Aplicação Web para geração automatizada de modelos de Aprendizado de Máquina

João Paulo Nóbrega Alvim

Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica

Uberlândia
2019

João Paulo Nóbrega Alvim

Aplicação Web para geração automatizada de modelos de Aprendizado de Máquina

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do diploma de graduação em Engenharia de Computação.

Área de concentração: Engenharia de Computação

Orientador: Keiji Yamanaka

Uberlândia
2019

Este trabalho é dedicado à minha família e aos meus amigos, os quais sempre estiveram ao meu lado em toda minha trajetória.

Agradecimentos

Gostaria de agradecer aos meus pais, Gustavo Alessandri Alvim e Polyana Fernandes Nóbrega Alvim, os quais sempre me deram todas as condições necessárias para que eu chegasse até aqui.

Também, deixo meu agradecimento aos meus avós maternos Floro Nóbrega e Maria da Glória Fernandes Nóbrega, e à minha companheira Dayane Yumi Yamamoto, por me darem todo o apoio e zelo necessário em momentos de muita dificuldade, contribuindo muito para o meu crescimento pessoal e profissional.

Agradeço também ao Prof. Dr. Keiji Yamanaka pela tutoria deste projeto. A atenção, ética e profissionalismo do mesmo foram indispensáveis no desenvolvimento dessa aplicação. Além disso, agradeço à Universidade Federal de Uberlândia, à Faculdade de Engenharia Elétrica, e em especial à coordenação do curso de Engenharia de Computação, por fornecerem um ensino público gratuito e de qualidade.

"Those who can imagine anything, can create the impossible."
(Alan Turing)

Resumo

Aprendizado de Máquina e *Inteligência Artificial (IA)* são termos que estão cada vez mais populares na academia, na indústria e no vocabulário popular, devido aos recorrentes avanços tecnológicos e às aplicações práticas em produtos industriais, como sistemas de recomendação, reconhecimento de padrões e imagens e análises de crédito bancário.

Os algoritmos de Aprendizado de Máquina são programados com a finalidade de criarem modelos estatísticos e matemáticos através de dados, e isso exige certo conhecimento prévio sobre matemática, estatística, álgebra e programação de computadores. Porém, não são todas as pessoas que possuem tais pré-requisitos, o que faz com que se crie uma barreira de entrada a elas no universo da *IA*.

Dessa forma, o trabalho aqui desenvolvido visa construir uma Aplicação Web de fácil acesso e manipulação, utilizando a linguagem de programação R, para promover a criação automatizada de modelos de Aprendizado de Máquina, a fim de facilitar o uso e entendimento dos algoritmos para pessoas não versadas na área.

Palavras-chave: Aprendizado de Máquina. Inteligência Artificial. Aplicação Web.

Abstract

Machine Learning and ***AI!*** *(**AI!**)* are terms that are increasingly popular in academia, industry and popular vocabulary, due to recurring technological advances and practical applications in industrial products such as recommendation systems, pattern and image recognition and bank credit analyzes.

Machine Learning algorithms are programmed to create statistical and mathematical models through data, and this requires some prior knowledge of mathematics, statistics, algebra, and computer programming. However, not all people have such prerequisites, which creates a barrier into the ***AI!*** universe.

Thus, the work developed here aims to construct a Web Application that is easy to access and manipulate, using the R programming language, to promote the automated creation of Machine Learning models, in order to facilitate the use and understanding of algorithms for non- versed in the area.

Keywords: Machine Learning. Artificial Intelligence. Web Application.

Lista de ilustrações

Lista de siglas

AM Aprendizado de Máquina

AD Árvore de Decisão

IA Inteligência Artificial

FA Floresta Aleatória

PMC Perceptron Multicamadas

NB Naive Bayes

KVP K-Vizinhos Mais Próximos

MVS Máquina Vetor de Suporte

VC Validação Cruzada

Sumário

Capítulo 1

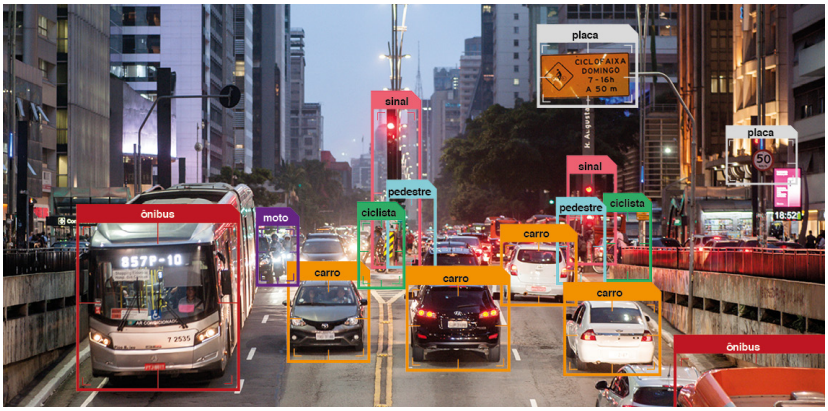
Introdução

1.1 Motivação

O Aprendizado de Máquina está tendo um efeito substancial em muitas áreas de tecnologia e ciência. Exemplos recentes de casos de uso que obtiveram sucesso através da aplicação do AM incluem robótica, controle de veículos autônomos, processamento de linguagem natural, reconhecimento de voz, pesquisas em neurociência, e aplicações em visão computacional (QIN; CHIANG, 2019).

O campo do Aprendizado de Máquina é suficientemente jovem, e isso faz com que se encontre em um processo rápido de expansão, muitas vezes criando novas formalizações de problemas a serem resolvidos com esta tecnologia, impulsionados por aplicações práticas, como segmentação de consumidores e sistemas de recomendação, aplicações de recomendação de plantio na indústria agroalimentar, entre outros. (DIEZ-OLIVAN et al., 2019).

Figura 1 – Exemplo de aplicação de visão computacional: detecção e reconhecimento de objetos.



Fonte: disponível em *https://revistapesquisa.fapesp.br/2019/03/14/as-maquinas-que-tudo-veem/*

De acordo com o *site* de empregos Indeed, a demanda por profissionais capacitados nas áreas de Inteligêcia Artificial e Aprendizado de Máquina, cresceu, entre junho de 2015 e junho de 2018, em 344% (INDEED, 2019). Considerações como essas sugerem que o AM provavelmente figurará entre uma das tecnologias mais transformadoras do século XXI. Assim, não é descabido dizer que é extremamente necessário que a sociedade comece agora a considerar como maximizar seus benefícios.

Apesar de tudo, muitas pessoas não possuem conhecimentos técnicos relacionados à matemática, estatística, AM e programação de computadores, principalmente quando se trata de áreas do conhecimentos diferentes da Ciência e Engenharia da Computação. Dessa forma, nestas áreas do conhecimento, perde-se a oportunidade de extrair o maior potencial de tecnologias que podem revolucionar a forma como pesquisas, procedimentos e trabalhos são conduzidos.

Portanto, o trabalho proposto visa construir uma aplicação gratuita que objetiva facilitar o acesso e a implementação de técnicas de AM por pessoas que possuem pouco ou nenhum conhecimento prévio a respeito do tema. Além disso, tem como objetivo promover a atuação da Inteligência Artificial e do Aprendizado de Máquina em campos ainda pouco explorados por essas tecnologias, contribuindo para a otenção de melhores resultados em pesquisas acadêmicas, processos industriais, criação de novos produtos, entre outras aplicações.

1.2 Objetivos

1.2.1 Objetivos Gerais

Apresentar a construção de uma aplicação *Web* intuitiva, de fácil manipulação, gratuita e de código aberto, que tem como objetivo prover a criação automática de modelos supervisionados de Aprendizado de Máquina. A aplicação visa atender as necessidades de usuários que possuem pouca ou nenhuma experiência prévia em programação de computadores e Aprendizado de Máquina.

1.2.2 Objetivos Específicos

1. Definir a linguagem de programação de computadores a ser utilizada na construção da aplicação.

2. Definir os principais pacotes ou bibliotecas a serem utilizadas na construção da aplicação, baseado na escolha da linguagem de programação.

3. Definir quais as principais funcionalidades, o conteúdo geral da aplicação, e de que forma ambos estarão dispostos na interface gráfica proposta.

4. Definir os algoritmos de Aprendizado de Máquina a serem implementados na ferramenta.

5. Realizar testes de usabilidade da aplicação com usuários voluntários de diversos perfis profissionais e técnicos e diferentes áreas de interesse.

6. Analisar os resultados obtidos através dos testes de usabilidade e compará-los com a proposta inicial da aplicação.

1.3 Organização do Trabalho

Este trabalho está estruturado de acordo com a descrição a seguir:

Capítulo 1 - Introdução

Este capítulo aborda a motivação do trabalho, os objetivos gerais e específicos, e como o trabalho está estruturado.

Capítulo 2 - Fundamentação Teórica

Este capítulo aborda de forma introdutória os conceitos envolvidos nas etapas necessárias para a criação de um modelo de Aprendizado de Máquina, através de fundamentações teóricas históricas, exemplos e análises gráficas. Além disso, expõe também de forma introdutória, o funcionamento dos principais algoritmos de aprendizado supervisionado.

Capítulo 3 - Metodologia

Este capítulo aborda as ferramentas utilizadas na construção da aplicação, as etapas de sua construção, e os métodos de avaliação de usabilidade utilizados.

Capítulo 4 - Machine Learning Toolkit

Este Capítulo aborda as principais características e funcionalidades da aplicação Machine Learning Toolkit. Além disso, ilustra através de imagens exemplos de como utilizar a aplicação.

Capítulo 5 - Resultados e Discussões

Este capítulo apresenta os resultados obtidos através das pesquisas de usabilidade realizadas pelos usuários voluntários, e as considerações do autor a respeito destes resultados.

Capítulo 6 - Conclusão

Este capítulo apresenta as considerações finais a respeito do trabalho proposto.

Referências Bibliográficas

Apresenta as referências bibliográficas utilizadas na construção deste trabalho.

Anexo A

Apresenta os resultados que são disponibilizados pelo Machine Learning Toolkit.

Anexo B

Apresenta os modelos do pacote *Caret* utilizados na aplicação.

Anexo C

Apresenta os questionários de avaliação do sistema utilizados neste trabalho.

Capítulo 2

Fundamentação Teórica

2.1 Aprendizado de Máquina

Existem diversas definições de Aprendizado de Máquina. Muitas delas surgiram há muito tempo, e outras, são criadas e adaptadas continuamente, devido à grande popularidade que o termo obteve e continua obtendo com o passar dos anos.

Uma definição geral explica AM como o campo de estudos que dá aos computadores a habilidade de aprenderem sem que sejam explicitamente programados para isso (SAMUEL, 1959). Em outras palavras, é a ciência de programar computadores de forma que eles possam aprender a partir de dados e criar modelos estatísticos ou matemáticos para resolver problemas. (RUSSELL, 1996).

Sistemas construídos a partir de algoritmos de AM têm a capacidade de aprender com dados históricos ou em tempo real. Assim, o Aprendizado de Máquina pode atuar em inúmeras aplicações, como reconhecimento de padrões, análises de sentimentos, sistemas de recomendação, entre outras.

2.2 Tipos de sistemas baseados em Aprendizado de Máquina

Existem diversos tipos de sistemas baseados em Aprendizado de Máquina que podem ser muito diferentes entre si (RUSSELL; NORVIG, 2009a). Dessa forma, convém classificá-los em categorias baseadas em fatores bem definidos, os principais abordados neste trabalho são:

1. Se são treinados com ou sem supervisão humana (aprendizado supervisionado, não supervisionado, semi-supervisionados ou por reforço);[1]

[1] Algoritmos de aprendizado não supervisionado, semi-supervisionado e de aprendizado por reforço não serão abordados com detalhes neste trabalho.

2. Se funcionam através da comparação de novos dados com dados conhecidos, ou detectam padrões nos dados e constroem modelos (baseados em instância ou baseados em modelo).

3. Se podem ou não aprender incrementalmente, onde existe um fluxo de dados constante recebido (aprendizado em *streaming/online* ou em *bacth*)

Esses e outros critérios não são exclusivos. Isso significa que podem ser combinados dependendo do tipo problemas que se queira resolver. Por exemplo, pode-se construir um sistema supervisionado baseado em instância.

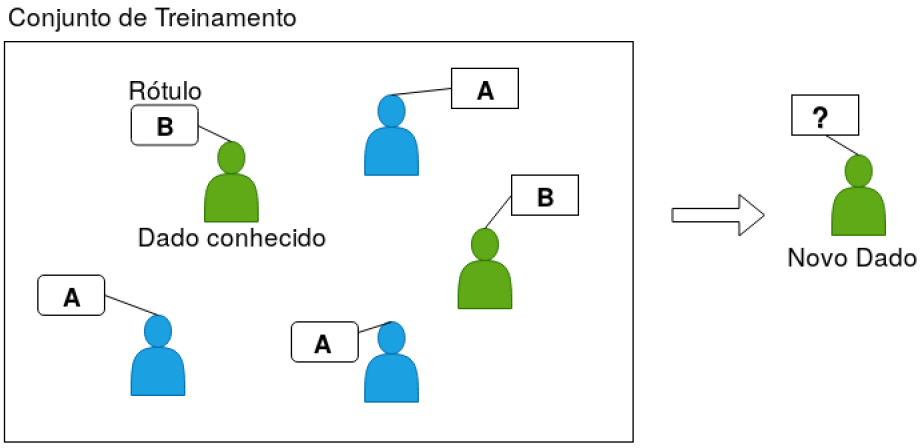
Além disso, o escopo deste trabalho contemplará apenas o fator relacionado ao papel da supervisão humana no treinamento dos modelos, ou seja, as diferenciações entre Aprendizado Supervisionado e Aprendizado Não Supervisionado.

2.2.1 Aprendizado Supervisionado *versus* Não Supervisionado

1. **Aprendizado Supervisionado**: Neste tipo de aprendizado, tanto as variáveis de entrada como as variáveis de saída são conhecidas (MITCHELL, 1997a). Em outras palavras, os dados que são utilizados para alimentar o modelo incluem também as soluções desejadas.

   Observe a figura abaixo:

Figura 2 – Exemplo de conjunto de dados em aplicações supervisionadas (classificação de nacionalidade, brasileira e argentina).



Fonte: Autor

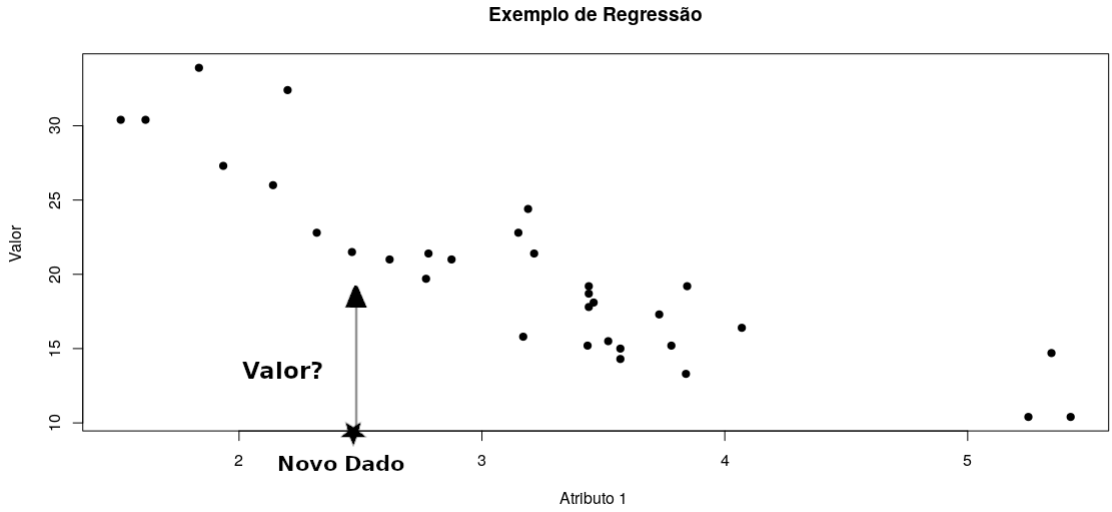
Neste exemplo de aprendizado supervisionado, o algoritmo tem como objetivo classificar a nacionalidade de uma pessoa (brasileira ou argentina) baseada em dados já rotulados previamente.

No contexto do aprendizado supervisionado, existem dois tipos de problemas principais: **regressão** e **classificação**.

Problemas de classificação possuem saídas discretas. Como pode ser observado na *Figura 1*, o algoritmo classifica a nacionalidade de um novo usuário baseado em dados históricos de usuários já conhecidos e classificados.

Problemas de regressão possuem saída contínua. Por exemplo, o preço de uma casa deve ser predito baseado em atributos como área do imóvel, o número de quartos, e assim por diante. O preço a ser predito é uma variável contínua.

Figura 3 – Exemplo de regressão.



Fonte: Autor

Alguns dos algoritmos de aprendizado supervisionado mais comuns são:

- ❏ Regressão Linear;
- ❏ Árvore de Decisão e Floresta Aleatória;
- ❏ Máquina Vetor de Suporte;
- ❏ K-Vizinhos Mais Próximos;
- ❏ Redes Neurais Artificiais. [2]

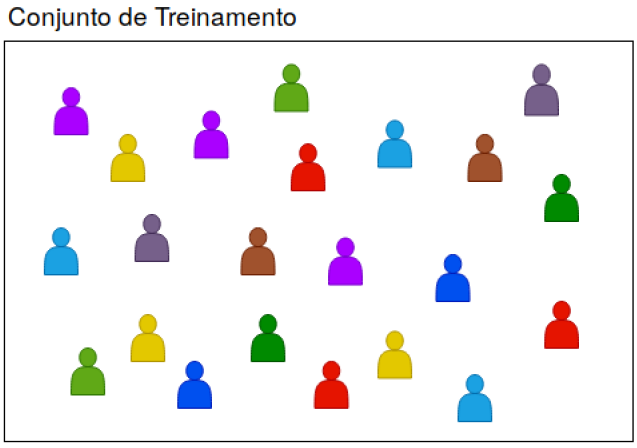
2. **Aprendizado Não Supervisionado**: Neste tipo de aprendizado, os dados que são utilizados para alimentar o modelo não incluem as soluções desejadas. Dessa forma, o algoritmo tem como objetivo encontrar padrões baseados em todas as variáveis

[2] Algoritmos de Redes Neurais Artificiais podem possuir diferentes arquiteturas, e algumas delas podem ser não supervisionadas.

disponibilizadas (MITCHELL, 1997a). Por exemplo, realizar a segmentação de consumidores baseado em idade, gênero, renda, etc.

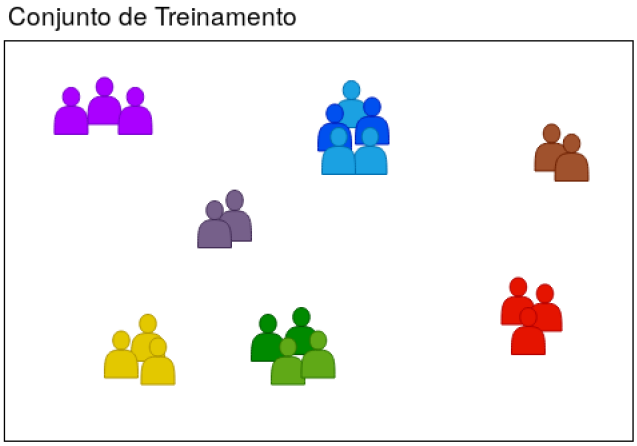
Observe as figuras 3 e 4 a seguir:

Figura 4 – Exemplo de conjunto de dados em aplicações não supervisionadas



Fonte: Autor

Figura 5 – Exemplo de aplicação de um algoritmo de agrupamento.



Fonte: Autor

No exemplo acima, o algoritmo segmentou os dados baseado em uma característica em comum, a cor.

Alguns dos algoritmos de aprendizado não supervisionado mais comuns são:

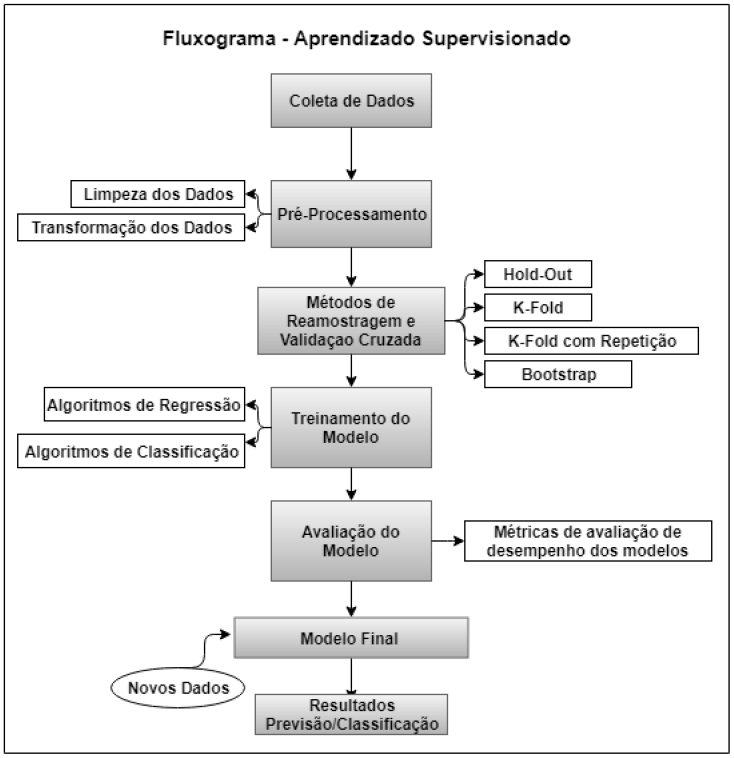
- Algoritmos de Agrupamento (K-Means, Agrupamento Hierárquico, etc);
- Algoritmos de Visualização e Redução de Dimensionalidade (Análise de Componentes Principais);

- ❏ Algoritmos de Regras de Associação (Apriori, Eclat, etc).

2.3 Processo de desenvolvimento de um modelo de AM

O processo de desenvolvimento de um modelo de AM é composto por várias etapas distintas, conforme pode ser observado no fluxograma abaixo:

Figura 6 – Fluxograma proposto para o desenvolvimento de modelos de aprendizado supervisionado.



Fonte: Autor

Desta forma, nesta seção e nas suas seguintes subseções, serão discutidos com maiores detalhes os passos necessários para o desenvolvimento dos modelos de AM do início ao fim.

2.3.1 Coleta dos dados

A coleta dos dados é a primeira etapa necessária para o desenvolvimento de um modelo de AM. Apesar de parecer simples, este é um passo crucial no processo e, futuramente, irá impactar em quão bom o modelo criado será.

Algoritmos de AM são altamente dependentes de dados, frequentemente necessitam de uma enorme quantidade de observações para atingirem um nível de performance aceitável (PEREIRA; NORVIG; HALEVY, 2009). Ainda, dados enviezados podem afetar consideravelmente a performance e a capacidade de generalização do modelo (OBERMEYER; EMANUEL, 2016). Dessa forma, quanto maior a quantidade e qualidade dos dados coletada, maior a chance do modelo criado se comportar da maneira esperada.

O volume de dados existente é cada vez maior. Estes dados podem ser estruturados ou não estruturados, numéricos (temperatura, preço, distância, etc), categóricos (gênero, cor, espécie, etc) ou até mesmo textos livres (poemas, receitas médicas) (DATAROBOT, 2019).

Os dados podem surgir em diversos formatos, e derivados de fontes diferentes. Por isso, existem diversas formas de se coletar dados, como através da utilização de sensores, *websites* que disponibilizam conjuntos de dados prontos para uso, técnicas de coleta como *web scrapping*, inserção de dados em sistemas por usuários, entre outras.

2.3.2 Pré-processamento

Algoritmos de AM aprendem com dados, por isso é fundamental que se forneça os dados de forma correta para que se possam obter os melhores resultados do modelo aplicado. Mesmo se existir uma grande quantidade de dados, é necessário certificar-se de que eles estão, por exemplo, em certa escala ou formato adequado (FAMILI et al., 1997).

Alguns modelos de AM precisam de dados com características específicas, por exemplo, o algoritmo de Floresta Aleatória (FA) não suporta valores ausentes, portanto, para executá-lo, é necessário tratar no conjuntos de dados originais esse problema.

Existem diversas técnicas de pré-processamento de dados que podem ser aplicadas. Nesta seção, abordaremos algumas das principais delas.

2.3.2.1 Limpeza dos dados

1. **Valores ausentes**: Valores ausentes estão presentes nos dados do mundo real. Podem haver várias razões pelas quais eles ocorrem - desde erros humanos durante a inserção de dados em um sistema, leituras incorretas de sensores, até erros de software no fluxo de processamento de dados. Porém, é importante que eles sejam tratados, visto que alguns algoritmos não suportam valores ausentes como entrada (TANG; KASSIM; ABUBAKAR, 1996). Existem algumas técnicas para lidar esse tipo de problema.[3]

- ❑ **Remoção dos Valores**: Essa abordagem consiste em eliminar as amostras ou atributos que contém valores ausentes. Apesar de simples, é arriscada, pois corre-se

[3] Essas mesmas técnicas podem ser também aplicadas quando lida-se com valores inconsistentes, como *outliers* no conjunto de dados.

o risco de se eliminar informações relevantes ou muitas amostras. Dessa forma, é recomendada quando o conjunto de dados é grande o suficiente, ou o número de valores ausentes é pequeno.

- ❑ **Imputação de Valores**: Consiste em substituir os valores ausentes de uma coluna pela média, moda, ou mediana dos dados existentes na mesma coluna. É uma abordagem mais segura quando comparada à remoção dos valores.

  Para valores categóricos ausentes, usa-se a moda na substituição (classe que mais aparece na coluna).

2.3.2.2 Transformação dos dados

1. **Escalonamento de atributos**

   Muitas técnicas de AM são sensíveis à escala de seus dados (AKSOY; HARALICK, 2001), por isso é importante executar o Escalonamento de Atributos. Esse método consiste em limitar o intervalo de atributos para que eles possam ser comparados em bases comuns. É realizado em atributos de variáveis contínuas.

   Em muitos casos, o conjunto de dados conterá atributos altamente variados em magnitude, unidades e alcance, como por exemplo:

   - Número de quartos (1 a 3)
   - Área (50 a 150 m^2)
   - Preço do imóvel (150.000 a 600.000 reais)

   Se tentarmos aplicar métodos baseados em distância, como K-Vizinhos Mais Próximos, nesses atributos, aquele com o maior intervalo dominará os resultados e obteremos previsões menos precisas. Para suprimir esse efeito, precisamos trazer todos os atributos para o mesmo nível de magnitude.

   Existem diversas técnicas para realizar o Escalonamento de Atributos (RASCHKA S., 2014). Algumas delas são:

   - ❑ **Patronização ou Normalização por escore-Z**

     O resultado da padronização é tal que todos os atributos possuirão propriedades de uma distribuição normal padrão, com média $\mu = 0$ e desvio padrão $\sigma = 1$. O escore-z das amostras pode ser calculado da seguinte forma:

     $$z = \frac{x - \mu}{\delta}$$

     Onde:

     - x = valor antigo

- μ = média de todos valores da coluna
- δ = desvio padrão da coluna

❑ **Escalonamento Mínimo-Máximo**:

Consiste em trazer os valores para uma intervalo definido, normalmente entre 0 e 1. O custo de ter esse intervalo limitado - diferente da padronização - é que obtém-se valores menores de desvios-padrão, o que pode suprimir o efeito de *outliers*. Pode ser aplicado da seguinte maneira:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Onde:

- X = valor antigo
- X_{min} = valor mínimo na coluna
- X_{max} = valor máximo na coluna

2. **Transformação de atributos categóricos**

Em projetos de AM, é muito comum se deparar com atributos categóricos em um conjunto de dados ("Objeto", seria um exemplo de uma variável categórica, com as duas classes: Livro e Lápis) (BISHOP et al., 1977).A maioria dos algoritmos de AM não lida bem com atributos categóricos, sendo necessário convertê-los para atributos numéricos.

Existem algumas técnicas de transformação de atributos categóricos, que serão abordadas neste item através de um exemplo.

Considere o seguinte conjunto de dados:

Tabela 1 – Exemplo de conjunto de dados para aplicação de transformação de atributos categóricos

ID	Objeto	Cor
1	Lápis	Verde
2	Caneta	Vermelho
3	Lápis	Vermelho
4	Caneta	Azul
5	Caderno	Verde

Os atributos *Objeto* e *Cor* são categóricos, necessitando assim serem transformados para numéricos.Agora aplicam-se diferentes métodos de transformação de atributos categóricos no conjunto de dados acima.

- **Codificação por Rótulo (*Label Encoding*)** : Converte cada valor de texto em um número inteiro. Codifica os rótulos com valor entre 0 e $n_{classes} - 1$. Observe:

Tabela 2 – Aplicação do método de Codificação por Rótulo

ID	Objeto	Cor
1	0	0
2	1	1
3	0	1
4	1	2
5	2	0

O problema deste método é que como há diferentes números na mesma coluna, o modelo pode se confundir e tentar construir algum tipo de relação entre eles, como por exemplo uma relação de ordem

$$(0 < 1 < 2)$$

, e este não é o caso. Para solucionar esse problema, utiliza-se o método Codificação *OneHot*, apresentado abaixo.

- **Codificação *OneHot***:

O que o método da Codificação *OneHot* faz é usar uma coluna com dados categóricos, que foi codificada por rótulo e, em seguida, dividir a coluna em várias outras colunas. Os números são substituídos por 1s e 0s, dependendo de qual coluna tem qual valor. Observe:

Tabela 3 – Aplicação do método de Codificação *OneHot*

ID	Objeto.Caderno	Objeto.Caneta	Objeto.Lápis	Cor.Azul	Cor.Verde	Cor.Vermelho
1	0	0	1	0	1	0
2	0	1	0	0	0	1
3	0	0	1	0	0	1
4	0	1	0	1	0	0
5	1	0	0	0	1	0

Aplicando-se o Codificador *OneHot* na tabela inicial, as classes se transformaram em colunas onde o número 1 representa o valor afirmativo e o 0 negativo.

2.3.3 Métodos de Reamostragem e Validação Cruzada

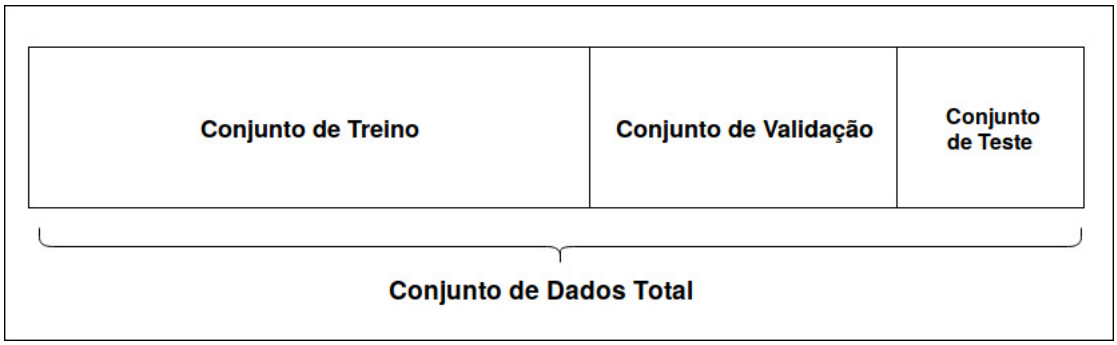
2.3.3.1 Conjuntos de Treino, Teste e Validação

Quando se constrói um modelo estatístico ou de AM necessitamos avaliar sua performance, baseada em alguma métrica. Aditivamente, este "teste de qualidade"deve ser

feito em dados que não foram usados na construção do modelo, assim poderemos inferir o comportamento do mesmo em dados novos (que nunca foram vistos) (KOHAVI, 2001). É para isso que servem os métodos de reamostragem e de validação cruzada.

Esta etapa é uma das mais importantes no processo de criação de um modelo (PICARD; COOK, 1984). Depois que se obtém dados suficientes para que se comece a modelagem, é necessário realizar a segmentação do conjunto de dados em três partes distintas: **Conjunto de Treino**, **Conjunto de Validação** e **Conjunto de Testes**:

Figura 7 – Representação de partição do conjunto de dados



Fonte: Autor

O objetivo deste três conjuntos é:

- **Conjunto de Treino**:

  É neste conjunto que treinamos o modelo. Quando aplicamos um algoritmo, é aqui que ajustamos seus parâmetros. Por exemplo, No caso do treinamento de uma rede neural, utiliza-se o conjunto de treinamento para encontrar os pesos ótimos do modelo.

  Depois que o modelo foi ajustado, podemos calcular o erro de treinamento da seguinte maneira:

$$ErroTreinamento = ValorVerdadeiro - ValorAjustado$$

  Porém, não se pode utilizar essa métrica para avaliar o desempenho do modelo. Caso um modelo for testado com os dados de treino (os mesmos dados em que foi construído) não se pode generalizar sua performance para o ambiente de produção, já que não se sabe qual será seu comportamento em dados nunca vistos. É aí que entra o papel dos outros dois conjuntos.

- **Conjunto de Validação**:

O objetivo deste conjunto é ajustar os parâmetros que foram obtidos no conjunto de treinamento. Em geral, ajustam-se uma quantidade grande de modelos e verificam-se o erro de predição no conjunto de validação.

Ao final deste processo, escolhe-se o modelo com menor erro obtido. O problema é aqui é que como se ajustam muitos modelos, é comum encontrar um modelo que se torna superajustado (incapaz de generalizar para novos dados) para o conjunto de validação, e não funciona para outros conjuntos de dados. Por isso, deixa-se o conjunto de teste para estimar o erro de predição do modelo escolhido e ter certeza de que o modelo não sofrerá superajuste.

- ❑ **Conjunto de Teste**: É utilizado para avaliar o melhor modelo escolhido nas etapas anteriores. Neste passo, a métrica de avaliação é calculada e, caso seja satisfatória para aplicação, envia-se o modelo para a fase de produção.

Geralmente, a proporção de divisão dos dados é dada da seguinte forma:

- 70% dos dados dedicados ao conjunto de treinamento;
- 20% dos dados dedicados ao conjunto de validação;
- 10% dos dados dedicados ao conjunto de testes.

Porém, essa divisão pode variar. Outras possíveis proporções são: 60/20/20, 70/15/15, entre outras.

2.3.3.2 Validação Cruzada

A Validação Cruzada (VC) é um método estatístico de avaliação e comparação de algoritmos de aprendizado o qual divide o conjunto de dados em dois segmentos: um usado para aprender ou treinar um modelo e o outro usado para validar o modelo (REFAEILZADEH; TANG; LIU, 2009). Em outras palavras, consiste em diferentes técnicas responsáveis pela segmentação dos dados em conjunto de treino, validação e teste, como apresentado na subseção anterior. É, então, utilizada para avaliar quão bom o modelo generaliza para dados que ele ainda não viu.

Existem múltiplas ténicas de VC. Algumas das mais comuns são:

- ❑ **Método *Hold-Out***

O método *Hold-Out* é o tipo mais simples de VC. O conjunto de dados é separado em dois conjuntos, conjuntos de treinamento de testes. O modelo então é treinado no conjunto de treino apenas, e sua performance é avaliada no conjunto de testes, em dados que ainda não foram vistos pelo modelo.

A desvantagem é que este procedimento não utiliza todos os dados disponíveis, e os resultados são altamente dependentes da escolha das amostras para a divisão entre

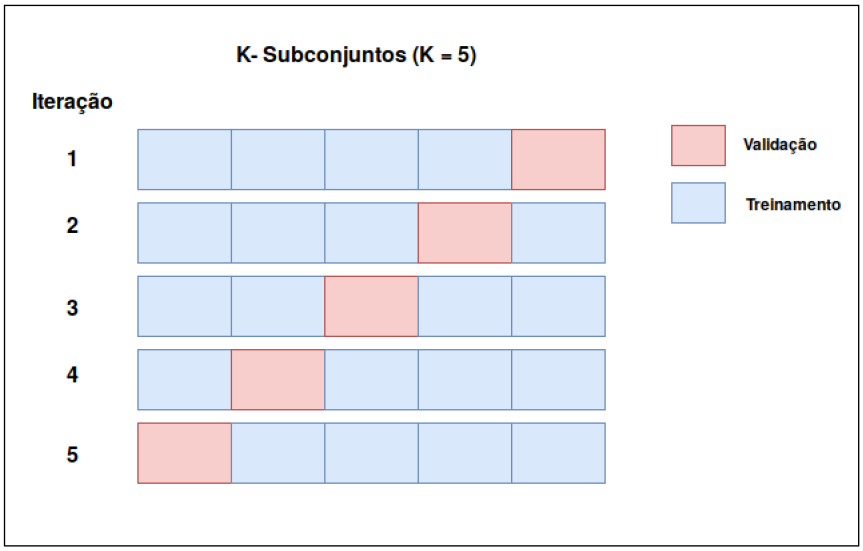
treinamento e teste e, portanto, a avaliação pode ser significativamente diferente dependendo de como a divisão é feita.

❑ **Validação Cruzada *k-fold***

Neste método, o conjunto de dados é dividido em k subconjuntos e o método de validação é repetido k vezes. Cada vez, um dos k subconjuntos é usado como o conjunto de testes e os outros subconjuntos k-1 são colocados juntos para formar um conjunto de treinamento. Em seguida, o erro médio em todas as k tentativas é calculado.

Por exemplo, considere a imagem abaixo.

Figura 8 – Representação do método de validação cruzada k-fold.



Fonte: Autor

Neste caso, o conjunto de dados foi dividido em 5 partes ($k = 5$). Para cada uma dessas partes, o modelo irá usar 4 partes ($k - 1$) para treinar, e usará 1 parte para validar. Ao final do processo, quando o modelo iterar/treinar 5 vezes, é possível obter um verdadeiro score de como seu modelo está generalizando, ao tirar a média do erro de todos os treinamentos.

❑ **Validação Cruzada *k-fold* com repetição**

A VC k-fold com repetição simplesmente repete várias vezes a VC k-fold, onde, em cada repetição, os subconjuntos são divididos de uma maneira diferente (HASTIE; TIBSHIRANI; FRIEDMAN, 2001).

Após cada repetição , a métrica de avaliação do modelo escolhida é calculada. Então, é aplicada uma medida estatística de sumarização, como a média, para obter uma pontuação final na avaliação do modelo.

Esse método é utilizado para garantir uma maior robustez na avaliação do modelo.

❑ ***Bootstrap***

O método *bootstrap* é uma técnica estatística utilizada para estimar quantidades sobre uma população, calculando a média de estimativas a partir de múltiplas amostras menores de dados. (EFRON; TIBSHIRANI, 1997). Ou seja, é o processo em que a amostragem aleatória com substituição é realizada em uma população composta de n observações.

Considere o exemplo: Dada uma população composta pelos elementos (1,2,3,4,5), busca-se obter duas amostras aleatórias com substituição de tamanho 3. Dessa forma, a primeira amostra pode ser dada por (1,2,2) e a segunda (4,4,3).

No contexto da validação da performance de um modelo, o método *Bootstrap* é realizado das seguinte maneira (BOWNLEE J, 2018):

- Escolhe-se um número de amostras do conjuntos;
- Escolhe-se o tamanho da amostra;
- Ajusta-se o modelo na amostra de dados;
- Avalia-se o modelo em amostras que não foram escolhidas;
- Calcula-se a média dos valores resultantes da avaliação.

2.4 Treinamento do Modelo

Existem muitos algoritmos de AMs que são utilizados para treinar um modelo. Esta seção aborda uma introdução a respeito do funcionamento de alguns dos algoritmos mais utilizados, dentro do espectro do aprendizado supervisionado.

2.4.1 Regressão Linear

Regressão linear é um modelo que assume uma relação de linearidade entre as variáveis de entrada x_n e a variável dependente única de saída y (SEAL, 1967). De forma genérica, o modelo linear faz uma previsão computando uma soma ponderada dos atributos de entrada, acrescida de uma constante chamada de *bias* (ou termo de interceptação), como mostrado na equação abaixo.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

Onde:

- $\hat{y}$ é o valor predito;

– n é o número de atributos;

– x_i é o $i_{ésimo}$ valor do atributo.

Pode-se reescrever a equação acima em sua forma vetorizada, sendo esta dada por:

$$\hat{y} = h_\theta(x) = \theta^T x$$

Onde:

– θ é o vetor de parâmetros do modelo, contendo o termo de interceptação θ_0 e os pesos dos atributos θ_1 até θ_n ;

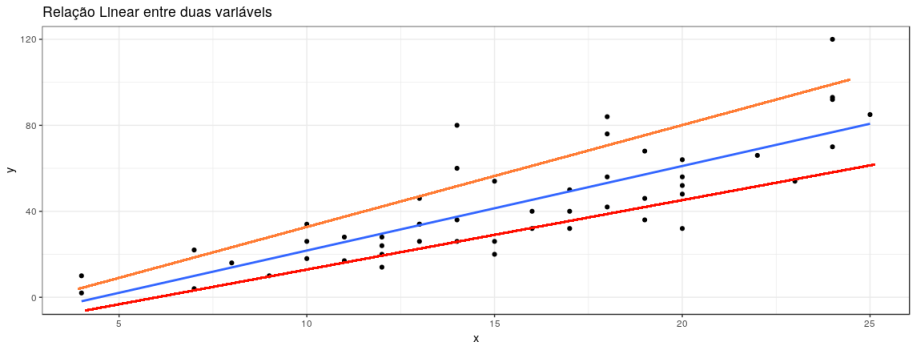
– θ^T é a transposta de θ;

– x é o vetor de instâncias dos atributos, contendo os valores de x_0 até x_n, onde x_0 é sempre igual a 1;

– h_θ é a função hipótese, que utiliza dos parâmetros θ do modelo.

Note que o valor de $\hat{y}$ é alterado pela mudança dos valores relativos de θ_n de um termo para outro, e, consequentemente, a linha de regressão gerada também se altera. (RUSSELL; NORVIG, 2009b). A figura abaixo mostra diferentes linhas de regressão para diferentes valores de θ.

Figura 9 – Relação Linear entre duas variáveis com diferentes valores de θ



Fonte: Autor

Como os diferentes valores de θ_n acarretam em diferentes ajustes da linha de regressão no plano, o objetivo é encontrar os valores de θ_n que promovem o melhor ajuste da linha de regressão aos pontos de dados. Em outras palavras, é preciso encontrar os valores de θ_n que minimizam o erro da métrica de avaliação escolhida, e, consequentemente, geram a linha que melhor se ajusta aos pontos de dados. Introduz-se o conceito de Função Custo.

2.4.1.1 Função Custo

A função custo é um parâmetro que mede, baseado em uma métrica escolhida, e dado um valor particular para os θ, o quão próximos os $\hat{y}$ estão dos valores verdadeiros correspondentes. Isto é, quão bem um determinado conjunto de pesos prediz o valor alvo. Dessa forma, minimizando a Função Custo, minimiza-se o erro da métrica escolhida, e obtém-se os valores de θ_n que gerarão o modelo melhor ajustado aos pontos do conjunto de dados (MASON et al., 1999).

Considere como exemplo a métrica de avaliação do Erro Quadrático Médio.[4] Sua função custo é dada por:

$$EQM(X, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$$

Onde:

- m é o número de instâncias no conjunto de dados as quais se mede o Erro Quadrático Médio;
- X é uma matriz contendo todos os valores dos atributos (excluindo seus rótulos) de todas as instâncias no conjunto de dados.

A função custo varia com a métrica utilizada para a avaliação do modelo, e, para minimizá-las, existem metodologias diferentes, como Gradiente Descendente, Equações Normais, entre outras. Técnicas de otimização fogem do escopo deste trabalho, portanto, nos limitaremos a indicar sua existência.

2.4.2 Árvore de Decisão

Árvore de Decisão (AD) é uma técnica de AM supervisionado, utilizado para resolver problemas tanto de regressão quanto de classificação, que funciona na abordagem "dividir para conquistar"(BREIMAN et al., 1984). O algoritmo utiliza de uma estrutura de árvores para representar possíveis diferentes caminhos para uma decisão, assim como resultados diferentes para cada caminho. Assim, no decorrer do processo, a população (conjunto de dados total) sofre uma divisão em duas ou mais partes (amostras homogêneas) com base no atributo mais significativo.

Considere a seguinte terminologia e a figura que segue:

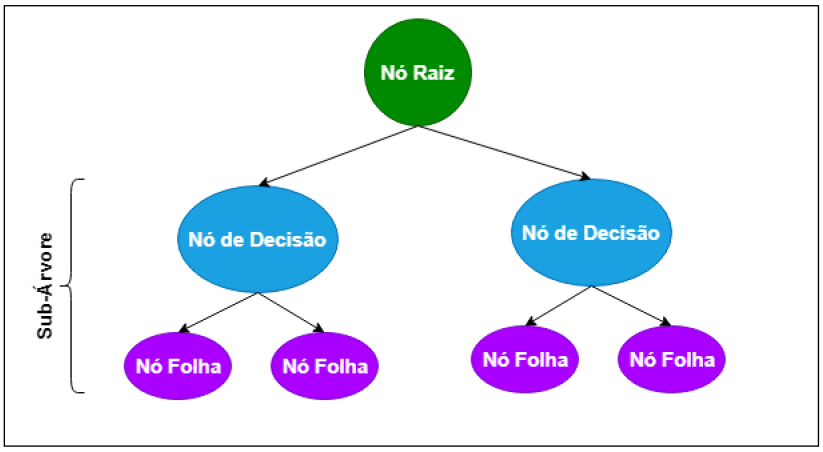
- **Nó Raíz**: Representa todo o conjunto de dados que sofrerá divisão em dois ou mais conjuntos homogêneos.
- **Nó de Decisão**: O nó de decisão é criado quando um nó é dividido em subnós adicionais.

[4] As métricas de avaliação dos modelos são abordadas na seção 3 deste capítulo.

- **Nó Folha**: O nó folha é criado quando não há mais possibilidade de divisão dos nós, sendo também chamado de nó terminal.

- **Sub-Árvore**: É uma subseção da ávore inteira.

Figura 10 – Representação das terminologias em uma árvore de decisão

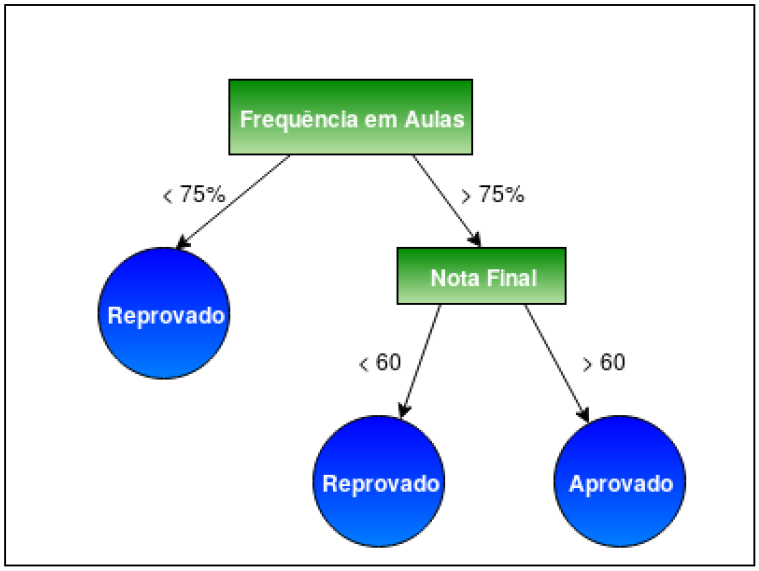


Fonte: Autor

Em cada nó no caminho do Nó Raíz para o Nó Folha, o Nó de Decisão sucessor é escolhido com base em uma divisão do conjunto de dados de entrada. Normalmente, a divisão é baseada em uma dos atributos do conjunto ou em um conjunto pré-estabelecido de regras de divisão (QUINLAN, 1986).

Para ilustrar o funcionamento de uma AD, utilizaremos como exemplo uma AD binária e de classificação. Como explicitado no início da sessão, vale ressaltar que AD também pode ser utilizada para resolver problemas de regressão. Considere a seguinte AD:

Figura 11 – Representação de uma árvore de decisão binária de classificação



Fonte: Autor

Para verificar se um determinado aluno será aprovado ou não em uma disciplina, a AD primeiro examina a frequência do aluno nas aulas. Se esta frequência não for maior que 75%, então a árvore imediatamente prevê que o aluno será reprovado sem testes adicionais. Caso contrário, a árvore se volta para examinar a nota final do aluno. Se a nota final do aluno for maior que 60, a AD prevê que o aluno será aprovado. Caso contrário, a previsão é "reprovado".

O exemplo anterior ressalta uma das principais vantagens das AD - o classificador resultante é muito simples de entender e interpretar (SHALEV-SHWARTZ; BEN-DAVID, 2014).

2.4.3 Floresta Aleatória

FA é um algoritmo derivado das AD que faz parte da família de algoritmos de AM denominados *ensemble*[5], e que pode ser usado em tarefas de regressão e classificação (AMIT; GEMAN, 1997).

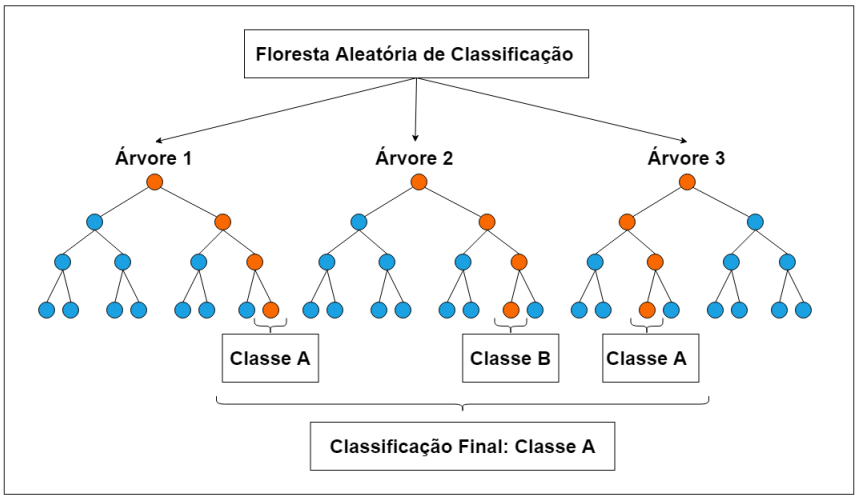
As AD são atraentes por possuirem muitas vantagens – o seu funcionamento é de fácil compreensão, o treinamento do modelo geralmente é direto, e as classificações e previsões são extremamente rápidas (HO, 1995). Porém, as AD podem se ajustar quase perfeitamente a seus dados em treinamento, o que pode fazer com que o modelo não obtenha bons resultados em dados ainda desconhecidos. Para minimizar este problema, utilizam-se as Florestas Aleatórias.

[5] Algoritmos *ensemble* são aqueles que combinam variadas técnicas de AM a fim de obter um melhor desempenho preditivo do que quando comparado aos resultados obtidos apenas com um dos algoritmos que o compõe

O algoritmo opera construindo diversas AD, as quais são treinadas em amostras diferentes do conjunto de dados, e que geram a classe resultante como a moda de todas as classes obtidas (classificação), ou utiliza da previsão média das árvores individuais (regressão).

Considere o exemplo a seguir:

Figura 12 – Representação de uma Floresta Aleatória de classificação

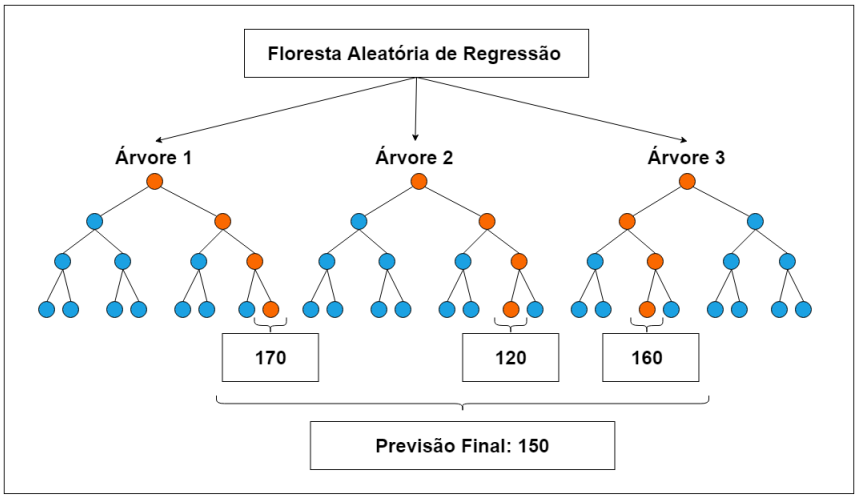


Fonte: Autor

No caso acima, a FA composta por três AD, baseado na moda (valor mais frequente obtido) de todos os resultados individuais de cada árvore, obteve como classificação final a Classe A.

O entendimento para uma aplicação de regressão é similar. Observe:

Figura 13 – Representação de uma Floresta Aleatória de regressão



Fonte: Autor

Agora, utiliza-se da média obtida considearando os resultados das três árvores. Portanto, a previsão final é dada pelo valor 150.

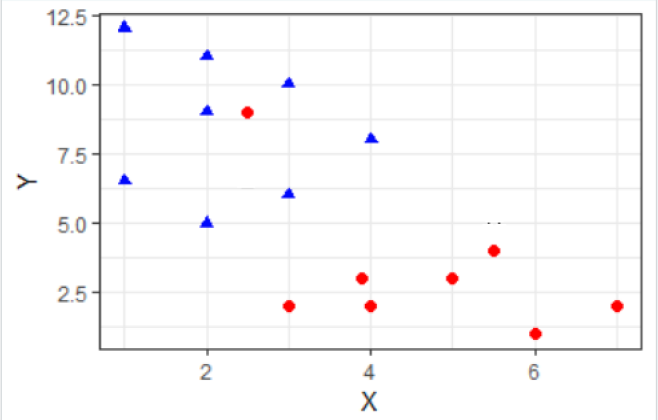
2.4.4 K-Vizinhos Mais Próximos

O algoritmo dos K-Vizinhos Mais Próximos (KVP) é um método de aprendizado utilizado para solucionar problemas de regressão e classificação. Esse algoritmo é considerado um dos algoritmos de AM mais simples, visto que não possui premissas matemáticas e não requer nenhum tipo de poder computacional alto (COOMANS; MASSART, 1982). Por ser mais comum em aplicações de classificação, as explicações e exemplificações a seguir serão baseadas neste tipo de tarefa.

A ideia do algoritmo é memorizar o conjunto de treinamento e, após isso, prever a classe de qualquer nova instância baseada na classe de seus vizinhos mais próximos no conjunto de treinamento (ALTMAN, 1992). A lógica por trás deste método é baseada na suposição de que os atributos que são utilizados para descrever os pontos de domínio são relevantes para sua classificação de tal maneira que pontos próximos pertençam à mesma classe. Dessa forma, o K-Vizinho Mais Próximos utiliza a distância entre os pontos no espaço para decidir a que classe os novos dados pertencem.

Para ilustrar de forma mais clara o funcionamento do algoritmo, considere o seguinte conjunto de dados composto por duas classes: triângulos azuis e círculos vermelhos.

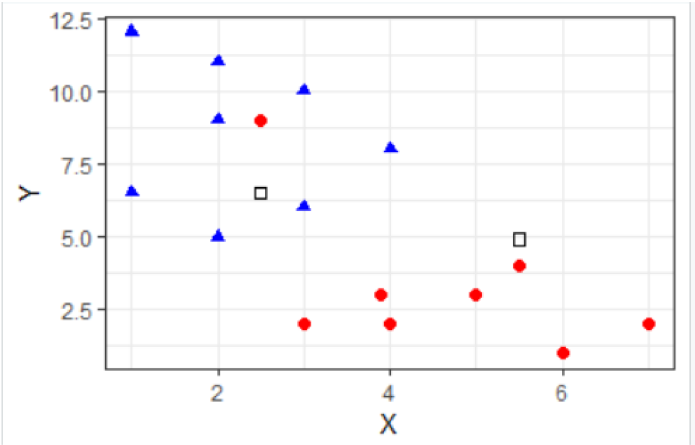
Figura 14 – Exemplo de aplicação K-Vizinhos Mais Próximos - Parte I



Fonte: Autor

Agora, dado um conjunto de pontos de dados não classificados, o algoritmo terá que alocá-los a um grupo já existente (triângulos ou círculos), analisando o conjunto de treinamento. Observe que os pontos não classificados estão sinalizados como quadrados.

Figura 15 – Exemplo de aplicação K-Vizinhos Mais Próximos - Parte II



Fonte: Autor

Utilizando de uma análise gráfica simples, pode-se atribuir os pontos não classificados a um grupo, observando a que grupo seus vizinhos mais próximos pertencem. Isso significa que um ponto próximo a um grupo de “Triângulos Azuis"tem uma probabilidade maior de ser classificado como "Triângulo Azul". Intuitivamente, pode-se inferir que o primeiro ponto $(2.5, 7)$ deve ser classificado como "Triângulo Azul"e o segundo ponto $(5.7, 4.9)$ deve ser classificado como "Círculo Vermelho".

Através do exemplo, foi possível identificar as novas classes intuitivamente. O algoritmo dos KVP, porém, utiliza a distância entre os novos pontos e os pontos já existentes no espaço para dizer a que grupo os novos pontos pertencem. O K, presente no nome KVP, representa o número de vizinhos que serão considerados para realizar a classificação.

A distância mais utilizada no algoritmo dos KVP é a Euclidiana. Considere A e B os vetores $A = (x_1, x_2, x_3, ..., x_k)$ e $B = (y_1, y_2, y_3, ..., y_k)$ A Distância Euclidiana entre A e B, dada por:

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Pode-se simplificar o processo de treinamento do algoritmo nos seguintes passos:

1. O algoritmo recebe um dado não classificado, ou seja, os dados do conjunto de teste, ou novos dados do usuário;

2. Mede a Distância Euclidiana entre novo dado com todos os outros dados que já estão classificados;

3. Obtém as K menores distâncias ao redor do ponto;

4. Verifica a classe de cada um dos dados que tiveram a menor distância obtida, e conta a quantidade de cada classe que aparecer;

5. Toma como resultado a classe que mais apareceu (moda) dentre os dados que tiveram as menores distâncias.

6. Classifica o novo dado com a classe tomada como resultado do passo anterior.

Em problemas de regressão o racional de funcionamento do algoritmo é o mesmo. A diferença é tal que ao invés de se utilizar a moda das classes obtidas no vetor dos vizinhos mais próximos, é utilizada a média dos valores desses pontos.

2.4.5 Naive Bayes

Naive Bayes (NB) é um algoritmo de classificação supervisionada baseado no teorema de Bayes, sendo assim, um algoritmo probabilístico. Ele é chamado *Naive* (inocente) pelo fato de funcionar supondo que todos os atributos são independentes entre si (ZHANG, 2004). No entanto, sabe-se que a independência de atributos pode não estar presente em cenários do mundo real.

Tomemos como exemplo a classificação de uma fruta como maçã, ou não. Uma fruta pode ser considerada uma maçã se for vermelha, redonda e com cerca de 6 cm de diâmetro. Um classificador NB considera que cada uma desses atributos (cor, formato e diâmetro) contribuem independentemente para a probabilidade de que a fruta seja uma maçã, independentemente de quaisquer correlações entre os atributos.

O Teorema de Bayes nos diz como prever a classe de um exemplo desconhecido, dado um exemplo de treinamento conhecido (DUDA; HART, 1973). Em outras palavras, nos ajuda a encontrar uma probabilidade posterior, dada uma certa condição. É dado por:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Onde:

- A = Classe original;
- B = Atributo (Preditor);
- $P(A|B)$ = É a probabilidade posterior da classe (A, alvo) dada preditor (B, atributos);
- $P(B|A)$ = É a probabilidade do preditor dada a classe.
- $P(A)$ = É a probabilidade original da classe;
- $P(B)$ = É a probabilidade original do preditor

Considere um conjunto de dados de treinamento que sugere a possibilidade de se jogar futebol ao ar livre, baseado na condição climática momentânea (à esquerda), e sua Tabela de Frequência (à direita) (ANALYTICS VIDHYA, 2017).

Tabela 4 – Exemplo de conjunto de dados

Clima	**Jogar**
Ensolarado	Não
Nublado	Sim
Chuvoso	Sim
Ensolarado	Sim
Ensolarado	Sim
Nublado	Sim
Chuvoso	Não
Chuvoso	Não
Ensolarado	Sim
Chuvoso	Sim
Chuvoso	Não
Nublado	Sim
Nublado	Sim
Chuvoso	Não

Clima	**Não**	**Sim**
Nublado	0	4
Ensolarado	3	2
Chuvoso	2	3
Total	5	9

Dada a afirmação: "O jogador irá jogar futebol ao ar livre se o tempo estiver ensolarado". Está afirmação está correta?

Pode-se resolver esse problema aplicando o Teorema de Bayes no seguinte formato:

$$P(Sim|Ensolarado) = \frac{P(Ensolarado|Sim)P(Sim)}{P(Ensolarado)}$$

Da Tabela de Frequência, pode-se calcular:

- $$P(Ensolarado|Sim) = \frac{3}{9} = 0,33$$
- $$P(Ensolarado) = \frac{5}{14} = 0,36$$
- $$P(Sim) = \frac{9}{14} = 0,64$$

Então:

$$P(Sim|Ensolarado) = \frac{0.33 \cdot 0.64}{0.36} = 0.60$$

Portanto, pode-se classificar que o jogador irá jogar futebol em um dia ensolarado.

No algoritmo NB, aplica-se a equação do Teorema de Bayes para se calcular a probabilidade posterior de cada classe. A classe com a maior probabilidade posterior é tomada como o resultado da classificação.

2.4.6 Máquina Vetor de Suporte

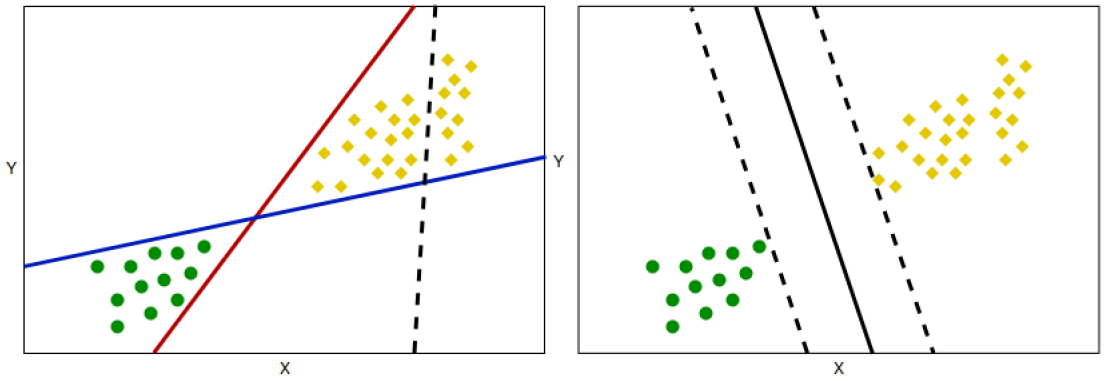
Máquina Vetor de Suporte (MVS) é um modelo de AM supervisionado, capaz de realizar classificações lineares e não-lineares, regressão, e até mesmo detecção de *outliers*. É um dos modelos mais populares no AM (MITCHELL, 1997b). As MVS são utilizadas principalmente para classificação de conjuntos de dados pequenos ou de tamanho médio, porém com alto nível de complexidade(CORTES; VAPNIK, 1995).

Nesse algoritmo, o objetivo é distribuir cada instância de dados em um espaço n-dimensional, onde n é o números de atributos do seu conjunto de dados. Então, a classificação é feita encontrando a linha ou o hiperplano que separa melhor as classes no espaço (BOSER; GUYON; VAPNIK, 1992).

2.4.6.1 Classificação Linear com Máquina Vetor de Suporte

Considere a figura abaixo:

Figura 16 – Exemplo de funcionamento de uma Máquina Vetor de Suporte.



Fonte: Autor

Na figura acima, o objetivo é encontrar um modelo capaz de separar as duas classes no espaço: círculos verdes e losangos amarelos, que são classificadas baseadas em dois atributos: X e Y. É fácil perceber que as classes podem ser facilmente separadas através de uma linha reta, ou seja, elas são linearmente separáveis.

Na imagem da esquerda, são ilustrados três possíveis separadores lineares, representados por uma linha azul, uma linha vermelha, e uma linha pontilhada. O modelo representado pela linha pontilhada possui um ajuste tão inadequado que não consegue separar as classes de forma apropriada no espaço. Já os outros dois modelos, representados pelas linhas azul e vermelha, conseguem separar bem as classes, porém, as instâncias de treinamento se encontram tão próximas do limite de decisão, que provavelmente o modelo não se ajustará bem à novos dados.

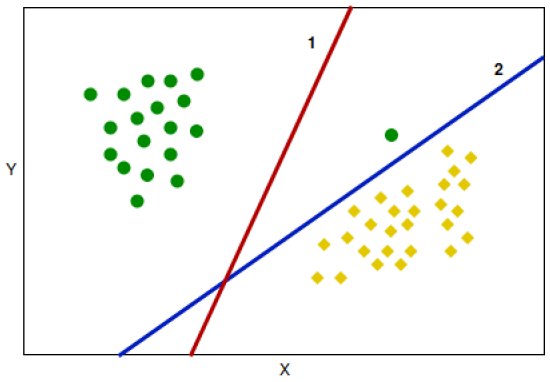
Agora, analise a figura posicionada à direita. A linha preta ao centro da imagem representa o limite de decisão de uma MVS. Neste caso, o modelo não só separa as classes no espaço, como também se posiciona o mais distante possível da instância de treinamento mais próxima.

Se os dados de treinamento forem linearmente separáveis, como mostrado acima, podemos selecionar duas linhas paralelas (linhas pontilhadas) que separam as duas classes de dados, de modo que a distância entre elas seja a maior possível. A região delimitada por essas duas linhas é chamada de "margem", e a linha de margem máxima (linha preta) é aquela posicionada entre as as duas outras paralelas.Dessa forma, maximizar as distâncias entre a instância mais próxima (de qualquer classe) e a linha ou hiperplano, é o critério necessário para se encontrar o separador correto.

O algoritmo de MVS possui duas características importantes:

1. A MVS seleciona o hiperplano que classifica as classes com maior precisão antes de selecionar aquele que possui a maior margem. Considere a figura abaixo:

Figura 17 – Máquina Vetor de Suporte e classificações equivocadas.

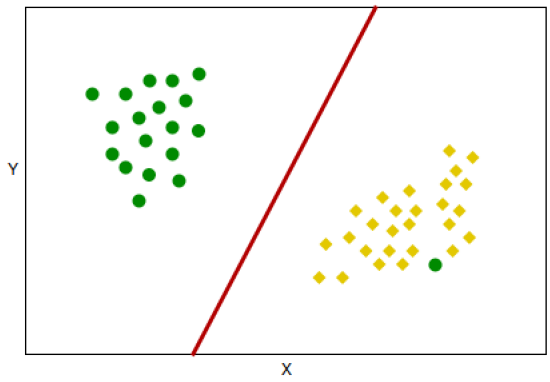


Fonte: Autor

Na figura acima, o modelo representado pela linha 1, em vermelho, apresenta um erro de classificação, apesar de possuir a maior margem. Por outro lado, o modelo representado pela linha 2, em azul, classificou todas as instâncias corretamente. Dessa forma, a MVS optará pelo modelo 2.

2. A MVS possui um recurso para ignorar valores discrepantes (outliers) e encontrar o hiperplano que possui margem máxima. Portanto, pode-se dizer que o algoritmo é robusto para outliers. Por exemplo, na figura abaixo, o algoritmo identifica o ponto discrepante e encontra o melhor hiperplano sem sofrer influência deste valor.

Figura 18 – Máquina Vetor de Suporte e valores discrepantes.



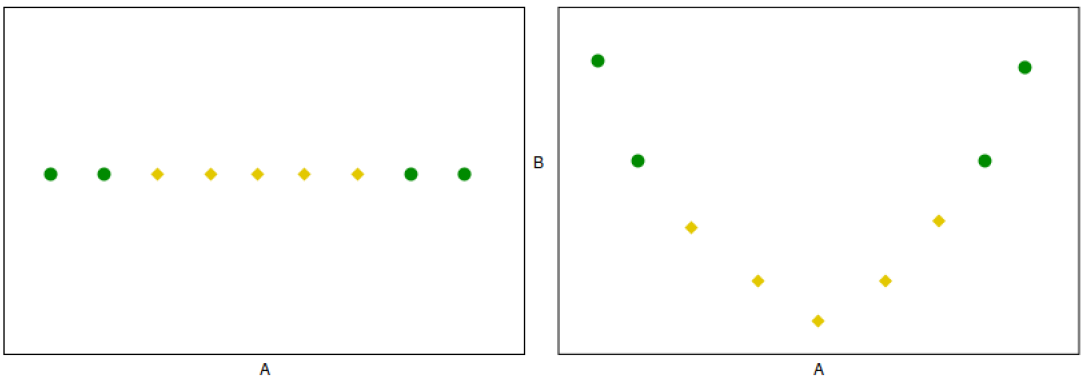
Fonte: Autor

2.4.6.2 Classificação Não Linear com Máquina Vetor de Suporte

Existem casos onde os conjuntos de dados não são linearmente separáveis. Uma forma de lidar com esse problema, é adicionar mais atributos, como atributos polinomiais (BOSER; GUYON; VAPNIK, 1992). Em alguns casos, essa metodologia pode resultar em um conjunto de dados linearmente separável.

Na figura abaixo, o gráfico à esquerda é composto por apenas um atributo A, e não é linearmente separável. Porém, caso seja acrescentado um segundo atributo $B = A^2$, o conjunto de dados resultante se torna linearmente separável.

Figura 19 – Abordagem para conjuntos de dados não lineares na Máquina Vetor de Suporte.



Fonte: Autor

O algoritmo da MVS possui uma técnica chamada Truque do Kernel (*Kernel Trick*)[6], a qual é um conjunto funções que transformam um espaço de entrada dimensional baixo

[6] Discussões a respeito da técnica Truque do Kernel fogem do escopo deste trabalho.

em um espaço dimensional superior (CORTES; VAPNIK, 1995). Isto é, a técnica é capaz converter um problema linearmente não separável em um problema linearmente separável. Essas funções são chamadas de *kernels*, e são principalmente úteis em problemas de separação não linear.

2.4.7 Perceptron Multicamadas

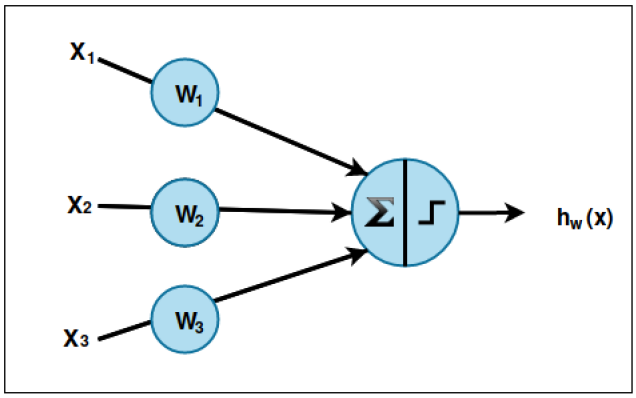
2.4.7.1 Rede Neural Artificial

Inspirando-se no funcionamento dos neurônios biológicos do sistema nervoso dos animais, estabeleceu-se na área da Inteligência Artificial um modelo computacional denominado Rede Neural Artificial. É um modelo preditivo motivado pela forma como o cérebro funciona, capaz de resolver problemas complexos de classificação e regressão.

2.4.7.2 O Perceptron

O Perceptron é a mais simples arquitetura de uma Rede Neural Artificial (ROSENBLATT, 1958), representada da seguinte forma:

Figura 20 – Arquitetura de um Perceptron.



Fonte: Autor

Considerando a estrutura acima, os sinais da entrada do Perceptron são representados pelo vetor $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$. Ao chegarem ao neurônio, são multiplicados pelos respectivos pesos sinápticos, que são os elementos do vetor $w = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$, gerando assim o valor $h_w(x) = z$, comumente denominado potencial de ativação, e definido por:

$$z = w_1x_1 + w_2x_2 + w_3x_3 = w^T \cdot x$$

O valor de z então é aplicado à uma de Função de Ativação para obter seu resultado final. Uma das funções de ativação mais comuns é a Função Degrau, definida por:

$$degrau(z) = \begin{cases} 0, & \text{se } z < 0 \\ 1, & \text{se } z >= 0 \end{cases}$$

Dessa forma, obtém-se:

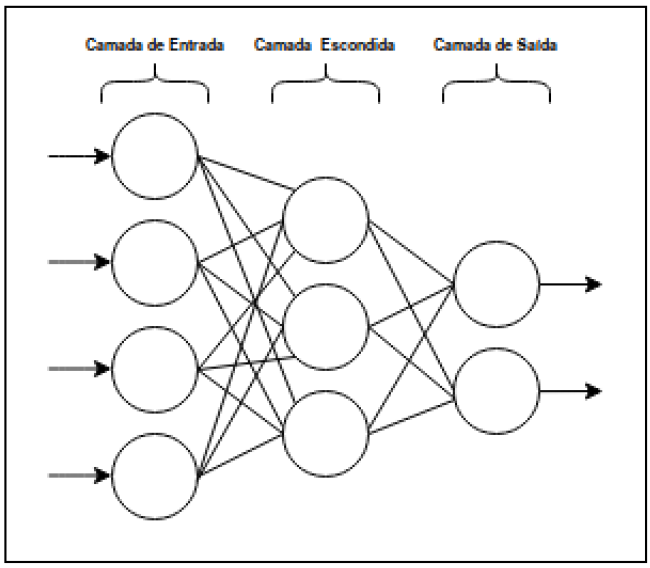
$$degrau(z) = degrau(w^T \cdot x)$$

Uma arquitetura simples como a apresentada, pode ser usada por exemplo, para uma classificação binária em um conjunto linear. O Perceptron calcula uma combinação linear das entradas e, se o resultado for maior que o *treshold* definido pela função de ativação, classifica a classe como a classe positiva, caso contrário, como a classe negativa.

2.4.7.3 O Perceptron Multicamadas

O Perceptron composto por apenas um neurônio não é capaz de resolver problemas muito complexos, porém, pode-se combinar vários neurônios em uma estrutura em camadas. O Perceptron Multicamadas (PMC) é uma rede neural com uma ou mais camadas escondidas, com um número ajustável de neurônios em cada camada, e que possui uma função de ativação não linear (ROSENBLATT, 1959), como mostra a figura a seguir:

Figura 21 – Arquitetura de um Perceptron Multicamadas.



Fonte: Autor

O treinamento de uma rede PMC insere-se no contexto do AM supervisionado. Existem diversos métodos para se treinar um PMC, e um deles comumente utilizado é a Retropropagação do Erro (WERBOS; JOHN, 1974) [7].

2.5 Avaliando modelos de Aprendizado de Máquina

Após a criação de um modelo de AM, é necessário de alguma forma, mensurar o quão bom ele é. Para isso, existem diferentes métricas de avaliação de modelos de AM (STEHMAN, 1997). Esta seção visa introduzir as principais delas.

2.5.0.1 Métricas de Classificação

Considere como exemplo um classificador binário, responsável por classificar emails em duas classes distintas: spam ou não spam, onde spam é a classe positiva, e não spam é a classe negativa. Dessa forma, define-se:

- Verdadeiro Positivo (VP) : significa uma classificação correta de SPAM. Por exemplo, o email era SPAM e o modelo classificou como SPAM.
- Verdadeiro Negativo (VN) : significa uma classificação correta da classe negativa. Por exemplo, o email era comum e o modelo classificou como comum.
- Falso Positivo (FP) : significa uma classificação errada da classe positiva. Por exemplo, o email é comum e o modelo classificou como SPAM.
- Falso Negativo (FN) : significa uma classificação errada da classe negativa. Por exemplo, o email é SPAM e o modelo classificou como comum.
- Acurácia

  A acurácia mede a frequência com que o classificador faz a classificação correta. É dada pela razão entre o número de classificações corretas e o número total de classificações (dado pelo número de instâncias no conjunto de testes).

$$Acurácia = \frac{\#\ classificações\ corretas}{\#total\ de\ classificações}$$

  Ou seja:

$$Acurácia = \frac{(VP + VN)}{(VP + VN + FP + FN)}$$

[7] Métodos como a Retropropagação do Erro fogem do escopo deste trabalho

❑ Matriz de Confusão

A Acurácia é uma boa métrica de avaliação em classifcadores binários. No entanto, não faz distinção entre classes. Isso quer dizer que respostas corretas para a classe negativa e a classe positiva são tratadas igualmente - o que às vezes não é suficiente.

Em alguns casos, pode ser necessário analisar, por exemplo, quantos exemplos falharam para a classe negativa em comparação com a classe classe positiva. Isso porque o custo de uma classificação errada pode diferir para as duas classes, ou pode-se ter muito mais dados de teste de uma classe do que da outra.

Por exemplo, se um médico fizer um diagnóstico de que um paciente tem câncer, quando na verdade ele não tem (falso positivo) as consequências são muito diferentes do que diagnosticar que um paciente não tem câncer, quando ele tem de fato (falso negativo).

Uma matriz de confusão mostra uma análise mais detalhada a respeito das classificações para cada classe (STEHMAN, 1997). Ela mostra a relação entre os valores reais e os valores preditos pelo classificador. Considere que um conjunto de dados de teste contenha 1000 exemplos na classe positiva e 2000 exemplos na classe negativa (ZHENG, 2015). Então, a tabela de confusão deste conjunto pode ser dada por:

Tabela 5 – Exemplo de Matriz de Confusão

	Classificado como Positivo	**Classificado como Negativo**
Rotulado como Positivo	800	200
Rotulado como Negativo	50	1950

Caso seja calculada a Acurácia da classe positiva, temos $\frac{800}{200+800} = 80\%$. Já para a classe negativa, $\frac{1950}{50+1950} = 97.5\%$, obtendo assim uma diferença considerável entre as acurácias das classes. Essa informação é perdida caso seja calculada a Acurácia geral, dada por $\frac{800+1950}{1000+2000} = 91.7\%$

❑ Precisão

A precisão corresponde à fração dada entre os Verdadeiros Positivos e todos os resultados positivos, ou seja:

$$Precisão = \frac{(VP)}{(VP + FP)}$$

Essa métrica expressa a relação entre as instâncias classificadas como positivas, e aquelas que efetivamente são positivas.

❑ Revocação

Essa métrica expressa a proporção de instâncias positivas que foram classificadas corretamente. É dada por:

$$Revocação = \frac{(VP)}{(VP + FN)}$$

- F1

Essa métrica utiliza da Precisão e do Recall de modo a trazer um valor único que indique a qualidade geral do modelo de classificação. É dada pela média harmônica entre Precisão e Revocação, onde seu melhor valor é 1, e o pior é 0.

$$F1 = \frac{2 \cdot (Precisão \cdot Revocação)}{(Precisão + Revocação)}$$

2.5.0.2 Métricas de Regressão

- Erro Quadrático Médio

O Erro Quadrático Médio mede a magnitude média do erro. É dado pela raíz quadrada da média das diferenças quadradas entre o valor da previsão e o valor da observação real (ARMSTRONG et al., 1992). Ou seja, é a diferença entre o valor que foi previsto pelo modelo e o valor real que foi observado, sendo mostrada na unidade da variável dependente.

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Aqui, y_i denota o valor verdadeiro para o $i_{ésimo}$ ponto de dados, e $\hat{y}_i$ denota o valor previsto.

- R^2

O R^2, ou coeficiente de determinação, fornece uma medida de quão bem o modelo foi ajustado em relação aos novos dados, ou seja quão bem as previsões de regressão se aproximam dos valores dos pontos de dados reais. (CAMERON; WINDMEIJER, 1997). É um valor compreendido entre 0 e 1, onde 0 significa nenhum ajuste e 1 significa um ajuste perfeito. É dado por:

$$R^2 = 1 - \frac{EQ_{\hat{y}}}{EQ_{\overline{y}}}$$

Onde:

- $EQ_{\hat{y}}$ = Erro quadrático da linha de regressão;
- $EQ_{\overline{y}}$ = Erro quadrático da linha média $\overline{y}$, correspondente à média de todos os valores y do conjunto de dados.

Capítulo 3

Metodologia

3.1 A linguagem R

R é um ambiente de software de código aberto multiplataforma projetada para análise estatística de dados, e inicialmente desenvolvida por Ihaka e Gentleman em 1996 (IHAKA; GENTLEMAN, 1996). Atualmente, figura-se entre as 10 linguagens mais populares no mundo (IEEE SPECTRUM, 2018), possuindo assim uma robusta documentação, e um excelente suporte *online* entre as comunidades de usuários.

R é uma linguagem de programação procedural, funcional e orientada a objetos, a qual possui milhares de pacotes que podem executar muitas tarefas, porém, é amplamente utilizada em aplicações estatísticas, de análises gráficas e de dados e AM (R PROJECT, 2019). As estruturas de dados nativas do R são vetores, matrizes, arrays, listas e *data frames*.

R é capaz de executar quase todas as computações padrão em conjuntos de dados, o que inclui a implementação de uma infinidade de algoritmos e procedimentos estatísticos e de AM.Por possuir milhares de pacotes escritos por voluntários, muitos deles pesquisadores, é seguro dizer, então, que todos os procedimentos estatísticos concebíveis e algoritmos de AM tem uma ou várias implementações em R. (BARR, 2018).

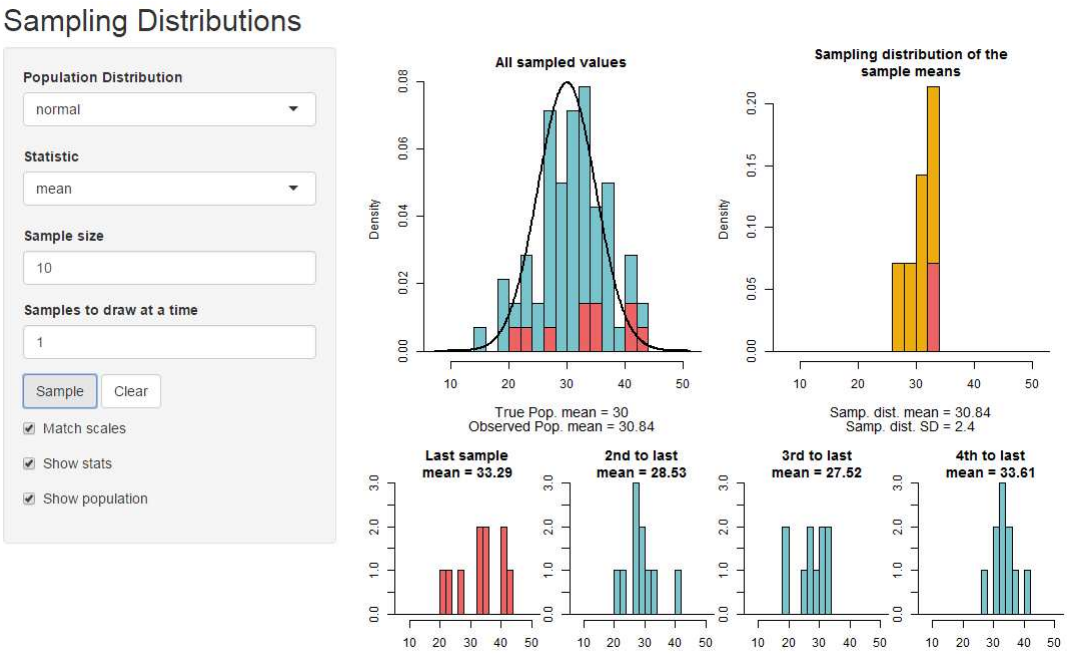
3.2 O pacote *Shiny*

Shiny é um pacote R de código aberto que fornece uma estrutura *Web* elegante e poderosa para a criação de aplicativos *Web* usando R. O *Shiny* tem a função de transformar as análises feitas em R nativo, em aplicativos *Web* interativos sem a necessidade da utilização de linguagens específicas como *HTML*, *CSS* ou *JavaScript* (RSTUDIO, 2019a).

A implementação direta da interface *Web* do *Shiny* torna a ferramenta ideal para criar aplicativos que podem produzir grande eficiência para as tarefas repetitivas que um programador estatístico encontra com frequência (GUNUGANTI, 2018). Além disso, faz

com que usuários não versados em programação de computadores e métodos de análise estatística e de AM possam desfrutar de suas funcionalidades de forma fácil e intuitiva.

Figura 22 – Exemplo de Aplicação Shiny



Disponível em: *http://kylehardman.com/BlogPosts/View/8*

3.3 O pacote *Caret*

Em algumas ferramentas, existem diferentes pacotes que nos ajudam no pré-processamento de dados e na implementação de algoritmos de Aprendizado de Máquina. No R, um dos principais pacotes utilizados neste contexto é o pacote *Caret* (KUMAR, 2018).

Caret é abreviação de *Classification And REgression Training*, sendo um conjunto de funções que objetivam simplificar o processo de criação de modelos preditivos. O pacote contém ferramentas para: divisão de dados, pré-processamento, seleção de recursos, ajuste e treinamento de modelos, estimativa de importância de variável, bem como outras funcionalidades (CARET OFFICIAL DOCUMENTATION, 2019).

Existem muitas funções de modelagem diferentes em R. Porém, essas funções possuem sintaxes diferentes para o treinamento ou previsão dos modelos criados. Dessa forma o pacote fornece uma interface uniforme para as próprias funções, bem como uma maneira de padronizar tarefas comuns (como ajuste de parâmetro e importância de variável).

3.4 Construção da Aplicação

Machine Learning Toolkit é uma aplicação *Web* construída utilizando a linguagem de programação R e o pacote *Shiny*. O objetivo é que o usuário consiga aplicar diferentes algoritmos de AM no seu próprio conjunto de dados de forma simples e intuitiva, sem que sejam necessários conhecimentos prévios aprofundados sobre *AM* e Programação de Computadores.

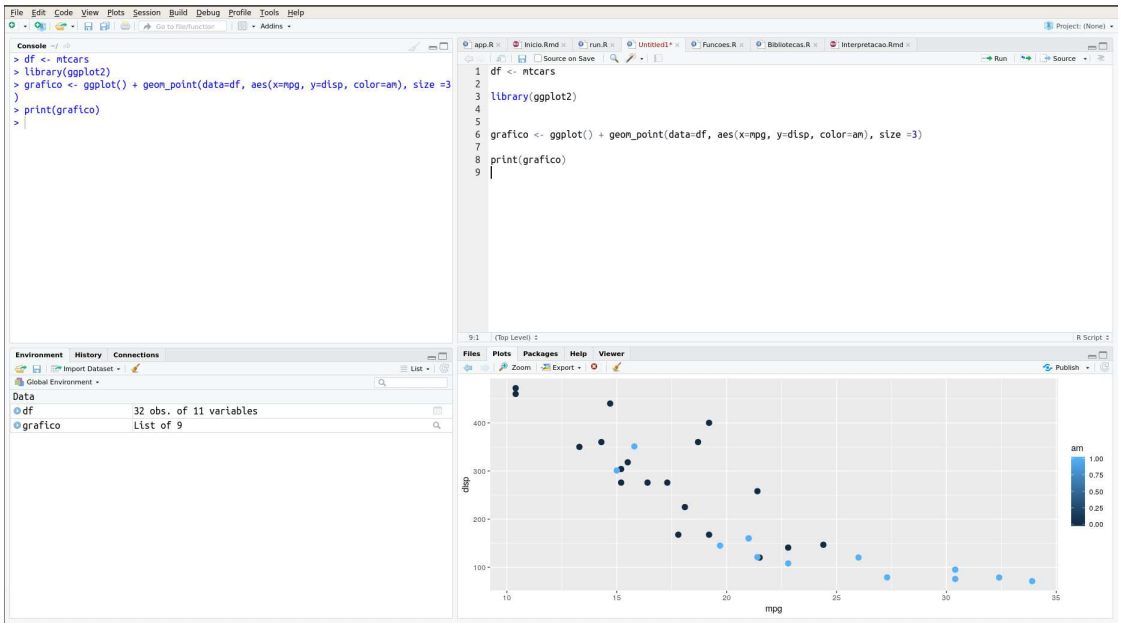
A construção do código fonte da aplicação, foi realizada utilizando o Ambiente de Desenvolvimento Integrado gratuito *RStudio*. A escolha do *RStudio* se deu pelas diversas funcionalidades disponibilizadas pela ferramenta, como por exemplo:

- O RStudio permite uma navegação rápida entre arquivos e funções. Como a aplicação proposta é composta por diversos arquivos distintos, esta funcionalidade facilita a manipulação e implementação de todos eles.

- O RStudio suporta gráficos interativos com os pacotes *Shiny* e *ggvis*. Esta funcionalidade foi importante no processo de criação das análises gráficas apresentadas na ferramenta, permitindo com que testes e diversas implementações diferentes pudessem ser realizadas de forma rápida.

- O RStudio integra as ferramentas R em um único ambiente. Essa funcionalidade permite com que seja possível visualizar, em um único local, as variáveis de ambiente, análises gráficas, documentações de pacotes e funções, etc. Isso fez com que o desenvolvimento da aplicação fosse acelerado, visto que não era necessário navegar por diferentes locais para encontrar informações relacionadas.

Figura 23 – Ambiente de Desenvolvimento Integrado RStudio



Fonte: Autor

A interface gráfica da aplicação foi construída utilizando o pacote *Shiny*, permitindo assim disponibilizar ao usuário funcionalidades interativas, visualizações gráficas, imagens e textos explicativos e vídeos tutoriais.

Os modelos AM implementados utilizam do pacote *Caret*, e mais informações podem ser consultadas no **Anexo B** deste trabalho.

3.5 Avaliação da aplicação

O teste de usabilidade é uma maneira de avaliar um produto ou serviço testando uma amostra de usuários. Esse processo é feito observando o trabalho dos respondentes do teste, dando-lhes algumas tarefas a serem realizadas em determinada aplicação. A observação pode ser feita olhando, ouvindo e gravando com o objetivo de identificar problemas de usabilidade, coletando dados qualitativos e quantitativos e determinando a satisfação do respondente com o produto ou serviço (BASYIR et al., 2019).

Para avaliar se os objetivos propostos pela aplicação foram cumpridos, foram selecionados 23 usuários voluntários de diferentes idades, perfis profissionais, e áreas de interesse e atuação. Os usuários foram submetidos à um teste de usabilidade, composto pelas seguintes etapas:

1. Introdução à proposta da aplicação

   Nesta etapa, foi apresentado o objetivo da aplicação, as etapas do processo de avaliação as quais o usuário seria submetido, e a localização de arquivos necessários para

a realização das tarefas propostas. Além disso, foram sanadas quaisquer dúvidas apresentadas nesta fase inicial.

2. Apresentação das tarefas a serem cumpridas pelo usuário

   Essa etapa consistiu em explicitar ao usuário quais seriam as tarefas as quais ele deveria cumprir dentro da aplicação. As tarefas foram elaboradas com o intuito de avaliar as principais funcionalidades do Machine Learning Toolkit. São elas:

   - ❏ Criar um modelo

     O usuário teve como objetivo criar um modelo de AM dentro da aplicação. Aqui, não houveram restrições quanto ao tipo ou especificidade de modelo que deveria ser criado. Dessa forma, o usuário teve a liberade de explorar as opções disponíveis na aplicação e escolher aquele que melhor lhe conviesse.

     Foram disponibilizados aos usuários dois conjuntos de dados diferentes para que esta tarefa pudesse ser cumprida, um adequado para aplicações de regressão, e outro para classificação.

   - ❏ Exportar o modelo criado

     O usuário teve como objetivo exportar o modelo criado na tarefa anterior para a máquina que estava utilizando. Essa tarefa só é possível de ser cumprida caso o usuário tenha obtido êxito na criação do modelo.

   - ❏ Realizar Previsões ou Classificações

     Utilizando o modelo criado na primeira tarefa, o usuário teve como objetivo realizar previsões ou classificações em novos dados, dentro da aplicação. Para esta tarefa, também foram disponiblizados conjuntos de dados distintos ao usuário para que ele pudesse cumpri-la.

3. Uso da aplicação

   Esta etapa consiste no efetivo uso da aplicação pelo usuário para realizar as tarefas propostas. Aqui, não houve nenhuma supervisão ou intervenção do autor durante o processo[1].

4. Resposta do questionário proposto

   Após a realização das tarefas propostas, o usuário foi submetido à um questionário de avaliação do sistema composto por duas etapas. Na primeira delas, foram apresentados questionamentos propostos pelo autor, já na segunda, questionamentos propostos pelo método *System Usability Scale*[2].

[1] As únicas intervenções realizadas nessa etapa, em alguns casos, se deram a fim de solucionar problemas técnicos com a máquina, ou com os arquivos necessários para a boa condução do teste.

[2] Detalhes referentes aos questionamentos são explicados nas próximas subsessões desta mesma sessão

5. Discussão da experiência com o autor

   Ao fim da atividade, foi realizada uma conversa informal entre o usuário e o autor. Nesta fase, o intuito foi apresentar outras possibilidades e funcionalidades da aplicação que não foram exploradas durante o teste de usabilidade, e coletar *feedbacks* a respeito da experiência vivenciada pelo usuário.

3.5.1 Questionário proposto pelo autor

O questionário proposto pelo autor é dividido em 3 seções:

1. Informações Preliminares: visa obter informações a respeito da escolaridade e áreas de atuação profissional e interesse do usuário.

2. Análise do Perfil do Usuário: visa entender o perfil técnico do usuário, avaliando o nível de conhecimento prévio a respeito de conhecimentos como AM e programação de programadores.

3. Análise da Experiência do Usuário: busca compreender de que forma o usuário interagiu com o sistema, se conseguiu ou não cumprir as tarefas propostas, e quais suas considerações a respeito da interface, facilidade de navegação, compreensão do objetivo da aplicação, entre outras.

Neste questionário, o usuário deve optar por uma resposta para cada item. As respostam variam em uma escala de 1 a 5, onde 1 significa "Discordo Fortemente", e 5 significa "Concordo Fortemente"

O questionário proposto pelo autor pode ser consultado no **Anexo C**.

3.5.2 *System Usability Scale - SUS*

3.5.2.1 O que é o SUS

O *System Usability Scale - (SUS)* fornece uma metodologia confiável e rápida para medir a usabilidade de um produto, incluindo aplicativos, sistemas interativos, *softwares*, etc (BANGOR; KORTUM; MILLER, 2009).

Originalmente criado por John Brooke em 1986, o SUS consiste em um questionário de 10 itens com cinco opções de respostas para os entrevistados. As respostam variam entre 1 e 5, onde 1 significa "Concordar Fortemente", e 5 significa "Discordar Fortemente"(BROOKE, 1996).

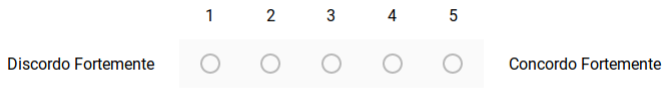
Os SUS pode ajudar a avaliar importantes critérios de uma aplicação:

- ❏ Efetividade: Avalia se os usuários conseguem completar os objetivos propostos.

- ❏ Eficiência: Avalia quanto esforço é necessário para que o usuário consiga completar os objetivos propostos.
- ❏ Satisfação: Avalia a experiência do usuário.

As perguntas que compõe o método SUS podem ser consultadas no **Anexo C**.

Figura 24 – Escala de avaliação do SUS



Fonte: Autor

3.5.2.2 Como calcular a pontuação final

Existem alguns passos necessários para se calcular a pontuação final do SUS. São eles:

1. Para as respostas ímpares (1, 3, 5), é necessário subtrair 1 da pontuação que o usuário respondeu.
2. Para as respostas pares (2 e 4), é necessário subtrair a resposta de 5. Ou seja, se o usuário respondeu 2, contabilize 3. Se o usuário respondeu 4, contabilize 1.
3. Agora, é necessário somar todos os valores das dez perguntas, e multiplicar o resultado por 2.5.
4. Após repetir o procedimento para cada usuário, extrair a média dos resultados entre todos os usuários.

O resultado obtido é a pontuação final, um valor compreendido entre 0 e 100.

3.5.2.3 Como interpretar o resultado

O resultado do SUS pode ser interpretado de algumas maneiras diferentes (SAURO, 2018). Serão abordadas neste trabalho a interpretação de Notas e Adjetivo.

1. Notas: As notas variam de A (indica desempenho superior), até F (para desempenho insatisfatório), com C indicando “média”.
2. Adjetivo: Consiste na ideia de usar palavras em vez de números para descrever a experiência do usuário. A escala contém adjetivos, incluindo "Excelente", "Bom", "Regular"e "Ruim", palavras que os usuários associam com frequência de forma direta à usabilidade de um produto.

As interpretações seguem a tabela abaixo:

Tabela 6 – SUS - Tabela de interpretação dos resultados

Nota	SUS	Adjetivo
A	80.4 - 100	Excelente
B	68.1 - 80.3	Bom
C	68	Regular
D	51 - 67.9	Ruim
F	0 - 51	Péssimo

Capítulo 4

Machine Learning Toolkit

4.0.1 Visão Geral da aplicação

A aplicação é dividida em cinco grandes abas principais:

1. **Página Inicial**

   Apresenta uma visão geral sobre a aplicação, explicita os objetivos principais da mesma, fornece instruções de utilização e apresenta conceitos introdutórios a respeito de Aprendizado de Máquina.

Figura 25 – Machine Learning Toolkit - Página Inicial

Fonte: Autor

2. **Informações sobre os modelos**

   Apresenta as informações necessárias para a compreensão de quaisquer características dos modelos ou termos apresentados na aplicação. É dividida em 5 sub-abas:

- ❏ Visão Geral dos Modelos

  Apresenta uma introdução a respeito do funcionamento dos modelos de AM disponíveis na aplicação.

- ❏ Pré-Processamento de Dados

  Apresenta conceitos introdutórios a respeito de técnicas essenciais de pré-processamento de dados.

- ❏ Hiperparâmetros dos Modelos

  Lista os hiperparâmetros utilizados por cada modelo presente na aplicação, bem como busca explicar de forma breve o significado de cada um deles.

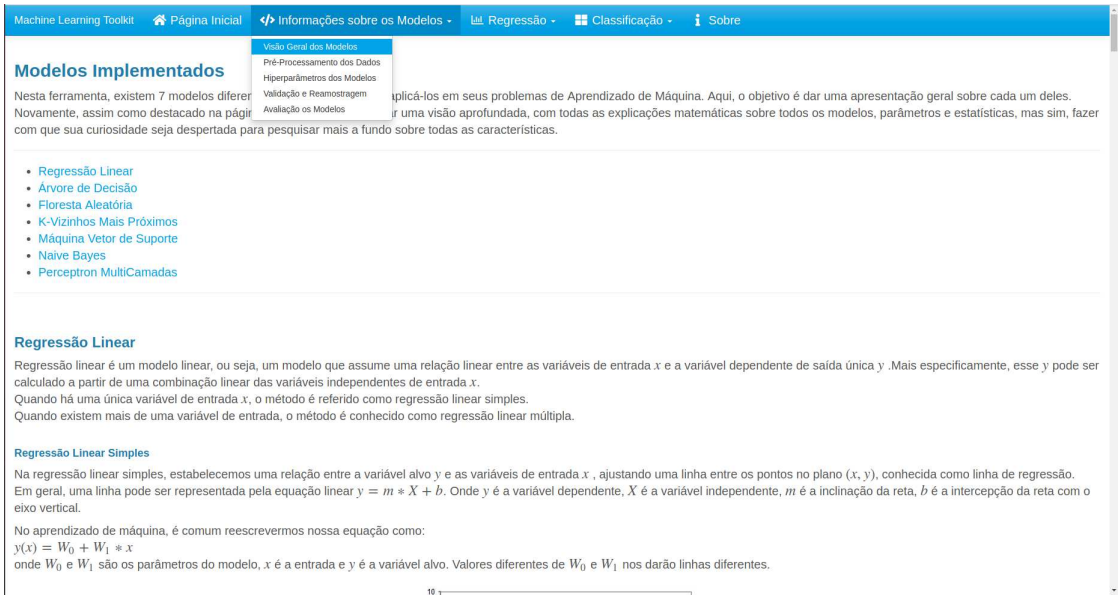
- ❏ Validação e Reamostragem

  Apresenta de forma introdutória os conceitos e técnicas principais a respeito de técnicas de validação e reamostragem disponíveis na aplicação, e utilizadas no processo de treinamento em AM.

- ❏ Avaliação dos Modelos

  Apresenta conceitos introdutórios a respeito dos métodos de avaliação dos modelos de regressão e classificação disponíveis na aplicação.

Figura 26 – Machine Learning Toolkit - Informações sobre os modelos



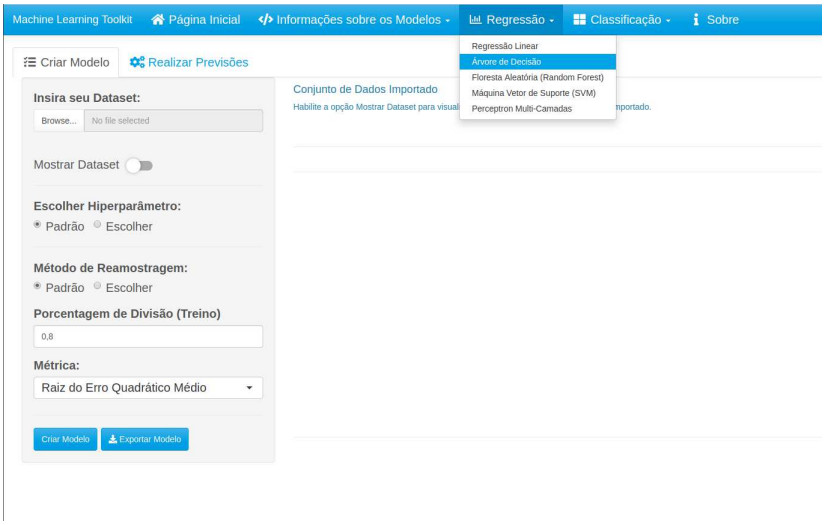
Fonte: Autor

3. **Regressão**

   Essa aba é responsável pela criação e utilização dos modelos de regressão disponíveis na ferramenta. São dispononibilizados os seguintes algoritmos:

- ❏ Regressão Linear;
- ❏ Árvore de Decisão;
- ❏ Floresta Aleatória;
- ❏ Máquina Vetor de Suporte;
- ❏ Perceptron Multicamadas;

Figura 27 – Machine Learning Toolkit - Regressão



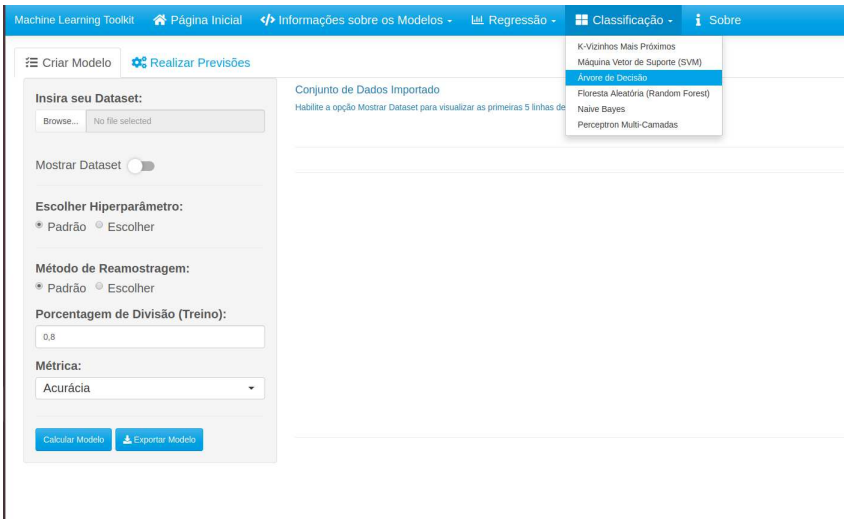
Fonte: Autor

4. **Classificação**

Essa aba é responsável pela criação e utilização dos modelos de regressão disponíveis na ferramenta. São dispononibilizados os seguintes algoritmos:

- ❏ K-Vizinhos Mais Próximos;
- ❏ Naive Bayes;
- ❏ Árvore de Decisão;
- ❏ Floresta Aleatória;
- ❏ Máquina Vetor de Suporte;

Figura 28 – Machine Learning Toolkit - Classificação

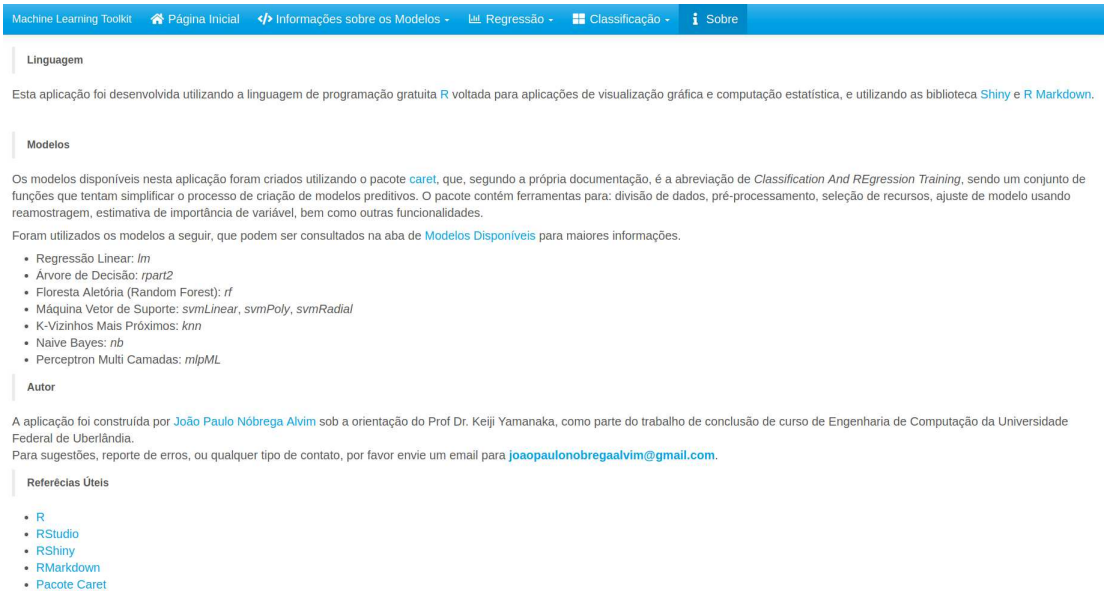


Fonte: Autor

5. **Sobre**

Apresenta informações a respeito da construção da aplicação, as ferramentas principais utilizadas, considerações sobre o autor, informações de contato e referências úteis.

Figura 29 – Machine Learning Toolkit - Sobre



Fonte: Autor

4.0.2 Funcionalidades da aplicação

O Machine Learning Toolkit permite ao usuário criar modelos de Aprendizado de Máquina utilizando seu próprio conjunto de dados. A partir do modelo criado, é possível então realizar previsões ou classificações, utilizando novos dados, também dentro da aplicação.

Para exemplificar as principais funcionalidades da aplicação, considere o conjunto de dados *Iris Flower*.

Figura 30 – Amostra de 20 observações do conjunto de dados *Iris Flower*

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.3	3.0	1.1	0.1	setosa
5.0	3.5	1.3	0.3	setosa
5.5	2.3	4.0	1.3	versicolor
5.9	3.0	4.2	1.5	versicolor
5.8	2.7	5.1	1.9	virginica
5.1	3.5	1.4	0.2	setosa
6.2	2.9	4.3	1.3	versicolor
6.7	3.3	5.7	2.1	virginica
7.6	3.0	6.6	2.1	virginica
5.5	2.4	3.7	1.0	versicolor
7.2	3.6	6.1	2.5	virginica
5.6	2.8	4.9	2.0	virginica
7.2	3.0	5.8	1.6	virginica
4.6	3.6	1.0	0.2	setosa
6.7	3.1	5.6	2.4	virginica
5.4	3.7	1.5	0.2	setosa
6.1	2.8	4.7	1.2	versicolor
6.0	2.7	5.1	1.6	versicolor
6.0	2.2	4.0	1.0	versicolor
6.0	2.9	4.5	1.5	versicolor

Fonte: Autor

Introduzido por Ronald Fisher em 1936, o conjunto de dados consiste em 50 amostras de cada uma das três espécies de Iris (Iris setosa, Iris virginica e Iris versicolor). Quatro características foram medidas a partir de cada amostra: o comprimento e a largura das sépalas e pétalas, em centímetros. Com base na combinação dessas quatro características, Fisher desenvolveu um modelo discriminante linear para distinguir as espécies umas das outras (FISHER, 1936).

Esse conjunto de dados será utilizado para exemplificar a utilização do algoritmo K-Vizinhos Mais Próximos para classificação dentro do Machine Learning Toolkit.

4.0.2.1 Criar um modelo

Os passos necessários para a criação de um modelo são:

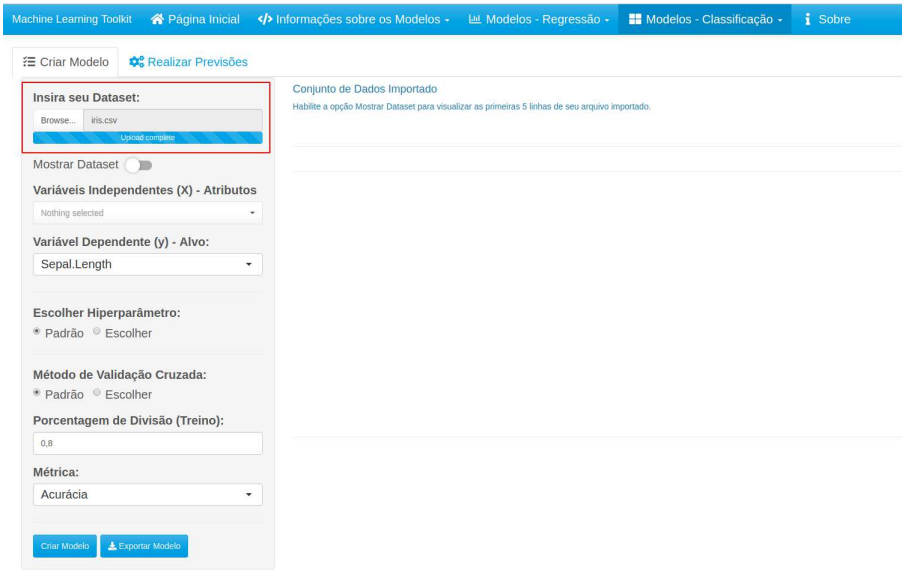
1. Selecionar o algoritmo navegando pelas opções da aplicação.

   Neste exemplo, será utilizado o algotimo KVP para classificação.

2. Importar um conjunto de dados

Para importar um conjunto de dados, basta clicar na opção *Browse* localizada na parte superior esquerda da página. O conjunto de dados necessita estar no formato Separado por Vírgula (*CSV - Comma Separated Values*), obedecendo à extensão *.csv*.

Figura 31 – Processo de criação de um modelo - importar conjunto de dados

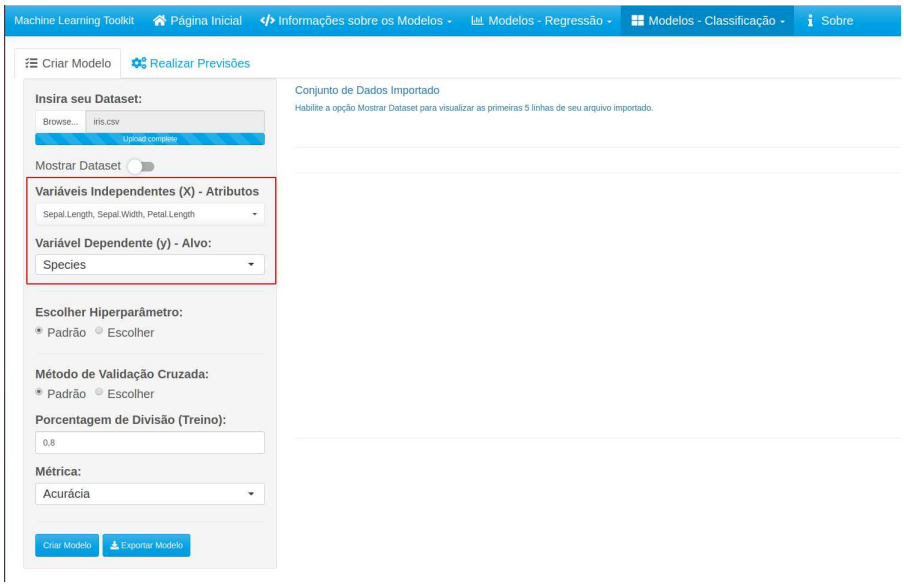


Fonte: Autor

3. Selecionar as variáveis do modelo

O usuário deve indicar quais variáveis de seu conjunto de dados serão utilizados na criação do modelo. Para isso, deve indicar quais variáveis são independentes (atributos), e qual a variável dependente (alvo).

Figura 32 – Processo de criação de um modelo - selecionar variáveis



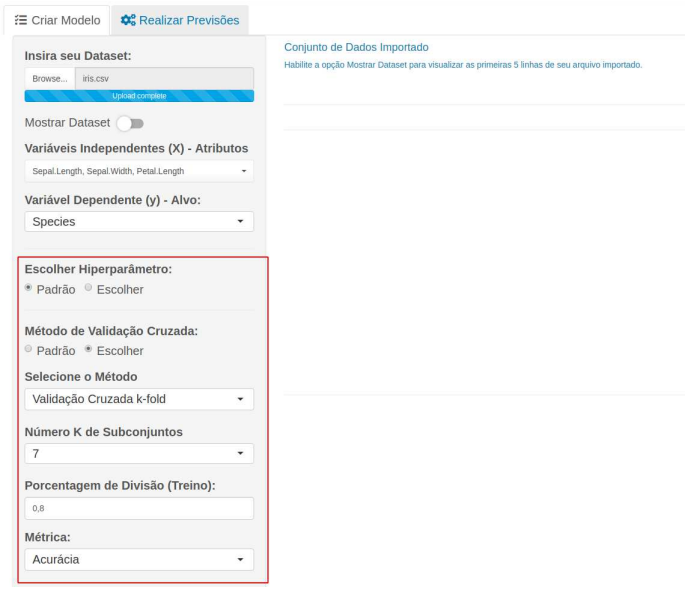
Fonte: Autor

4. Selecionar as configurações desejadas

Existem diversas configurações que podem ser modificadas pelo usuário: Hiperparâmetros, Método de Validação Cruzada, porcentagem de divisão do conjunto de dados e a métrica de avaliação.

Aqui, criaremos um modelo utilizando o método de VC *k-fold*, o qual buscaremos maximizar a acurácia.

Figura 33 – Processo de criação de um modelo - selecionar parâmetros do modelo

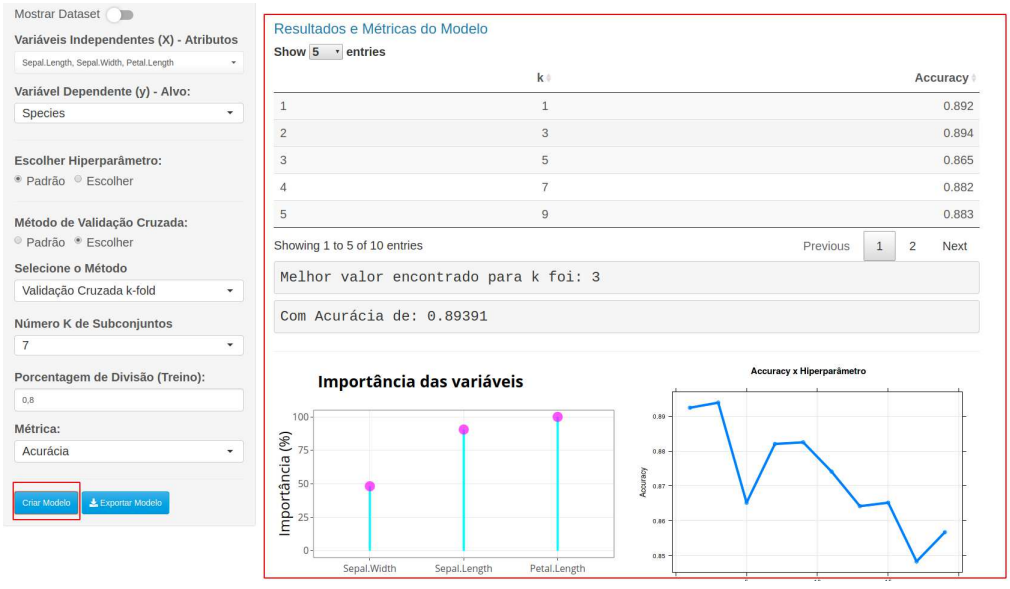


Fonte: Autor

5. Criar o modelo

   Após selecionadas as opções, basta clicar no botão **Criar Modelo**, e os resultados serão disponibilizados à direita da interface.

Figura 34 – Processo de criação de um modelo - gerar resultados



Fonte: Autor

Todos os resultados disponíveis podem ser consultados no **Apêndice A**

4.0.2.2 Exportar um modelo criado

Para que os modelos criados possam ser compartilhados com outros usuários, ou utilizados em outras oportunidades, é disponibilizada a opção de exportar o modelo. Para isso, após a criação do modelo, basta clicar no botão **Exportar Modelo**.

Figura 35 – Processo de criação de um modelo - exportar modelo criado

Fonte: Autor

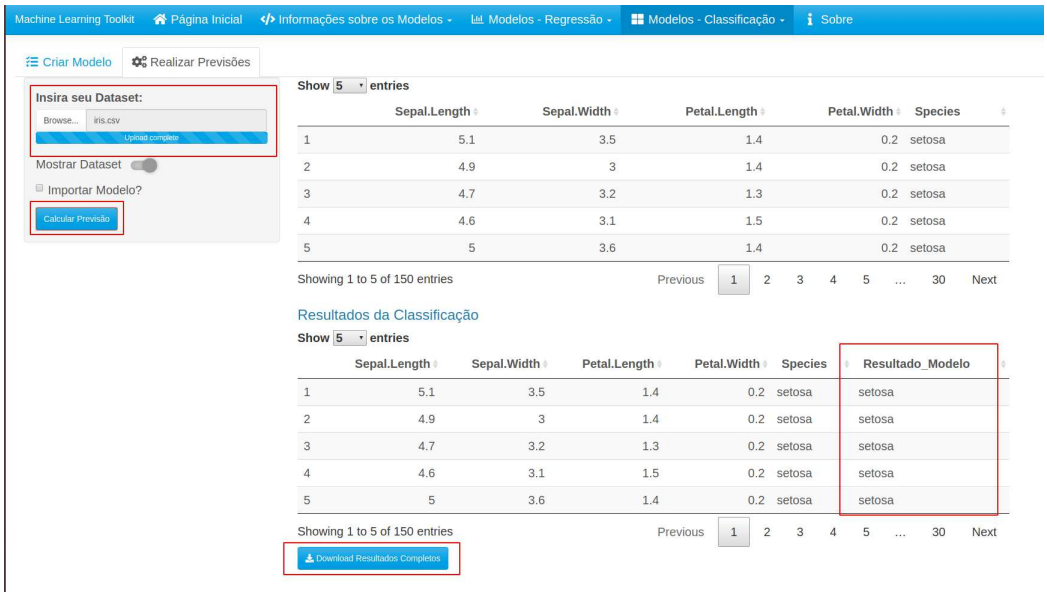
4.0.2.3 Realizar Previsões e Classificações

Após a criação do modelo, na aba **Realizar Previsões** é possível aplicar o modelo em novos dados. Para isso, existem duas opções: utilizar o modelo recém criado, ou importar um modelo criado em outras ocasiões.[1].

Para realizar previsões, basta importar um conjunto de dados com dados novos, e clicar no botão **Calcular Previsão**. Os resultados serão disponibilizados em uma nova coluna, como mostrado:

[1] Para realizar previsões utilizando um modelo importado, este deve ter sido criado utilizando os mesmos atributos presentes no novo conjunto de dados.

Figura 36 – Processo de criação de um modelo - realizar previsões

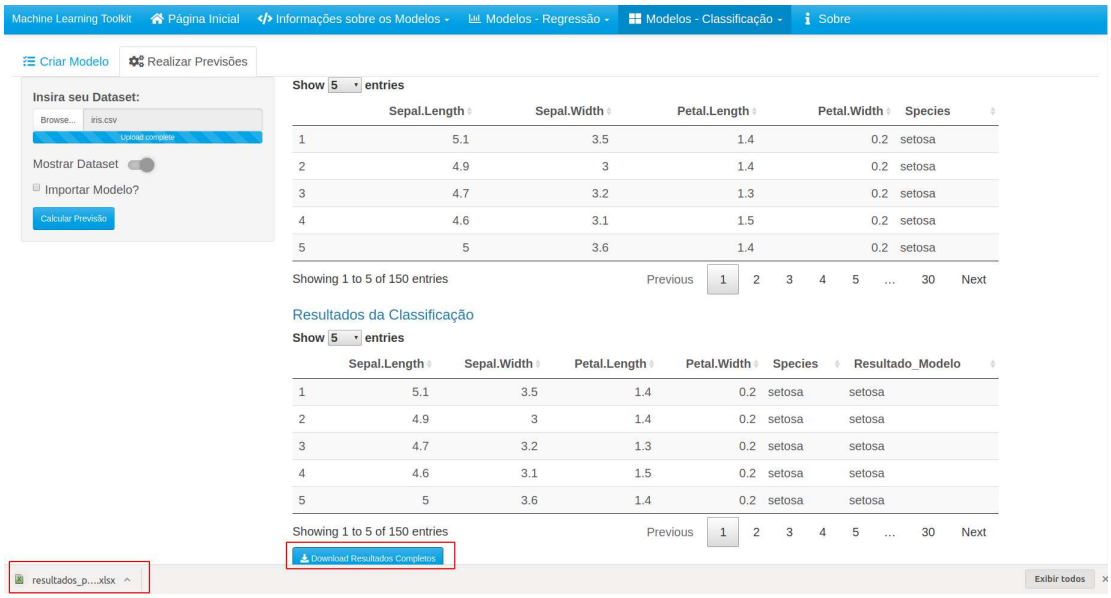


Fonte: Autor

4.0.2.4 Exportar os resultados

Após realizar previsões, é possível exportar os resultados obtidos para um arquivo, em formato de tabela, disponibilizado na extensão *.xlsx*. Para isso, basta clicar no botão **Download Resultados Completos**

Figura 37 – Processo de criação de um modelo - baixar resultados das previsões



Fonte: Autor

4.0.3 Disponibilização da aplicação

O Machine Learning Toolkit pode ser acessado de duas maneiras diferentes:

1. *Shinyapps.io*

   O *shinyapps.io* é uma plataforma que facilita o compartilhamento de aplicativos *Shiny* na *Web* de forma simplificada. O serviço é executado na nuvem em servidores compartilhados que são operados pelo RStudio. (RSTUDIO, 2019b)

   A aplicação pode ser acessada utilizando um navegador *Web* através do link:
   https://machinelearningtoolkit.shinyapps.io/mltoolkit/

   Por este trabalho utilizar a versão gratuita do servidor de hospedagem fornecido pelo *shinyapps.io*, há a limitação de 25 horas mensais de funcionamento da aplicação *online*. Para ter acesso em tempo integral ao Machine Learning Toolkit, recomenda-se utilizar a opção abaixo.

2. *Github*

   O GitHub é uma plataforma de hospedagem de código para colaboração e controle de versão. Dessa forma, permite que múltiplos usuários trabalhem juntos em projetos compartilhados.

   Todos os arquivos necessários para a execução da aplicação estão disponíveis no repositório do *Github*, e podem ser acessados em:
   https://github.com/jpcaico/Machine-Learning-Toolkit

   Neste caso, é necessária a instalação do software R, e dos pacotes utilizados na ferramenta. As instruções para esses procedimentos são indicadas no link acima.

Capítulo 5

Resultados e Discussões

5.1 Resultados - Construção da Aplicação

Através da linguagem de programação R, e os pacotes *Shiny* e *Caret*, foi possível criar uma aplicação constituída por diversas funcionalidades, e que disponibiliza diferentes opções de interação através de textos, vídeos e imagens. Utilizando a estrutura de menus, abas e sub-abas, foi possível criar uma interface que facilitasse a navegação do usuário pela ferramenta de forma rápida. Durante a realização dos testes de usabilidade, foi possível perceber que os usuários exploraram os menus da aplicação de forma intuitiva e sem maiores dificuldades.

Os conteúdos disponibilizados na aplicação englobam diversos aspectos, como instruções de uso, referenciais teóricos, e aplicações práticas dos algoritmos em si. As instruções objetivam direcionar o usuário a como utilizar a ferramenta, e estas informações são dispostas através de vídeos e textos. Notou-se, na realização dos testes de usabilidade, que muitos usuários utilizaram os recursos de vídeo da ferramenta para que conseguissem cumprir as tarefas propostas.

Os referenciais teóricos foram disponibilizados na forma de textos e imagens. Aqui, buscou-se apresentar ao usuário uma introdução teórica a qualquer termo encontrado por ele na aplicação que fosse de seu desconhecimento. Por exemplo, nas explicações a respeito dos algoritmos disponíveis na ferramenta, foram utilizados exemplos práticos para que o usuário conseguisse compreender de forma clara o funcionamento de cada técnica. Foi possível perceber, nos testes, que alguns usuários recorreram a estas explanações para que escolhessem de forma mais seletiva qual algoritmo utilizar.

Quando trata-se das aplicações práticas do algoritmos, foram utilizadas grandes quantidades de recursos que facilitassem a utilização da aplicação de forma clara e com o mínimo de complexidade possível. Dessa forma, construiu-se uma interface composta por recursos gráficos como botões, caixas e botões de seleção, menus *dropdown*, entre outros. Tais recursos permitiram que as interações na aplicação, como a criação e a utilização dos modelos de AM fossem realizadas sem que o usuário necessitasse ter conhecimento a

respeito de programação de computadores.

Os resultados obtidos através da criação dos modelos foram disponibilizados utilizando gráficos, tabelas, e poucos textos. Dessa forma, permitiu destacar informações relevantes que pudessem auxiliar o usuário na interpretação da performance dos modelos criados.

Como dificuldades encontradas na construção da ferramenta, destaca-se a necessidade de encontrar o nível de complexidade adequado das informações disponiblizadas na aplicação. Isso acontece pois, ao mesmo tempo em que é necessário que se incluam informações a respeito de parâmetros, modelos, e termos técnicos, a fim de auxiliar o usuário na utilização da ferramenta, é preciso que essas informações sejam apresentadas de tal maneira que qualquer pessoa, com conhecimento técnico ou não, possa compreendê-las e interpretá-las. Para solucionar este problema, foram disponibilizados *links* de direcionamento para referências úteis ao usuários, com níveis variados de aprofundamento sobre o tema.

5.2 Resultados - Questionário proposto pelo autor

A tabela que representa os resultados completos obtidos pelo questionário proposto pelo autor pode ser consultada no **Anexo C**.

Os critérios adotados para a interpretação dos resultados obtidos no questionário proposto pelo autor são os seguintes:

As respostas estão compreendidas em um intervalo que varia de 1 a 5. Dessa forma:

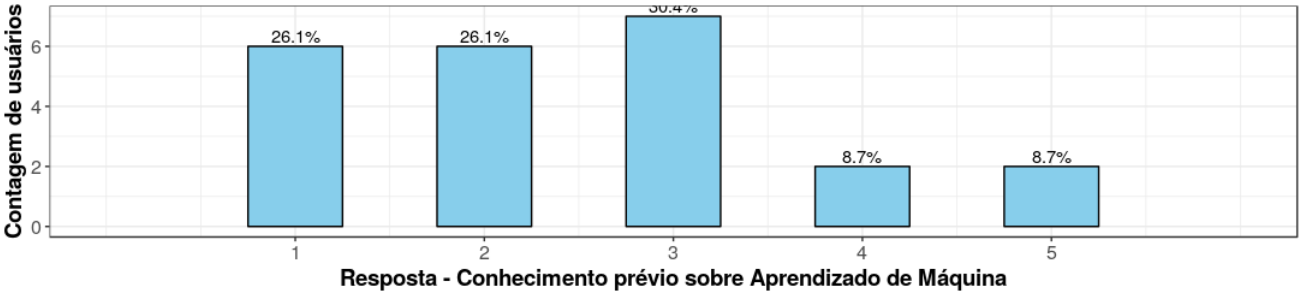
1. Respostas com notas 1 e 2: Caracterizam negação do usuário à proposição apresentada.

2. Respostas com nota 3: Caracterizam neutralidade ou incerteza do usuário à proposição apresentada.

3. Respostas com notas 4 e 5: Caracterizam afirmação do usuário à proposição apresentada.

Dessa forma, dos 23 usuários voluntários que responderam ao questionário proposto, 52.2% (12) declararam possuir conhecimento prévio nulo ou básico a respeito de AM, enquanto que 47.8% (11) declararam possuir conhecimento prévio regular ou avançado a respeito de AM, como mostrado na figura abaixo.

Figura 38 – Conhecimento prévio dos usuários sobre AM



Fonte: Autor

A partir de agora, serão feitas comparações das respostas dos questionamentos propostos, considerando dois grupos:

- **GRUPO A**: Refere-se ao grupo de todos os usuários voluntários.
- **GRUPO B**: Refere-se ao grupo de usuários que declararam possuir pouco ou nenhum conhecimento a respeito de AM.

O objetivo desta separação é analisar o desempenho da aplicação em termos gerais, considerando todos os usuários, e também analisar o desempenho dentro do grupo de pessoas que não conhecem o tema Aprendizado de Máquina.

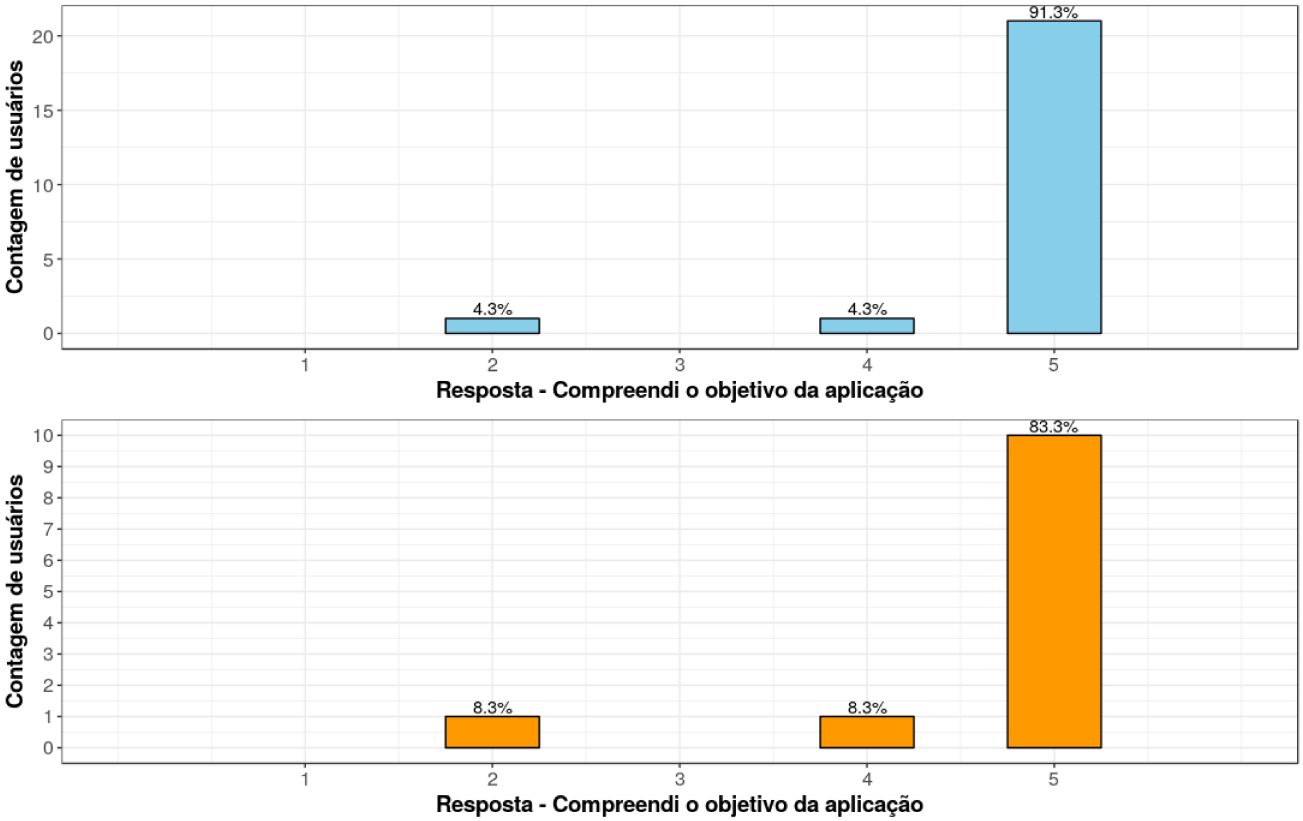
Os gráficos em cor **azul** representam dados do **GRUPO A**, enquanto que os gráficos em cor **laranja** representam dados do **GRUPO B**.

A análise dos resultados será dividida da seguinte maneira:

- Se o usuário compreendeu ou não o objetivo da aplicação;
- Se o usuário considerou ou não a aplicação intuitiva;
- Se o usuário considerou ou não a aplicação fácil de navegar;
- Se o usuário conseguiu ou não encontrar ajuda dentro da aplicação;
- Se o usuário conseguiu ou não resolver as tarefas propostas;
- Se o usuário achou ou não a apresentação dos resultados clara.

Foi possível observar que 95,6% dos usuários do **GRUPO A** conseguiram compreender o objetivo principal da aplicação. Por outro lado, 91,6% dos usuários do **GRUPO B** compreenderam o objetivo principal da aplicação. Como mostra a Figura 38 abaixo.

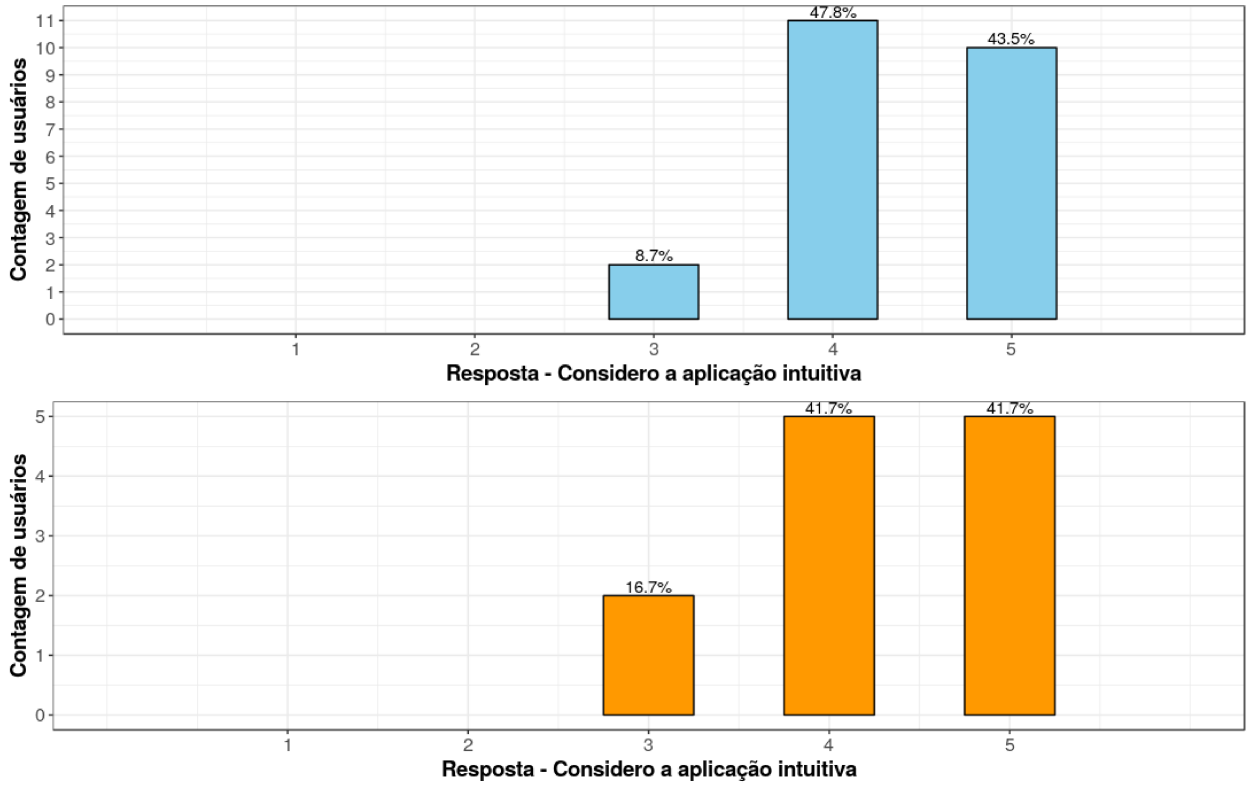
Figura 39 – Compreender o objetivo da aplicação



Fonte: Autor

A Figura 39 abaixo mostra que 91,3% dos usuários do **GRUPO A** consideraram a aplicação intuitiva. Já no **GRUPO B**, 83,4% dos usuários expressaram este mesmo posicionamento.

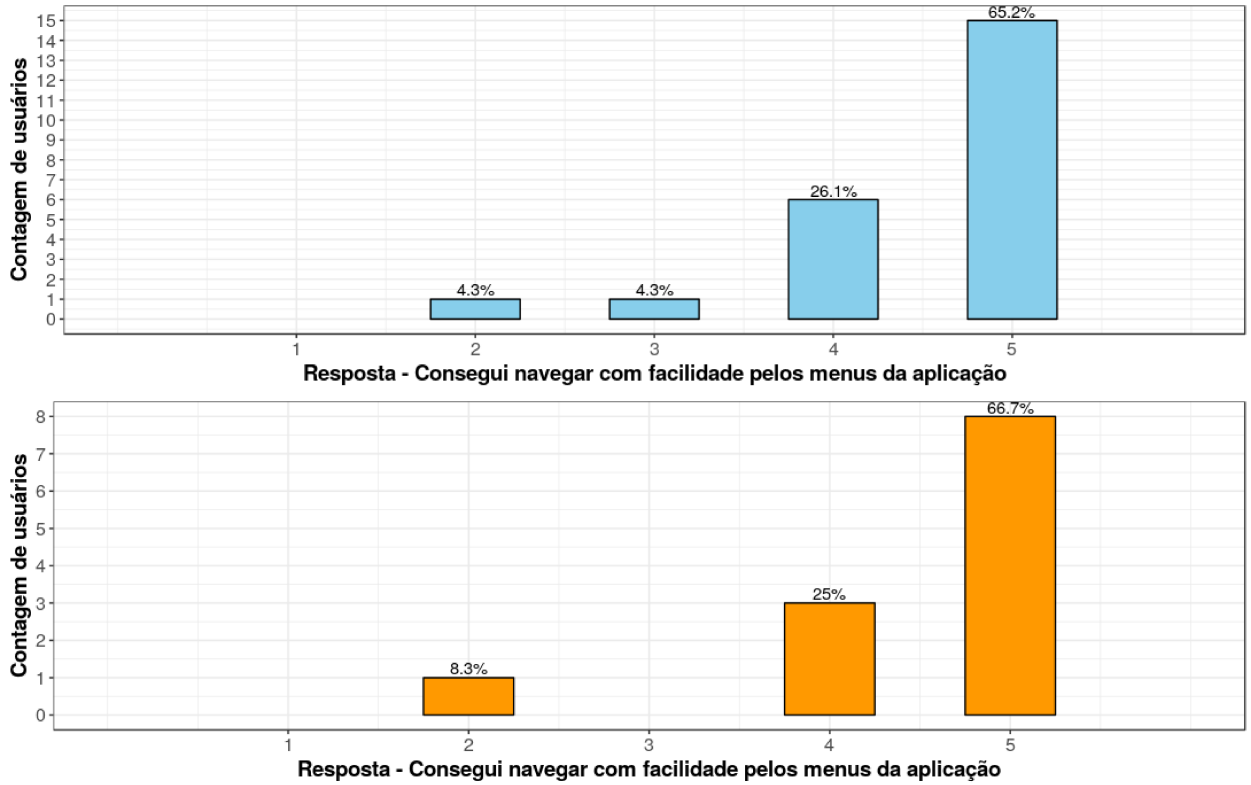
Figura 40 – Considerar a aplicação intuitiva



Fonte: Autor

No **GRUPO A**, 91,3% dos usuários expressaram conseguir navegar com facilidade pelos menus da aplicação. Enquanto isso, no **GRUPO B**, 91,7% responderam da mesma forma, como mostra a figura abaixo.

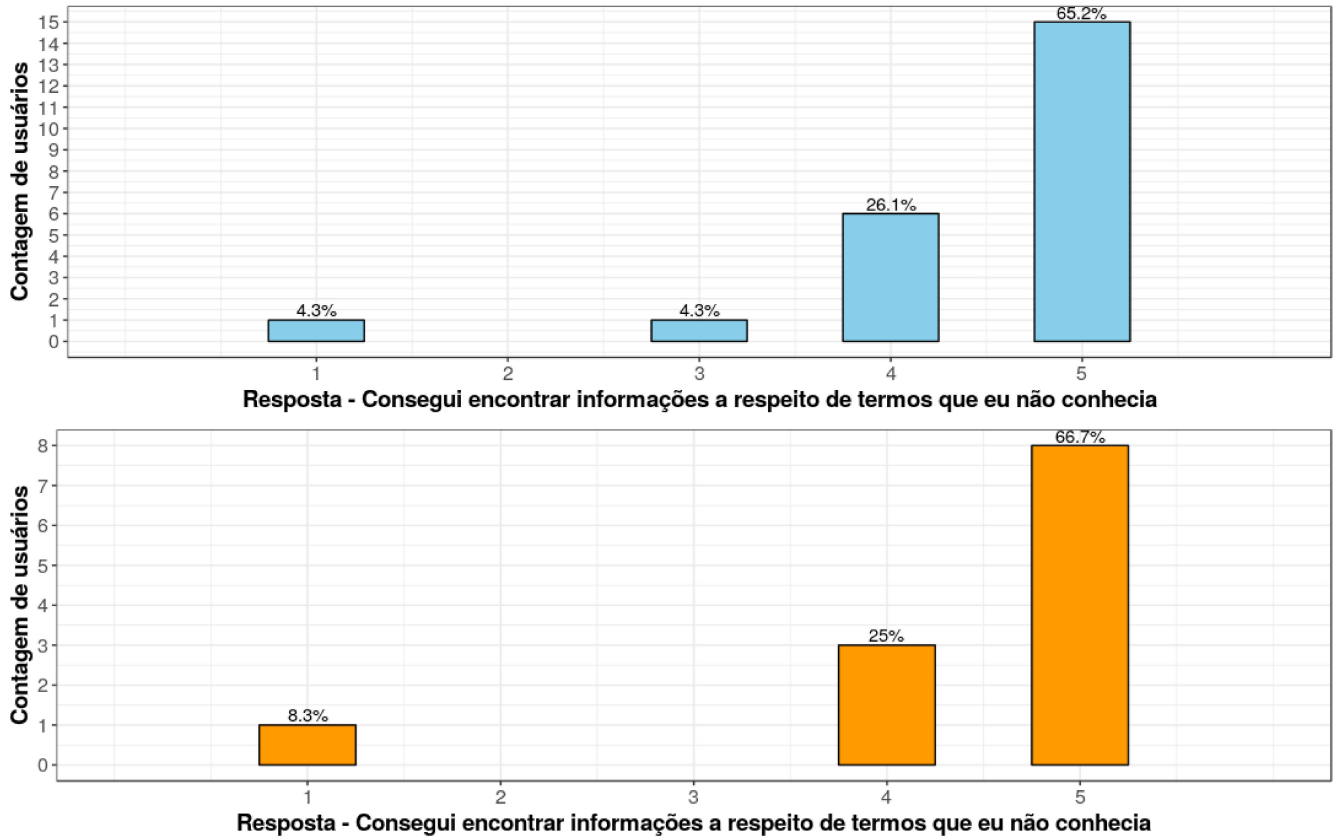
Figura 41 – Conseguir navegar com facilidade pelos menus da aplicação



Fonte: Autor

A Figura 41 expressa as respostas relacionadas à facilidade de encontrar mais informações a respeito de termos desconhecidos pelo usuário dentro da ferramenta. Assim, 91,3% dos usuários do **GRUPO A** declararam conseguir encontrar informações, enquanto que 91,7% dos usuários do **GRUPO B** também responderam da mesma maneira.

Figura 42 – Encontrar informações sobre termos desconhecidos

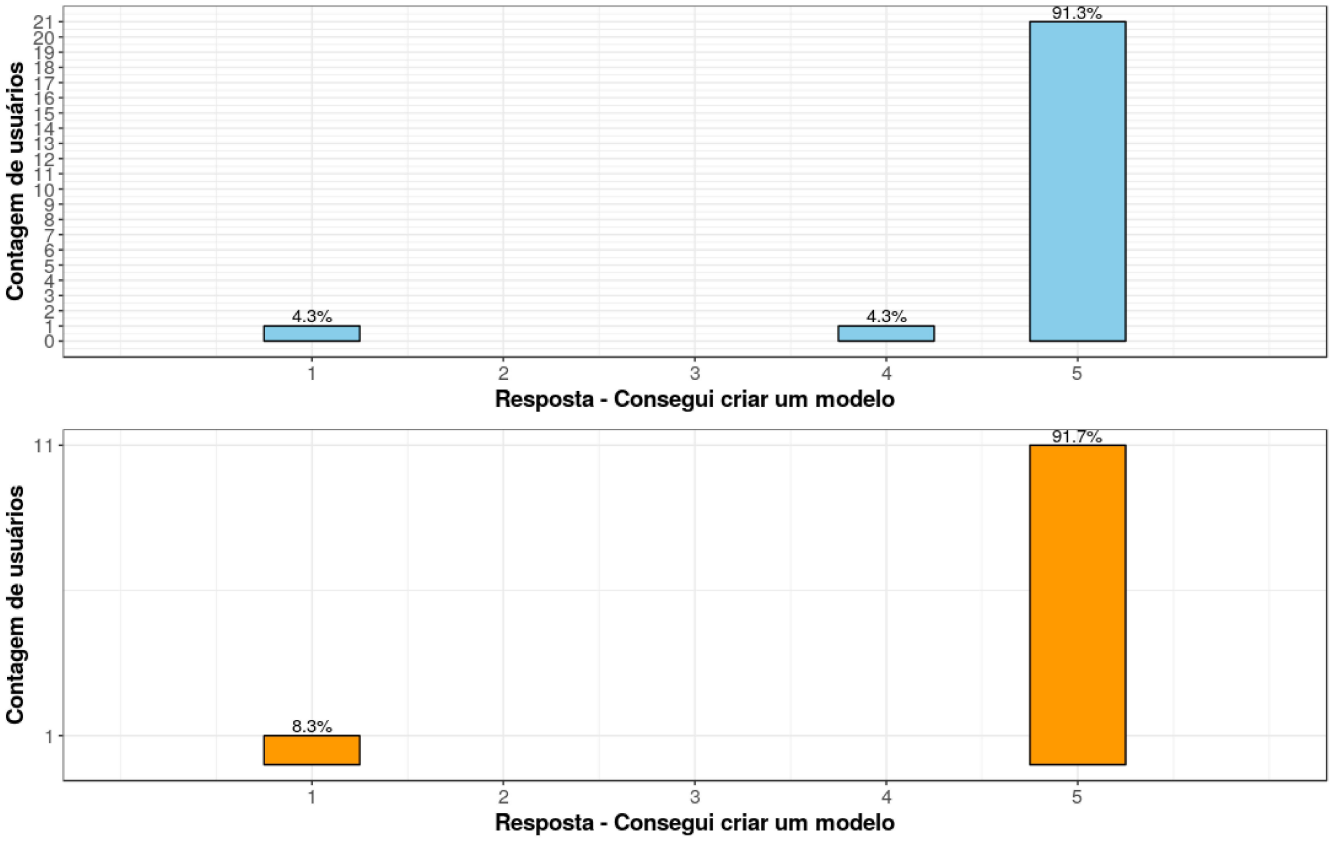


Fonte: Autor

Como apresentado no **Capítulo 3**, foram apresentadas aos usuários três tarefas que deveriam ser realizadas dentro da aplicação. São elas: criar um modelo, exportar o modelo criado, e realizar previsões ou classificações utilizando o modelo criado.

Assim, 95,6% dos usuários do **GRUPO A** expressaram conseguir criar um modelo. No **GRUPO B**, 91,7% dos usuários também conseguiram criar um modelo dentro da aplicação.

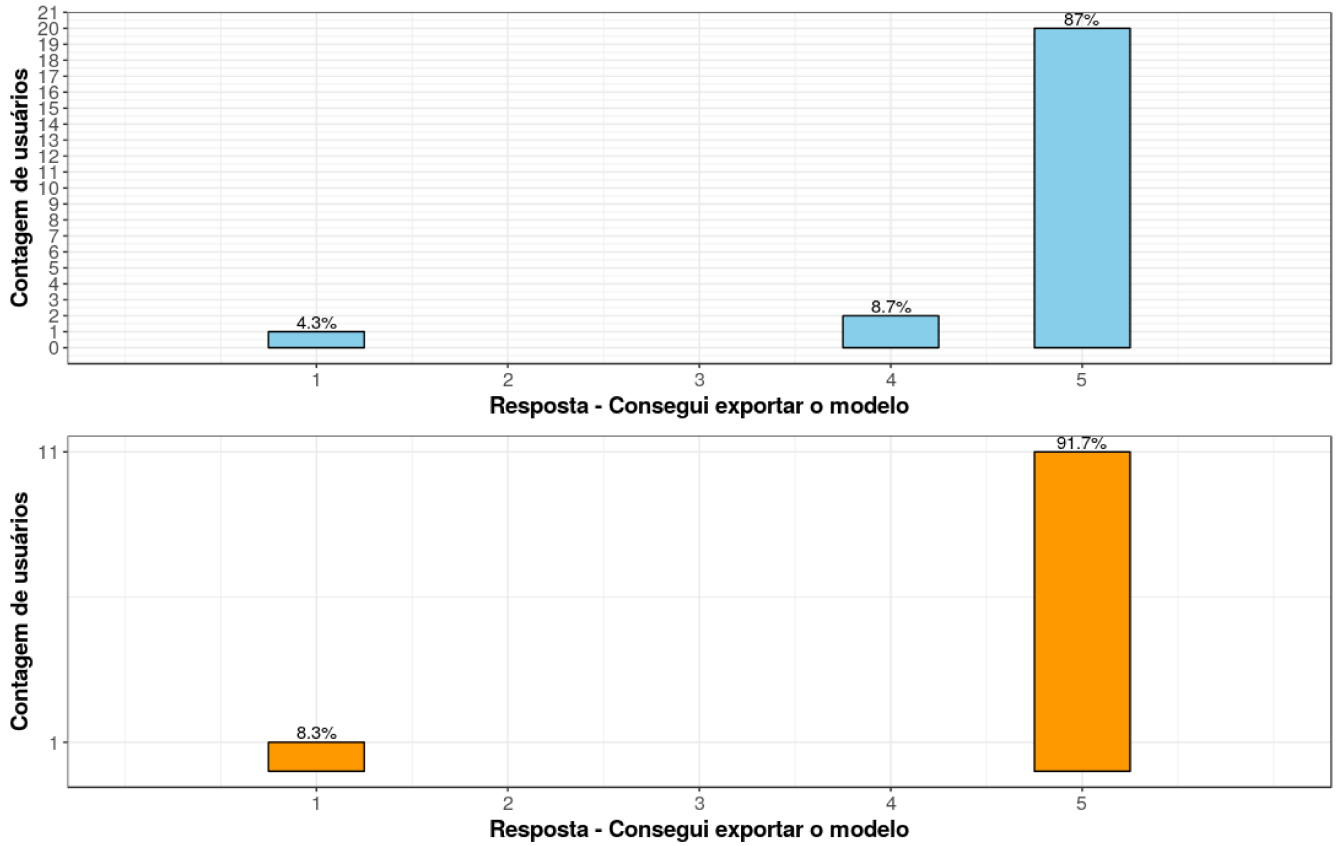
Figura 43 – Criar um modelo



Fonte: Autor

No **GRUPO A**, 95,7% dos usuários conseguiram exportar o modelo criado, enquanto que no **GRUPO B**, 91,7% dos usuários expressaram o mesmo.

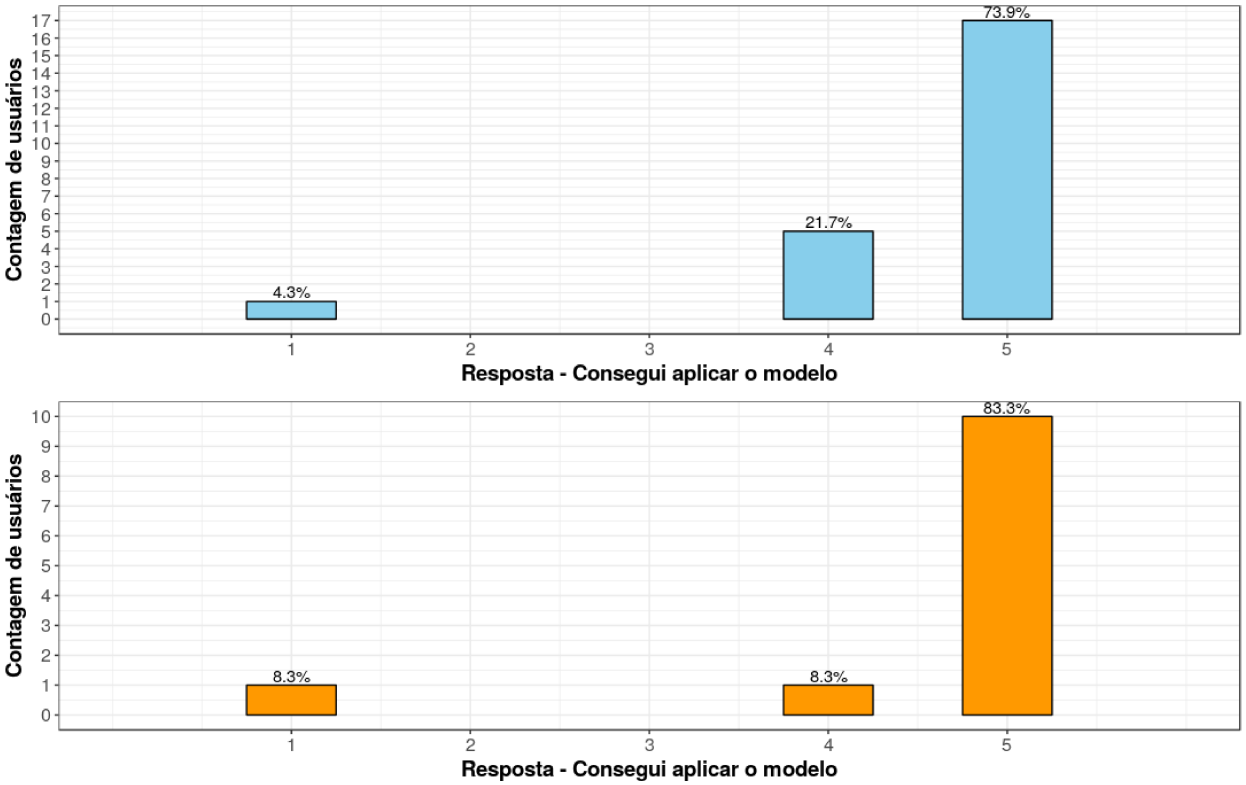
Figura 44 – Exportar o modelo criado



Fonte: Autor

A tarefa relacionada a realizar previsões ou classificações obteve as seguintes respostas: No **GRUPO A**, 95,6% dos usuários conseguiram cumprir esta tarefa, enquanto que no **GRUPO B**, 91,6% dos usuários declararam o mesmo.

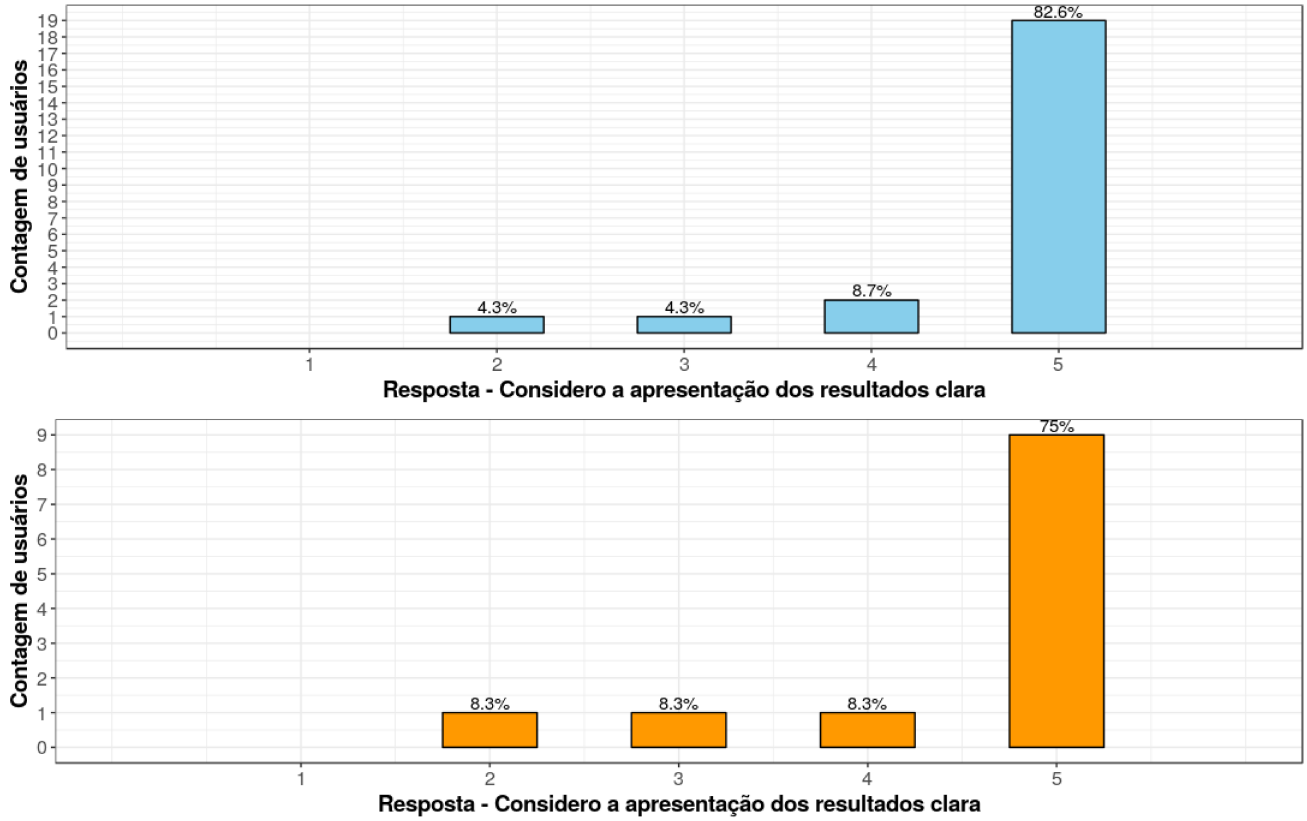
Figura 45 – Aplicação do modelo criado



Fonte: Autor

Por fim, 91.3% dos usuários do **GRUPO A** consideram a apresentação dos resultados dos modelos clara e de fácil visualização. Já no **GRUPO B**, 83,3% dos usuários concordaram com esse pressuposto.

Figura 46 – Apresentação dos Resultados



Fonte: Autor

5.3 Resultados - *System Usability Scale (SUS)*

A tabela que representa os resultados completos obtidos pelo questionário proposto pelo autor pode ser consultada no **Anexo C**.

Os resultados obtidos através do método de avaliação de usabilidade do sistema *System Usability Scale (SUS)* foram:

❏ Considerando todos os usuários:

Tabela 7 – Resultados do método SUS considerando todos os usuários

Usuário	**Conhecimento Prévio sobre AM**	**Escore SUS**	**Nota**
Usuário 1	3	87,5	A
Usuário 2	2	72,5	B
Usuário 3	3	80	B
Usuário 4	4	72,5	B
Usuário 5	4	97,5	A
Usuário 6	3	90	A
Usuário 7	3	82,5	A
Usuário 8	2	90	A
Usuário 9	2	95	A
Usuário 10	1	85	A
Usuário 11	5	77,5	B
Usuário 12	3	90	A
Usuário 13	5	77,5	B
Usuário 14	2	60	D
Usuário 15	2	90	A
Usuário 16	1	82,5	A
Usuário 17	3	82,5	A
Usuário 18	3	92,5	A
Usuário 19	1	80	B
Usuário 20	1	82,5	A
Usuário 21	1	67,5	D
Usuário 22	1	52,5	D
Usuário 23	2	92,5	A
Média		**81,739**	**A**

Pelos resultados obtidos, o sistema obteve uma pontuação média de 81.739, o que, de acordo com a escala de avaliação proposta, foi classificado como **Excelente** (Nota A) pelos usuários.

❑ Considerando apenas os usuários com pouco ou nenhum conhecimento prévio a respeito de Aprendizado de Máquina:

Tabela 8 – Resultados do método SUS considerando apenas os usuários com pouco ou nenhum conhecimento prévio sobre AM

Usuário	**Conhecimento Prévio sobre AM**	**Escore SUS**	**Nota**
Usuário 2	2	72,5	B
Usuário 8	2	90	A
Usuário 9	2	95	A
Usuário 10	1	85	A
Usuário 14	2	60	D
Usuário 15	2	90	A
Usuário 16	1	82,5	A
Usuário 19	1	80	B
Usuário 20	1	82,5	A
Usuário 21	1	67,5	D
Usuário 22	1	52,5	D
Usuário 23	2	92,5	A
Média Final		**79,166**	**B**

De acordo com os resultados obtidos, os usuários que declararam possuir pouco ou nenhum conhecimento prévio sobre o tema AM classificaram o sistema como **Bom** (Nota B), atribuindo uma pontuação média de 79.166.

De acordo com os resultados obtidos, tanto através do questionário proposto pelo autor, quanto através do método de avaliação de usabilidade do sistema *System Usability Scale*, foi possível perceber que a aplicação Machine Learning Toolkit cumpriu com os objetivos em sua totalidade.

Baseando-se nas respostas dos usuários voluntários, a aplicação é, de fato, de fácil utilização. É importante frisar também que os usuários do **GRUPO B**, ou seja, os quais possuiam pouco ou nenhum conhecimento prévio a respeito de *AM* conseguiram, em sua maioria, realizar as tarefas propostas, interagindo com o sistema para promover a criação e utilização dos modelos intuitivamente.

Capítulo 6

Conclusão

Com o constante avanço da tecnologia em diversas áreas do conhecimento, como agricultura, engenharia, educação e saúde, e o consequente aumento na geração e coleta de dados de diferentes naturezas, criaram-se grandes oportunidades para que o Aprendizado de Máquina pudesse auxiliar na resolução de problemas complexos, além de atuar na otimização de diversos processos em todos estes campos distintos. Dessa forma, a tendência é que o AM esteja cada vez mais presente no cotidiano das pessoas, e que se caminhe para um futuro o qual os produtos e tecnologias estejam cada vez mais integrados e automatizados.

Apesar do avanço tecnológico e da maior presença do AM na indústria, ambiente acadêmico, e na vida cotidiana da população, nem sempre existem pessoas com conhecimento necessário para aplicar tais metodologias e extrair delas seu maior potencial. Assim, o Machine Learning Toolkit se apresenta como uma proposta de amenização deste problema.

As etapas de construção da aplicação, em conjunto, contribuiram para o cumprimento de todos os objetivos propostos neste trabalho. A linguagem R, juntamente com os diversos recursos do pacote *Shiny*, se mostrou eficiente em construir uma aplicação *Web* intuitiva e de funcionalidade simples, e que permitiu a inserção, na aplicação, de recursos visuais gráficos, de texto e vídeo, os quais foram fundamentais para a boa utilização da ferramenta pelos usuários.

O pacote *Caret* também se mostrou eficaz em seu propósito. Por disponiblizar recursos de pré-processamento e treinamento e validação de diversos algoritmos de AM, contribuiu para que todas as funcionalidades da aplicação estivessem bem integradas e padronizadas de forma coerente.

O *shinyapps.io* e o *Github* possibilitaram o acesso da aplicação de forma gratuita a qualquer usuário. Dessa forma, promovem a democratização e a disseminação do conhecimento a respeito do tema proposto.

Os questionários e métodos de avaliação da aplicação propostos foram de suma importância para a validação dos objetivos inicialmente apresentados. Neste contexto, vale

destacar o principal deles: permitir que pessoas sem conhecimento prévio a respeito de Aprendizado de Máquina possam aplicar a tecnologia em seus próprios problemas e gerar soluções para problemas específicos em diversas áreas do conhecimento.

Dessa forma, a aplicação apresentada em sua primeira versão cumpre com os objetivos iniciais propostos. Além disso, busca-se para trabalhos futuros e versões futuras da aplicação, a implementação de uma série de melhorias como a adição de algoritmos de aprendizado não supervisionado, a disponibilização de resultados mais detalhados, inserção de novos vídeos explicativos, recursos de comparação de desempenho entre os algoritmos, entre outros.

Referências

AKSOY, S.; HARALICK, R. M. Feature normalization and likelihood-based similarity measures for image retrieval. **Pattern Recognition Letters**, v. 22, n. 5, p. 563 – 582, 2001. ISSN 0167-8655. Image/Video Indexing and Retrieval. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167865500001124>.

ALTMAN, N. An introduction to kernel and nearest-neighbor nonparametric regression. **American Statistician - AMER STATIST**, v. 46, p. 175–185, 08 1992.

AMIT, Y.; GEMAN, D. Shape quantization and recognition with randomized trees. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 9, n. 7, p. 1545–1588, out. 1997. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/neco.1997.9.7.1545>.

ANALYTICS VIDHYA. **Six Steps to Learn Naive Bayes Algorithm**. 2017. Disponível em: <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>. Acesso em: 11 jun. 2019.

ARMSTRONG, J. S. et al. Error measures for generalizing about forecasting methods: Empirical comparisons. **International Journal of Forecasting**, p. 69–80, 1992.

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **J. Usability Stud.**, v. 4, p. 114–123, 04 2009.

BARR, J. R. Machine learning, A tutorial with R. In: **First International Conference on Artificial Intelligence for Industries, AI4I 2018, Laguna Hills, CA, USA, September 26-28, 2018**. [s.n.], 2018. p. 120–121. Disponível em: <https://doi.org/10.1109/AI4I.2018.8665676>.

BASYIR, M. et al. Evaluating the quality of emergency reporting mobile application on usage service decision. **IOP Conference Series: Materials Science and Engineering**, v. 536, p. 012147, 06 2019.

BISHOP, Y. M. et al. Discrete multivariate analysis: Theory and practice. **Applied Psychological Measurement**, v. 1, 02 1977.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: **Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory**. [S.l.]: ACM Press, 1992. p. 144–152.

BOWNLEE J. **A Gentle Introduction to the Bootstrap Method**. 2018. Disponível em: <https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/>. Acesso em: 7 jun. 2019.

BREIMAN, L. et al. **Classification and Regression Trees**. Monterey, CA: Wadsworth and Brooks, 1984.

BROOKE, J. **SUS: A quick and dirty usability scale**. 1996.

CAMERON, A. C.; WINDMEIJER, F. A. An r-squared measure of goodness of fit for some common nonlinear regression models. **Journal of Econometrics**, v. 77, n. 2, p. 329 – 342, 1997. ISSN 0304-4076. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0304407696018180>.

CARET OFFICIAL DOCUMENTATION. **The caret Package**. 2019. Disponível em: <http://topepo.github.io/caret/>. Acesso em: 15 jun. 2019.

COOMANS, D.; MASSART, D. Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules. **Analytica Chimica Acta**, v. 136, p. 15 – 27, 1982. ISSN 0003-2670. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0003267001953590>.

CORTES, C.; VAPNIK, V. Support-vector networks. In: **Machine Learning**. [S.l.: s.n.], 1995. p. 273–297.

DATAROBOT. **Data Collection | Data Robot Artificial Intelligence Wiki**. https://www.datarobot.com/wiki/data-collection/, 2019.

DIEZ-OLIVAN, A. et al. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0. **Information Fusion**, v. 50, p. 92 – 111, 2019. ISSN 1566-2535. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1566253518304706>.

DUDA, R. O.; HART, P. E. **Pattern Classification and Scene Analysis**. New Yotk: John Willey & Sons, 1973.

EFRON, B.; TIBSHIRANI, R. Improvements on cross-validation: The .632+ bootstrap method. **Journal of the American Statistical Association**, [American Statistical Association, Taylor Francis, Ltd.], v. 92, n. 438, p. 548–560, 1997. ISSN 01621459. Disponível em: <http://www.jstor.org/stable/2965703>.

FAMILI, A. et al. Data preprocessing and intelligent data analysis. **Intelligent Data Analysis**, v. 1, n. 1, p. 3 – 23, 1997. ISSN 1088-467X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1088467X98000079>.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals of Eugenics**, v. 7, n. 2, p. 179–188, 1936. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x>.

GUNUGANTI, A. Application development framework for r/shiny. **PharmaSUG 2018 Conference**, 2018.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**. New York, NY, USA: Springer New York Inc., 2001. (Springer Series in Statistics).

HO, T. K. Random decision forests. In: **Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1**. [s.n.], 1995. (ICDAR '95), p. 278–. ISBN 0-8186-7128-9. Disponível em: <http://dl.acm.org/citation.cfm?id=844379.844681>.

IEEE SPECTRUM. **The 2018 Top Programming Languages**. 2018. Disponível em: <https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>. Acesso em: 15 jun. 2019.

IHAKA, R.; GENTLEMAN, R. R: A language for data analysis and graphics. **Journal of Computational and Graphical Statistics**, v. 5, p. 299–314, 09 1996.

INDEED. **The Best Jobs in the U.S.: 2019**. 2019. Disponível em: <http://blog.indeed.com/2019/03/14/best-jobs-2019/>. Acesso em: 22 jun. 2019.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. v. 14, 03 2001.

KUMAR, A. Pre-processing and modelling using caret package in r. In: . [S.l.: s.n.], 2018.

MASON, L. et al. Boosting algorithms as gradient descent. In: **Proceedings of the 12th International Conference on Neural Information Processing Systems**. Cambridge, MA, USA: MIT Press, 1999. (NIPS'99), p. 512–518. Disponível em: <http://dl.acm.org/citation.cfm?id=3009657.3009730>.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

_____. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

OBERMEYER, Z.; EMANUEL, E. J. Predicting the future — big data, machine learning, and clinical medicine. **The New England journal of medicine**, PubMed Central, v. 375(13), p. 1216–1219, 2016.

PEREIRA, F.; NORVIG, P.; HALEVY, A. The unreasonable effectiveness of data. **IEEE Intelligent Systems**, IEEE Computer Society, Los Alamitos, CA, USA, v. 24, n. 02, p. 8–12, mar 2009. ISSN 1541-1672.

PICARD, R. R.; COOK, R. D. Cross-validation of regression models. **Journal of the American Statistical Association**, [American Statistical Association, Taylor Francis, Ltd.], v. 79, n. 387, p. 575–583, 1984. ISSN 01621459.

QIN, S. J.; CHIANG, L. H. Advances and opportunities in machine learning for process data analytics. **Computers Chemical Engineering**, v. 126, p. 465 – 473, 2019. ISSN 0098-1354. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0098135419302248>.

QUINLAN, J. R. Induction of decision trees. **Machine Learning**, v. 1, n. 1, p. 81–106, Mar 1986. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00116251>.

R PROJECT. **The R Project for Statistical Computing**. 2019. Disponível em: <https://www.r-project.org/about.html>. Acesso em: 15 jun. 2019.

RASCHKA S. **About Feature Scaling and Normalization**. 2014. Disponível em: <https://sebastianraschka.com/Articles/2014_about_feature_scaling.html>. Acesso em: 6 jun. 2019.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. **Encyclopedia of Database Systems**, v. 532–538, p. 532–538, 01 2009.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, p. 65–386, 1958.

_____. **Principles of Neurodynamics**. [S.l.]: Spartan Books, 1959.

RSTUDIO. **Shiny From RStudio**. 2019. Disponível em: <https://shiny.rstudio.com/>. Acesso em: 15 jun. 2019.

_____. **shinyapps.io user guide**. https://docs.rstudio.com/shinyapps.io/, 2019.

RUSSELL, S. Chapter 4 - machine learning. In: BODEN, M. A. (Ed.). **Artificial Intelligence**. San Diego: Academic Press, 1996, (Handbook of Perception and Cognition). p. 89 – 133. ISBN 978-0-12-161964-0. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780121619640500066>.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594.

_____. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, p. 210–229, 1959.

SAURO, J. **5 Ways to Interpret a SUS Score**. 2018. <https://measuringu.com/interpret-sus-score/>. Accessed: 2010-09-30.

SEAL, H. L. Studies in the history of probability and statistics. xv: The historical development of the gauss linear model. **Biometrika**, [Oxford University Press, Biometrika Trust], v. 54, n. 1/2, p. 1–24, 1967. ISSN 00063444. Disponível em: <http://www.jstor.org/stable/2333849>.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding Machine Learning: From Theory to Algorithms**. New York, NY, USA: Cambridge University Press, 2014. ISBN 1107057132, 9781107057135.

STEHMAN, S. Selecting and interpreting measures of thematic classification accuracy. **Remote Sensing of Environment**, v. 62, p. 77–89, 10 1997.

TANG, W.; KASSIM, A.; ABUBAKAR, S. Comparative studies of various missing data treatment methods - malaysian experience. **Atmospheric Research**, v. 42, n. 1, p. 247 – 262, 1996. ISSN 0169-8095. Closing the gap between theory and practice in urban rainfall applications. Disponível em: <http://www.sciencedirect.com/science/article/pii/0169809595000674>.

WERBOS, P.; JOHN, P. J. P. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. 01 1974.

ZHANG, H. The optimality of naive bayes. In: . [S.l.: s.n.], 2004. v. 2.

ZHENG, A. **Evaluating Machine Learning Models**. [S.l.]: O'Reilly, 2015. 37 p. Bibliografia: p. 8. ISBN 978-1-491-93246-9.

Anexos

ANEXO **A**

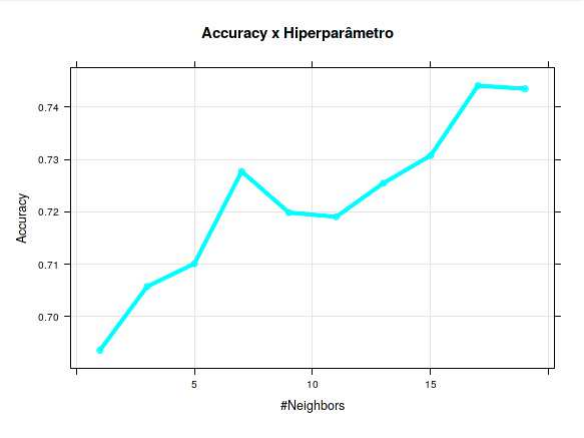
Resultados Disponibilizados pelo Machine Learning Toolkit

A.1 Gráficos

❑ Métrica x Hiperparâmetro:

Gráfico disponível quando a opção **Padrão** de Parâmetros do Modelo for selecionada. Mostra a relação entre a métrica obtida e os hiperparâmetros selecionados.

Figura 47 – Exemplo do gráfico Métrica x Hiperparâmetro

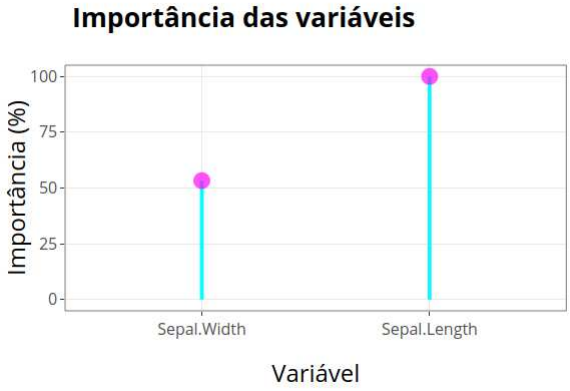


Fonte: Autor

❑ Importância das Variáveis:

Este gráfico mostra a importância que cada variável independente teve no modelo. Ou seja, caso o modelo possua muitas variáveis independentes com baixa importância nos resultados, convém, geralmente, retirá-las do modelo e retreiná-lo utilizando apenas variáveis de alta importância.

Figura 48 – Exemplo do gráfico Importância das Variáveis

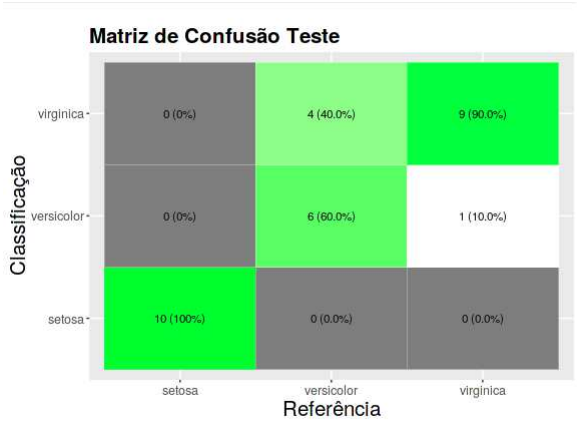


Fonte: Autor

❏ Matriz de Confusão - Teste:

Mostra a relação entre os valores reais e os valores preditos, classe a classe, tendo em um eixo a referência, e no outro o resultado. A diagonal da matriz, contém os valores acertados pelo modelo.
Disponível em algoritmos de classificação.

Figura 49 – Exemplo do gráfico Matriz de Confusão



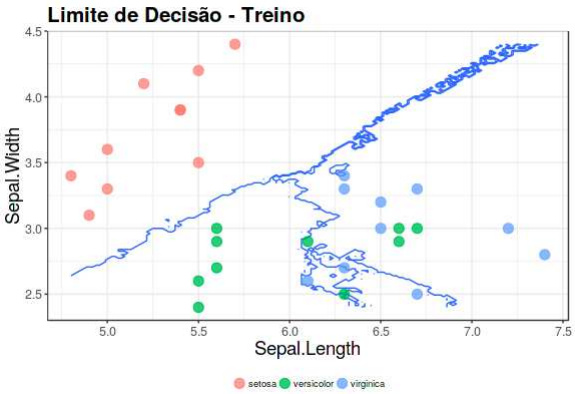
Fonte: Autor

❏ Limite de Decisão:

Gráfico disponível quando selecionadas apenas duas variáveis independentes. Mostra, o limite de decisão que o algoritmo criou para classificar os alvos escolhidos. Além disso, os pontos observados referem-se aos pontos preditos do conjunto de teste. Desta forma, convém analisar, em conjunto, a Matriz de Confusão, a Região

de Decisão e o Limite de Decisão para interpretar os resultados.
Disponível em algoritmos de classificação.

Figura 50 – Exemplo do gráfico Limite de Decisão

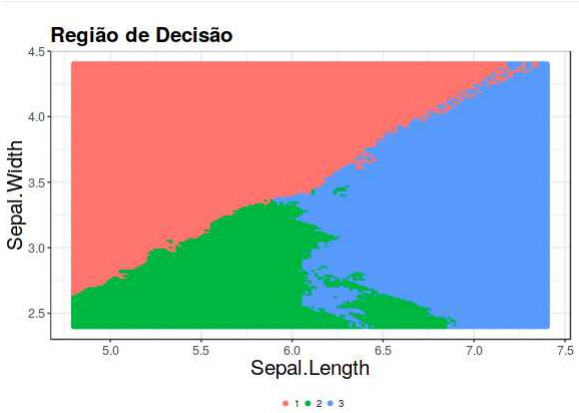


Fonte: Autor

❏ Região de Decisão:

Gráfico disponível quando selecionadas apenas duas variáveis independentes. Mostra, no treino, qual foi a área de decisão do algoritmo selecionado para classificar os alvos escolhidos. Este gráfico é importante para observar o comportamento do modelo e compará-lo com os demais (modelos lineares vs modelos não lineares por exemplo)
Disponível em algoritmos de classificação.

Figura 51 – Exemplo do gráfico Região de Decisão

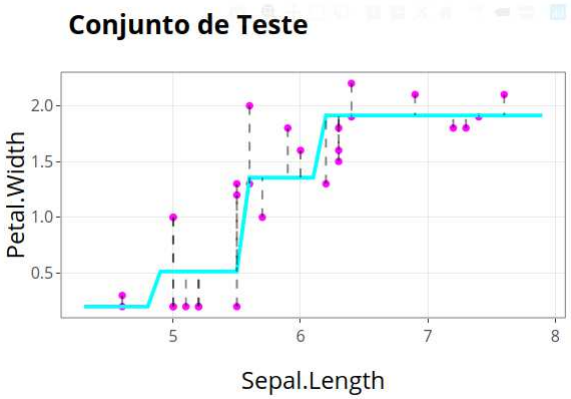


Fonte: Autor

❏ Resultados do Conjunto de Teste:

Este gráfico está disponível quando seleciona apenas uma variável independente. Ele mostra como o modelo gerado se ajustou quando aplicado aos dados do conjunto de teste (ainda não conhecidos pelo modelo). Os pontos na cor magenta referem-se aos pontos do conjunto de teste, enquanto que a linha ou curva em azul refere-se ao resultado obtido pelo modelo.
Disponível em algoritmos de regressão.

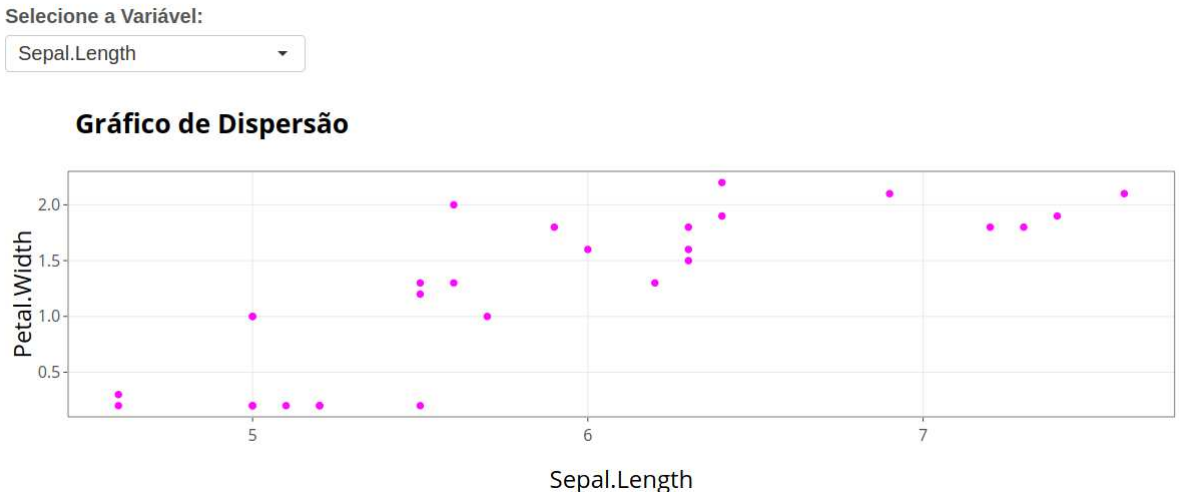
Figura 52 – Exemplo do gráfico Resultados do Conjunto de Testes



Fonte: Autor

- Relação das Variáveis: Este gráfico mostra a relação entre as variáveis independentes do modelo e a variável dependente, através de um gráfico de dispersão. Convém analisar em conjunto o gráfico de Importância das Variáveis e Relação das Variáveis para interpretar melhor os resultados obtidos.
  Disponível em algoritmos de regressão.

Figura 53 – Exemplo do gráfico Relação das Variáveis



Fonte: Autor

A.2 Tabelas

Executando os modelos de regressão na aplicação, é produzida a tabela contendo os valores de R^2 e o Erro Quadrático Médio para cada algoritmo de aprendizado de máquina avaliado.

Figura 54 – Exemplo da Tabela Resultado para um algoritmo de regressão

Show 5 entries

	maxdepth	RMSE	Rsquared
1	5	0.074	0.719
2	10	0.073	0.723
3	15	0.073	0.723
4	20	0.073	0.723

Showing 1 to 4 of 4 entries Previous 1 Next

```
Melhor valor encontrado para maxdepth foi: 10
```

```
Com Raíz do Erro Quadrático Médio de: 0.07348
```

Fonte: Autor

Executando os modelos da aplicação, é produzida a tabela contendo os valores de Acurácia para cada algoritmo de aprendizado de máquina avaliado.

Figura 55 – Exemplo da Tabela Resultado para um algoritmo de classificação

Show 5 entries

	k	Accuracy
1	1	0.892
2	3	0.894
3	5	0.865
4	7	0.882
5	9	0.883

Showing 1 to 5 of 10 entries Previous 1 2 Next

```
Melhor valor encontrado para k foi: 3
```

```
Com Acurácia de: 0.89391
```

Fonte: Autor

ANEXO **B**

Modelos Utilizados no pacote *Caret*

Os modelos disponíveis nesta aplicação foram criados utilizando o pacote *Caret*. A documentação oficial pode ser consultada em: http://topepo.github.io/caret/index.html

Foram utilizados os modelos[1] a seguir:

- ❏ Regressão Linear: *lm*
- ❏ Árvore de Decisão: *rpart2*
- ❏ Floresta Aletória (Random Forest): *rf*
- ❏ Máquina Vetor de Suporte: *svmLinear, svmPoly, svmRadial*
- ❏ K-Vizinhos Mais Próximos: *knn*
- ❏ Naive Bayes: *nb*
- ❏ Perceptron Multi Camadas: *neuralnet*

[1] Para maiores informações, consultar *https://topepo.github.io/caret/available-models.html*

ANEXO C

Questionário de avaliação do Sistema

C.1 Questionário proposto pelo autor

1. Informações Preliminares

   ❏ Escolaridade:

   - Fundamental Incompleto
   - Fundamental Completo
   - Médio Incompleto
   - Médio Completo
   - Superior Incompleto
   - Superior Completo

   ❏ Curso (Se Aplicável): _____.

   ❏ Área de interesse: _____.

2. Análise de Perfil do Usuário

   As respostam variam de 1 a 5, com 1 indicando a opção "Discordo Fortemente"e 5 indicando a opção "Concordo Fortemente".

   ❏ P01 - Eu tenho conhecimento prévio sobre programação.

   ❏ P02 - Eu tenho conhecimento prévio sobre o tema Aprendizado de Máquina.

   ❏ P03 - Eu tenho interesse sobre o tema Aprendizado de Máquina.

3. Análise da Experiência do Usuário

   As respostam variam de 1 a 5, com 1 indicando a opção "Discordo Fortemente"e 5 indicando a opção "Concordo Fortemente".

   ❏ P04 - O conteúdo da aplicação é relevante para mim.

- ❏ P05 - Eu considero a utilização aplicação intuitiva.
- ❏ P06 - Eu compreendi o objetivo da aplicação.
- ❏ P07 - Eu consegui navegar com facilidade pelos menus da aplicação.
- ❏ P08 - Eu consegui criar um modelo.
- ❏ P09 - Eu consegui aplicar o modelo depois de criá-lo.
- ❏ P10 - Eu consegui exportar o modelo criado.
- ❏ P11 - Eu consegui encontrar informações na aplicação a respeito de termos que eu não conhecia.
- ❏ P12 - Eu considero que a apresentação dos resultados dos modelos foi clara e de fácil visualização.
- ❏ Comentários/Sugestões : _____.

C.2 Resultados obtidos pelo questionário proposto pelo autor

Tabela 9 – Resultados obtidos pelo questionário proposto pelo autor - Pontuações

Nome	P01	P02	P03	P04	P05	P06	P07	P8	P9	P10	P11	P12
Usuário 1	5	3	4	5	5	5	4	5	5	5	4	5
Usuário 2	3	2	3	3	5	5	4	5	5	5	4	3
Usuário 3	3	3	4	3	4	5	5	5	5	5	5	5
Usuário 4	5	4	5	5	5	5	5	5	4	5	5	5
Usuário 5	5	4	5	5	5	5	5	5	5	5	5	5
Usuário 6	5	3	5	5	4	5	4	5	5	5	5	5
Usuário 7	4	3	3	3	4	5	5	5	5	5	4	5
Usuário 8	4	2	4	4	4	5	5	5	5	5	5	5
Usuário 9	4	2	3	4	5	5	5	5	5	5	5	5
Usuário 10	2	1	5	5	4	5	4	5	5	5	4	5
Usuário 11	5	5	5	3	4	5	5	5	5	5	5	5
Usuário 12	3	3	4	5	5	5	5	5	4	5	4	5
Usuário 13	5	5	5	5	5	5	5	4	4	4	5	5
Usuário 14	2	2	3	3	3	4	5	5	5	5	5	5
Usuário 15	2	2	4	5	5	5	5	5	4	5	5	5
Usuário 16	1	1	5	5	5	5	5	5	5	5	5	5
Usuário 17	2	3	4	4	4	5	3	5	4	4	3	5
Usuário 18	3	3	4	4	4	5	4	5	5	5	5	4
Usuário 19	1	1	1	3	3	5	4	5	5	5	5	4
Usuário 20	3	1	3	3	4	5	5	5	5	5	5	5
Usuário 21	3	1	3	3	4	5	5	5	5	5	4	5
Usuário 22	1	1	5	5	5	5	2	1	1	1	5	2
Usuário 23	2	2	5	4	4	2	5	5	5	5	1	5

Tabela 10 – Resultados obtidos pelo questionário proposto pelo autor - Áreas de Interesse

Nome	Áreas de Interesse
Usuário 1	Controle e Automação
Usuário 2	Automação
Usuário 3	Estruturas e Ciência de dados
Usuário 4	Tecnologias Web e Mobile
Usuário 5	RPA / Artificial Intelligence / Machine Learning
Usuário 6	Engenharia da Computação, Biomédica, Inteligência Artificial, Computação Gráfica
Usuário 7	Certificação e Homologação, Termoflúidos, Pogramação
Usuário 8	Automação
Usuário 9	Games
Usuário 10	Medicina
Usuário 11	Telecomunicações
Usuário 12	Manutenção
Usuário 13	Automação
Usuário 14	ENTRETENIMENTO, EMPREENDEDORISMO
Usuário 15	agronomia
Usuário 16	Agricultura
Usuário 17	Produção Vegetal
Usuário 18	Controladores Lógicos Programáveis (CLP)
Usuário 19	Psicologia
Usuário 20	Fabricação
Usuário 21	Processos Químicos Industriais
Usuário 22	Advogado
Usuário 23	Bolsa de Valores

C.3 *System Usability Scale*

Perguntas referentes ao método SUS.

- ❏ P01 - Eu acho que gostaria de usar esse sistema com frequência.
- ❏ P02 - Eu acho o sistema desnecessariamente complexo.
- ❏ P03 - Eu achei o sistema fácil de usar.
- ❏ P04 - Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
- ❏ P05 - Eu acho que as várias funções do sistema estão muito bem integradas.
- ❏ P06 - Eu acho que o sistema apresenta muita inconsistência.
- ❏ P07 - Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.
- ❏ P08 - Eu achei o sistema atrapalhado de usar.

- ❏ P09 - Eu me senti confiante ao usar o sistema.
- ❏ P10 - Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

C.4 Resultados obtidos pelo *System Usability Scale*

Tabela 11 – Resultados obtidos pelo método System Usability Scale

Nome	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10
Usuário 1	5	1	5	1	5	1	5	1	4	5
Usuário 2	3	2	4	3	4	2	5	2	4	2
Usuário 3	3	1	5	1	4	3	4	1	5	3
Usuário 4	5	2	4	4	4	3	5	2	5	3
Usuário 5	5	1	5	2	5	1	5	1	5	1
Usuário 6	4	1	5	2	5	1	4	1	4	1
Usuário 7	3	1	4	1	5	2	5	2	5	3
Usuário 8	4	2	4	1	5	1	5	1	4	1
Usuário 9	5	1	5	1	5	1	3	1	5	1
Usuário 10	5	2	5	3	5	1	4	1	4	2
Usuário 11	5	3	5	3	5	1	5	2	5	5
Usuário 12	4	1	5	1	4	1	5	1	4	2
Usuário 13	5	2	4	4	4	2	4	1	5	2
Usuário 14	2	2	5	5	4	2	5	1	3	5
Usuário 15	4	1	5	2	5	1	5	1	4	2
Usuário 16	4	5	5	2	5	1	4	1	5	1
Usuário 17	4	2	3	1	4	1	4	1	4	1
Usuário 18	4	1	4	1	5	1	4	1	5	1
Usuário 19	4	1	4	3	5	1	5	2	4	3
Usuário 20	3	1	5	4	5	1	5	1	4	2
Usuário 21	3	1	4	3	3	2	4	2	3	2
Usuário 22	5	1	2	5	2	1	3	2	2	4
Usuário 23	5	1	5	3	5	1	5	2	5	1