Instituto de MATEMÁTICA E ESTATÍSTICA UFRGS

DEPARTAMENTO DE ESTATÍSTICA
E-mail: dest@mat.ufrgs.br

Trabalho de Conclusão de Curso

Tempo de carregamento de páginas web e fatores associados: aplicação de métodos de aprendizado de máquina supervisionados

Victor Frank Gomes

16 de maio de 2022

Victor Frank Gomes

Tempo de carregamento de páginas web e fatores associados: aplicação de métodos de aprendizado de máquina supervisionados

Trabalho de Conclusão apresentado à comissão de Graduação do Departamento de Estatística da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do título de Bacharel em Estatística.

Orientador: Prof. Dr. João Henrique Ferreira Flores

Porto Alegre
Maio de 2022

Victor Frank Gomes

Tempo de carregamento de páginas web e fatores associados: aplicação de métodos de aprendizado de máquina supervisionados

Este Trabalho foi julgado adequado para obtenção dos créditos da disciplina Trabalho de Conclusão de Curso em Estatística e aprovado em sua forma final pela Orientador e pela Banca Examinadora.

Orientador:_____
Prof. Dr. João Henrique Ferreira Flores, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul, Porto Alegre, RS

Banca Examinadora:

Prof. Dr. João Henrique Ferreira Flores, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul, Porto Alegre, RS

Prof. Dr. Rodrigo Citton Padilha dos Reis, UFRGS
Doutor pela Universidade Federal de Minas Gerais, Belo Horizonte, MG

Porto Alegre
Maio de 2022

Resumo

Com a propagação de lojas de comércio eletrônico e a democratização tecnológica, as empresas estão buscando meios de aprimorar o conteúdo que disponibilizam *online* a fins de oferecer uma melhor experiência ao usuário final e, consequentemente, obter maior lucro. Neste contexto, a velocidade de carregamento das páginas é tida como fator relevante de retenção de clientes, sendo utilizada também como um dos critérios de colocação de um *site* perante os motores de pesquisa. Atualmente, diversas soluções computacionais foram criadas para capturar métricas de performance das sessões de navegação a fim de auxiliar no processo de otimização destas.

Com isso em mente, este trabalho propõe-se a identificar os principais fatores vinculados ao tempo de carregamento de páginas *web* de um banco de dados de alta dimensionalidade extraído de uma destas soluções, composto por registros individuais de carregamento de páginas pertencentes ao domínio virtual de uma empresa multinacional de tecnologia. O banco em questão foi analisado através da Regressão Linear Múltipla e seus resultados foram posteriormente comparados à técnicas de *Machine Learning* como Redes Neurais Artificiais e o algoritmo `xgBoost`. Em termos preditivos, os resultados mostram que o modelo `xgbDART`, uma variação do algoritmo `xgBoost` apresentou melhor desempenho em todos os indicadores de avaliação. Contudo, o objetivo de encontrar as variáveis mais influentes foi plenamente atingido através da Regressão Linear Múltipla, que permite fácil interpretação dos resultados através da análise de seus coeficientes.

Palavras-Chave: Aprendizagem de Máquina, Redes Neurais, Regressão Linear, Performance de Páginas *Web*, Experiência do Usuário.

Abstract

With the spread of e-commerce stores and the technological democratization, companies are looking for ways to improve the content they make available *online* in order to offer a better user and end experience and, consequently, obtain greater profit. In this context, page loading speed is considered a relevant factor not only on customer retention, but it is also used as one of the criteria for placing a *site* before search engines. Currently, several computational solutions have been created to capture performance metrics of browsing sessions in order to assist in their optimization process.

With that in mind, this work aims to identify the main factors linked to the loading time of *web* pages from a high-dimensional database extracted from one of these solutions, composed of individual page loading times belonging to to the virtual domain of a multinational technology company. The database was analyzed using Multiple Linear Regression and its results were later compared to *Machine Learning* techniques such as Artificial Neural Networks and the `xgBoost` algorithm. In predictive terms, the results show that the `xgbDART` model, a variation of the `xgBoost` algorithm, performed better in all evaluation indicators. However, the objective of finding the most influential variables was fully achieved through the Multiple Linear Regression, which allows easy interpretation of the results through the analysis of it's coefficients.

Keywords: Machine Learning, Neural Networks, Linear Regression, *Web Performance*, *User Experience*.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

DNS *Domain Name System*

DOM *Document Object Model*

MAE *Mean Absolute Error*

MLP *Multilayer Perceptron*

ML *Machine Learning*

PCA *Principal Component Analysis*

PLT *Page Load Time*

RMSE *Root mean squared error*

UX *User Experience*

1 Introdução

1.1 Comentários Iniciais

Com base no conceito de *User Experience* (UX), busca-se entender a interação entre o cliente e um produto ou sistema com o qual ele interaja, objetivando otimizá-la. Na perspectiva das empresas de tecnologia, mais especificamente no design de páginas *web*, a UX se propõe a implementar soluções tanto em termos de forma (aparência dos produtos) quanto em funcionalidades disponíveis em seus aplicativos, páginas e serviços. Para (Hartson e Pyla, 2012), a experiência do usuário é composta de cinco qualidades distintas: utilidade, funcionalidade, usabilidade, persuasão e *design* gráfico.

Neste contexto, a velocidade de carregamento de página é considerada um dos principais indicadores em termos de funcionalidade que influenciarão a satisfação do usuário: conforme mais lentas são as páginas, maiores são as chances do usuário encerrar ou desistir de sua sessão de navegação. Desde 2010, a Google anunciou que começaria a considerar a velocidade de carregamento das páginas durante o processo de ranqueamento das páginas encontradas pelo seu motor de pesquisa (Singhal e Cutts, 2010). Páginas mais lentas também acarretam em menores taxas de conversão (quando uma sessão resulta em ao menos um pedido ou ordem de compra) para sites de comércio eletrônico, como também menores taxas de engajamento, em casos de sites de notícias/geração de conteúdo (Bojan et al., 2020).

1.1.1 Análise de Performance *Web*

Quando falamos de performance no contexto de páginas *web*, estamos dentre outras coisas nos referindo principalmente ao tempo de carregamento dessa página ou conjunto de páginas. O tempo de carregamento nada mais é que o tempo levado para que todo o conteúdo da página seja carregado e exibido no navegador, normalmente medido em segundos. Deste modo, ao analisar diferentes registros de carregamentos de página, buscamos ponderar quais fatores mais explicam a variabilidade do tempo despendido neste processo.

Métricas dessa mesma natureza já foram avaliadas por outros artigos em diferentes formas: (Fetouni, 2018) propõe uma análise correlacional com foco em taxas de conversão, já (Manhas, 2013) traz a influência em termos de velocidade de carregamento relacionados aos diferentes tipos de recursos carregados nas páginas. No primeiro trabalho, por lidar com uma variável resposta de natureza binária, o autor optou por ajustar uma regressão logística que melhor modelasse seu conjunto de

dados, trazendo em sua análise diferentes aspectos que pudessem explicar as taxas de conversão, como horário de acesso, número de ações tomadas pelos clientes durante a navegação e tempo aí despendido. Já o segundo trabalho focou-se em fazer a análise das estatísticas descritivas dos recursos carregados por páginas, fazendo ligação com boas práticas existentes de criação de páginas *web*, bem como trazer recomendações para o desenvolvimento destas.

Dada a especificidade do tema em questão, que trata-se também de um problema de negócio, é válido mencionar que encontrar resultados imediatamente comparáveis torna-se um desafio.

1.1.2 *Page Load Time*

Conforme mencionado, os tempos de carregamento de páginas podem ser compreendidos através de uma única métrica, denominada *Page Load Time* (PLT). Essa métrica representa o tempo tomado entre os diversos componentes que fazem parte do processo de carregamento. Tais componentes são ilustrados na Figura (1.1) a seguir:

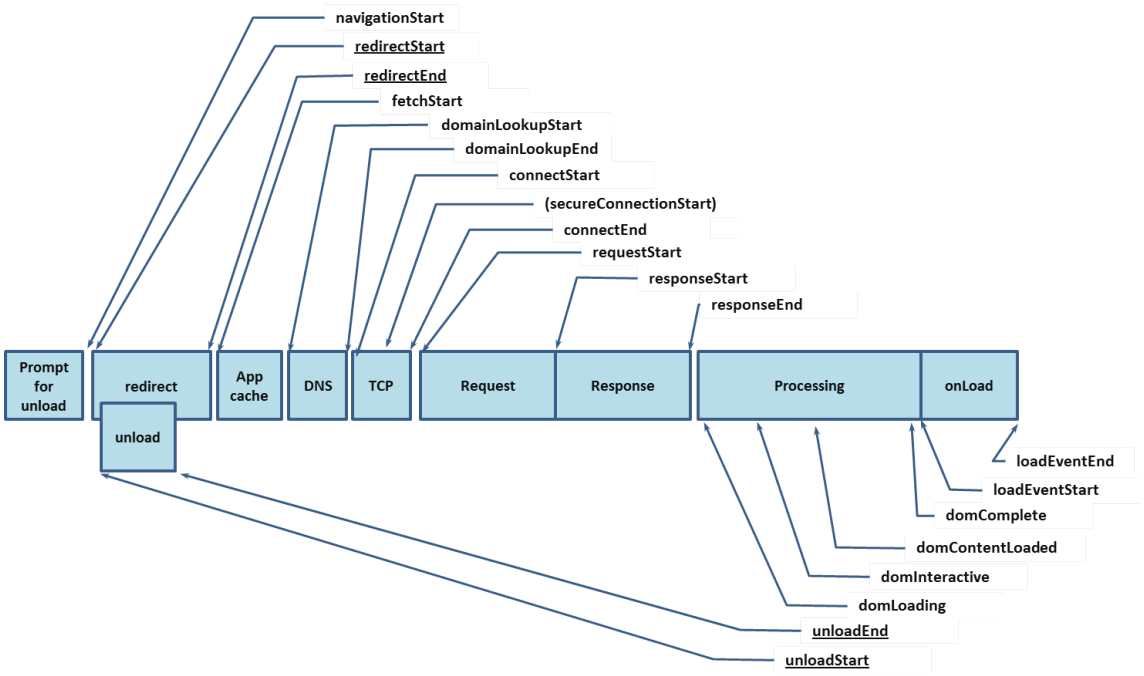


Figura 1.1: Diagrama ilustrando ciclo de vida de um carregamento de página
Fonte: (Wang, 2012)

De maneira resumida, os elementos acima ilustrados medem o tempo tomado para o carregamento inicial de uma página, não incluindo o tempo levado por quaisquer novas ações desencadeadas pelo usuário, exceto as que de fato provoquem o carregamento de uma nova página. Além disso, os componentes dentro do ciclo de vida de carregamento da página podem ser agrupados em três categorias: interações de rede, interações *backend* e interações *frontend*. As interações de rede compreendem todas as requisições feitas durante o carregamento que estão ligadas a transferência de conteúdo entre o domínio e o navegador, como os cliques feitos pelo usuário, a resolução da página e seu *host* através do domain name system (DNS), e a transferência de dados de autenticação. As interações *backend* correspondem

as requisições ligadas à configuração da aplicação, como consultas à bancos de dados internos a esta, requisição de arquivos e parâmetros da página. Finalmente, as interações *frontend* correspondem ao carregamento dos demais *scripts/snippets* e renderização de todos os elementos gráficos da página existentes no *document object model* (DOM).

1.1.3 Justificativa

É de interesse da companhia extrair informações relacionadas à experiência do cliente, para que esta possa se posicionar de maneira a oferecer uma navegação mais rápida e consequentemente obter melhores resultados; assim sendo, uma solução capaz de coletar tais dados foi implementada nas páginas dentro de seu domínio virtual. Dispondo do acesso a este grande conjunto de dados, é de intuito da pesquisa aqui apresentada analisar a eficácia de diferentes métodos supervisionados de *Machine Learning* que possam auxiliar a definir quais variáveis mais têm influência sobre o tempo de carregamento dessas páginas, trazendo com estes resultados, possíveis novas perspectivas no processo de tomada de decisão acerca destes dados.

Além de ser uma aplicação de métodos de *Machine Learning*, a pesquisa traz benefício também ao expandir a literatura disponível nas áreas de *User Experience*, *Machine Learning* e *Real User Monitoring*. Assim sendo, o caso de uso provido nesta pesquisa é de interesse não só de empresas da área de Tecnologia da Informação mas qualquer companhia que possua páginas *web* a sua disposição e que busque melhorar a experiência online de seus usuários. Finalmente, o trabalho ajudará também a expor o quão adequadas as técnicas utilizadas são quando se depararam com conjuntos de dados de alta dimensionalidade.

1.1.4 Objetivo Geral

O objetivo principal do trabalho é de aplicar modelos analíticos de *Machine Learning* para selecionar as variáveis mais influentes sobre o tempo de carregamento das páginas.

1.1.5 Objetivos Específicos

- Definir qual(is) característica(s) mais impactam o tempo de carregamento usando diferentes abordagens como Regressão Linear, *Feature Selection*, entre outras;
- Comparar resultados preditivos dos algoritmos utilizados para contextualização.

1.1.6 Hipótese de Pesquisa

A pesquisa tem como hipótese a relação existente entre os dados auxiliares durante as navegações de páginas e o seu tempo de carregamento. Assim sendo, esperamos poder extrair através dos métodos empregados as características que explicam maior a variabilidade dessa métrica, para auxiliar na tomada de decisão ligada ao desenho/construção das páginas.

1.1.7 Fontes de dados

Os dados utilizados na pesquisa foram capturados durante sessões de navegação de usuários nos domínios *web* da empresa, ou seja, todos os websites ligados a vendas que esta possui, por meio de uma solução de código aberto implementada no código fonte de suas páginas. Estes foram obtidos pelo autor, funcionário ativo da empresa e refletem inicialmente 108.081 registros de carregamento de páginas, representando o período de 21 de Novembro de 2021 a 1 de Dezembro de 2021.

2 Métodos

Neste capítulo são apresentadas as abordagens empregadas ao longo do trabalho. Inicialmente, define-se o método de Regressão Linear Múltipla (Seção 2.1), entrando depois na aprendizagem de máquina (Seção 2.2) os algoritmos supervisionados (Seção 2.3) e finalmente as Redes Neurais Artificiais (Seção 2.4). Ao fim do capítulo, descreve-se também o banco de dados utilizado em maior profundidade.

2.1 Regressão Linear Múltipla

A regressão linear múltipla é uma extensão da técnica de regressão linear simples, que objetiva modelar o comportamento da variável resposta através de uma equação da forma:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_j X_j + \epsilon \tag{2.1}$$

Onde X_j representa o j-ésimo preditor e β_j quantificará a associação entre aquela variável e a resposta. Assim, β_j pode ser interpretado como o efeito médio na variável resposta Y de um aumento unitário em X_j, mantendo os demais preditores constantes (James et al., 2013).

Como na regressão linear simples, os parâmetros são estimados usando o método de mínimos quadrados, escolhendo os coeficientes $\beta_0, \ldots, \beta_j$ que minimizem a soma dos quadrados dos resíduos ($SQRes$):

$$SQRes = \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_j x_{ij} \right)^2. \tag{2.2}$$

Desta forma, a regressão ajusta um hiperplano que minimiza a soma dos quadrados das distâncias verticais entre as observações e o plano em si. A relação entre os preditores e a resposta então pode ser avaliada através de um teste de significância.

A regressão ainda pode ser avaliada em termos de ajuste geral, através do coeficiente de determinação (R^2) e outros indicadores, como a *Root Mean Squared Error* (RMSE) e o *Mean Absolute Error* (MAE). Estes dois últimos indicadores tem o intuito de olhar a magnitude dos resíduos sob diferentes enfoques:

1. O RMSE mede a diferença entre a predição e o valor observado ao quadrado, onde o efeito de cada erro é proporcional ao tamanho do erro ao quadrado, potencializando o efeito de erros grandes no seu cálculo. Sua equação é dada por:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{2.3}$$

2. O MAE pode ser interpretado como a magnitude média dos valores absolutos erros do modelo ajustado. Sua equação é dada por:

$$\text{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n} = \frac{\sum_{i=1}^{n}|e_i|}{n} \tag{2.4}$$

Ambos indicadores variam no intervalo $[0, +\infty)$ e valores menores indicam melhor ajuste dos modelos.

2.1.1 Variance Inflaction Factor

O *Variance Inflaction Factor* (VIF) é uma medida calculada com o intuito de indicar multicolinearidade entre variáveis. Este efeito ocorre quando uma variável preditora num modelo de regressão múltipla pode ser linearmente predito a partir dos demais com certa acurácia, ou seja, as variáveis independentes apresentam relações exatas ou aproximadamente exatas entre si. Quando isto ocorre, um modelo multivariado indicará quão bem o conjunto de preditores consegue predizer a variável resposta, porém não será capaz de discernir a importância individual das variáveis independentes. Desta forma, a inclusão de preditores com um escore VIF alto não é recomendada.

O VIF é dado pela razão da variância de um preditor $\hat{\beta}_j$ quando ajustando o modelo completo dividido pela variância de $\hat{\beta}_j$ ajustado individualmente. O menor valor possível é 1, o que indica a ausência completa de colinearidade. Normalmente sempre existe uma quantidade pequena de colinearidade entre os preditores; de maneira geral, um VIF acima de 5 ou 10 indica uma quantia problemática d e colinearidade (James et al., 2013). O VIF para cada variável pode ser computado usando a seguinte fórmula:

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X-j}} \tag{2.5}$$

Onde $R^2_{X_j|X-j}$ é o R^2 da regressão de X_j sobre todos os demais preditores. Se $R^2_{X_j|X-j}$ é próximo de 1, então há colinearidade e consequentemente o VIF será grande.

2.2 Aprendizagem de Máquina

A aprendizagem de máquina (*Machine Learning*), uma área da inteligência artificial, pode ser definida como um programa/solução computacional capaz de ganhar experiência com respeito a um conjunto de tarefas e sua(s) respectiva(s) métricas de performance, onde sua eficiência com respeito a tais tarefas é melhorada através da experiência adquirida (Mitchell, 1997).

Atualmente, técnicas de aprendizagem de máquina tem ganhado ainda mais relevância dada a evolução em termos de capacidade de processamento dos computadores, bem como do crescimento no volume de dados disponíveis das mais diversas fontes, em conjunto com a chamada Internet das coisas, que interliga outros objetos físicos capazes de se conectarem a uma rede, transmitindo e recebendo dados. Os métodos dentro dessa área são comumente divididos em aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem semi-supervisionada, dependendo do conjunto de dados utilizado.

2.3 Aprendizagem Supervisionada

A aprendizagem supervisionada refere-se aos métodos que trabalham com conjuntos de dados que já possuem um vetor de resposta (Bell, 2020). Os dados de entrada são chamados de *features*, e os dados de saída são chamados de resposta ou de objetivo. Os métodos de aprendizagem supervisionada fazem a divisão dos dados entre o conjunto de treino, utilizado de fato para treinar o algoritmo iterativamente, e o conjunto de teste, utilizado para fins de validação.

2.3.1 *Cross-Validation*

O *Cross-Validation* é um método de amostragem que usa diferentes subconjuntos de um banco de dados para testar e treinar modelos, de maneira iterada. No contexto de predição, um modelo geralmente recebe um conjunto de dados conhecidos no qual o treinamento é executado (conjunto de treino), contra o qual este é testado (conjunto de teste). O objetivo do *Cross-Validation* é testar a capacidade do modelo de prever novos dados que não foram usados na estimativa, a fim de sinalizar problemas como *overfitting* ou viés de seleção (Cawley e Talbot, 2010), fornecendo uma visão de como o modelo se generalizará para um conjunto de dados a parte.

Ainda neste contexto, existe outra técnica de amostragem chamada *k-Fold Cross-Validation*. Segundo (James et al., 2013), esta abordagem consiste em dividir o conjunto de dados de maneira aleatória em k grupos, também chamado de *folds* ou dobras, de tamanhos aproximadamente iguais. A primeira dobra então é tratada como o conjunto de teste, e o modelo é então treinados nas $k-1$ dobras remanescentes. Desta forma, as medidas de ajuste, como por exemplo o RMSE, são computadas sobre as observações de cada dobra independente. Este procedimento é repetido k vezes; a cada iteração, um grupo diferente de observações é tratado como conjunto de teste, resultando em k estimativas do RMSE. A estimativa final da métrica com validação cruzada na dobra k é calculada fazendo a média deste valores, expressada pela equação a seguir:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} RMSE_i \tag{2.6}$$

2.3.2 Árvores de Decisão

Árvores de decisão são estruturas recursivas que expressam um processo sequencial de classificação. Na modelagem de problemas de regressão, segundo (James et al., 2013), as árvores dividem o espaço de variáveis preditoras em partições simples, que não se sobreponham. Desta forma, os valores preditos $(X_1, \ldots, X_n)$ serão dados em função dos valores médios de cada partição. O objetivo é encontrar partições $P_1, \ldots, P_n$ que minimizem a soma dos quadrados dos resíduos dada por:

$$\sum_{j=1}^{J}\sum_{i\in P_n}(y_i - \hat{y}_{P_n})^2, \tag{2.7}$$

onde $\hat{y}_{P_n}$ é a resposta média das observações do conjunto de treino dentro da j-ésima partição. Como esse processo é computacionalmente custoso, o algoritmo é

executado de maneira recursiva, reparticionando o espaço em dois novos nodos folha por sub-árvore, sempre minimizando a soma dos quadrados dos resíduos das novas partições, até que o processo atinja seu critério de parada. As árvores de decisão possuem algumas vantagens em relação aos métodos clássicos, pois conseguem lidar com preditores qualitativos sem a necessidade de variáveis *dummy*, além disso, são consideradas de fácil interpretação e explicabilidade por possuírem uma representação gráfica. No entanto, o método de árvores de decisão tradicional não possui o mesmo nível de acurácia preditiva nem robustez, onde pequenas mudanças nos dados utilizados pelo método acabam mudando a árvore ajustada fácilmente.

No entanto, usando este método como ponto de partida, foram derivadas, dentro dos chamados métodos *ensemble*, as técnicas de *Random Forest*, *Bagging* e *Boosting*. A ideia central destes métodos é melhorar as predições obtidas com as Árvores de Decisão, propondo diferentes modificações ao algoritmo original e melhorando a performance preditiva substancialmente (James et al., 2013).

2.3.3 *Random Forest*

São chamadas de *Random Forests* a combinação de árvores de decisão nas quais cada árvore depende dos valores de um vetor aleatório amostrado de forma independente fazendo uso da técnica de *bootstrap* e com a mesma distribuição para todas as árvores dentro do conjunto (Breiman, 2001). Em problemas de regressão, os preditores assumem valores numéricos ao invés de categóricos, como encontrados em problemas de classificação. As predições são então obtidas tomando a média dos resultados para as n árvores criadas.

Definindo p como o total de preditores, para cada amostra tomada na divisão nas árvores, o algoritmo mantém apenas uma parte $m = \sqrt{p}$ dos preditores originais.

2.3.4 *Boosting*

A abordagem de *Boosting*, no contexto de árvores de decisão, consiste basicamente em criar novas árvores a partir de árvores já ajustadas. Diferentemente do que acontece em uma *Random Forest*, onde os ajustes são baseados em amostras independentes dos dados obtidas através de *bootstrap*, este método ajusta cada árvore de decisão em uma versão modificada do conjunto de dados original.

Considerando um conjunto de árvores de decisão $\hat{f}^1, \ldots, \hat{f}^N$, um vetor de variáveis preditoras X e seus resíduos r, o *Boosting* aplica o seguinte algoritmo:

1. Defina $\hat{f}(x) = 0$ e $r_i = y_i$ para todo i no conjunto de treino.
2. Para $n = 1, 2, \ldots, N$, repetir:
   (a) Ajustar uma árvore de decisão $\hat{f}^n$ com d splits e $d + 1$ nodos terminais ao conjunto de treino (X, r).
   (b) Atualizar $\hat{f}$ adicionando uma versão reduzida da nova árvore:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^n(x) \tag{2.8}$$

(c) Atualizar os resíduos,

$$r_i \leftarrow r_i - \lambda \hat{f}^n(x_i) \tag{2.9}$$

3. Retornar o modelo atualizado, de tal forma:

$$\hat{f}(x) = \sum_{n=1}^{N} \lambda \hat{f}^n(x) \tag{2.10}$$

Por ajustar as árvores usando os resíduos atuais a cada passo, o algoritmo cria uma sequência de classificadores que são atualizados iterativamente, dando mais peso as observações que ainda não foram classificadas, porém, este processo pode ser mais lento se comparado a outras técnicas.

2.3.5 xgBoost

O algoritmo `xgBoost` é uma adaptação/melhoria em cima do método original proposto por (Friedman, 2001). A ideia central deste algoritmo é melhorar um modelo mais simples combinando-o com outros modelos mais simples, gerando um modelo coletivamente mais robusto. Trata-se de uma extensão do *Boosting* onde o processo de ajuste iterativo é formalizado através do algoritmo de gradiente descendente sobre uma função objetivo, minimizando o viés e o *underfitting*. O algoritmo genérico é dado pela seguinte forma, considerando um conjunto de treino $(x_i, y_i)_{i=1}^N$, uma função perda diferenciável $L(y, F(x))$ e uma taxa de aprendizagem α:

1. Inicializar o modelo com um valor constante:

$$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta) \tag{2.11}$$

2. Para $m = 1, \ldots, M$:

(a) Calcular os gradientes e hessianos:

$$\begin{aligned} \hat{g}_m(x_i) &= \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)} \\ \hat{h}_m(x_i) &= \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{(m-1)}(x)} \end{aligned} \tag{2.12}$$

(b) Ajustar os modelos usando o conjunto de treino $\left\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^N$ resolvendo o seguinte problema de otimização:

$$\begin{aligned} \hat{\phi}_m &= \arg\min_{\phi \in \Phi} \sum_{i=1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i)\right]^2. \\ \hat{f}_m(x) &= \alpha \hat{\phi}_m(x). \end{aligned} \tag{2.13}$$

3. Retornar

$$\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x) \tag{2.14}$$

A implementação computacional mais comumente utilizada nos dias de hoje foi proposta por (Chen e Guestrin, 2016), após otimizações em termos de processamento que garantiram maiores capacidades de paralelismo. A solução foi criada inicialmente para as linguagens `Python` e `R`, expandindo para diversas linguagens de programação por se tratar de código aberto.

2.4 Redes Neurais Artificiais

As redes neurais artificiais, ou simplesmente redes neurais, são uma arquitetura computacional, formadas por um conjunto de algoritmos cuja estrutura de camadas se assemelha ao cérebro humano, onde seu funcionamento busca replicar a transmissão sináptica entre neurônios (Haykin, 2008).

A estrutura de uma rede neural, observada na Figura (2.1), é formada por 3 camadas ou *layers*:

1. *Input Layer*: camada de entrada de dados
2. *Hidden Layer(s)*: camada(s) oculta(s) onde os cálculos são realizados e o modelo aprende as relações presentes nos dados
3. *Output Layer*: camada de saída do modelo

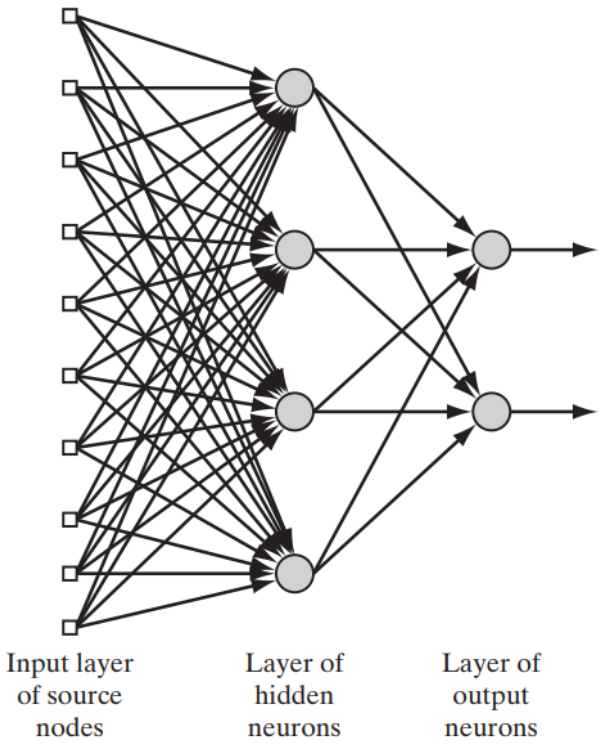


Figura 2.1: Estrutura de uma Rede Neural com 10 neurônios na camada de entrada, uma camada oculta com 4 neurônios e uma camada de saída com 2 neurônios
Fonte: (Haykin, 2008)

Existem 4 características que determinam os valores dos neurônios dentro de uma rede neural e influenciam o funcionamento desta: peso, *bias*, *input* e função de

ativação compõem o valor do neurônio. A função de ativação converte o somatório dos sinais de entrada para valores no intervalo $[0, 1]$ ou $[-1, 1]$, mais comumente, representando a intensidade das conexões. Os valores dos neurônios são então transmitidos às próximas camadas, até chegarem à camada de saída. Este processo se repete múltiplas vezes em conjunto com outras funções, determinando os pesos que melhor se ajustam às variáveis e explicam o conjunto de dados, minimizando o erro de saída do modelo. Podemos observar na Figura (2.2) a esquemática de um modelo não-linear para um único neurônio.

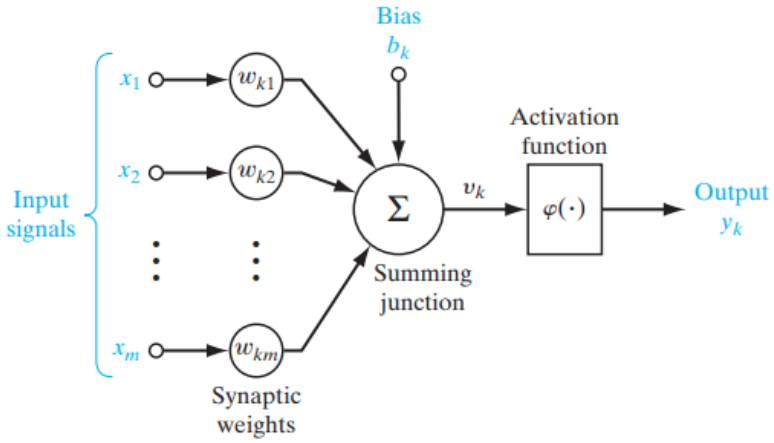


Figura 2.2: Modelo não-linear de um neurônio com m pesos
Fonte: (Haykin, 2008)

2.4.1 Funções de ativação

A função de ativação, denotada por $\varphi(v)$, define a saída de um neuron em termos de um campo local induzido v. Segundo (Haykin, 2008), existem diferentes tipos de função de ativação, que podem ser divididos em dois tipos básicos, a saber:

1. Função Limiar
   Este tipo de função possui a seguinte forma:

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \tag{2.15}$$

   Ou seja, a função é binária, determinando se um neurônio é ativado ou não. De maneira correspondente, a saída de um neurônio k é expressada de forma similar:

$$y_k = \begin{cases} 1 & \text{se } v_k \geq 0 \\ 0 & \text{se } v_k < 0 \end{cases} \tag{2.16}$$

   onde v_k é o campo local induzido do neurônio; isto é,

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{2.17}$$

2. Função Sigmoidal
   Sendo um balanço entre comportamento linear e não-linear, um exemplo de

função sigmoidal comumente utilizada em redes neurais é a função logística, definida por

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{2.18}$$

onde a é o parâmetro de inclinação da função sigmoidal. Ao contrário da função limiar, a função sigmoidal assume valores contínuos no intervalo $[0, 1]$. A função também é diferenciável.

Em problemas de regressão linear, objeto deste estudo, a função de ativação também é utilizada, costumeiramente na camada de saída. Como os neurônios são lineares, a saída é a mesma que o campo local do neurônio $v(i)$, ou seja

$$y(i) = v(i) = \sum_{k=1}^{M} w_k(i)x_k(i) \tag{2.19}$$

onde $w_1(i), w_2(i), \ldots, w_M(i)$ são os M pesos sinápticos do neurônio, medidos no tempo i.

2.4.2 *Multilayer Perceptron*

Segundo (Haykin, 2008) o *Multilayer Perceptron* (MLP) é uma rede neural com ao menos uma camada oculta, onde cada camada é conectada à subsequente de forma que o fluxo de sinal através da rede progrida em direção direta, da esquerda para a direita e camada por camada:

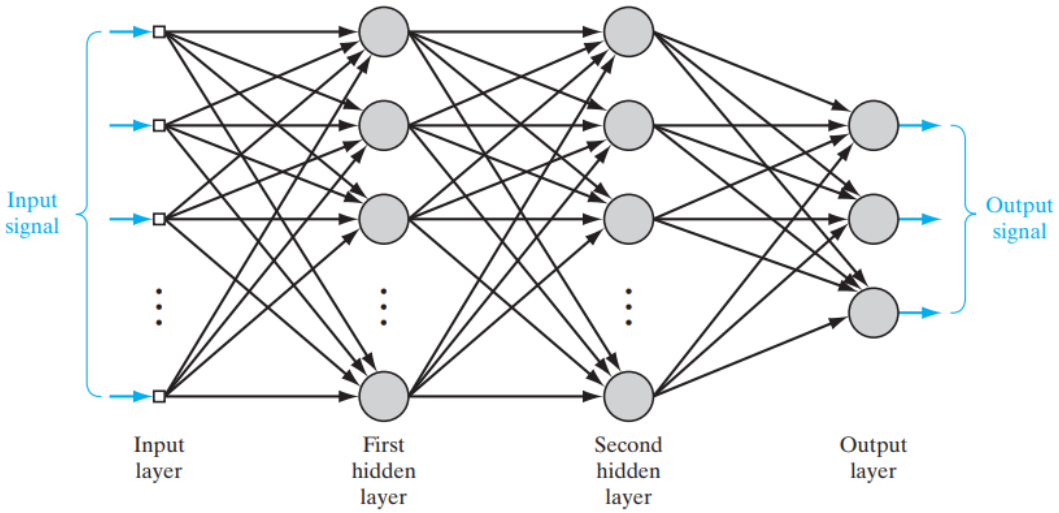


Figura 2.3: Estrutura de uma rede MLP de n neurônios nas camadas de entrada e duas camadas ocultas e 3 neurônios na camada de saída
Fonte: (Haykin, 2008)

Sendo a generalização de um *Single Layer Perceptron*, o objetivo da rede MLP é reduzir o erro preditivo, fazendo o ajuste dos pesos sinápticos durante a aprendizagem da rede neural através da utilização do algoritmo *backpropagation*. Em resumo, cada neurônio oculto ou de saída em uma arquitetura MLP objetiva realizar dois cálculos:

1. o cálculo do sinal de função que aparece na saída de cada neurônio, expresso como uma função não linear contínua do sinal de entrada e pesos sinápticos associados a esse neurônio;

2. o cálculo de uma estimativa do vetor gradiente (ou seja, os gradientes da superfície de erro em relação aos pesos conectados às entradas de um neurônio), que é necessário para enviar o sinal de erro para trás pelas camadas rede.

Com o subsídio teórico definido, passamos então a apresentação do banco de dados que será utilizado e suas características.

2.5 Banco de Dados

Neste trabalho dispomos de um banco de dados de dimensões 96.836×36, após a exclusão das entradas ligadas à tráfego sintético e tempos de carregamento inválidos (inferiores a 1 milissegundo) do banco original com 108.081 linhas, onde cada registro corresponde a um carregamento de página. Estes dados foram extraídos de um banco de dados interno da empresa, que é por sua vez alimentado diariamente com dados coletados através de um *script* criado através da linguagem de programação `JavaScript`, implementado nas páginas pertencentes ao domínio virtual da empresa. Assim que um usuário acessa uma das páginas da empresa, este *script* preenche um dicionário de dados chamado *beacon*, que é então transmitido aos bancos de dados internos da empresa após o fim das sessões de navegação. É importante notar que o *script* coleta apenas informações estatísticas não sensitivas das sessões de navegação. O fluxo de captura e extração dos dados pode ser resumido pela Figura (2.4) abaixo:

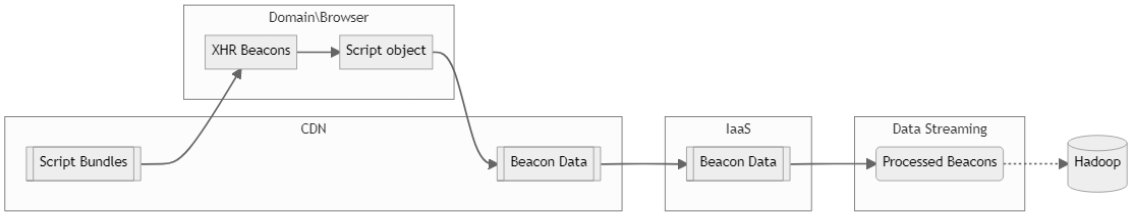


Figura 2.4: Diagrama de alto nível ilustrando fluxo de dados de navegação

O banco contém, além das informações relacionadas ao ciclo de vida do tempo de carregamento da página, como descrito na Figura (1.1), outras variáveis úteis de identificação, como tipo de aplicação, marcadores de tempo, tipo de dispositivo utilizado, sistema operacional e informações geográficas, tratadas na pesquisa como variáveis independentes. A variável dependente é dada pelo *Page Load Time* (PLT), que representa o tempo de carregamento final de uma página.

Para as análises presentes neste trabalho, as variáveis categóricas do banco foram reprocessadas através de *one-hot encoding*, onde cada categoria se torna uma coluna binária. Os tempos de navegação estão registrados em milissegundos. Para fins de clareza, foi disponibilizado na seção de apêndices o sumário do conjunto de dados, bem como seu dicionário de dados.

3 Análise dos dados de tempo de carregamento de páginas *web*

Neste capítulo apresentamos os resultados provenientes da aplicação das metodologias mencionadas previamente neste trabalho. As etapas de manipulação e análise de dados aqui descritas foram realizados fazendo uso do IDE *RStudio* e a linguagem de programação `R` (R Core Team, 2021), versão 4.0.4.

3.1 Análise Exploratória

Inicialmente foi realizada uma análise descritiva da variável resposta em alguns grupos, mantendo apenas as categorias de maior ocorrência nestes, com o intuito de expôr algumas características do banco em questão, a saber:

Tabela 3.1: PLT (ms) por Sistema Operacional

Sist. Oper.	Frequência (%)	Média	Mediana	Desvio Padrão	Mínimo	Máximo
Windows	72.729 (67,3)	3.172	2.187	3.864	98	236.856
Android	25.790 (23,9)	4.276	3.288	3.956	199	201.806
Mac OS X	3.377 (3,1)	2.275	1.688	2.170	132	53.120
Linux	2.964 (2,7)	2.723	2.131	4.850	265	229.925
Outros	3.221 (3,0)	3.221	3.381	2.309	183	46.554
Total	108.081 (100,0)	–	–	–	–	–

Comparando os sistemas operacionais da Tabela (3.1), vemos *Page Load Time* (PLT) médio menor, isto é, respostas mais rápidas em sistemas com menor frequência de acessos (Linux, Mac OS).

Tabela 3.2: PLT (ms) por País

País	Frequência (%)	Média	Mediana	Desvio Padrão	Mínimo	Máximo
US	42.384 (39,2)	3.088	2.227	3.321	174	79.559
BR	12.351 (11,4)	4.179	3.098	4.046	200	79.987
IN	9.937 (9,2)	4.120	2.963	4.740	138	236.856
CA	6.419 (5,9)	2.583	1.903	2.540	149	45.195
CN	5.127 (4,7)	3.581	2.556	3.880	147	52.296
Outros	31.863 (29,5)	5.229	4.323	4.225	98	229.925
Total	108.081 (100,0)	–	–	–	–	–

Comparando entre países dispostos na Tabela (3.2), podemos ver que Brasil e Índia estão equiparados em termos da posição central da distribuição de velocidade, porém o desvio padrão e valores extremos apresentam uma distribuição distinta entre estes dois países. Ainda em termos de velocidade vemos tempos de carregamento cada vez mais rápidos na China, Estados Unidos e Canadá, respectivamente. Estes resultados estão ligados à distância destes países com os servidores da empresa, bem como a variação das velocidades de banda larga dos usuários.

Tabela 3.3: PLT (ms) por Tipo de Dispositivo

Dispositivo	Frequência (%)	Média	Mediana	Desvio Padrão	Mínimo	Máximo
Desktop	83.591 (77,3)	3.157	2.185	3.855	98	236.856
Mobile	24.490 (22,7)	4.157	3.189	3.909	199	201.806
Total	108.081 (100,0)	–	–	–	–	–

A Tabela (3.3) nos mostra também que algumas diferenças em termos de velocidade ocorrem ao compararmos o tipo de dispositivo, onde dispositivos móveis apresentam em média 1 segundo a mais de lentidão em relação aos dispositivos de mesa. Isto sugere que os navegadores em dispositivos móveis, do mesmo modo que as páginas em sua versão móvel, não são tão otimizadas à nível de desenho e código.

Tabela 3.4: PLT (ms) por Navegador

Browser	Frequência (%)	Média	Mediana	Desvio Padrão	Mínimo	Máximo
Chrome	66.852 (61,9)	3.261	2.260	3.833	98	236.856
Edge	15.575 (14,4)	3.109	2.130	4.181	112	208.264
Chrome Mobile	14.337 (13,3)	3.693	2.823	3.599	199	201.806
Facebook	2.109 (2,0)	4.576	3.830	3.495	219	50.424
Outros	9.208 (8,5)	4.370	3.830	3.912	169	60.649
Total	108.081 (100,0)	——	——	——	–	–

Entre navegadores, notamos através da Tabela (3.4) pouca diferença em velocidade comparando o Google Chrome contra o Microsoft Edge, possivelmente por estes compartilharem seu código base, tendo sido desenvolvidos partindo do navegador *open-source* Chromium, escrito na linguagem de programação `C++`. Corroborando com a análise entre tipos de dispositivo, entre os navegadores notamos maior latência naqueles que são acessíveis através de dispositivos móveis.

Com a avaliação das tabelas acima, é possível traçar o perfil de usuário que normalmente visita as páginas da companhia: este faz o acesso das páginas através de um computador de mesa ou *notebook*, utilizando o sistema operacional Windows e o navegador Google Chrome, o que é coerente com as pesquisas encontradas amplamente pela internet, como exposta em (NetMarketshare Team, 2018).

Levando em consideração que o interesse da pesquisa está em encontrar quais variáveis dentro do controle da companhia mais impactam o tempo de carregamento, e que o banco contava com registros ligados a tráfego sintético (realizado por *bots*), estes foram removidos da análise, bem como as entradas cujos tempos de navegação foram considerados inválidos, isto é, observações com tempos inferiores a 1 milissegundo. Ademais, como não é parte do escopo deste trabalho o tratamento de dados faltantes, estes também foram excluídos.

Ainda com relação ao PLT, após as remoções citadas, temos o histograma da variável resposta, irrespectivo de grupos, apresentado na Figura (3.1) seguir:

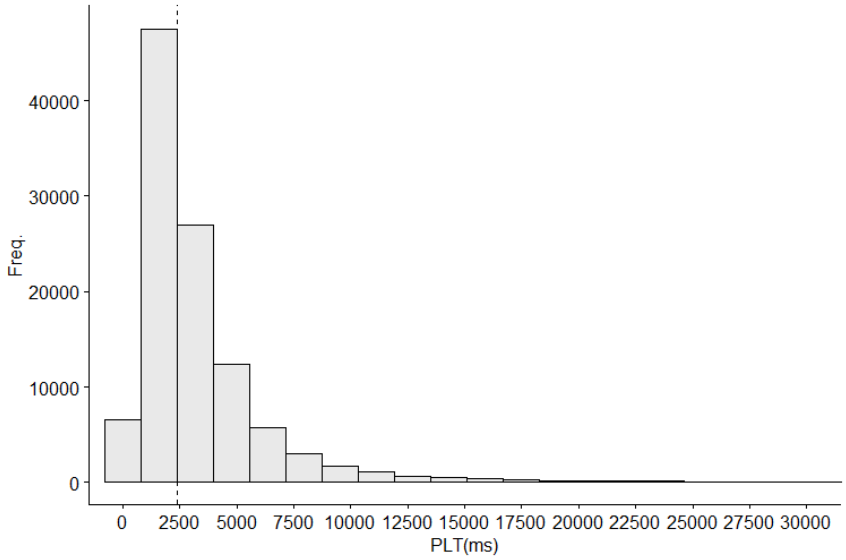


Figura 3.1: Histograma - *Page Load Time* (PLT)

Como podemos ver na Figura (3.1), a distribuição do tempo de carregamento é assimétrica positiva, com mediana em torno de 2.5 segundos, assinalada pela linha pontilhada no gráfico.

Para prosseguirmos com o ajuste de uma análise de regressão, seleção de covariáveis e especificação de modelos, uma inspeção do banco é necessária. Inicialmente, precisamos entender a relação entre as variáveis quantitativas contínuas, expondo-a através de um correlograma:

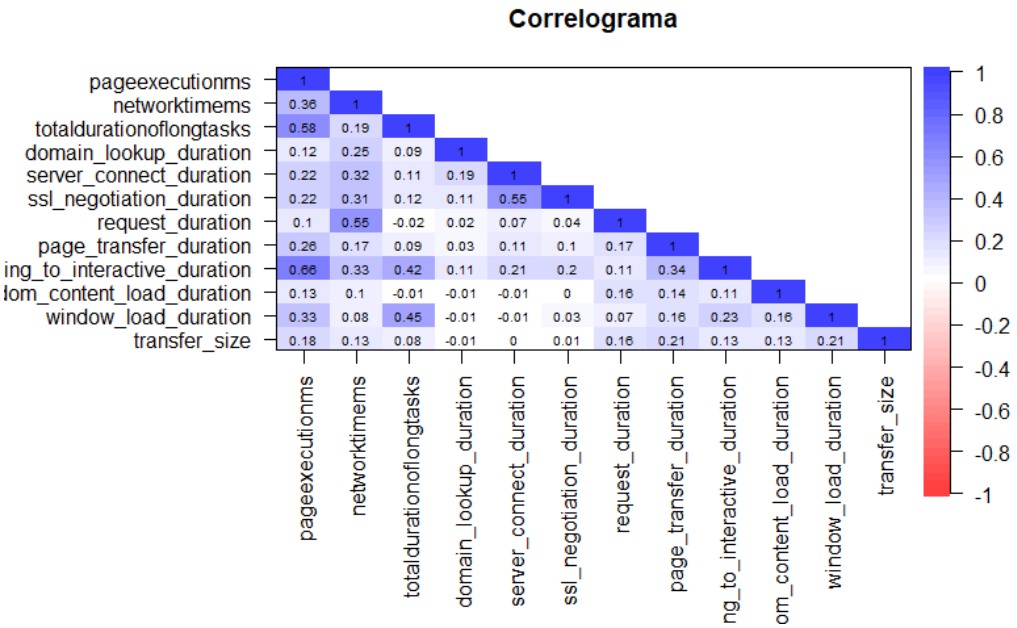


Figura 3.2: Correlograma - variáveis quantitativas

Percebe-se na Figura (3.2) que em sua maioria as variáveis quantitativas apresentam correlação positiva fraca a moderada entre si. Para avaliarmos a possível presença de multicolinearidade nos dados, foram calculados os escores *variance inflation factor* (VIF), incluindo também, as demais variáveis qualitativas:

Tabela 3.5: Tabela VIF

Variável	VIF
applicationid	4.13
isinternal	1.05
uo_language	1.59
is_entry	1.43
is_exit	1.23
pageexecutionms	2.57
networktimems	2.06
totaldurationoflongtasks	2.03
domain_lookup_duration	1.12
server_connect_duration	1.54
ssl_negotiation_duration	1.50
request_duration	1.78
page_transfer_duration	1.22
dom_loading_to_interactive_duration	2.02
dom_content_load_duration	1.45
window_load_duration	1.47
transfer_size	1.20

Segundo (James et al., 2013), variáveis que apresentam multicolinearidade devem ser removidas, já que a informação que estas provêm em relação à variável resposta é redundante. Como regra geral, um VIF acima de 5 indica colinearidade, situação que não encontramos ao analisarmos a Tabela (3.5) acima.

Para subsequente aplicação dos métodos de modelagem e seleção de variáveis, as variáveis categóricas do banco *applicationid*, *isinternal*, *uo_language*, *is_entry* e *is_exit* foram transformadas da seguinte maneira:

Variável	Descrição
applicationid	*One-hot encoding*
isinternal	Binária - isinternal = "no"
uo_language	Binária - uo_language = "en"
is_entry	Binária - is_entry = "no"
is_exit	Binária - is_exit = "no"

Como etapa final de pré-processamento, as variáveis quantitativas restantes foram padronizadas para que pudessem ser exploradas através de algoritmos de *Machine Learning* (ML) sem maiores empecilhos.

3.2 Ajuste de Modelos

Para dar seguimento à análise, será feito o ajuste de modelos através de três métodos distintos, compartilhando a mesma variável resposta. Os métodos propostos são os seguintes:

1. Regressão Linear
2. *Multilayer Perceptron* (MLP)
3. `xgBoost`

O ajuste destes modelos foram feitos utilizando a função `train` do pacote `caret` dentro do ambiente `R` (R Core Team, 2021), criado por (Kuhn, 2008).

3.2.1 Regressão Linear

Como passo inicial deste processo, utilizamos o método de regressão linear múltipla. É importante ressaltar que uma das categorias da variável *applicationid* foi removida para a aplicação deste método, para evitar problemas de multicolinearidade. Assim sendo, os coeficientes estimados mostram a diferença da variável resposta em relação a categoria incluída na média (*applicationidpremierro_shop*).

O modelo é dado por:

$$plt = \beta_0 + \beta_1 applicationid + \beta_2 isinternal + \beta_3 uo\_language + \beta_4 is\_entry + \beta_5 is\_exit$$
$$+ \beta_6 pageexecutionms + \beta_7 networktimems + \beta_8 totaldurationoflongtasks$$
$$+ \beta_9 domain\_lookup\_duration + \beta_{10} server\_connect\_duration + \beta_{11} ssl\_negotiation\_duration$$
$$+ \beta_{12} request\_duration + \beta_{13} page\_transfer\_duration$$
$$+ \beta_{14} dom\_loading\_to\_interactive\_duration + \beta_{15} dom\_content\_load\_duration$$
$$+ \beta_{16} window\_load\_duration + \beta_{17} transfer\_size$$

O ajuste do modelo foi feito fazendo uso de *5-fold cross-validation*. Podemos observar na Figura (3.3) as estimativas dos parâmetros e as variáveis do modelo.

```
Residuals:
   Min     1Q Median     3Q    Max
-51512   -483    -91    239  56438

Coefficients:
                                        Estimate   Std. Error t value             Pr(>|t|)
(Intercept)                          1946.973329    46.321525   42.03 < 0.0000000000000002 ***
applicationidcsb_accessoriesresults -1647.486202    47.354481  -34.79 < 0.0000000000000002 ***
applicationidcsb_browse             -1748.786427    41.753025  -41.88 < 0.0000000000000002 ***
applicationidcsb_cart               -1809.854744    42.709272  -42.38 < 0.0000000000000002 ***
applicationidcsb_category           -1805.777567    46.848937  -38.54 < 0.0000000000000002 ***
applicationidcsb_configurator       -2518.150421    42.685307  -58.99 < 0.0000000000000002 ***
applicationidcsb_deals              -1760.341496    44.861938  -39.24 < 0.0000000000000002 ***
applicationidcsb_dealsexperience    -2163.271652    52.746056  -41.01 < 0.0000000000000002 ***
applicationidcsb_homepage           -2123.605499    41.330617  -51.38 < 0.0000000000000002 ***
applicationidcsb_productdetail      -2030.269490    50.005241  -40.60 < 0.0000000000000002 ***
applicationidcsb_search.ux          -1721.227290    52.795456  -32.60 < 0.0000000000000002 ***
applicationidcsb_snp_productdetail  -2297.944200    45.809420  -50.16 < 0.0000000000000002 ***
applicationidcsb_specialdeals       -1987.126916    42.590635  -46.66 < 0.0000000000000002 ***
applicationidcsb_systemresult       -1901.144795    41.150110  -46.20 < 0.0000000000000002 ***
applicationidcsb_unifiedpd          -2220.873893    54.384575  -40.84 < 0.0000000000000002 ***
applicationiddce_dais                -886.191206    41.311896  -21.45 < 0.0000000000000002 ***
applicationiddelltech_www           -1547.429558    47.780309  -32.39 < 0.0000000000000002 ***
applicationidservices_online_community  -349.888325    42.145560   -8.30 < 0.0000000000000002 ***
applicationidsupport_online         -1175.065442    40.881057  -28.74 < 0.0000000000000002 ***
uo_language                           -36.747219    10.316098   -3.56              0.00037 ***
isinternal                             54.824599    22.913379    2.39              0.01673 *
is_entry                             -106.354737    11.793981   -9.02 < 0.0000000000000002 ***
is_exit                                78.134714    11.290999    6.92      0.0000000000045 ***
pageexecutionms                         0.046263     0.000874   52.96 < 0.0000000000000002 ***
networktimems                           1.001765     0.003465  289.10 < 0.0000000000000002 ***
totaldurationoflongtasks                0.188585     0.002859   65.96 < 0.0000000000000002 ***
domain_lookup_duration                  0.074270     0.019957    3.72              0.00020 ***
server_connect_duration                 0.097600     0.019224    5.08      0.0000003839988 ***
ssl_negotiation_duration                0.202184     0.017578   11.50 < 0.0000000000000002 ***
request_duration                        0.014394     0.006020    2.39              0.01681 *
page_transfer_duration                  0.126876     0.012473   10.17 < 0.0000000000000002 ***
dom_loading_to_interactive_duration     1.023207     0.003775  271.06 < 0.0000000000000002 ***
dom_content_load_duration               1.111502     0.013803   80.53 < 0.0000000000000002 ***
window_load_duration                   -0.577774     0.021814  -26.49 < 0.0000000000000002 ***
transfer_size                          -0.104277     0.047328   -2.20              0.02758 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1450 on 96801 degrees of freedom
Multiple R-squared:  0.863,     Adjusted R-squared:  0.863
F-statistic: 1.79e+04 on 34 and 96801 DF,  p-value: <0.0000000000000002
```

Figura 3.3: Ajuste Modelo - Regressão Linear

Vale observar que as estimativas dos parâmetros da Figura (3.3) se referem ao tempo de carregamento da página em milissegundos, incluindo a categoria *applicationidpremierro_shop* na média. Inspecionando os resultados, identificamos que apenas algumas das variáveis possuem p-valores ligeiramente maiores, mas ainda significativos a nível $\alpha = 0.05$.

Analisando de maneira mais aprofundada, os coeficientes na Figura (3.3), percebemos que a maior influência na estimativa do valor do tempo de carregamento de página está associada a variável *applicationid*, que indica os nomes das aplicações internas associadas as sessões de navegação. As demais variáveis categóricas binárias afetam o tempo em até $\pm$ 100ms: quando a sessão é a primeira de uma série (*is_entry*), apresenta respostas em torno 100ms mais rápidas, o que sugere que as páginas iniciais das diversas aplicações são ligeiramente mais otimizadas. Por outro lado, as sessões associadas ao último passo de uma série (*is_exit*) de *page loads* parecem contribuir negativamente, adicionando cerca de 80ms de lentidão.

Adicionalmente, o fato das páginas terem sido carregadas por usuários internos (*isinternal*) da empresa, o tempo de solicitação de conteúdo em memória *cache* (*request_duration*) ou o fato das páginas estarem configuradas em inglês (*uo_language*) não demonstram grande influência sobre o tempo médio de carregamento da aplicação base.

Quanto às variáveis quantitativas, notamos que a maior influência no tempo de carregamento vem daquelas relacionadas ao carregamento de conteúdo da página durante a seção de *frontend*, mais especificamente ligados à interação com os elementos do DOM (*dom_loading_to_interactive_duration*, *dom_content_load_duration*), execução de *Long Tasks* (*totaldurationoflongtasks*) e o tempo de rede até o recebimento da primeira informação da página (*networktimems*).

Como complemento e validação dos pontos supracitados, utilizamos o gráfico de importância das variáveis, que ordena seus coeficientes segundo o valor de sua estatística t:

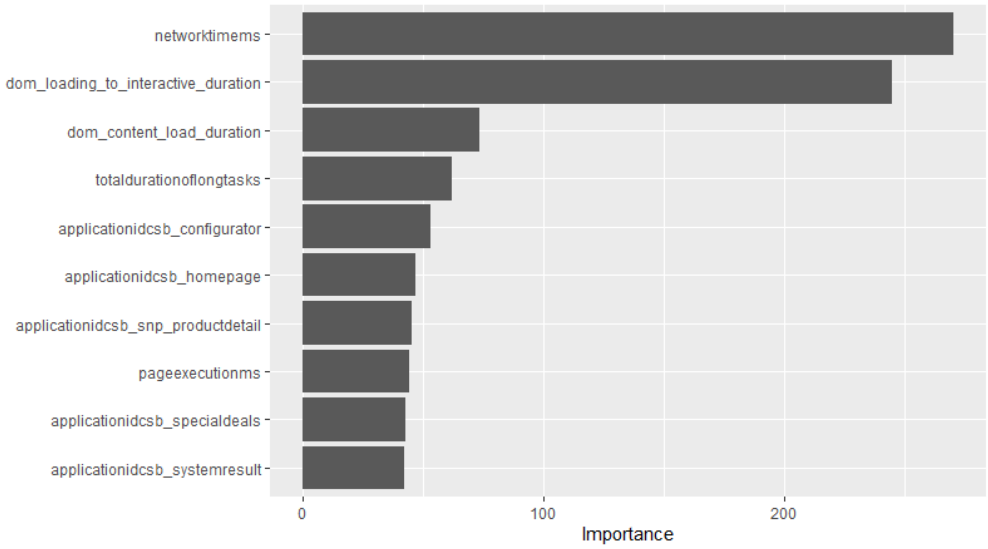


Figura 3.4: Gráfico de importância no modelo de Regressão Linear

3.2.2 *Multilayer Perceptron*

No tocante a aplicação de algoritmos de redes neurais, foi feito o ajuste de uma rede com arquitetura *Multilayer Perceptron* (MLP). O modelo, assim como a regressão linear, foi feito utilizando *5-fold cross-validation*. O treinamento da rede neural é computacionalmente intensivo, e foi feito com diferentes combinações de neurônios:

- 1ª Camada oculta: 20 a 40 *neurons*
- 2ª Camada oculta: 20 a 30 *neurons*
- 3ª Camada oculta: 20 a 30 *neurons*

O modelo utilizado é totalmente conectado e, por se tratar de regressão linear, foi utilizada uma função de ativação linear na camada de saída. As camadas ocultas compartilham a mesma função de ligação logística definida pela Equação (2.18). O treinamento do modelo foi feito utilizando 10^5 épocas de treinamento ou até a convergência do algoritmo. Cabe mencionar ainda que não foi utilizado nenhum critério de parada antecipada.

O modelo com melhor performance, que minimiza o *Root Mean Squared Error* (RMSE), foi o MLP(36,24,24,20,1), ou seja, o modelo com 36 neurônios na camada de entrada, 24 neurônios na primeira e segunda camada oculta, 20 neurônios na terceira camada oculta e 1 neurônio na camada de saída.

3.2.3 `xgBoost`

Como etapa final, fazemos uso do algoritmo `xgBoost`. Mais especificamente, utilizaremos uma versão modificada deste, chamada `xgbDART`, proposto em (Vinayak e Gilad-Bachrach, 2015). Em relação ao algoritmo original, o `xgbDART` se diferencia por aplicar a técnica de *dropout*, comumente utilizada no contexto de aprendizagem profunda, durante o treinamento das árvores de regressão. De maneira simplificada, o *dropout* é uma técnica de regularização. que consiste em treinar diferentes arquiteturas de uma rede, eliminando alguns dos neurônios ocultos na rede aleatoriamente, mantendo os neurônios de entrada e saída originais, minimizando o risco de *overfitting*. O `xgbDART` funciona de maneira similar, porém ignorando/eliminando árvores inteiras ao longo do processo.

O modelo foi treinado utilizando os seguintes parâmetros:

- Número de árvores variando na sequência: (50, 75, 100, 150, 200, 250, 300, 500)
- Profundidade máxima variando na sequência: (2, 4, 6)
- Taxa de aprendizagem (η) variando na sequência: (0.05, 0.1, 0.15)

Após o treinamento, a melhor performance apresentada foi do modelo que usou os parâmetros (500, 6, 0.15), como descritos na lista imediatamente anterior.

Para fins de comparação, podemos observar abaixo na Figura (3.5) o gráfico de importância das variáveis gerada pelo algoritmo:

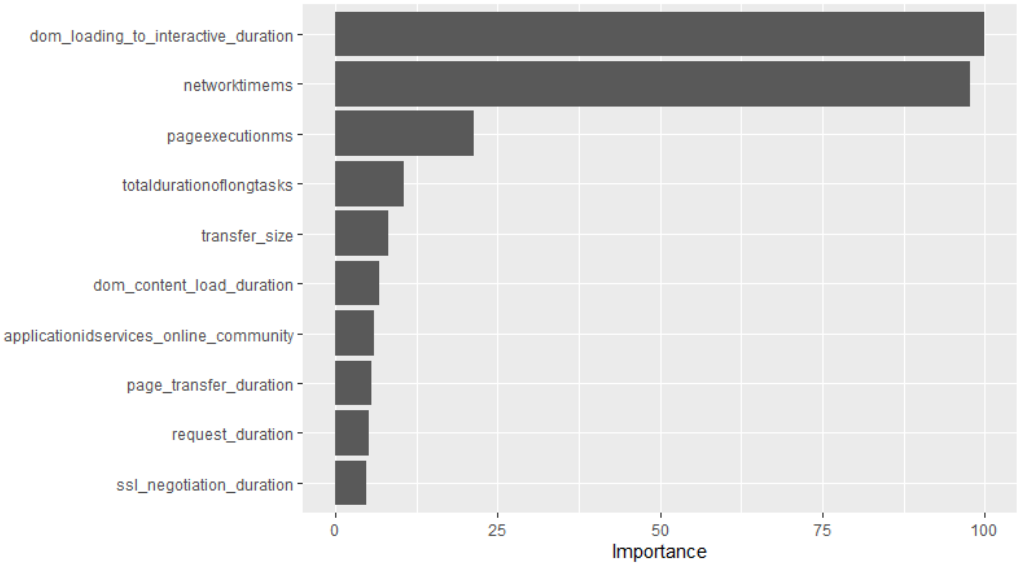


Figura 3.5: Gráfico de importância no algoritmo `xgBoost`

Ao compararmos a Figura (3.5) com o gráfico de importância da regressão linear na Figura (3.4), notamos que o algoritmo `xgBoost` dá mais peso para as variáveis quantitativas, mantendo entre sua lista de 10 melhores variáveis independentes apenas uma das variáveis qualitativas, contra 5 no modelo linear. Por outro lado, podemos notar também que todas as variáveis quantitativas encontradas no gráfico do modelo linear estão em concordância com os do algoritmo `xgBoost`. Além disso, ambos os métodos de importância de variáveis concordam com as duas primeiras variáveis mais importantes (*dom_loading_to_interactive_duration* e *networktimems*), que no contexto da regressão linear, crescem conjuntamente com a variável resposta de 1 para 1.

É importante salientar que embora os gráficos possuam o mesmo intuito, as métricas de importância são diferentes: na regressão linear, a importância das variáveis é dada pelo valor de sua estatística t, que pode ser interpretada como a força da relação linear entre a resposta e a variável independente. Já no `xgBoost`, as variáveis selecionadas são aquelas que trazem maior ganho preditivo, ou seja, o quanto as predições das árvores ajustadas tem sua variância reduzida em média quando aquela variável é inclusa no modelo, somando todas as importâncias e escalonando-as até 100, podendo serem interpretadas como a participação de cada variável dentro do modelo geral.

3.3 Comparação de Modelos

Nesta seção exploraremos os resultados da Seção 3.2. Para fazê-lo, podemos observar os resultados de ajuste dos modelos conforme consolidado na Tabela (3.6):

Tabela 3.6: Tabela Ajuste - Comparação de Modelos

Modelo	RMSE	MAE	R^2
Regressão Linear	0.3731	0.1690	0.8611
MLP	0.3610	0.1731	0.8239
`xgBoost`	0.3392	0.1499	0.8465

Examinando a Tabela (3.6), podemos argumentar que todos os modelos apresentaram um ótimo ajuste ao conjunto de dados ao compararmos os coeficientes R^2, explicando mais de 80% da variabilidade da variável dependente em todos os casos.

No que se refere ao poder preditivo, podemos afirmar que a acurácia dos métodos é similar. Levando em consideração o fato da variável resposta mostrar alta variabilidade, contando com registros de carregamentos de até 240 segundos, a distância dos resíduos expressada pelo *Root mean squared error* (RMSE) estar na casa de 0.3 desvios é um excelente resultado. Os valores computados do *Mean Absolute Error* (MAE) corroboram com esta conclusão.

O melhor modelo, que apresentou o menor RMSE, foi o `xgbDART`, que também possui o menor MAE, podendo ser considerado o mais parcimonioso entre os métodos testados, com uma média de erro menor e variabilidade explicada inferior somente ao método de regressão linear. Vale mencionar que os resultados encontrados poderiam ter sido ainda melhores caso optássemos por usar algum limite para o tempo de carregamento, de maneira a reduzir a quantidade de *outliers* de tempo de carregamento muito elevados. Com o objetivo geral da pesquisa de compreender a influência das variáveis independentes sendo plenamente atingido, não houve a preocupação de refinar ainda mais os resultados preditivos aqui expostos.

4 Conclusão

Este trabalho tinha como objetivo apresentar e examinar dados pertinentes ligados à um dos principais fatores que compõem a experiência do usuário de plataformas *web*: o tempo de carregamento das páginas. Empregando procedimentos de *machine learning*, foi avaliada sob duas diferentes lentes: de que forma as covariáveis do banco influenciam a resposta e qual tipo de modelagem oferece maior poder preditivo.

Para isso, foram apresentados os conceitos em relação ao problema de análise de desempenho de páginas (Seção 1.1.1) e como a variável resposta é capturada e subdividida em componentes: rede, *backend* e *frontend* (Seção 1.1.2). Enquanto a literatura a cerca de *User Experience* (UX) é extensa, não foram encontrados trabalhos com enfoque e métodos de análise imediatamente comparáveis a este.

Dentre os métodos aplicados, foi possível estabelecermos quais aspectos específicos mais pesam em relação a latência de página: o tempo de rede até um usuário receber o primeiro dado da página, assim como os tempos de carga dos elementos visuais, que aumentam em razão próxima de 1:1 com o tempo final de carregamento. Corroborando com (Osmani, 2019), a execução de *long tasks*, que monopolizam o uso dos demais recursos da página até que estejam finalizadas também mostraram impacto razoável na resposta. Além disso, o tempo de carregamento é majoritariamente influenciado pelo tipo de aplicação específica associada a um conjunto de páginas.

Quanto aos métodos de seleção de variáveis utilizados ao longo da Seção 3.2, notamos que enquanto houve concordância entre as variáveis mais impactantes, muito da interpretabilidade dos resultados remanescentes depende do conhecimento do pesquisador do seu conjunto de dados, já que a análise aqui presente gira em torno de dados coletados por uma solução de código específica. Por mais que exista uma padronização de como os tempos de navegação são subdivididos, proposto em (Wang, 2012), outros *softwares* podem coletar dados de maneira diferente, bem como versões atualizadas do mesmo *software* podem incluir ou excluir parâmetros a serem capturados durante a navegação das páginas.

Como resultado complementar, todos os métodos empregados culminaram em modelos capazes de predizer a resposta de forma acurada e robusta, agregando ao processo de descoberta do conhecimento e auxiliando a entender a relação entre as variáveis em questão. Ademais, os dados e resultados podem auxiliar no desenvolvimento de novas páginas e tomada de ações por parte dos times de desenvolvimento baseados nos fatores que contribuem negativamente ao tempo de carregamento aqui expostos.

Potenciais trabalhos futuros podem ser realizados tomando maior vantagem dos métodos de previsão aqui empregados, no contexto de previsão de taxas de conversão de vendas. Dentro deste mesmo contexto, poderiam ser empregadas abordagens do ramo de análise de sobrevivência, dada a natureza assimétrica dos dados em questão.

Referências Bibliográficas

Bell, J. (2020). *Machine Learning: Hands-On for Developers and Technical Professionals*. John Wiley & Sons, 2nd edition.

Bojan, P., Anstey, C., e Wagner, J. (2020). Why does speed matter? Google Developers. https://web.dev/why-speed-matters/. Acesso em: 30 set. 2021.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Cawley, G. C. e Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107.

Chen, T. e Guestrin, C. (2016). XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Fetouni, A. V. (2018). Evaluating the correlation between site speed and its effect on conversion rates for norwegian air shuttle asa. Master's thesis, Norwegian University of Life Sciences, Ås.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Hartson, R. e Pyla, P. (2012). *The UX Book: Process and guidelines for ensuring a quality user experience*. Elsevier.

Haykin, S. (2008). *Neural Networks and Learning Machines*. Pearson, 3rd edition.

James, G., Witten, D., Hastie, T., e Tibshirani, R. (2013). *An Introduction To Statistical Learning*. Springer.

Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of statistical software*, 28:1–26.

Manhas, J. (2013). A study of factors affecting websites page loading speed for efficient web performance. *International Journal of Computer Sciences and Engineering*, 1(3):32–35.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

NetMarketshare Team (2018). Operating system market share. Net Market Share. https://netmarketshare.com/operating-system-market-share.aspx. Acesso em: 2 Mar. 2022.

Osmani, A. (2019). Are long javascript tasks delaying your time to interactive? Google Developers. https://web.dev/long-tasks-devtools/. Acesso em: 7 Abr. 2022.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Singhal, A. e Cutts, M. (2010). Using site speed in web search ranking. Google Developers. https://developers.google.com/search/blog/2010/04/using-site-speed-in-web-search-ranking. Acesso em: 9 Nov. 2021.

Vinayak, R. K. e Gilad-Bachrach, R. (2015). Dart: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics*, pages 489–497. PMLR.

Wang, Z. (2012). Navigation timing. W3C. https://www.w3.org/TR/navigation-timing/. Acesso em: 9 Nov. 2021.

Apêndices

Apêndice A - Rotina R:

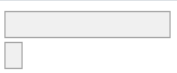
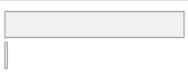
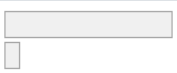
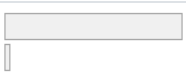
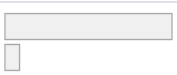
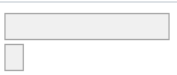
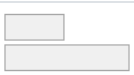
```r
# Packages ------------------------------------------------------------------
pkgs <- c("dplyr", "readr", "caret", "pls", "neuralnet", "RSNNS", "rpart",
          "vip", "doParallel", "doSNOW", "xgboost", "plyr", "stringr")
lapply(pkgs, require, character.only = T) #Importing list of packages using apply
rm(pkgs)
# Data Import & Specification -----------------------------------------------
path <- "~/bmrng_data/"; setwd(path) # Definindo diretório de trabalho
# Definindo e lendo arquivo csv - banco de dados
path_scaled <- "~/bmrng_data/scaled"
base_full <- read_csv(str_c(path_scaled, "/bmrng_scaled.csv"))
rm(path, path_scaled, path_unscaled)
# Linear Models -------------------------------------------------------------
# Utilizando trainControl para definir parâmetros da validação cruzada
# ??caret::trainControl
ctrl_lm <- trainControl(method = "cv", number = 5)
modCV <- caret::train(plt ~ ., data = base_full, method = "lm",
    trControl = ctrl_lm) # Treinamento modelo linear simples usando 5-fold cross-validation
# Neural Network ------------------------------------------------------------
# Habilitando paralelismo para ganho de performance - 11 núcleos de processamento
# ??parallel::makeCluster
cl <- makeCluster(11, outfile = "")
registerDoSNOW(cl)
ctrl_nn <- trainControl(method = "cv", number = 5, allowParallel = TRUE,
                        savePredictions = TRUE, verboseIter = TRUE)
# Criando grid de parâmetros - combinação dos argumentos da NN passados à expand.grid()
tune_nn <- expand.grid(layer1 = c(20:40),
                       layer2 = c(20:30),
                       layer3 = c(20:30))
# Treinamento rede MLP com uma fração do dataset
# Redes Neurais são computacionalmente intensivas, o usuário pode identificar o tempo despendido
# através do parametro times : mod$times
# O argumento metric indica qual parâmetro de ajuste deve ser minimizado, neste caso o RMSE
mod_nn <- caret::train(plt ~., data = base_full[1:5000,],
                       method = "neuralnet",
                       threshold = 3,
                       linear.output = TRUE, metric = "RMSE",
                       trControl = ctrl_nn, tuneGrid = tune_nn)
# xgBoost -------------------------------------------------------------------
ctrl_boost <- trainControl(method = "cv", number = 5, allowParallel = TRUE,
                           savePredictions = TRUE, verboseIter = TRUE)
# Criando grid de parâmetros do modelo xgbDART: profundidade, taxa de aprendizagem
# e número de árvores variáveis, demais constantes
tune_boost <- expand.grid(nrounds = c(50, 75, 100, 150, 200, 250, 300, 500),
                          max_depth = c(2, 4, 6),
                          eta = c(0.05, 0.1, 0.15),
                          rate_drop = 0.10, skip_drop = 0.10,
                          colsample_bytree = 0.90,
                          min_child_weight = 2, subsample = 0.75,
                          gamma = 0.10)
# Treinamento da regressão linear com boosting, minimizando RMSE
mod_boost <- caret::train(plt ~., data = base_full_nn[1:5000,],
                       method = "xgbDART", metric = "RMSE",
                       trControl = ctrl_boost, tuneGrid = tune_boost)
# Model Evaluation ----------------------------------------------------------
modCV[[4]]
vip(modCV, num_features = 10) # Plot de importância - reg linear
mod_nn[[6]] # Equivale a mod_nn$bestTune, acessa informações do melhor modelo ajustado
mod_nn[[4]][529, ] # Equivale a mod_nn$results, acessa resultados do n-ésimo modelo indicado
mod_boost[[6]]
mod_boost[[4]][72, ]
vip(mod_boost, num_features = 10) # Plot de importância - xgBoost
```

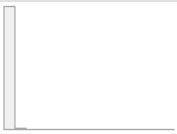
Apêndice B - Dicionário de Dados:

Dicionário Banco de Dados

base_full

Dimensões: 96836 x 35
Duplicações: 0

Variável	Descrição	Estatísticas Descritivas	Freq. (%)	Gráficos
plt [Numérica]	Tempo de carregamento final da página (page load time)	Média (desvio) : 3350.8 (3901.7) min < med < max: 98 < 2354 < 236856 IQR (CV) : 2487 (1.2)	12219 valores distintos	
applicationidcsb_accessoriesresults [Binária]	Indicador aplicação - csb_accessoriesresults	Min : 0 Mean : 0 Max : 1	0 : 94188 (97.3%) 1 : 2648 (2.7%)	
applicationidcsb_browse [Binária]	Indicador aplicação - csb_browse	Min : 0 Mean : 0.1 Max : 1	0 : 88291 (91.2%) 1 : 8545 (8.8%)	
applicationidcsb_cart [Binária]	Indicador aplicação - csb_cart	Min : 0 Mean : 0.1 Max : 1	0 : 89208 (92.1%) 1 : 7628 (7.9%)	
applicationidcsb_category [Binária]	Indicador aplicação - csb_category	Min : 0 Mean : 0 Max : 1	0 : 94007 (97.1%) 1 : 2829 (2.9%)	
applicationidcsb_configurator [Binária]	Indicador aplicação - csb_configurator	Min : 0 Mean : 0.1 Max : 1	0 : 90802 (93.8%) 1 : 6034 (6.2%)	
applicationidcsb_deals [Binária]	Indicador aplicação - csb_deals	Min : 0 Mean : 0 Max : 1	0 : 93024 (96.1%) 1 : 3812 (3.9%)	
applicationidcsb_dealsexperience [Binária]	Indicador aplicação - csb_dealsexperience	Min : 0 Mean : 0 Max : 1	0 : 95229 (98.3%) 1 : 1607 (1.7%)	
applicationidcsb_homepage [Binária]	Indicador aplicação - csb_homepage	Min : 0 Mean : 0.1 Max : 1	0 : 87968 (90.8%) 1 : 8868 (9.2%)	
applicationidcsb_productdetail [Binária]	Indicador aplicação - csb_productdetail	Min : 0 Mean : 0 Max : 1	0 : 94791 (97.9%) 1 : 2045 (2.1%)	
applicationidcsb_search.ux [Binária]	Indicador aplicação - csb_search.ux	Min : 0 Mean : 0 Max : 1	0 : 95238 (98.3%) 1 : 1598 (1.7%)	
applicationidcsb_snp_productdetail [Binária]	Indicador aplicação - csb_snp_productdetail	Min : 0 Mean : 0 Max : 1	0 : 93468 (96.5%) 1 : 3368 (3.5%)	
applicationidcsb_specialdeals [Binária]	Indicador aplicação - csb_specialdeals	Min : 0 Mean : 0.1 Max : 1	0 : 89802 (92.7%) 1 : 7034 (7.3%)	
applicationidcsb_systemresult [Binária]	Indicador aplicação - csb_systemresult	Min : 0 Mean : 0.1 Max : 1	0 : 87859 (90.7%) 1 : 8977 (9.3%)	
applicationidcsb_unifiedpd [Binária]	Indicador aplicação - csb_unifiedpd	Min : 0 Mean : 0 Max : 1	0 : 95446 (98.6%) 1 : 1390 (1.4%)	
applicationiddce_dais [Binária]	Indicador aplicação - dce_dais	Min : 0 Mean : 0.1 Max : 1	0 : 88637 (91.5%) 1 : 8199 (8.5%)	
applicationiddelltech_www [Binária]	Indicador aplicação - delltech_www	Min : 0 Mean : 0 Max : 1	0 : 94170 (97.2%) 1 : 2666 (2.8%)	
applicationidservices_online_community [Binária]	Indicador aplicação - services_online_community	Min : 0 Mean : 0.1 Max : 1	0 : 88854 (91.8%) 1 : 7982 (8.2%)	
applicationidsupport_online [Binária]	Indicador aplicação - support_online	Min : 0 Mean : 0.1 Max : 1	0 : 86693 (89.5%) 1 : 10143 (10.5%)	
uo_language [Binária]	Sinalizador de linguagem da página	Min : 0 Mean : 0.7 Max : 1	0 : 31221 (32.2%) 1 : 65615 (67.8%)	

Variável	Descrição	Estatísticas Descritivas	Freq. (%)	Gráficos
isinternal [Binária]	Sinalizador de tráfego interno à empresa	Min : 0 Mean : 1 Max : 1	0 : 4383 (4.5%) 1 : 92453 (95.5%)	
is_entry [Binária]	Sinalizador de primeiro acesso em uma sequência	Min : 0 Mean : 0.7 Max : 1	0 : 32031 (33.1%) 1 : 64805 (66.9%)	
is_exit [Binária]	Sinalizador de último acesso em uma sequência	Min : 0 Mean : 0.7 Max : 1	0 : 28849 (29.8%) 1 : 67987 (70.2%)	
pageexecutionms [Numérica]	Tempo de carregamento frontend genérico	Média (desvio) : 6122.3 (8428.3) min < med < max: 63 < 3606 < 978722 IQR (CV) : 5096.2 (1.4)	20251 valores distintos	
networktimems [Numérica]	Tempo entre solicitação do usuário até recebimento do primeiro byte	Média (desvio) : 1240.5 (1925.5) min < med < max: 1 < 807 < 155459 IQR (CV) : 1048 (1.6)	6654 valores distintos	
totaldurationoflongtasks [Numérica]	Tempo de duração de tarefas longas	Média (desvio) : 1461 (2303.3) min < med < max: 49 < 693 < 64778 IQR (CV) : 1406 (1.6)	8644 valores distintos	
domain_lookup_duration [Numérica]	Tempo de execução da consulta de domínio	Média (desvio) : 39 (246.4) min < med < max: 0 < 0 < 16251 IQR (CV) : 0 (6.3)	1349 valores distintos	
server_connect_duration [Numérica]	Tempo de duração do pedido de conexão à rede	Média (desvio) : 63.2 (300.2) min < med < max: 0 < 0 < 25304 IQR (CV) : 45 (4.8)	1525 valores distintos	
ssl_negotiation_duration [Numérica]	Tempo de duração da autenticação segura (SSL), se houver	Média (desvio) : 63.4 (324.5) min < med < max: 0 < 13 < 46630 IQR (CV) : 42 (5.1)	1449 valores distintos	
request_duration [Numérica]	Tempo de duração da solicitação do documento ao servidor/cache	Média (desvio) : 676 (1026.3) min < med < max: 1 < 441 < 50325 IQR (CV) : 659 (1.5)	4243 valores distintos	
page_transfer_duration [Numérica]	Tempo de duração da transferência entre servidor e memória cache	Média (desvio) : 119.7 (413) min < med < max: 1 < 28 < 28343 IQR (CV) : 87 (3.5)	1993 valores distintos	
dom_loading_to_interactive_duration [Numérica]	Tempo de duração da renderização do documento principal até página se tornar interagível	Média (desvio) : 1117.5 (1749.5) min < med < max: 30 < 673 < 157468 IQR (CV) : 889 (1.6)	6346 valores distintos	
dom_content_load_duration [Numérica]	Tempo de duração da renderização de todos os elementos visuais do documento	Média (desvio) : 104.5 (405.9) min < med < max: 1 < 3 < 41874 IQR (CV) : 27 (3.9)	2170 valores distintos	
window_load_duration [Numérica]	Tempo de duração do carregamento de recursos dependentes da página (fontes, folhas de estilo)	Média (desvio) : 144.6 (258.2) min < med < max: 0 < 65 < 9188 IQR (CV) : 149 (1.8)	1849 valores distintos	
transfer_size [Numérica]	Tamanho em KBytes da página web acessada	Média (desvio) : 66.4 (107.5) min < med < max: 1 < 45.8 < 7187.5 IQR (CV) : 36.8 (1.6)	16717 valores distintos