respectivamente. De fato, levando em conta que todas as 25 estações metereológicas estão situadas na mesma cidade, as latitudes e longitudes das estações diferem muito pouco quando se considera uma escala global.

3.3. Particionamento dos Dados

O conjunto de dados, que conta com um total de 7588 registros, foi particionado em conjuntos de treino e teste, cada um com 75% e 25% dos dados, respectivamente. Este particionamento foi feito de maneira estratificada para garantir que os números de regis-

tros disponíveis para diferentes estações sejam balanceados, tanto no conjunto de treino quanto de teste. O número de registros disponíveis para cada estação é apresentado na Figura 16 para o conjunto de treino e na Figura 17 para o conjunto de teste.

Além disso, o conjunto de treino foi particionado em 10 para validação cruzada com 10 *folds*. Seguindo as recomendações em [10], para cada ciclo da validação cruzada é ajustado uma transformação de padronização Z-score utilizando apenas conjunto de treino respectivo aquele ciclo. Uma vez ajustado, esta transformação é aplicada tanto ao conjuntos de treino e teste respectivos aquele ciclo.

3.4. Seleção de Atributos

Como alguns pares de variáveis regressoras têm coeficiente de correlação significativo, optamos por fazer um procedimento de seleção de atributos [10]. Em particular, optamos pelo procedimento de "backward-stepwise feature selection" [10]. Em geral, este procedimento é feito da seguinte maneira: considera-se um conjunto V_0 de variáveis regressoras tal que $|V_0| > 1$. Na k-ésima iteração, o conjunto de variáveis regressoras é V_k e determina-se a variável regressora $\underline{v} \in V_k$ de menor significância para predição da variável resposta, segundo algum critério escolhido. Elimina-se a variável $\underline{v}$, i.e, faz-se $V_{k+1} = V_k \setminus \{\underline{v}\}$. O procedimento continua enquanto $|V_k| \geq 1$. A ordem na qual as variáveis regressoras são eliminadas sugere uma ordem de significância destas variáveis para predição da variável resposta.

Há diversas possibilidades de escolha do critério para a determinação da significância de cada variável regressora. Por exemplo, em [10], os autores utilizam o Z-score da variável regressora. Outros exemplos são o uso do coeficiente de impuridade ou Gini de árvores aleatórias e o uso de máquinas de vetor de suporte [11]. Uma outra maneira comum é considerar um modelo de regressão linear. Com isso, a significância de cada variável regressora é dada pelo valor absoluto do coeficiente de regressão associado àquela variável. Em [11], referindo-se à efetividade de um modelo de classificação como medida da significância das variáveis regressoras, os autores escreveram: "We found the ability of a classifier for penalizing redundant features and promoting independent features in the recursive process has a strong influence on its success". Assim, neste trabalho, ao invés da regressão linear, vamos considerar um modelo Ridge ou Lasso, por conta da penalização imposta nos valores absolutos dos coeficientes pelas normas ℓ^2 e ℓ^1, respectivamente. Neste caso, vamos considerar como medida de significância de cada variável regressora o valor absoluto do coeficiente do modelo associado àquela variável. Note que, no caso do Lasso, um coeficiente pode tomar valor igual a 0, por conta da norma ℓ^1. Portanto, a cada iteração, será eliminada a variável regressora associada ao coeficiente de menor valor absoluto. Como os modelos Ridge e Lasso dependem de um hiperparâmetro α que regula o nível de penalização, a cada iteração determinaremos o melhor valor de $\alpha \in A$, para um conjunto finito A, por meio de validação cruzada. Será considerado o melhor valor de $\alpha \in A$ aquele valor $\alpha^* \in A$ que minimiza a média dos *root mean squared error* (RMSE) durante os 10 ciclos da validação cruzada. Uma vez determinado α^*, é então treinado o modelo utilizando o conjunto de treino e tendo como hiperparâmetro α^*. Os valores absolutos dos coeficientes deste modelo determinam a significância de cada variável regressora. É eliminada aquela variável associada ao coeficiente de menor valor absoluto. O processo é repetido até restar apenas uma variável regressora. A ordem na qual as variáveis regressoras são eliminadas sugere uma ordem de significância destas

variáveis para predição da variável resposta.

O procedimento é resumido a seguir:

1. Inicializamos um conjunto V contendo todas as variáveis regressoras.
2. Consideramos um conjunto finito $A \subseteq \mathbb{R}$.
3. Enquanto $|V| \geq 1$:
   (a) Para cada $\alpha \in A$ e cada $k = 1, \ldots, 10$, treinamos um modelo (Ridge ou Lasso) com o conjunto de treino do k-ésimo fold.
   (b) Escolhemos $\alpha \in A$ e o respectivo modelo que leva a menor média de RMSE na validação cruzada.
   (c) Eliminamos a variável com coeficiente de menor valor absoluto.

Na Figura 18 apresentamos o comportamento do RMSE durante o procedimento de *backward-stepwise feature selection* utilizando o Ridge. Para cada iteração, uma vez determinado α^*, é treinado o modelo utilizando o conjunto de treino e tendo como hiperparâmetro α^*. Então é avaliado o RMSE deste modelo no conjunto de treino. Na Figura 18, cada ponto denota uma iteração deste procedimento na qual $|V_k| = x_k$ e o RMSE do modelo é y_k.

Similarmente, na Figura 19 apresentamos o comportamento do RMSE durante o procedimento de *backward-stepwise feature selection* utilizando o Lasso.

Ponderando as ordens de significância das variáveis regressoras sugeridas pelos procedimentos com o Ridge e com o Lasso, chegamos à ordem de significância apresentada na Figura 20.

Com base nas Figuras 18, 19 e 20, optamos por considerar no treinamento dos modelos apenas as doze primeiras variáveis segundo a ordem apresentada na Figura 20.

4. Modelos

Nesta seção apresentamos a formulação matemática de cada um dos modelos de regressão considerados neste trabalho. Para isso, no que segue, vamos considerar um conjunto finito $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R} : i = 1, \ldots, N\}$. Intuitivamente, este é o conjunto de treino para cada um dos modelos.

4.1. Regressão Linear

No caso da regressão linear, assumimos que a relação entre a variável resposta y e as variáveis regressoras $\mathbf{x}$ é da forma:

$$y(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}\mathbf{x} + \epsilon = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_n x_{in} + \epsilon$$

para algum $\boldsymbol{\beta} = (\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{n+1}$ e onde ϵ é uma variável aleatória [12, 10, 13].

Assumindo que $\mathbb{E}[\epsilon] = 0$ e $\text{Var}[\epsilon] = \sigma^2$, temos que

$$\mathbb{E}[y|\mathbf{x}] = \beta_0 + \boldsymbol{\beta}\mathbf{x}$$

e

$$\text{Var}[y|\mathbf{x}] = \sigma^2$$

Portanto, se $\boldsymbol{\beta}$ fosse conhecido, o estimador $\hat{y}(\mathbf{x}; \boldsymbol{\beta}) = \beta_0 + \boldsymbol{\beta}\mathbf{x}$ é o estimador que, dado $\mathbf{x} \in \mathbb{R}^n$, minimiza o erro quadrático esperado [14]

$$\mathbb{E}\big[\{y - \hat{f}(\mathbf{x})\}^2 | \mathbf{x}\big]$$

Levando isso em conta, o objetivo da regressão linear é estimar o valor dos parâmetros $\boldsymbol{\beta}$ com um estimador $\hat{\boldsymbol{\beta}}$. Assim, para dado $\hat{\boldsymbol{\beta}}$, obtém se um estimador $\hat{y}(\mathbf{x}; \hat{\boldsymbol{\beta}}) = \hat{\beta}_0 + \hat{\boldsymbol{\beta}}\mathbf{x}$ para $y(\mathbf{x})$.

Dado um conjunto $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R} : i = 1, \ldots, N\}$, vamos assumir que, para todo $i = 1, \ldots, N$:

$$y_i = \beta_0 + \boldsymbol{\beta}\mathbf{x}_i + \epsilon_i$$

para algum $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$ e onde ϵ_i é uma variável aleatória.

Agora, dado $\boldsymbol{\beta}' \in \mathbb{R}^{n+1}$, define-se os resíduos

$$\varepsilon_i(\boldsymbol{\beta}') = y_i - (\beta_0' + \hat{\boldsymbol{\beta}}'\mathbf{x}_i) = y_i - (\beta_0' + \beta_1' x_{i1} + \cdots + \beta_n' x_{in})$$

para cada $i = 1, \ldots, N$.

No caso da *regressão linear por mínimos quadrados*, que é mais comumente conhecida apenas por *regressão linear*, dado o conjunto $\mathcal{D}$, o estimador $\hat{\boldsymbol{\beta}}$ para $\boldsymbol{\beta}$ é definido como a solução do seguinte problema de otimização [12, 10, 13]

$$\begin{array}{ll} \min & \sum_{i=1}^{N} \left\{y_i - (\beta_0' + \boldsymbol{\beta}'\mathbf{x}_i)\right\}^2 \\ \text{s.a} & \boldsymbol{\beta}' \in \mathbb{R}^{n+1} \end{array}$$

E assim o estimador $\hat{y}(\hat{\boldsymbol{\beta}})$ obtido pela regressão linear é dado por

$$\hat{y}(x; \hat{\boldsymbol{\beta}}) = \hat{\beta}_0 + \hat{\boldsymbol{\beta}}\mathbf{x}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_n x_{in}$$

Esta explicação busca apresentar a motivação e formulação da regressão linear como um problema de otimização. Contudo, a formalização da teoria de regressão linear requer que se assuma algumas hipóteses [12], como:

- Independência dos erros: assume-se que os diferentes erros ϵ_i não são correlacionados.
- Homocedasticidade dos erros: assume que $\text{Var}[\epsilon_i] = \text{Var}[\epsilon_i], \forall i, j = 1, \ldots, N$.

Uma discussão mais detalhada das hipóteses dos modelos de regressão linear é apresentada em [12] e especialmente em [15].

4.2. Ridge

O *Ridge* pode ser entendido como uma variação da regressão linear na qual é incluído um fator de *regularização*. A motivação do Ridge é similar a da regressão linear, mas o Ridge difere da regressão linear ao considerar o estimador $\hat{\boldsymbol{\beta}}$ que é solução do seguinte problema de otimização

$$\begin{array}{ll} \min & \sum_{i=1}^{N} \left\{ y_i - (\beta_0' + \boldsymbol{\beta}' \mathbf{x}_i) \right\}^2 + \lambda \sum_{i=1}^{N} (\beta_i')^2 \\ \text{s.a} & \boldsymbol{\beta}' \in \mathbb{R}^{n+1} \end{array}$$

para um dado $\lambda \geq 0$.

Note que o Ridge inclui o termo $\lambda \sum_{i=1}^{N} (\beta_i')^2$ na função objetivo do problema de minimização. Este termo é a norma ℓ^2 do vetor $\boldsymbol{\beta}$. A motivação deste termo é a penalização na função objetivo do valor absoluto dos coeficientes do modelo.

Observe que um conjunto de dados $\mathcal{D}'$ a ser coletado é também uma variável aleatória. Suponha que $\mathcal{D} \sim \mathbf{D}$. Assim, um estimador $\hat{\boldsymbol{\beta}}(\mathcal{D}')$ treinado com o conjunto $\mathcal{D}'$ também é uma variável aleatória, e é função de $\mathcal{D}'$, e podemos nos referir a $\mathbb{E}_{\mathbf{D}}[\hat{\boldsymbol{\beta}}(\mathcal{D}')]$, i.e o valor esperado de $\hat{\boldsymbol{\beta}}(\mathcal{D}')$ segundo a distribuição $\mathbf{D}$ e $\text{Var}_{\mathbf{D}}[\hat{\boldsymbol{\beta}}(\mathcal{D}')]$, i.e a variância de $\hat{\boldsymbol{\beta}}(\mathcal{D}')$ segundo a distribuição $\mathbf{D}$. Efetivamente o propósito da penalização imposta pelo Ridge é reduzir $\text{Var}_{\mathbf{D}}[\hat{\boldsymbol{\beta}}(\mathcal{D}')]$, i.e reduzir a variância dos parâmetros estimados dados diferentes conjuntos de treinamento e consequentemente reduzir a variância do estimador $\hat{y}(\hat{\boldsymbol{\beta}})$. Na prática, isso busca reduzir as chances de *overfitting*.

4.3. Lasso

O Lasso, assim como o Ridge, é um método no qual é incluído um fator de regularização. A única diferença entre os dois é que o Lasso incluí um termo de penalização na função objetivo que leva em conta a norma ℓ^1, em vez da norma ℓ^2. Assim, o Lasso considera o estimador $\hat{\boldsymbol{\beta}}$ que é solução do seguinte problema de otimização

$$\begin{array}{ll} \min & \sum_{i=1}^{N} \left\{ y_i - (\beta_0' + \boldsymbol{\beta}' \mathbf{x}_i) \right\}^2 + \lambda \sum_{i=1}^{N} |\beta_i'| \\ \text{s.a} & \boldsymbol{\beta}' \in \mathbb{R}^{n+1} \end{array}$$

para um dado $\lambda \geq 0$.

4.4. Árvore de Decisão

Assim como apresentado em [16], os modelos de *árvore de decisão* para regressão diferem dos modelos de regressão linear uma vez que assumem que a relação entre a variável resposta y e as variáveis regressoras $\mathbf{x}$ é da forma:

$$y(\mathbf{x}) = \sum_{j=1}^{M} c_j \cdot \mathbb{1}[\mathbf{x} \in R_j]$$

onde $\{R_1, \ldots, R_M\}$ é uma partição do domínio de $\mathbf{x}$ tal que $R_i \cap R_j = \emptyset, \forall i, j = 1, \ldots, M$.

Por outro lado, os modelos de regressão linear assumem que a relação entre a variável resposta y e as variáveis regressoras $\mathbf{x}$ é da forma:

$$y(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}\mathbf{x}$$

para algum $\boldsymbol{\beta} = (\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{n+1}$.

Portanto os modelos de árvore de decisão para regressão assumem que $y(\mathbf{x})$ assume um valor constante c_j em cada intervalo $R_j \subseteq \text{Dom}(\mathbf{x})$.

Sendo assim, o procedimento de construção de uma árvore de decisão pode ser resumido em duas etapas [16]:

1. Particionamento de $\text{Dom}(\mathbf{x})$ em $\mathcal{R} = \{R_1, \ldots, R_M\}$ tal que $R_i \cap R_j = \emptyset, \forall i, j = 1, \ldots, M$.
2. Dado o particionamento $\mathcal{R}$, o estimador da árvore de decisão é dado por $\hat{y}(\mathbf{x}; \mathcal{R}) = \sum_{j=1}^{M} \mu_j \cdot \mathbb{1}[\mathbf{x} \in R_j]$, onde $\mu_j = \frac{\sum_{i=1}^{N} y_i \cdot \mathbb{1}[\mathbf{x} \in R_j]}{\sum_{i=1}^{N} \mathbb{1}[\mathbf{x} \in R_j]}$, ou seja, μ_j é a média dos y_i tais que $\mathbf{x}_i \in R_j$, para $i = 1, \ldots, N$.

Como na prática é inviável considerar todas as possíveis partições de Dom, opta-se por um tipo de partição conhecido como *recursive binary splitting*.

A árvore de decisão é representada por uma árvore binária. Cada vértice v da árvore representa um particionamento $\mathcal{R}_v$. Além disso, cada vértice ou da árvore ou é uma folha ou tem exatamente dois filhos v_l, v_r. Os dois filhos v_l, v_r de um vértice v representam dois novos particionamentos $\mathcal{R}_{v_l}, \mathcal{R}_{v_r}$ obtidos a partir de $\mathcal{R}_v$ por meio de um split. Mais especificamente, os particionamentos $\mathcal{R}_{v_l}, \mathcal{R}_{v_r}$ dos filhos v_l, v_r de um vértice v diferem de $\mathcal{R}_v$ por um split cada, sendo $\{\mathbf{x}|x_i < s\}$ no caso de $\mathcal{R}_{v_l}$ e $\{\mathbf{x}|x_i \geq s\}$ no caso de $\mathcal{R}_{v_r}$.

Durante a construção da árvore de decisão, o procedimento de split de um dado vértice v ocorre da seguinte maneira: é selecionada uma variável regressora x_i, para algum $i = 1, \ldots, n$, e selecionado um valor $s \in \mathbb{R}$ para realizar o split $\{x_i|x_i < s\}$ e $\{x_i|x_i \geq s\}$. Feito isso, são incluídos dois novos vértices v_l, v_r na árvore como filhos de v. O procedimento de split é então realizado para v_l, v_r. Portanto a construção da árvore de decisão segue um procedimento recursivo. Sendo assim, para garantir a convergência, é necessário definir um critério de parada. Uma opção comum é escolher um número $k \in \mathbb{N}$ tal que um vértice v não passará por um split se $\sum_{i=1}^{N} \mathbb{1}[\mathbf{x} \in R_j] \leq k, \forall R_j \in \mathcal{R}_v$, ou seja, se o número de registros em todos os conjuntos $R_j \in \mathcal{R}_v$ não excede k.

Dado $i \in \{1, \ldots, n\}$ e $s \in \mathbb{R}$, seja $R_1(i, s) = \{\mathbf{x}|x_i < s\}$ e $R_2(i, s) = \{\mathbf{x}|x_i \geq s\}$.

Além disso, para $R \subseteq \mathbb{R}^n$, seja

$$\mu_R = \frac{\sum_{i=1}^{N} y_i \cdot \mathbb{1}[\mathbf{x} \in R]}{\sum_{i=1}^{N} \mathbb{1}[\mathbf{x} \in R]}$$

ou seja, μ_R é a média dos y_i tais que $\mathbf{x}_i \in R$, para $i = 1, \ldots, N$.

Para cada split, é selecionado o índice $i^* \in \{1, \ldots, n\}$ e o valor $s^* \in \mathbb{R}$ que minimizam

$$\sum_{i=1}^{N} \mathbb{1}[\mathbf{x}_i \in R_1(i,s)] \cdot (y_i - \mu_{R_1})^2 + \sum_{i=1}^{N} \mathbb{1}[\mathbf{x}_i \in R_2(i,s)] \cdot (y_i - \mu_{R_2})^2$$

Na prática, contudo, a construção do modelo pode abranger outros critérios e outras etapas. Um exemplo é o procedimento de poda da árvore descrito em [16]. Outro exemplo, no caso de árvores de decisão para classificação, é que a seleção de (i^*, s^*) para o split é feita com base na minimização do índice de impureza de Gini [10, 16].

4.5. Métodos Ensemble

Um *método ensemble* é um modelo que combina vários outros modelos de maneira a obter um único estimador. Os modelos que compõe um método ensemble são chamados de *weak models*, pois cada um destes modelos individualmente pode não ser um bom estimador. Por outro lado, espera-se que o método ensemble, que combina vários *weak models*, seja um bom estimador.

Em geral, dado um conjunto de estimadores $\hat{F} = \{\hat{f}_1, \ldots, \hat{f}_K\}$ como *weak models*, podemos definir um estimador ensemble $\hat{f}$ como uma combinação linear dos weak models, ou seja, para dado $\mathbf{x}$:

$$\hat{f}(\mathbf{x}; \boldsymbol{\alpha}) = \sum_{i=1}^{K} \alpha_i \hat{f}_i(\mathbf{x})$$

Suponha que é dado um conjunto $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R} : i = 1, \ldots, N\}$.

No caso do método *bagging* [16], $B \in \mathbb{N}$ conjuntos de treinamento diferentes são produzidos por meio de *bootstrapping* do conjunto original $\mathcal{D}$. Para cada $b = 1, \ldots, B$ é treinado um weak model $\hat{f}_b$. Finalmente, o estimador *bagging* é definido como

$$\hat{f}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(\mathbf{x})$$

No caso particular em que os weak models do *bagging* são árvores de decisão, temos o chamado *bagged forest* [16].

Florestas aleatórias e gradient boosting são dois métodos ensemble.

Florestas aleatórias tem como weak models especificamente árvores de decisão, por isso o nome *floresta*. O treinamento de uma floresta aleatória consiste do treinamento de M árvores de decisão. Assim como no *bagging*, é usado *bootstrapping* para a produção de M conjuntos de treinamento a partir do conjunto original $\mathcal{D}$. Contudo, no treinamento das árvores de decisão, na determinação de cada split, apenas um subconjunto de $p < n$ variáveis regressoras é considerado para o split. As p variáveis regressoras consideradas para cada split são selecionadas aleatóriamente. Efetivamente, o propósito desta ideia

é reduzir a correlação entre as diferentes árvores que compõe a floresta aleatória. Uma explicação mais detalhada é apresentada em [16].

Boosting é um procedimento similar ao *bagging*. Assim como no bagging, o conjunto de dados original é utilizado para produzir novos M conjuntos, e para cada um destes conjuntos é treinado um weak model para compor o modelo ensemble. No caso do bagging, cada um destes M conjuntos é produzido por meio de *bootstrapping* de maneira independente dos demais. No caso do *boosting*, os conjuntos (e consequentemente os *weak models*) são construídos de maneira sequencial, com a construção do k-ésimo *weak model* levando em conta as informações dos i-ésimos *weak models*, para $i = 1, \ldots, k-1$. Sendo assim, para o *boosting* não é utilizado *bootstrapping*.

Na k-ésima iteração do *boosting*, já foram treinados $k-1$ *weak models* $\hat{f}_1, \ldots, \hat{f}_{k-1}$ e já foram determinados $k-1$ parâmetros $\gamma_1, \ldots, \gamma_{k-1}$ associados a cada *weak model*. Estes $k-1$ *weak models* compõe um estimador $\hat{F}_{k-1}(\mathbf{x}) = \sum_{b=1}^{k-1} \gamma_b \hat{f}_b(\mathbf{x})$.

O *boosting* inicialmente treina um *weak model* $\hat{f}_1$ tendo como variável resposta a variável original y.

A partir de então, na k-ésima iteração, para $k > 1$, é treinado um novo *weak model* tendo como variável resposta o resíduo $y_i - \hat{F}_{k-1}(\mathbf{x})$. Com o treinamento é obtido um novo *weak model* $\hat{f}_k$ e determinado um novo valor γ_k. E com isso é obtido um novo estimador $\hat{F}_k(\mathbf{x}) = \sum_{b=1}^{k} \gamma_b \hat{f}_b(\mathbf{x})$.

Este procedimento é repetido até que o critério de parada seja atingido.

Uma explicação mais detalhada do *boosting* é apresentada em [16].

No caso em que os *weak models* são árvores de decisão, chamamos o método de *tree boosting*.

No caso do *boosting*, dada uma função de perda $\mathcal{L}$, na k-ésima iteração o valor de γ_k é determinado por

$$\gamma_k = \arg\min_\gamma \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{F}^{k-1}(\mathbf{x}_i) + \gamma \hat{f}_k(\mathbf{x}_i))$$

Sendo assim, no caso do *boosting*, dada uma função de perda $\mathcal{L}$, na k-ésima iteração é obtido um novo estimador $\hat{F}_k$ da seguinte maneira:

$$\hat{F}_k(\mathbf{x}) = \hat{F}_k(\mathbf{x}) + \gamma_k \sum_{j=1}^{M_k} \mu_{kj} \mathbb{1}[\mathbf{x} \in R_{kj}]$$

onde

$$\gamma_k = \arg\min_\gamma \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{F}^{k-1}(\mathbf{x}_i) + \gamma \hat{f}_k(\mathbf{x}_i))$$

Já no caso do *gradient boosting*, dada uma função de perda $\mathcal{L}$, na k-ésima iteração é determinado um valor γ_{kj} para cada $R_j \in \mathcal{R}$, onde $\mathcal{R}$ é o particionamento dado pela

árvore de decisão treinada na k-ésima iteração.

Sendo assim, no caso do *gradient boosting*, dada uma função de perda $\mathcal{L}$, na k-ésima iteração é obtido um novo estimador $\hat{F}_k$ da seguinte maneira:

$$\hat{F}_k(\mathbf{x}) = \hat{F}_k(\mathbf{x}) + \sum_{j=1}^{M} \gamma_{kj} \cdot \mu_{kj} \cdot \mathbb{1}[\mathbf{x} \in R_{kj}] = \hat{F}_k(\mathbf{x}) + \sum_{j=1}^{M} \gamma'_{kj} \cdot \mathbb{1}[\mathbf{x} \in R_{kj}]$$

onde

$$\gamma'_{kj} = \arg\min_{\gamma} \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{F}^{k-1}(\mathbf{x}_i) + \gamma \hat{f}_k(\mathbf{x}_i)) \cdot \mathbb{1}[\mathbf{x}_i \in R_{kj}]$$

4.6. Máquinas de Vetor Suporte

Nesta seção nos dedicamos a apresntar as *support vector machines*, ou máquinas de vetor suporte, exclusivamente para o caso de regressão.

Suponha que é dado o conjunto $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R} : i = 1, \ldots, N\}$.

O objetivo de regressão ε-VR [17] é encontrar um estimador $\hat{f}$ tal que $|y_i - \hat{f}(\mathbf{x}_i)| < \varepsilon$ para todo $i = 1, \ldots, N$, ou seja, o objetivo é encontrar um estimador que não tenha um erro maior que ε para qualquer ponto no conjunto de treino.

Suponha, em um primeiro momento, que buscamos um estimador $\hat{f}$ do tipo

$$\hat{f}(\mathbf{x}; \mathbf{w}) = \mathbf{w} \cdot \mathbf{x} + w_0$$

De maneira similar ao que foi explicado na Seção 4.2 para o Ridge, $\mathcal{D}$ é uma variável aleatória e consequentemente um estimador treinado com base em $\mathcal{D}$ também é. No caso do estimador linear acima, uma maneira de reduzir a variância de $\hat{\mathbf{w}}$ e, consequentemente, de $\hat{f}$, é controlar o crescimento dos valores absolutos de w_j, para $j = 1, \ldots, n$.

Assim, o modelo ε-VR linear é dado pela solução do seguinte problema de otimização:

$$\begin{array}{ll} \min & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.a} & y_i - (\mathbf{w} \cdot \mathbf{x}_i + w_0) < \varepsilon \quad i = 1, \ldots, N \\ & (\mathbf{w} \cdot \mathbf{x}_i + w_0) - y_i < \varepsilon \quad i = 1, \ldots, N \\ & \mathbf{w} \in \mathbb{R}^n \end{array}$$

Note que este é um problema convexo. Contudo, por conta da restrição $|y_i - (\mathbf{w} \cdot \mathbf{x}_i + w_0)| < \varepsilon, \forall i = 1, \ldots, N$, o problema não necessariamente é viável para todo $\mathcal{D}$. Esta versão do problema é análoga à *hard margin* no caso de SVM para classificação. A fim de garantir a viabilidade do problema para qualquer $\mathcal{D}$, podemos passar a permitir erros e então incluir na função objetivo um termo que leve em conta a minimização dos erros. Introduzindo variáveis de folga ξ_i, ξ'_i para cada $i = 1, \ldots, N$, isso nos leva ao seguinte problema de otimização:

$$
\begin{array}{lll}
\text{min} & \frac{1}{2}||\mathbf{w}||^2 + \lambda \sum_{i=1}^{N}(\xi_i + \xi_i') & \\
\text{s.a} & y_i - (\mathbf{w}\cdot\mathbf{x}_i + w_0) < \varepsilon + \xi_i & i = 1,\ldots,N \\
& (\mathbf{w}\cdot\mathbf{x}_i + w_0) - y_i < \varepsilon + \xi_i' & i = 1,\ldots,N \\
& \xi_i, \xi_i' \geq 0 & i = 1,\ldots,N \\
& \mathbf{w} \in \mathbb{R}^n &
\end{array}
$$

para um dado $\lambda \geq 0$.

Esta versão do problema é análoga à *soft margin* no caso de SVM para classificação.

Como é de prática comum em otimização, podemos considerar o *dual lagrangeano* deste problema. Isso é feito introduzindo multiplicadores de lagrange da seguinte maneira:

- Para cada $i = 1,\ldots,N$, dualizamos a restrição $y_i - (\mathbf{w}\cdot\mathbf{x}_i + w_0) < \varepsilon + \xi_i$ e associamos o multiplicador de lagrange α_i.
- Para cada $i = 1,\ldots,N$, dualizamos a restrição $(\mathbf{w}\cdot\mathbf{x}_i + w_0) - y_i < \varepsilon + \xi_i'$ e associamos o multiplicador de lagrange α_i'.
- Para cada $i = 1,\ldots,N$, dualizamos a restrição $\xi_i \geq 0$ e associamos o multiplicador de lagrange η_i.
- Para cada $i = 1,\ldots,N$, dualizamos a restrição $\xi_i' \geq 0$ e associamos o multiplicador de lagrange η_i'.

Após o devido desenvolvimento [17], conclui-se que o problema dual lagrangeano é dado por:

$$
\begin{array}{lll}
\text{max} & \dfrac{-1}{2}\displaystyle\sum_{i=1}^{N}\sum_{j>i}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')\mathbf{x}_i\cdot\mathbf{x}_j & \\
& -\varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i') + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i') & \\
\text{s.a} & \sum_{i=1}^{N}(\alpha_i - \alpha_i') = 0 & \\
& 0 \leq \alpha_i \leq \lambda & i = 1,\ldots,N \\
& 0 \leq \alpha_i' \leq \lambda & i = 1,\ldots,N
\end{array}
$$

A solução do problema dual pode então ser mapeada para uma solução do problema dual da seguinte maneira:

$$\hat{w}_i = (\alpha_i - \alpha_i')\mathbf{x}_i$$

e portanto

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N}(\alpha_i - \alpha_i')\mathbf{x}_i\cdot\mathbf{x} + w_0$$

O produto interno $\mathbf{x}_i\cdot\mathbf{x}$ é chamado de *kernel*.

Uma maneira de considerar estimadores $\hat{f}$ não-lineares seria de fazer um pré-processamento de $\mathcal{D}$ mapeando $\text{Dom}(\mathbf{x})$ em $\Phi(\text{Dom}(\mathbf{x}))$ por meio de uma dada função Φ.

Porém, note que o problema dual depende apenas de $\mathbf{x}_i \cdot \mathbf{x}_j$, para $i, j = 1, \ldots, N$. Sendo assim, o problema dual após aplicação de Φ a $\mathcal{D}$ se torna

$$\begin{aligned}
\text{max} \quad & \frac{-1}{2}\sum_{i=1}^{N}\sum_{j>i}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j) \\
& -\varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i') + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i') \\
\text{s.a} \quad & \sum_{i=1}^{N}(\alpha_i - \alpha_i') = 0 \\
& 0 \leq \alpha_i \leq \lambda \qquad\qquad i = 1, \ldots, N \\
& 0 \leq \alpha_i' \leq \lambda \qquad\qquad i = 1, \ldots, N
\end{aligned}$$

Portanto, para dado Φ, podemos expressar $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$ por um kernel K_Φ tal que $K_\Phi(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$.

E portanto a dependência de $\hat{f}$ em Φ pode ser expressa de maneira implícita pelo kernel K_Φ, e o problema de otimização se torna:

$$\begin{aligned}
\text{max} \quad & \frac{-1}{2}\sum_{i=1}^{N}\sum_{j>i}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K_\Phi\big(\mathbf{x}_i, \mathbf{x}_j\big) \\
& -\varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i') + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i') \\
\text{s.a} \quad & \sum_{i=1}^{N}(\alpha_i - \alpha_i') = 0 \\
& 0 \leq \alpha_i \leq \lambda \qquad\qquad i = 1, \ldots, N \\
& 0 \leq \alpha_i' \leq \lambda \qquad\qquad i = 1, \ldots, N
\end{aligned}$$

Assim, diferentes estimadores não-lineares podem ser obtidos por meio da especificação de uma função Φ, o seu respectivo *kernel* K_Φ e pela resolução do problema otimização acima. Isso leva aos seguintes estimadores não-lineares:

$$\hat{w}_i = (\alpha_i - \alpha_i')\Phi(\mathbf{x}_i)$$

e portanto [17]

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N}(\alpha_i - \alpha_i')K_\Phi(\mathbf{x}_i, \mathbf{x}) + w_0$$

Neste trabalho consideramos o kernel linear $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}\cdot\mathbf{y}$, o kernel polinomial de grau $d \in \mathbb{N}$ $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d$ e o kernel radial basis function (RBF) $K(\mathbf{x}, \mathbf{y}) = \exp\big\{-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}\big\}$ para dado $\sigma^2 > 0$.

4.7. Redes Neurais

Uma *rede neural feed-forward* é composta por *camadas*, com cada camada sendo formada por um certo número de *perceptrons*. A estrutura da rede neural pode ser representada por uma árvore direcionada e sem ciclos.

Dada uma função de ativação σ e um vetor de parâmetros $\omega \in \mathbb{R}^{n+1}$, um *perceptron* tem como entrada um vetor $z \in \mathbb{R}^n$ e tem como saída o valor $\sigma(\omega \cdot z + \omega_0)$.

A rede neural *feedforward* sucessivamente compõe a aplicação de *perceptrons*. Os resultados da k-ésima camada constituem um vetor z^k que é passado como vetor de entrada para os perceptrons da $k+1$-ésima camada.

Vamos agora formalizar a teoria matemática da rede neural [10, 16, 13]. Vamos considerar o caso de *redes neurais feedforward*. Um input da rede neural é um vetor $\mathbf{x} \in \mathbb{R}^n$. Defina o vetor $z^0 \in \mathbb{R}^n$ tal que $z_i^0 = x_i$ para $i = 1, \ldots, n$. Sem perda de generalidade, não faremos distinção entre as *camadas escondidas* e as *camadas externas*. A rede neural tem L camadas e, para cada $1 \leq k \leq L$, a k-ésima camada tem m_k *perceptrons*. A k-ésima camada tem como resultado o vetor $z^k \in \mathbb{R}^{m_k}$, que é passado como input para a $k+1$-ésima camada. Por simplicidade, vamos omitir os termos de bias ω_0. Sem perda de generalidade, pode-se supor que os termos de bias são a primeira componente do vetor de parâmetros ω, e que o a primeira coordenada de cada input tem valor 1, que multiplica o bias ω_0.

O p-ésimo perceptron da k-ésima camada é da forma

$$\sigma_p^k(\omega_p^k z^{k-1})$$

onde $\omega_p^k \in \mathbb{R}^{m_{k-1}}$ e σ_p^k é uma função derivável com relação a variável ω_p^k e é chamada de *função de ativação*.

O output da rede neural feedforward é o output da L-ésima camada. Portanto, o output da rede neural é o vetor

$$z^L = \left[\sigma_1^L(\omega_1^L z^{L-1}), \sigma_2^L(\omega_2^L z^{L-1}), \ldots, \sigma_{m_L}^L(\omega_{m_L}^L z^{L-1})\right]^T$$

Seja $\mathbf{w} = \{\omega_p^k \mid 1 \leq k \leq L, 1 \leq p \leq m_k\}$. Então, dado o conjunto $\mathcal{D}$ e dado $\mathbf{w}$, atribui-se um erro à rede neural por meio de uma *função de perda* $L(\mathbf{w}, \mathcal{D})$.

Suponha que

$$L(\mathbf{w}) = \sum_{i=1}^{N} \sum_{j=1}^{m_L} \ell_j\left(\sigma_j^L(\omega_j^{(L)} z^{(i,L-1)})\right)$$

onde ℓ_j é derivável para todo $1 \leq j \leq m_L$.

Note que para $\mathcal{D}$ fixo, $L(\mathbf{w}, \mathcal{D})$ é função de $\mathbf{w}$.

Assim, para treinar a rede neural com base no conjunto $\mathcal{D}$, minimizamos a função L em função de $\mathbf{w}$. Uma maneira comumente escolhida para fazer isso é por meio de gradiente descendente de L em função de $\mathbf{w}$.

Seja $\omega_{p,i}^k$ a i-ésima coordenada do vetor ω_p^k. A cada iteração do gradiente descendente, cada k, p, i, fazemos um update no valor de $\omega_{p,i}^k$ da seguinte maneira:

$$\omega_{p,i}^k \leftarrow \omega_{p,i}^k - \lambda \frac{\partial L}{\partial \omega_{p,i}^k}$$

Aqui, λ é um parâmetro de aprendizado. Na prática, o valor de λ é alterado durante o procedimento de gradiente descendente e costuma ter seu valor reduzido com o número de iterações.

Portanto, para realizar o gradiente descendente, é necessário, a cada iteração, calcular $\frac{\partial L}{\partial \omega_{p,i}^k}$ para cada $1 \leq k \leq L, 1 \leq p \leq m_k, 1 \leq i \leq m_{k-1}$.

No caso em que σ_p^k é uma função derivável com relação a variável ω_p^k para todo $1 \leq k \leq L, 1 \leq p \leq m_k$, e que ℓ_j é derivável para todo $1 \leq j \leq m_L$, então o gradiente de L em função de $\mathbf{w}$ pode ser calculado de maneira exata por meio do procedimento conhecido como *backpropagation*.

Seja $1 \leq k \leq L, 1 \leq p \leq m_k, 1 \leq i \leq m_{k-1}$. A ideia do *backpropagation* é aplicar sucessivamente a regra da cadeia, começando da L-ésima camada, para calcular $\frac{\partial L}{\partial \omega_{p,i}^k}$.

Note que, pela regra da cadeia:

$$\frac{\partial \ell_j}{\partial \omega_{p,i}^k} = \frac{\partial \ell_j}{\partial \sigma_p^k} \frac{\partial \sigma_p^k}{\partial \omega_{p,i}^k}$$

Daí

$$\frac{\partial \mathcal{L}}{\partial \omega_{p,i}^k} = \frac{\partial}{\partial \omega_{p,i}^k} \left(\sum_{i=1}^{N} \sum_{j=1}^{m_L} \ell_j \right) = \sum_{i=1}^{N} \sum_{j=1}^{m_L} \frac{\partial \ell_j}{\partial \omega_{p,i}^k} = \sum_{i=1}^{N} \sum_{j=1}^{m_L} \frac{\partial \ell_j}{\partial \sigma_p^k} \frac{\partial \sigma_p^k}{\partial \omega_{p,i}^k} \tag{1}$$

Agora, como $\sigma_p^k = \sigma_p^k(\omega_p^k z^{k-1})$, então, pela regra da cadeia:

$$\begin{aligned} \frac{\partial \sigma_p^k}{\partial \omega_{p,i}^k} &= \frac{\partial}{\partial \omega_{p,i}^k} \left(\sigma_p^k(\omega_p^k z^{k-1}) \right) \\ &= \frac{\partial}{\partial \omega_{p,i}^k} \left(\sigma_p^k(\omega_{p,i}^k z_i^{k-1} + \sum_{j \neq i} \omega_{p,j}^k z_j^{k-1}) \right) \\ &= \sigma_p^{k\prime}(\omega_p^k z^{k-1}) z_i^{k-1} \\ &= \sigma_p^{k\prime} z_i^{k-1} \end{aligned} \tag{2}$$

Tomando $k = L$ e $p = j$, temos

$$\frac{\partial \ell_j}{\partial \sigma_j^L} = \frac{\partial}{\partial \sigma_j^L}\left(\ell_j\big(\sigma_j^L\big)\right) = \ell_j'\big(\sigma_j^L\big)$$

e, para $1 \leq k \leq L-1$, temos, pela regra da cadeia,

$$\frac{\partial \ell_j}{\partial \sigma_p^{k-1}} = \sum_{q=1}^{m_k} \frac{\partial \ell_j}{\partial \sigma_q^k}\frac{\partial \sigma_q^k}{\partial \sigma_p^{k-1}} = \sum_{q=1}^{m_k} \frac{\partial \ell_j}{\partial \sigma_q^k}\omega_{q,p}^k \tag{3}$$

onde na última igualdade usamos que:

$$\sigma_q^k = \sigma_q^k\big(\omega_q^k z^{k-1}\big) = \sigma_q^k\big(\omega_{q,p}^k z_p^{k-1} + \sum_{j\neq p}\omega_{q,j}^k z_j^{k-1}\big)$$

e que

$$z_p^{k-1} = \sigma_p^{k-1}$$

e assim, portanto:

$$\frac{\partial \sigma_q^k}{\partial \sigma_p^{k-1}} = \omega_{q,p}^k$$

Com isso, obtemos uma relação entre $\frac{\partial \ell_j}{\partial \sigma_p^{k-1}}$ e $\frac{\partial \ell_j}{\partial \sigma_p^k}$. Assim, podemos calcular $\frac{\partial \ell_j}{\partial \sigma_p^k}$ para todo $1 \leq k \leq L$. Para isso, começamos calculando para a $k = L$, depois para $k = L-1$, depois $k = L-2$ e assim por diante.

Defina, para $1 \leq k \leq L$, $\delta_{j,p}^k = \frac{\partial \ell_j}{\partial \sigma_p^k}$. Então vale que

$$\delta_{j,j}^L = \ell_j'\big(\sigma_j^L\big)\ ,\ \delta_{j,p}^L = 0 \text{ se } j \neq p$$

Substituindo de volta em (3), temos que:

$$\delta_{j,p}^{k-1} = \sum_{q=1}^{m_k} \delta_{j,q}^k \omega_{q,p}^k$$

para $1 \leq k \leq L-1$.

Assim, usando (2) e substituindo de volta em (1), temos que:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \omega_{p,i}^k} &= \sum_{i=1}^{N}\sum_{j=1}^{m_L} \frac{\partial \ell_j}{\partial \sigma_p^k}\frac{\partial \sigma_p^k}{\partial \omega_{p,i}^k} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{m_L} \delta_{j,p}^k \sigma_p^{k\prime} z_i^{k-1}\end{aligned}$$

Assim, para cada $\mathbf{w}$, para calcular $\frac{\partial L(\mathbf{w})}{\partial \omega_{p,i}^k}$, basta conhecermos z_i^{k-1} e $\delta_{j,p}^k$. Os valores z^{k-1} são calculados em sentido feedforward pela rede neural. Uma vez conhecido z^L, que é o output da rede neural, podemos calcular ℓ_j e, com isso, δ^L. Daí, podemos, de maneira recursiva, calcular δ^{L-1}. Calculados δ^{L-1}, podemos então calcular δ^{L-2}, e assim por diante. Ou seja, os δ^k são propagados pela rede neural no sentido backward, dando origem ao nome *backpropagation*. O *backpropagation* proporciona uma maneira eficiente de calcular o gradiente de L em função $\mathbf{w}$. Assim, o *backpropagation* é uma das maneiras de calcular gradientes e realizar gradiente descendente para o treinamento da rede neural feedforward. Contudo, os métodos do gradiente descendente e *backpropagation* não são as únicas opções para o treinamento de redes neurais. Outros métodos podem ser consultados em [16, 10, 13].

5. Resultados

5.1. Conjunto de Treino

Para o conjunto de treino, treinamos modelos de regressão linear, Ridge, Lasso, árvore de decisão, floresta aleatória, gradient boosting, support vector machines e redes neurais *feedforward*. Pra cada arquitetura de modelo, executamos 10 ciclos de validação cruzada com 10 fold. Para cada ciclo, treinamos um modelo utilizando o conjunto de treino associado àquele ciclo e avaliamos o modelo no conjunto de teste associado àquele ciclo.

No caso do Ridge e do Lasso, primeiro é utilizada validação cruzada para determinar o valor para o hiperparâmetro λ que regula a penalização.

No caso da árvore de decisão, da floresta aleatória e do gradient boosting, consideramos duas opções do hiperparâmetro que determina a *profundidade* da árvore: uma com profundidade igual a 5; outra com profundidade igual a 10. Além disso, para o treinamento destes modelos é considerado como função de perda o erro quadrático.

No caso da rede neural, consideramos três opções de arquitetura: uma com 10 camadas; outra com 20 camadas; outra com 30 camadas. Para todas estas três opções a função de ativação escolhida é a função *relu* [16]. Nos testes preeliminares esta opção de função de ativação levou aos melhores resultados e por isso esta escolha. Observamos que, dos modelos considerados neste trabalho, as redes neurais foram aqueles que exigiram maior tempo de treinamento. Para esta etapa de validação cruzada, por conta dos longos tempos de treinamento, o número máximo de iterações para o treinamento das redes neurais foi especificado como 150. Isso pode ter levado a um comprometimento da performance das redes neurais nesta etapa. Para a etapa seguinte, que leva em consideração o conjunto de teste e treina cada rede apenas uma vez, especificamos o número máximo de iterações como 500.

As métricas consideradas para a avaliação dos modelos são o R^2 e o *root mean squared error* (RMSE) [16, 10]. Note que, como um mesmo modelo é treinado uma vez para cada um dos dez ciclos de validação cruzada, para cada modelo temos dez valores de R^2 e RMSE disponíveis. Na Tabela 1 apresentamos os valores de R^2 obtidos para cada modelo. Na Tabela 1, cada linha corresponde a um modelo. Os modelos estão ordenados em ordem crescente da média dos R^2. A coluna *Modelo* denota o nome do modelo. A coluna *Min* denota o valor mínimo de R^2 para aquele modelo durante os dez ciclos de validação cruzada. A coluna *Média* denota o valor médio de R^2 para aquele modelo

durante os dez ciclos de validação cruzada. A coluna *Max* denota o valor máximo de R^2 para aquele modelo durante os dez ciclos de validação cruzada. A coluna *Std* denota o desvio padrão dos valores de R^2 para aquele modelo durante os dez ciclos de validação cruzada.

Similarmente, na Tabela 2 apresentamos os valores de RMSE obtidos para cada modelo. Na Tabela 2, cada linha corresponde a um modelo. Os modelos estão ordenados em ordem decrescente da média dos RMSE. A coluna *Modelo* denota o nome do modelo. A coluna *Min* denota o valor mínimo de RMSE para aquele modelo durante os dez ciclos de validação cruzada. A coluna *Média* denota o valor médio de RMSE para aquele modelo durante os dez ciclos de validação cruzada. A coluna *Max* denota o valor máximo de RMSE para aquele modelo durante os dez ciclos de validação cruzada. A coluna *Std* denota o desvio padrão dos valores de RMSE para aquele modelo durante os dez ciclos de validação cruzada.

5.2. Conjunto de Teste

Selecionamos os seguintes modelos para serem treinados usando todo o conjunto de treino: SVM com kernel *RBF*; rede neural com 20 camadas; gradient boosting com profundidade máxima igual a 5; floresta aleatória com profundidade máxima igual a 100. Após o treinamento, avaliamos o R^2 e o RMSE do modelo no conjunto de treinamento. Os resultados são apresentados na Tabela 3. Na Tabela 3, cada linha corresponde a um modelo. Os modelos estão ordenados em ordem crescente da média dos R^2. A coluna *Modelo* denota o nome do modelo. A coluna R^2 denota o valor de R^2 para aquele modelo no conjunto de treino. A coluna *RMSE* denota o valor de RMSE para aquele modelo no conjunto de treino.

Para mérito de comparação, também consideramos as previsões dadas pelo LDAPS.

Além da Tabela 3, apresentamos, para cada modelo, o gráfico de dispersão com os valores preditos pelo modelo e os valores da variável resposta.

Na Figura 21 apresentamos o gráfico de dispersão para o LDAPS.

Na Figura 22 apresentamos o gráfico de dispersão para o modelo SVM com kernel RBF.

Na Figura 23 apresentamos o gráfico de dispersão para a rede neural com 20 camadas.

Na Figura 24 apresentamos o gráfico de dispersão para o gradient boosting com profundidade máxima igual a 5.

Na Figura 25 apresentamos o gráfico de dispersão para a floresta aleatória com profundidade máxima igual a 100.

6. Conclusão

Levando em conta os resultados apresentados nas Tabelas 1, 2 e 3, podemos ver que o uso de diferentes modelos de regressão como pós-processamento dos dados preditos pelo modelo numérico LDAPS levaram a melhores estimativas da temperatura máxima no dia seguinte. De fato, para a validação cruzada, o modelo LDAPS apresentou média dos

R^2 igual a 0.64 e média dos RMSE igual a 1.86. Por outro lado, o modelo de gradient boosting, para os mesmos dez ciclos de validação cruzada, apresentou média dos R^2 igual a 0.87 e média dos RMSE igual a 1.13. Já no conjunto de teste, o modelo LDAPS apresentou R^2 igual a 0.67 e RMSE igual a 1.82, enquanto o modelo de floresta aleatória com profundidade máxima igual a 5 apresentou R^2 igual a 0.89 e RMSE igual a 1.04.

Observe que o conjunto de dados conta com 25 estações metereológicas que estão situadas próximas das outras, estando todas situadas na mesma cidade, Seul. Uma possível maneira de obter modelos com melhor performance é desenvolvendo modelos que, na estimativa para uma estação d_i, leve em conta os dados associados às outras estações $d_j, j \neq i$. Assim, uma vez que a posição geográfica de cada estação é conhecida, a previsão da temperatura máxima do dia seguinte levaria em conta não apenas os dados associados àquela estação, mas também os dados das outras estações e suas proximidades.

De qualquer forma, concluímos que, pelo menos para esta base de dados, os modelos apresentados neste relatório levaram a uma melhor performance nas estimativas quando comparados ao modelo LDAPS.

Referências

[1] K. E. Smoyer-Tomic, R. Kuhn, and A. Hudson, "Heat wave hazards: an overview of heat wave impacts in canada," *Natural hazards*, vol. 28, no. 2, pp. 465–486, 2003.

[2] J. E. Walsh, A. S. Phillips, D. H. Portis, and W. L. Chapman, "Extreme cold outbreaks in the united states and europe, 1948–99," *Journal of climate*, vol. 14, no. 12, pp. 2642–2658, 2001.

[3] D. Cho, C. Yoo, J. Im, and D.-H. Cha, "Comparative assessment of various machine learning-based bias correction methods for numerical weather prediction model forecasts of extreme air temperatures in urban areas," *Earth and Space Science*, vol. 7, no. 4, p. e2019EA000740, 2020.

[4] G. Zhang, "Machine learning for the bias correction of ldaps air temperature prediction model," in *2021 6th International Conference on Machine Learning Technologies*, pp. 1–6, 2021.

[5] C. Marzban, "Neural networks for postprocessing model output: Arps," *Monthly Weather Review*, vol. 131, no. 6, pp. 1103–1111, 2003.

[6] R. Isaksson, "Reduction of temperature forecast errors with deep neural networks," 2018.

[7] C. Yi, Y. Shin, and J.-W. Roh, "Development of an urban high-resolution air temperature forecast system for local weather information services based on statistical downscaling," *Atmosphere*, vol. 9, no. 5, p. 164, 2018.

[8] "UCI Machine Learning Repository." `https://archive.ics.uci.edu/ml/datasets/Bias+correction+of+numerical+prediction+model+temperature+forecast`. Accessed: 2022-06-29.

[9] G. van Rossum, "Python reference manual," *Department of Computer Science [CS]*, no. R 9525, 1995.

[10] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[11] F. Li and Y. Yang, "Analysis of recursive feature elimination methods," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 633–634, 2005.

[12] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[13] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[14] S. Ross, *A first course in probability*. Pearson, 2010.

[15] W. D. Berry, *Understanding regression assumptions*, vol. 92. Sage, 1993.

[16] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[17] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

7. Apêndice A - Figuras

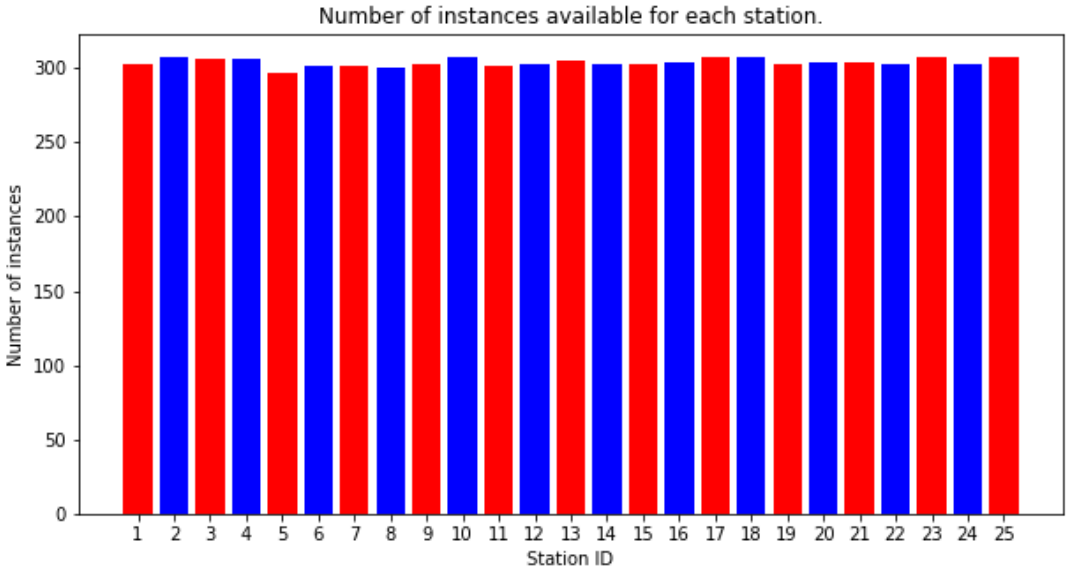

Figura 1. Número de registros disponíveis para cada estação metereológica.

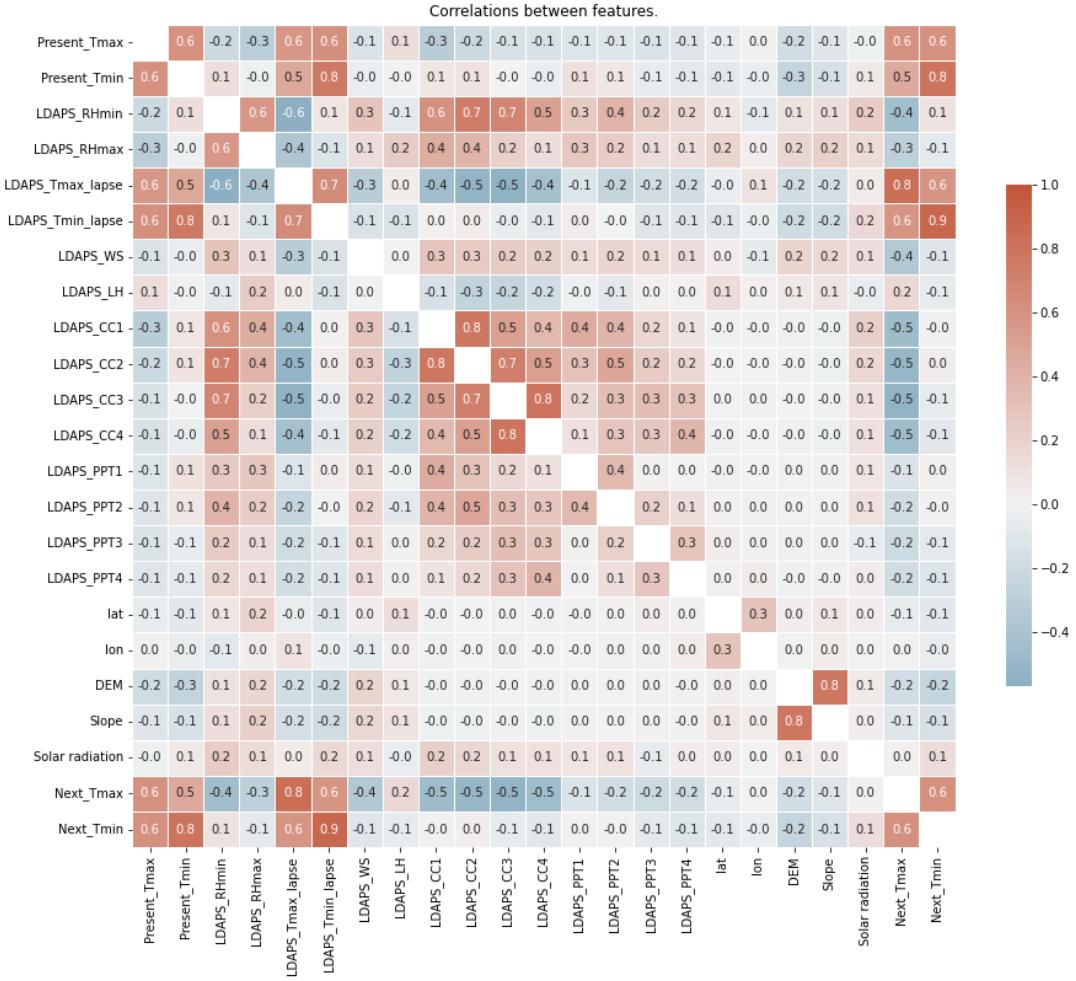

Figura 2. Correlações entre pares de atributos.

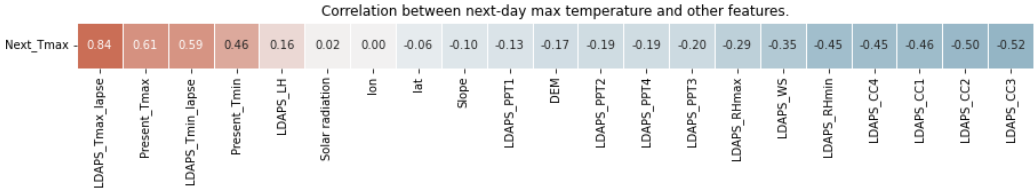

Figura 3. Correlações entre a temperatura máxima do dia seguinte e os demais atributos.

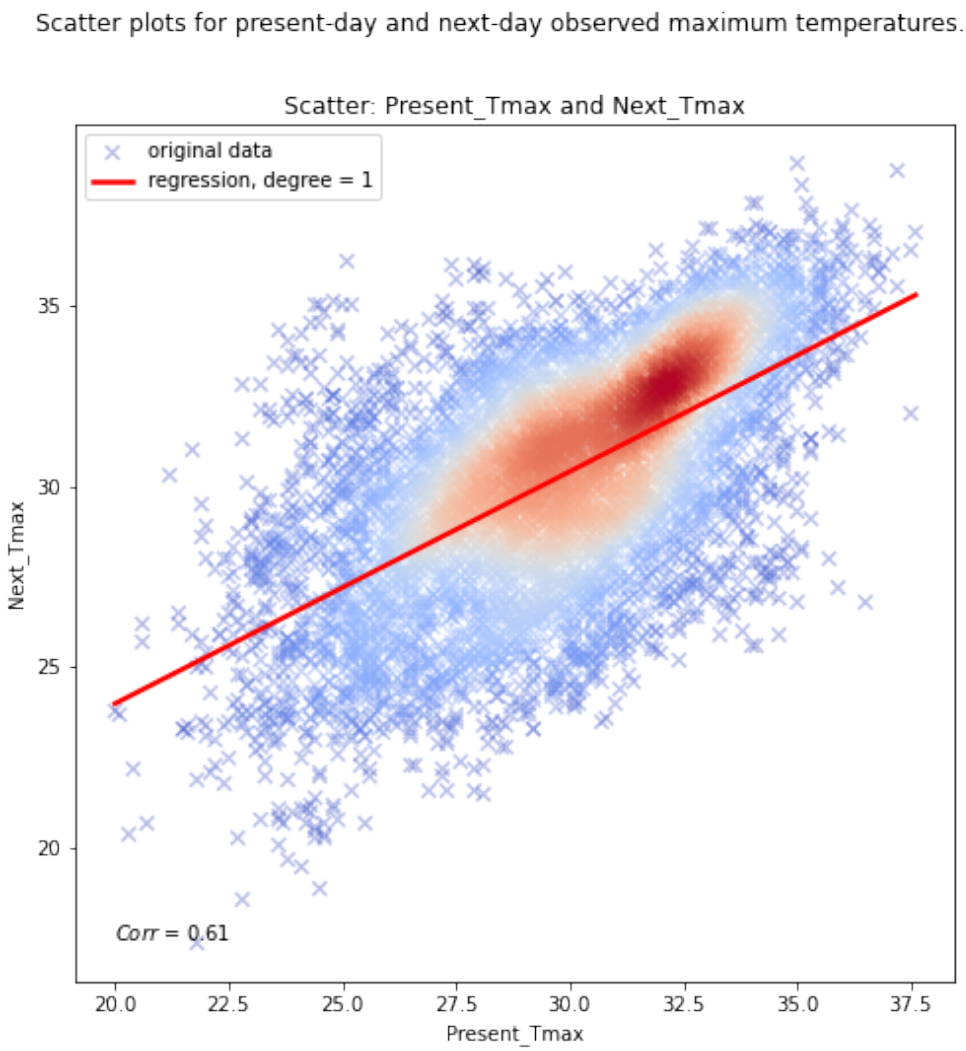

Figura 4. Gráfico de dispersão da temperatura máxima do dia e a temperatura máxima observada no dia seguinte.

Scatter plots for present-day minimum and next-day observed maximum temperatures.

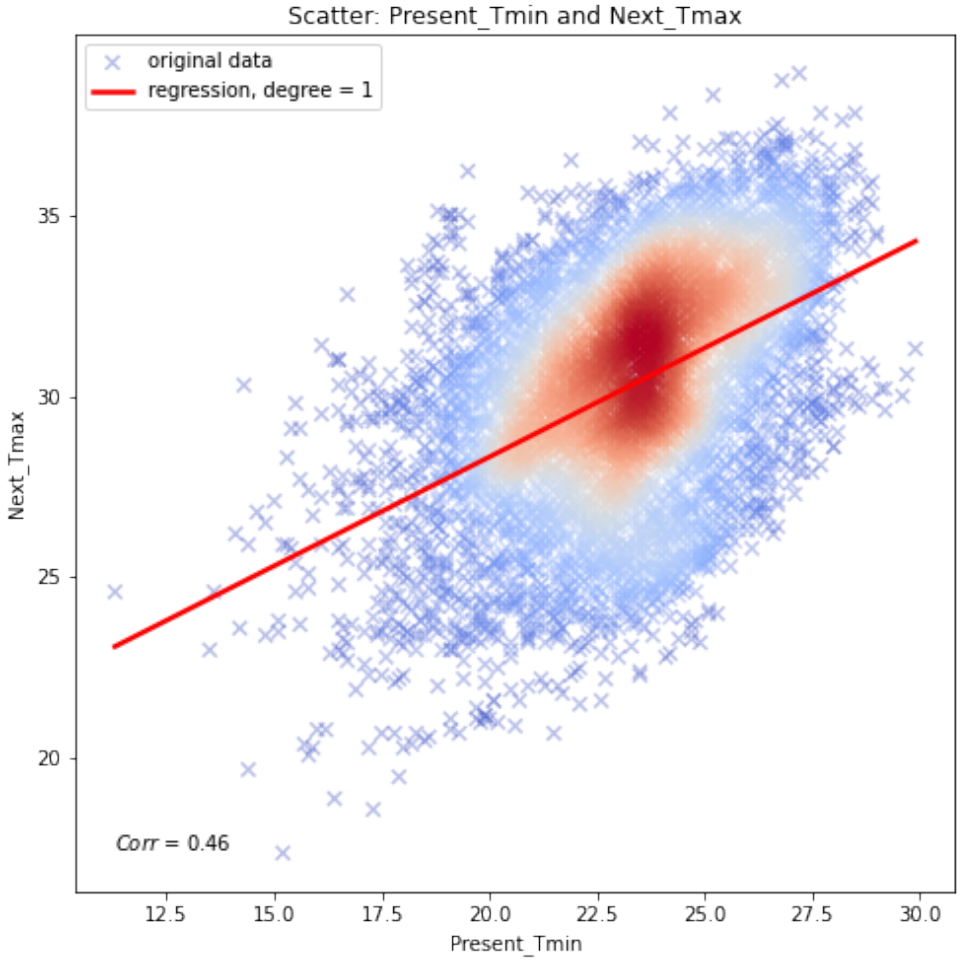

Figura 5. Gráfico de dispersão da temperatura mínima do dia e a temperatura máxima observada no dia seguinte.

atter plots for LDAPS next-day maximum forecast and next-day observed maximum temperatur

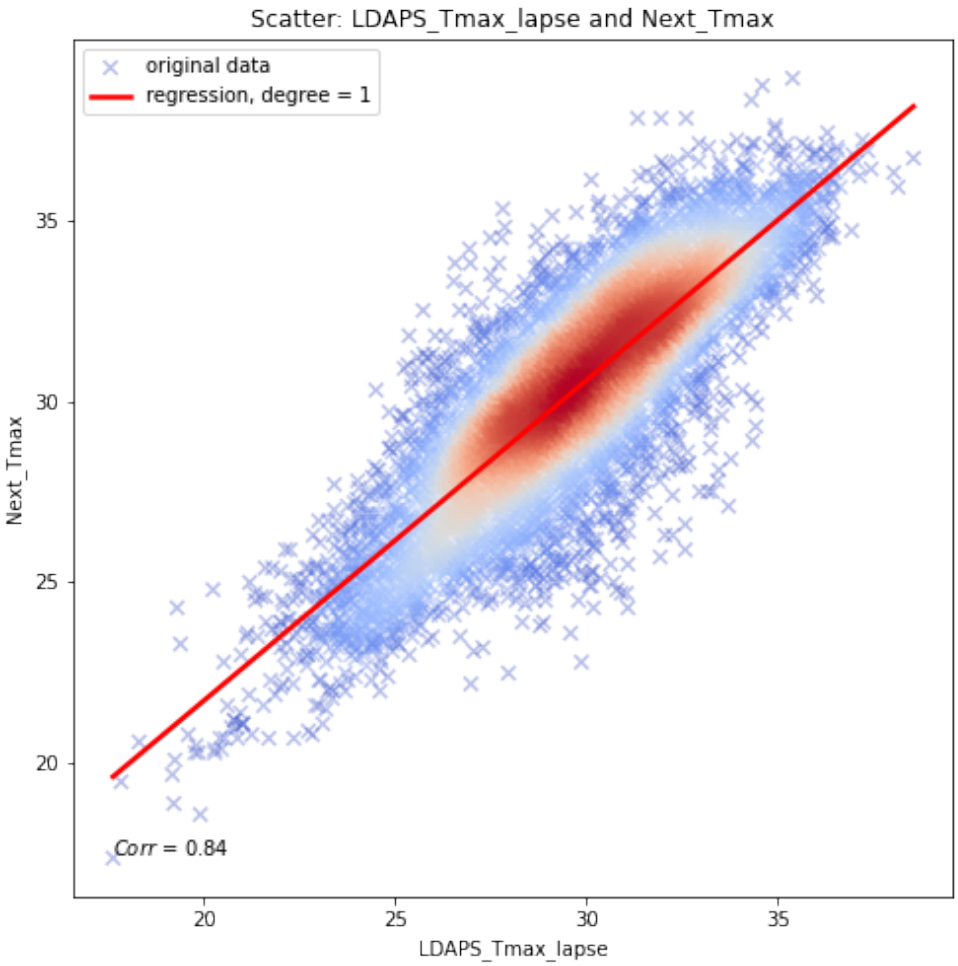

Figura 6. Gráfico de dispersão da temperatura máxima prevista pelo LDPAS para o dia seguinte e a temperatura máxima observada no dia seguinte.

atter plots for LDAPS next-day minimum forecast and next-day observed maximum temperatur

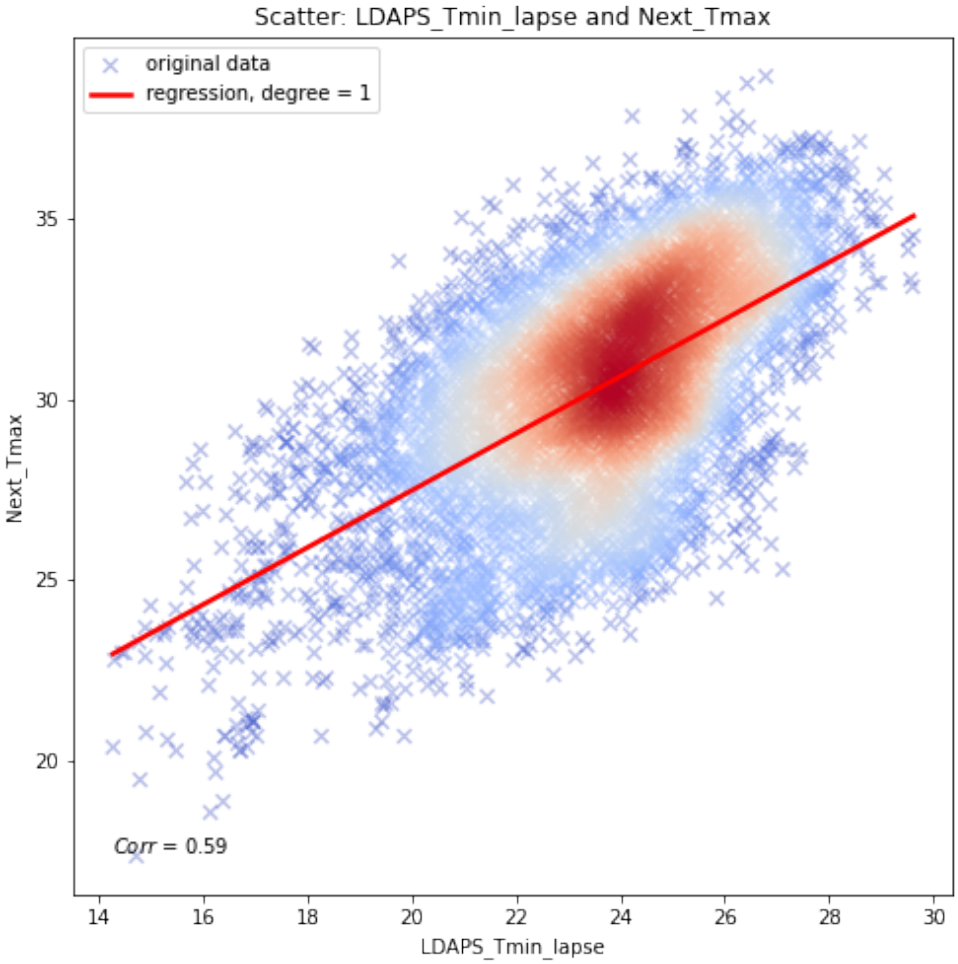

Figura 7. Gráfico de dispersão da temperatura mínima prevista pelo LDPAS para o dia seguinte e a temperatura máxima observada no dia seguinte.

plots for LDAPS next-day maximum humidity forecast and next-day observed maximum tempe

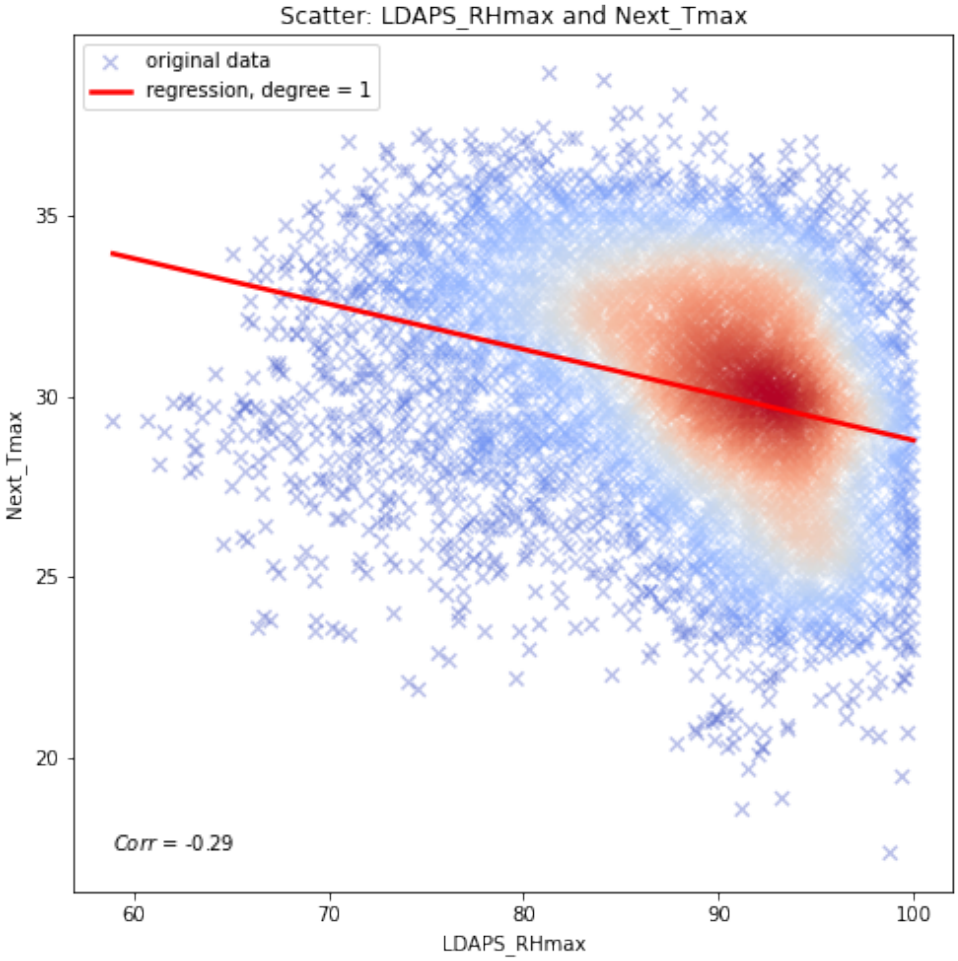

Figura 8. Gráfico de dispersão da humidade relativa máxima prevista pelo LDPAS para o dia seguinte e a temperatura máxima observada no dia seguinte.

· plots for LDAPS next-day minimum humidity forecast and next-day observed maximum tempe

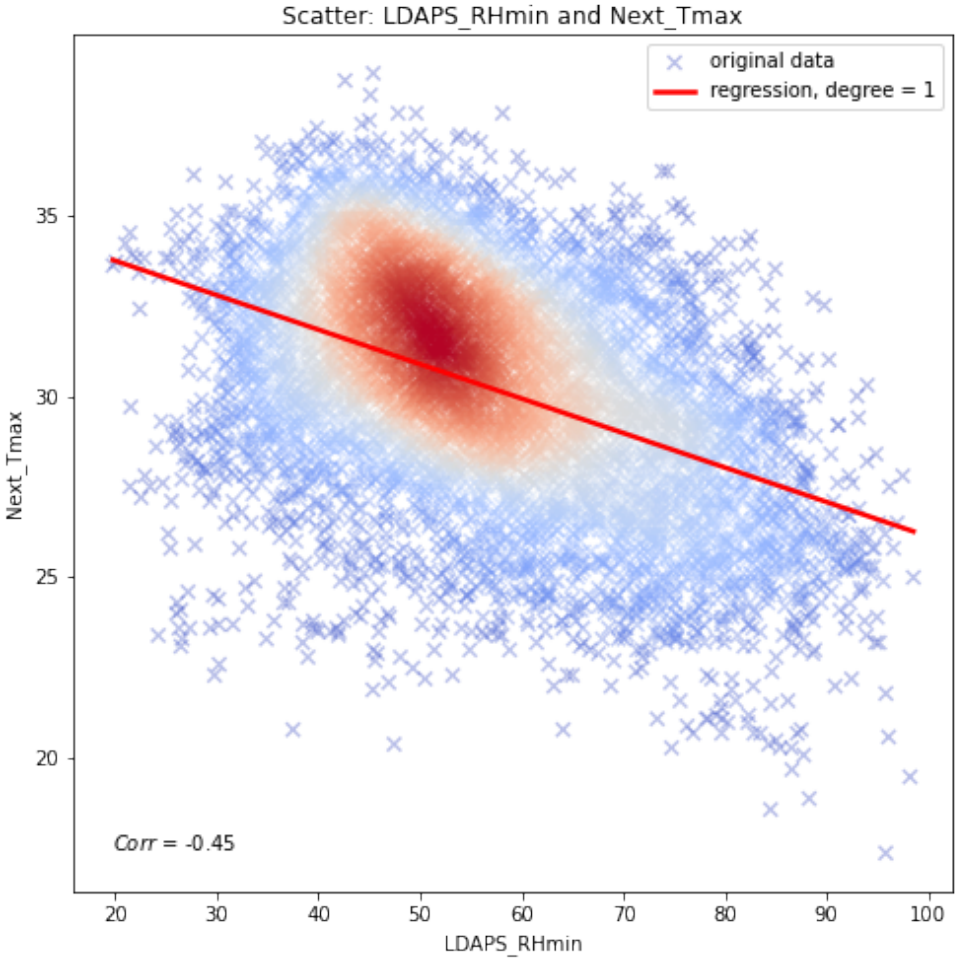

Figura 9. Gráfico de dispersão da humidade relativa mínima prevista pelo LDPAS para o dia seguinte e a temperatura máxima observada no dia seguinte.

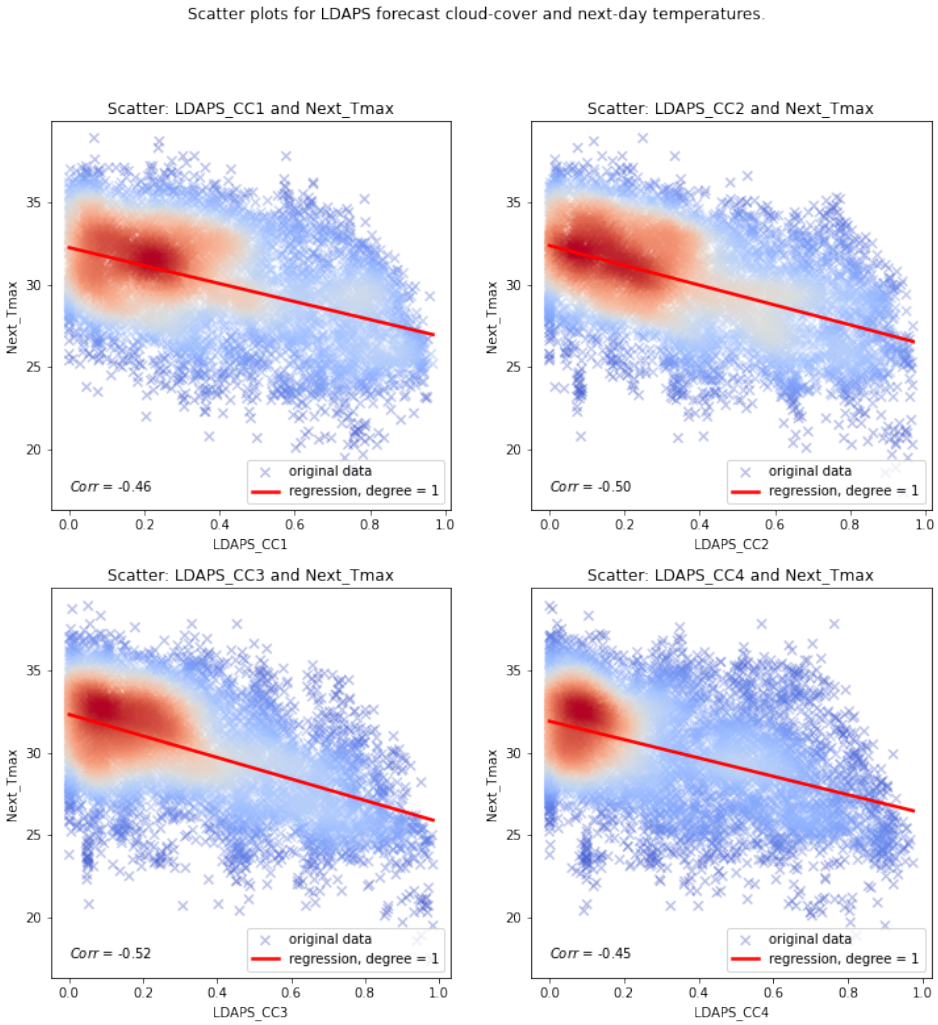

Figura 10. Gráfico de dispersão da nebulosidade percentual prevista pelo LDPAS para diferentes quartos do dia seguinte e a temperatura máxima observada no dia seguinte.

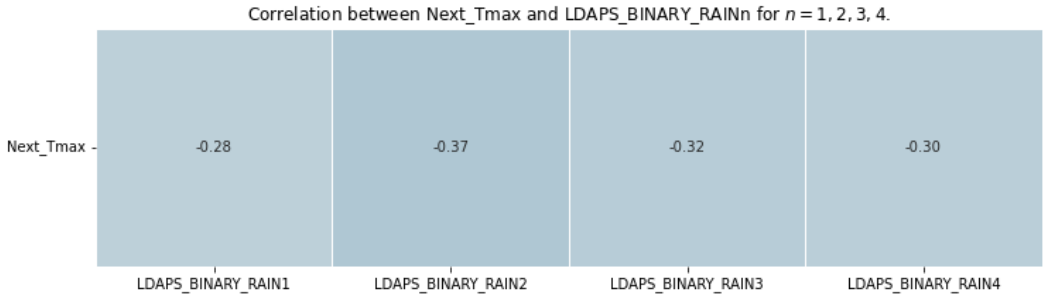

Figura 11. Correlações entre a temperatura máxima observada no dia seguinte e LDAPS_BINARY_RAINi, para i = 1,2,3,4.

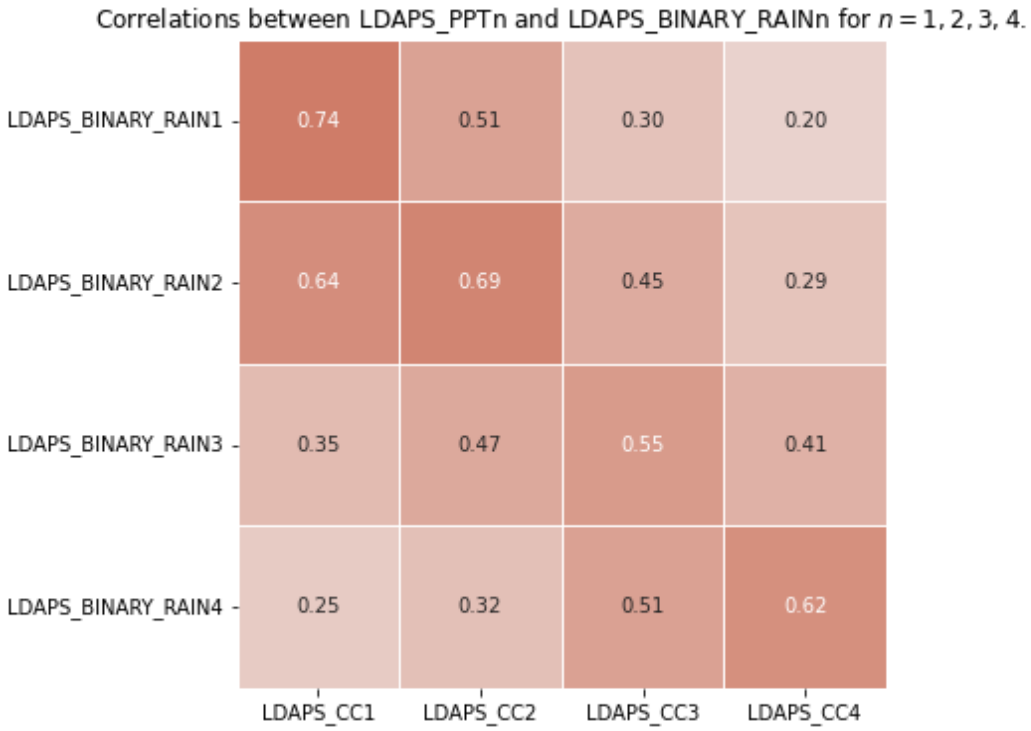

Figura 12. Correlações entre LDAPS_BINARY_RAINi e LDAPS_CCj, para i, j = 1,2,3,4.

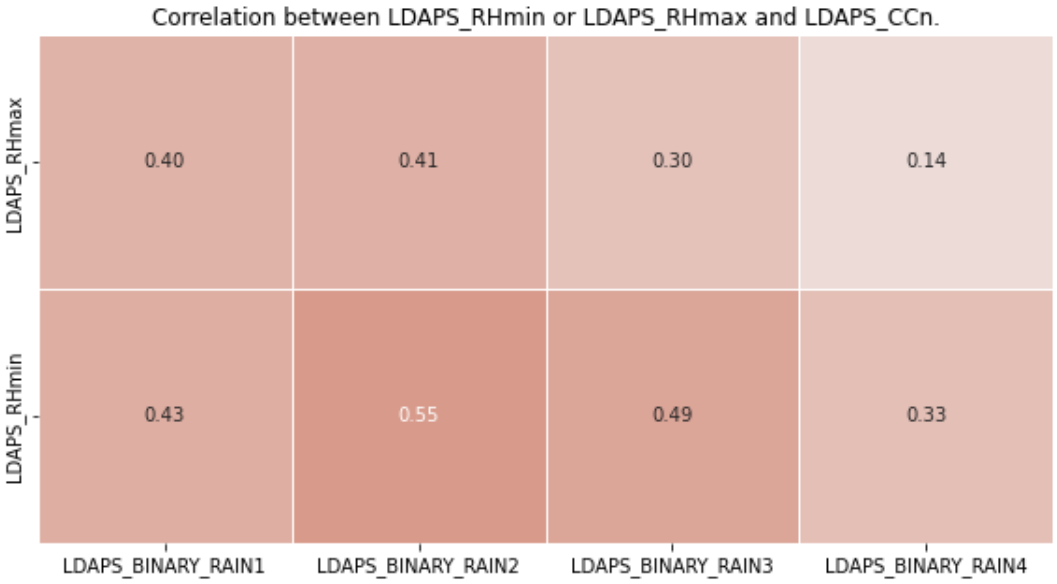

Figura 13. Correlações entre LDAPS_BINARY_RAINi, para i = 1,2,3,4, e LDAPS_RHmax ou LDAPS_RHmin.

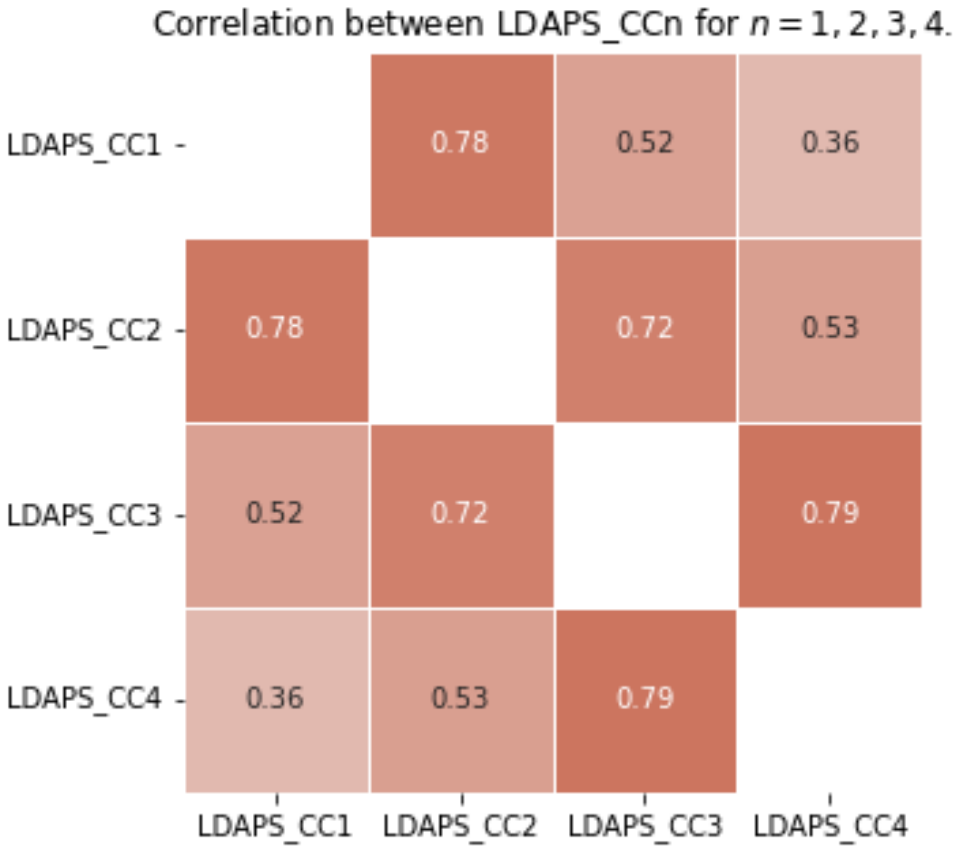

Figura 14. Correlações entre LDAPS_CCi e LDAPS_CCj, para i,j = 1,2,3,4.

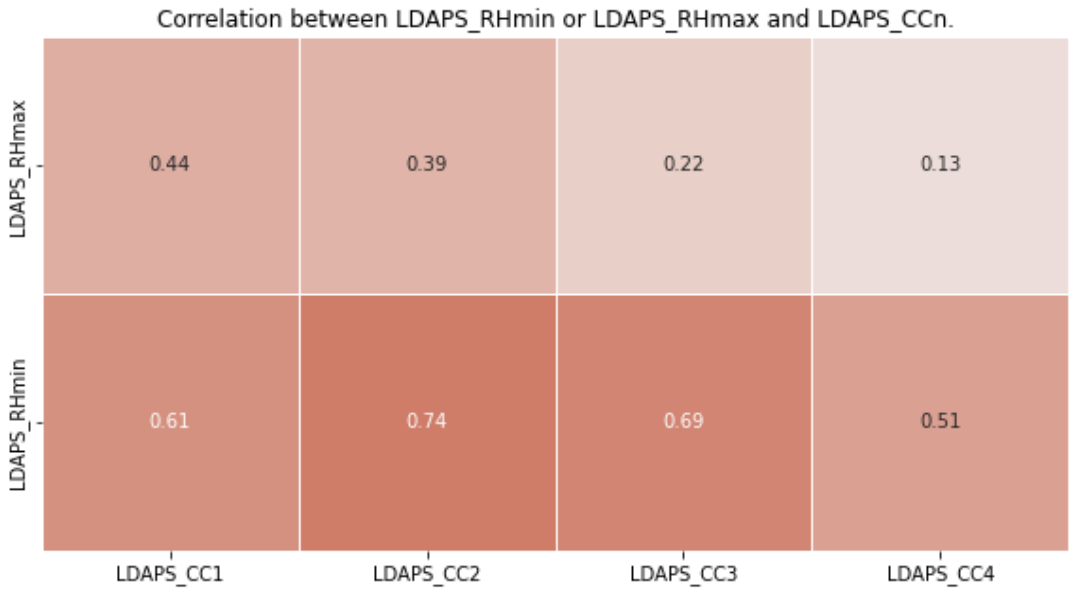

Figura 15. Correlações entre LDAPS_CCi e LDAPS_RHmax ou LDAPS_RHmin, para i,j = 1,2,3,4.

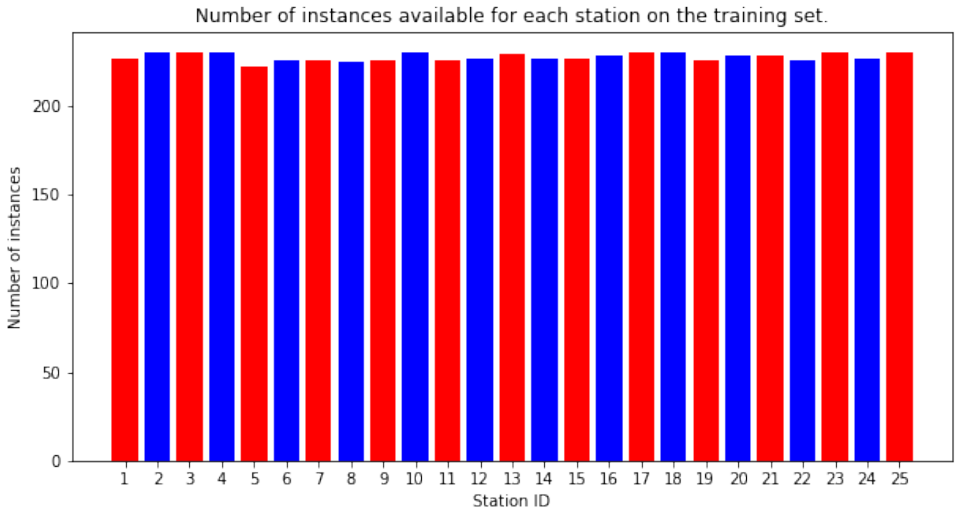

Figura 16. Número de registros disponíveis para cada estação metereológica no conjunto de treino.

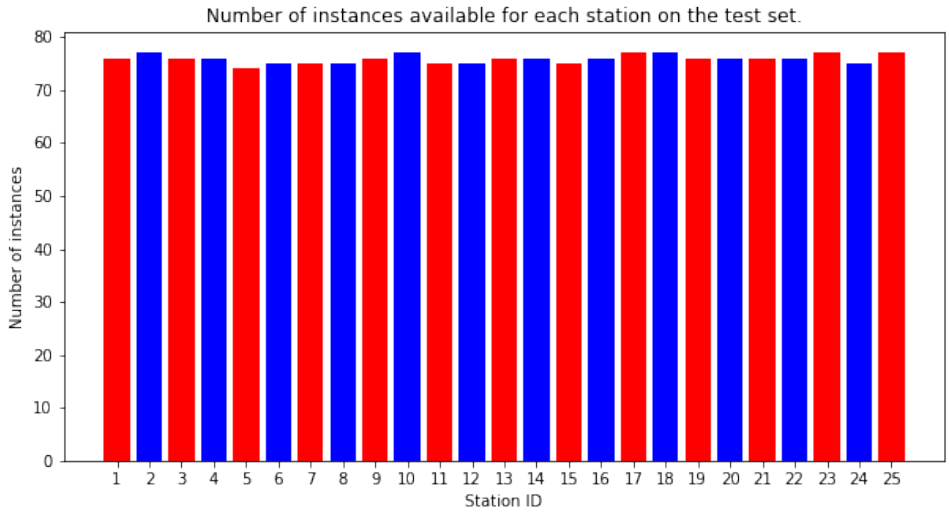

Figura 17. Número de registros disponíveis para cada estação metereológica no conjunto de teste.

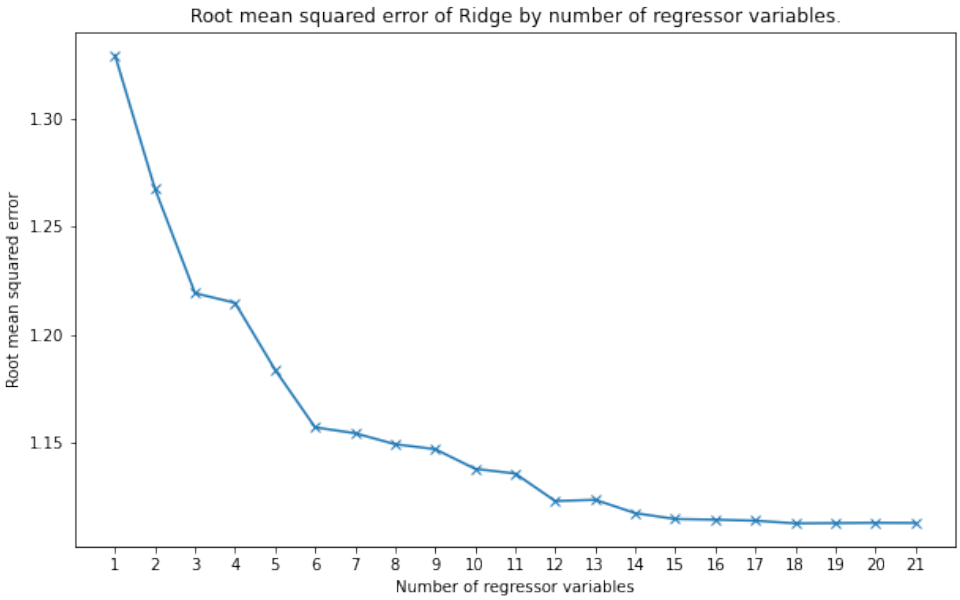

Figura 18. RMSE do Ridge ao longo do procedimento de backward-stepwise feature selection.

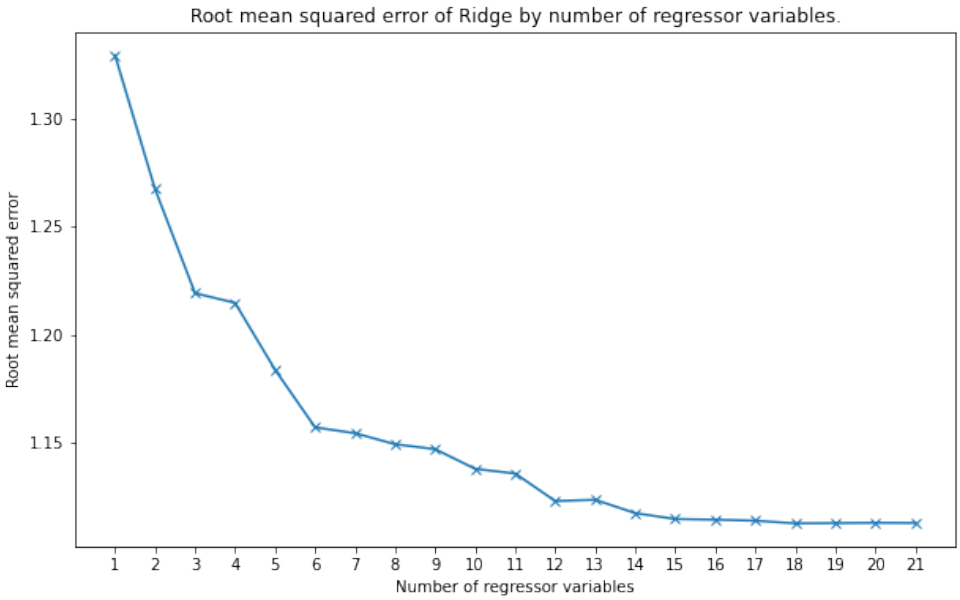

Figura 19. RMSE do Lasso ao longo do procedimento de backward-stepwise feature selection.

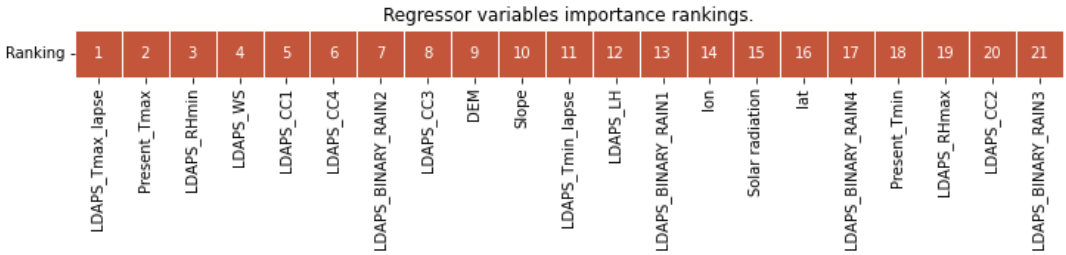

Figura 20. Ordem decrescente de significância das variáveis regressoras sugerido pelo procedimento de backward-stepwise feature selection.

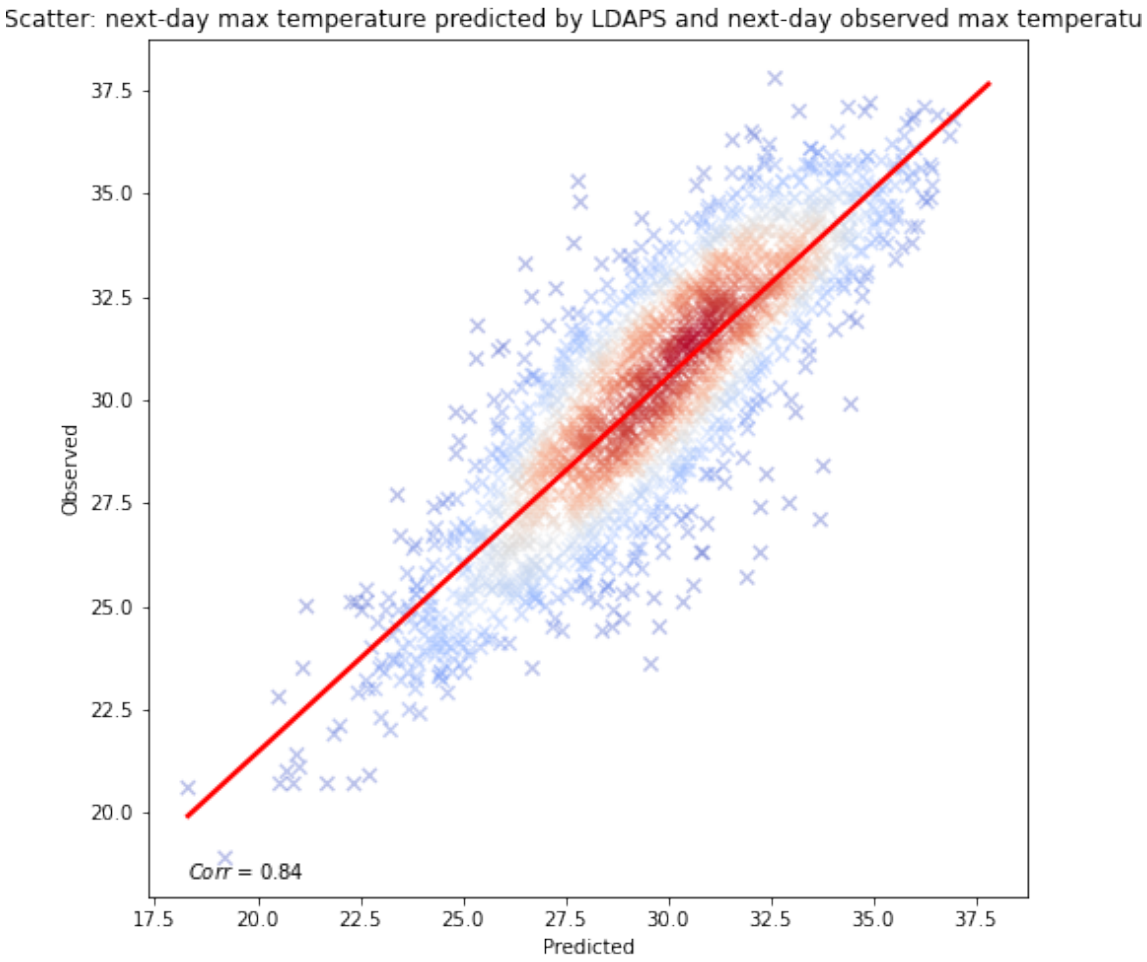

Figura 21. Gráfico de dispersão das estimativas feitas pelo modelo LDAPS e a temperatura máxima observada no dia seguinte.

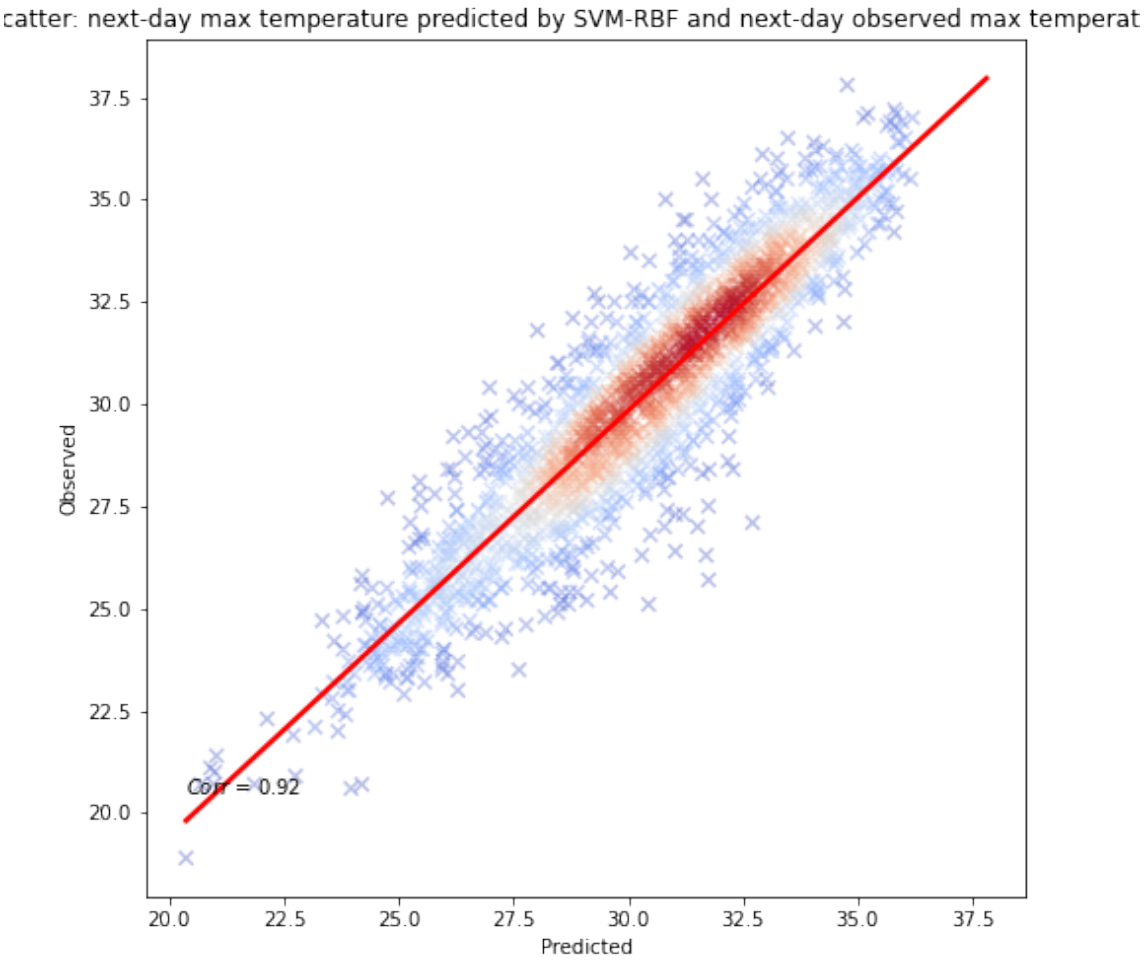

Figura 22. Gráfico de dispersão das estimativas feitas pela SVM e a temperatura máxima observada no dia seguinte.

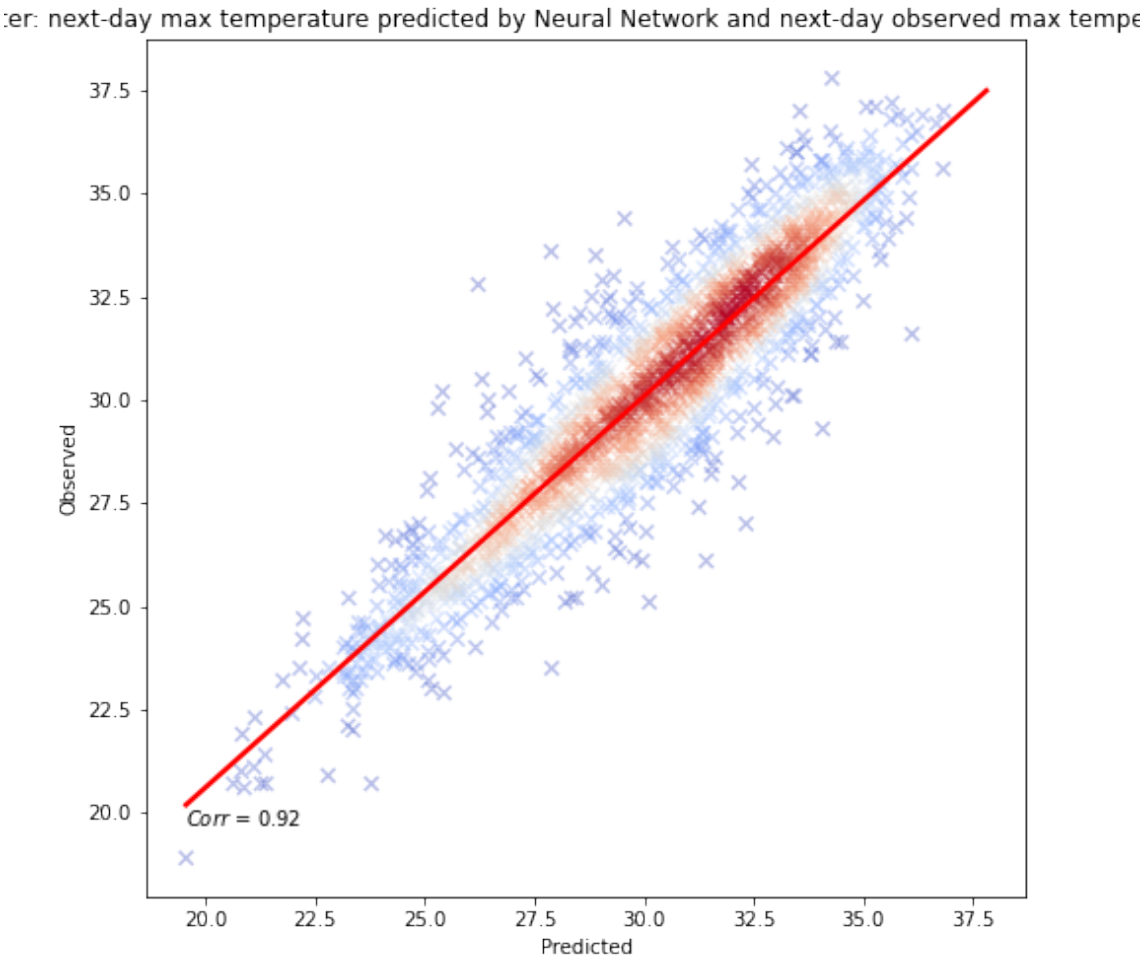

Figura 23. Gráfico de dispersão das estimativas feitas pela rede neural e a temperatura máxima observada no dia seguinte.

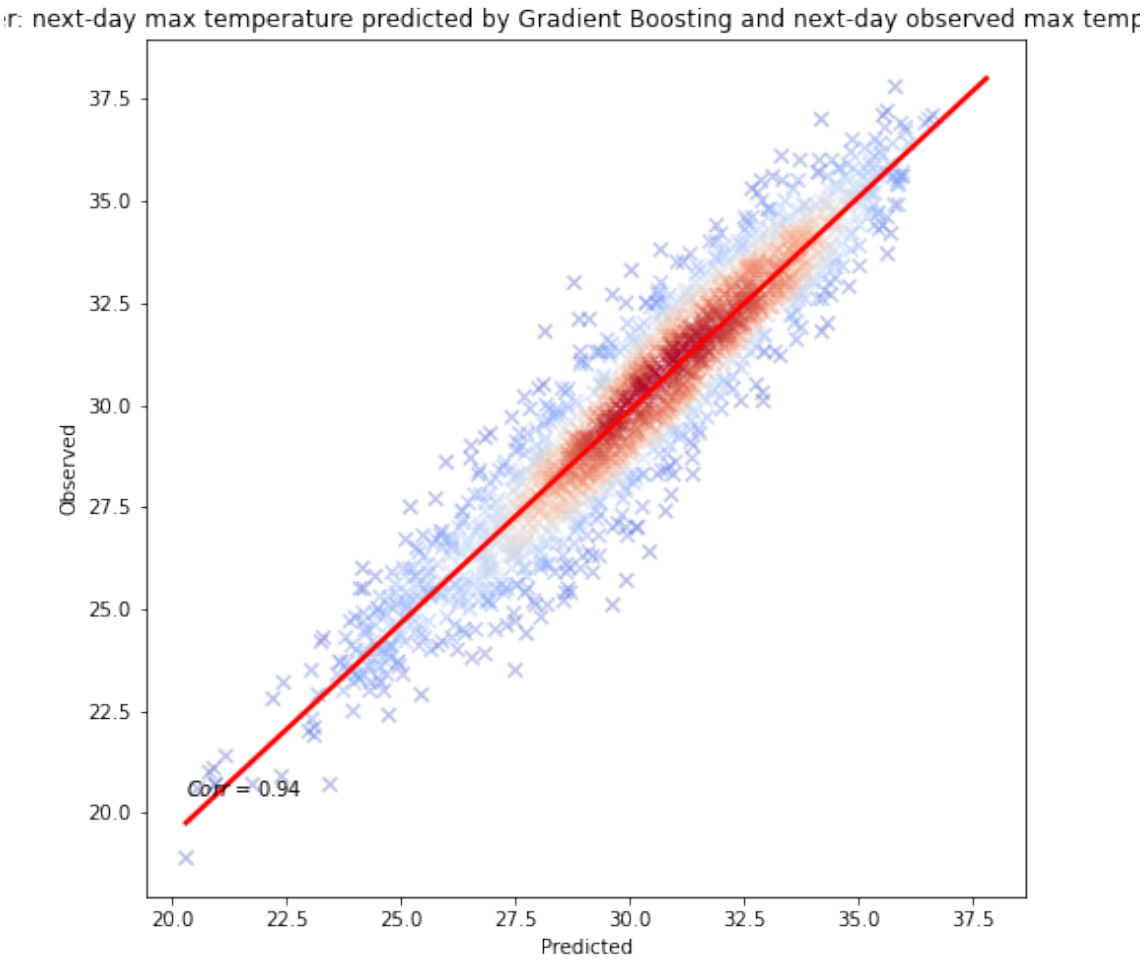

Figura 24. Gráfico de dispersão das estimativas feitas pelo gradient boosting e a temperatura máxima observada no dia seguinte.

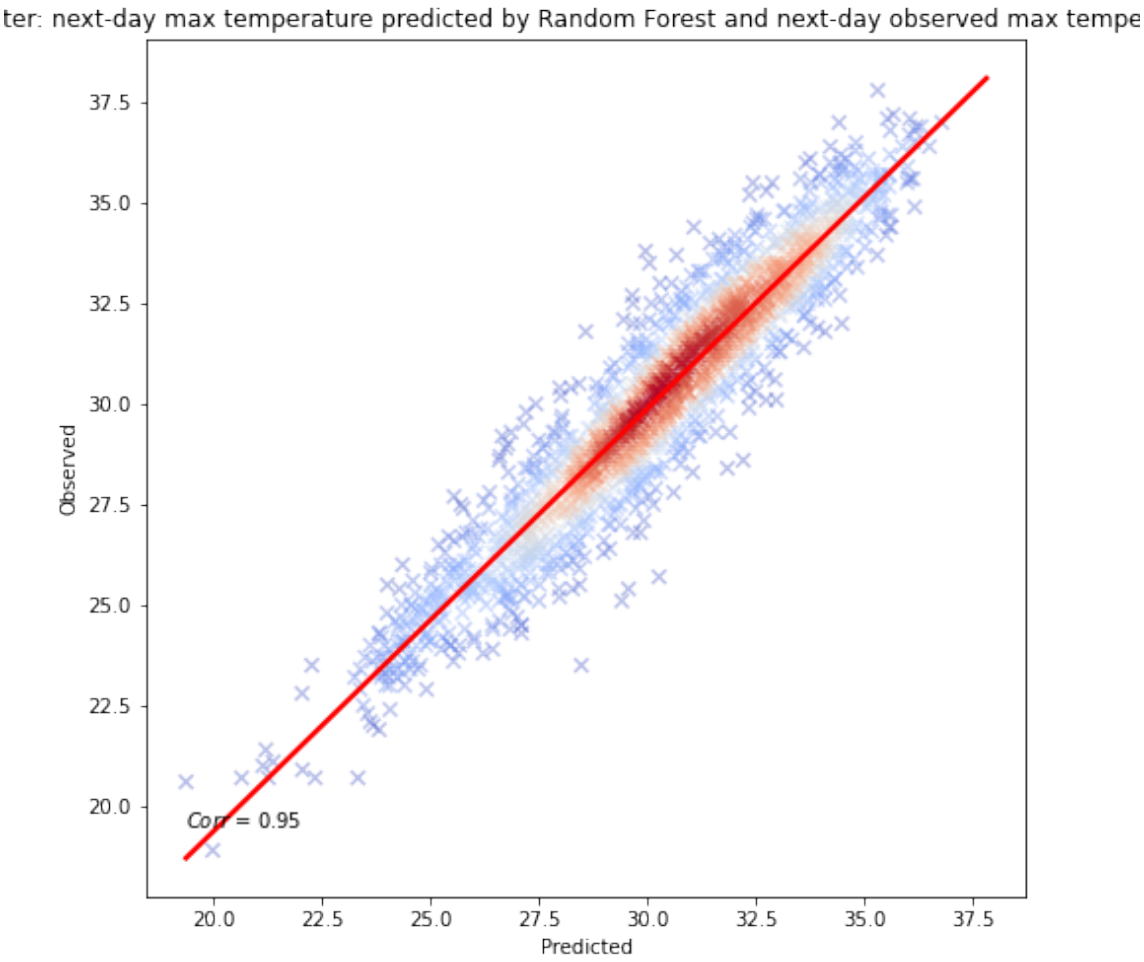

Figura 25. Gráfico de dispersão das estimativas feitas pela floresta aleatória e a temperatura máxima observada no dia seguinte.

8. Apêndice B - Tabelas

Model	Min	Média	Max	Std
LDAPS Forecast	0.59	0.64	0.67	0.03
Decision Tree, d=100	0.66	0.72	0.76	0.03
Gradient Boosting, d=100	0.67	0.74	0.79	0.03
Decision Tree, d=5	0.70	0.74	0.77	0.02
SVM, Poly Kernel	0.72	0.76	0.79	0.02
SVM, Linear Kernel	0.73	0.77	0.79	0.02
Lasso	0.74	0.77	0.79	0.02
Ridge	0.74	0.77	0.79	0.02
Linear Regression	0.74	0.77	0.79	0.02
Random Forest MSE, d=5	0.74	0.77	0.80	0.02
Neural Net, L=30	0.72	0.78	0.83	0.03
Neural Net, L=10	0.77	0.79	0.81	0.01
Neural Net, L=20	0.74	0.80	0.83	0.03
SVM, RBF Kernel	0.79	0.82	0.85	0.02
Random Forest, d=100	0.81	0.84	0.86	0.02
Gradient Boosting, d=5	0.84	0.87	0.89	0.02

Tabela 1. Valores de R^2 para os diferentes modelos ao longo dos dez ciclos de validação cruzada. Os modelos estão ordenados em ordem crescente da média dos R^2.

Modelo	Min	Média	Max	Std
LDAPS Forecast	1.77	1.86	1.97	0.06
Decision Tree MSE, d=100	1.53	1.64	1.74	0.07
Gradient Boosting MSE, d=100	1.47	1.57	1.70	0.08
Decision Tree MSE, d=5	1.49	1.56	1.66	0.05
SVM, Poly Kernel	1.43	1.52	1.62	0.06
Lasso	1.42	1.49	1.57	0.04
SVM, Linear Kernel	1.42	1.49	1.58	0.05
Ridge	1.42	1.49	1.57	0.04
Linear Regression	1.42	1.49	1.57	0.05
Random Forest MSE, d=5	1.39	1.47	1.55	0.05
Neural Net, L=30	1.27	1.44	1.64	0.11
Neural Net, L=10	1.34	1.40	1.47	0.04
Neural Net, L=20	1.26	1.37	1.51	0.07
SVM, RBF Kernel	1.22	1.29	1.39	0.05
Random Forest MSE, d=100	1.14	1.23	1.34	0.06
Gradient Boosting MSE, d=5	1.03	1.13	1.23	0.06

Tabela 2. Valores de RMSE para os diferentes modelos ao longo dos dez ciclos de validação cruzada. Os modelos estão ordenados em ordem decrescente da média dos RMSE.

Model	R^2	RMSE
LDAPS Forecast	0.67	1.82
SVM, RBF Kernel	0.84	1.26
Neural Network, L=10	0.84	1.26
Gradient Boosting, d=5	0.88	1.09
Random Forest, d=5	0.89	1.04

Tabela 3. Valores de R^2 e RMSE para os diferentes modelos no conjunto de teste. Os modelos estão ordenados em ordem crescente de R^2.