PREVISÃO DE INSUFICIÊNCIA CARDÍACA COM APRENDIZADO DE MÁQUINA EM UM WEBSITE

ANDRÉ FILIPE CALDAS LARANJEIRA

MONOGRAFIA DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

INSTITUTO DE CIÊNCIAS EXATAS

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

PREVISÃO DE INSUFICIÊNCIA CARDÍACA COM APRENDIZADO DE MÁQUINA EM UM WEBSITE

ANDRÉ FILIPE CALDAS LARANJEIRA

Orientador: PROF. DR. ALEXANDRE RICARDO SOARES ROMARIZ, ENE/UNB

MONOGRAFIA DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

PUBLICAÇÃO IE.TG - XXX/AAAA
BRASÍLIA-DF, 19 DE MAIO DE 2021.

UNIVERSIDADE DE BRASÍLIA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

PREVISÃO DE INSUFICIÊNCIA CARDÍACA COM APRENDIZADO DE MÁQUINA EM UM WEBSITE

ANDRÉ FILIPE CALDAS LARANJEIRA

MONOGRAFIA DE BACHARELADO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM ENGENHARIA DA COMPUTAÇÃO.

APROVADA POR:

Prof. Dr. Alexandre Ricardo Soares Romariz, ENE/UnB
Orientador

Prof. Dr. Adson Ferreira da Rocha, ENE/UnB
Examinador interno

Prof. Dr. Li Weigang, CIC/UnB
Examinador interno

BRASÍLIA, 19 DE MAIO DE 2021.

FICHA CATALOGRÁFICA
ANDRÉ FILIPE CALDAS LARANJEIRA
Previsão de insuficiência cardíaca com aprendizado de máquina em um website
2021xv, 32p., 201x297 mm
(CIC/IE/UnB, Bacharel, Engenharia da computação, 2021)
Monografia de Bacharelado - Universidade de Brasília
INSTITUTO DE CIÊNCIAS EXATAS - Departamento de Ciência da computação

REFERÊNCIA BIBLIOGRÁFICA

ANDRÉ FILIPE CALDAS LARANJEIRA (2021) Previsão de insuficiência cardíaca com aprendizado de máquina em um website. Monografia de Bacharelado em Engenharia da computação, Publicação xxx/AAAA, Departamento de Ciência da computação, Universidade de Brasília, Brasília, DF, 32p.

CESSÃO DE DIREITOS

AUTOR: André Filipe Caldas Laranjeira
TÍTULO: Previsão de insuficiência cardíaca com aprendizado de máquina em um website.
GRAU: Bacharel ANO: 2021

André Filipe Caldas Laranjeira
Área Octogonal Sul 2, Bloco G, Apartamento 601; Octogonal, Brasília

Agradecimentos

Agradeço a Deus por ter me abençoado e me guiado durante todo o meu curso, cuidando de minha vida de maneira maravilhosa. Agradeço à minha família por ter me sustentado até aqui e me proporcionado amor e encorajamento. Agradeço aos meus professores por tudo o que eles me ensinaram e pelos desafios que eles me propuseram. Agradeço aos meus colegas de curso por terem me ajudado ao longo do curso e me inspirado a sempre dar o melhor de mim.

Resumo

Este trabalho consiste em uma comparação entre modelos de aprendizado de máquina do tipo perceptron multicamada e floresta aleatória treinados para a previsão de sobrevivência à insuficiência cardíaca e em um website auxiliar para utilização do melhor modelo. A avaliação dos modelos de treinamento se baseou na melhor média de acurácia de previsão envolvendo 20 subconjuntos de validação e 100 subconjuntos de teste. Ao total 5346 modelos de treinamento foram avaliados e o modelo mais bem classificado obteve uma média de acurácia comparável àquela do artigo de referência utilizado. A implementação do website auxiliar também obteve êxito ao simplificar o acesso ao melhor modelo de previsão. Para trabalho futuros, planeja-se a avaliação de mais tipos de modelo de aprendizado de máquina e suas combinações de hiperparâmetros e a realização de testes do melhor modelo com pacientes contemporâneos.

Abstract

This graduation project report consists of a comparison between multilayer perceptron and random forest machine learning models trained to predict heart failure survival and of an auxiliar website to facilitate the use of the best machine learning model. The machine learning models were evaluated based on the best average of prediction accuracies for 20 validation subsets and 100 test subsets. In total, 5346 machine learning models were evaluated and the best model displayed an average of prediction accuracies on par with the results obtained in an article referenced by this thesis. The auxiliar website was also successful in simplifying access to the best machine learning model. For future works, we plan on evaluating more classes or categories of machine learning models and more combinations of hyperparameters and to test the best machine learning model with contemporary patients.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS FONTE

LISTA DE TERMOS E SIGLAS

AHA	Associação Americana do Coração
API	Interface de Programação de Aplicações
HTTP	Protocolo de Transferência de Hipertexto
MCC	Coeficiente de Correlação de Matthews
ReLU	Unidade Linear Retificada

Capítulo 1

Introdução

Este trabalho realiza um estudo comparativo de modelos de aprendizado de máquina do tipo perceptron multicamada e floresta aleatória treinados com um conjunto de variáveis relacionadas à saúde cardiovascular e geral para realizar uma previsão da sobrevivência à insuficiência cardíaca de um paciente humano antes do final de um período de acompanhamento médico de duração média de 130 dias. Além disso, neste trabalho também foi implementado um website para permitir que o modelo de treinamento com a melhor acurácia de previsão no estudo mencionado possa ser acessado e utilizado de forma simples e direta por um público alvo abrangente.

Com este trabalho, espera-se exemplificar como uma aplicação concreta, mais especificamente um website, pode ser construída ao redor de um estudo de aprendizado de máquina para permitir que um modelo de aprendizado de máquina seja utilizado por profissionais da saúde para fornecer benefícios concretos no atendimento a pacientes.

O trabalho está organizado no formato descrito a seguir. O capítulo 2 apresenta ao leitor conceitos teóricos fundamentais para o entendimento deste trabalho. O capítulo 3 descreve o procedimento adotado pelo autor na realização deste trabalho. O capítulo 4 apresenta os resultados obtidos neste trabalho. Por fim, o capítulo 5 apresenta a conclusão deste trabalho e sugere futuras pesquisas a serem realizadas com base neste trabalho.

Todo o código fonte utilizado para este trabalho pode ser encontrado no repositório do *GitHub* para este trabalho[1].

1.1 Relevância da temática

A insuficiência cardíaca é uma condição clínica crônica e progressiva em que o coração não consegue bombear sangue suficiente para todo o corpo resultando em sintomas como fadiga e problemas respiratórios [American Heart Association, Inc. 2017]. Os principais fa-

[1] `https://github.com/AndreLaranjeira/ML-HeartFailureSurvival`

tores que aumentam o risco de desenvolvimento de insuficiência cardíaca são doenças arteriais coronarianas, hipertensão, diabetes, obesidade e fumo [Virani *et al.* 2021, p.399]. Além disso, estudos recentes demonstraram que a falta de boas condições físicas nos sistemas cardíaco e respiratório também contribui para o aumento do risco de desenvolvimento de insuficiência cardíaca [Virani *et al.* 2021, p.62].

A ocorrência de insuficiência cardíaca possui grande relevância nos dias atuais. Segundo a Associação Americana do Coração (AHA), dados coletados entre 2015 e 2018 apontam que 6 milhões de adultos estadunidenses com mais de 20 anos tiveram insuficiência cardíaca [Virani *et al.* 2021, p.8] e que 83.616 estadunidenses vieram a óbito em 2018 por insuficiência cardíaca [Virani *et al.* 2021, p.485]. No Brasil, um estudo publicado em 2019 na revista Acta Fisiátrica [Nogueira *et al.* 2019] utilizou dados de 2013 para estimar que 1,7 milhões de brasileiros já tiveram insuficiência cardíaca e outro estudo publicado em 2014 nos Arquivos Brasileiros de Cardiologia [Gaui *et al.* 2014] aponta que aproximadamente 27.702 brasileiros vieram a óbito em 2011 por insuficiência cardíaca.

Por fim, devemos salientar o impacto econômico causado por essa condição clínica. De acordo com uma diretriz de insuficiência cardíaca dos Arquivos Brasileiros de Cardiologia [Comitê Coordenador da Diretriz de Insuficiência Cardíaca 2018], os gastos globais governamentais e privados com essa condição clínica foram de R$ 14,5 bilhões apenas em 2015.

O uso de programas de aprendizado de máquina que prevejam a ocorrência de insuficiência cardíaca e a mortalidade por insuficiência cardíaca pode auxiliar na redução do número de pessoas que sejam afligidas e que venham a óbito por essa condição médica, respectivamente. Isso resultaria na diminuição dos recursos monetários despendidos globalmente, direta ou indiretamente, no tratamento da insuficiência cardíaca. Além disso, a saúde geral da população mundial seria beneficiada com a diminuição do número de ocorrências de insuficiência cardíaca.

1.2 Literatura existente

Artigos recentes exploraram vários aspectos sobre como o aprendizado de máquina pode ser utilizado para combater a insuficiência cardíaca. Um artigo escrito por Awan, Sohel e outros revisou estudos existentes acerca das possibilidades de uso de aprendizado de máquina para melhorar os tratamentos e diagnósticos de insuficiência cardíaca e assim diminuir gastos com essa condição médica [Awan *et al.* 2018]. Um outro artigo publicado no jornal europeu de insuficiência cardíaca utiliza o aprendizado de máquina para correlacionar pacientes e desenvolver uma nova métrica de risco de óbito por insuficiência cardíaca que obteve mais acurácia que as métricas existentes [Adler *et al.* 2020]. Por fim, um outro artigo publicado no jornal da sociedade americana de ecocardiografia propós um programa de aprendizado de máquina para identificar a presença de insuficiência cardíaca com preservação na taxa de ejeção, um tipo específico de insuficiência cardíaca, por meio de imagens

ecocardiográficas que mensurem as velocidades miocárdicas em repouso e durante exercício [Tabassian *et al.* 2018].

Dentre a literatura existente, forneceremos destaque especial ao artigo publicado por Chicco e Jurman [Chicco and Jurman 2020]. Esse artigo faz uso de um conjunto de dados de pacientes que foram hospitalizados com insuficiência cardíaca [Larxel 2020] disponibilizado em um artigo científico de 2017 [Ahmad *et al.* 2017], utilizando esse conjunto de dados tanto para realizar uma análise comparativa da performance obtida na previsão de sobrevivência à insuficiência cardíaca com o uso de difentes modelos de treinamento e variáveis dos pacientes como para identificar quais variáveis do conjunto de dados possuem maior correlação com a ocorrência de óbito por insuficiência cardíaca. Adotamos o artigo publicado por Chicco e Jurman como a principal referência utilizada neste trabalho devido à semelhança dos objetivos propostos para o uso do aprendizado de máquina neste trabalho e no artigo em questão e ao fato de que este trabalho utiliza o mesmo conjunto de dados que o artigo em questão como fonte de experiência do aprendizado de máquina.

Capítulo 2

Apresentação teórica

Neste capítulo, alguns conceitos teóricos são explanados com o intuito de fornecer ao leitor o embasamento teórico necessário para a compreensão completa deste trabalho.

2.1 Aprendizado de máquina

O campo de estudo de aprendizado de máquina é uma vasta área da computação, possuindo várias aplicações, objetos de estudo e focos de pesquisa interdisciplinares. Resumidamente, podemos dizer que essa área estuda como "construir programas de computador que melhoram seu desempenho em alguma tarefa por meio da experiência" [Mitchell 1997, p.29]. Atualmente, utilizamos o aprendizado de máquina para várias aplicações como algoritmos de recomendações de conteúdo, programas de reconhecimento e classificação de imagens e a realização de análises de risco financeiro.

Para utilizarmos o aprendizado de máquina para resolvermos alguma problema, faz-se necessário definir matematicamente uma tarefa a ser realizada, uma ou mais métricas de desempenho atreladas à realização da tarefa e a fonte de experiência que será utilizada pelo modelo para aprender a realizar a tarefa [Mitchell 1997, p.29]. Também precisamos escolher um ou mais tipos de modelo de aprendizado de máquina que serão utilizados para aprender a realizar a tarefa com base em um treinamento feito com a fonte de experiência avaliado sob a ótica das métricas de desempenho escolhidas.

2.1.1 Especificação de uma tarefa

Qualquer problema de aprendizado de máquina deve possuir uma tarefa a ser realizada, que representa o objetivo a ser atingido pelo uso de aprendizado de máquina. A especificação dessa tarefa sempre deve possuir o formato de uma função matemática para permitir que um programa de computador consiga aprendê-la. Assim, podemos afirmar que qualquer tarefa de aprendizado de máquina pode ser representada, genericamente, pela função $T : V \rightarrow R$,

onde V é um conjunto de variáveis disponibilizado para a realização da tarefa de aprendizado de máquina e R é o resultado esperado da tarefa de aprendizado de máquina.

Para um programa de aprendizado de máquina realizar a tarefa proposta, este deve aprender a função T. Entretanto, na grande maioria dos problemas de aprendizado de máquina, a função T não é conhecida e o problema proposto se resume a aproximar uma *descrição operacional* de T [Mitchell 1997, p.8]. Dessa forma, o programa de aprendizado de máquina deve então utilizar o processo de aprendizado com base na fonte de experiência para adquirir uma função $\hat{T}$ que seja uma boa aproximação da função T.

Neste trabalho, o problema de previsão de sobrevivência à insuficiência cardíaca pode ser descrito como sendo um problema de *classificação binária*, em que a tarefa proposta é representada pela função $P : V_p \rightarrow \{0, 1\}$, onde V_p são as variáveis fornecidas que descrevem o paciente e o conjunto imagem $\{0, 1\}$ representa uma previsão se o paciente irá sobreviver à insuficiência cardíaca (0) ou não (1).

2.1.2 Fonte de experiência

Para permitir que um programa de aprendizado de máquina aprenda uma aproximação $\hat{T}$ da função T, é necessário a utilização de uma fonte de experiência que forneça, direta ou indiretamente, uma maneira do programa de treinamento inferir o comportamento da função T. Essa fonte de experiência pode ser obtida de várias formas, as quais variam consideravelmente dependendo da tarefa em questão e do método de aprendizado almejado para o programa computacional, de forma que o tipo de recurso utilizado como fonte de experiência não apenas é determinante no sucesso ou fracasso do aprendizado, como também define uma categoria de aprendizado que será adotada pelo programa computacional.

O tipo mais comum de fonte de experiência utilizada é um conjunto de dados com exemplos que possuem variáveis e o resultado da aplicação dessas variáveis à função T, categoria de aprendizado conhecida como aprendizado supervisionado. Alguns outros tipos de fontes de experiência e suas respectivas categorias de aprendizado incluem: o uso de um conjunto de dados com exemplos com variáveis mas sem nenhum resultado da função T, categoria conhecida como aprendizado não supervisionado; o uso de um conjunto de dados com exemplos com variáveis mas nem sempre com o resultado da aplicação dessas variáveis à função T, categoria conhecida como aprendizado semi-supervisionado; e uma exploração da função T feita pelo próprio programa computacional com base em uma métrica de recompensa, categoria conhecida como aprendizado por reforço.

Para que a função de aproximação $\hat{T}$ aprendida pelo programa de aprendizado de máquina com base na fonte de experiência utilizada seja uma boa aproximação da função T, é necessário que a fonte de experiência utilizada seja uma boa aproximação dos exemplos que o programa de aprendizado de máquina encontrará ao longo de sua avaliação e uso, uma suposição que é apenas parcialmente verdadeira [Mitchell 1997, p.6]. Isso pode levar a ocor-

rência de um fenômeno denominado sobreajuste (ou *overfitting*) no processo de aprendizado do programa computacional. O sobreajuste ocorre quando o uso de uma fonte de experiência pequena ou com ruídos estatísticos resulta em uma aproximação $\hat{T}$ aprendida pelo programa computacional que se adequa melhor do que T aos dados da fonte de experiência utilizada mas que não generaliza corretamente os dados englobados em todo o domínio da função T [Mitchell 1997, p.79-80].

Neste trabalho, o programa de previsão de sobrevivência à insuficiência cardíaca utilizou como fonte de experiência para seu treinamento um conjunto de dados [Larxel 2020] composto de exemplos com variáveis e o resultado da aplicação dessas variáveis à função T, tomando parte, assim, na categoria de problemas de aprendizado supervisionado. O capítulo 3 aborda alguns cuidados utilizados para evitar a ocorrência de sobreajuste no processo de aprendizado.

2.1.3 Modelos de aprendizado

Com a definição da tarefa a ser realizada e da fonte de experiência, devemos escolher um ou mais modelos de aprendizado de máquina que serão treinados pelo programa computacional. Cada modelo de aprendizado de máquina possui uma forma específica para representar a função $\hat{T}$ e para aprender com a fonte de experiência disponibilizada. Dessa forma, antes de qualquer outra consideração, o modelo de treinamento escolhido deve ser compatível com a categoria de aprendizado estabelecida pela fonte de experiência escolhida. Cada modelo de aprendizado também possui um conjunto de hiper-parâmetros que são utilizados para a realização de ajustes finos na representação de $\hat{T}$ utilizada pelo modelo e no método de aprendizado empregado. Com a otimização desses hiper-parâmetros, o modelo de aprendizado pode aprender mais com a fonte de experiência e, consequentemente, melhorar a aproximação $\hat{T}$ obtida em relação à função T.

A escolha do modelo de aprendizado apresenta um *trade-off* importante na representação escolhida para a função $\hat{T}$: quanto maior a representatividade do modelo, maior é o número de dados necessários para treiná-lo [Mitchell 1997, p.8] e menor é a interpretabilidade do modelo [James *et al.* 2013, p.25]. Essa é uma consideração importante a ser feita dependendo do tamanho da fonte de experiência utilizada e do próposito para o qual o aprendizado de máquina está sendo empregado.

Neste trabalho, o programa de previsão de sobrevivência à insuficiência cardíaca avaliou dois tipos de modelos de aprendizado distintos para aprendizado de máquina: perceptron multicamada e floresta aleatória. Ambos estes modelos são utilizados para problemas de aprendizado da categoria de aprendizado supervisionado e possuem um grande conjunto de combinações possíveis de hiper-parâmetros.

2.1.3.1 Perceptron multicamada

O modelo de aprendizado de perceptron multicamada é um tipo de rede neural artificial, classe de modelos de aprendizado inspirada pelos sistemas de aprendizado biológicos compostos por redes de neurônios interconectados. As redes neurais são compostas por várias unidades computacionais agrupadas em camadas, cada unidade computacional recebendo um conjunto de entradas reais e fornecendo uma saída real, a qual pode ser utilizada como entrada de outra unidade computacional da rede neural [Mitchell 1997, p.82].

O perceptron é uma das unidades computacionais que podem ser utilizadas para compor uma rede neural artificial. Cada perceptron pode ser descrito por uma função $P(\vec{x} \cdot \vec{w})$, onde $\vec{x} = (1, x_1, x_2, ..., x_n) \in \Re^{n+1}$ é um vetor de entradas reais, e $\vec{w} = (w_0, w_1, ..., w_n) \in \Re^{n+1}$ é um vetor de pesos, onde cada valor w_i representa o peso que o perceptron fornece à entrada x_i. Existem várias funções que podem ser utilizadas como a função P que descreve o perceptron com alguns dos exemplos mais comuns sendo as funções degrau, as funções sigmoide, as funções de Unidade Linear Retificada (ReLU) e a função identidade. Os valores de cada peso w_i do perceptron variam ao longo do processo de aprendizagem com a fonte de experiência e geralmente possuem como valor inicial um número aleatório pertencente ao intervalo $(-1, 1)$.

Quando temos uma rede neural composta por várias camadas de perceptrons, temos o modelo de aprendizado conhecido como perceptron multicamada. Ao longo do processo de aprendizado supervisionado do perceptron multicamada, as entradas de cada exemplo do conjunto de dados são inseridas na rede e a saída fornecida pela rede é comparada com a saída registrada no conjunto de dados. Após a execução de certo número de exemplos, o perceptron multicamada executa um algoritmo de retropropagação para ajustar os pesos de seus perceptrons de forma a minimizar uma medida de erro, que geralmente é o erro quadrático entre as saídas geradas pela rede neural e as saídas registradas no conjunto de dados [Mitchell 1997, p.97]. Cada treinamento realizado desta maneira sobre todo o conjunto de dados é conhecido como uma época e, após certo número de épocas de aprendizado, o modelo de perceptron multicamada obtém uma boa aproximação $\hat{T}$. Para a execução do algoritmo de retropropagação, é necessário que a função P utilizada pelos perceptrons de cada camada seja uma função diferenciável, de forma que todas as funções de ativação utilizadas em um perceptron multicamada devem ser diferenciáveis.

No modelo de perceptron multicamada, alguns dos hiper-parâmetros que podemos modificar, além das dimensões da rede neural em termos de número de camadas e de número de perceptrons por camada, incluem a função diferenciável utilizada como função de ativação P dos perceptrons de uma camada, a medida de erro minimizada na execução do algoritmo de retropropagação, o número de épocas de aprendizado, a aplicação de um fator de *dropout* no aprendizado de uma camada e a utilização de um fator de regularização sobre os pesos dos perceptrons de uma camada. Destes, os fatores de *dropout* e de regularização de pesos merecem maior explicação. Um fator $d \in [0, 1)$ de *dropout* para uma camada de perceptrons

faz com que, durante o treinamento, os perceptrons dessa camada sobrescrevam aleatoriamente uma fração d de suas entradas com o valor 0, permitindo que todas as entradas sejam treinadas para influenciarem a ativação do perceptron. Já o fator de regularização de pesos estabelece, para cada perceptron de uma camada, uma penalidade na medida de erro que é proporcional à uma função dos pesos do perceptron, como soma ou soma quadrática, incentivando os pesos do perceptron a se manterem baixos para diminuir a complexidade da aproximação $\hat{T}$ aprendida [Mitchell 1997, p.111].

2.1.3.2 Floresta aleatória

O modelo de aprendizado do tipo floresta aleatória é um tipo de modelo agrupado sob a classificação de modelos de aprendizado baseados em árvores de decisão, os quais utilizam uma ou mais árvores de decisão para gerar uma aproximação $\hat{T}$ para a função T. Dessa forma, antes de podermos explicar o modelo de aprendizado do tipo floresta aleatória, devemos explicar o modelo de aprendizado do tipo árvore de decisão.

Uma árvore de decisão é um modelo de aprendizado que busca subdividir o domínio V da função T em um número de regiões menores $R_1, R_2, ..., R_n$ de forma que $R_i \cap R_j = \emptyset \; \forall i \neq j$ e que essas regiões agrupem os exemplos fornecidos pelo conjunto de dados utilizado como fonte de experiência. A representação final da função $\hat{T}$ é um conjunto de regras de divisão de V cuja representação gráfica se assemelha a uma árvore e cujo resultado fornecido para um arranjo de variáveis que se encontre dentro de uma região $R_i \subset V$ equivale à média ou à moda dos resultados de todos os exemplos do conjunto de dados utilizado como fonte de experiência cujas variáveis também se encontrem dentro da região R_i [James *et al.* 2013, p.303]. Cada subdivisão feita em V para gerar uma nova região $R_i \subset V$ busca sempre diminuir ao máximo, por meio de uma abordagem gulosa, a soma dos quadrados dos erros entre os resultados da aplicação de $\hat{T}$ e os resultados dos exemplos do conjunto de dados usado como fonte de experiência, no caso de um problema de regressão, ou uma medida de variância dos agrupamentos de classes dos exemplos do conjunto de dados da fonte de experiência, no caso de um problema de classificação [James *et al.* 2013, p.306-307, 312].

A utilização de uma única árvore de decisão para gerar uma aproximação $\hat{T}$ para a função T não fornece bons resultados, mas a utilização de uma aproximação $\hat{T}$ gerada pelo consenso do resultado de várias árvores de decisão pode fornecer resultados com excelente acurácia [James *et al.* 2013, p.303]. O modelo de aprendizado de floresta aleatória gera uma aproximação $\hat{T}$ com base no consenso de um número arbitrário de árvores de decisão, onde cada árvore de decisão é gerada utilizando um conjunto de dados obtido pela escolha aleatória com possibilidade de repetição dos exemplos do conjunto de dados que serve como fonte de experiência, de forma que ambos conjuntos de dados tenham o mesmo tamanho (técnica conhecida como *bootstrapping*). Além disso, na construção de uma floresta aleatória, sempre que uma das árvores de decisão subdivide o dominío V, esta considera apenas um subconjunto, também aleatório, de variáveis do conjunto de dados utilizado como fonte

de experiência. Estas duas características auxiliam na diminuição da variância dos resultados gerados pelo consenso das árvores de decisão, aumentando a acurácia dos resultados da aproximação $\hat{T}$ fornecida por uma floresta aleatória [James *et al.* 2013, p.316-321].

No modelo de floresta aleatória, alguns dos hiper-parâmetros que podemos modificar incluem: o número de árvores de decisão utilizado; a medida de variância ou de erro a ser minimizada na realização das subdivisões do domínio V; o tamanho do subconjunto de variáveis, pertencentes ao conjunto de dados utilizado como fonte de experiência, consideradas na realização de cada subdivisão do domínio V; e vários hiper-parâmetros utilizados para limitar o crescimento de cada árvore de decisão como número mínimo de exemplos para compor uma folha, número mínimo de exemplos para realizar uma nova subdivisão do domínio V, profundidade máxima das árvores de decisão e número máximo de folhas das árvores de decisão. A utilização de hiper-parâmetros que limitam o crescimento das árvores de decisão resulta em uma aproximação $\hat{T}$ com menor complexidade, o que diminui a probabilidade de ocorrência de sobreajuste [James *et al.* 2013, p.307].

2.1.4 Métricas de desempenho

Por fim, devemos escolher uma ou mais métricas de desempenho para mensurar objetivamente o desempenho do programa de aprendizado de máquina no aprendizado da tarefa T proposta. Cada categoria de aprendizado possui suas próprias métricas de desempenho e a natureza do problema proposto também influencia as métricas utilizadas. Por exemplo, para a categoria de aprendizado supervisionado, as métricas de desempenho utilizadas diferem em casos de problemas de regressão e de classificação. A escolha das métricas de desempenho é importante para que a avaliação dos modelos de aprendizado treinados ilustre adequadamente o desempenho do modelo em aprender a tarefa T em conformidade com o objetivo a ser atingido pelo uso do aprendizado de máquina.

Neste trabalho, em que o problema estudado é categorizado como um problema de aprendizado supervisionado de classificação, a métrica de desempenho adotada foi a média da acurácia das classificações feitas por um modelo de treinamento para os exemplos contidos em um número de subconjuntos de dados utilizados para fins de validação ou de teste e, portanto, não utilizados no processo de aprendizado do modelo. Outras métricas que podem ser utilizadas para problemas dessa categoria incluem a precisão das classificações realizadas, a sensibilidade (*recall*) das classificações realizadas e o Coeficiente de Correlação de Matthews (MCC).

Para definirmos matematicamente cada uma dessas métricas de problemas de aprendizado supervisionado de classificação, vamos estabelecer que N é o número total de previsões feitas por um modelo, T_p é o número de previsões que foram verdadeiros positivos, T_n é o número de previsões que foram verdadeiros negativos, F_p é o número de previsões que foram falsos positivos, e F_n é o número de previsões que foram falsos negativos. Dessa forma, temos que $N = T_p + F_p + T_n + F_n$. Com essa definições, podemos definir a acurácia

(Acc) pela equação 2.1, a precisão ($Precision$) pela equação 2.2, a sensibilidade ($Recall$) pela equação 2.3, e o Coeficiente de Correlação de Matthews (MCC) pela equação 2.4.

$$Acc = \frac{T_p + T_n}{N} \tag{2.1}$$

$$Precision = \frac{T_p}{T_p + F_p} \tag{2.2}$$

$$Recall = \frac{T_p}{T_p + F_n} \tag{2.3}$$

$$MCC = \frac{(T_p \times T_n) - (F_p \times F_n)}{\sqrt{(T_p + F_p) \times (T_p + F_n) \times (T_n + F_p) \times (T_n + F_n)}} \tag{2.4}$$

2.2 Funcionamento de um website

O funcionamento de um website é um processo complexo que envolve vários aspectos relacionados ao servidor que hospeda o website, ao protocolo utilizado pelo navegador de internet do cliente para localizar esse servidor na internet, e ao processo de comunicação entre o servidor que hospeda o website e o navegador de internet para permitir que o cliente utilize o website. Para este trabalho, nosso foco estará apenas na estrutura do website em si, e não nos protocolos utilizados por um navegador de internet para localizar o website na internet ou em detalhes mais complexos da comunicação entre o navegador de internet do cliente e o servidor que hospeda o website. De forma bem resumida, podemos dividir a estrutura padrão de um website em dois componentes básicos, os quais não precisam ser implementados no mesmo programa ou mesmo armazenados no mesmo servidor: o *backend* e o *frontend*.

O *backend* de um website engloba o banco de dados utilizado para o armazenamento dos dados do website e as funções ou métodos utilizados para acessar, criar, editar, apagar ou de alguma forma gerenciar esses dados. Dessa forma, o *backend* geralmente é visualizado apenas pelos programadores envolvidos na construção do website e não pelos usuários e é utilizado para fornecer os dados que são visualizados e acessados pelos usuários no uso do website.

O *frontend* de um website é composto pelas telas do website que são renderizadas pelo browser e por funções ou métodos utilizados para requisitar dados ao *backend* do website e determinar quais telas ou elementos de telas que devem ser renderizados conforme as ações do usuário. Dessa forma, os usuários de um website geralmente interagem apenas com o *frontend* do website ao utilizar um navegador de internet.

Geralmente, o *backend* e o *frontend* de um website são construídos no mesmo programa

para serem executados de maneira acoplada no mesmo servidor e poderem se comunicar por meio de chamadas locais de funções. Entretanto, um novo padrão de projeto de websites tem se estabelecido, no qual o *backend* do website é construído em um programa separado do *frontend* para poder ser acessado diretamente por meio de requisições do Protocolo de Transferência de Hipertexto (HTTP) ou até mesmo por mais de um *frontend* simultaneamente. Quando isso ocorre, dizemos que o *backend* do website se trata de uma Interface de Programação de Aplicações (*API*).

A utilização de uma *API* como *backend* do website faz com que tenhamos que executar o *backend* do website e o *frontend* do website em servidores diferentes ou em portas diferentes do mesmo servidor, de forma que a comunicação entre *backend* e *frontend* do website passa a ser feita por requisições HTTP e não pela execução de funções locais. Embora essa arquitetura gere mais trabalho na programação da lógica do website, ela abre um número maior de possibilidades para o programador ao permitir que diversos *frontends* no formato de interfaces web, aplicativos ou clientes de protocolo HTTP utilizem, simultaneamente, a API de *backend* criada. Além disso, essa arquitetura facilita a realização de testes com o website ao permitir que a separação dos testes de *backend* e de *frontend* seja feita com mais facilidade.

No próximo capítulo, detalharemos o procedimento adotado ao longo deste trabalho.

Capítulo 3

Procedimento adotado

Neste capítulo, o procedimento adotado ao longo do trabalho é detalhado com o intuito de permitir ao leitor compreender melhor a lógica por trás de algumas escolhas feitas no decorrer do trabalho e de explorar alguns detalhes importantes da implementação dos módulos de programação.

3.1 Conjunto de dados

Como primeiro passo no processo de aprendizado de máquina, foi necessária a escolha de um conjunto de dados que possibilitasse o treinamento de modelos de aprendizado de máquina para uso em uma aplicação completa em tempo útil para a realização deste trabalho. Tendo isso em mente, a decisão recaiu sobre um conjunto de dados [Larxel 2020] disponibilizado em um artigo científico de 2017 [Ahmad *et al.* 2017] e utilizado em um artigo científico escrito por Chicco e Jurman [Chicco and Jurman 2020] voltado para o estudo de aprendizado de máquina para a previsão de sobrevivência à insuficiência cardíaca.

O conjunto de dados escolhido contém 299 registros de pacientes, cada um consistindo em 11 variáveis relacionadas à saúde geral e cardíaca do paciente, 1 variável indicando o tempo de acompanhamento médico do paciente e 1 variável indicando se o paciente veio a óbito antes do fim do acompanhamento médico realizado. Dessa forma, trata-se de um conjunto de dados pequeno, de fácil compreensão, que pudesse ser explorado com um certo grau de profundidade em um curto espaço de tempo, e com uma clara utilidade prática de auxiliar a prevenção de óbitos causados por insuficiência cardíaca em pacientes com base em uma previsão realizada por meio de aprendizado de máquina.

As variáveis contidas no conjunto de dados são descritas em maiores detalhes na tabela 3.1. Neste trabalho, todas as variáveis fornecidas pelo conjunto de dados exceto sódio sérico, tempo e evento de morte foram utilizadas como características para o treinamento de modelos de aprendizado de máquina. A variável de evento de morte foi utilizada como resultado a ser previsto pelo aprendizado de máquina.

Nome da variável	Tipo de dado	Descrição
age (idade)	Número decimal	Idade do paciente.
anaemia (anemia)	Booleano (0 ou 1)	Se o paciente está anêmico (1) ou não (0). A definição adotada para anemia é um nível hematócrito abaixo de 36%.
creatinine_phosphokinase (creatinina fosfoquinase)	Número inteiro	Nível da enzima creatinina fosfoquinase presente no sangue (mcg/L).
diabetes (diabetes)	Booleano (0 ou 1)	Se o paciente é diabético (1) ou não (0).
ejection_fraction (fração de ejeção)	Número inteiro	Percentual de sangue que deixa o coração a cada contração.
high_blood_pressure (pressão alta)	Booleano (0 ou 1)	Se o paciente está com hipertensão (1) ou não (0).
platelets (plaquetas)	Número decimal	Número de plaquetas no sangue (Kiloplaquetas/mL).
serum_creatinine (creatinina sérica)	Número decimal	Nível de creatinina sérica no sangue (mg/dL).
serum_sodium (sódio sérico)	Número inteiro	Nível de sódio sérico no sangue (mEq/L).
sex (sexo)	Binário (0 ou 1)	Se o paciente é do sexo masculino (1) ou feminino (0).
smoking (fumante)	Booleano (0 ou 1)	Se o paciente fuma (1) ou não (0).
time (tempo)	Número inteiro	Número de dias em que o paciente recebeu acompanhamento médico após a ocorrência de insuficiência cardíaca.
DEATH_EVENT (evento de morte)	Booleano (0 ou 1)	Se o paciente veio a óbito antes do fim do período de acompanhamento médico (1) ou não (0). O período de acompanhamento médico tem duração média de 130 dias.

Tabela 3.1: Descrição das variáveis do conjunto de dados utilizado.

Além dos aspectos úteis do conjunto de dados em si para a realização deste trabalho, o artigo científico de Chicco e Jurman que utilizou o conjunto de dados consiste em um estudo comparativo do desempenho de diferentes tipos de modelos de dados na previsão de sobrevivência à insuficiência cardíaca com base no conjunto de dados em si, proporcionando uma referência útil para a realização de escolhas do tipo de modelo de aprendizado de máquina a ser utilizado no conjunto de dados e para a comparação dos resultados atingidos no treinamento de modelos de aprendizado de máquina.

Apesar dessas qualidades, devemos ressaltar que o conjunto de dados escolhido não é sem falhas. A baixa quantidade de registros no conjunto de dados faz com que o treinamento de modelos de aprendizado de máquina seja extremamente suscetível a *overfitting*, resultando em um modelo que não forneça bons resultados em aplicações práticas. Também podemos notar que a baixa quantidade de registros em que o paciente veio a óbito, agravada pelas subdivisões do conjunto de dados em dados de treinamento, de teste e de validação, pode gerar modelos com um viés para previsões positivas, aumentando a probabilidade de que o modelo de aprendizado de máquina gere falsos negativos, em que o sistema prediz a sobrevivência à insuficiência cardíaca e o paciente vem a óbito. A forma como essas limitações da base de dados foram tratadas será descrita posteriormente.

3.2 Modelos de treinamento de aprendizado de máquina

Com a escolha do conjunto de dados feita, o próximo passo seria definir quais tipos de modelos de treinamento de aprendizado de máquina seriam avaliados para uso na aplicação e estabelecer um método específico de avaliação para permitir a comparação entre os diferentes modelos de treinamento e determinar qual deveria ser utilizado na aplicação.

3.2.1 Tipos de modelos testados

Inicialmente, o propósito deste trabalho consistia em avaliar exclusivamente o desempenho de modelos do tipo redes neurais de perceptron multicamada. Entretanto, a leitura da análise realizada e dos resultados obtidos no artigo científico de Chicco e Jurman [Chicco and Jurman 2020] com o conjunto de dados utilizado [Larxel 2020], bem como o tempo necessário para o treinamento de um único modelo de perceptron multicamada, levaram a realização de análises adicionais com modelos do tipo floresta aleatória, os quais obtiveram a melhor acurácia de previsão de acordo com os resultados do artigo científico de Chicco e Jurman e podem ser treinados em uma fração do tempo de treinamento de modelos do tipo perceptron multicamada.

3.2.2 Método de avaliação dos modelos de treinamento

Com o intuito de comparar diferentes modelos de treinamento de aprendizado de máquina, um método padrão de avaliação foi adotado com base no método utilizado no artigo científico de Chicco e Jurman [Chicco and Jurman 2020]. Esta subseção, com o intuito de justificar algumas das escolhas feitas na montagem do método de avaliação adotado, descreve formalmente tanto o método de avaliação de modelos de treinamento utilizado no artigo científico de Chicco e Jurman, como o método de treinamento utilizado neste trabalho.

3.2.2.1 Método de avaliação de modelos de treinamento utilizado no artigo científico

O artigo científico de Chicco e Jurman, que buscou estudar vários aspectos do aprendizado de máquina aplicado ao conjunto de dados [Larxel 2020], utilizou um método específico para avaliar quais foram os melhores modelos de aprendizado de máquina. Esse método consistiu em dois submétodos diferentes para modelos com otimização de hiper-parâmetros e para modelos sem otimização de hiper-parâmetros.

Para tipos de modelos com otimização de hiper-parâmetros, o conjunto de dados foi dividido em 60% de dados para treinamento, 20% de dados para validação e 20% de dados para teste. Primeiramente, vários modelos com diferentes hiper-parâmetros foram treinados com o subconjunto de dados de treinamento com o intuito de encontrar o conjunto de

hiper-parâmetros que gerasse o melhor coeficiente de correlação de Matthews (MCC) relativo à previsão feita sobre o subconjunto de dados de validação. Esse conjunto de hiper-parâmetros então foi utilizado em um novo modelo, treinado com o subconjunto de dados de treinamento, para obter o coeficiente de correlação de Matthews (MCC) relativo à previsão feita sobre o subconjunto de dados de teste. Esse procedimento foi realizado sobre 100 diferentes partições do conjunto de dados, para que a média e mediana dos 100 resultados de coeficiente de correlação de Matthews (MCC) relativos às previsões feitas sobre os subconjuntos de dados de teste com aquele tipo de modelo fossem calculados.

Para tipos de modelos sem otimização de hiper-parâmetros, o conjunto de dados foi dividido em 80% de dados para treinamento e 20% de dados para teste. Um modelo foi treinado com o subconjunto de dados de treinamento para obter o coeficiente de correlação de Matthews (MCC) relativo à previsão feita sobre o subconjunto de dados de teste. Esse procedimento foi realizado sobre 100 diferentes partições do conjunto de dados, para que a média e mediana dos 100 resultados de coeficiente de correlação de Matthews (MCC) relativos às previsões feitas sobre os subconjuntos de dados de teste com aquele tipo de modelo fossem calculados.

3.2.2.2 Método de avaliação de modelos de treinamento utilizado neste trabalho

Neste trabalho, o método utilizado para avaliar quais modelos de treinamento obtiveram os melhores resultados baseou-se no método de avaliação utilizado no artigo científico de Chicco e Jurman, mas com algumas alterações.

Em primeiro lugar, 100 inteiros de 32 bits sem sinal foram gerados e armazenados em um arquivo para serem utilizados como sementes na aleatoriedade inerente ao particionamento do conjunto de dados [Larxel 2020], de forma que todos os modelos agora utilizam as mesmas partições do conjunto de dados para realizarem seu treinamento e obterem seus resultados de validação e teste. Essa mudança foi feita com o intuito de permitir a replicação dos resultados obtidos e proporcionar uma forma mais justa de comparação entre diferentes modelos com diferentes hiper-parâmetros. Além disso, como o tempo de treinamento de alguns modelos pode ser consideravelmente longo, essa técnica permite que a obtenção de um resultado de validação ou teste para um mesmo modelo seja dividida em mais de uma execução, dado que um subconjunto de dados de validação e teste é fixo para uma dada semente escolhida.

Em segundo lugar, a otimização de hiper-parâmetros não pôde ser feita da mesma forma que o artigo científico de Chicco e Jurman, pois a quantidade de combinações de hiper-parâmetros existente não permitiria que todas as variações de hiper-parâmetros fossem testadas para um dado modelo e uma dada partição do conjunto de dados. Dessa forma, decidiu-se que a única otimização de hiper-parâmetros feita seria na quantidade de épocas de treinamento para modelos do tipo perceptron multicamada. O número de épocas escolhido para treinamento na obtenção dos resultados de teste será equivalente à época em que o modelo

teve a menor perda nos dados de validação (abordagem conhecida como *early stopping*). Para os demais parâmetros, decidiu-se adotar o seguinte procedimento: várias variações de hiper-parâmetros seriam utilizadas para se obter o resultado de predição sobre os subconjuntos de validação do primeiro quinto das partições de dados disponíveis (20), e as combinações de hiper-parâmetros com os resultados mais promissores (mais especificamente, os modelos que estiveram entre os melhores 20%) em relação ao subconjunto de validação seriam testadas sobre todos os subconjuntos de teste disponíveis (100) com o intuito de se calcular os seus resultados de predição. Isso permitiria que uma gigantesca quantidade de variações de hiper-parâmetros fosse experimentada para cada modelo, com apenas as variações de hiper-parâmetros mais promissoras sendo realmente testadas em todas as 100 possíveis partições do conjunto de dados disponível. Além disso, como o escopo desse trabalho envolve menos a comparação de diferentes modelos e mais a busca por um único conjunto de hiper-parâmetros que forneça bons resultados, podemos concluir que esse formato de análise seria mais adequado que aquele utilizado no artigo científico em que diferentes combinações de hiper-parâmetros poderiam ser utilizadas em diferentes partições de dados no cálculo do resultado de teste de um modelo.

Por fim, a medida utilizada para medir o desempenho de um modelo (nos resultados de validação e teste) foi a média da acurácia obtida em cada particionamento do conjunto de dados, em oposição ao uso do coeficiente de correlação de Matthews (MCC) utilizado no artigo científico.

Portanto, podemos resumir o procedimento adotado neste trabalho como sendo, para um dado conjunto de modelos com diferentes hiper-parâmetros, a obtenção dos resultados de validação para o primeiro quinto das sementes disponíveis e o cálculo subsequente dos resultados de teste com todas as sementes disponíveis para os modelos entre os melhores 20% no quesito melhor média de acurácia para os dados de validação. Os resultados de validação influenciariam na escolha de quais modelos seriam utilizados para o cálculo de resultados de teste e também, no caso de modelos do tipo perceptron multicamada, na escolha do número de épocas de treinamento (*early stopping*) para a obtenção de resultados de teste.

Por meio deste método de avaliação, espera-se que as falhas inerentes ao conjunto de dados utilizado sejam mitigadas. A utilização do critério de melhor média de acurácia na previsão de dados de 20 subconjuntos de validação e de 100 subconjuntos de teste é uma forma de se compensar a propensão a *overfitting* e a viés no treinamento com esse conjunto de dados. Isso ocorre pois a existência de múltiplos subconjuntos distintos de treinamento assegura que os melhores modelos serão aqueles capazes de obter resultados consistentes na previsão de validação e de teste em diversos cenários, desfavorecendo combinações de hiper-parâmetros mais suscetíveis à geração de modelos que apresentem *overfitting* e viés. Embora essa abordagem não solucione por completo as dificuldades apresentadas, sua utilização é um bom recurso na ausência de um conjunto de dados mais robusto e representativo da realidade.

3.3 Programação da avaliação do treinamento

Com os tipos de modelos a serem avaliados e o método de avaliação determinados, o próximo passo foi criar um programa na linguagem de programação *Python* para se automatizar a aplicação do método de avaliação em diferentes modelos de treinamento e salvar os resultados obtidos. Além disso, também foram criados 2 programas adicionais: um com o intuito de permitir que os resultados de avaliação obtidos fossem mais facilmente visualizados em formatos gráficos e o outro com o intuito de permitir que um modelo de treinamento pronto para realizar previsões de sobrevivência à insuficiência cardíaca fosse salvo em um arquivo.

Todos os programas foram construídos de maneira modularizada e buscando capturar a intenção do autor ao programá-los pela utilização de nomes significativos para variáveis e funções e pela utilização de funções com um único propósito sempre que possível. Dessa forma, essa seção não tem o intuito de descrever todos os detalhes de implementação dos programas, mas sim fazer uma descrição geral da forma como os programas funcionam, apresentando ao leitor imagens e trechos de código explicativos.

3.3.1 Programa para automatizar a avaliação

O programa para automatizar a avaliação dos modelos de treinamento foi escrito no arquivo *main.py* e permite que o usuário defina um conjunto de modelos de treinamento (não necessariamente do mesmo tipo de modelo) com diferentes hiperparâmetros, aplique um método de avaliação sobre os modelos definidos e salve os resultados obtidos. O usuário pode passar como argumentos na chamada do programa o tamanho do subconjunto de validação (utilizado apenas se o tipo de modelo requerer dados de validação), o tamanho do subconjunto de teste e uma *flag* para que o programa cronometre o tempo decorrido durante a avaliação. Naturalmente, o tamanho do subconjunto de validação e o tamanho do subconjunto de teste têm como valor padrão 20%, conforme especificado anteriormente no método adotado, mas o usuário tem a liberdade para alterar esses valores para utilizar o programa em outras aplicações. O programa também possui um argumento de ajuda para explicar o uso dos demais argumentos e um argumento de versão.

Além dessas configurações por passagem de argumentos, o corpo do programa pode ser modificado para alterar o funcionamento do programa. Pelo corpo do programa, o usuário pode definir o arquivo que contém o conjunto de dados a ser utilizado e quais colunas desse conjunto serão utilizadas como características de treinamento e rótulos, qual arquivo contém as sementes de aleatoriedade, quais modelos de treinamento serão testados nessa avaliação, o número da avaliação em si e o número de particionamentos do conjunto de dados utilizados para validação e teste. A modularização do programa permite que essas características sejam todas definidas como parâmetros de construção de classes, tornando sua modificação simples e intuitiva e reduzindo bastante o tamanho do programa principal em si.

Como exemplo simples da modularização do programa, apresentamos o trecho de código

que constrói a classe de extrator de conjunto de dados utilizada para ler os dados contidos em um arquivo de conjunto de dados 3.1. Nesse trecho de código, a classe *DatasetExtractor* foi importada do módulo *data_extractors.py*, o qual foi criado pelo autor para encapsular a lógica de leitura de dados de arquivos *csv*. Caso o usuário queira modificar o arquivo de conjunto de dados utilizado, as colunas de características de treinamento utilizadas ou as colunas de rótulo utilizadas, basta alterar os parâmetros fornecidos ao construtor da classe.

Listagem 3.1: Construção do extrator de conjunto de dados no programa de avaliação dos modelos de treinamento

```
dataset_extractor = DatasetExtractor(
    dataset_file_name='./data/data.csv',
    feature_columns_list=[
        'age',
        'anaemia',
        'creatinine_phosphokinase',
        'diabetes',
        'ejection_fraction',
        'high_blood_pressure',
        'platelets',
        'serum_creatinine',
        'sex',
        'smoking'
    ],
    label_columns_list=['DEATH_EVENT'],
    train_size=args.train_size,
    validation_size=args.validation_size
)
```

Após o usuário definir as configurações desejadas no corpo do programa e executar o programa com os argumentos apropriados, o programa iniciará a avaliação dos modelos de treinamento definidos. O tempo médio de avaliação depende da quantidade de modelos avaliados, do tipo dos modelos avaliados e dos hiperparâmetros dos modelos em si. As avaliações realizadas pelo autor que utilizaram modelos do tipo floresta aleatória foram relativamente rápidas, demorando algumas horas para serem executadas quando centenas de modelos haviam sido definidos. Já as avaliações feitas pelo autor com modelos do tipo perceptron multicamada foram consideravelmente lentas, demorando de 36 a 48 horas para serem executadas quando apenas 5 modelos haviam sido definidos. Essa discrepância significativa foi um grande entrave para a realização de mais avaliações com modelos do tipo perceptron multicamada.

Após o programa ter sido iniciado, algumas mensagens serão mostradas na tela para informar o progresso da avaliação. Após a avaliação ter sido finalizada, os resultados obtidos serão também mostrados na tela para conhecimento do usuário e salvos em um arquivo com extensão *csv*. O arquivo de resultados armazena, para cada modelo avaliado, o posicionamento do modelo em um rankeamento iniciado em 0 (porque essa posição é originada a partir de uma lista ordenada, a qual é indexada em 0), o tipo do modelo, os hiperparâmetros

do modelo, as acurácias de previsão para os subconjuntos de validação dos particionamentos utilizados para validação e a média dessas acurácias, e, caso o modelo tenha sido utilizado para realizar previsões nos subconjuntos de teste, as acurácias de previsão para os subconjuntos de teste dos particionamentos utilizados para teste e a média dessas acurácias. Podemos visualizar um exemplo de arquivo de resultados aberto no *LibreOffice Calc* e com algumas estilizações aplicadas na figura 3.1.

	A	B	C	D	E	F	G
1	model_number	model_type	model_params	all_validation_scores	mean_validation_score	all_test_scores	mean_test_score
2	0	random_forest	{'n_estimators': 210, 'criterion': 'gini',	[0.68, 0.78, 0.68, 0.77, 0	0.75	[0.717, 0.767, 0.75, 0	0.735
3	1	random_forest	{'n_estimators': 140, 'criterion': 'gini',	[0.67, 0.78, 0.7, 0.82, 0.	0.748	[0.7, 0.75, 0.767, 0.6	0.732
4	2	random_forest	{'n_estimators': 230, 'criterion': 'gini',	[0.68, 0.78, 0.67, 0.78, 0	0.748	[0.683, 0.717, 0.767,	0.732
5	3	random_forest	{'n_estimators': 170, 'criterion': 'gini',	[0.68, 0.78, 0.72, 0.8, 0.	0.748	[0.7, 0.75, 0.767, 0.7	0.734
6	4	random_forest	{'n_estimators': 240, 'criterion': 'gini',	[0.73, 0.78, 0.68, 0.8, 0.	0.747	[0.683, 0.75, 0.75, 0.0	0.733
7	5	random_forest	{'n_estimators': 40, 'criterion': 'gini',	[0.7, 0.78, 0.68, 0.8, 0.7	0.746	[0.717, 0.75, 0.75, 0.0	0.731
8	6	random_forest	{'n_estimators': 290, 'criterion': 'gini',	[0.68, 0.78, 0.68, 0.75, 0	0.746	[]	-
9	7	random_forest	{'n_estimators': 30, 'criterion': 'gini',	[0.72, 0.78, 0.7, 0.77, 0.	0.746	[]	-
10	8	random_forest	{'n_estimators': 260, 'criterion': 'gini',	[0.68, 0.78, 0.68, 0.75, 0	0.745	[]	-
11	9	random_forest	{'n_estimators': 250, 'criterion': 'gini',	[0.72, 0.78, 0.68, 0.77, 0	0.745	[]	-
12	10	random_forest	{'n_estimators': 160, 'criterion': 'gini',	[0.77, 0.78, 0.67, 0.73, 0	0.745	[]	-
13	11	random_forest	{'n_estimators': 120, 'criterion': 'gini',	[0.75, 0.78, 0.7, 0.8, 0.7	0.744	[]	-
14	12	random_forest	{'n_estimators': 220, 'criterion': 'gini',	[0.68, 0.78, 0.68, 0.75, 0	0.744	[]	-
15	13	random_forest	{'n_estimators': 70, 'criterion': 'gini',	[0.7, 0.77, 0.67, 0.8, 0.7	0.744	[]	-
16	14	random_forest	{'n_estimators': 110, 'criterion': 'gini',	[0.7, 0.78, 0.67, 0.8, 0.7	0.743	[]	-
17	15	random_forest	{'n_estimators': 270, 'criterion': 'gini',	[0.7, 0.78, 0.68, 0.75, 0.	0.741	[]	-
18	16	random_forest	{'n_estimators': 150, 'criterion': 'gini',	[0.68, 0.78, 0.67, 0.77, 0	0.74	[]	-
19	17	random_forest	{'n_estimators': 280, 'criterion': 'gini',	[0.72, 0.78, 0.67, 0.78, 0	0.74	[]	-
20	18	random_forest	{'n_estimators': 60, 'criterion': 'gini',	[0.72, 0.78, 0.67, 0.75, 0	0.74	[]	-
21	19	random_forest	{'n_estimators': 80, 'criterion': 'gini',	[0.7, 0.78, 0.68, 0.77, 0.	0.74	[]	-
22	20	random_forest	{'n_estimators': 300, 'criterion': 'gini',	[0.72, 0.78, 0.62, 0.78, 0	0.74	[]	-
23	21	random_forest	{'n_estimators': 180, 'criterion': 'gini',	[0.72, 0.78, 0.63, 0.78, 0	0.738	[]	-
24	22	random_forest	{'n_estimators': 100, 'criterion': 'gini',	[0.7, 0.78, 0.67, 0.75, 0.	0.738	[]	-
25	23	random_forest	{'n_estimators': 50, 'criterion': 'gini',	[0.7, 0.78, 0.67, 0.77, 0.	0.737	[]	-
26	24	random_forest	{'n_estimators': 130, 'criterion': 'gini',	[0.7, 0.78, 0.7, 0.75, 0.7	0.736	[]	-
27	25	random_forest	{'n_estimators': 200, 'criterion': 'gini',	[0.67, 0.78, 0.68, 0.75, 0	0.736	[]	-
28	26	random_forest	{'n_estimators': 20, 'criterion': 'gini',	[0.73, 0.75, 0.68, 0.8, 0.	0.736	[]	-
29	27	random_forest	{'n_estimators': 90, 'criterion': 'gini',	[0.65, 0.78, 0.68, 0.73, 0	0.735	[]	-
30	28	random_forest	{'n_estimators': 190, 'criterion': 'gini',	[0.68, 0.78, 0.62, 0.77, 0	0.735	[]	-
31	29	random_forest	{'n_estimators': 10, 'criterion': 'gini',	[0.67, 0.73, 0.65, 0.75, 0	0.713	[]	-
32							
33							
34							

Figura 3.1: Exemplo de arquivo de resultados de avaliação aberto no programa *LibreOffice Calc* e com algumas estilizações.

Para facilitar a identificação dos resultados de avaliação, o programa tenta gravar os resultados em um arquivo na pasta *results* com o nome *E[Número da avaliação com 4 dígitos].csv* (e.g: *results/E0001.csv*). Caso o número da avaliação não tenha sido fornecido, o programa tenta gravar os dados no arquivo *results/results_save.csv*. Caso o programa não consiga gravar os resultados em um desses arquivos por algum motivo, como, por exemplo, a inexistência da pasta *results*, o programa ainda tenta realizar uma segunda gravação dos resultados, no diretório de onde o programa foi chamado, em um arquivo de *fallback* de nome *fallback_results_save.csv*.

3.3.2 Programa para gerar gráficos dos resultados

O programa para gerar gráficos de resultados foi escrito no arquivo *plot_results.py* e permite que o usuário selecione um modelo de uma das avaliações realizadas e gere uma representação gráfica dos resultados obtidos para esse modelo. O usuário pode passar como argumentos na chamada do programa o número da avaliação selecionada, o número do modelo selecionado, a ação almejada do programa (apenas mostar o gráfico gerado ou salvar o gráfico gerado em um arquivo), um nome de arquivo para receber o gráfico gerado, o tipo

de gráfico a ser gerado (gráfico de barras, gráfico de caixa ou histograma) e a categoria de resultados a ser analisada (validação, teste ou uma comparação entre ambos).

O usuário é livre para combinar esses argumentos de maneira a extrair o maior valor possível dos resultados de avaliação, mas deve-se ressaltar que o argumento de nome de arquivo não será utilizado se a ação escolhida for apenas mostrar o gráfico gerado, que valores inválidos de número de avaliação ou de número de modelo geraram mensagens de erro e que a execução do programa deve ser feita a partir do diretório *AI*, para que o programa consiga ler adequadamente os arquivos da pasta *results*. Além disso, o programa possui um argumento de ajuda para explicar o uso dos demais argumentos e um argumento de versão.

Diferente do programa de avaliação automática de modelos de treinamento, este programa não possui configurações a serem alteradas no corpo do programa porque todas as opções de configuração são fornecidas por argumentos de chamada e repassadas para o construtor de classe utilizado e para a chamada de método realizada (a lógica do programa em si se resume a exatamente uma construção de classe e uma chamada de método devido a modularização utilizada).

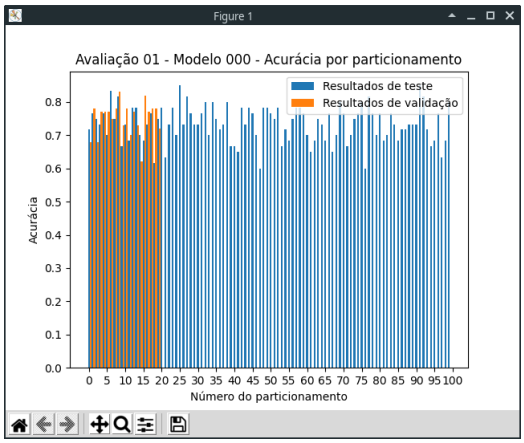
Após o programa ser chamado, a classe construída utiliza um extrator de dados feito pelo autor para ler os dados do arquivo de resultados selecionado e gera, por meio da biblioteca *matplotlib*, o gráfico desejado contendo os resultados lidos. Se necessário, esse gráfico então é salvo em um arquivo cujo nome é fornecido pelo usuário. Se o usuário não especificar o nome de arquivo desejado, o nome de arquivo *plot_output.png* é utilizado. Caso o programa não consiga salvar o gráfico gerado com o nome de arquivo fornecido, o programa ainda tenta realizar uma segunda gravação, no diretório de onde o programa foi chamado, em um arquivo de *fallback* de nome *fallback_plot_output.png*.

Para exemplificar o funcionamento do programa, fornecemos a imagem 3.2, a qual contém o gráfico de barras 3.2(a), o gráfico de caixa 3.2(b) e o histograma 3.2(c) gerados com os resultados de validação e de teste do melhor modelo (rankeamento 0) da avaliação número 1.

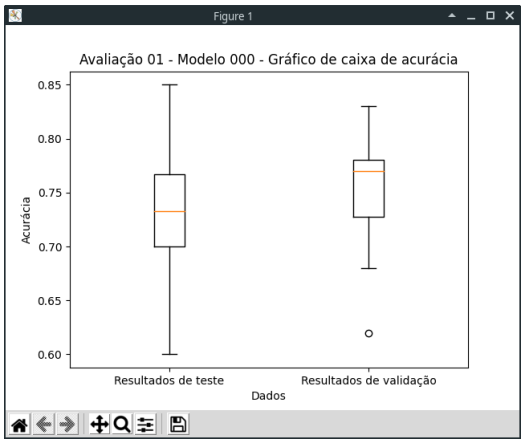
3.3.3 Programa para salvar um modelo pronto para realizar previsões

O programa para salvar um modelo pronto para realizar previsões de sobrevivência à insuficiência cardíaca foi escrito no arquivo *train_model.py* e permite que o usuário defina um modelo de treinamento para ser treinado sobre todo o conjunto de dados [Larxel 2020] e, posteriormente, exportado para um arquivo. O usuário pode passar como argumento na chamada do programa o nome do arquivo que receberá o modelo de treinamento. Além disso, o programa possui um argumento de ajuda para explicar o uso dos demais argumentos e um argumento de versão.

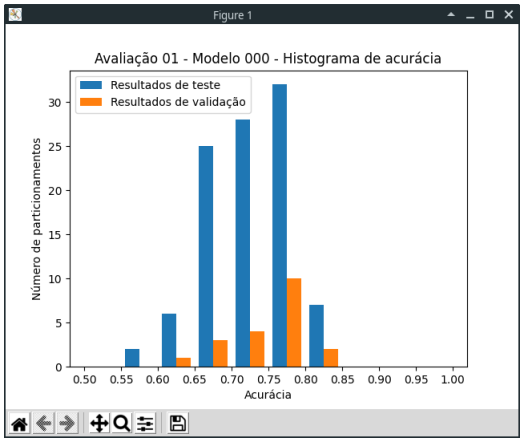
Este programa possui configurações a serem alteradas no corpo do programa como o tipo de modelo de treinamento utilizado, os hiperparâmetros do modelo de treinamento, o



(a) Gráfico de barras



(b) Gráfico de caixa



(c) Histograma

Figura 3.2: Exemplos de gráficos de resultados gerados pela execução do programa *plot_results.py*.

conjunto de dados utilizado para treinamento e as variáveis utilizadas do conjunto de dados.

Caso o usuário não especifique o nome do arquivo a ser utilizado para salvar o modelo, o nome de arquivo *prediction_model* é utilizado. Caso o programa não consiga salvar o modelo gerado com o nome de arquivo fornecido, o programa ainda tenta realizar uma segunda gravação, no diretório de onde o programa foi chamado, em um arquivo de *fallback* de nome *fallback_prediction_model*.

A extensão de arquivo utilizada para os arquivos que recebem os modelo de treinamento varia conforme o tipo de modelo, sendo *.rf.sav* para modelos do tipo floresta aleatória e *.h5* para modelos do tipo perceptron multicamada.

3.4 Implementação do website auxiliar

Após a avaliação dos modelos de treinamento e a análise dos resultados das avaliações, foi implementado um website auxiliar para permitir que os profissionais de saúde realizassem

previsões de sobrevivência à insuficiência cardíaca com o melhor modelo de treinamento obtido nas avaliações realizadas. Esse website tem como *backend* uma *API* escrita em NodeJS e como *frontend* uma interface de usuário escrita com React.

Para utilizar o website, o profissional de saúde precisará se cadastrar, fornecendo seu nome completo, um endereço válido de e-mail e uma senha para a sua conta. Após se cadastrar, o profissional de saúde poderá entrar em sua conta no website por meio do endereço de e-mail e pela senha utilizados no cadastro. Um profissional de saúde poderá cadastrar pacientes, requisitar previsões de sobrevivência à insuficiência cardíaca para um paciente cadastrado, modificar as informações dos pacientes cadastrados e excluir pacientes ou previsões realizadas.

Para gerar os resultados das previsões de sobrevivência à insuficiência cardíaca requisitadas que ainda não foram processadas, o *backend* do website utiliza um *job*, que é uma função cuja execução ocorre periodicamente conforme uma regra de recorrência. A previsão será feita com um modelo de treinamento armazenado no próprio *backend* do website e com um programa simples em *Python* que exporta uma função para fornecer acesso ao modelo de previsão. Isso permitirá que o sistema não se sobrecarregue tendo que realizar todas as previsões instantaneamente e possa agendar o processamento de um grupo significativo de previsões para ocorrer simultaneamente.

No próximo capítulo, abordaremos os resultados obtidos neste trabalho.

Capítulo 4

Resultados

Neste capítulo, os resultados obtidos neste trabalho são abordados com o intuito de explicá-los ao leitor e de realizar uma breve análise englobando as expectativas iniciais do autor e possíveis consequências dos resultados obtidos.

4.1 Avaliação de modelos de treinamento

Esta seção tem como objetivos descrever brevemente as avaliações realizadas, identificar os melhores modelos de treinamento dentre os modelos avaliados e analisar os resultados obtidos pelos melhores modelos de treinamento. Acreditamos que, dessa forma, o leitor será primeiro provido das informações relevantes das avaliações realizadas para apenas então ser apresentado aos resultados e análises baseados nessas avaliações.

4.1.1 Descrição das avaliações realizadas

No total, foram realizadas 30 avaliações de modelos de treinamento de aprendizado de máquina, 18 para modelos do tipo perceptron multicamada e 12 para modelos do tipo floresta aleatória. Estas avaliações permitiram que um total de 5346 modelos de treinamento fossem avaliados segundo o método de avaliação proposto, sendo 90 destes modelos do tipo perceptron multicamada e 5256 destes modelos do tipo floresta aleatória. Por fim, podemos observar que, dentre os modelos testados, um total de 18 modelos do tipo perceptron multicamada e 1053 modelos do tipo floresta aleatória foram classificados entre os melhores 20% de todos os modelos de sua avaliação no quesito melhor média de acurácia para os dados de validação e, assim, foram testados em todos os 100 subconjuntos de teste disponíveis.

Podemos observar na Tabela 4.1 uma breve descrição de cada avaliação realizada para modelos do tipo perceptron multicamada e na Tabela 4.2 uma descrição análoga de cada avaliação realizada para modelos do tipo floresta aleatória. Estas descrições abordam os números das avaliações, o número de modelos avaliados, e as variações de hiperparâmetros

empregadas. Para conveniência do leitor e concisão deste texto, as descrições de avaliações feitas para diferentes tipos de modelos de treinamento foram separadas em diferentes tabelas e algumas avaliações foram agrupadas na mesma descrição devido à sua semelhança nas variações de hiperparâmetros. As avaliações realizadas foram estruturadas de modo a testar uma grande quantidade de variações de hiperparâmetros para cada tipo de modelo com o intuito de determinar o conjunto de hiperparâmetros mais apto para uso em uma aplicação prática no website auxiliar. Em certos casos, os resultados de uma avaliação influenciaram diretamente as variações de hiperparâmetros das avaliações subsequentes.

Avalições	**Número total de modelos**	**Variações de hiperparâmetros**
03	5	Perceptron multicamada com 1 camada de entrada, 1 camada escondida e 1 camada de saída. Variação do tamanho da camada escondida no intervalo $[100..500]$ com deslocamentos de 100 em 100. Máximo de épocas de treinamento: 15 mil.
04, 05, 08, 09, 10	25	Perceptron multicamada com 1 camada de entrada, 2 camadas escondidas e 1 camada de saída. Variação do tamanho da primeira camada escondida no intervalo $[100..500]$ com deslocamentos de 100 em 100 e do tamanho da segunda camada escondida no intervalo $[20..100]$ com deslocamentos de 20 em 20. Máximo de épocas de treinamento: 15 mil.
19	5	Perceptron multicamada com 1 camada de entrada, 1 camada escondida, 1 camada com *dropout* com taxa de 0.2 e 1 camada de saída. Variação do tamanho da camada escondida no intervalo $[100..500]$ com deslocamentos de 100 em 100. Máximo de épocas de treinamento: 20 mil.
20 - 24	25	Perceptron multicamada com 1 camada de entrada, 2 camadas escondidas, 2 camadas com *dropout* com taxa de 0.2 (intercaladas na sequência camada escondida seguida de camada com *dropout*) e 1 camada de saída. Variação do tamanho da primeira camada escondida no intervalo $[100..500]$ com deslocamentos de 100 em 100 e do tamanho da segunda camada escondida no intervalo $[20..100]$ com deslocamentos de 20 em 20. Máximo de épocas de treinamento: 20 mil.
25	5	Perceptron multicamada com 1 camada de entrada, 1 camada escondida com regularização de *kernel* L2 com taxa 0.01 e 1 camada de saída. Variação do tamanho da camada escondida no intervalo $[100..500]$ com deslocamentos de 100 em 100. Máximo de épocas de treinamento: 20 mil.
26 - 30	25	Perceptron multicamada com 1 camada de entrada, 2 camadas escondidas com regularização de *kernel* L2 com taxa 0.01 e 1 camada de saída. Variação do tamanho da primeira camada escondida no intervalo $[100..500]$ com deslocamentos de 100 em 100 e do tamanho da segunda camada escondida no intervalo $[20..100]$ com deslocamentos de 20 em 20. Máximo de épocas de treinamento: 20 mil.

Tabela 4.1: Breve descrição das avaliações realizadas para modelos de treinamento do tipo perceptron multicamada.

Como já explicado no capítulo 3, a discrepância entre o número de modelos por avaliação e o número de avaliações em si para os modelos do tipo floresta aleatória e do tipo perceptron multicamada ocorre pois as avaliações feitas pelo autor com modelos do tipo perceptron multicamada foram consideravelmente lentas.

Avalições	Número total de modelos	Variações de hiperparâmetros
01	30	Variação do número de estimadores no intervalo [10..300] com deslocamentos de 10 em 10. Critério de divisão gini. Raiz quadrada do total de características como número máximo de características analisadas em uma divisão.
02	60	Variação do número de estimadores no intervalo [190..230] com deslocamentos de 5 em 5. Critérios de divisão: gini e entropia. Número máximo de características analisadas em uma divisão: raiz quadrada do total de características, logaritmo de base 2 do total de características e ausência de número máximo de características analisadas.
06	90	210 estimadores. Critério de divisão gini. Raiz quadrada do total de características como número máximo de características analisadas em uma divisão. Variação do mínimo de amostras para gerar uma folha no intervalo [1..10] e do mínimo de amostras para dividir um nó interno no intervalo [2..10].
07	450	Variação do número de estimadores no intervalo [200..220] com deslocamentos de 5 em 5. Critérios de divisão: gini e entropia. Número máximo de características analisadas em uma divisão: raiz quadrada do total de características, logaritmo de base 2 do total de características e ausência de número máximo de características analisadas. Variação do mínimo de amostras para gerar uma folha no intervalo [1..3] e do mínimo de amostras para dividir um nó interno no intervalo [5..9].
11	66	210 estimadores. Critérios de divisão: gini e entropia. Número máximo de características analisadas em uma divisão: raiz quadrada do total de características, logaritmo de base 2 do total de características e ausência de número máximo de características analisadas. Variação da profundidade máxima das árvores no intervalo [2..12].
12, 13, 14	2250	Variação do número de estimadores no intervalo [200..220] com deslocamentos de 5 em 5. Critérios de divisão: gini e entropia. Número máximo de características analisadas em uma divisão: raiz quadrada do total de características, logaritmo de base 2 do total de características e ausência de número máximo de características analisadas. Variação do mínimo de amostras para gerar uma folha no intervalo [1..3]; do mínimo de amostras para dividir um nó interno no intervalo [5..9]; e da profundidade máxima das árvores no intervalo [2..6].
15	66	210 estimadores. Critérios de divisão: gini e entropia. Número máximo de características analisadas em uma divisão: raiz quadrada do total de características, logaritmo de base 2 do total de características e ausência de número máximo de características analisadas. Variação do número máximo de folhas das árvores no intervalo [4..24] com deslocamentos de 2 em 2.
16, 17, 18	2250	Variação do número de estimadores no intervalo [200..220] com deslocamentos de 5 em 5. Critérios de divisão: gini e entropia. Número máximo de características analisadas em uma divisão: raiz quadrada do total de características, logaritmo de base 2 do total de características e ausência de número máximo de características analisadas. Variação do mínimo de amostras para gerar uma folha no intervalo [1..3]; do mínimo de amostras para dividir um nó interno no intervalo [5..9]; e do número máximo de folhas das árvores no intervalo [16..20].

Tabela 4.2: Breve descrição das avaliações realizadas para modelos de treinamento do tipo floresta aleatória.

4.1.2 Melhores resultados

Dentre os 18 modelos do tipo perceptron multicamada e 1053 modelos do tipo floresta aleatória testados nos 100 subconjuntos de teste disponíveis, podemos observar os resultados dos 3 melhores modelos do tipo perceptron multicamada e dos 5 melhores modelos do tipo floresta aleatória na Tabela 4.3, na qual os modelos foram rankeados pelo critério de melhor

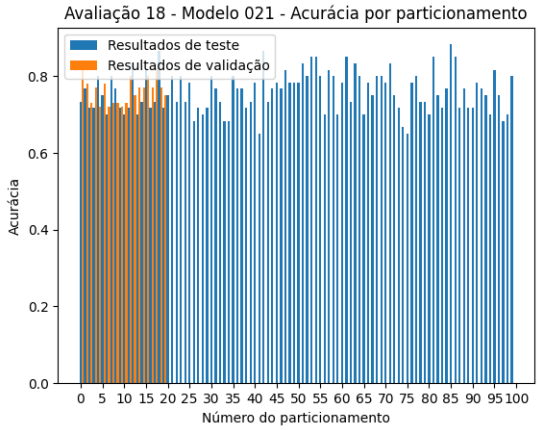
média de acurácia para os dados de teste.

Ranking	Tipo de modelo	Identificação do modelo	Médias de acurácia	Descrição do modelo
1º	Floresta aleatória	Número da avaliação: 18 Número do modelo: 21	Teste: 0.760 Validação: 0.762	215 estimadores. Critério de divisão gini. Ausência de número máximo de características analisadas em uma divisão. Mínimo de 3 amostras para gerar uma folha. Mínimo de 8 amostras para dividir um nó interno. Máximo de 19 folhas por árvore.
2º	Floresta aleatória	Número da avaliação: 18 Número do modelo: 58	Teste: 0.760 Validação: 0.759	220 estimadores. Critério de divisão gini. Ausência de número máximo de características analisadas em uma divisão. Mínimo de 2 amostras para gerar uma folha. Mínimo de 8 amostras para dividir um nó interno. Máximo de 19 folhas por árvore.
3º	Floresta aleatória	Número da avaliação: 18 Número do modelo: 25	Teste: 0.759 Validação: 0.761	210 estimadores. Critério de divisão gini. Ausência de número máximo de características analisadas em uma divisão. Mínimo de 3 amostras para gerar uma folha. Mínimo de 9 amostras para dividir um nó interno. Máximo de 20 folhas por árvore.
4º	Floresta aleatória	Número da avaliação: 18 Número do modelo: 27	Teste: 0.759 Validação: 0.761	205 estimadores. Critério de divisão gini. Ausência de número máximo de características analisadas em uma divisão. Mínimo de 3 amostras para gerar uma folha. Mínimo de 9 amostras para dividir um nó interno. Máximo de 20 folhas por árvore.
5º	Floresta aleatória	Número da avaliação: 18 Número do modelo: 49	Teste: 0.759 Validação: 0.760	200 estimadores. Critério de divisão gini. Ausência de número máximo de características analisadas em uma divisão. Mínimo de 3 amostras para gerar uma folha. Mínimo de 9 amostras para dividir um nó interno. Máximo de 17 folhas por árvore.
6º	Perceptron multicamada	Número da avaliação: 09 Número do modelo: 00	Teste: 0.707 Validação: 0.694	1 camada de entrada, 1 camada escondida de 400 unidades, 1 camada escondida de 20 unidades e 1 camada de saída. Máximo de épocas de treinamento: 15 mil.
7º	Perceptron multicamada	Número da avaliação: 24 Número do modelo: 00	Teste: 0.704 Validação: 0.716	1 camada de entrada, 1 camada escondida de 500 unidades, 1 camada com *dropout* com taxa de 0.2, 1 camada escondida de 100 unidades, 1 camada com *dropout* com taxa de 0.2 e 1 camada de saída. Máximo de épocas de treinamento: 20 mil.
8º	Perceptron multicamada	Número da avaliação: 30 Número do modelo: 00	Teste: 0.700 Validação: 0.704	1 camada de entrada, 1 camada escondida de 500 unidades com regularização de *kernel* L2 com taxa 0.01, 1 camada escondida de 100 unidades com regularização de *kernel* L2 com taxa 0.01 e 1 camada de saída. Máximo de épocas de treinamento: 20 mil.

Tabela 4.3: Ranking dos 3 melhores modelos do tipo perceptron multicamada e dos 5 melhores modelos do tipo floresta aleatória segundo a média de acurácia para dados de teste.

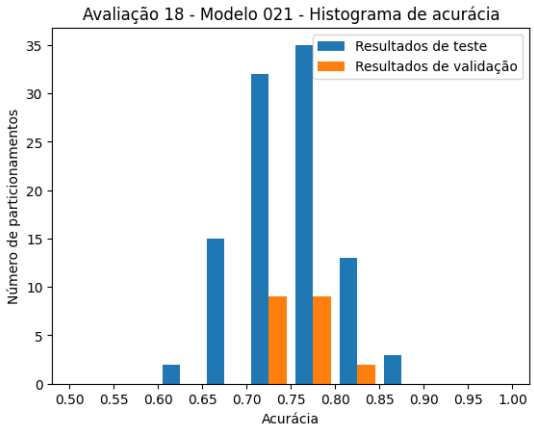
Para permitir ao leitor analisar com maior profundidade os resultados obtidos pelo modelo de treinamento que será utilizado para a previsão de dados no website, os gráficos de

resultados para os subconjuntos de validação e teste do melhor modelo do rankeamento apresentado na Tabela 4.3 foram disponibilizados na imagem 4.1.



(a) Gráfico de barras

(b) Gráfico de caixa



(c) Histograma

Figura 4.1: Gráficos de resultados para o melhor modelo do rankeamento.

Com as informações apresentadas na Tabela 4.3, poderemos realizar uma análise entre os diferentes tipos de modelos de treinamento e em relação à escolha de hiperparâmetros de cada tipo de modelo de treinamento. Também poderemos comparar os resultados obtidos neste trabalho com os resultados obtidos no artigo de Chicco e Jurman [Chicco and Jurman 2020], o qual utilizou o mesmo conjunto de dados [Larxel 2020] para avaliação de modelos de aprendizado de máquina.

Como observado na Tabela 4.3, podemos notar que os 5 melhores modelos de florestas aleatórias tiveram um desempenho superior em relação aos 3 melhores modelos do tipo perceptron multicamada. Embora isso possa ter sido exacerbado pela discrepância entre o número de modelos avaliados do tipo floresta aleatória e do tipo perceptron multicamada, esse resultado já era esperado antes da realização das avaliações. Isso porque o artigo de Chicco e Jurman também havia verificado um melhor desempenho dos modelos de treinamento do tipo floresta aleatória em relação aos modelos de treinamento do tipo perceptron multicamada no conjunto de dados utilizado, corroborando os resultados obtidos neste tra-

balho.

Analisando os hiperparâmetros dos modelos do tipo floresta aleatória, podemos realizar algumas constatações. Primeiramente, podemos notar que o número de estimadores utilizados não parece ter sido uma característica que influenciou significativamente os resultados, pois todos os 5 valores de número de estimadores utilizados na avaliação 18 produziram um modelo no ranking de melhores modelos. Além disso, podemos também constatar que a utilização do coeficiente de gini se mostrou superior ao uso da entropia como critério de divisão. Também observamos que a ausência de um número máximo de características consideradas por divisão superou o uso de um número máximo de características consideradas por divisão sendo igual à raiz quadrada do total de características ou ao logaritmo de base 2 do total de características, o que contraria a teoria existente segundo a qual o uso da raiz quadrada do total de características como número máximo de características consideradas por divisão forneceria os melhores resultados [James *et al.* 2013, p.319-321]. Outro aspecto que podemos observar é que a utilização de técnicas para limitar o número de divisões das árvores (mínimo de amostras para gerar uma folha, mínimo de amostras para dividir um nó interno) se mostraram úteis na obtenção dos melhores resultados, pois todos eles fizeram uso de ambas. Por fim, notamos que tanto a limitação no número de folhas como a limitação da profundidade máxima das árvores produziram bons resultados (embora esta última não tenha sido utilizada por nenhum dos melhores resultados, seu uso levou a um aumento da acurácia em relação a árvores que não a utilizaram). As duas últimas observações estão de acordo com a teoria por trás do modelo de árvores de decisão segundo a qual árvores complexas levam a *overfitting* dos dados de treino, de forma que recursos para limitar a complexidade de uma árvore ou para extrair uma sub-árvore da árvore final podem diminuir o erro obtido sobre o subconjunto de dados de teste [James *et al.* 2013, p.307-311].

Analisando os hiperparâmetros dos modelos do tipo perceptron multicamada, também podemos realizar algumas constatações. Em primeiro lugar, notamos que todos os modelos de perceptrons multicamada presentes no ranking apresentam duas camadas escondidas, indicando que os modelos com duas camadas escondidas obtiveram melhor desempenho do que os modelos com apenas uma camada escondida. Também é possível observar que os modelos com maior número de unidades nas camadas escondidas forneceram um melhor desempenho, em especial se combinados com um maior número de épocas de treinamento e uma camada com *dropout* ou um *kernel* de regularização aplicado aos pesos das camadas escondidas, tendo-se em vista que os 2 últimos modelos do tipo perceptron multicamada presentes no ranking possuem a maior quantidade de unidades para ambas as camadas escondidas dentre todos os modelos avaliados. O fato de o melhor modelo do tipo perceptron multicamada ter sido um modelo com camadas escondidas de tamanho 400 e 20, que não utiliza *dropout* ou um *kernel* de regularização aplicado aos pesos das camadas escondidas e que teve um limite máximo de 15 mil épocas de treinamento é um pouco surpreendente, em especial se considerarmos a média de acurácia para os dados de validação obtida por esse modelo. Com os resultados obtidos até aqui, creio que essa última constatação não possa ser

explicada de maneira satisfatória, sendo necessária a realização de mais avaliações para se chegar a um resultado conclusivo.

Comparando os resultados obtidos neste trabalho com os resultados obtidos sobre o mesmo conjunto de dados no artigo de Chicco e Jurman, podemos constatar que as avaliações realizadas neste trabalho obtiveram um bom desempenho. No artigo de Chicco e Jurman, cujo método de avaliação utilizado está descrito no capítulo 3, os melhores resultados de acurácia de previsão para o subconjunto de teste foram de 74% e de 68%, respectivamente, para modelos do tipo floresta aleatória e perceptron multicamada que utilizaram 11 variáveis fornecidas pelo conjunto de dados como características. Em comparação, os melhores resultados de acurácia de previsão para o subconjunto de teste obtidos neste trabalho e apresentados na Tabela 4.3 foram de 76% e 70,7%, respectivamente, para modelos do tipo floresta aleatória e perceptron multicamada que utilizaram 10 variáveis fornecidas pelo conjunto de dados como características. Como os métodos de avaliação e as características de conjunto de dados utilizados neste trabalho e no artigo de Chicco e Jurman diferem, não podemos afirmar conclusivamente que este trabalho conseguiu melhorar os resultados existentes para esse conjunto de dados, mas podemos afirmar que os resultados obtidos neste trabalho se encaixam nas expectativas estabelecidas pelo artigo de Chicco e Jurman.

4.2 Website

O resultado a ser apresentado relativo ao website seria o website em si, o qual funcionou adequadamente, permitindo que um usuário criasse pacientes e requisitasse previsões de sobrevivência à insuficiência cardíaca para estes utilizando o modelo de floresta aleatória com a melhor acurácia de previsão. Porém, como não é possível apresentar todo o website e seu funcionamento neste documento, algumas páginas do website foram fornecidas com o intuito de ilustar a aplicação desenvolvida.

Na imagem 4.2 temos a página de login, que descreve o website ao usuário e permite que este realize login no website ou seja direcionado para a página de cadastro. Na imagem 4.3 temos a página principal do usuário, na qual os pacientes deste podem ser visualizados e gerenciados. Por fim, na imagem 4.4 temos a página de previsões de um paciente, onde podemos visualizar as informações de um paciente e suas previsões realizadas e excluir previsões indesejadas.

Caso o leitor deseje explorar o funcionamento do website mais a fundo ou entender detalhes de sua implementação, o código fonte tanto do *backend* como do *frontend* pode ser encontrado no repositório do *GitHub* deste trabalho[1].

No próximo capítulo, apresentaremos as conclusões que resultaram deste trabalho e propostas para trabalhos futuros.

[1] `https://github.com/AndreLaranjeira/ML-HeartFailureSurvival`

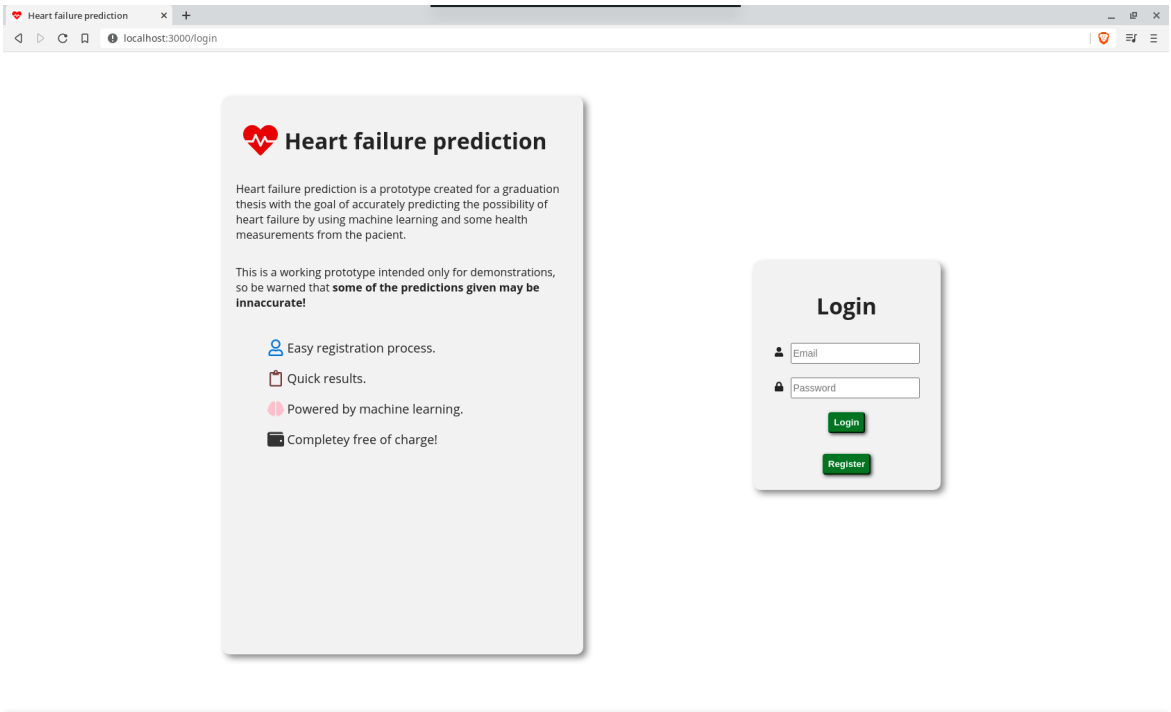


Figura 4.2: Página de login do website

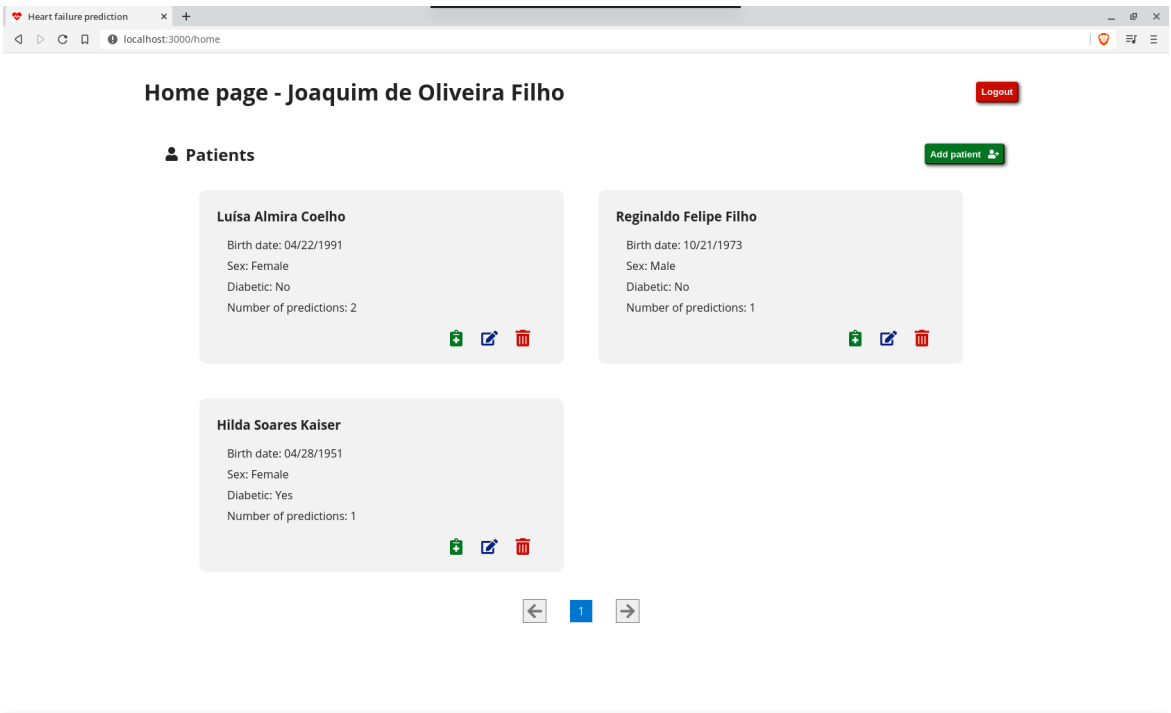


Figura 4.3: Página principal do usuário do website

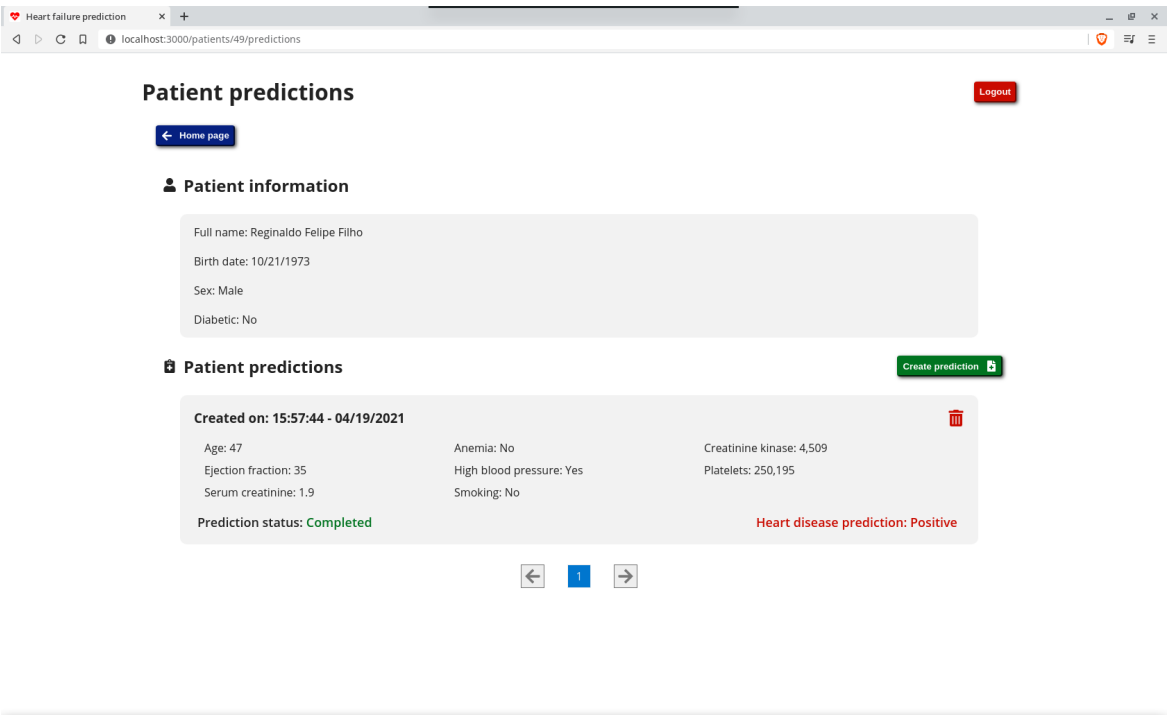


Figura 4.4: Página de previsões de um paciente do website

Capítulo 5

Conclusão

Com base na introdução feita no capítulo 1 acerca dos objetivos deste trabalho e com base nos resultados deste trabalho apresentados no capítulo 4, podemos concluir que este trabalho atingiu o objetivo proposto.

O estudo comparativo de modelos do tipo perceptron multicamada e floresta aleatória descrito nos capítulos 3 e 4 foi capaz de enunciar claras distinções na acurácia de previsão do modelo de treinamento geradas pelo tipo de modelo de aprendizado de máquina utilizado e pelos hiperparâmetros escolhidos. A melhor média de acurácia de previsão obtida sobre os 100 subconjuntos de teste por um modelo do tipo floresta aleatória com base em um treinamento feito sobre 10 características obtidas do conjunto de dados de sobrevivência à insuficiência cardíaca utilizado [Larxel 2020] foi de 76%, superando a melhor marca de 74% registrada no artigo de Chicco e Jurman [Chicco and Jurman 2020] para modelos do tipo floresta aleatória treinados com 11 características obtidas do conjunto de dados.

O website implementado funcionou adequadamente, conforme mencionado no capítulo 4. Embora uma aplicação que oferece uma previsão de sobrevivência à insuficiência cardíaca com acurácia de 76% não possa ser disponibilizada ao público por razões éticas e legais relacionadas ao campo da medicina, podemos afirmar que o website implementado exemplifica adequadamente como o aprendizado de máquinas pode ser utilizado em uma aplicação médica por profissionais de saúde para auxiliar a realização de suas funções.

5.1 Trabalhos futuros

Essa seção possui o intuito de abordar ideias para futuras pesquisas relacionadas a esse trabalho, incluindo suas motivações e impactos esperados.

5.1.1 Utilização de dados de pacientes contemporâneos

Este trabalho utilizou um conjunto de dados [Larxel 2020] para realizar o treinamento dos modelos de aprendizado de máquina, mas não foi possível utilizar dados de pacientes contemporâneos para treinar os modelos de aprendizado ou para testar a acurácia do modelo empregado no website auxiliar. A utilização de dados de pacientes não cadastrados no conjunto de dados seria interessante tanto para mitigar algumas das limitações do conjunto de dados em questão, as quais são abordadas no capítulo 3, quanto para verificar a acurácia do modelo em um caso contemporâneo.

A obtenção dos dados em questão de um paciente qualquer não é necessariamente trabalhosa para um profissional médico, mas contém uma gama de implicações legais e éticas que devem ser discutidas com o paciente, o que poderia ser trabalhado em uma pesquisa futura. Espera-se que a utilização de dados de pacientes contemporâneos torne o aprendizado dos modelos mais robusto, aumentado a acurácia obtida para os subconjuntos de teste e tornando mais viável o uso do website em casos cotidianos.

5.1.2 Avaliação de mais tipos de modelos de treinamento e variações de hiperparâmetros

Este trabalho avaliou modelos de treinamento do tipo floresta aleatória e perceptron multicamada, utilizando 5346 combinações diferentes de hiperparâmetros para a realização de 30 avaliações envolvendo ambos os tipos de modelos conforme o método descrito no capítulo 3. Entretanto, ainda existem outros tipos de modelos de treinamento que foram mencionados no artigo científico de Chicco e Jurman [Chicco and Jurman 2020] que estudou o conjunto de dados em questão e outras variações de hiperparâmetros para os tipos de modelos já avaliados neste trabalho.

O artigo em questão aponta os modelos do tipo árvore de decisão, *extreme gradient boosting* e regressão linear como modelos que fornecem boas acurácias de previsão para o conjunto de dados em questão. E, para modelos do tipo floresta aleatória e perceptron multicamada, podemos citar, respectivamente, a exploração do hiperparâmetro de valor mínimo de decréscimo de impureza por divisão e redes neurais com 3 camadas escondidas ou que utilizem *dropout* e *kernels* de regularaçõ *simultaneamente* como variações interessantes de hiperparâmetros que ainda podem ser exploradas. Com a utilização de um número maior de tipos de modelos de treinamento e de mais variações de hiperparâmetros para estes modelos de treinamento, podemos, possivelmente, melhorar os resultados de acurácia obtidos na previsão de sobrevivência à insuficiência cardíaca.

5.1.3 Criação de um aplicativo móvel para o projeto

Como mencionado no capítulo 3, o website auxiliar foi desenvolvido com um *backend* sendo uma *API*, de forma que um novo *frontend* pode ser facilmente desenvolvido para interagir com o banco de dados aproveitando o *backend* existente. Uma sugestão de *frontend* a ser desenvolvido é um aplicativo móvel que possa ser utilizado em *smartphones*.

O uso de *smartphones* nos dias atuais é uma alternativa extremamente popular ao uso de computadores e *notebooks* por sua versatilidade, facilidade de uso e maiores opções de quando e onde um serviço pode ser utilizado. Dessa forma, seria benéfico ao usuário do website auxiliar poder utilizar o serviço de previsão de sobrevivência à insuficiência cardíaca por meio de um aplicativo em seu *smartphone*.

Referências Bibliográficas

[Adler *et al.* 2020] Adler, E. D., Voors, A. A., Klein, L., Macheret, F., Braun, O. O., Urey, M. A., Zhu, W., Sama, I., Tadel, M., Campagnari, C., Greenberg, B., and Yagil, A. (2020). Improving risk prediction in heart failure using machine learning. *European Journal of Heart Failure*, 22(1).

[Ahmad *et al.* 2017] Ahmad, T., Munir, A., Bhatti, S. H., Aftab, M., and Raza, M. A. (2017). Survival analysis of heart failure patients: A case study. *PLoS ONE*, 12(7).

[American Heart Association, Inc. 2017] American Heart Association, Inc. (2017). What is Heart Failure? "https://www.heart.org/en/health-topics/heart-failure/what-is-heart-failure". Definição da condição clínica de insuficiência cardíaca; Acessado em 25 de Maio de 2021.

[Awan *et al.* 2018] Awan, S. E., Sohel, F., Sanfilippo, F. M., Bennamoun, M., and Dwivedi, G. (2018). Machine learning in heart failure. *Current Opinion in Cardiology*, 33(2).

[Chicco and Jurman 2020] Chicco, D. and Jurman, G. (2020). Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC Medical Informatics and Decision Making*, 20(16).

[Comitê Coordenador da Diretriz de Insuficiência Cardíaca 2018] Comitê Coordenador da Diretriz de Insuficiência Cardíaca (2018). Diretriz Brasileira de Insuficiência Cardíaca Crônica e Aguda. *Arquivos Brasileiros de Cardiologia*, 111(3).

[Gaui *et al.* 2014] Gaui, E., Oliveira, G., and Klein, C. (2014). Mortalidade por Insuficiência Cardíaca e Doença Isquêmica do Coração no Brasil de 1996 a 2011. *Arquivos Brasileiros de Cardiologia*, 102(6).

[James *et al.* 2013] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer, New York, NY.

[Larxel 2020] Larxel (2020). Heart Failure Prediction. "https://www.kaggle.com/andrewmvd/heart-failure-clinical-data". Conjunto de dados disponibilizado com licença CC BY 4.0; Acessado em 28 de Agosto de 2020.

[Mitchell 1997] Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

[Nogueira *et al.* 2019] Nogueira, I. D. B., Nogueira, P. A. M. S., Fonseca, A. M. C., Santos, T. Z. M., Souza, D. E., and Ferreira, G. M. H. (2019). Prevalência de insuficiência cardíaca e associação com saúde autorreferida no Brasil: Pesquisa Nacional de Saúde - 2013. *Acta Fisiátrica*, 26(2).

[Tabassian *et al.* 2018] Tabassian, M. *et al.* (2018). Diagnosis of Heart Failure With Preserved Ejection Fraction: Machine Learning of Spatiotemporal Variations in Left Ventricular Deformation. *Journal of the American Society of Echocardiography*, 31(12).

[Virani *et al.* 2021] Virani, S. S. *et al.* (2021). Heart Disease and Stroke Statistics-2021 Update: A Report From the American Heart Association. *Circulation*, 143(8).