Universidade de Brasília
Departamento de Estatística

Interpretação de Redes Neurais Artificiais

Lucas de Moraes Pinto Pereira

Brasília
2023

Lucas de Moraes Pinto Pereira

Interpretação de Redes Neurais Artificiais

Orientador(a): Prof(a). Thais Rodrigues

Projeto apresentado para o Departamento de Estatística da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Brasília
2023

Resumo

O artigo apresenta uma análise comparativa entre modelos estatísticos e de aprendizado de máquina, destacando que os modelos de aprendizado de máquina são mais competitivos em relação à previsão, classificação e agrupamento de dados, mas possuem limitações na interpretação e generalização de resultados. Nesse contexto, é abordada a técnica SHAP como uma forma de interpretar algoritmos complexos e melhorar a compreensão dos modelos de aprendizado de máquina, identificando as covariáveis mais importantes. O estudo também demonstra que a rede neural tem maior capacidade de predição para dados mais complexos, porém o tempo computacional necessário para ajustar esses modelos é significativamente maior em relação à Regressão Linear Múltipla. Portanto, a interpretabilidade dos modelos é fundamental para o entendimento das decisões tomadas pelo algoritmo, fornecendo insights valiosos para a otimização do processo preditivo.

Palavras-Chave: Redes Neurais Artificiais; Regressão Linear; SHAP; Inteligência Artificial Explicável.

Lista de Tabelas

Lista de Figuras

Sumário

1 Introdução

Historicamente, a comunidade estatística internacional dedicou grande parte dos seus esforços ao desenvolvimento de modelos probabilísticos analiticamente convenientes e altamente interpretáveis. Nessa abordagem, fortes suposições técnicas são adotadas, incluindo a forma funcional das relações entre as variáveis em estudo e as distribuições de probabilidade associadas, o que limita sobremaneira a capacidade de representação desses modelos.

No contexto atual, em virtude do crescimento da complexidade, bem como volume dos dados disponíveis, métodos algorítmicos de aprendizagem de máquina passaram a oferecer soluções competitivas com resultados extraordinários, sobretudo para as atividades de previsão, classificação e agrupamento. Entretanto, tais benefícios trouxeram consigo perdas substanciais na capacidade de abstração, interpretação e generalização.

Tendo em vista o dilema da interpretação em contraste com o desempenho, há um crescente interesse na criação de técnicas e ferramentas de interpretação e visualização do processo preditivo induzido pelos algoritmos do tipo caixa-preta.

Nesse sentido, uma questão importante no campo do aprendizado de máquina é por que um algoritmo tomou uma determinada decisão. Tal fato é de demasiada importância, pois para um usuário final, existe uma maior probabilidade de confiança em uma recomendação caso esse entenda o motivo de sua exposição a ele. Para uma organização, por exemplo, entender que os clientes fizeram uma compra porque essa campanha foi particularmente eficaz pode permitir adaptar os futuros esforços de divulgação.

No entanto, este é um campo desafiador e ainda em desenvolvimento no campo do aprendizado de máquina. Neste estudo, além de comparar modelos probabilísticos e de aprendizado de máquina, será discutido e trabalhado maneiras de interpretar um modelo explorando uma nova técnica chamada SHAP (SHapley Additive exPlanations) ((Lundberg and Lee, 2017)). Essa técnica mostrou-se eficaz na compreensão de algoritmos complexos nos trabalhos de Meng et al. ((2020)).

Existem também diversas outras técnicas para a interpretação de Redes Neurais, como por exemplo a Integrated Gradients ((Sundararajan et al., 2017)), e a DeepLift ((Shrikumar et al., 2017)). Entretanto, a técnica SHAP foi escolhida para este estudo pois pode ser aplicada em praticamente todo tipo de algoritmo de aprendizagem de máquina e também possui uma excelente implementação na linguagem Python, com muitos recursos gráficos interessantes.

2 Objetivos

2.1 Objetivos Gerais

O objetivo principal deste trabalho é analisar e ilustrar alguns dos principais métodos de interpretação e visualização de algoritmos de previsão, com foco em Redes Neurais Artificiais.

2.2 Objetivos Específicos

- Estudar modelos probabilísticos tradicionais.
- Estudar algoritmos de aprendizado de máquina.
- Desenvolver exemplos aplicados que contrastem as virtudes e limitações de cada abordagem em consideração, a saber, via modelos probabilísticos tradicionais e via algoritmos de aprendizado de máquina.
- Estudar e aplicar os métodos SHAP, para interpretação de Redes Neurais.

3 Metodologia

3.1 Conjunto de Dados

Os dados utilizados no presente trabalho serão gerados por meio de simulação computacional, a partir de funções pré-estabelecidas, visando criar diferentes cenários em que conhecemos o comportamento real dos dados, com o intuito de comparar as diferentes técnicas de modelagem nas variadas situações simuladas. Essa abordagem torna possível realizar uma melhor comparação entre o modelo probabilístico e o de aprendizado de máquina, pois, após o ajuste dos modelos, é possível constatar qual estimou melhor as funções pré-determinadas e obteve o menor erro em cada cenário de dados criado.

Os cenários escolhidos serão detalhados no Capítulo 4 Resultados.

3.2 Regressão Linear Múltipla

O modelo de regressão linear múltipla estimado por mínimos quadrados ordinários (MQO) permite modelar a relação entre uma variável dependente e um conjunto de variáveis explicativas. Este modelo assume uma relação linear entre uma variável dependente Y e um conjunto de variáveis explicativas $x_{i0}, x_{i1}, ..., x_{iK}$. x_{ik} estas também são chamadas de variáveis independentes ou covariáveis. A primeira covariável $x_0 = 1$ é uma constante, a menos que especificado de outra forma Neter et al. ((1996)).

Considere uma amostra de N observações y_i, i = 1, ... , N, obtida do modelo

$$Y_i = x_i'\beta + \varepsilon_i, \tag{3.2.1}$$

onde β é um vetor coluna (K + 1)-dimensional de parâmetros, x_i é um vetor linha (K + 1) dimensional de covariáveis e ε_i é um escalar chamado termo de erro. Em termos matriciais, temos que

$$Y = X\beta + \varepsilon, \tag{3.2.2}$$

onde Y é um vetor coluna N-dimensional, X é uma matriz N $\times$ (K + 1) e ε é um vetor coluna N-dimensional de erro, ou seja,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{11} & \cdots & x_{11} \\ 1 & x_{12} & x_{11} & \cdots & x_{11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{kn} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} + \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_k \end{bmatrix}. \tag{3.2.3}$$

Os mínimos quadrados ordinários (MQO) minimizam as distâncias quadradas entre a variável dependente observada e prevista pelo modelo::

$$S(\beta) = \sum_{i=1}^{N} (y_i - x'\beta)^2 = (y - X\beta)'(y - X\beta). \tag{3.2.4}$$

O estimador MQO resultante de β é:

$$\hat{\beta} = (X'X)^{-1} Xy. \tag{3.2.5}$$

Dado o estimador $\hat{\beta}$, podemos prever a variável dependente por $\hat{y}_i = x_i'\hat{\beta}$ e o termo de erro por $\hat{\epsilon}_i = y_i - x_i'\hat{\beta}$, este é chamado de resíduo Helene ((2006)).

A aplicação do modelo de regressão linear múltipla (bem como da simples) pressupõe a verificação de alguns pressupostos Neter et al. ((1996)):

1. Os erros ε_i são variáveis aleatórias de média zero;
2. Os erros ε_i são variáveis aleatórias de variância constante (σ^2) – hipótese de homocedasticidade;
3. As variáveis aleatórias $\varepsilon_1, \varepsilon_2, ..., \varepsilon_n$ são independentes;
4. As variáveis explicativas $X_1, X_2, ..., X_n$ são não correlacionadas – hipótese de ausência de multicolinearidade entre as variáveis explicativas;
5. Os erros ε_i seguem uma distribuição normal: $\varepsilon_i \sim N(0, \sigma^2)$.
6. Todos os expoentes dos parâmetros são iguais a 1 - hipótese da linearidade dos parâmetros.

Assumindo as hipóteses de 1 a 6, as seguintes propriedades podem ser estabelecidas para amostras finitas, ou seja, mesmo pequenas.

- O estimador MQO de β é não viesado:

$$E[\hat{\beta}|X] = \beta. \tag{3.2.6}$$

- O estimador MQO é (multivariado) normalmente distribuído:

$$\hat{\beta}|X \sim N(\beta, V[\hat{\beta}|X]), \tag{3.2.7}$$

com variância $V[\hat{\beta}|X] = \sigma^2(X'X)^{-1}$, sob a hipótese de homocedasticidade e $V[\hat{\beta}|X] = \sigma^2(X'X)^{-1}\ X'\Omega X(X'X)^{-1}$, considerando heterocedasticidade conhecida, onde Ω é a matriz de variância-covariância dos erros. Sob homocedasticidade, a variância V pode ser estimada não viesadamente como

$$\hat{V}(\hat{\beta}|X) = \hat{\sigma^2}(X'X)^{-1}, \tag{3.2.8}$$

com

$$\hat{\sigma^2} = \frac{\hat{\epsilon}'\hat{\epsilon}}{N-K-1}. \tag{3.2.9}$$

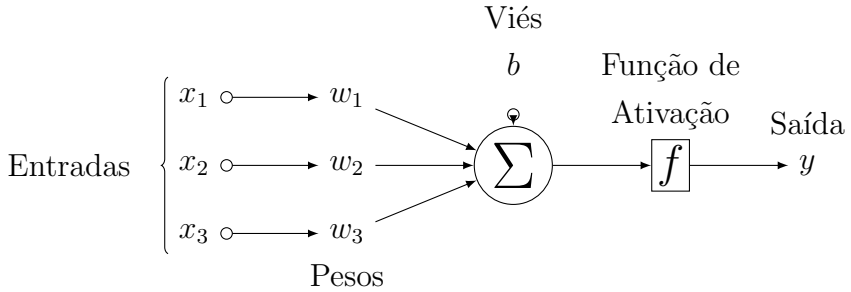
3.3 Redes Neurais Artificiais

Tendo em vista o grande número de hipóteses do modelo de regressão linear múltipla, em contraste com a complexidade natural dos dados reais, em muitas ocasiões essa técnica se mostra inapropriada. O modelo de Redes Neurais Artificiais é uma técnica alternativa para previsão, que não assume as hipóteses listadas anteriormente.

Antes de falar uma Rede Neural, é necessário examinar seu bloco de construção mais básico, ou seja, um Neurônio.

Como em um cérebro humano, o bloco de construção básico de uma Rede Neural é um neurônio. Sua funcionalidade é semelhante ao neurônio de um cérebro humano, ou seja, ele recebe algumas entradas e dispara uma saída. Cada neurônio é uma pequena unidade de computação que recebe um conjunto de números reais como entrada, realiza alguma computação sobre eles e produz um único valor de saída, conforme ilustrado na Figura 1 Beale and Jackson ((1990)).

Figura 1: Neurônio da Rede Neural



Cada entrada (x) para um neurônio tem um *peso* associado (w), que é atribuído com base em sua importância relativa para outras entradas. A maneira como um neurônio funciona é: se a soma ponderada das entradas for maior que um limite específico, ele fornece uma saída 1, caso contrário uma saída 0. Este é o modelo matemático de um neurônio, também conhecido como "Perceptron". Cada unidade neural recebe uma soma ponderada de suas entradas, com um termo adicional na soma chamado de *Viés*. O viés é uma constante que é usada para ajustar a saída junto com a soma ponderada das entradas, para que o modelo possa se ajustar melhor aos dados fornecidos.

Utilizando notação vetorial, define-se a soma ponderada z em termos do vetor de peso w, vetor de entrada x e um valor de viés b,

$$z = w \cdot x + b. \tag{3.3.1}$$

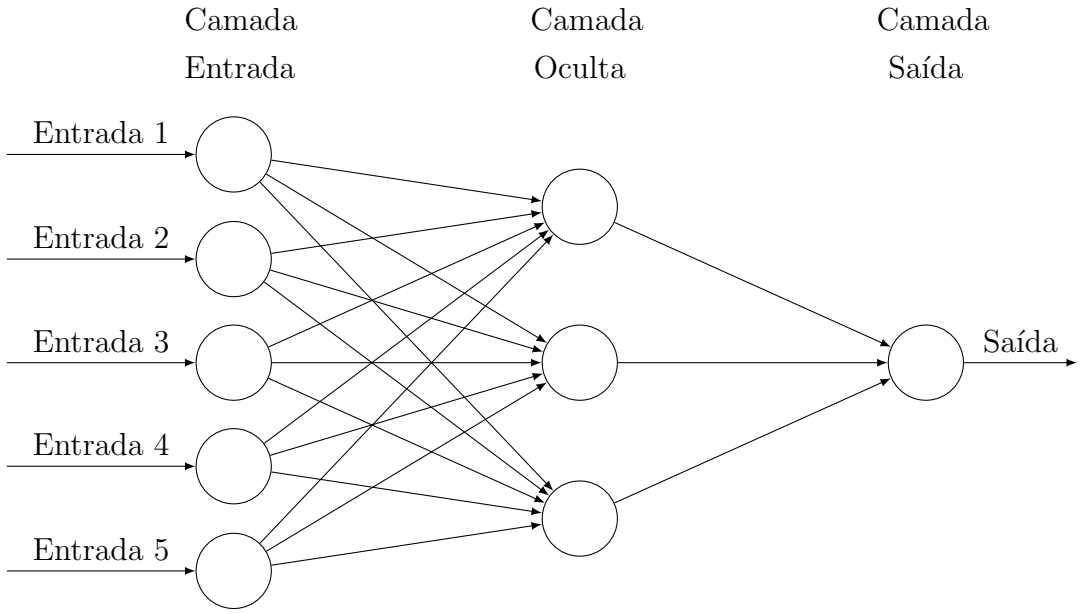
A saída (y) do neurônio é uma função f da soma ponderada das entradas z. Esta função, em geral, não é linear e é chamada de *Função de Ativação*,

$$y = a = f(z). \tag{3.3.2}$$

O objetivo da função de ativação é introduzir não linearidade na saída do neurônio, pegando um único número e executando alguma operação matemática nele.

Uma rede neural é composta de camadas, que é uma coleção de neurônios, com conexões entre elas. Essas camadas transformam os dados, primeiro calculando a soma ponderada das entradas e depois normalizando-a usando as funções de ativação atribuídas aos neurônios Haykin and Network ((2004)). A camada mais à esquerda em uma Rede Neural é chamada de camada de entrada e a camada mais à direita é chamada de camada de saída. As camadas entre a entrada e a saída são chamadas de camadas ocultas. Qualquer rede neural tem 1 camada de entrada e 1 camada de saída. No entanto, o número de camadas ocultas difere entre diferentes redes, dependendo da complexidade do problema. Além disso, cada camada oculta pode ter sua própria função de ativação. A figura 2 ilutra uma rede neural com 1 camada oculta apenas.

Figura 2: Rede Neural com uma camada oculta



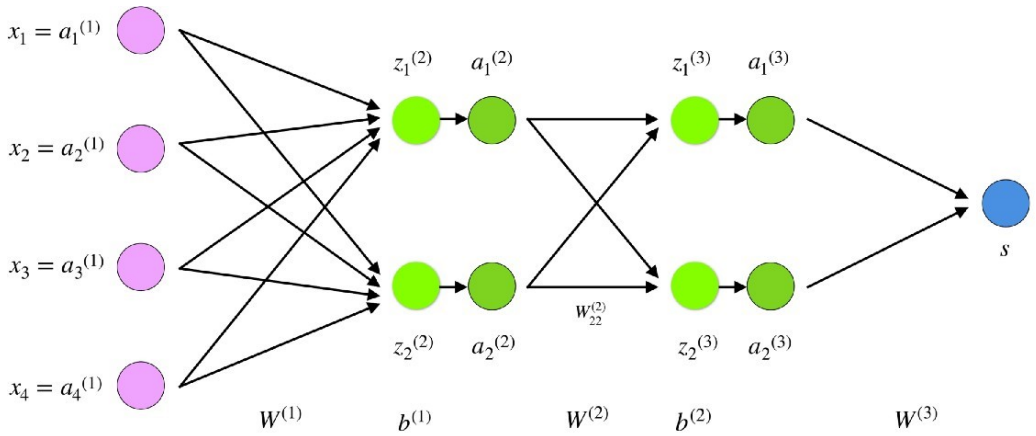
Qualquer rede neural com pelo menos 2 camadas ocultas é chamada de Rede Neural Profunda. Uma rede neural faz previsões aprendendo os pesos de cada um dos neurônios em cada camada. O algoritmo através do qual eles aprendem é chamado de "Back Propagation".

O algoritmo "Back Propagation", ou de retropropagação, é provavelmente o bloco de construção mais fundamental em uma rede neural. Foi introduzido pela primeira vez na década de 1960 e, quase 30 anos depois (1989) foi popularizado por Rumelhart, Hinton e Williams em um artigo chamado "Aprender representações por erros de retropropagação" Rumelhart et al. ((1986)).

O algoritmo é usado para treinar efetivamente uma rede neural por meio de um método chamado regra da cadeia. Em termos simples, após cada passagem para frente em uma rede, a retropropagação realiza uma passagem para trás enquanto ajusta os parâmetros do modelo (pesos e vieses).

Para exemplificar o algoritmo, será usada uma rede neural de 4 camadas, constituída por uma camada de entrada com 4 neurônios, 4 neurônios para as camadas ocultas e 1 neurônio para a camada de saída.

Figura 3: Ilustração simples de rede neural de 4 camadas



Fonte: "https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd"

3.3.1 Camada de entrada

Os neurônios, coloridos em roxo, representam os dados de entrada. Estes podem ser tão simples como escalares, ou mais complexos como vetores ou matrizes multidimensionais.

$$x_i = a_i^{(1)}, i \in 1, 2, 3, 4 \tag{3.3.3}$$

Conforme ilustrado na Figura 3, o primeiro conjunto de ativações $(a)^1$ são iguais aos valores de entrada x. "ativação" é o valor do neurônio após a aplicação de uma função de ativação.

3.3.2 Camadas ocultas

Os valores finais nos neurônios ocultos, coloridos em verde na Figura 3, são calculados usando z^l (entradas ponderadas na camada l) e a^l (ativações na camada l). Para as camadas 2 e 3 as equações são:

- l = 2

$$\begin{aligned} z^{(2)} &= W^{(1)}x + b^{(1)} \\ a^{(2)} &= f\left(z^{(2)}\right) \end{aligned} \tag{3.3.4}$$

- l = 3

$$\begin{aligned} z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= f\left(z^{(3)}\right) \end{aligned} \tag{3.3.5}$$

$W^{(2)}$ e $W^{(3)}$ são os pesos nas camadas 2 e 3, enquanto que $b^{(1)}$ e $b^{(2)}$ são os vieses nessas camadas. As ativações $a^{(3)}$ e $a^{(2)}$ são calculadas usando uma função de ativação f. Normalmente, essa função f é não linear (por exemplo sigmoid, ReLU, tanh) e permite que a rede aprenda padrões complexos em dados.

A seguir, será escolhida a camada 2 e seus parâmetros como exemplo, mas as mesmas operações podem ser aplicadas a qualquer camada da rede. $W^{(1)}$ é uma matriz de pesos de dimensão (n, m) onde n é o número de neurônios de saída (neurônios na próxima camada) e m é o número de neurônios de entrada (neurônios na camada anterior). Neste caso, n = 2 e m = 4, e a matriz $W^{(1)}$ tem os seguintes elementos

$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} & W_{14}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} & W_{24}^{(1)} \end{bmatrix}. \tag{3.3.6}$$

O primeiro número no subscrito de qualquer peso corresponde ao índice do neurônio na próxima camada e o segundo número corresponde ao índice do neurônio na camada anterior.

O vetor x é o vetor de entrada e tem dimensão (m, 1), onde m é o número de neurônios de entrada. Neste caso, m = 4, então $x = (x_1, x_2, x_3, x_4)^T$. Já $b^{(1)}$ é um vetor de vieses de tamanho (n , 1) onde n é o número de neurônios na camada atual. Aqui, n = 2, então $b^{(1)} = ({b_1}^{(1)}, {b_2}^{(1)})^T$.

Substituindo na Eq. 3.3.4, temos que $z^{(2)}$ pode ser calculado como

$$z^{(2)} = \begin{bmatrix} W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + W_{14}^{(1)}x_4 \\ W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + W_{24}^{(1)}x_4 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix}. \tag{3.3.7}$$

Tomando a função de ativação ReLU, as ativações da camada 2 podem ser calculadas segundo Eq. 3.3.4

$$a^{(2)} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} f(z_1^{(2)}) \\ f(z_2^{(2)}) \end{bmatrix} \tag{3.3.8}$$

3.3.3 Camada de saída

A parte final de uma rede neural é a camada de saída, que produz o valor predicado. No exemplo da Figura 3, ela é apresentada como um único neurônio, colorido em azul e avaliada da seguinte forma:

$$s = W^{(3)}a^{(3)} \tag{3.3.9}$$

3.3.4 Propagação direta e avaliação

Em resumo as equações a seguir formam a propagação direta da rede.

$$\begin{aligned}
&x = a^{(1)} \quad \text{Camada de entrada}\\
&z^{(2)} = W^{(1)}x + b^{(1)} \quad \text{Valor do neurônio na Camada 2}\\
&a^{(2)} = f\left(z^{(2)}\right) \quad \text{Valor de ativação na Camada 2}\\
&z^{(3)} = W^{(2)}a^{(2)} + b^{(2)} \quad \text{Valor do neurônio na Camada 3}\\
&a^{(3)} = f\left(z^{(3)}\right) \quad \text{Valor de ativação na Camada 3}\\
&s = W^{(3)}a^{(3)} \quad \text{Camada de saída}
\end{aligned} \tag{3.3.10}$$

O passo final em uma propagação direta é avaliar a saída prevista em relação a uma saída esperada y. A saída y faz parte do conjunto de dados de treinamento (x, y), onde x é a entrada.

A avaliação entre s e y ocorre por meio de uma função de custo. Isso pode ser tão simples quanto EQM (erro quadrático médio) ou mais complexo como entropia cruzada. A função de custo C é denotada da seguinte forma:

$$C = custo(s, y), \tag{3.3.11}$$

onde custo pode ser igual ao EQM, entropia cruzada ou qualquer outra função de custo.

Com base em C, o modelo aprende o quanto deve ajustar seus parâmetros para se aproximar da saída esperada y. Isso acontece usando o algoritmo de retropropagação, que será explicado a seguir.

3.3.5 Retropropagação e os Gradientes

De acordo com Rumelhart et al. ((1986)), a retropropagação: "ajusta repetidamente os pesos das conexões na rede de modo a minimizar uma medida da diferença entre o vetor de saída real da rede e o vetor de saída desejado".

Em outras palavras, a retropropagação visa minimizar a função de custo, ajus-

tando os pesos e tendências da rede. O nível de ajuste é determinado pelos gradientes da função de custo em relação a esses parâmetros.

O gradiente de uma função $C(x_1, x_2, \ldots, x_m)$ no ponto x é o vetor das derivadas parciais de C em x,

$$\frac{\partial C}{\partial x} = \left[\frac{\partial C}{\partial x_1}, \frac{\partial C}{\partial x_2}, \ldots, \frac{\partial C}{\partial x_m}\right]. \tag{3.3.12}$$

A derivada de uma função C mede a sensibilidade à mudança do valor da função (valor de saída) em relação a uma mudança em seu argumento x (valor de entrada). Em outras palavras, a derivada nos aponta a direção que C está indo e a declividade associada.

Sendo assim, o gradiente mostra o quanto o parâmetro x precisa mudar (no sentido positivo ou negativo) para minimizar C. O cálculo desses gradientes acontece usando a técnica chamada regra da cadeia. Para o peso w^l, o gradiente é:

$$\frac{\partial C}{\partial w^l} = \frac{\partial C}{\partial a^{l+1}} \frac{\partial a^{l+1}}{\partial z^{l+1}} \frac{\partial z^{l+1}}{\partial w^l} \tag{3.3.13}$$

Conjunto de equações semelhantes podem ser aplicadas a b^l:

$$\frac{\partial C}{\partial b^l} = \frac{\partial C}{\partial a^{l+1}} \frac{\partial a^{l+1}}{\partial z^{l+1}} \frac{\partial z^{l+1}}{\partial b^l} \tag{3.3.14}$$

Os gradientes permitem otimizar os parâmetros do modelo. A seguir está apresentado o algoritmo de descida do gradiente, ou do inglês, Gradient Descendent,

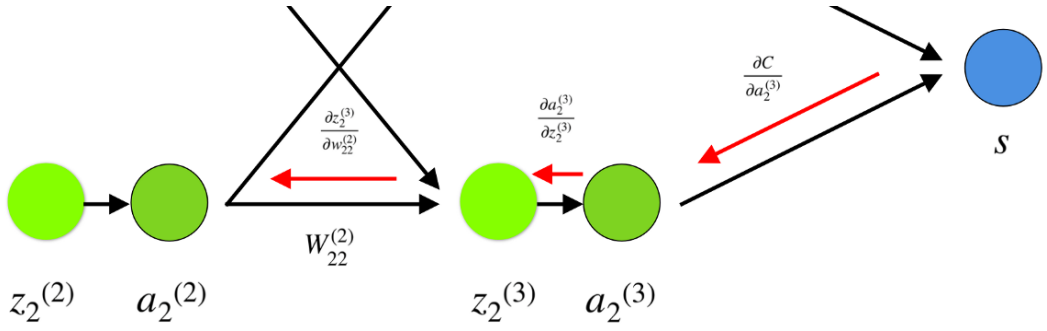
Algoritmo 1: Descida de Gradiente

while *condição de parada não atendida* **do**
  $w^{t+1} := w^t - \epsilon \frac{\partial C}{\partial w^t}$
  $b^{t+1} := b^t - \epsilon \frac{\partial C}{\partial b^t}$
end

w e b são representações matriciais dos pesos e vieses e seus valores iniciais são escolhidos aleatoriamente. Epsilon (ϵ) é a taxa de aprendizagem, a qual determina a influência do gradiente. A derivada de C em w, ou b, pode ser calculada usando derivadas parciais de C nos pesos ou vieses individuais. A condição de terminação é satisfeita quando a função de custo é minimizada, ou seja, estagna em algum mínimo local ou global.

A parte final desta seção mostrará um exemplo simples no qual será calculado o gradiente de C em relação a um único peso $(w_{22})^2$. A Figura 4 ilustra uma ampliação da parte inferior da rede neural da Figura 3:

Figura 4: Representação visual de retropropagação em uma rede neural



Fonte: "https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd"

Pela Figura 4, observa-se que o peso $(w_{22})^2$ conecta $(a_2)^2$ e $(z_2)^3$, portanto, calcular o gradiente requer a aplicação da regra da cadeia por meio de $(z_2)^3$ e $(a_2)^3$:

$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial a_2^{(3)}} \cdot \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \cdot a_2^{(2)} = \frac{\partial C}{\partial a_2^{(3)}} \cdot f'\left(z_2^{(3)}\right) \cdot a_2^{(2)} \tag{3.3.15}$$

Conforme indicado na Eq. 3.3.15 , calcular o valor final da derivada de C em $(a_2)^3$ requer o conhecimento da função C e da derivada da função de ativação f.

3.4 SHAP

A interpretabilidade do modelo de Redes Neurais Profundas sempre foi um fator limitante para casos de uso que exigem explicações das covariáveis envolvidas na modelagem, como é o caso de muitos setores, como por exemplo os Serviços Financeiros. As instituições financeiras, seja por regulamentação ou por escolha, preferem modelos fáceis de interpretar por humanos, por isso os modelos de aprendizado profundo nesses setores tiveram adoções lentas.

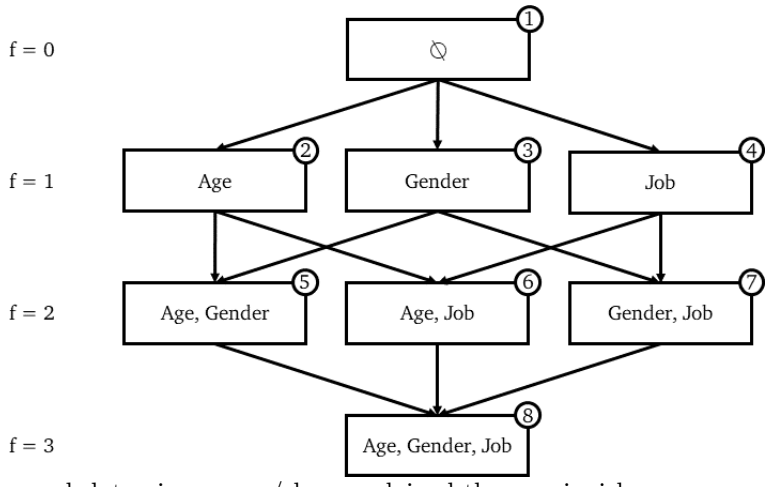
Nesse sentido, há um crescente interesse na criação de técnicas e ferramentas de interpretação e visualização do processo preditivo induzido pelos algoritmos do tipo caixa-preta e, em 2017, o método SHAP (SHapley Additive exPlanations) foi introduzido em Lundberg and Lee ((2017)).

SHAP é um método para explicar previsões individuais. O objetivo do SHAP é explicar a previsão de uma instância, calculando a contribuição de cada variável para a previsão, utilizando os valores de Shapley da teoria dos jogos. Para calcular os valores

de Shapley, simulamos que apenas alguns valores das variáveis estão sendo utilizados ("presentes") e outros não ("ausentes").

A título de exemplo, será utilizado um modelo de aprendizado de máquina que prevê a renda y de uma pessoa sabendo a idade, sexo e trabalho (x_1, x_2, x_3). Os valores de Shapley são baseados na ideia de que o resultado de cada possível combinação (ou coalizão) de variáveis deve ser considerado para determinar a importância de uma única variável x. Neste caso, isso corresponde a cada combinação possível de f características (f indo de 0 a F, sendo F o total de características disponíveis, e neste exemplo $F = 3$). Na matemática, isso é chamado de "conjunto de potências" e pode ser representado como uma árvore, como ilustrado na Figura 5.

Figura 5: Conjunto de potências

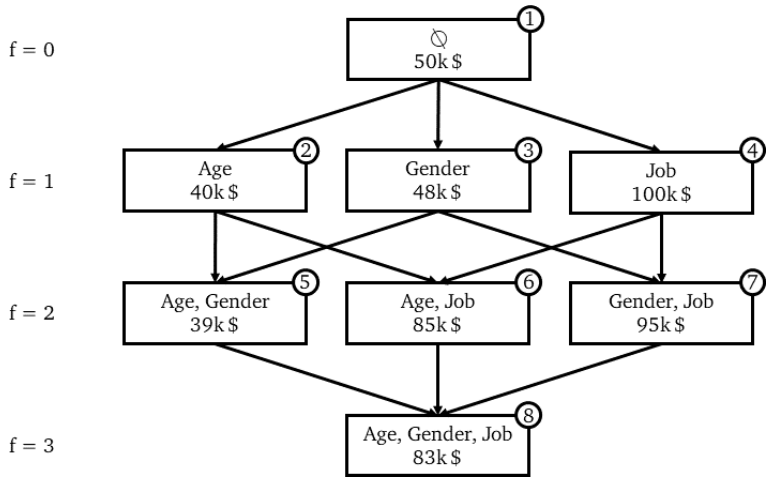


Fonte:"https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30"

Na Figura 5 cada nó representa uma coalizão de covariáveis. Cada aresta representa a inclusão de uma característica não presente na coalizão anterior. Sabe-se, pela matemática, que a cardinalidade de um conjunto de potências é 2^F, onde F é o número de elementos do conjunto original. Neste caso $2^F = 2^3 = 8$ possíveis coalizões de características. Agora, o SHAP requer treinar um modelo preditivo distinto para cada coalizão distinta no conjunto de poder, ou seja, F modelos. Obviamente, esses modelos são completamente equivalentes entre si no que diz respeito aos seus hiperparâmetros e dados de treinamento. O único fator que muda é o conjunto de variáveis incluídas no modelo. Supondo que todos os 8 modelos já foram treinados nos mesmos dados de treinamento, pode-se então tomar uma nova observação x_0 e observar o que os 8 modelos diferentes prevêem para esta mesma observação x_0.

A Figura 6 apresenta as previsões feitas por diferentes modelos para x_0. Em cada nó, a primeira linha relata a coalizão de covariáveis incluídas no modelo, a segunda linha relata a receita prevista para x_0 por esse modelo.

Figura 6



Fonte: "https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30"

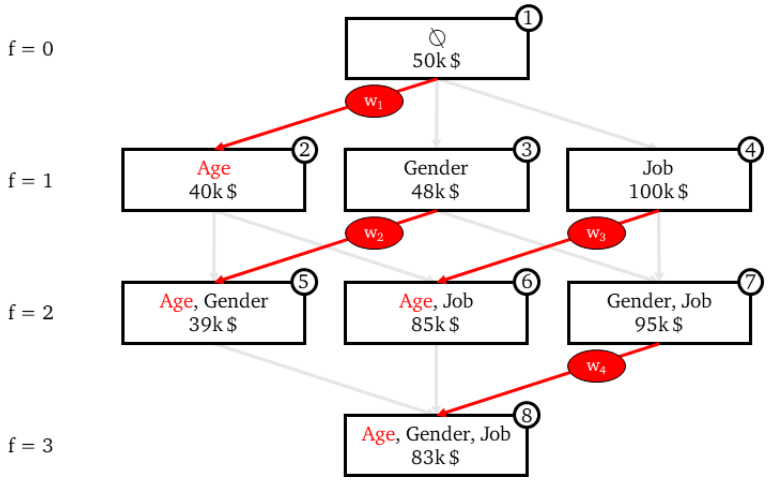
Conforme explicado anteriormente, dois nós conectados por uma aresta diferem por apenas uma covariável, de forma que o nó inferior possui exatamente as mesmas covariáveis do superior mais uma covariável adicional que o superior não possuía. Portanto, a lacuna entre as previsões de dois nós conectados pode ser imputada ao efeito dessa covariável adicional. Isso é chamado de "contribuição marginal" de uma covariável. Portanto, cada aresta representa a contribuição marginal trazida por uma variável para um modelo. Observando o nó 1, que é o modelo sem covariáveis, este modelo irá simplesmente prever o rendimento médio de todas as observações de treinamento (50k $). Observando o nó 2, que é o modelo com apenas uma covariável (Age), a previsão para x_0 agora é de 40k $. Isso significa que saber a idade de x_0 reduziu nossa previsão em 10k $. Assim, a contribuição marginal (CM) trazida pela variável "Age" para o modelo contendo apenas essa variável é de -10k $. Na fórmula:

$$CM_{\text{Age},\{Age\}}(x_0) = Predizer_{\{Age\}}(x_0) - Predizer_{\emptyset}(x_0) = 40k\$ - 50k\$ = -10k\$ \quad (3.4.1)$$

Para obter o efeito global da variável "Age" no modelo final (ou seja, o valor SHAP de "Age" para x_0), é necessário considerar a contribuição marginal de "Age" em todos os modelos onde esta covariável está presente. Na representação em árvore, isso

significa considerar todas as arestas conectando dois nós de forma que: o nó superior não contenha “Age” e o nó inferior contenha . Na Figura 7 a seguir, tais arestas foram destacadas em vermelho.

Figura 7: Contribuições marginais da covariável “Age”



Fonte: “https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30”

Todas essas contribuições marginais ilustradas na Figura 7 são, então, agregadas por meio de uma média ponderada,

$$\begin{aligned} SHAP_{Age}\left(x_0\right) = & w_1 \times CM_{\text{Age},\{Age\}}\left(x_0\right) + \\ & w_2 \times CM_{\text{Age },\{Age,Gender\}}\left(x_0\right) + \\ & w_3 \times CM_{\text{Age },\{Age,Job\}}\left(x_0\right) + \\ & w_4 \times CM_{\text{Age },\{Age,Gender,Job\}}\left(x_0\right) \end{aligned}, \tag{3.4.2}$$

onde $w_1 + w_2 + w_3 + w_4 = 1$. Mais ainda, é necessário que:

- A soma dos pesos de todas as contribuições marginais para modelos de 1 variável deve ser igual à soma dos pesos de todas as contribuições marginais para modelos de 2 variáveis, e assim por diante. Em outras palavras, a soma de todos os pesos na mesma “linha” deve ser igual à soma de todos os pesos em qualquer outra “linha” da árvore. No exemplo utilizado, isso implica que $w_1 = w_2 + w_3 = w_4$.

- Todos os pesos das contribuições marginais para modelos de f variáveis devem ser iguais entre si, para cada f. Em outras palavras, todas as arestas na mesma “linha” da árvore devem ser iguais entre si. No exemplo utilizado, isso significa que $w_2 = w_3$.

Portanto, observando que eles devem somar 1, a solução é:

- $w_1 = 1/3$
- $w_2 = 1/6$
- $w_3 = 1/6$
- $w_4 = 1/3$

Observando a Figura 6, pode-se afirmar que o peso de uma aresta é o inverso do número total de arestas na mesma "linha". Ou, de forma equivalente, o peso de uma contribuição marginal para um modelo com f variáveis é o recíproco do número de possíveis contribuições marginais para todos os modelo com f variáveis.

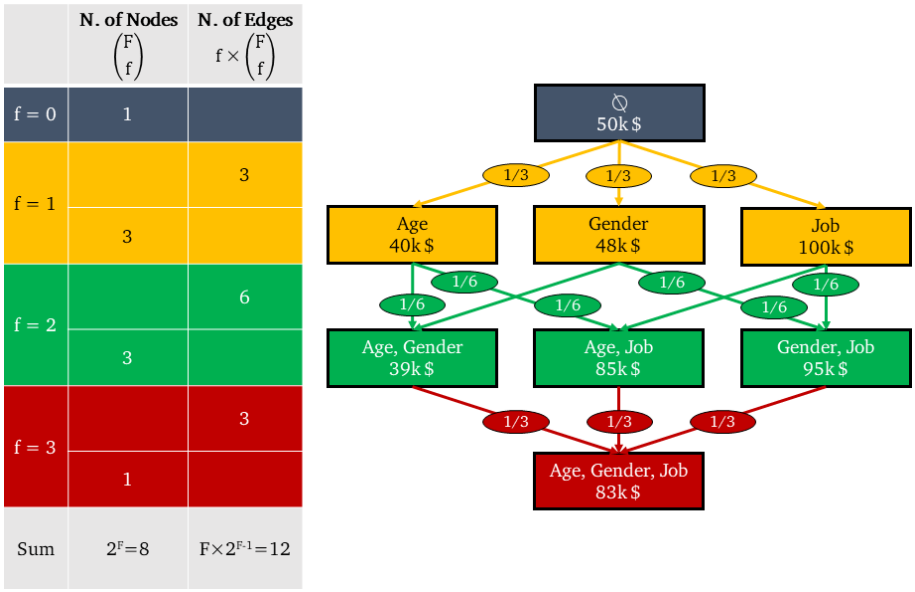
Cada modelo com f variáveis tem f contribuições marginais (uma por variável), então basta contar o número de modelos com f variáveis possíveis e multiplicá-lo por f. Assim, o problema se resume a contar o número de f-modelos de traços possíveis, sabendo que o número total de traços é F. Esta é simplesmente a definição de coeficiente binomial.

Portanto, temos que o número de todas as contribuições marginais de todos os modelo com f variáveis - em outras palavras, o número de arestas em cada linha - é:

$$f \times \binom{F}{f} \tag{3.4.3}$$

Tomando o recíproco da Eq.3.4.3 disso temos o peso de uma contribuição marginal para um modelo com f variáveis, conforme exemplificado na Figura 8:

Figura 8: Obtenção dos pesos a partir do número de arestas



Fonte: "https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30"

Agora, temos todos os elementos necessários para calcular o valor SHAP da variável "Age" para x_0:

$$\begin{aligned}
\text{SHAP}_{Age}(x_0) = &\left[1 \times \binom{3}{1}\right]^{-1} \times CM_{Age,\{Age\}}(x_0) + \\
&\left[2 \times \binom{3}{2}\right]^{-1} \times CM_{Age,\{Age,\ \text{Gender}\ \}}(x_0) + \\
&\left[2 \times \binom{3}{2}\right]^{-1} \times CM_{Age,\{Age,\ \text{Job}\ \}}(x_0) + \\
&\left[3 \times \binom{3}{3}\right]^{-1} \times CM_{Age,\{Age,\ \text{Gender}\ ,\ \text{Job}\ \}}(x_0) + \\
=&\frac{1}{3} \times (-10k\$) + \frac{1}{6} \times (-9k\$) + \frac{1}{6} \times (-15k\$) + \frac{1}{3} \times (-12k\$) \\
=&-11.33k\$
\end{aligned} \tag{3.4.4}$$

Resumidamente, foi construído aqui a fórmula para calcular o valor SHAP de "Age" em um modelo de 3 características. Generalizando para qualquer característica x e qualquer F, obtemos a fórmula relatada no artigo de Slundberg e Lee Lundberg and Lee ((2017)):

$$\text{SHAP}_{\text{var.}}(x) = \sum_{\text{conj.:var.}\in\text{ conj.}} \left[\mid \text{conj.} \mid \times \begin{pmatrix} F \\ \mid \text{conj.} \mid \end{pmatrix} \right]^{-1} \left[\text{ Predizer }_{\text{conj.}}(x) - \text{ Predizer }_{\text{conj.}\setminus\text{var.}}(x) \right] \tag{3.4.5}$$

Aplicada ao exemplo utilizado, a Equação 3.4.5. produz:

- $SHAP_{Age}(x_0) = -11.33k\$$
- $SHAP_{Gender}(x_0) = -2.33k\$$
- $SHAP_{Job}(x_0) = +46.66k\$$

Somando as 3 contribuições, obtemos +33k $, que é exatamente a diferença entre a saída do modelo completo (83k $) e a saída do modelo fictício sem covariáveis (50k $). Esta é uma característica fundamental dos valores de SHAP: somar os valores de SHAP de cada característica de uma dada observação produz a diferença entre a previsão do modelo e o modelo nulo. Esta é realmente a razão de seu nome: SHapley Additive exPlanations.

Como visto acima, a fórmula SHAP original requer treinar F modelos. Para um modelo com apenas 50 covariáveis, isso significaria treinar 10^{15} modelos. À medida que F aumenta, a fórmula vista acima se torna inaplicável. No entanto, bibliotecas como a de Slundberg empregam algumas aproximações que tornam o trabalho viável Shrikumar et al. ((2017)).

4 Resultados

Neste capítulo serão apresentados os resultados do trabalho. Na seção 4.1 será apresentado um exemplo com base em 1 conjunto de dados fictícios com o intuito de ilustrar o método SHAP para a interpretação de redes neurais. Na seção 4.2 serão apresentados estudos de simulação visando comparar os resultados do modelo de regressão Normal e da Rede Neural, com foco na capacidade de interpretação da técnica SHAP.

4.1 Exemplo

Nessa seção as técnicas previamente explicadas foram aplicadas e ilustradas em um conjunto simulado com 1500 observações, contendo 10 covariáveis $X_1, X_2, ..., X_{10}$, todas com distribuição $N(0, 1)$, com seus respectivos Betas, $\beta_1 = 1$, $\beta_2 = 2$, $\beta_3 = 3$, $\beta_4 = 4$, $\beta_5 = 5$ e $\beta_6 = ... = \beta_{10} = 0$. Além disso, foi acrescido um termo de erro com distribuição Normal, $\varepsilon_i \sim N(0, 1)$. A variável resposta Y é formada pela combinação linear entre as covariáveis e seus respectivos Betas mais as interações 1 a 1 entre as covariáveis, somadas ao erro. Sua função pode ser escrita por

$$y = x_1\beta_1 + ... + x_5\beta 5 + x_1x_1 + x_1x_2 + ... + x_4x_5 + x_5x_5 + \varepsilon.$$

Esse conjunto de dados foi dividido em 70% para treinamento e 30% para teste, e com os dados de treinamento foi ajustado um modelo de regressão linear múltipla e uma rede neural artificial com 5 camadas ocultas com 400 neurônios cada. O modelo de regressão ajustado não incluiu nenhum termo de interação, apenas as 10 covariáveis originais. Ignorar as interações é muito comum no ajuste real de modelos com muitas covariáveis, e por isso o ajuste aqui foi feito dessa maneira para avaliar o impacto dessa formulação incompleta.

Após o ajuste da regressão, obtivemos valores de betas estimados presentes na Tabela 1 a seguir

Tabela 1: Valores dos Betas Estimados e os Betas reais para cada covariável

Variáveis	**Betas Estimados**	**Betas reais**
Var 1	0.787	1
Var 2	1.787	2
Var 3	2.756	3
Var 4	3.821	4
Var 5	4.601	5
Var 6	0.200	0
Var 7	-0.068	0
Var 8	0.198	0
Var 9	0.061	0
Var 10	0.204	0

Assim, utilizando o conjunto de teste para fazer a previsão, chegamos a um erro quadrático médio (EQM) de 20.875.

Para a rede neural não é possível calcular os betas como é realizado nos modelos probabilísticos e por isso utilizamos a técnica SHAP para mensurar o peso das variáveis na previsão do modelo, e assim obtivemos os valores de SHAP presentes na tabela a seguir

Tabela 2: Valores de SHAP e os Betas reais para cada variável

Variáveis	**Valores de SHAP**	**Betas reais**
Var 1	1.355	1
Var 2	1.765	2
Var 3	2.497	3
Var 4	3.161	4
Var 5	4.078	5
Var 6	0.126	0
Var 7	0.157	0
Var 8	0.109	0
Var 9	0.108	0
Var 10	0.114	0

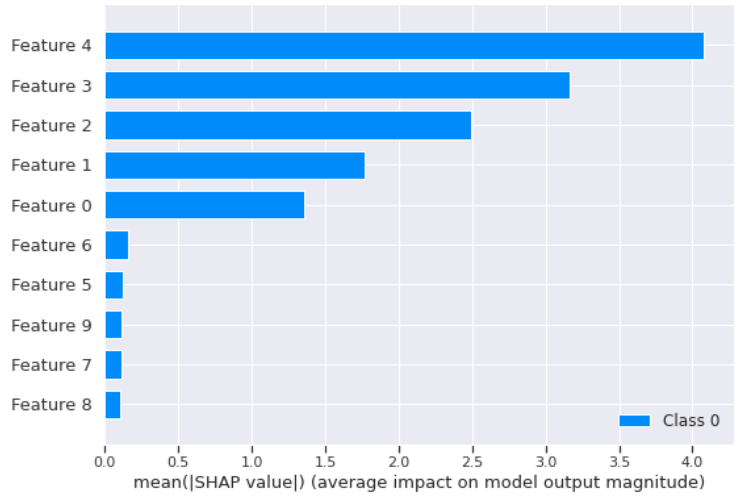
Para a Rede Neural, o EQM foi de 2.225. Nesse sentido, analisando os EQMs, é possivel perceber que a rede neural obteve um poder preditivo bem superior comparado ao modelo linear e isso se deve a capacidade da rede neural em compreender as não linearidades (interações) dos dados utilizados. Observando os coeficientes betas e os valores de SHAP presentes nas tabelas 1 e 2, percebe-se que ambas as técnicas foram capazes

de captar o peso das variáveis 1 a 5 e a irrelevância das variáveis 6 a 10, atribuindo-as valores próximos a 0 para estas últimas.

Como dito anteriormente, uma das vantagens do SHAP são seus recursos gráficos implementados na linguagem Python. Nesse sentido, foram gerados alguns gráficos que nos permitem entender as previsões dos modelos de maneira tanto global como pontual.

A Figura 9 a seguir nos mostra por meio de um gráfico de barras a importância média de cada variável nas previsões dos modelos, de maneira ordenada. Por meio desse gráfico, é possível ter um panorama geral de como determinado modelo chega em suas conclusões a partir do conjunto de treinamento e das covariáveis (features).

Figura 9: Contribuições marginais das covariáveis (features) do modelo



Basicamente, como o próprio título do eixo X demonstra, cada barra representa a média dos valores SHAP, em módulo, assim, é avaliada a contribuição média das covariáveis nas respostas do modelo. Considerando que o feature 0 é a covariável X_1 e o feature 9 é a covariável X_{10}, conclui-se que as covariáveis $X_6, ...X_{10}$ tem pouca relevância no modelo, enquanto que as covariáveis $X_1, ..., X_5$ tem relevância crescente.

Outra maneira de ter um panorama global do modelo de Rede Neural é por meio do gráfico de dependência, o qual é um gráfico de dispersão que mostra o efeito que uma única variável tem nas previsões feitas pelo modelo e está representado na Figura 10 a seguir.

Figura 10: Gráfico de dependência da Covariável X_2

Cada ponto é referente a uma única previsão (observação) do conjunto de dados. O eixo x é o valor da variável. O eixo y é o valor SHAP para essa variável, que representa o quanto o valor dessa covariável altera a saída do modelo. A cor corresponde a uma segunda variável que pode ter um efeito de interação com a variável que estamos utilizando (por padrão esta segunda variável é escolhida automaticamente). Se um efeito de interação estiver presente entre esta outra variável e a variável que estamos utilizando, ele aparecerá como um padrão vertical distinto de coloração. Para o exemplo acima, quando a covariável X_2 apresenta valores abaixo de 0, seu impacto nos dados apresenta grande variabilidade, que vai diminuindo quando X_2 se aproxima de 0. Para valores positivos da variável, o impacto no modelo cresce de maneira aproximadamente linear. Conforme exemplificado pelo SHAP, a rede neural permite contribuições não lineares da covariável na variável resposta, sendo um modelo mais flexível que a regressão convencional. Além disso, observa-se que o valor SHAP da covariável X2 (1,765) é dado pela média dos valores SHAP individuais, em módulo. O fato das cores estarem de certa maneira misturadas, sugere que não existe uma grande interação entre as covariáveis X_2 e X_5.

A Figura 11 a seguir exemplifica um pouco da explicabilidade local do modelo, ou seja, o quanto e como cada variável influenciou em uma previsão específica. Esse gráfico é conhecido como gráfico de força.

Figura 11: Gráfico de força da previsão da terceira observação

Para entender a Figura 11, é preciso explicar alguns fatores:

- O valor f(x) da saída é a previsão da Rede Neural para essa observação, nesse caso 3.38.
- O valor base (base value) é o valor que seria previsto se não fosse conhecida nenhuma variável para a saída atual. Em outras palavras, é a previsão média. Você pode se perguntar por que é 5.406? Isso ocorre porque a previsão média do Y de treinamento é 5.406.
- As covariáveis que empurram a previsão para cima (à direita) são mostrados em vermelho, e aqueles que empurram a previsão para baixo estão em azul.
- A variável 4 (ou seja, X_5), tem um impacto positivo na previsão do modelo (barra vermelha), e o tamanho da barra mostra a intensidade desse impacto.
- O valor descrito abaixo das barras mostra o valor da covariável para essa observação em específico.

Portanto, os gráficos 9, 10 e 11 auxiliam na interpretação dos valores SHAP, ilustrando a contribuição global da covariável no modelo (Figuras 9 e 10) ou a contribuição local da covariável nas observações (Figura 11).

4.2 Simulações

A fim de comparar a regressão linear múltipla e os modelos de redes neurais, e avaliar a capacidade de interpretação das redes utilizando os valores de SHAP, alguns cenários de simulação serão utilizados. Foram simulados 4 cenários e todos com a seguinte configuração: 10 covariáveis $X_1, X_2, ..., X_{10}$, todas com distribuição $N(0, 1)$, com seus respectivos Betas, $\beta_1 = 1$, $\beta_2 = 2$, $\beta_3 = 3$, $\beta_4 = 4$, $\beta_5 = 5$ e $\beta_6 = ... = \beta_{10} = 0$, e um termo de erro com distribuição Normal, $\varepsilon_i \sim N(0, 1)$.

O cenário 1 possui 150 observações e a variável resposta Y é formada pela combinação linear entre as covariáveis e seus respectivos Betas somada ao erro, sua função pode ser escrita por

$$y = x_1\beta_1 + x_2\beta_2 + ... + x_5\beta_5 + \varepsilon.$$

O cenário 2 apresenta forma funcional idêntica ao cenário 1, mas contém 1500 observações.

O cenário 3 possui 150 observações e a variável resposta Y é formada pela combinação linear entre as covariáveis e seus respectivos Betas mais as interações 1 a 1 entre as covariáveis, somadas ao erro, sua função pode ser escrita por

$$y = x_1\beta_1 + ... + x_5\beta_5 + x_1x_1 + x_1x_2 + ... + x_4x_5 + x_5x_5 + \varepsilon.$$

O cenário 4 possui 1500 observações e e tem forma idêntica ao cenário 3.

Para cada cenário foram gerados 1000 conjuntos de dados seguindo as funções descritas acima. Cada conjunto de dados foi dividido em 70% para treinamento e 30% para teste, e com os dados de treinamento foi ajustado um modelo de regressão linear múltipla e uma rede neural artificial com 5 camadas ocultas com 400 neurônios cada.

Com os modelos ajustados, obtivemos os betas estimados e calculamos os valores de SHAP para cada variável e utilizando os dados de treinamento, foi calculado o EQM para cada modelo.

4.2.1 Resultado das simulações

Nesta seção apresentaremos os resultados das simulações explicadas acima. A realização das simulações resultou em 1000 ajustes correspondentes a cada conjunto de dados, com os Betas estimados, os valores de SHAP e o EQM de cada modelo ajustado.

Com o intuito de interpretar esses modelos, analisaremos os Betas estimados pelo modelo probabilístico de Regressão Linear Múltipla e os valores de SHAP obtidos do modelo de Redes Neurais Artificiais. Assim, será possível identificar a relevância que cada modelo atribuiu para as variáveis em diferentes cenários e compará-los entre si e com a real relevância de cada variável.

As Tabelas 3, 4, 5 e 6 a seguir contém os valores médios dos Betas estimados pela Regressão Linear Múltipla e os valores médios do SHAP obtidos da Rede Neural

Artifical dos 1000 conjuntos de dados. As tabelas também apresentam os Betas reais das covariáveris como referência.

Tabela 3: Valores Médios dos Betas Estimados, dos Valores de SHAP e os Betas reais para o Cenário 1

Variáveis	**Betas Estimados**	**Valores SHAP**	**Betas reais**
Var 1	0.991	0.772	1
Var 2	1.999	1.560	2
Var 3	3.004	2.329	3
Var 4	3.997	3.127	4
Var 5	4.992	3.880	5
Var 6	0.000	0.188	0
Var 7	0.003	0.185	0
Var 8	0.000	0.186	0
Var 9	0.004	0.188	0
Var 10	0.001	0.186	0

Tabela 4: Valores Médios dos Betas Estimados, dos Valores de SHAP e os Betas reais para o Cenário 2

Variáveis	**Betas Estimados**	**Valores SHAP**	**Betas reais**
Var 1	0.999	0.799	1
Var 2	2.000	1.605	2
Var 3	2.999	2.407	3
Var 4	4.000	3.212	4
Var 5	5.001	4.007	5
Var 6	0.000	0.111	0
Var 7	0.000	0.109	0
Var 8	0.000	0.108	0
Var 9	0.000	0.109	0
Var 10	0.000	0.111	0

Tabela 5: Valores Médios dos Betas Estimados, dos Valores de SHAP e os Betas reais para o Cenário 3

Variáveis	**Betas Estimados**	**Valores SHAP**	**Betas reais**
Var 1	0.996	1.092	1
Var 2	2.014	1.598	2
Var 3	3.021	2.289	3
Var 4	3.999	3.021	4
Var 5	5.022	3.748	5
Var 6	0.006	0.268	0
Var 7	0.004	0.257	0
Var 8	0.003	0.268	0
Var 9	0.005	0.265	0
Var 10	0.000	0.264	0

Tabela 6: Valores Médios dos Betas Estimados, dos Valores de SHAP e os Betas reais para o Cenário 4

Variáveis	**Betas Estimados**	**Valores SHAP**	**Betas reais**
Var 1	1.008	1.497	1
Var 2	2.009	1.823	2
Var 3	3.007	2.525	3
Var 4	4.000	3.245	4
Var 5	5.012	3.992	5
Var 6	0.005	0.130	0
Var 7	0.000	0.132	0
Var 8	0.003	0.130	0
Var 9	0.003	0.130	0
Var 10	0.002	0.131	0

Analisando as tabelas 3 a 6, pôde-se perceber que, em todos os cenários, tanto os modelos lineares quanto as redes neurais foram capazes de identificar a relevância de cada variável. Ou seja, quando comparados os valores estimados com os betas reais que foram utilizados para gerar os conjuntos de dados, os valores são próximos.

Buscando entender o poder preditivo dessas duas diferentes abordagens de modelagem, iremos analisar e comparar o EQM de cada modelo quando aplicado aos diferentes cenários escolhidos.

A Tabela 7 a seguir apresenta os valores médios do EQM do modelo de Regressão Linear Múltipla e da Rede Neural Artificial com base nos 1000 dados simulados de cada cenário.

Tabela 7: Média dos 1000 EQM´s do modelo de Regressão Linear Múltipla e da Rede Neural Artificial, para cada Cenário

Cenário	**EQM Regressão Linear**	**EQM Rede Neural**
Cenário 1	1.113	2.906
Cenário 2	1.009	1.515
Cenário 3	27.393	10.079
Cenário 4	23.242	2.662

De acordo com a Tabela 7, para os cenários 1 e 2, os quais foram gerados por funções mais simples, ou seja, sem as interações entre as covariáveis, apesar da Rede Neural ter alcançado um desempenho satisfatório, a Regressão Linear obteve um melhor poder preditivo, apresentando valores de EQM mais baixos, inclusive para o Cenário 2, o qual apresenta uma quantidade maior de observações.

Para os cenários 3 e 4, os quais possuem funções mais complexas, ou seja, além da combinação linear entre as covariáveis e os betas, também possui a interação 1 a 1 entre as covariáveis, é possível perceber que, ao contrário dos outros dois cenários, o modelo linear não obteve um bom desempenho e o poder preditivo das Redes Neurais se destaca, principalmente no Cenário 4, que apresenta uma maior quantidade de observações. Isso se deve ao fato de que os modelos lineares não são capazes de predizer de forma acurada dados mais complexos, enquanto que a Rede Neural consegue aprender padrões não lineares com maior facilidade.

Por fim, analisaremos o tempo computacional gasto por uma GPU para o ajuste desses modelos. Assim, será possível entender a viabilidade de utilizar determinado modelo dependendo no cenário.

A Tabela 8 a seguir apresenta o tempo computacional (em segundos) para a realização do ajuste dos modelo de Regressão Linear Múltipla e da Rede Neural Artificial, para cada cenário.

Tabela 8: Tempo computacional (em segundos) para a realização do ajuste dos modelo de Regressão Linear Múltipla e da Rede Neural Artificial, para cada cenário

Cenário	**Regressão Linear**	**Rede Neural**
Cenário 1	0.023	4.049
Cenário 2	0.006	6.123
Cenário 3	0.009	1.998
Cenário 4	0.010	4.839

Por meio da análise da Tabela 8, percebe-se que, para todos os cenários, o ajuste de uma Rede Neural consome muito mais tempo computacional comparado ao modelo linear. Mesmo utilizando uma GPU, a Rede Neural demorou cerca de 475 vezes mais do que a Regressão Linear, em média.

5 Conclusão

O presente trabalho comparou modelos de Regressão Linear Múltipla com as Redes Neurais Artificiais e buscou explorar potenciais técnicas para a interpretação dos modelos conhecidos como "caixa-preta". Nesse sentido, observa-se que, para dados mais complexos, a Rede Neural com a configuração escolhida obteve uma maior capacidade de predição comparado à Regressão Linear Múltipla, principalmente quando foi utilizado uma maior quantidade de dados para o treinamento dos modelos. Entretanto, para dados menos complexos, ou seja, gerados por funções lineares, a Regressão se mostrou superior. Vale ressaltar que, apesar do bom desempenho das Redes Neurais, o tempo computacional utilizado para ajustar esses modelos é significativamente maior quando comparado à Regressão Linear, portanto, é um fator que precisa ser levado em consideração ao escolher um dos modelos.

Tendo em vista o bom desempenho das Redes Neurais Artificiais, interpretar estes modelos é de suma importância. Assim, com a técnica de dos valores SHAP foi possível esclarecer as predições dos algoritmos "caixa-preta" e ter um ganho significativo na interpretabilidade desses modelos, tornando possível identificar as covariáveis mais importantes para os modelos de aprendizado de máquina.

Referências

R. Beale and T. Jackson. *Neural Computing-an introduction*. CRC Press, 1990.

S. Haykin and N. Network. A comprehensive foundation. *Neural networks*, 2(2004):41, 2004.

O. Helene. *Metodos dos Minimos Quadrados*. Editora Livraria da Física, 2006.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Y. Meng, N. Yang, Z. Qian, and G. Zhang. What makes an online review more helpful: An interpretation framework using xgboost and shap values. *Journal of Theoretical and Applied Electronic Commerce Research*, 16(3):466–490, Nov 2020. ISSN 0718-1876. doi: 10.3390/jtaer16030029. URL `http://dx.doi.org/10.3390/jtaer16030029`.

J. Neter, M. H. Kutner, C. J. Nachtsheim, W. Wasserman, et al. Applied linear statistical models. 1996.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

Anexo

A Notebook com o desenvolvimento do trabalho

https://colab.research.google.com/drive/1e3wraTNpQzePwUVLBO4khAQrXj1JE6aE?usp=sharing