Universidade Federal de São Carlos
Centro de Ciências Exatas e de tecnologia
Departamento de Estatística

Utilização de métodos de Aprendizado de Máquina para estimação de escores de propensão

Amanda Kely Faria dos Santos

Trabalho de Conclusão de Curso

Amanda Kely Faria dos Santos

Utilização de métodos de Aprendizado de Máquina para estimação de escores de propensão

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Amanda Kely Faria dos Santos e aprovado pela banca examinadora.

São Carlos, 5 de Maio de 2022

Banca Examinadora:

- Profa. Dra. Teresa Cristina Martins Dias (orientadora)
- Prof. Dr. Afrânio Márcio Corrêa Vieira
- Prof. Dr. Danilo Louenço Lopes

Dedicatória

Dedico este trabalho aos meus pais, Elaine e Rodrigo e aos meus irmãos, Carina e Felipe, que sempre acreditaram em mim e me proporcionaram todo o suporte necessário para a realização dessa conquista, nada teria sentido sem vocês.

Agradecimentos

Agradeço a minha orientadora, Teresa Cristina Martins Dias, pela paciência, apoio, parceria e aprendizados durante esse período.

Agradeço as minhas amigas Thaina, Ana, Bruna e Tatiana pela parceria e amizade durante todos esses anos, tudo seria mais difícil sem vocês.

Resumo

Bases de dados cada vez maiores e mais complexas podem ser facilmente obtidas e tecnologias apropriadas para modelagem de quantidades massivas de dados tornam-se cada vez mais necessárias, a fim de que seja possível otimizar resultados e predições. Técnicas de aprendizado de máquina estão tendo destaque em diversas áreas e uma dessas é na análise de escores de propensão. Neste trabalho o objetivo é apresentar e comparar técnicas de aprendizado de máquina. Mais especificamente, o objetivo é comparar as metodologias árvore de classificação e redes neurais com regressão logística, técnica bastante utilizada na análise de escores de propensão. Também, conhecer quais benefícios dessas técnicas de aprendizado de máquina agregam mais, em detrimento de modelos obtidos via regressão logística, na estimação de escores de propensão. Para atingir o objetivo, algumas aplicacoes sao apresentadas e discutidas. O procedimento de analise foi desenvolvido com funções já implementadas nas bibliotecas em Python, linguagem que cada vez mais se destaca no mercado de trabalho.

Palavras-chave: *árvore de classificação, aprendizado de máquina, perceptron, redes neurais, scores de propensão.*

Abstract

Increasingly larger and more complex databases can be easily obtained and appropriate technologies for modeling massive amounts of data become increasingly necessary in order to optimize results and predictions. Machine learning techniques are gaining prominence in several areas and one of these is the analysis of propensity scores. In this work, the objective is to present and compare machine learning techniques. More specifically, the objective is to compare the classification tree and neural networks methodologies with logistic regression, a technique widely used in the analysis of propensity scores. Also, to know which benefits these machine learning techniques add more, to the detriment of models obtained via logistic regression, in the estimation of propensity scores. To achieve the objective, some applications are presented and discussed. The analysis procedure was developed with functions already implemented in Python libraries.

Sumário

Lista de Tabelas iii

Lista de Figuras v

1 Introdução 1

2 Métodos de Classificação 3

2.1 Árvore de Classificação . . . 3

2.2 Entropia . . . 5

2.3 Índice de Gini . . . 10

2.4 Redes Neurais . . . 11

2.5 Modelo perceptron simples . . . 12

2.5.1 Redes Perceptron Multicamadas . . . 14

2.5.2 Função de ligação . . . 18

2.6 Regressão Logística . . . 18

2.6.1 Função de Máxima Verossimilhança . . . 21

2.7 Métricas de Classificação . . . 23

3 Aplicação das Metodologias 25

3.1 Exemplo 1: Conjunto de Dados Simulados . . . 25

3.1.1 Aspectos da simulação . . . 25

3.2 Exemplo 2: Banco Português . . . 33

4 Conclusão e Discussão 45

5. Referências Bibliográficas 49

A Códigos utilizados no projeto 53

Lista de Tabelas

2.1 Dados do Exemplo. 7
2.2 Frequência da variável Escolaridade, segundo a Renda. 7
2.3 Estrutura da matriz de confusão. 23

3.1 Variáveis utilizadas na base de dados simulados. 27
3.2 Medidas resumo para as variáveis quantitativas. 28
3.3 Estimativas obtidas via Regressão logística. 29
3.4 Matriz de confusão para o modelo de Regressão Logística. 30
3.5 Desempenho preditivo do modelo de regressão logística. 31
3.6 Matriz de confusão para o modelo de Redes Neurais. 31
3.7 Desempenho preditivo do modelo de Redes Neurais. 32
3.8 Matriz de confusão para o modelo de Redes Neurais. 32
3.9 Desempenho preditivo do modelo de Árvore de Classificação. 33
3.10 Variáveis base do Banco Português. 33
3.11 Medidas resumo das variáveis Idade, Balanço médio anual e Duração do Contato. 34
3.12 Proporção das categorias da variável Profissão. 35
3.13 Matriz de confusão para o modelo de Redes Neurais. 39
3.14 Desempenho preditivo do modelo de Redes Neurais. 40
3.15 Matriz de confusão para o modelo de Árvore de Classificação. 40
3.16 Desempenho preditivo do modelo de Árvore de Classificação. 41
3.17 Coeficientes do modelo gerado com o conjunto de treino. 42
3.18 Matriz de confusão para o modelo de Regressão Logística. 43
3.19 Desempenho preditivo do modelo de Regressão Logística. 43

Lista de Figuras

2.1 Ilustração de uma árvore de decisão simples. 4
2.2 Exemplo de árvore de decisão. 10
2.3 Exemplo de uma rede neural artificial. 11
2.4 Exempo de um neurônio biológico. 11
2.5 Modelo computacional de um neurônio. 13
2.6 Arquitetura RNA perceptron multicamadas. 15
2.7 Exemplo de uma Rede MLP. 17

3.1 Matriz de correlação das variáveis Idade, Balanço médio anual e Duração do Contato. 35
3.2 Gráfico de barras davariável estado civil. 36
3.3 Gráfico de barras da variável escolaridade. 36
3.4 Gráfico de barras da variável empréstimo em atraso. 37
3.5 Gráfico de barras da variável financiamento imobiliário. 37
3.6 Gráfico de barras da variável empréstimo pessoal. 38
3.7 Gráfico de barras da variável comunicação da campanha. 38
3.8 Gráfico de barras da variável resultado da última campanha. 39

Capítulo 1

Introdução

Estamos vivendo "era da informação", o que é resultado da quantidade e da facilidade em receber ou obter informação, no dia a dia, vindas de todos os lados. Informações essas que de nada valem se não bem organizadas e exploradas. O Instituto Garter estimou que chegaríamos a 40 trilhões de *gigabytes* de dados no mundo ao final de 2020 (Exame, 2019), mas na prática, isso não nos favorece, uma vez que dados brutos não nos trazem informação. Porém, quando tratados de forma correta, os dados podem ser de grande valia para entender o passado e, principalmente, como forma de predição.

Para explorá-los da melhor forma possível e conseguir extrair informações relevantes, são necessárias técnicas mais refinadas que auxiliem na extração dessas informações, lembrando sempre da grande quantidade de dados. Para isso, existem muitas técnicas de aprendizado de máquina, onde a ideia é fazer com que a máquina "aprenda" por experiência e não necessite de programação sequencial.

O uso de escores de propensão teve um aumento exponencial no decorrer dos últimos anos em análises estatísticas. Escore de propensão é definido como a probabilidade (ou propensão) de um indivíduo da amostra ser alocado em determinado grupo, condicional às covariáveis observadas (informações cadastrais) (Souza, 2010). Para estimação desses escores, uma técnica muito utilizada é a regressão logística, pela fácil interpretação dos parâmetros do modelo ajustado. Souza (2010) ajusta um modelo de regressão logística para estimar os escores de propensão, na área epidemiológica.

Porém, outras técnicas podem ser exploradas. Mas é importante destacar as diversas vantagens na utilização de aprendizado de máquina. Sob esta perspectiva, neste projeto são apresentadas as técnicas de redes neurais e árvore de classificação, como alternativas à regressão logística, na estimação de escores de propensão.

Como motivação para o trabalho, é utilizado o artigo *Propensity score estimation: machine learning and classification methods as alternatives to logistic regression* (Westreich *et al.*, 2010), utilizando como base para discussão os métodos árvore de decisão e redes neurais.

Este Trabalho de Conlusão de Curso contém 3 capítulos, além deste (Capítulo 1). No Capítulo 2 são apresentadas as metodologias árvore de decisão, redes neurais e regressão logística e no Capítulo 3 são mostradas duas aplicações das metodologias usando um conjunto de dados simulados e um conjunto de dados de um repositório. Por fim, no Capítulo 4 os resultados das aplicações, bem como uma discussão dos resultados, são apresentados.

Capítulo 2

Métodos de Classificação

Neste capítulo são apresentadas as metodologias necessárias para a criação dos modelos de Árvore de Classificação, Redes Neurais e Regressão Logística e as métricas definidas para validar a performance destes modelos.

2.1 Árvore de Classificação

Árvores de classificação são algoritmos de aprendizado de máquina que podem ser utilizados para identificar, explorar e classificar conjuntos de dados, inclusive quando estes apresentam estrutura complexa (McLachlan, 2004). Esta técnica pode ser utilizada em diversas situações que envolvem tomada de decisão.

Algumas vantagens na utilização deste método são: possibilidade de decompor um problema complexo em sub-problemas, com um algoritmo flexível, já que funciona com variáveis qualitativas e quantitativas: utilização como base de funcionamento de outros algoritmos (não sendo este o foco neste trabalho); apresenta uma estrutura simples que segue uma relação hierárquica dos dados. Porém, para utilizá-lo é necessário um tamanho amostral razoável para se obter bons resultados.

A Figura 2.1 ilustra uma ávore de classificação simples, para auxiliar no entendimento do funcionamento do algoritmo.

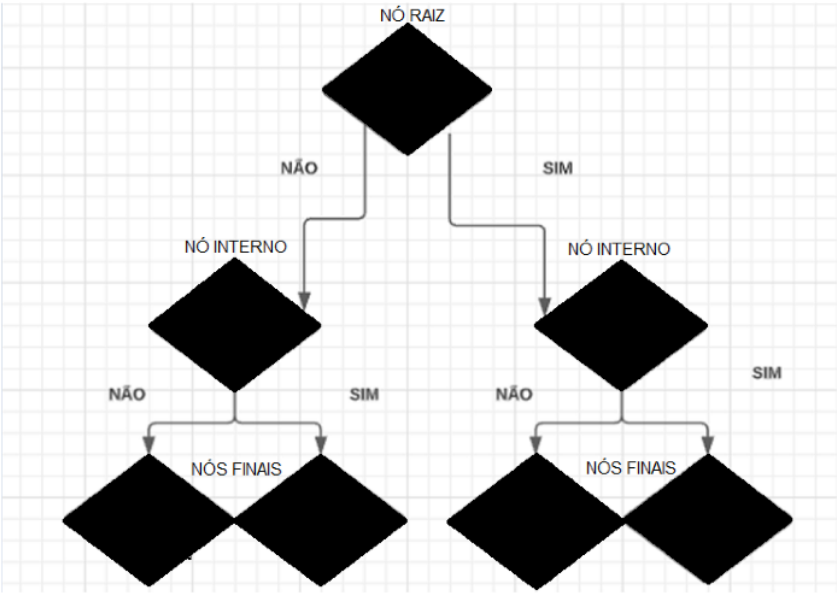

Figura 2.1: Ilustração de uma árvore de decisão simples.

Fonte: Ilustrado pela autora.

É possível observar (Figura 2.1) que a árvore de classificação vai separando as observações em grupos cada vez mais homogêneos em relação ao desfecho de interesse, começando por um nó raiz e passando para novos nós tomando decisões até chegar no que chamamos de folha que é a decisão final desse resultado. Cada nó normalmente tem duas decisões, como é apresentado na figura mas existem algoritmos que utilizam mais.

Basicamente, o algoritmo de árvore vai procurando decisões nessas variáveis preditoras afim de melhorar essa homogeneidade de um lado e de outro. Se existir alguma variável que não é importante no modelo, a árvore simplesmente não a utiliza em suas ramificações, sendo esta outra grande vantagem na utilização deste método. É importante que a profundidade da árvore seja controlada na construção do algoritmo para que não aconteça *overfitting* sendo essa uma desvantagem do algoritmo, além do baixo poder preditivo (James *et al.*, 2013).

Um dos algoritmos mais conhecidos e completos de árvore de classificação é o CART (do inglês *Classification and Regression Tree*), proposto por Breiman *et al.* (1984), também conhecida como partição recursiva binária, pois os nós são sempre divididos exatamente em dois outros nós. O processo pode ser repetido, tratando cada próximo nó como ponto de partida para duas novas repartições(processo recursivo). As principais características do CART são: definir o conjunto de regras para dividir cada nó da árvore, decidir quando a árvore está completa e associar cada nó terminal a uma categoria. Para dividir um

nó em outros dois nós, o algoritmo "usa" perguntas cujas respostas são binárias (sim ou não). Por exemplo, você já foi inadimplente anteriormente? ou qual sua idade? (menor do que 40 anos ou maior ou igual a 40 anos).

A regra para divisão dos dados pode ser baseada, por exemplo, no critério de qualidade da divisão. O critério mais utilizado para classificação é o índice de Gini, baseado na entropia. A construção da árvore começa com a variável que maxima a pureza do conjunto de dados e minimiza a entropia.

Para a construção da árvore, utlizamos entropia e o índice Gini, descritos nas seções 2.2 e 2.3.

2.2 Entropia

O objetivo na utilização de árvore de decisão é dividir o conjunto de dados em subconjuntos, tal que estes fiquem mais puros. Um subconjunto se torna mais puro à medida em que contém menos classes (ou apenas uma) da variável de interesse.

A entropia é uma medida da impureza dos dados. É comum dizer que é uma medida da desordem ou incerteza do conjunto de dados. Denotando a variável de interesse por X, a entropia é definida por

$$E(X) = -\sum_{i=1}^{d} p_i \times \log_2 p_i,$$

em que d é o número de classes na variável em estudo e p_i é a probabilidade de cada classe ocorrer, dado que a variável assume a classe i.

Com base nesta medida, o algoritmo verifica como os dados estão distribuídos nas variáveis preditoras em relação à variável resposta. A unidade de medida da entropia é o *bit* (menor parcela de informação processada por um computador), e quanto maior, significa maior desordem entre os dados, observando de acordo com a variável resposta. Utilizando a entropia, o algoritmo calcula o ganho de informação de cada uma das variáveis preditoras e aquela com maior ganho de informação comporá o nó raiz da árvore.

Usando a entropia (2.1), o ganho de informação, denotado por (GI), é dado por,

$$GI(X) = E(\text{Classe}) - E(X),$$

em que

- $E(\text{Classe})$: entropia das classes da variável resposta do conjunto de dados e
- $E(X)$: entropia da variável que, neste caso, é a soma ponderada das entropias de suas partições.

Observe que para calcular o ganho de informação de cada conjunto de dados é necessário calcular a entropia da variável resposta dos dados, de cada variável e por fim o ganho de informação de cada variável. Para a construção da árvore de classificação, fazemos:

Algoritmo base

Passo 1: Escolher a variável independente por meio dos cálculos de entropia e ganho de informação;

Passo 2: Estender a árvore adicionando um ramo para cada valor que a variável escolhida assume;

Passo 3: Passamos cada valor para as folhas (levando em consideração o valor da variável escolhida);

Passo 4: Para cada folha:

- se todas as observações são da mesma classe da variável resposta, associar esta classe à folha.
- senão, repetir os passos anteriores.

Quando as variáveis do conjunto de dados não são binárias, é necessário calcular o ganho de informação das variáveis envolvidas, a fim de determinar qual a melhor, para que seja possível começar a árvore de decisão.

Com o objetivo de ilustrar como é construído passo a passo um algoritmo de árvore de decisão, considere o exemplo proposto por Prof. Serafim Nascimento (Nascimento, 2016), utilizando a entropia como critério de encontrar o nó raiz.

Exemplo

O objetivo aqui é identificar a Renda (Baixa ou Alta) de oito pessoas observando as variáveis Idade (≤ 30 ou > 30) e Escolaridade (G: Graduação, M: Mestrado e D: Doutorado). Os dados estão na Tabela 2.1.

Tabela 2.1: Dados do Exemplo.

Escolaridade	**Idade**	**Renda**
Mestrado	> 30	Alta
Doutorado	≤ 30	Alta
Mestrado	≤ 30	Baixa
Doutorado	>30	Alta
Graduação	≤ 30	Baixa
Graduação	>30	Baixa
Mestrado	>30	Alta
Mestrado	≤ 30	Baixa

Calculando a entropia da variável resposta, temos:

$$E(\text{Renda}) = -\sum_{i=1}^{2} p_i \times \log_2 p_i \tag{2.1}$$

com $p_1 = \text{P}(\text{Renda} = \text{Alta}) = \frac{1}{2}$ e $p_2 = \text{P}(\text{Renda} = \text{Baixa}) = \frac{1}{2}$, logo:

$$E(\text{Renda}) = -\left(\frac{1}{2} \times \log_2 \frac{1}{2}\right) - \left(\frac{1}{2} \times \log_2 \frac{1}{2}\right) = 1$$

Como a unidade de medida da entropia é o *bit*, $\text{E(Renda)} = 1bit$.

Para calcular a entropia de uma variável nominal, é necessário criar um subconjunto de dados para a variável. Nesse caso, para construir, observamos a classe da variável resposta, e então qual valor pode assumir, obtendo a seguinte tabela:

Tabela 2.2: Frequência da variável Escolaridade, segundo a Renda.

Renda	G	M	D
Alta	0	2	2
Baixa	2	2	0

Calculando para cada variável, temos:

- Escolaridade

$$E(\text{Escolaridade}) = -\sum_{i=1}^{2} p_i \times \log_2 p_i$$

- $\text{P}(\text{Renda} = \text{Alta}|\text{Escolaridade} = \text{G}) = 0$

- P(Renda = Baixa|Escolaridade = G) = $\frac{1}{2}$

Assim,

$$E_G(\text{Escolaridade} = \text{G}) = -0 \times \log_2 0 - 1 \times \log_2 1 = 0bit$$

- P(Renda = Alta|Escolaridade = M) = $\frac{1}{2}$
- P(Renda = Baixa|Escolaridade = M) = $\frac{1}{2}$

$$E_M(\text{Escolaridade} = \text{M}) = -\left(\frac{1}{2} \times \log_2 \frac{1}{2}\right) - \left(\frac{1}{2} \times \log_2 \frac{1}{2}\right) = 1bit$$

- P(Renda = Alta|Escolaridade = D) = $\frac{2}{2} = 1$
- P(Renda = Baixa|Escolaridade = D) = $\frac{0}{2} = 0$

$$E_D(\text{Escolaridade} = \text{D}) = -(1 \times \log_2 1) - (0 \times \log_2 0) = 0bit$$

É necessário conhecer a entropia total da variável Escolaridade. Logo,

$$E(\text{Escolaridade}) = p_G \times E_G + p_m \times E_M + p_D \times E_D$$

$$\frac{2}{8} \times 0 + \frac{4}{8} \times 1 + \frac{2}{8} \times 0 = 0,5bit$$

Agora calculamos a entropia da variável Idade. Utilizando a medida entropia (2.1),

$$E(\text{Idade}) = -\sum_{1}^{2} p_i \times \log_2 p_i$$

com probabilidades

- P(Renda = Alta |Idade) > 30) = $\frac{3}{4}$
- P(Renda = Baixa |Idade) > 30) = $\frac{1}{4}$

e portanto,

$$E_{>30}(\text{Idade} > 30) = -(p_1 \times \log_2 p_1) - (p_2 \times \log_2 p_2)$$

$$E_{>30}(\text{Idade} > 30) = -\left(\frac{3}{4} \times (-0,415)\right) - \left(\frac{1}{4} \times (-2)\right) = 0,811bits.$$

e,

- P(Renda = Alta |Idade) $\leq 30) = \frac{1}{4}$
- P(Renda = Baixa |Idade) $\leq 30) = \frac{3}{4}$

$$E_{\leq 30}(\text{Idade} \leq 30) = -\left(\frac{1}{4} \times (-2)\right) - \left(\frac{3}{4} \times (-0,415)\right) = 0,811 bits$$

A entropia total da variável Idade é:

$$E(\text{Idade}) = p_{>30} \times E_{>30} + p_{\leq 30} \times E_{\leq 30}$$

$$E(\text{Idade}) = \frac{4}{8} \times 0,811 + \frac{4}{8} \times 0,811 = 0,811 bits$$

O ganho de informação de cada variável independente é dado por:

- $GI(\text{Escolaridade}) = E(\text{Renda}) - E(\text{Escolaridade}) = 1bit - 5bit = 0,5bit$
- $GI(\text{Idade}) = E(\text{Renda}) - E(\text{Idade}) = 1bit - 0,811bits = 0,189bits$

Como $GI(\text{Escolaridade}) > GI(\text{Idade})$, Escolaridade é o primeiro nó, denominado nó raiz de divisão da árvore de decisão.

Sabendo que o nó raiz é a variável Escolaridade, abrimos um ramo para cada um dos "valores" que a variável assume. Em seguida, observamos a variável resposta (valores que esta assume) e, caso assuma os mesmos valores, adicionamos a folha com a informação; caso contrário, não tendo a classe definida, partimos para a outra variável calculando a entropia de forma análoga ao que já foi apresentado, e apenas depois expandimos o ramo, até atingir a classe (folha).

Como ilustrado na Figura 2.2, o nó raiz contém a variável Escolaridade, a partir deste se inicia o processo de estratificação sendo a regra de decisão do fluxo: Graduação, Mestrado e Doutorado. Os dados seguem o fluxo e permanecem em um nó terminal (folha), nos caminhos da esquerda e da direita enquanto que no meio, o fluxo segue para a variável Idade, já que os dados ainda são heretogêneos, sendo esse um nó decisão, nele o teste feito é sobre a Idade, sendo dividido em dois ramos: idade > 30 ou idade ≤ 30 e então o fluxo segue para o nó terminal.

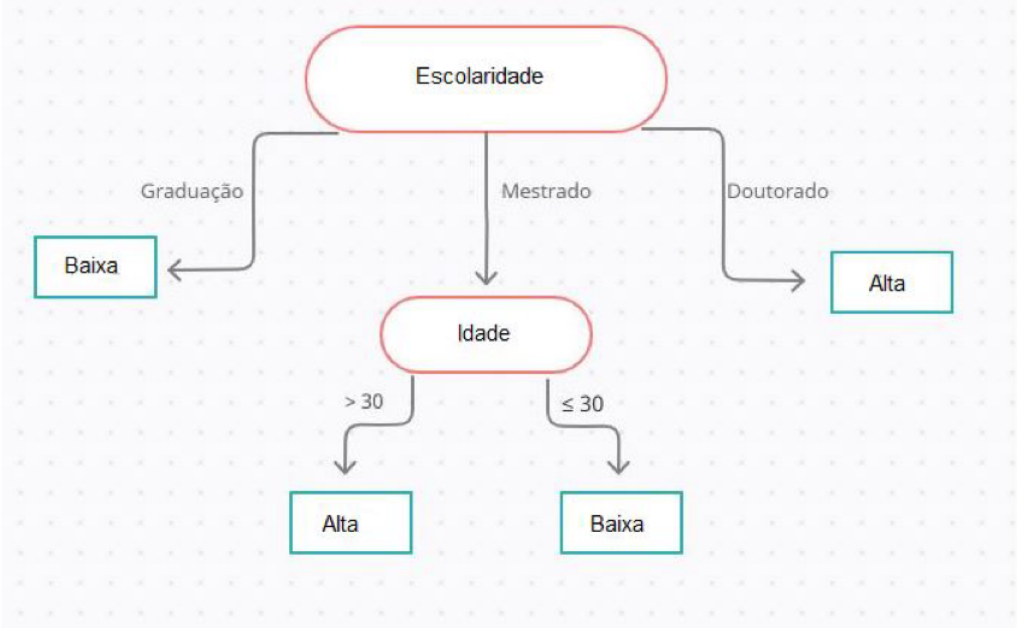

Figura 2.2: Exemplo de árvore de decisão.
Fonte: Ilustrado pela autora.

2.3 Índice de Gini

Desenvolvido por Conrado Gini (1912), mede a heterogeneidade dos dados e então é utilizado como medida de impureza de um nó (Onoda, 2001). Como na entropia, é verificada a distribuição dos dados nas variáveis preditoras de acordo com a variável resposta. Calculado o índice Gini, a preditora com menor índice será escolhida como nó raiz, já que um baixo valor do indice indica uma maior ordem na distribuição dos dados. O índice Gini é definido por,

$$\text{Gini}(t) = 1 - \sum_i p_i^2,$$

t é o nó, p_i a probabilidade da i-ésima folha do nó t. Se a variável resposta for binária, simplificamos:

$$\text{Gini}(t) = 2p(1|t)(1 - p(1|t))$$

sendo t arbitrário, $p(1|t)$ a probabilidade da variável resposta ser classificada na categoria ou primeira folha do nó t enquanto que a probabilidade de ser classificada na segunda categoria do nó t é dada por $(1 - p(1|t))$ (Diniz e Louzada, 2013). Este coeficiente foi utilizado no algoritmo desenvolvido em (Van Rossum e Drake Jr, 1995) na aplicação.

2.4 Redes Neurais

Rede neural artificial é um modelo preditivo, que recebeu esse nome por ser motivado pela forma como o cérebro funciona, com vários neurônios conectados. As redes neurais são como neurônios artificiais, que desenvolvem cálculos similares sobre suas entradas, muito utilizadas em *deep learning* (aprendizado profundo). As figuras 2.3 (Fonte: https://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio_artificial/index.html) e 2.4 ilustram a analogia entre rede neural artificial e um neurônio biológico.

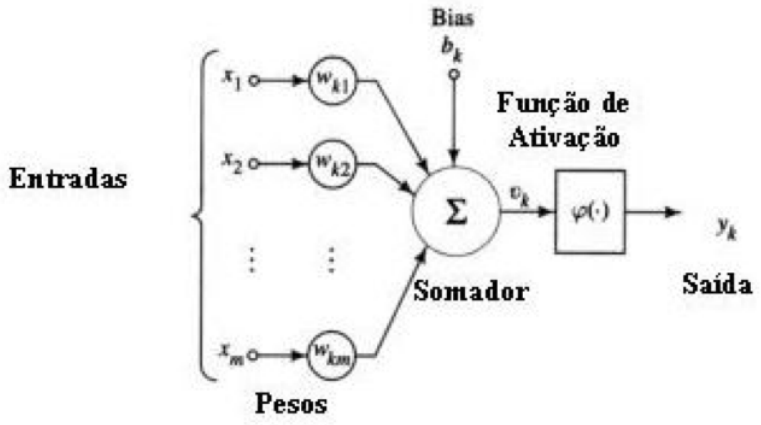

Figura 2.3: Exemplo de uma rede neural artificial.

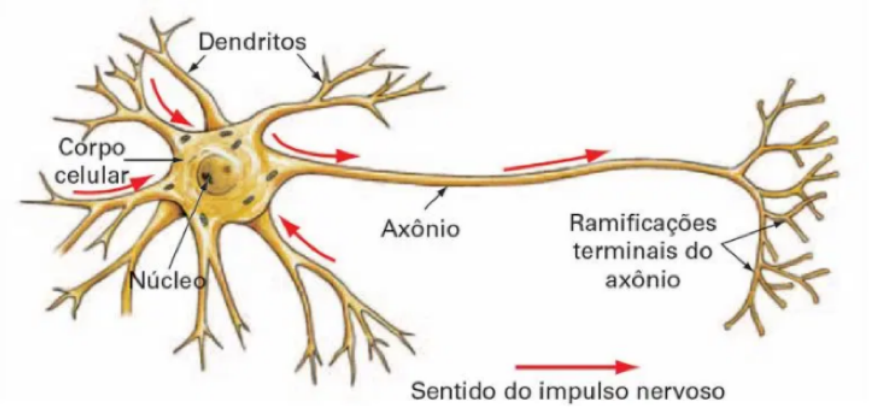

Figura 2.4: Exempo de um neurônio biológico.
Fonte: https://www.embarcados.com.br

Neurônios biológicos individuais são células fundamentais do cérebro humano, capazes de reconhecer padrões e relacioná-los e com isso, armazenar conhecimento por experiência,

dentre outras funções. São responsáveis pela transmissão dos impulsos nervosos, isto é, traduzem os estímulos do ambiente e preparam uma resposta. O cérebro humano possui cerca de 86 bilhões de neurônios e cada um desses faz, em média, 7 mil conexões com outros neurônios; esse processo é denominado sinapse, enviando informações por meio de impulsos elétricos para outro neurônio. Quando um neurônio está enviando esses impulsos a outro, dizemos que este está ativado; logo, o desativado é aquele que apenas recebe o impulso e não transmite (Silva, 2012).

Modelos de aprendizagem profunda (*deep learning*), como o nome sugere, são redes neurais artificias com muitas camadas ocultas (ou intermediárias). Existem modelos cujas estruturas são mais simples e um exemplo de tais modelos é o perceptron. Atualmente existem outros modelos, no entanto, o perceptron trás clara compreensão do funcionamento de uma rede neural em termos matemáticos, sendo uma boa introdução.

2.5 Modelo perceptron simples

O modelo perceptron foi proposto por Frank Rosenblatt nas décadas de 1950 e 1960, inspirado em trabalhos de Warren McCulloch e Walter Pitts, que apresentaram em 1940 o que se tratava do primeiro modelo artificial baseado no que se sabia na época sobre um neurônio biológico. Perceptron é uma rede, cujos neurônios estão dispostos em camadas. Tendo como estrutura apenas uma camada de rede, o modelo perceptron simples, permite uma compreensão mais clara do funcionamento de uma rede neural do ponto de vista matemático.

Rosenblatt (1961) demonstrou o teorema de convergência do perceptron, que prova que o neurônio MCP (unidade básica do perceptron) se treinado com o algoritmo de aprendizado do perceptron converge caso o problema em questão seja linearmente separável, sendo um problema linearmente separável, aqueles em que suas soluções podem ser separadas por uma reta.

Na configuração original proposta por Rosenblatt (1961), o modelo perceptron recebe várias entradas, por exemplo N denotadas por $x_1, x_2, \ldots, x_N$ tal que para cada uma dessas são atribuídos pesos $w_1, w_2, \ldots, w_N$ que expressam a importância dessa entrada, produzindo uma única saída binária, como mostrado na Figura 2.5, tendo também um "nível intermediário" formado pelas unidades de associação.

Os neurônios (conjunto) de entrada estão conectados a uma regra de aprendizado

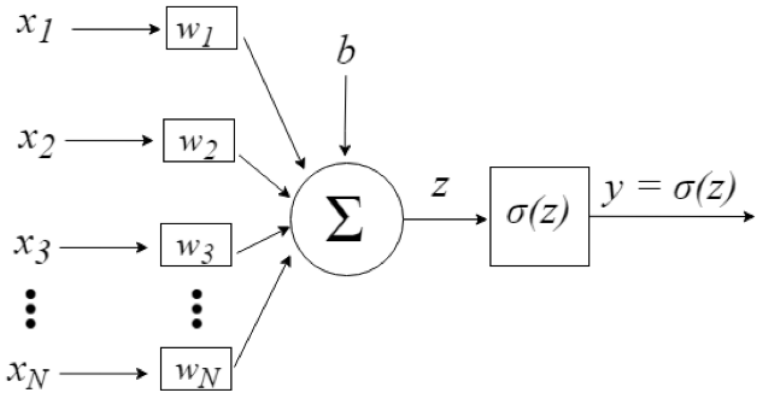

Figura 2.5: Modelo computacional de um neurônio.
Fonte: (Leite, 2016).

e durante esse processo a ideia é que no instante t se obtenha o valor de incremento $\Delta\mathbf{w}(t)$ afim de aplicá-lo no vetor de pesos pré definidos $\mathbf{w}(t)$ de forma que esse peso seja atualizado e $\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$ seja mais próximo do peso ideal do que $\mathbf{w}(t)$, com o objetivo de obter um valor $\Delta\mathbf{w}(t)$ mais próximo a solução do problema.

O teorema da convergência (Rosenblatt, 1958) diz que independente do valor de t, haverá a convergência em tempo finito, mas é preciso levar em consideração, que um valor inicial muito pequeno de t pode levar um grande tempo de convergência, enquanto que um valor muito alto, pode gerar instabilidade no processo de aprendizagem do perceptron. Não existe uma recomendação geral para todos os casos, então, o ajuste de t dependerá do problema proposto.

Resumidamente, o algoritmo visa obter o valor de $\Delta\mathbf{w}(t)$ mais apropriado para se obter a solução do problema proposto. Rosenblatt (1961) propôs uma regra básica para calcular a saída, introduzindo pesos, que são números reais que expressam a importância de cada entrada para a saída. Determinada pela soma ponderada aos pesos e aplicada na função de ativação, a saída do neurônio é binária.

Implementação do algoritmo do modelo Perceptron Simples

1. Escolher um valor t;

2. Escolher os valores dos pesos $\mathbf{w}$ de acordo com o problema;

3. Aplicar a regra de atualização dos pesos $\mathbf{w}(t+1) = \mathbf{w}(t) + n\epsilon\mathbf{x}(t)$;

4. Repetir os passos até obter $\varepsilon_T = 0$ para todos os casos.

O processo de treinamendo de uma RNA (Redes Neurais Artificiais) de um modelo Perceptron simples consiste em fazer com que o modelo aprenda valores de pesos (w) e vício (do inglês *bias* (b)). O algoritmo começa com a leitura dos dados dando início ao treinamento do modelo, fazendo com que os pesos e vícios sejam aprendidos e, com o modelo treinado podemos apresentar novos dados de entrada e o modelo será capaz de prever a saída.

Como citado, RNA's não são capazes de solucionar problemas não linearmente separáveis, como foi demonstrado por Minsky e Papert (2017). Para esses problemas são utilizadas as RNA's Perceptron Multicamadas, aprensentadas na próxima subseção.

2.5.1 Redes Perceptron Multicamadas

Quando o problema em questão não admite uma separação linear exata, torna-se necessário o uso da RNA perceptron multicamadas (Ambrósio, 2002). Estas tem poder computacional superior ao modelo simples, podendo ser implementadas quaisquer funções, seja separável linearmente ou não, o que é uma evolução em relação aos perceptrons simples (Cybenko, 1989). As MLP's são subdivididas em uma camada de entrada, camada(s) intermediária(s) e a camada de saída (Nied, 2007), em que,

- **Primeira camada**: cada neurônio apresenta retas para que seja formada a superfície no espaço de entrada;
- **Camada(s) intermediárias**: os neurônios dessa(s) camada(s) combinam as retas descritas pelos neurônios da camada anterior conectados a ele, formando regiões convexas, em que o número de lados é definido pelo número de unidades a ele conectadas;
- **Camada de saída**: cada neurônio forma regiões que são combinações das regiões convexas definidas pelos neurônios a ele conectados da camada anterior. Os neurônios definem, dessa maneira, regiões com formatos diversos.

A primeira camada dispara as informações de entrada para as camadas intermediárias da rede e na camada de saída, é obtida a solução do problema, sendo que os neurônios de uma determinada camada está conectado apenas aos neurônios adjacentes, camadas estas, que são totalmente conectadas como é possível observar na Figura 2.6.

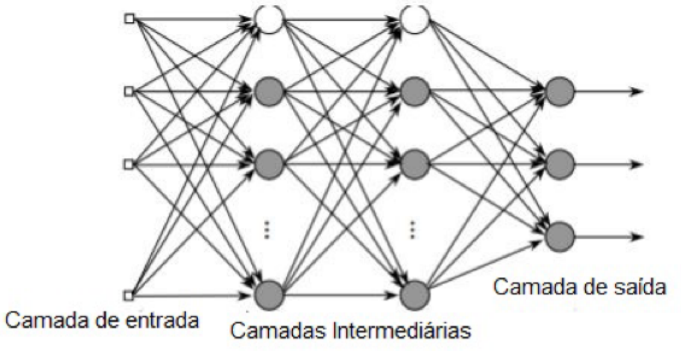

Figura 2.6: Arquitetura RNA perceptron multicamadas.
Fonte: Ilustrado pela autora.

A informação é passada de camada a camada, da esquerda para a direita, como mostrado na Figura 2.6.

Vantagens e desvantagens.

Segundo Simon (1999) as RNA's do tipo MLP tem como vantagem:

- Um neurônio artificial pode ser linear ou não, sendo a não linearidade distribuída por toda a rede;
- Adaptação dos pesos sinapticos quando o meio é modificado, então uma RNA já treinada, pode resolver problemas ainda quando forem feitas pequenas alterações;
- Possibilita rejeitar padrões indefinidos, melhorando o desempenho da classificação.

Ambrósio (2002) destaca as desvantagens:

- Dependendo da rede, o treinamento desta pode ser muito demorado;
- Não é possível descobrir o motivo por trás que levou a rede à determinada conclusão para o problema proposto;
- Necessita de um grande volume de dados;
- É necessário preparar os dados: estes precisam ser normalizados e bem selecionados para apresentar resultados confiáveis e não podem ser utilizados dados categoricos, sendo necessária a transformação em *dummies*.

Para problemas de maior complexidade, é necessário a utilização de estruturas com características não lineares, essas não linearidades são transferidas para o modelo através

das funções de ativação não lineares de cada neurônio da rede. Neste trabalho utilizamos o algoritmo *Backpropagation* para treinar a rede.

Backpropagation

Rumelhart *et al.* (1986) desenvolveram o algoritmo *backpropagation* de treinamento, que foi o pioneiro e é ainda o mais popular para treinar as redes perceptron multicamadas. Cada camada da rede tem uma função específica. A de saída, recebe os estímulos da(s) intermediária(s) e a partir destas, produz o padrão de resposta. O algoritmo utiliza um mecanismo para a correção de erros afim de ajustar os pesos da rede (Widrow e Lehr, 1995), sendo o treinamento da rede feito em duas partes:

- Fase *Forward*: define a saída da rede para certo padrão de entrada.
- Fase *Backward*: compara a saída desejada com a fornecida pela rede para atualizar os pesos das conexões.

Sendo os pesos das camadas modificados conforme o erro é retroprogramado, o objetivo sempre é minimizar o erro na saída do algoritmo. Beale e Jackson (1990) apontaram essa questão como sendo uma dificuldade, pois esse cálculo dos pesos se torna ainda mais difícil quando aumentados o número de camadas intermediárias. A minimização desse erro é feita através do gradiente descendente. Em cada iteração de treino, os pesos são atualizados afim de minimizar o erro quadrático médio entre y's e $\widehat{y}'$s, sendo y's os verdadeiros resultados e $\widehat{y}'$s as previsões da rede.

Um ponto interessante do algoritmo *backpropagation* é que este começa enviando um sinal funcional, que vai da camada de entrada até a saída no processo de treinamento da rede, fazendo esse processo camada a camada, onde é produzido um conjunto de saída como resposta da rede, tendo nessa fase, todos os pesos sinápticos fixos. Então, o algoritmo faz o caminho inverso, da saída até a entrada, camada a camada, e esse processo é denominado retroprogramação, e durante este, os pesos sinápticos são ajustados utilizando a regra de correção do erro (descrito em implementação do modelo Perceptrom Simples, passo 3). Nessa retroprogramação é verificada se a saída apresentada foi satisfatória. Para melhor entendimento, os passos do algoritmo nessas duas etapas são:

- *Forward*

1. É inserido um vetor $\boldsymbol{x}$ de entrada e as saídas C_1 da primeira camada escondida são calculadas.
2. As saídas C_1 serão as entradas da camada adjacente C_2. As saídas dessa segunda camada são calculadas e assim por diante até a última camada intermediária.
3. As saídas da última camada da RNA são comparadas com as saídas desejadas para cada valor de $\boldsymbol{x}$ e então o erro é calculado.

- *Backward*

1. O erro encontrado na camada de saída do algoritmo é utilizado para atualizar os pesos, via gradiente descendente.
2. Os erros da camada de saída farão o caminho "inverso", sendo multiplicado pela camada adjacente anterior, resultando em um erro estimado, que representa o "peso"do neurônio no erro final da RNA.
3. O item anterior será repetido até chegar na camada de entrada.
4. O processo é feito até que todos os pesos sejam ajustados.

Na Figura 2.7 o passo a passo descrito.

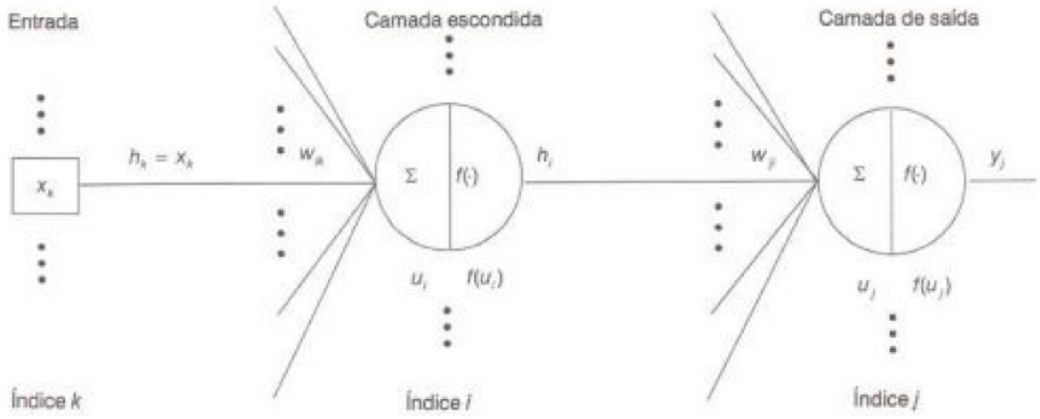

Figura 2.7: Exemplo de uma Rede MLP.
Fonte: Silva (2012)

Segundo Silva (2012) um neurônio j, com saída linear u_j, como denotado na Figura 2.7, corresponde à soma ponderada das entradas e da saída, que é obtida após aplicar a função de ativação sobre u_j, então, $\gamma_j = f(u_j)$. Para melhor entendimento denotamos:

- $h(u_j)$ é a saída do neurônio das camadas intermediárias, para um neurônio j qualquer;
- $f(u_j)$ corresponde a camada de saída da RNA (última camada).

A t-ésima iteração de um neurônio de saída j é definido por $e_j(t) = \gamma_d^j(t) - \gamma_j(t)$ e a soma dos erros quadráticos desses neurônios de saída na t-ésima iteração é dado por (Silva, 2012):

$$\varepsilon(t) = \frac{1}{2}\sum_j e_j^2(t).$$

O erro do neurônio j pode ser escrito como $e_j(t) = \gamma_d^j(t) - f(u_j(t))$ sendo $u_j(t) = \sum_i h_i(t)w_{ji}(t)$ a saída do neurônio j na camada de saída, logo, a soma dos erros quadráticos na t-ésima iteração é reescrita por:

$$\varepsilon(t) = \frac{1}{2}\sum_j (\gamma_g^j(t) - f(u_j(t)))^2.$$

Silva (2012) descreve com detalhes as equações necessárias para obtenção das camadas de saída e intermediária.

2.5.2 Função de ligação

A função utilizada 'dentro' de cada neurônio, chamada de função de ativação, definem como devem ser seus dados de entrada. A mais utilizada em classificação é a SoftMax, que possui a forma:

$$f_i(z) = \frac{exp(z_i)}{\sum_{j=1}^{C} exp(z_j)}.$$

A função SoftMax, faz com que a saída da rede neural, seja a probabilidade dos dados pertencerem a uma das classes definidas. Essa função vai garantir que a saída do neurônio fique entre 0 e 1 e que a soma de todas as saídas seja 1.

2.6 Regressão Logística

Modelos de regressão são uma das principais ferramentas para a modelagem de dados e, podem ser usados nas mais variadas situações. Para aplicar esta metodologia fazemos

algumas suposições; por exemplo, a distribuição do termo aleatório segue uma distribuição normal com média zero e variância constante. Porém, em diversas situações a variável resposta. Esta classe de modelos generaliza a classe de modelos de regressão, pois a variável resposta pode ser assumir outras distribuições diferentes da normal, como por exemplo, Bernoulli, Poisson, entre outras, que pertencem á família exponencial. Nelder e Wedderburn (1972) definiram os modelos lineares generalizados (MLG's).

Sejam $Y_1, Y_2, \ldots, Y_n$ variáveis aleatórias independentes sendo que cada uma possui uma função densidade de probabilidade $f(y_i|.)$. Para pertencer à família exponencial $f(y_i|.)$ deve ser escrita na forma:

$$f(y_i, \theta_i, \phi) = \exp\{\phi^{-1}[y_i\theta_i - b(\theta_i)] + c(y_i, \phi_i)\}, \quad i = 1, \ldots, n,$$

em que $b(\theta)$ e $c(.)$ são funções reais, ϕ é o parâmetro de dispersão.

Um modelo, escrito como em (2.2) é composto por três componentes: uma parte aleatória, que se refere à distribuição da variável, uma componente sistemática, que especifica um preditor linear entre as covariáveis, e uma função de ligação, responsável pela ligação entre as componentes aleatória e sistemática dada por,

$$g(\mu_i) = \eta_i = X_i^T\beta.$$

em que,

- $\boldsymbol{x}_i = (x_{i1}, x_{i2} \ldots, x_{ik})$ representa o vetor de covariáveis para a observação i,
- $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)$ correspondente ao vetor dos parâmetros do modelo, associado ao vetor de covariáveis $\boldsymbol{x}$.

Como a variável resposta é binária, o modelo binomial é considerado e a função de ligação, mais utilizada neste caso, é a logito. O modelo com a função logito é dado por:

$$\ln(\frac{\mu_i}{1-\mu_i}) = X_i^T\boldsymbol{\beta},$$

que pode ser reescrito na forma:

$$\mu_i = \frac{\exp(x_i^T\boldsymbol{\beta})}{1+\exp(x_i^T\boldsymbol{\beta})}.$$

A regressão logística é um modelo que relaciona um conjunto de n variáveis independentes $(X_1, X_2, \ldots, X_n)$ a uma variável dependente Y que assume apenas dois possíveis resultados, digamos 0 ou 1. Segundo os autores, há pelo menos duas razões para utilização do modelo logístico na análise de variáveis resposta dicotômicas, sendo estas, do ponto de vista matemático, é extremamente flexível e fácil de ser utilizado e, permite uma interpretação de resultados bastante rica (Hosmer e Lemeshow, 2000).

As estimativas para os coeficientes do modelo são obtidos via método de máxima verossimilhança (Nelder e Wedderburn, 1972).

Considere uma amostra com n elementos, neste caso, uma covariável denotada por X, a variável resposta Y que assume dois valores. Seja $\pi(x)$ a proporção de maus pagadores, sendo então uma função que assume valores entre zero e um, e $x_1, \ldots, x_n$ valores independentes da variável aleatória X.

Como $\pi(.)$ varia entre zero e um, uma representação para π sobre todos os valores possíveis de X é não adequada (pois caso contrário, tem-se o modelo de regressão linear), então considera-se a transformação logística de $\pi(.)$ sob a forma linear:

$$g(x) = \ln\left(\frac{\pi(x)}{1-\pi(x)}\right),$$

com

$$g(x) = \beta_0 + \beta_1 x.$$

Então,

$$\ln\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x$$

$$\frac{\pi}{1-\pi} = \exp\{\beta_0 + \beta_1 x\}$$

$$\pi(x) = \exp\{\beta_0 + \beta_1 x\}(1 - \pi(x))$$

$$\pi(x) = \exp\{\beta_0 + \beta_1 x\} - \exp\{\beta_0 + \beta_1 x\}\pi(x)$$

$$\pi(x) + \exp\{\beta_0 + \beta_1 x\}\pi(x) = \exp\{\beta_0 + \beta_1 x\}$$

$$\pi(x)(1 + \exp\{\beta_0 + \beta_1 x\}) = \exp\{\beta_0 + \beta_1 x\}$$

$$\pi(x) = \frac{\exp\{\beta_0 + \beta_1 x\}}{1 + \exp\{\beta_0 + \beta_1 x\}} \tag{2.7}$$

O interesse é estimar os parâmetros e para isso, usamos o método da máxima veros-

similhança.

2.6.1 Função de Máxima Verossimilhança

A função de máxima verossimilhança é um dos procedimentos usuais utilizados para obtenção de estimadores. Esse método trata o problema de estimação usando uma amostra aleatória retirada da população em estudo (Casella e Berger, 2021).

Considere X uma variável aleatória observada em uma população e definida a função densidade de probabilidade de X, f(x,θ), com θ desconhecido. Dessa população, retiramos uma amostra aleatória simples de tamanho n de X, denotado por $x_1, \ldots, x_n$. O estimador de máxima verossimilhança é encontrado derivando a função de verossimilhança com relação ao parâmetro, igualando a zero e avaliando a segunda derivada. Ou ainda, derivar o logaritmo da func cão de verossimilhança (Casella e Berger, 2021).

Como caso geral, considere $\boldsymbol{\theta}$ um vetor de dimensão k, isto é $\theta_1, \ldots, \theta_k$. Desta forma, a função de máxima verossimilhança L é definida por:

$$L(\boldsymbol{\theta}; x_1, \ldots, x_n) = f(x_1; \boldsymbol{\theta}) \times \ldots \times f(x_n; \boldsymbol{\theta})$$

Considerando que a variável resposta Y tem distribuição binomial, a função de máxima verossimilhança é dada por:

$$L(\beta_0, \beta_1 | y, x) = \prod_{i=1}^{n} \pi(x_i)^{y_i} (1 - \pi(x_i))^{1-y_i}$$

Substituindo $\pi(x)$, a função de verossimilhança é reescrita da seguinte forma:

$$L(\beta_0, \beta_1 | y, x) = \prod_{i=1}^{n} \left(\frac{\exp\{\beta_0 + \beta_1 x_i\}}{1 + \exp\{\beta_0 + \beta_1 x_i\}} \right)^{y_i} \left(1 - \left(\frac{\exp\{\beta_0 + \beta_1 x_i\}}{1 + \exp\{\beta_0 + \beta_1 x_i\}} \right) \right)^{1-y_i}$$

Aplicando a função logaritmo na função de verossimilhança, temos:

$$L(\beta_0, \beta_1 \mid y, x) = \sum_{i=1}^{n} \left[y_i \ln \left(\frac{\exp\{\beta_0 + \beta_1 x\}}{1 + \beta_0 + \beta_1 x} \right) + (1 - y_i) \ln \left(1 - \frac{\exp\{\beta_0 + \beta_1 x\}}{1 + \beta_0 + \beta_1 x} \right) \right]$$

$$\ln(L(\beta_0,\beta_1|y,x)) = \sum_{i=1}^{n}(y_i(\beta_0+\beta_1 x_i) - y_i(\ln((1+exp\{\beta_0+\beta_1 x_i\}))+$$

$$(1-y_i)(\ln(1+\exp\{\beta_0+\beta_1 x_i\} - \exp\{\beta_0+\beta_1 x_i\}) - \ln(1+\exp\{\beta_0+\beta_1 x_i\}))$$

$$\ln(L(\beta_0,\beta_1|y,x)) = \sum_{i=1}^{n}(y_i(\beta_0+\beta_1 x_i) - y_i(\ln(1+\exp\{\beta_0+\beta_1 x_i\}))+$$

$$(\ln(1)) - \ln(1+\exp\{\beta_0+\beta_1 x_i\} - y_i\ln(1) + y_i\ln(1+\exp\{\beta_0+\beta_1 x_i\}))$$

$$\ln(L(\beta_0,\beta_1|y,x)) = \sum_{i=1}^{n}(y_i\beta_0 + y_i\beta_1 x_i - \ln(1+\exp\{\beta_0+\beta_1 xi\}))$$

$$\ln(L(\beta_0,\beta_1|y,x)) = \sum_{i=1}^{n} y_i\beta_0 + \sum_{i=1}^{n} y_i\beta_1 x_i - \sum_{i=1}^{n}\ln(1+\exp\{\beta_0+\beta_1 x_i\})$$

Derivando em relação a β_0 e a β_1, obtemos:

$$\frac{\partial \ln(L(.))}{\partial \beta_0} = \sum_{i=1}^{n} y_i - \sum_{i=1}^{n}\frac{\exp\{\beta_0+\beta_1 x_i\}}{1+\exp\{\beta_0+\beta_1 x_i\}}\}$$

$$\frac{\partial \ln(L(.))}{\partial \beta_1} = \sum_{i=1}^{n} y_i x_i - \sum_{i=1}^{n}\frac{\exp\{\beta_0+\beta_1 x_i\}\, x_i}{1+\exp\{\beta_0+\beta_1 x_i\}}.$$

Os estimadores são encontrados resolvendo o sistema formado pelas $p+1$ equações, sendo p o número de covariáveis no modelo. As equações encontradas são não-lineares nos parâmetros, e desta forma, é necessário o uso de um método iterativo, como por exemplo, o de Newton-Raphson.

A partir das derivadas encontradas, temos:

$$\sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{n}\frac{\exp\{\beta_0+\beta_1 x_i\}}{1+exp\{\beta_0+\beta_1 x_i\}}x_i$$

$$\sum_{i=1}^{n} y_i = \frac{\exp\{\beta_0+\beta_1 x_i\}}{1+exp\{\beta_0+\beta_1 x_i\}}$$

2.7 Métricas de Classificação

Uma métrica muito utilizada para avaliar o desempenho das classificassões é a matriz de confusão. Esta é baseada na quantidade de acertos, ou seja, na porcentagem de observações classificadas corretamente. Defina as classes: sucesso (evento de interesse) e fracasso. A Tabela 2.3 é chamada de matriz de confusão:

Tabela 2.3: Estrutura da matriz de confusão.

Predito	Observados		Total
	Positivo	Negativo	
Positivo	VP	FP	p
Negativo	FN	VN	n

em que,

VP : verdadeiro positivo: valor observado de sucesso que foi classificado corretamente como sucesso;

VN : verdadeiro negativo: valor observado de fracasso que foi classificado corretamente como fracasso;

FP : falso positivo: valor observado de fracasso que foi classificado como sucesso;

FN : falso negativo: valor observado de sucesso que foi classificado como fracasso;

p : número de observações classificadas como sucesso;

n : número de observações classificadas como fracasso;

O objetivo é comparar classificações feitas pelo modelo com as observações do conjunto de dados. Com base na matriz de correlação, é possível calcular:

Sensibilidade: probabilidade de uma observação ser classificada como *sucesso*, dado que realmente é *sucesso*;

$$\text{Sensibilidade} = \frac{VP}{VP + FN}. \tag{2.8}$$

Especificidade: probabilidade de uma observação ser classificada como *fracasso* dado que realmente é *fracasso*.

$$\text{Especificidade} = \frac{VN}{VN + FP}. \tag{2.9}$$

Acurácia: avalia simplesmente o percentual de acertos.

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN}. \tag{2.10}$$

É necessário definir, para a construção da matriz de correlação, um ponto de corte, que é o responsável por classificar os valores gerados em 0 e 1. Um ponto de corte usual, que é considerado conservador, é $0,5$. Em geral, queremos que as três métricas ofereçam altos resultados, mas dependendo do contexto, pode existir preferência entre alguma das métricas.

Capítulo 3

Aplicação das Metodologias

Para analisar as performances dos modelos, em cenários diferentes, foram utlizados um conjunto de dados simulados e um conjunto de dados extraído do repositório da UCI Machine Learning (Repository, 2019) para ap;icacção das metodologias estudadas. Neste capítulo apresentamos os ajustes dos modelos e as métricas para comparação dos resultados, nas três técnicas.

3.1 Exemplo 1: Conjunto de Dados Simulados

Para a simulação da base de dados, foram consideradas variáveis geralmente usadas no contexto de estimação de escores de propensão.

3.1.1 Aspectos da simulação

Utilizando a função random(do pacote random), já implementada no *Python* (Van Rossum e Drake Jr, 1995), utilizada para gerar números aleatórios, definidos a partir dos parâmetros de alguma distribuição específica. As variáveis score e limite de crédito foram geradas seguindo distribuição Beta. Para a variável saldo em conta, foi usada a distribuição Exponencial.

Na construção da variável salário, foi levada em conta as variáveis saldo em conta e limite de crédito e também adicionada uma constante para que a variável não fosse menor ou igual a zero. Nas variáveis gênero, membro ativo e conta em outro banco, foi utilizada a função 'choice', que seleciona qualquer elemento aleatoriamente, então foram fixados os elementos 0 e 1 e associada uma probabilidade para cada um deles em cada uma dessas

variáveis, a variável escolaridade também foi construída utilizando essa mesma função, com outros elementos fixados e levando em considração a variável gênero.

Para a variável saldo bancário, foi utilizada a função Exponencial. A distribuição binomial foi utilizada para a construção da variável usa ou não cartão de crédito, e por fim, para as variáveis idade e estado civil, foi usada também a função choice, com estado civil levando em consideração as categorias da variável idade, sendo assossiadas probabilidades maiores para os indivíduos que tinham mais idade. Após construir o conjunto de valores para as covariáveis, atribuímos valores coerentes para os β, com "chute"inicial sendo os gerados por um modelo inicial com variável resposta gerada através da distribuição Bernoulli, as estimativas desse modelo inicial sofreram alguns ajustes até chegar na proporção desejada para a variável resposta (50% de valores igual a 0 e 50% de valores iguais a 1) , e assim criamos a variável resposta com distribuição Bernoulli ($\pi(x)$). O código da simulação está presente no apêndice A.

A variável resposta, classificação do cliente ("bom" ou "mau" pagador), as variáveis preditoras e os respectivos tipos são mostrados na Tabela 3.1.

Tabela 3.1: Variáveis utilizadas na base de dados simulados.

Variável	Tipo
Y (Situação do cliente)	Binária
Renda	Numérica
Limite de Crédito	Numérica
Saldo em conta	Numérica
Credit Score	Numérica
Gênero	Categórica
Membro ativo	Binária
Conta em outro banco	Binária
Escolaridade	Categórica
Idade	Categórica
Estado Civil	Categórica
Cartão de Crédito	Binária

As variáveis foram simuladas tais que 45% dos clientes são do gênero feminino, 52% são membros ativos e 31% tem conta corrente em outro banco. No caso do nível de escolaridade, os níveis educacionais são Fundamental Incompleto (15%), Médio incompleto (9%), Superior incompleto (48%) e Superior completo (28%). Com relação à idade dos clientes, 12% tem de 18 até 25 anos, 72% tem entre 26 e 59 anos e 16% tem idade acima de 60 anos. Por fim, 35% faz uso do cartão de crédito.

Ainda, sobre a simulação das variáveis, destacamos que: A variável Limite Crédito foi definida utilizando a variável Crédito Score, ou seja, para atribuir os valores que fizessem sentido para a variável, por exemplo, para uma maior pontuação de Crédito Score, foi atribuído um maior valor na variável Limite. Para a criação da variável Renda foi levado

em conta as variáveis Saldo em Conta e também Limite de Crédito, sendo controlada para que não houvessem valores negativos; gênero e idade e, idade e escolaridade foram simuladas considerando uma coeficiente de dependência. No caso do uso de cartão de crédito também foi pré estabelecido uma dependência com a variável Membro Ativo.

Para as variáveis Limite de Crédito, Renda e Saldo em Conta, registradas em reais(R$) e Crédito *Score*(pontuação) algumas medidas resumo são apresentadas na Tabela 3.2.

Tabela 3.2: Medidas resumo para as variáveis quantitativas.

Medidas	Crédito Score	Limite de Crédito	Renda	Saldo em Conta
Média	494,56	4120,79	3517,93	8054,66
Desvio Padrão	200,11	3007,86	2630,15	9831,90
Mínimo	30,00	200,00	70,83	0,00
Quartil 1	340,00	1733,50	1626,06	756,77
Quartil 2	490,00	3443,00	2904,03	4859,22
Quartil 3	650,00	5866,00	4670,98	11618,0
Máximo	980,00	17168,00	29325,04	103430,78

A partir dos resumos estatísticos (Tabela 3.2) é possível observar que as varéveis apresentam alta dispersão. A variável Saldo em Conta tem a maior dispersão ao redor da média. O coefciente de variação é aproximadamente igual a 122%, sendo que a variável que apresenta menor coeficiente de variação é a Crédito Score (aproximadamente 40%).

No caso de ajuste de um modelo de regressão logística, as estimativas, bem como os percentis 2,5% e 97,5% são mostrados na Tabela 3.3

Tabela 3.3: Estimativas obtidas via Regressão logística.

Coeficientes	Estimativas	$P[Z \geq \|z\|]$	0,025	0,975
Intercepto	-0,2365	0,000	-0,481	0,008
Crédito Score	0,0002	0,022	-5,87e-05	0,000
Limite de crédito	7,99e-06	0,456	-1,3e-05	2,9e-05
Renda	-9,47e-06	0,005	-3,38e-05	1,48e-05
Saldo em conta	4,63e-06	0,115	-1,12e-06	1,04e-05
Gênero	0,026	0,649	-0,086	0,138
Membro Ativo	0,073	0,000	-0,088	0,234
Conta em outro banco	0,117	0,000	-0,004	0,239
Ensino Médio incompleto	0,168	0,002	-0,082	0,419
Ensino Superior incompleto	0,056	0,551	-0,127	0,239
Ensino Superior completo	0,038	0,704	-0,158	0,233
Idade entre 26 e 59 anos	-0,159	0,008	-0,276	-0,042
Jovem até 25 anos	0,033	0,686	-0,125	0,190
Casado	0,072	0,001	-0,152	0,296
Divorciado	-0,142	0,314	-0,418	0,134
Solteiro	0,027	0,818	-0,205	0,259
Cartão de Crédito	0,025	0,001	-0,142	0,193

As variáveis: Membro Ativo, Conta em outro banco, Escolaridade['Ensino fund completo e médio incompleto'], Estado Civil['Casado'] e Cartao de Crédito foram significativas

para o modelo se considerarmos um nível de 5% de significância. O intervalo de confiança dos coeficientes, últimas duas colunas da Tabela 3.3, não possui muita variabilidade, logo, a estimação foi precisa.

Para as métricas de performance dos modelos foi considerado o ponto de corte 0,5 e nenhum método de seleção de variáveis foi utilizado.

A Tabela 3.4 apresenta os resultados da matriz de confusão do modelo de regressão logística (2.3), sendo que 120.642 observações foram classificadas corretamente.

Tabela 3.4: Matriz de confusão para o modelo de Regressão Logística.

Predito	Observados		Total
	Fracasso	Sucesso	
Fracasso	41%	9%	50%
Sucesso	10,5%	39,5%	50%
Total	51,5%	48,5%	100%

Pela medida acurácia do modelo, apresentada na Tabela 3.5, notamos que não ocorreu diminuição no resultado dessa métrica nos dados usados na parte de teste quando comparado aos resultados obtidos na parte de treinamento do modelo. Portanto não temos indicativo de *overfitting*, ou seja, quando um modelo se ajusta muito bem ao conjunto de dados de treino mas se mostra ineficiente para predizer novos resultados. O tempo de processamento (tempo que demorou para que o algoritmo chegasse ao resultado do modelo), foi obtido via função *time* da biblioteca *time* da linguagem Python.

Tabela 3.5: Desempenho preditivo do modelo de regressão logística.

Medidas	Valores
Acurácia treino	0,801
Acurácia teste	0,804
Especificidade	0,806
Sensibilidade	0,802
Tempo de processamento	2minutos e 8segundos

Para a metodologia de Redes Neurais, seguem os resultados:

A Tabela 3.6 apresenta a matriz de confusão do modelo de Redes Neurais, sendo que 123.090 observações foram classificadas corretamente.

Tabela 3.6: Matriz de confusão para o modelo de Redes Neurais.

Predito	Observados		Total
	Fracasso	Sucesso	
Fracasso	42,58%	8%	50,58%
Sucesso	9,94%	39,48%	49,42%
Total	52,52%	47,48%	100%

Destacando os resultados das acurácias apresentadas na tabela 3.5, não temos indicativo de *overfitting*, no modelo de Redes Neurais.

Tabela 3.7: Desempenho preditivo do modelo de Redes Neurais.

	Redes Neurais
Acurácia treino	0,821
Acurácia teste	0,820
Especificidade	0,831
Sensibilidade	0,810
Tempo de processamento	9Minutos 02Segundos

Para a metodologia de Árvore de Classificação, seguem os resultados:

A Tabela 3.8 apresenta a matriz de confusão do modelo de Árvore de Classificação, 107.673 observações foram classificadas corretamente.

Tabela 3.8: Matriz de confusão para o modelo de Redes Neurais.

Predito	Observados		Total
	Fracasso	Sucesso	
Fracasso	37,34%	11,10%	48,44%
Sucesso	17,13%	34,43%	51,56%
Total	54,47%	45,53%	100%

Destacando novamente os resultados das acurácias, apresentadas na tabela 3.9, vemos uma leve diminuição da acurácia calculada com os dados de teste em relação a calculada com o dados de treino, mas ainda não é uma diferença considerável que seria um indicativo de *overfitting*.

Tabela 3.9: Desempenho preditivo do modelo de Árvore de Classificação.

	Árvore de Classificação
Acurácia treino	0,741
Acurácia teste	0,717
Especificidade	0,756
Sensibilidade	0,685
Tempo de processamento	6Minutos 48Segundos

3.2 Exemplo 2: Banco Português

Este banco de dados, retirado do repositório da UCI Machine Learning (Repository, 2019), é referente à campanha de *marketing* direto de uma instituição bancária portuguesa, baseada em ligações telefônicas no período de maio de 2008 até novembro de 2010. Este possui 45.211observações e 11 variáveis. O objetivo da classificação é predizer se o cliente irá assinar ou não um depósito a prazo. Na Tabela 3.10 é apresentada a descrição e tipo das variáveis originais presentes no banco de dados do Banco Português.

Tabela 3.10: Variáveis base do Banco Português.

Variável	Descrição	Tipo
Y	Assinatura de um depósito a prazo	Binária
X_1	Idade	Numérica
X_2	Profissão	Categórica
X_3	Estado Civil	Categórica
X_4	Formação	Categórica
X_5	Empréstimo em atraso	Binária
X_6	Balanço médio anual	Numérico
X_7	Financiamento Imobiliário	Binária
X_8	Empréstimo pessoal	Binária
X_9	Meio de comunicação da campanha	Categórica
X_{10}	Duração do contato	Numérico
X_{11}	Resultado da última campanha	Categórica

Como foi feito no conjunto de dados simulados, as variáveis categóricas foram trans-

formadas em variáveis *dummies*. Também, a variável resposta foi balanceada, uma vez que na base inicial, 88% dos clientes não haviam assinado o depósito, o que pode prejudicar o desenvolvimento do ajuste (Diniz e Louzada, 2013), já que a porcentagem de clientes que haviam assinado o depósito era de apenas 12%, que pode ser insuficiente para observar possíveis diferenças em relação aos clientes que assinaram ou não um depósito a prazo. Além disso foi verificada a ausência de dados faltantes e a não existência de dados duplicados.

Feita as tratativas citadas, a base utilizada na modelagem contém 10.576 observações e a seguir (Tabela 3.11) uma breve análise descritiva dos dados é apresentada.

Tabela 3.11: Medidas resumo das variáveis Idade, Balanço médio anual e Duração do Contato.

	Idade	Balanço médio anual	Duração do Contato
Média	41	1578,49	380,71
Desvio Padrão	12	3368,57	354,11
Mínimo	18	-3372,00	2,00
Quartil 1	32	122,00	143,00
Quartil 2	39	557,00	262,00
Quartil 3	49	1747,50	507,00
Máximo	95	81204,00	4918,00

Destacamos, com base na Tabela 3.11 que a variável Idade, apesar da média e mediana serem valores próximos, não tem distribuição simétrica, uma vez que o terceiro quartil e o valor máximo são valores muito distantes. A variável Balanço médio anual possui alta variabilidade.

A Figura 3.1 apresenta a matriz de correlaçao das variáveis quantitativas Idade, Balanço médio anual e Duração do Contato. Pela figura, podemos verificar que não se existe multicolinearidade entre as variáveis, uma vez que a maior correlação que é entre as variáveis Idade e Balanço médio anual, é baixa. Logo, não é um alerta de possível prejuízo na estimação dos parâmetos do modelo.

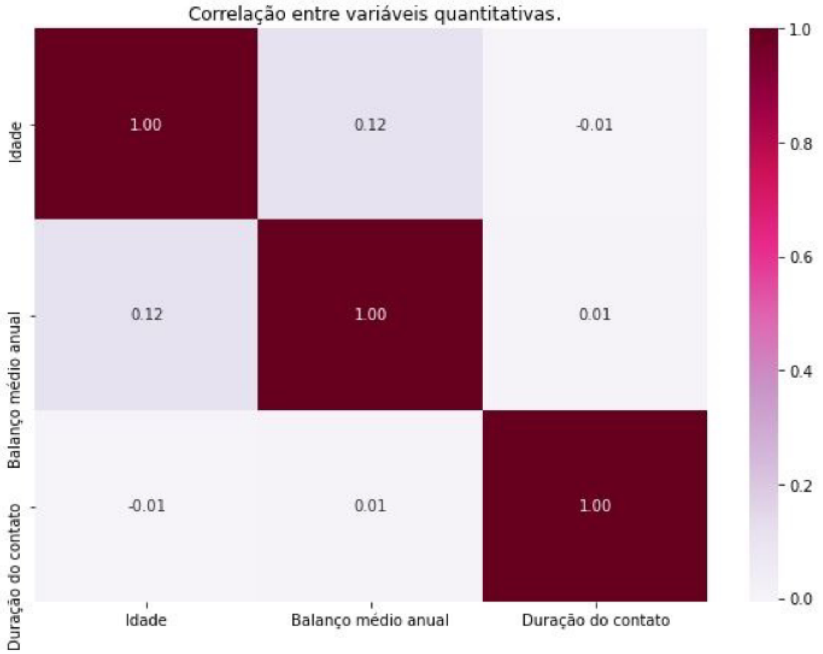

Figura 3.1: Matriz de correlação das variáveis Idade, Balanço médio anual e Duração do Contato.

Tabela 3.12: Proporção das categorias da variável Profissão.

Profissão	Proporção
Administrador(a)	12%
Advogado(a)	18%
Empreendedor(a)	3%
Doméstico(a)	2%
Cargo de gestão	22%
Aposentado(a)	7%
Autônomo(a)	3%
Serviços gerais	8%
Estudante	4%
Técnico(a)	16%
Desempregado(a)	3%

A Tabela 3.12 apresenta as proporções das categorias da variável Profissão presentes

na base, com destaque para o cargo de gestão (22%) e Doméstico(a) (2%).

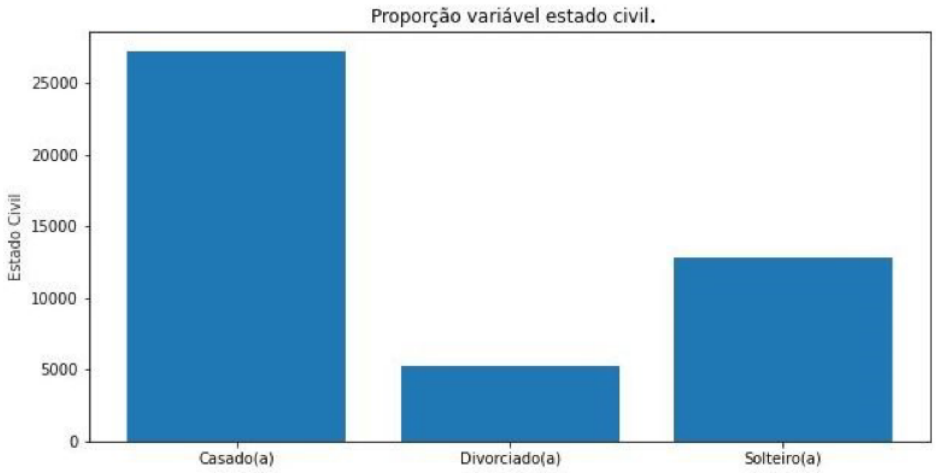

Figura 3.2: Gráfico de barras davariável estado civil.

No Gráfico 3.2 observamos que 57% dos clientes são casados.

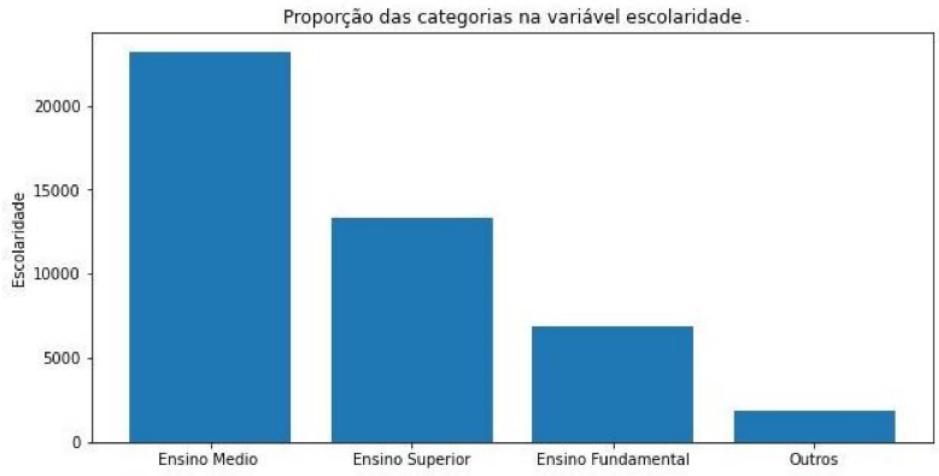

Figura 3.3: Gráfico de barras da variável escolaridade.

O Gráfico 3.3 apresenta as proporções das categorias da variável Escolaridade, a menor proporção é de clientes com apenas o ensino fundamental e a maior é de clientes que fizeram até o ensino médio. A categoria outros é composta por clientes que não forneceram essa informação.

Figura 3.4: Gráfico de barras da variável empréstimo em atraso.

Na variável Empréstimo, 98% não atrasaram o pagamento (Gráfico

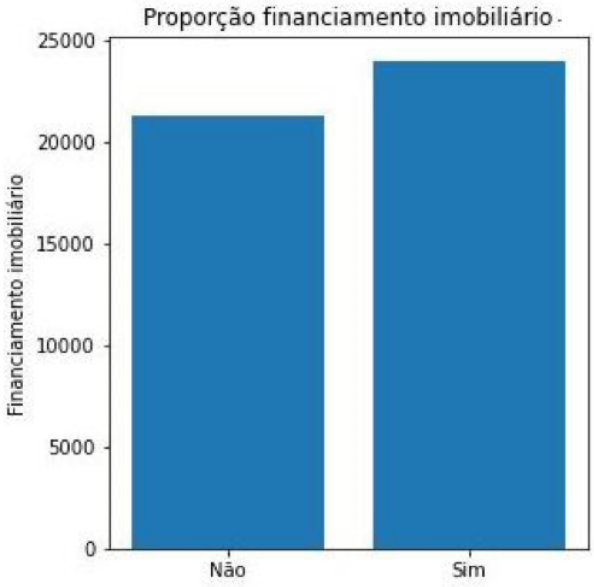

Figura 3.5: Gráfico de barras da variável financiamento imobiliário.

Com relação a variável Financiamento Imobiliário, 53% não possuem(Gráfico 3.5).

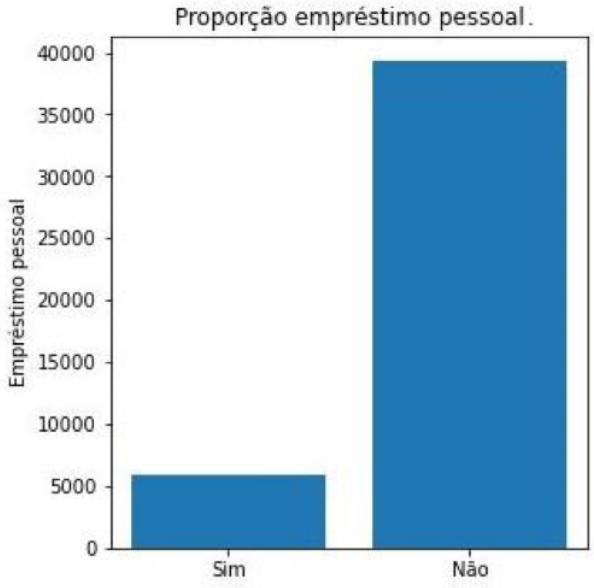

Figura 3.6: Gráfico de barras da variável empréstimo pessoal.

O gráfico 3.6 apresenta as proporções das categorias da variável Empréstimo Pessoal, 87% dos clientes não fizeram empréstimo pessoal.

Figura 3.7: Gráfico de barras da variável comunicação da campanha.

Na variável Meio de comunicação da campanha, a maior parte dos clientes foi contactado através do celular (Gráfico 3.7).

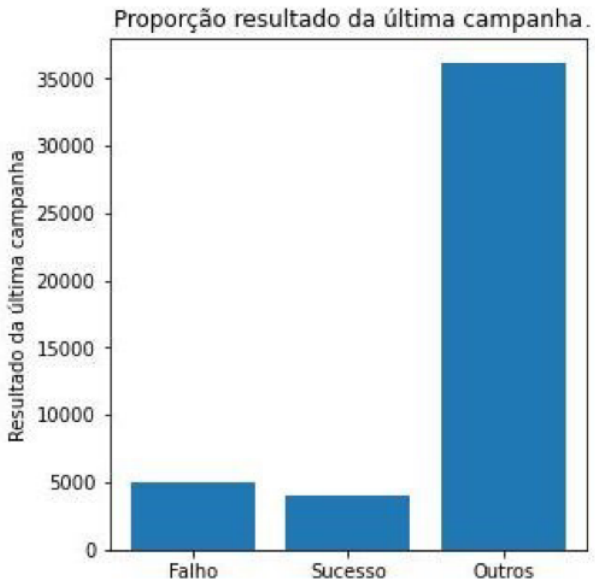

Figura 3.8: Gráfico de barras da variável resultado da última campanha.

No Resultado da última campanha, se obteve sucesso em apenas 9% dos clientes (Gráfico 3.8).

O conjunto de dados foi separado em 70% para treino e 30% para teste. Assim como no conjunto de dados que foi simulado, para as métricas de performance dos modelos foi utilizado o ponto de corte 0,5 e nenhum método de seleção de variáveis.

Para a metodologia de Redes Neurais, seguem os resultados. A Tabela 3.13 apresenta a matriz de confusão do modelo de Redes Neurais, 2.096 observções foram classificadas corretamente.

Tabela 3.13: Matriz de confusão para o modelo de Redes Neurais.

Predito	Observados		Total
	Fracasso	Sucesso	
Fracasso	25,18%	25,21%	50,39%
Sucesso	8,73%	40,88%	49,61%
Total	33,91%	66,09%	100%

As acurácias do modelo apresentadas na Tabela 3.14 mostra que não ocorreu uma

diminuição no resultado dessa métrica nos dados de teste quando comparado ao resultado nos dados de treino, portanto, não temos indicativo de *overfitting*.

Tabela 3.14: Desempenho preditivo do modelo de Redes Neurais.

	Redes Neurais
Acurácia treino	0,660
Acurácia teste	0,669
Especificidade	0,824
Sensibilidade	0,499
Tempo de processamento	4Minutos 16Segundos

Para a metodologia de Árvore de Classificação, seguem os resultados:

A Tabela 3.15 apresenta a matriz de confusão do modelo de Árvore de Classificação, 2.317 observações foram classificadas corretamente.

Tabela 3.15: Matriz de confusão para o modelo de Árvore de Classificação.

Predito	Observados		Total
	Fracasso	Sucesso	
Fracasso	36,37%	14,07	50,43%
Sucesso	12,92%	36,65%	49,57%
Total	49,28	50,72	100%

As acurácias do modelo apresentadas na Tabela 3.16, mostra que não ocorreu uma diminuição no resultado dessa métrica nos dados de teste quando comparado ao resultado nos dados de treino, portanto, não temos indicativo de *overfitting*.

Tabela 3.16: Desempenho preditivo do modelo de Árvore de Classificação.

	Árvore de Classificação
Acurácia treino	0,751
Acurácia teste	0,744
Especificidade	0,754
Sensibilidade	0,733
Tempo de processamento	48 Segundos

Para a metodologia de Regressão Logística, seguem os resultados:

A Tabela 3.17 apresenta os coeficientes do modelo de Regressão Logística, gerado com todas as variáveis já descritas.

As variáveis: Profissão(aposentado), Profissão(estudante), Estado Civil(Casado), Financiamento Imobiliário, Empréstimo Pessoal, Contato(Celular), Contato(Telefone), Resultado(Falho), Resultado(Sucesso), Balanço Médio Anual e Duração do Contato, foram significativas para o modelo ao nível de 5% de significância. O intervalo de confiança dos coeficientes, últimas duas colunas da Tabela 3.17, não possui muita variabilidade, logo, a estimação é precisa.

A Tabela 3.18 apresenta a matriz de confusão do modelo de Regressão Logística, 2.552 observções foram classificadas corretamente.

As acurácias do modelo apresentadas na Tabela 3.19, mostra que não ocorreu uma diminuição no resultado dessa métrica nos dados de teste quando comparado ao resultado nos dados de treino, portanto, não temos indicativo de *overfitting*.

Tabela 3.17: Coeficientes do modelo gerado com o conjunto de treino.

	Estimativas	P[$Z \geq \|z\|$]	0,025	0,975
Intercepto	-4,1451	0,000	-4,632	-3,658
Profissão[‘Administrador’]	0,3753	0,099	-0.071	0.821
Profissão[‘Advogado’]	-0.0226	0.921	-0,467	0,422
Profissão[‘Empreendedor’]	-0,1209	0,625	-0,606	0,364
Profissão[‘Domestico’]	-0,1584	0,528	-0,650	0,334
Profissão[‘Gestao’]	0,1303	0,564	-0,313	0,573
Profissão[‘Aposentado’]	0,7547	0,001	0,301	1,208
Profissão[‘Autonomo’]	-0,0208	0,931	-0,493	0,451
Profissão[‘Serviços Gerais’]	0,0553	0,811	-0,398	0,509
Profissão[‘Estudante’]	0.8888	0.000	0,421	1,357
Profissão[‘Técnico’]	0,0802	0,722	-0,362	0,523
Estado Civil[‘Divorciado’]	-0,1684	0,010	-0,296	-0,041
Estado Civil[‘Casado’]	-0,3408	0,000	-0,428	-0,254
Educação[‘Ensino Fundamental’]	-0,2834	0,005	-0,481	-0,085
Educação[‘Ensino Médio’]	-0,1112	0,209	-0,285	0,062
Educação[‘Ensino Superior’]	0,1184	0,204	-0,064	0,301
Empréstimo em atraso	-0,1845	0,251	-0,500	0,131
Financiamento Imobiliário	-0,7509	0,000	-0,828	-0,674
Empréstimo Pessoal	-0,5646	0,000	-0,679	-0,451
Contato[‘celular’]	1,1468	0,000	1,033	1,260
Contato[‘telefone’]	1,0046	0,000	0,832	1,177
Resultato[‘Falho’]	0,3871	0,000	0,281	0,493
Resultado[‘sucesso’]	2,6827	0,000	2,558	2,807
Idade	0,0014	0,527	-0,003	0,006
Balanço médio anual	1,834e-05	0,000	8,89e-06	2,78e-05
Duração do contato	0,0040	0,000	0,004	0,004

Tabela 3.18: Matriz de confusão para o modelo de Regressão Logística.

Predito	Observados		Total
	Fracasso	Sucesso	
Fracasso	40,91%	9,48%	50,39%
Sucesso	10,09%	39,52%	49,61%
Total	51%	49%	100%

Tabela 3.19: Desempenho preditivo do modelo de Regressão Logística.

	Regressão Logística
Acurácia treino	0,805
Acurácia teste	0,804
Especificidade	0,796
Sensibilidade	0,811
Tempo de processamento	76 Segundos

Capítulo 4

Conclusão e Discussão

O objetivo principal neste estudo é fazer uma comparação entre o método de Regressão Logística com os métodos Redes Neurais e Árvore de Classificação e verificar possíveis fatores que podem interferir na performance dos modelos, cuja variável resposta é binária.

No conjunto de dados simulados, ao analisar as métricas obtidas pelos modelos de Regressão Logística e Redes Neurais, verificamos que as duas técnicas obtiveram resultados muito similares entre si. O modelo de Regressão Logística classificou corretamente 120.642 mil observações das 150.000 mil contidas na base de teste, enquanto que o modelo de Redes Neurais classificou corretamente 123.090 mil, apenas 2% a mais do que Regressão Logística. O tempo de processamento, via Redes Neurais por sua vez, foi mais de 4 vezes maior, embora não tenha levado um tempo consideravelmente grande (aproximadamente 9 minutos).

Ao comparar Regressão Logística com Árvore de Classificação, as métricas já se distanciam. Árvore de classificação classificou corretamente 107.673 mil das 150.000 observações da base de teste, aproximadamente 10% a menos do que Regressão Logística, com um tempo de processamento duas vezes maior. Além disso, a métrica sensibilidade, no modelo de Árvore de Classificação é igual a 0,685 enquanto que no modelo de Regressão Logística, ficou em 0,801.

Já em relação ao conjunto de dados reais, o modelo de Regressão Logística classificou corretamente 2.552 observações das 3.173 da base de teste, enquanto que o modelo de Redes Neurais, classificou corretamente 2.096, 17% a menos. Outro ponto a se destacar, foi a métrica Sensibilidade, em que no modelo de Regressão Logística é igual a 0,811, e no de Redes Neurais é igual a 0,499, quase 40% menor.

Quando comparado o modelo de Árvore de Classificação, o modelo de Regressão

Logística também apresentou vantagem, não tanta quanto em relação ao modelo de Redes Neurais. A diferença nos acertos nas classificações ficou em torno de 10% maior no modelo de Regressão Logística, que também apresentou leve vantagem quando levada em consideração a métrica sensibilidade, com um valor 5% maior e com tempo de processamento muito parecido.

No geral, os três algoritmos apresentaram bons resultados em ambas as bases, mas uma vez que o objetivo também é classificar corretamente os clientes, tanto as métricas de Sensibilidade quanto a de Especificidade são muito importante e o modelo de Árvore de Classificação se mostrou inferior aos outros neste caso.

Dada a diferença entre as bases, os três algoritmos foram mais eficazes no conjunto de dados com volumetria maior. A volumetria dos dados parece ter influenciado no resultados dos modelos, já que na base menor, o modelo de Regressão Logística teve desempenho superior aos modelos de aprendizado de máquina Redes Neurais e Árvore de Classificação.

5. Referências Bibliográficas

Ambrósio, P. E. *Redes neurais artificiais no apoio ao diagnóstico diferencial de lesões intersticiais pulmonares.* Tese (Doutorado), Faculdade de Filosofia, Ciências, e Letras de Ribeirão Preto, USP, Ribeirão Preto, SP, 2002.

Beale, R.; Jackson, T. *Neural Computing-an introduction.* Taylor & Francis, CRC Press, NY, 1990.

Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. Classification and regression trees. wadsworth int. *Group*, v. 37(15), p. 237–251, 1984.

Casella, G.; Berger, R. L. *Statistical inference.* Cengage Learning, 2021.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, v. 2(4), p. 303–314, 1989.

Diniz, C.; Louzada, F. Métodos estatísticos para análise de dados de crédito. In 6*th Brazilian Conference on Statistical Modelling in Insurance and finance*, volume 6, pages 1–124, Maresias, São Paulo, 2013.

Exame, R. Temos mais dados do que nunca: Como usá-los a nosso favor?, 2019. URL https://exame.com/carreira/dados-uso-favor/. último acesso em 12 de maio 2021.

Hosmer, D. W.; Lemeshow, S. *Applied Logistic Regression.* John Wiley, New York, 1^{st} ed., 2000.

James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An introduction to Statistical Learning: with Applications in R*, volume 112. Springer, 2013.

Leite, T. M. O Neurônio Artificial, 2016. URL https://medium.com/ensina-ai/redes-neurais-perceptron-multicamadas-e-o-algoritmo-backpropagation-/eaf89778f5b8. Último acesso em 8 de março de 2022.

McLachlan, G. J. *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons, 2004.

Minsky, M.; Papert, S. A. *Perceptrons: An introduction to Computational Geometry*. MIT press, 2017.

Nascimento, S. Árvores de Decisão (vídeo aula), 2016. URL https://www.youtube.com/channel/UC4nbGdTDlcNrbZGVt732lmQ. Último acesso em 20 de maio de 2021.

Nelder, J. A.; Wedderburn, R. W. M. Generalized linear models. *Journal of the Royal Statistical Society, Series A (General)*, v. 135(3), p. 370–384, 1972.

Nied, A. *Treinamento de redes neurais artificiais baseado em sistemas de estrutura variável com taxa de aprendizado adaptativa*. Tese (Doutorado), Programa de Pós-Gradua ção em Engenharia Elétrica da Universidade Federal de Minas Gerias, Belo Horizonte, MG, 2007.

Onoda, M. *Estudo sobre um algoritmo de árvores de decisão acoplado a um sistema de banco de dados relacional*. 2001. Dissertação (Mestrado), Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2001. 27, 2001.

Repository, U. M. L. Bank marketing data set, 2019. URL https://archive.ics.uci.edu/ml/datasets/bank+marketing. último acesso em 13 de agosto de 2021.

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65(6), p. 386–408, 1958.

Rosenblatt, F. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Laboratory, Inc., Buffalo, NY, 1961.

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

Silva, A. M. M. d. *Técnicas de Data Mining na aquisição de clientes para financiamento de Crédito Direto ao Consumidor-CDC*. 2012. Dissertação (Mestrado), Escola Superior de Agricultura Luiz de Queiróz, ESALQ, USP, Piracicaba, São Paulo, 2012.

Simon, H. *Neural networks: a comprehensive foundation*, volume 13. Cambridge University Press, 1999. ISBN 0023527817.

Souza, M. C. S. C. d. Escores de propensão: aplicações à Epidemiologia. Monografia para obtenção do título de Bacharel em Estatística, Departamento de Estatística, UFRGS, 2010.

Van Rossum, G.; Drake Jr, F. L. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995. URL https://www.python.org/.

Westreich, D.; Lessler, J.; Funk, M. J. Propensity score estimation: machine learning and classification methods as alternatives to logistic regression. *Journal of Clinical Epidemiology*, v. 63(8), p. 826, 2010.

Widrow, B.; Lehr, M. A. Perceptrons, Adalines, and Backpropagation. *Arbib*, v. 4, p. 719–724, 1995.

Apêndice A

Códigos utilizados no projeto

```
np.random.seed(7)

#Descrição banco de dados:
n=500000
#1 - Score
score_credito = 1000*np.random.beta(2.5,2.5,n).round(2)

#2 - limite cartão de crédito
aux = np.random.beta(1.2,4.8,n).round(2)
limite_credito = (20000*aux + 0.3*score_credito)

for i in range(0,n):
    if limite_credito[i] < 200:
        limite_credito[i] = 200

#3saldo conta
n1=round(n*0.8)
aux = [*np.random.exponential(10000,n1).round(2),*np.zeros(n-n1)]
saldo_conta = np.array(my_shuffle(aux))

#4- salário
renda = -9.17 + 0.23*saldo_conta + 0.4*limite_credito
```

```
#intercepto controlado pra começar em zero
renda = np.around(renda,2)

#5 - Genero (0=homem, 1=mulher)
genero1=['homem','mulher']
genero_prob=[0.55,0.45]
genero=np.random.choice(genero1, n, p=genero_prob)

#6 -Membro ativo
ativo1=[1,0]
ativo_prob=[0.52,0.48]
ativo=np.random.choice(ativo1, n, p=ativo_prob)

#7 - Tem conta em outro banco (0 não e 1 sim)
outro_banco1=[1,0]
outro_banco_prob=[0.3,0.7]
outro_banco=np.random.choice(outro_banco1, n, p=outro_banco_prob)

#8- Estimativa saldo bancario
n1=round(n*0.8)
aux = [*np.random.exponential(10000,n1).round(2),*np.zeros(n-n1)]
saldo_conta = np.array(my_shuffle(aux))

#9 - Escolaridade
escolaridade = []
prop_mulheres = [.1,.1,.51,.29]
prop_homens = [.1,.12,.48,.30]
niveis = ["ensino_fundamental_incompleto",
"Ensino_fundamental_completo_e medio_incompleto",
"Ensino_medio_completo_e_superior_incompleto",
"Ensino_superior_completo"]
```

```
nivel_mulher = np.random.choice(niveis, n, p=prop_mulheres)
nivel_homem = np.random.choice(niveis, n, p=prop_homens)

for i in range(0,n):
    if genero[i] == "mulher":
        escolaridade.append(nivel_mulher[i])
    else:
        escolaridade.append(nivel_homem[i])

#10- idade
idade = []
prop_mulheres = [.12,.71,.17]
prop_homens = [.14,.71,.15]
idade_classe= ["Jovem_ate_25_anos", "Adulto_de_26_a_59_anos",
"Idoso_acima_de_60_anos"]

idade_mulher = np.random.choice(idade_classe, n, p=prop_mulheres)
idade_homem = np.random.choice(idade_classe, n, p=prop_homens)

for i in range(0,n):
    if genero[i] == "mulher":
        idade.append(idade_mulher[i])
    else:
        idade.append(idade_homem[i])

for i in range(0,n):
    if idade[i] == "Jovem_ate_25_anos":
        escolaridade[i] =  np.random.choice(["ensino_fundamental_incompleto",
        "Ensino_fundamental_completo_e_medio_incompleto",
        "Ensino_medio_completo_e_superior_incompleto", "Ensino_superior_completo"],
        p=[0.1,0.25,0.45,0.20])
```

```
#11 - estado civil
estado_civil_categ = ['Casado', 'Solteiro', 'Divorciado', 'Viuvo']
estado_civil_prob = [.45, .45, .05, .05]
estado_civil = np.random.choice(estado_civil_categ, n,
p=estado_civil_prob)

for i in range(0,n):
    if idade[i] == "Jovem_ate_25_anos":
        estado_civil[i] =  np.random.choice(['Casado', 'Solteiro'],p=[0.25,0.75])
    elif idade[i] == "Idoso_acima_de_60_anos":
        estado_civil[i] =  np.random.choice(estado_civil_categ,p=[0.4,0.05,0.3,0.25])

#12- Usa ou não cartão de crédito - 1 se usa, 0 se não

cart_credito = []
cont = np.random.binomial(1,0.7,n)
for i in range(0,n):
    if ativo[i] == 1:
        cart_credito.append(cont[i])
    else:
        cart_credito.append(0)




#Variável resposta - 0 bom pagador 1 mau pagador
y= np.random.binomial(1,0.5,n)




#Gerando a variável resposta Y
```

```
beta=[-0.2,-1.2, 0.16, 1.4, 1.6,0.3,1.4 ,0.8 ,-1.37, 1.37,
            0.5, 0.45, 0.22, 0.1, -0.2, -0.5, -0.231, -0.0211,
            0.281, -0.7,
                0.455, -0.325, 0.214, -0.156, -0.246,
                -0.201, 0.343,
                 0.0118, -0.160,-1.3, 0.2, -0.3, -0.4,
                 -0.2,-0.3,
                 -0.2,-0.3,0.1,
                 0.2,0.20,0.01,
     0.4,-0.3,0.2,0.4,0.8]
soma = []
for i in range(0,n):
    coef = beta*df.iloc[i,:]
    soma.append(round(sum(coef),4))

beta_0 = round(-2.15*np.mean(sum(coef)),4)

yy = []
for i in range(0,n):
    pi = 1/(1 + np.exp(-(beta_0+soma[i])))
    pi.round(5)
    yy_i = np.random.binomial(1,pi)
    y.append(yy_i)

np.mean(yy)



#chamando bibliotecas

import pandas as pd
import numpy as np
```

```
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_curve
from sklearn import tree
from sklearn import metrics
import time
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve, classification_report,\
                            accuracy_score, confusion_matrix, auc


#A variável resposta não foi declarada porque ainda vai ser transformada em zero e um
#Tirei pdays da base pq nao fazia sentido, campaign e previous também
data = pd.read_csv('bank-full.csv', sep = ';', dtype = { 'y':'category'})
#data.head

job_aux = pd.get_dummies(data['job'])
marital_aux = pd.get_dummies(data['marital'])
education_aux = pd.get_dummies(data['education'])
default_aux = pd.get_dummies(data['default'])
housing_aux = pd.get_dummies(data['housing'])
loan_aux = pd.get_dummies(data['loan'])
contact_aux = pd.get_dummies(data['contact'])
```

```
poutcome_aux = pd.get_dummies(data['poutcome'])

dic = {'job_admin':job_aux['admin.'],
       'job_blue_collar':job_aux['blue-collar'],
       'job_entrepreneur':job_aux['entrepreneur'],
       'job_housemaid':job_aux['housemaid'],
       'job_management':job_aux['management'],
       'job_retired':job_aux['retired'],
       'job_self_employed':job_aux['self-employed'],
       'job_services':job_aux['services'],
       'job_student':job_aux['student'],
       'job_technician':job_aux['technician'],
       'job_unemployed':job_aux['unemployed'],
       'EC_Divorciado':marital_aux['divorced'],
       'EC_Casado':marital_aux['married'],
       'Educação_Primaria':education_aux['primary'],
       'Educação_Secundaria':education_aux['secondary'],
       'Educação_terciaria':education_aux['tertiary'],
       'Flag_default':default_aux['yes'],
       'Flag_casa':housing_aux['yes'],
       'Flag_Loan':loan_aux['yes'],
       'Contato_celular':contact_aux['cellular'],
       'Contato_telefone':contact_aux['telephone'],
       'Resultato_Falho':poutcome_aux['failure'],
       'Resultado_other':poutcome_aux['other'],
       'Resultato_sucesso':poutcome_aux['success'],
       'Idade':data['age'],
       'Balanço_médio_anual':data['balance'],
       'Duração_do_contato':data['duration'],
       'y':data['y']}


dt = pd.DataFrame(dic)
```

```
dt

#Verificando valores ausentes
dt.isna().sum()

#Balanceando a base
df_aux=dt.query("y == 0")
df0 = df_aux.sample(n=5287)
df1 = dt.query("y == 1")
df = pd.concat([df0, df1])
df

dt_aux = df[['Idade', 'Balanço médio anual', 'Duração do contato']]
print(dt_aux.describe())

###Correlação das variáveis quantitativas
import seaborn as sns
plt.figure(figsize=(10, 7))
sns.heatmap(dt_aux[['Idade', 'Balanço médio anual', 'Duração do contato']].corr(),
            annot = True,
            fmt = '.2f',
            cmap='PuRd')
plt.title('Correlação entre variáveis quantitativas')
plt.show()

X = df[['job_admin','job_blue_collar','job_entrepreneur',
'job_housemaid','job_management',
'job_retired','job_self_employed',
        'job_services','job_student',
        'job_technician','job_unemployed', 'EC_Divorciado','EC_Casado',
        'Educação_Primaria',
        'Educação_Secundaria',
        'Educação_terciaria',
```

```
        'Flag_default','Flag_casa', 'Flag_Loan', 'Contato_celular',
        'Contato_telefone', 'Resultato_Falho', 'Resultado_other',
        'Resultato_sucesso', 'Idade', 'Balanço_médio_anual',
        'Duração_do_contato']]

y = df['y']


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7)

inicio_rl = time.time()
modelo_logist = LogisticRegression(C = 0.5, random_state = 42, max_iter = 1000)
modelo_logist.fit(X_train, y_train.ravel())
lr_predict_test = modelo_logist.predict(X_test)
fim_rl = time.time()

print(fim_rl - inicio_rl)
print(modelo_logist.coef_)

#Testando a performance do modelo
modelo_logist_train = modelo_logist.predict(X_train)
metrics.accuracy_score(y_train, modelo_logist_train)

#Performance no teste
modelo_logist_test = modelo_logist.predict(X_test)

#Y Chapeu
predictions = modelo_logist.predict_proba(X_test)
predictions = predictions[:,1]
predictions

#Matriz de confusão Logística
```

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, modelo_logist_test))

metrics.accuracy_score(y_test, modelo_logist_test)

##'sgd'
##quantidade de neuronio na camada escondida -
terá duas camada escondida com 3 neurônios
inicio_rn = time.time()
clf = MLPClassifier(solver ='sgd',
hidden_layer_sizes =(3,2),
learning_rate_init = 0.01,
                    activation = 'logistic', max_iter=1500,random_state = 1)

clf.fit(X_train, y_train.ravel())
fim_rn = time.time()
print(fim_rn - inicio_rn)

#Testando a performance do modelo
modelo_resdes_train = clf.predict(X_train)
metrics.accuracy_score(y_train, modelo_resdes_train)

clf_predict_test = clf.predict(X_test)

#Performance no teste
modelo_clf_test = clf.predict(X_test)
metrics.accuracy_score(y_test, modelo_clf_test)

#Y Chapeu
predictions_rn = clf.predict_proba(X_test)
predictions_rn = predictions_rn[:,1]
predictions_rn
```

```
#Matriz de confusão redes
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, modelo_clf_test))
print(metrics.classification_report(y_test, modelo_clf_test, labels = [1,0]))

###Modelo árvore
inicio_arv = time.time()
model_arv = tree.DecisionTreeClassifier()
model_arv.fit(X_train, y_train.ravel())
fim_arv = time.time()
print(fim_arv - inicio_arv)

arv_predict_test = model_arv.predict(X_test)

#Performance no teste
modelo_arv_test = model_arv.predict(X_test)
#Y Chapeu
predictions_arv = model_arv.predict_proba(X_test)
predictions_arv = predictions_arv[:,1]
predictions_arv

#Matriz de confusão Logística
print(confusion_matrix(y_test, modelo_arv_test))
print(metrics.classification_report(y_test, modelo_arv_test, labels = [1,0]))

arv_predict_test = model_arv.predict(X_test)
metrics.accuracy_score(y_test, arv_predict_test)

#Testando a performance do modelo
modelo_arv_train = model_arv.predict(X_train)
metrics.accuracy_score(y_train, modelo_arv_train)
```