Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Predição da mortalidade em Unidades de Terapia Intensiva utilizando Redes Neurais Artificiais

Filipi Teles da Silva

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Pedro de Azevedo Berger

Brasília
2022

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Engenharia da Computação

Coordenador: Prof. Dr. João José Costa Gondim

Banca examinadora composta por:

Prof. Dr. Pedro de Azevedo Berger (Orientador) — CIC/UnB
Prof. Dr. Ricardo Pezzuol Jacobi — CIC/UnB
Prof. Dr. Bruno Luiggi Macchiavello Espinoza — CIC/UnB

CIP — Catalogação Internacional na Publicação

Silva, Filipi Teles da.

Predição da mortalidade em Unidades de Terapia Intensiva utilizando Redes Neurais Artificiais / Filipi Teles da Silva. Brasília : UnB, 2022. 57 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2022.

1. Redes Neurais Artificiais, 2. *Deep Learning*, 3. MIMIC III,
4. reconhecimento de padrões, 5. predição de mortalidade, 6. Unidade de Terapia Intensiva

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Predição da mortalidade em Unidades de Terapia Intensiva utilizando Redes Neurais Artificiais

Filipi Teles da Silva

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Pedro de Azevedo Berger (Orientador)
CIC/UnB

Prof. Dr. Ricardo Pezzuol Jacobi
CIC/UnB

Prof. Dr. Bruno Luiggi Macchiavello Espinoza
CIC/UnB

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 22 de abril de 2022

Dedicatória

Dedico este projeto aos meus pais, Milson e Heroilda, que sempre me apoiaram, a minha namorada Larissa Martins por acreditar em meu potencial, ao meu irmão Rodrigo e aos demais familiares e amigos que me apoiaram na minha longa jornada acadêmica.

Agradecimentos

Agradeço ao meu orientador, Professor Dr. Pedro de Azevedo Berger, pelo apoio ao longo do projeto, sempre disposto a tirar minhas dúvidas, e pela idéia do projeto, que foi uma grande oportunidade de aprendizado.

Resumo

Pacientes com quadro médico grave são admitidos na Unidade de Terapia Intensiva (UTI) para que seus sinais vitais e exames médicos sejam monitorados frequentemente, a fim de evitar que seu quadro evolua para óbito. No entanto, são muitos os dados monitorados, dificultando a análise e o prognóstico. Com o desenvolvimento das redes neurais artificiais, um subconjunto da Inteligência Artificial, tornou-se possível a utilização de algoritmos em vários campos, capazes de identificar padrões em grandes conjuntos de dados e auxiliar na tomada de decisões. Esse trabalho tem como objetivo a criação de uma rede neural artificial para ser utilizada no campo da medicina, para a predição da mortalidade de pacientes na UTI. Para isso, utilizou-se o banco de dados Medical Information Mart for Intensive Care (MIMIC-III) [1] para extrair os sinais vitais e exame laboratoriais dos pacientes nas primeiras 24 horas após a internação e alimentar um modelo de rede neural artificial para treinamento. O projeto consiste em quatro etapas principais: extração dos dados do MIMIC-III, pré-processamento e adequação dos dados, treinamento da rede neural artificial para a escolha do melhor modelo e avaliação do modelo. O método utilizado obteve 77% de acurácia e Area Under the Receiver Operating Characteristic Curve (AUROC) de 0.86.

Palavras-chave: Redes Neurais Artificiais, *Deep Learning*, MIMIC III, reconhecimento de padrões, predição de mortalidade, Unidade de Terapia Intensiva

Abstract

Patients with a serious medical condition are admitted to the Intensive Care Unit (ICU) so that their vital signs and medical exams are monitored frequently, in order to prevent their condition from progressing to death. However, there is a lot of monitored data, making analysis and prognosis difficult. With the development of Neural Networks, a subset of Artificial Intelligence, it became possible to use algorithms in various fields, capable of identifying patterns in large data sets and assisting in decision making. This project aims to create an Artificial Neural Network to be used in the field of medicine, to predict patients mortality inside ICU. For this, Medical Information Mart for Intensive Care (MIMIC-III) [1] database was used to extract the vital signals and blood tests of patients 24 hours after ICU admission and feed a Artificial Neural Network model for training. The project consists of four steps: data extraction from MIMIC-III, preprocessing and data adequacy, training of the Artificial Neural Network for the best model choice and model evaluation. The proposed methods achieved 77% accuracy and Area Under the Receiver Operating Characteristic Curve (AUROC) of 0.86.

Keywords: Artificial Neural Networks, Deep Learning, MIMIC III, pattern recognition, mortality prediction, Intensive Care Unit

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

APACHE The Acute Physiology and Chronic Health Evaluation. 1

AUROC Area Under the Receiver Operating Characteristic Curve. vi, vii, 15, 31, 36, 38, 39

BUN Blood Urea Nitrogen. 22

CID Classificação Estatística Internacional de Doenças e Problemas Relacionados à Saúde. 20

ECG Eletrocardiograma. 19

GD Gradiente Descendente. 6–8, 32

HIPAA Health Insurance Portability and Accountability Act. 21

ICU Intensive Care Unit. vii

MIMIC-III Medical Information Mart for Intensive Care. vi, vii, 2, 17, 18, 20, 22, 25, 30, 38–40

NaN Not a Number. 26

ROC Receiver Operating Characteristic. x, 15

SAPS The Simplified Acute Physiology Score. 1

SMOTE Synthetic Minority Over-sampling Technique. 31

SOFA Sequential Organ Failure Assessment. 1

UTI Unidade de Terapia Intensiva. vi, viii, 1, 2, 17–19, 23, 24, 26, 27, 30, 32, 38, 40, 41

WBC White Blood Cell Count. 22

Capítulo 1

Introdução

Atualmente, é notável a crescente utilização de Inteligência Artificial em diversos campos, como reconhecimento de fala [2] e processamento de linguagem natural [3], sobretudo com o desenvolvimento do conceito de *Deep Learning*. Redes Neurais Artificiais foram propostas primeiramente em 1944, por Warren McCullough e Walter Pitts [4]. Posteriormente, outros trabalhos foram aprimorando esse conceito, permitindo as inúmeras aplicações que vemos hoje em dia.

Na medicina, cada vez mais *Deep Learning* vem sendo utilizado, seja para a análise de imagens médicas [5] ou para auxiliar em diagnósticos de diversas doenças [6].

A Unidade de Terapia Intensiva (UTI) ganhou um grande destaque desde a pandemia de covid-19 [7]. Sistemas de saúde de vários países colapsaram pois o número de pacientes era muito superior ao número de leitos disponíveis. Além disso, o número de profissionais disponíveis não era o suficiente para realizar o correto atendimento e análise dos sinais dos pacientes. Segundo o Conselho Federal de Medicina [8], UTI é o local dentro dos hospitais com um sistema organizado para oferecer suporte vital de alta complexidade, com diversas modalidades de monitorização das funções corporais essenciais para a vida.

Dentro de uma UTI, os níveis de atenção que um paciente necessita também podem variar:

- tipo II — atendem a pacientes que necessitam de nível de atenção alto
- tipo III — atendem a pacientes que necessitam de nível de atenção muito alto. As UTIs devem apresentar leitos com equipamentos e médicos capazes de cuidar desses dois níveis de complexidade

Algumas ferramentas de escores foram desenvolvidas para avaliar a probabilidade de morte de pacientes na UTI. Dentre elas, The Simplified Acute Physiology Score (SAPS)II [9]; The Acute Physiology and Chronic Health Evaluation (APACHE) II, III e IV [10] [11] [12]; e Sequential Organ Failure Assessment (SOFA). Entretanto, a maioria desses escores

de gravidade foram desenvolvidos há décadas, então o desempenho diminuiu devido ao fato de que a população e os cenários clínicos mudaram ao longo do tempo.

O objetivo desse trabalho é utilizar redes neurais artificiais para realizar o predição de mortalidade na UTI, usando 66 variáveis preditoras, dentre elas sinais vitais, exames laboratoriais, idade e gênero.

1.1 Objetivo geral

O objetivo geral do trabalho é o desenvolvimento de um modelo preditivo classificatório utilizando redes neurais artificias para predizer a morte de pacientes em UTI's. Após o desenvolvimento, seu desempenho foi analisado através de métricas para avaliar sua capacidade de generalização em novos dados.

1.2 Objetivos específicos

Com o propósito de organizar o trabalho, são enumerados os objetivos específicos:

1. Selecionar e extrair dados do MIMIC-III utilizando o Google BigQuery
2. Analisar os dados extraídos usando as bibliotecas pandas, numpy e matplotlib
3. Realizar o pré-processamento dos dados utilizando scikit-learn
4. Desenvolver e treinar o modelo de rede neural artificial utilizando Tensorflow
5. Avaliar o desempenho do modelo escolhido

1.3 Organização do trabalho

O trabalho foi organizado da seguinte forma: no Capítulo 2 é apresentado o referencial teórico necessário à compreensão do problema. No Capítulo 3 é apresentado toda a metodologia, desde a extração dos dados até o desenvolvimento do modelo de rede neural artificial. Os resultados obtidos são discutidos no Capítulo 4. Finalmente, o Capítulo 5 apresenta as conclusões deste trabalho e proposta de trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo serão abordados os principais conceitos que serão explorados ao longo do trabalho.

2.1 Machine Learning

Segundo Arthur Samuel, Aprendizado de Máquina (*Machine Learning*) é o campo de estudo que dá aos computadores a habilidade de aprender a serem explicitamente programados [13]. É um subcampo da inteligência artificial, que é amplamente definido como a capacidade de uma máquina de imitar o comportamento humano inteligente. Os sistemas de inteligência artificial são usados para executar tarefas complexas de maneira semelhante à forma como os humanos resolvem problemas [14].

Em alto nível, o aprendizado de máquina é a capacidade de se adaptar a novos dados de forma independente e por meio de iterações. Os aplicativos aprendem com cálculos e transações anteriores e usam o "reconhecimento de padrões" para produzir resultados confiáveis.

2.1.1 Redes Neurais Artificiais

Uma rede neural artificial pode ser considerada como um modelo altamente simplificado da estrutura da rede neural biológica [15]. Há diversos tipos de redes neurais artificiais, como redes neurais convolucionais, redes neurais recorrentes e perceptron multicamadas. Nessa seção, o foco será no perceptron e perceptron multicamadas.

Perceptron

O perceptron é um modelo matemático de um neurônio biológico [16], que pode ser descrito conforme mostra a Figura 2.1

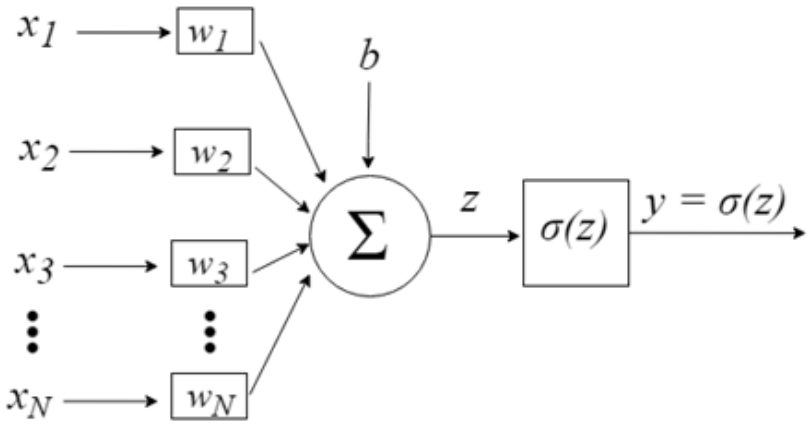

Figura 2.1: *Perceptron*

Normalmente, um perceptron consiste em duas funções matemáticas: uma função linear e uma função não linear, também chamada de função de ativação [17].

A função linear, ou componente linear do perceptron, denotada na Figura 2.1 como z, é descrita conforme a equação Equação 2.1

$$z = (x_1.w_1 + x_2.w_2 + ... + x_N.w_N) + b = \sum_{i=1}^{N} x_i w_i + b \tag{2.1}$$

.

- Os elementos do vetor $x = [x_1,\ x_2,\ x_3,\ ...,\ x_N]$ são as entradas, sendo que esses elementos podem ser dados brutos fornecidos ou saídas de outros perceptrons
- Os elementos do vetor $w = [w_1,\ w_2,\ w_3,\ ...,\ w_N]$ são os pesos que controlam a importância de cada entrada
- O termo b é o viés, também chamado de *bias*. Esse termo desloca a função de ativação de cada perceptron para não obter valor 0.

Os pesos e vieses são chamados de parâmetros em um modelo de rede neural.

Função de Ativação

O tópico aqui discutido foi separado para uma melhor organização, mas pode ser considerado uma continuação da seção do perceptron.

A função não linear do perceptron, também chamada de função de ativação, é denotada por σ, sendo aplicada na função linear z para obter a saída y. Há alguns motivos para se utilizar funções de ativação não lineares [18]:

- Fornecer não linearidade, sem a qual as redes neurais não podem modelar relacionamentos não lineares

- Limitar o valor a um certo intervalo, produzindo o valor final de saída y do neurônio
- Decidir como um neurônio é ativado

As funções de ativação usadas no trabalho foram: sigmoide e ReLU.

- **Função Sigmoide**

Na Equação 2.1, a notação $\sigma(.)$ é uma função sigmoide [19], que tem como saída um número entre 0 e 1 e é definida pela Equação 2.2

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

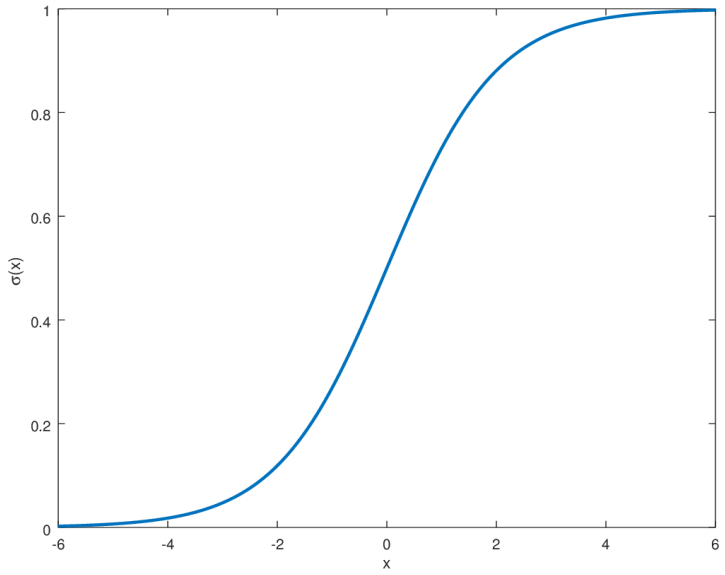

Figura 2.2: Função sigmoide

Essa função de ativação possui a vantagem de comprimir as entradas para um valor entre 0 e 1, tornando-a perfeita para modelar probabilidade [20].

Geralmente é usada em modelos de regressão logística e classificação binária na camada de saída.

- **ReLU**

Rectified Linear Unit [21], ou ReLU, é uma função de ativação descrita pela Equação 2.3

$$y = \begin{cases} 0, & \text{se } x \leq 0 \\ x, & \text{se } x > 0 \end{cases} \tag{2.3}$$

Seu gráfico é descrito na Figura 2.3

$$\boldsymbol{ReLU} = max(0, x)$$

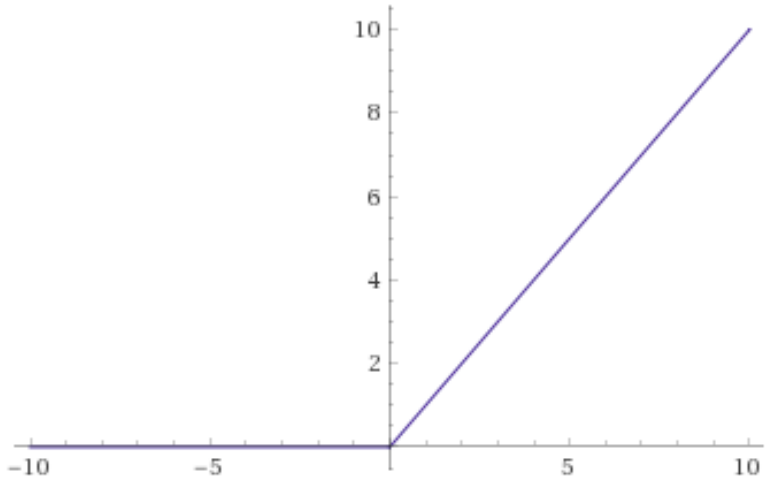

Figura 2.3: ReLU

Normalmente, a função de ativação ReLU é usada nas camadas ocultas, pois é um pouco mais rápida de calcular do que outras funções de ativação, e o Gradiente Descendente (GD) não fica preso tanto em platôs, graças ao fato de que a função não satura para grandes valores de entrada (ao contrário da função sigmoide ou da função tangente hiperbólica, que saturam em 1).

Essa função foi utilizada nesse projeto nas camadas ocultas, e a função sigmoide foi utilizada na camada de saída.

Perceptron Multicamadas

Normalmente, os problemas do mundo real são muito complexos, portanto um neurônio não é suficiente. Entretanto, é possível combiná-los em uma estrutura em camadas, onde cada camada pode ter um número diferente de neurônios, formando uma rede neural denominada *Multi-Layer Perceptron*, ou Perceptron Multicamadas [22].

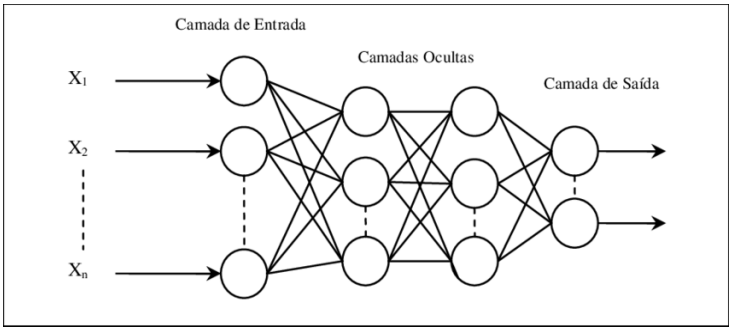

Figura 2.4: Perceptron Multicamadas

Treinamento do Perceptron Multicamadas

Normalmente, um modelo de rede neural é treinado usando o algoritmo de otimização de gradiente descendente estocástico e os pesos são atualizados usando o algoritmo de retropropagação de erro [23].

- **Função de custo**

No contexto de um algoritmo de otimização, a função usada para avaliar um modelo candidato (ou seja, um conjunto de pesos) é chamada de função objetivo. Normalmente, com redes neurais, procuramos minimizar o erro. Como tal, a função objetivo é muitas vezes referida como uma função de custo ou uma função de perda e o valor calculado pela função de perda é referido simplesmente como perda [23].

Com isso, a função de custo informa o quão bom seu modelo é nas previsões. Se as previsões do modelo estiverem mais próximas dos valores reais, o custo será mínimo e se as previsões estiverem totalmente distantes dos valores originais, o valor do custo será o máximo.

- **Gradiente Descendente**

O Gradiente Descendente (GD) [24], definido pela Equação 2.4, é um dos algoritmos mais populares para realizar a otimização e, de longe, a maneira mais comum de otimizar redes neurais. A ideia geral do GD é ajustar os parâmetros iterativamente para minimizar uma função de custo. O algoritmo mede o gradiente local da função de erro em relação ao vetor de parâmetro θ, e vai na direção do GD. Uma vez que o gradiente é zero, o mínimo é atingido.

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2.4}$$

O GD atualiza os pesos θ subtraindo diretamente o gradiente da função de custo $J(\theta)$ em relação aos pesos ($\nabla_\theta J(\theta)$) multiplicados pela taxa de aprendizagem η.

Pressupõe-se que a inicialização dos pesos (θ) seja feito com valores aleatórios, melhorando-os gradualmente, dando um pequeno passo de cada vez tentando diminuir a função de custo, até que o algoritmo convirja ao mínimo.

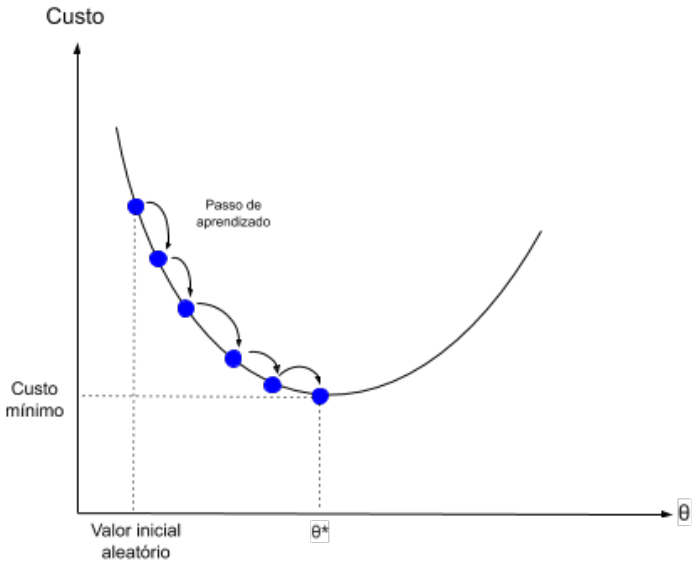

Figura 2.5: Gradiente Descendente

Um parâmetro muito importante no GD é o tamanho do passo de aprendizado, determinado pelo hiperparâmetro *learning rate* [25], ou taxa de aprendizagem. A definição de um valor muito alto para a taxa de aprendizagem pode levar o modelo a divergir, ou seja, o modelo não consegue chegar em seu melhor ajuste. Já quando a taxa de aprendizagem é definida com um valor muito baixo, o modelo demora mais tempo para chegar no ajuste ideal, necessitando de muito mais tempo e processamento até que haja a convergência. A Figura 2.6 mostra o comportamento do GD com diferentes valores de taxa de aprendizagem.

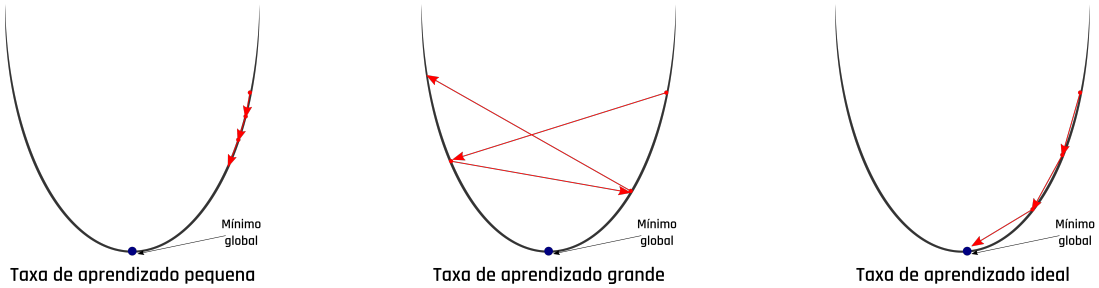

Figura 2.6: Efeitos da taxa de aprendizagem

- **Gradiente Descendente Estocástico**

Gradiente descendente, embora seja um bom método de otimização, peca por sua baixa eficiência para computar os gradientes, já que, devido ao alto número de exemplos, a função de custo se torna complicada. Uma alternativa mais eficiente é o gradiente descendente estocástico [26] [27], que escolhe apenas uma instância aleatória no conjunto de treinamento em cada etapa e calcula os gradientes baseado apenas nessa única instância. Obviamente, isso torna o algoritmo muito mais rápido, uma vez que tem poucos dados para manipular em cada iteração. Também possibilita treinar em grandes conjuntos de treinamento, já que apenas uma instância precisa estar na memória a cada iteração. No caso do gradiente descendente estocástico, a taxa de aprendizagem é tipicamente muito menor.

- ***Momentum***

Uma boa prática ao realizar gradiente descendente estocástico consiste em fazer uso de todas as direções (gradientes) computadas previamente, de forma que a direção atual passa a funcionar de maneira similar a uma média delas. Essa direção é chamada *momentum* [28].

A otimização do *momentum* se preocupa muito com os gradientes anteriores: em cada iteração, ele adiciona o gradiente local ao vetor *momentum* $\mathbf{m}$ (multiplicado pelo taxa de aprendizado η), e atualiza os pesos simplesmente subtraindo esse vetor *momentum* (veja a Equação 2.5 e Equação 2.6). Em outras palavras, o gradiente é usado como uma aceleração, não como uma velocidade. Para simular algum tipo de mecanismo de atrito e evitar o impulso de crescer muito, o algoritmo introduz um novo hiperparâmetro β, que deve ser definido entre 0 (alto atrito) e 1 (sem atrito).

$$\mathbf{m} \leftarrow \beta\mathbf{m} + \eta\nabla_\theta J(\theta) \tag{2.5}$$

$$\theta \leftarrow \theta - \mathbf{m} \tag{2.6}$$

- **Adam**

Esse trabalho utilizou o otimizador Adam [29], derivado de *adaptive moment estimation*. É um método para otimização estocástica eficiente que requer apenas gradientes de primeira ordem com pouca necessidade de memória. O método calcula as taxas individuais de aprendizado adaptativo para parâmetros diferentes das estimativas de primeiro e segundo momentos dos gradientes. É simples de implementar, é computacionalmente eficiente, tem poucos requisitos de memória, é invariável ao redimensionamento diagonal

dos gradientes e é adequado para problemas que são grandes em termos de dados e/ou parâmetros.

- **Retropropagação**

Por muitos anos os pesquisadores lutaram para encontrar uma maneira de treinar redes neurais, sem sucesso. Até que em 1986, foi publicado um trabalho apresentando o algoritmo de treinamento de *backpropagation* [30].

Para cada instância de treinamento, o algoritmo de *backpropagation* (ou retropropagação) primeiro faz uma previsão (*forward pass*), mede o erro e depois passa por cada camada em sentido inverso para medir a contribuição de cada conexão para o erro (*reverse pass*) e finalmente ajusta ligeiramente os pesos da conexão para reduzir o erro.

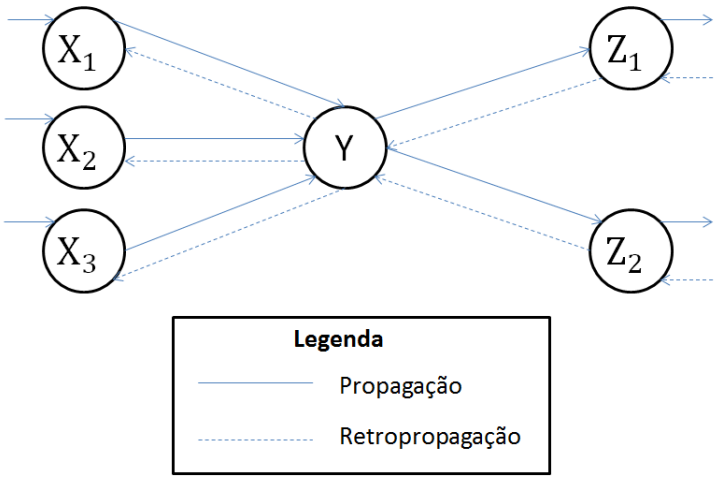

Figura 2.7: Retropropagação

- **Superajuste e Subajuste**

Pressupõe-se que o objetivo de um bom modelo é generalizar bem os dados de treinamento para quaisquer dados do domínio do problema. Isso nos permite fazer previsões no futuro sobre dados que o modelo nunca viu. Portanto, deve-se avaliar o quão bem um modelo é capaz de generalizar, evitando assim as duas maiores causas do baixo desempenho dos algoritmos: Superajuste (*Overfitting*) e o subajuste (*Underfitting*).

Superajuste ocorre quando o modelo se adequa muito bem ao conjunto de dados de treinamento, aprendendo os detalhes e o ruído, de forma que isso afeta negativamente seu desempenho em dados novos [31]. Isso significa que o ruído ou as flutuações aleatórias nos dados de treinamento são captados e aprendidos como conceitos pelo modelo. O problema é que esses conceitos não se aplicam a novos dados e impactam negativamente a capacidade de generalização dos modelos.

Subajuste ocorre quando o modelo não consegue aprender com o conjunto de dados de treinamento, tendo um desempenho ruim na fase de treino, portanto também não terá bom desempenho em um conjunto de dados novo [32].

Já o Ajuste de boa qualidade (*Good fit/Robust*) ocorre quando o modelo tem um bom desempenho no conjunto de dados de treinamento e também tem um bom desempenho em conjunto de dados novos, ou seja, tem uma boa capacidade de generalização.

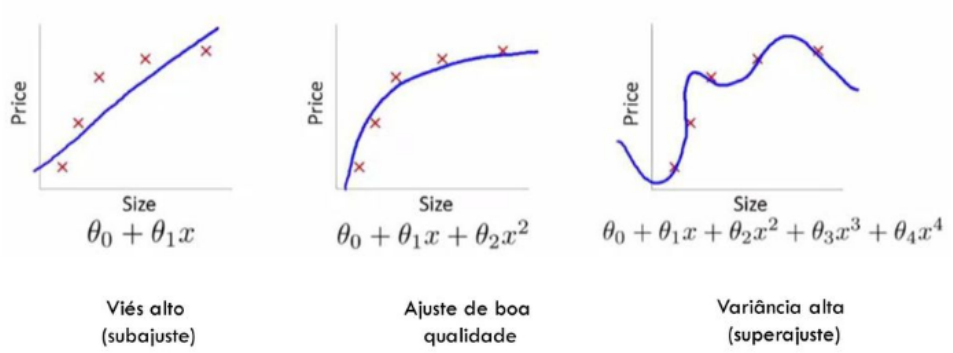

Figura 2.8: Subajuste, Ajuste de boa qualidade e superajuste

- **Controle de Superajuste**

Durante a fase de treinamento, é importante ficar atento para identificar como está o desempenho do modelo, tanto no conjunto de dados de treinamento, quanto no conjunto de dados de validação. Caso seja identificado que está ocorrendo Superajuste, é possível utilizar métodos para controlar o problema. Nesse trabalho foi utilizado o método de Parada Antecipada (*Early Stopping*) [33].

O conjunto de validação pode ser utilizado para detectar quando o superajuste começa durante o treinamento supervisionado de uma rede neural artificial. No caso mais simples, o treinamento é interrompido assim que o desempenho no conjunto de dados de validação diminui em comparação com o desempenho no conjunto de dados de validação na época (*epoch*) de treinamento anterior (por exemplo, um aumento na perda),ou seja, quando temos o erro mínimo no conjunto de validação. Esse processo é chamado de Parada Antecipada.

Entretanto, o treinamento de uma rede neural é estocástico e pode ser ruidoso. Em um gráfico, o desempenho de um modelo em um conjunto de dados de validação pode aumentar e diminuir muitas vezes. Isso significa que o primeiro sinal de superajuste pode não ser um bom lugar para parar de treinar. Dessa forma, utiliza-se um atraso ou "patience", que é uma tolerância para evitar terminar o treinamento muito cedo.

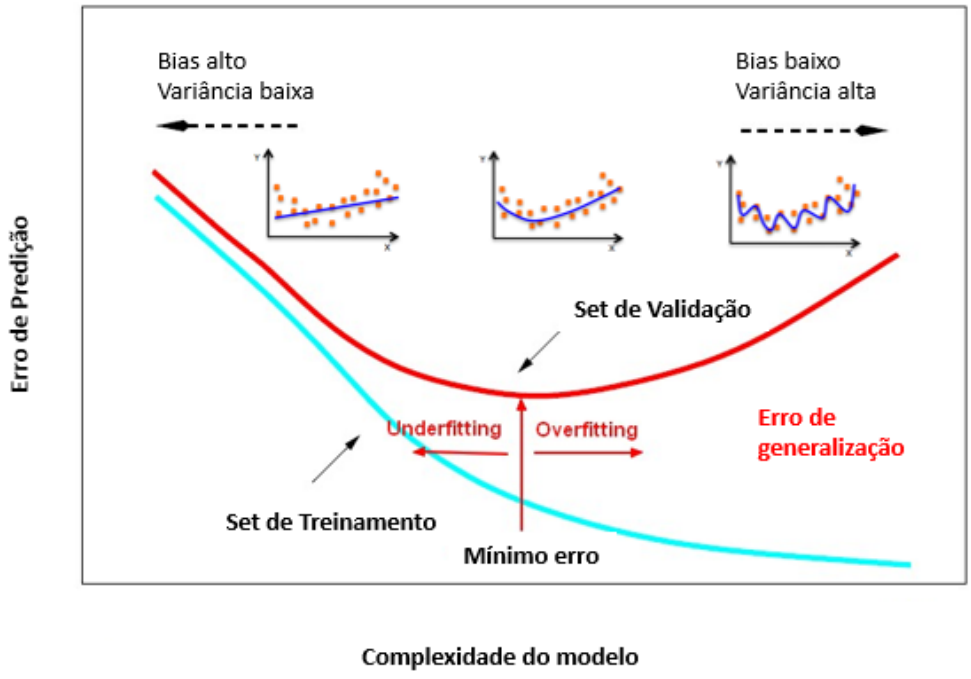

Figura 2.9: *Early Stopping*

Após a apresentação desses conceitos, podemos resumir redes neurais artificiais da seguinte forma:

As redes neurais, ou redes neurais artificiais [34], podem ser compostas por camadas de nós, contendo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada nó é composto de entradas, pesos, vieses (*bias* ou *threshold*) e uma saída. Se esse valor de saída exceder um determinado limite, ele "dispara" ou ativa o nó, passando dados para a próxima camada da rede. As redes neurais aprendem essa função de mapeamento por meio de aprendizado supervisionado, ajustando com base na função de perda por meio do processo de gradiente descendente. Quando a função de custo é igual ou próxima de zero, podemos confiar na precisão do modelo para fornecer a resposta correta.

2.1.2 Deep Learning

O Aprendizado Profundo [35] (*Deep Learning*) é um subconjunto do aprendizado de máquina, que é essencialmente uma rede neural com duas ou mais camadas. Embora uma rede neural com uma única camada ainda possa fazer previsões aproximadas, camadas ocultas adicionais podem ajudar a otimizar e refinar a precisão.

Algoritmos de aprendizado profundo podem ingerir e processar dados não estruturados, como texto e imagens, e automatizar a extração de recursos. Por exemplo, digamos

que exista um conjunto de fotos de diferentes animais de estimação e que queremos categorizar por “gato”, “cachorro”, “hamster”, etc. Algoritmos de aprendizado profundo podem determinar quais características (por exemplo, orelhas) são mais importantes para distinguir um animal do outro.

Então, por meio dos processos de gradiente descendente e retropropagação, o algoritmo de aprendizado profundo se ajusta, permitindo fazer previsões sobre uma nova foto de um animal com maior precisão.

2.1.3 Métricas de desempenho usadas em problemas de classificação

Diferentes métricas de desempenho são usadas para avaliar diferentes algoritmos de aprendizado de máquina. O foco será nas métricas utilizadas em problemas de classificação, já que o trabalho trata de um problema de classificação binária.

Algumas métricas comumente usadas são baseadas na matriz de confusão. Uma matriz de confusão é uma matriz N x N usada para avaliar o desempenho de um modelo de classificação, onde N é o número de classes alvo. A matriz compara os valores reais da variável alvo com os valores previstos pelo modelo. Isso nos dá uma visão holística do desempenho do nosso modelo de classificação e quais tipos de erros ele está cometendo.

Para um problema de classificação binária, temos uma matriz 2×2, como mostrado na Figura 2.10

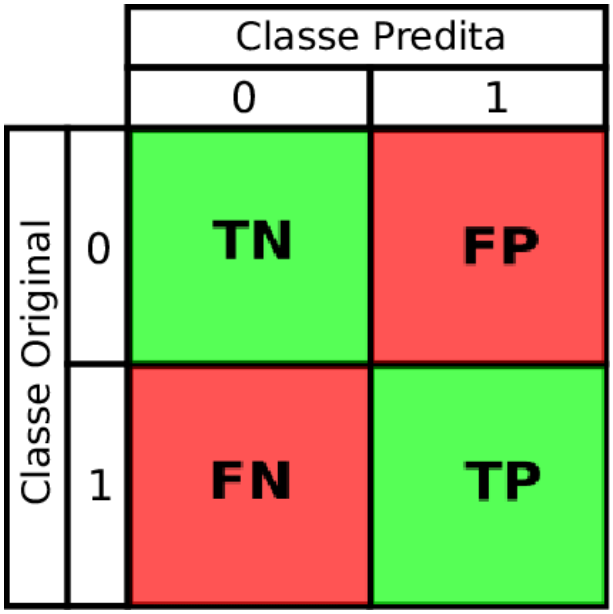

Figura 2.10: Matriz de confusão

Seus elementos são denominados como a seguir:

- ***True positives (TP)*:** são os casos em que a classe real do ponto de dados é 1 (Verdadeiro) e o previsto é 1 (Verdadeiro).
- ***True negatives (TN)*:** são os casos em que a classe real do ponto de dados é 0 (Falso) e o previsto também é 0 (Falso).
- ***False positives (FP)*:** são os casos em que a classe real do ponto de dados é 0 (Falso) e o previsto é 1 (Verdadeiro). Falso porque o modelo previu incorretamente e positivo porque a classe prevista foi positiva (1).
- ***False negatives (FN)*:** são os casos em que a classe real do ponto de dados foi 1 (Verdadeiro) e o previsto é 0 (Falso). Falso porque o modelo previu incorretamente e negativo porque a classe prevista foi negativa (0).

A partir dos elementos descritos, algumas métricas de desempenho do classificador binário são definidas[36]

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.10}$$

$$True\_Positive\_Rate = Recall \tag{2.11}$$

$$False\_Positive\_Rate = 1 - specificity = \frac{FP}{TN + FP} \tag{2.12}$$

Uma das medidas mais simples e mais utilizada é acurácia (*accuracy*), porém deve-se tomar cuidado ao utilizá-la para avaliar o desempenho. Há casos em que o conjunto de dados não está balanceado, então uma das classes está presente em maior quantidade que a outra classe. Supondo que a classe negativa represente 90% de um conjunto de dados e o classificador sempre atribua negativo aos exemplos, a acurácia do modelo será alta, mas somente essa métrica não indica que é um bom modelo, já que ele falha em classificar exemplos da classe positiva.

Por isso é importante a utilização de outras métricas em conjunto com a acurácia. No exemplo dado, *precision* ficará como indeterminada, já que não pode ser calculada, e *recall* terá valor 0, indicando que na verdade o classificador não é bom.

Também é comum o uso da métrica *F1-score*, calculada de acordo com a seguinte equação:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.13}$$

Outro modo de avaliar o desempenho do modelo é atráves da AUROC, uma medida de desempenho para os problemas de classificação em várias configurações de *threshold*. Receiver Operating Characteristic (ROC) é uma curva de probabilidade e AUROC representa o grau ou medida de separabilidade. Ele informa o quanto o modelo é capaz de distinguir entre as classes. Quanto maior a AUROC, melhor o modelo é para prever classes negativas como sendo negativas e classes positivas como sendo positivas.

A curva ROC é plotada com *True Positive Rate* em relação ao *False Positive Rate* onde *True Positive Rate* está no eixo y e *False Positive Rate* está no eixo x.

Idealmente, um classificador teria *True Positive Rate* igual a 1, ou seja, acertaria todos os exemplos classificados como positivos em relação aos exemplos que de fato são positivos e *False Positive Rate* com valor 0, já que não faria classificações incorretas. Dessa forma, o classificador ideal corresponde ao ponto $(0, 1)$. Por isso, o ponto ideal na curva ROC é aquele com menor distância euclidiana em relação ao ponto $(0, 1)$, conforme mostra a Figura 2.11

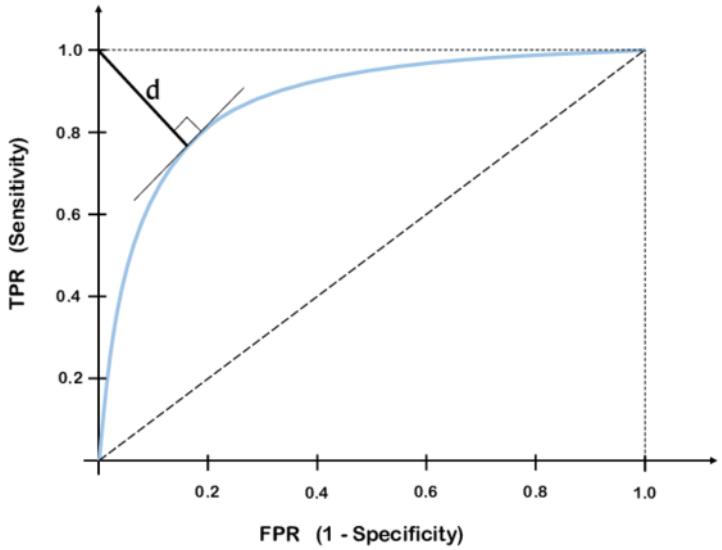

Figura 2.11: Exemplo de ponto ideal na curva ROC

Após a apresentação desses conceitos, o leitor está pronto para entender as discussões das seções seguintes, onde serão apresentadas a metodologia do trabalho e os resultados obtidos no desenvolvimento da rede neural.

Capítulo 3

Metodologia

A principal idéia desse trabalho é a utilização de Redes Neurais Artificiais para auxiliar no prognóstico de pacientes, tendo como principal objetivo a predição de mortalidade dentro da UTI. Nesse capítulo serão descritas todas as etapas do projeto.

Este capítulo se inicia com apresentação das ferramentas utilizadas. Em seguida, discorre-se sobre a fonte dos dados, e o respectivo banco de dados. Após a obtenção de dados, a próxima etapa apresentada é o pré-processamento dos dados, etapa na qual os dados foram ajustados para se organizarem em um formato adequado e alimentar a rede neural. Essa etapa do projeto ainda tratou da falta de dados, variáveis categóricas, a normalização das variáveis numéricas e o balanceamento. A próxima etapa aborda o desenvolvimento do modelo de rede neural, os ajustes de hiperparâmetros e arquitetura da rede neural. A última etapa aqui tratada é a avaliação do modelo, onde as métricas analisadas mostram o desempenho do modelo desenvolvido.

3.1 Ferramentas

Diversas ferramentas foram utilizadas para a realização desse projeto. Na etapa de seleção de dados, foi utilizado o *Google Big Query*, sua arquitetura sem servidor permitiu realizar consultas em *Structured Query Language*. Foi atráves dessa ferramenta que as tabelas do MIMIC-III foram consultadas e manipuladas para obter os dados desejados.

Já na etapa de pré-processamento, foi utilizado o *Google Colab*, ou *Google Colaboratory*, que permite escrever e executar a linguagem de programação Python no navegador sem nenhuma configuração necessária. As bibliotecas usadas foram pandas [37], numpy [38], matplotlib [39], seaborn [40] e scikit-learn [41].

Nas etapas de desenvolvimento e avaliação do modelo, além do scikit-learn, foram utilizadas as bibliotecas *tensorflow* [42] e keras [43].

3.2 Fonte de Dados

MIMIC-III [1] é um grande banco de dados relacional disponível gratuitamente que inclui dados anonimizados relacionados à saúde, associados a pacientes que permaneceram em UTI's do Beth Israel Deaconess Medical Center entre 2001 e 2012. O banco de dados inclui informações como medições de sinais feitas à beira do leito (1 ponto de dados por hora), resultados de exames laboratoriais, procedimentos, medicamentos, anotações do cuidador, relatórios de imagem e mortalidade (incluindo alta pós-hospitalar).

3.2.1 Características dos pacientes

O banco contém dados associados a 53.423 internações hospitalares distintas para pacientes adultos (16 anos ou mais) internados em UTI entre 2001 e 2012. Além disso, contém dados de 7.870 neonatos internados entre 2001 e 2008. A idade média dos pacientes adultos é de 65,8 anos (Q1–Q3: 52,8–77,8), 55,9% dos pacientes são do sexo masculino e a mortalidade hospitalar é de 11,5%. A duração média de uma internação na UTI é de 2,1 dias (Q1-Q3: 1,2–4,6) e a duração média de uma internação hospitalar é de 6,9 dias (Q1-Q3: 4,1–11,9).

3.2.2 Tabelas

O MIMIC-III é formado por 26 tabelas. Uma tabela é uma estrutura de armazenamento de dados semelhante a uma planilha: cada coluna contém informações consistentes (por exemplo, identificadores de paciente) e cada linha contém uma instanciação dessa informação.

As tabelas são vinculadas por identificadores que geralmente possuem o sufixo "ID". Por exemplo, HADM_ID refere-se a uma admissão hospitalar única e SUBJECT_ID refere-se a um paciente único. Uma exceção é ROW_ID, que é simplesmente um identificador de linha exclusivo para essa tabela.

Tabelas prefixadas com "D_" são dicionários e fornecem definições para identificadores. Por exemplo, cada linha de OUTPUTEVENTS está associada a um único ITEMID que representa o conceito medido, mas não contém o nome real do medicamento. Ao unir OUTPUTEVENTS e D_ITEMS no ITEMID, é possível identificar qual conceito um determinado ITEMID representa.

As tabelas a seguir são usadas para definir e rastrear as estadias dos pacientes:

- **ADMISSIONS:** Cada hospitalização única para cada paciente no banco de dados (define HADM_ID).
- **CALLOUT:** Informações sobre quando um paciente foi liberado para alta da UTI.

- **ICUSTAYS:** Cada estadia única na UTI no banco de dados (define ICUSTAY_ID).
- **PATIENTS:** Cada paciente único no banco de dados (define SUBJECT_ID).
- **SERVICES:** O serviço clínico sob o qual um paciente está registrado.
- **TRANSFERS:** Movimentação de pacientes de um leito para outro leito dentro do hospital, incluindo admissão e alta na UTI.

Cada ICUSTAY_ID corresponde a um único HADM_ID e a um único SUBJECT_ID.Cada HADM_ID corresponde a um único SUBJECT_ID. Um único SUBJECT_ID pode corresponder a vários HADM_ID (várias hospitalizações do mesmo paciente) e vários ICUSTAY_ID (várias estadias na UTI na mesma hospitalização, em várias hospitalizações ou em ambas).

As tabelas a seguir contêm dados coletados na unidade de terapia intensiva:

- **CAREGIVERS:** Todo cuidador que tenha dados cadastrados no banco de dados (define CGID).
- **CHARTEVENTS:** Todas as observações registradas para pacientes.
- **DATETIMEEVENTS:** Todas as observações registradas que são datas, por exemplo, hora da diálise ou inserção de linhas.
- **INPUTEVENTS_CV:** Ingestão para pacientes monitorados usando o sistema Philips CareVue enquanto internados na UTI.
- **INPUTEVENTS_MV:** Ingestão para pacientes monitorados usando o sistema iMDSoft Metavision enquanto internados na UTI.
- **NOTEEVENTS:** Notas desidentificadas, incluindo notas de enfermagem e de médicos, relatórios de Eletrocardiograma (ECG), relatórios de imagem e resumos de alta.
- **OUTPUTEVENTS:** Saída de informações para pacientes enquanto internados na UTI.
- **PROCEDUREEVENTS_MV:** Procedimentos do paciente para o subconjunto de pacientes que foram monitorados na UTI usando o sistema iMDSoft MetaVision.

As tabelas a seguir contêm dados coletados no sistema de registro hospitalar:

- **CPTEVENTS:** Procedimentos registrados como códigos de Terminologia Processual Atual (CPT).

- **DIAGNOSES_ICD:** Diagnósticos atribuídos ao hospital, codificados usando o sistema Classificação Estatística Internacional de Doenças e Problemas Relacionados à Saúde (CID).
- **DRGCODES:** Grupos Relacionados ao Diagnóstico (GRD), que são utilizados pelo hospital para fins de faturamento.
- **LABEVENTS:** Medições laboratoriais para pacientes tanto dentro do hospital quanto em ambulatórios.
- **MICROBIOLOGYEVENTS:** Medições de microbiologia do banco de dados do hospital.
- **PRESCRIPTIONS:** Medicamentos prescritos, e não necessariamente administrados, para um determinado paciente.
- **PROCEDURES_ICD:** Procedimentos do paciente, codificados usando o sistema CID.

As seguintes tabelas são dicionários:

- **D_CPT:** Dicionário de alto nível de códigos de terminologia processual atual(CTP).
- **D_ICD_DIAGNOSES:** CID relativos a diagnósticos.
- **D_ICD_PROCEDURES:** CID relativos a procedimentos.
- **D_ITEMS:** Dicionário de ITEMID's que aparecem no banco de dados. MIMIC-III, exceto aqueles relacionados a exames laboratoriais.
- **D_LABITEMS:** Dicionário de ITEMID's no banco de dados do laboratório que se relacionam com exames laboratoriais.

3.3 Seleção de internações

Para esse trabalho, o critério de exclusão de internações está descrito no diagrama apresentado na Figura 3.1.

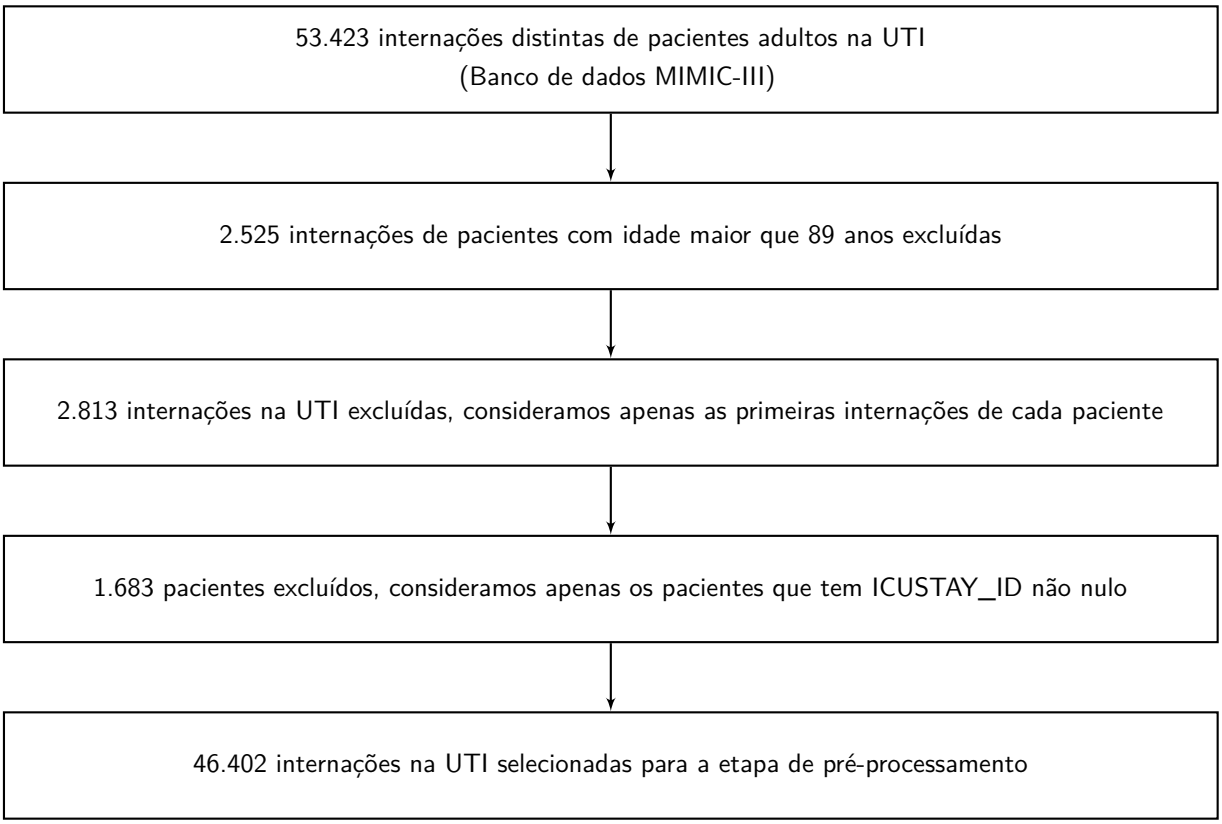

Figura 3.1: Fluxograma da seleção de internações para a etapa de pré-processamento

3.4 Variáveis preditoras

Variáveis preditoras, também conhecidas como variáveis independentes ou *features*, são variáveis utilizadas pelo modelo para encontrar padrões e determinar (predizer) a variável alvo (*target variable*). Nesse trabalho foram selecionadas as seguintes *features*:

3.4.1 Idade

Foram considerados pacientes com mais de 15 anos e menos de 90 anos, pois pacientes com mais de 89 anos de idade em qualquer momento no banco de dados tiveram sua data de nascimento alterada para ocultar sua idade e cumprir a Health Insurance Portability and Accountability Act (HIPAA). O processo de mudança da idade ocorreu da seguinte forma: foi determinada a idade do paciente em sua primeira admissão. A data de nascimento foi então definida para exatamente 300 anos antes de sua primeira admissão.

3.4.2 Gênero

A Tabela 3.1 apresenta a distribuição dos gêneros antes do pré-processamento.

Gênero	Internações	Proporção
Masculino	26.635	57,40%
Feminino	19.767	42,60%

Tabela 3.1: Distribuição de gêneros

3.4.3 Sinais vitais e exames laboratoriais

O MIMIC-III possui um grande número de medições de sinais vitais e exames laboratoriais. Os sinais vitais e exames laboratoriais selecionados foram:

- **Batimentos Cardíacos (*Heart Rate)*.**
- **Frequência respiratória (*Respiratory Rate)*.**
- **Pressão Arterial Sistólica (*Systolic Blood Pressure)*.**
- **Pressão Arterial Diastólica (*Diastolic Blood Pressure)*.**
- **Pressão Arterial Média (*Mean Blood Pressure)*.**
- **Saturação do Oxigênio (*Oxygen Saturation)*.**
- **Temperatura (*Temperature)*.**
- **Creatinina (*Creatinine)*.**
- **Glicose (*Glucose)*.**
- **pH**.
- **Uréia (*Blood Urea Nitrogen (BUN))*.**
- **Hemoglobina (*Hemoglobin)*.**
- **Contagem de glóbulos brancos (*White Blood Cell Count (WBC))*.**

3.4.4 Escala de Coma de Glasgow

A escala de coma de glasgow [44] foi inicialmente utilizada para avaliar o nível de consciência em pacientes vítimas de traumatismo cranioencefálico, sendo depois amplamente aplicada em outros pacientes agudamente doentes. Em hospitais, também é usada para monitorização de pacientes em UTI. A Tabela 3.2 mostra os parâmetros da escala e como a pontuação é calculada.

Parâmetro	Resposta	Pontuação
Abertura Ocular (*Eye Opening*)	Espontânea	4
	Ao comando verbal	3
	Ao estímulo doloroso	2
	Nenhuma	1
Resposta Verbal (*Verbal Response*)	Orientada	5
	Confusa	4
	Palavras	3
	Sons	2
	Nenhuma	1
Resposta Motora (*Motor Response*)	Obedece comandos	6
	Localiza a dor	5
	Flexão normal	4
	Flexão anormal	3
	Extensão	2
	Nenhuma	1

Tabela 3.2: Parâmetros usados para calcular a pontuação da escala de coma de Glasgow

A Tabela 3.3 mostra dados estatísticos, como a média, desvio-padrão, valor mínimo, primeiro quartil, segundo quartil, terceiro quartil e valor máximo antes do pré-processamento.

	μ	σ	mín	25%	50%	75%	máx	unidade
Batimentos Cardíacos	96,024	9.880,47	0	73	85	97	100.000	bpm
Frequência Respiratória	20,88	2.337,14	0	15	18	22	23.555	mrm
Pressão Arterial Diastólica	61,35	218,7	-16	51	59	69	114.109	mmHg
Pressão Arterial Sistólica	119,09	179,24	-11	102	116	133	127.106	mmHg
Pressão Arterial Média	78,73	125,60	-38	67	77	88	120.130	mmHg
Saturação do Oxigênio	97,27	13,88	0	96	98	100	9.696	%
Abertura Ocular	3,28	1,09	1	3	4	4	4	-
Resposta Verbal	3,47	1,86	1	1	5	5	5	-
Resposta Motora	5,35	1,41	1	6	6	6	6	-
Temperatura	36,83	2,20	-17,78	36,3	36,89	37,44	536,38	°C
Uréia	26,63	22,56	0	13	19	32	270	mg/dL
Creatinina	1,57	3,06	0	0.7	1	1,5	255	mg/dL
Glicose	151,05	2.443,14	0	108	131	164	999.999	mg/dL
Hemoglobina	10,67	1,93	0	9,3	10,5	11,9	32,6	g/dl
pH	7,38	3,1	0	7,3	7,4	7,4	750	-
Glóbulos Brancos	12,27	11,21	0	7,7	10,8	14,7	665,6	K/uL

Tabela 3.3: Dados estatísticos dos sinais vitais, exames laboratoriais e escala de coma de glasgow

3.5 Variável alvo: Mortalidade na UTI

A variável alvo, também chamada de variável dependente ou *target variable*, é a variável que queremos predizer. Nesse trabalho essa variável é a mortalidade de pacientes na UTI. Foi obtida através da diferença entre a data de morte do paciente e a data de saída da UTI. Se as datas forem iguais, a variável tem valor 1, caso contrário, valor 0. A Tabela 3.4 mostra a distribuição de sobreviventes e não sobreviventes antes do pré-processamento.

Sobreviventes	Não sobreviventes	Taxa de mortalidade na UTI
41.233	5.169	12,54%

Tabela 3.4: Taxa de mortalidade

3.6 Pré-processamento de dados

Antes de criar um modelo de rede neural e realizar o treinamento, é importante compreender melhor os dados para identificar e tratar valores nulos, medições erradas, diferenças em escalas, balanceamento dos dados, codificação de variáveis categóricas e *feature extraction*, com o objetivo de melhorar o desempenho do modelo.

3.6.1 Medições Erradas

MIMIC-III possui valores extremamente altos ou baixos, considerados impossíveis para alguns sinais vitais e exames laboratoriais, e portanto devem ser retirados. Esses valores são causados por falhas nas medições ou no armazenamento da informação. É importante, no entanto, que valores extremamente altos ou baixos, mas possíveis, sejam mantidos para indicar a severidade no estado de saúde do paciente.

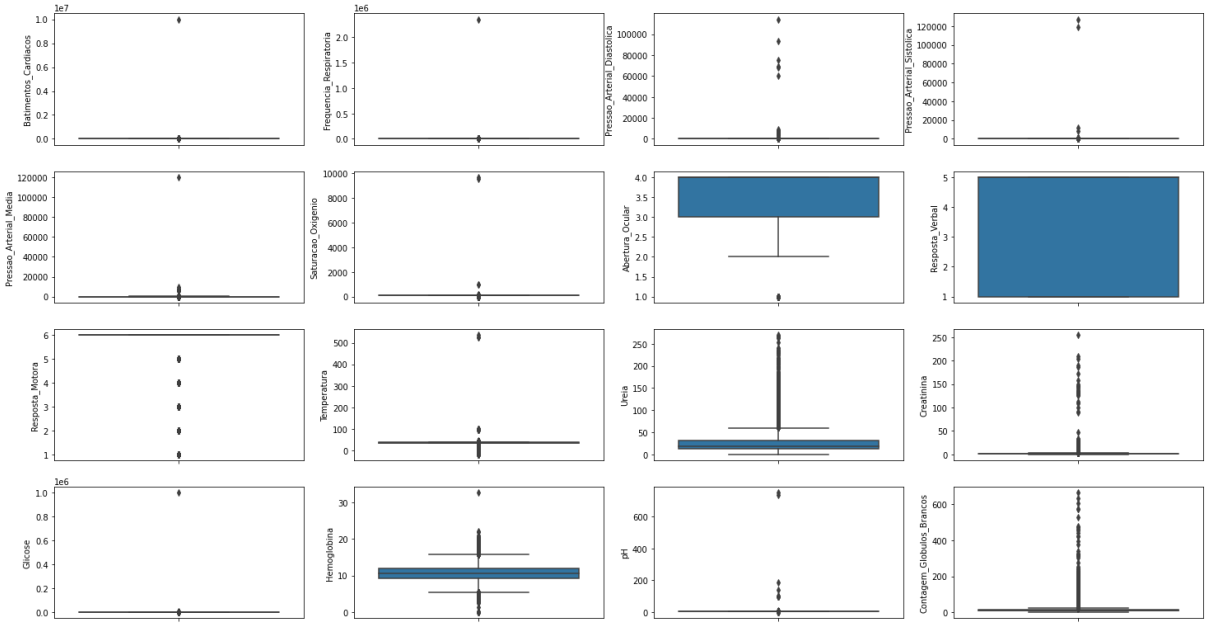

Figura 3.2: Boxplot

Os valores abaixo do percentil 5% e acima do percentil 95% foram excluídos, resultando na Tabela 3.5, que mostra o resultado após a exclusão dos valores anormais.

	μ	σ	mín	25%	50%	75%	máx	unidade
Batimentos Cardíacos	86,20	18,84	45	73	85	97	459	bpm
Frequência Respiratória	18,54	6,06	4	15	18	22	250	mrm
Pressão Arterial Diastólica	60,60	14,76	30	51	59	69	268	mmHg
Pressão Arterial Sistólica	118,71	23,65	82	102	116	133	270	mmHg
Pressão Arterial Média	78,39	16,81	63	67	76,33	87,33	270	mmHg
Saturação do Oxigênio	97,26	3,85	82	96	98	100	100	%
Abertura Ocular	3,28	1,09	1	3	4	4	4	-
Resposta Verbal	3,47	1,86	1	1	5	5	5	-
Resposta Motora	5,35	1,41	1	6	6	6	6	-
Temperatura	36,87	0,89	26	36,3	36,89	37,44	46,5	°C
Uréia	26,49	22,24	5,2	13	19	32	200	mg/dL
Creatinina	1,49	1,55	0,31	0.7	1	1,5	13	mg/dL
Glicose	144,25	62,97	74,3	108	131	163	1348	mg/dL
Hemoglobina	10,65	1,93	6,4	9,3	10,5	11,9	32,6	g/dl
pH	7,36	0,09	6,61	7,32	7,37	7,42	7.78	-
Glóbulos Brancos	11,98	7,16	2,4	7,7	10,7	14,6	110	K/uL

Tabela 3.5: Dados estatísticos dos sinais vitais, exames laboratoriais e escala de coma de Glasgow após exclusão de valores anormais.

3.6.2 Valores ausentes

Além de erros de medição e imputação de dados, é normal ocorrer ausência de medições em um banco de dados médico, que muitas vezes são codificados como espaços em branco ou Not a Number (NaN). Alguns algoritmos de Aprendizado de Máquina toleram a ausência de dados, porém quando se trata de redes neurais devemos tratar essa ausência, seja excluindo ou imputando valores usando a média, por exemplo. A Figura 3.3 e a Figura 3.4 mostram os sinais vitais e exames laboratorias de dois pacientes durante as primeiras 24 horas após internação na UTI.

É possível ver que há ausência de medições em ambos os pacientes. Na Figura 3.3 o paciente tem uma quantidade razoável de valores ausentes, porém há uma boa variedade de sinais vitais e exames laboratoriais medidos. Já na Figura 3.4, o paciente quase não tem medições realizadas e são poucos os sinais vitais e exames laboratoriais medidos.

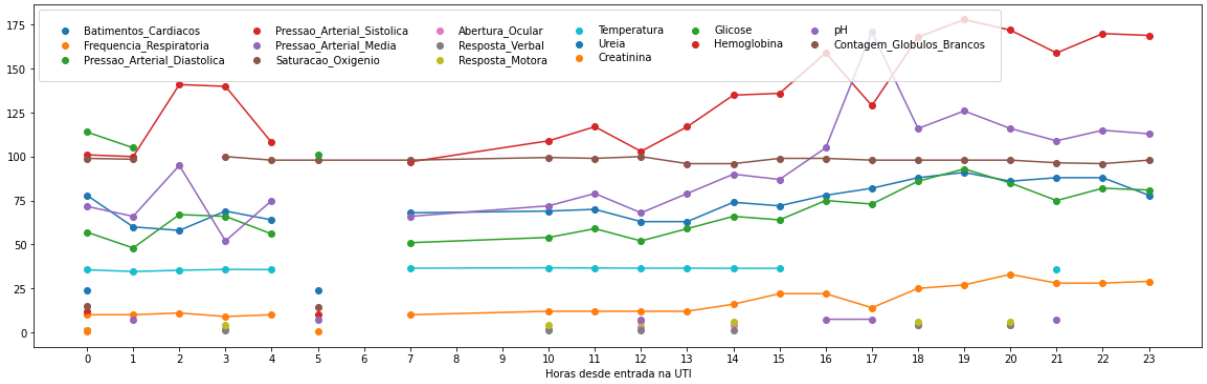

Figura 3.3: Sinais vitais e exames laboratoriais de um paciente

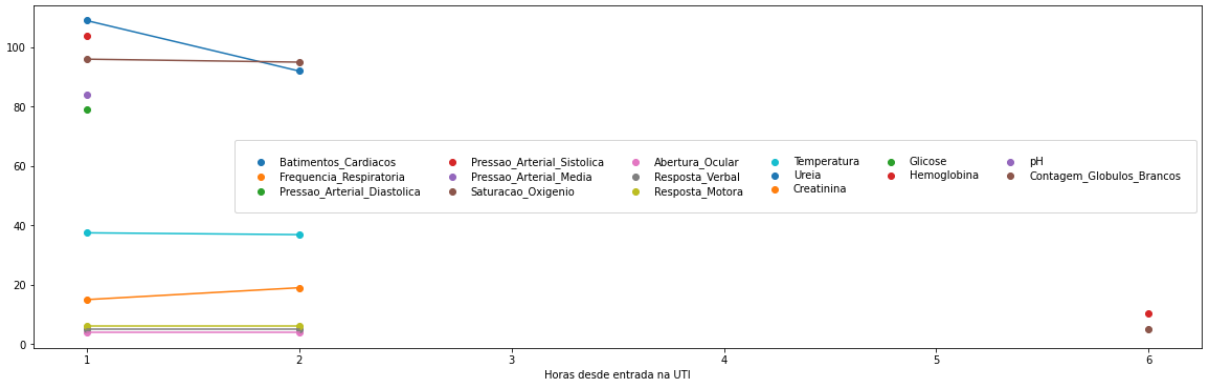

Figura 3.4: Sinais vitais e exames laboratoriais de outro paciente

A Tabela 3.6 mostra a porcentagem de valores ausentes por *feature*. Os sinais vitais possuem uma baixa quantidade de valores ausentes, o que condiz com a forma como são medidos na UTI. Já os exames laboratoriais possuem uma alta quantidade de valores ausentes, pois esses exames não são realizados de hora em hora e levam tempo para ser processados pelo laboratório.

Ureia, creatinina, hemoglobina, pH e glóbulos brancos são importantes para avaliar a situação de um paciente, e mesmo tendo uma alta quantidade de valores ausentes, serão mantidos.

Feature	Valores ausentes
Batimentos Cardíacos	12,9%
Frequência Respiratória	13,9%
Pressão Arterial Diastólica	12,4%
Pressão Arterial Sistólica	12,4%
Pressão Arterial Média	12,4%
Saturação do Oxigênio	12,7%
Abertura Ocular	69%
Resposta Verbal	69%
Resposta Motora	69,2%
Temperatura	53,8%
Uréia	86,6%
Creatinina	86,6%
Glicose	67,1%
Hemoglobina	85,3%
pH	70,2%
Glóbulos Brancos	86,1%

Tabela 3.6: Porcentagem de valores ausentes no *dataframe*.

Como dito anteriormente, há inúmeras formas de se lidar com a ausência de valores, seja realizando a imputação de valores ou excluindo as linhas e/ou colunas com valores ausentes.

A estratégia adotada foi excluir as internações na UTI que possuem menos do que duas medições por variável. Ou seja, para cada sinal vital, exame laboratorial e parâmetro da escala de Glasgow, o paciente deve ter no mínimo duas medições realizadas nas primeiras 24 horas. Ao adotar esse critério, o paciente da Figura 3.4 foi excluído.

No entanto, ainda há valores ausentes após esse critério, pois idealmente, cada paciente deveria possuir 24 medições por sinal vital e exame laboratorial, ou seja, uma medição por hora para todas as variáveis. Como selecionamos os pacientes que contém no mínimo duas medições, ainda haverá valores ausentes. Esse problema será resolvido no tópico de *Feature extraction*.

3.6.3 *Feature extraction*

A próxima etapa é extrair *features* relevantes da série temporal. Seria possível utilizar a série temporal completa, ou seja, teríamos 24 para cada sinal vital e exame laboratorial, começando em $t = 0$h até $t = 23$h. No entanto, seria necessário imputar valores ou excluir novamente mais variáveis e medições. Além disso, como há 16 variáveis, haveriam 16 variáveis x 24 horas, resultando em 384 variáveis preditoras, o que tornaria o modelo muito mais complexo.

Uma solução simples para esse problema é usar apenas parte das informações disponíveis, sendo idealmente as informações mais importantes para a predição. Dessa forma, foram usadas operações simples para construir/extrair características importantes. Para cada variável preditora (sinais vitais, exames laboratoriais e parâmetros da escala de coma de Glasgow), foram extraídas as seguintes informações:

- **Máximo**.
- **Mínimo**.
- **Desvio padrão**.
- **Média**.

Essas variáveis resumiram a pior medição, a melhor, a variação e a condição média de cada paciente de $t = 0$h a $t = 23$h.

Após essas etapas, 8.380 internações na UTI foram selecionadas.

Antes da *Feature extraction*	Depois da *Feature extraction*
Batimentos Cardíacos	min_Batimentos_Cardíacos max_Batimentos_Cardíacos media_Batimentos_Cardíacos desvio_padrão_Batimentos_Cardíacos

Tabela 3.7: Exemplo da transformação realizada para cada sinal vital, exame laboratorial e pârametro da escala de coma de Glasgow.

3.6.4 Divisão do conjunto de dados

O conjunto de dados foi divido em três: o conjunto de dados de treinamento, utilizado para treinar a rede neural, o conjunto de dados de validação, que é conjunto de exemplos usados para ajustar os hiperparâmetros de um classificador, como o número de camadas em uma rede neural e o número de camadas escondidas e o conjunto de dados de teste, uma amostra de dados nunca vista pelo modelo, usada para fornecer uma avaliação imparcial do desempenho de um classificador.

A Tabela 3.8 mostra como a divisão foi realizada.

	Quantidade de Internações na UTI	%
Conjunto de treinamento	5.866	70
Conjunto de validação	1.257	15
Conjunto de teste	1.257	15
Total	8.380	100

Tabela 3.8: Divisão do conjunto de dados

3.6.5 Codificação de variáveis categóricas

Variáveis categóricas normalmente são representadas como *'strings'* ou 'categorias' e são finitas em número. São divididas em variáveis categóricas nominais, onde as categorias não possuem ordem, e em variáveis categóricas ordinais, onde as categorias possuem ordem.

Como a maioria dos modelos de aprendizado de máquina aceitam apenas variáveis numéricas, o pré-processamento das variáveis categóricas torna-se uma etapa necessária. Dessa forma, foi preciso converter essas variáveis categóricas em números para que o modelo fosse capaz de entender e extrair informações valiosas.

No caso desse projeto, há apenas uma variável categórica, o gênero dos pacientes.

Gênero	Codificação
Masculino	1
Feminimo	0

Tabela 3.9: Codificação da variável categórica

3.6.6 Balanceamento do conjunto de dados

No MIMIC-III os dados estão estruturados de tal forma que a variável alvo (a mortalidade na UTI) não está balanceada, o que pode prejudicar o desempenho do modelo. A quantidade da classe de sobreviventes, ou classe majoritária, é muito maior do que a quantidade de não sobreviventes, a classe minoritária, conforme é mostrado na Tabela 3.10.

	Quantidade	%
Sobreviventes	4.968	84,7
Não sobreviventes	898	15,3
Total	5.866	100

Tabela 3.10: Taxa de mortalidade na UTI no conjunto de treinamento

Para lidar com o problema do balanceamento no conjunto de dados, o procedimento mais comum é alterar a distribuição da variável alvo, tornando-a mais balanceada. Basicamente, existem dois métodos para realizar o balanceamento:

1. ***Over-sampling***: Consiste em replicar exemplos da classe minoritária para alcançar uma distribuição mais balanceada.
2. ***Under-sampling***: Consiste em balancear o conjunto de dados eliminando exemplos da classe majoritária.

Ambos os métodos apresentam vantagens e desvantagens. Ao usar *Over-sampling* a chance de ocorrer *overfitting* aumenta. Já o uso de *Under-sampling* pode excluir dados úteis.

A técnica aqui utilizada, denominada Synthetic Minority Over-sampling Technique (SMOTE) [45], tem como ideia principal formar novos exemplos da classe minoritária interpolando entre vários exemplos de classe minoritária que se encontram juntos.

Em conjunto com o SMOTE, *Tomek links* [46] foram utilizados como método de *Under-sampling* [47]. Um *Tomek link* pode ser definido da seguinte forma: dados dois exemplos x e y pertencentes a classes diferentes, e seja $d(x, y)$ a distância entre x e y. Um par (x, y) é chamado de *Tomek link* se não houver um caso z, tal que $d(x, z) < d(x, y)$ ou $d(y, z) < d(y, x)$. Se dois exemplos formam um *Tomek link*, então ou um desses exemplos é ruído ou ambos os exemplos são limítrofes.

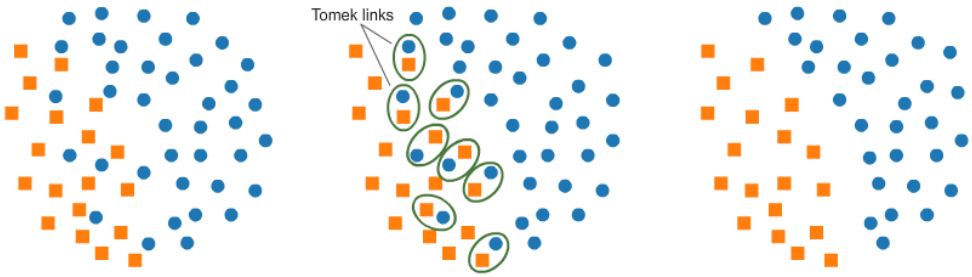

Figura 3.5: *Tomek links*

Dessa forma, utilizou-se SMOTE como método de *Over-sampling* em conjunto com *Tomek links* como método de *Under-sampling*.

A combinação desses métodos, comparada com outros métodos, resultou na menor taxa de Falsos negativos (FN), na maior taxa de Falsos positivos (FP) e a maior AUROC [48].

Ao aplicar SMOTE + *Tomek links* no conjunto de treinamento, obtemos a seguinte distribuição de classes:

	Quantidade	%
Sobreviventes	4.961	50
Não sobreviventes	4.961	50
Total	9.922	100

Tabela 3.11: Taxa de mortalidade na UTI no conjunto de treinamento após SMOTE + *Tomek links*

3.6.7 Normalização

Na maioria das vezes, o conjunto de dados conterá *features* (também chamadas de variáveis preditoras ou variáveis independentes) que variam muito em magnitude, unidade e alcance. Como a maioria dos algoritmos de aprendizado de máquina usa a distância Euclidiana entre dois pontos de dados em seus cálculos, isso é um problema.

Se a variância de uma *feature* for muito maior do que a variância de outras *features*, essa *feature* específica pode dominar outros recursos no conjunto de dados, tendo um maior peso na saída.

Ao usar GD, deve-se garantir que todos as *features* têm uma escala semelhante, ou então levará muito mais tempo para convergir. Por isso, é importante realizar normalização conforme a Equação 3.1

$$X_{proc} = \frac{X - \mu}{\sigma} \tag{3.1}$$

dessa forma as *features* são redistribuídas de tal forma que sua média $\mu = 0$ e desvio padrão $\sigma = 1$.

3.7 Desenvolvimento do modelo de Redes Neurais para classificação

Após a realização do pré-processamento, o conjunto de dados está no formato adequado para ser utilizado pela rede neural. O conjunto de dados que será utilizado é composto por 8.380 internações na UTI, sendo que cada internação representa um paciente, e 66 *features*, que foram descritas anteriormente.

3.7.1 Ajuste de hiperparâmetros

Muitas vezes é necessário procurar um conjunto de hiperparâmetros que resultem no melhor desempenho de um modelo, porém não há uma regra definida para esse ajuste, já que os hiperparâmetros devem ser adequados ao conjunto de dados utilizado. *Ran-*

dom Search e *Grid Search* são métodos de otimização de hiperparâmetros amplamente utilizados.

- ***Random Search*:** Método onde combinações aleatórias dos hiperparâmetros são usadas para encontrar a melhor solução para o modelo construído. Comparado com redes neurais configuradas por *Grid Search*, *Random Search*, no mesmo domínio, é capaz de encontrar modelos tão bons ou melhores dentro de uma pequena fração do tempo de computação[49].
- ***Grid Search*:** Envolve a especificação de uma lista de possíveis valores de hiperparâmetros que se deseja testar e, em seguida, o algoritmo treinará modelos com todas as combinações possíveis dos valores de hiperparâmetros fornecidos e avaliará o desempenho de cada modelo treinado usando uma métrica especificada (por exemplo, acurácia de previsões em um conjunto de dados de validação).

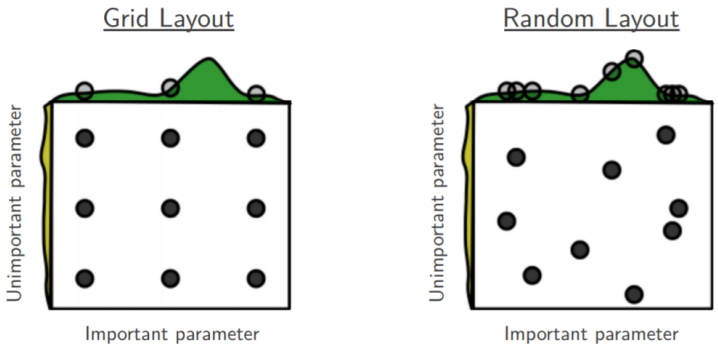

Figura 3.6: *Grid* e *Random Search* (Fonte[49])

3.7.2 *Training loss* e *Validation loss*

Na etapa de treinamento do modelo, o *Random Search* foi utilizado para ajustar os hiperparâmetros, com o objetivo de obter o modelo com maior acurácia. Entretanto, também é importante analisar a perda no conjunto de treinamento (*Training loss*) e a perda no conjunto de validação (*Validation loss*) para verificar a ocorrência de superajuste e subajuste. A Figura 3.7 mostra as curvas do modelo utilizado.

É possível notar que as duas curvas tem um comportamento parecido até aproximadamente 60 épocas (*epochs*). A partir desse ponto, a curva de perda no conjunto de validação é maior do que a curva de perda no conjunto de treinamento, caracterizando superajuste.

Conforme descrito na seção teórica, há um momento adequado para terminar o treinamento, evitando que esse tipo de comportamento ocorra. Para isso foi utilizado a parada

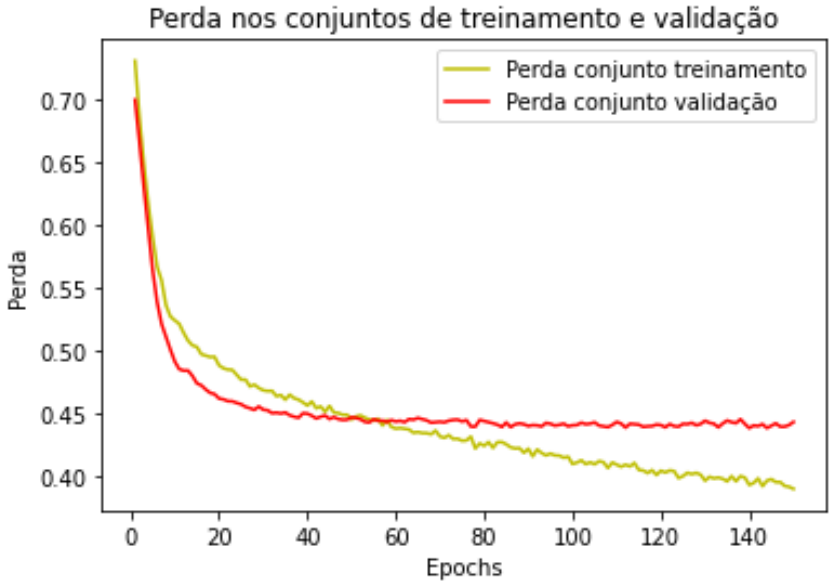

Figura 3.7: *Training loss* e *Validation loss* do modelo

antecipada (*Early stopping*), um método que para o treinamento no momento em que a curva de perda no conjunto de validação começa a aumentar, evitando assim o superajuste.

3.7.3 *Training accuracy* e *Validation accuracy*

Também é importante comparar a acurácia no conjunto de treinamento (*Training accuracy*) e a acurácia no conjunto de validação (*Validation accuracy*). A Figura 3.8 mostra as curvas do modelo utilizado.

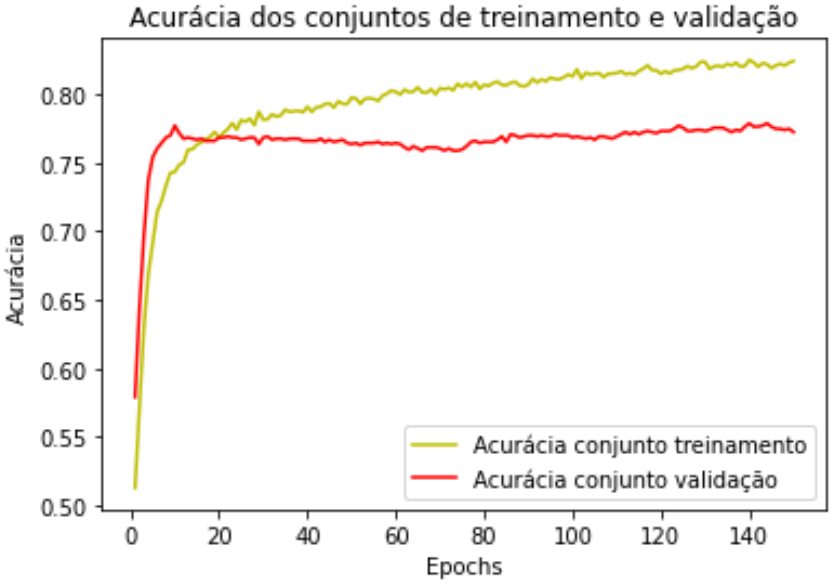

Figura 3.8: *Training accuracy* e *Validation accuracy* do modelo

Nesse caso, as curvas tem um comportamento parecido até aproximadamente 20 épocas. A partir desse ponto, a curva de acurácia no conjunto de treinamento começa a crescer muito, enquanto a acurácia no conjunto de validação permanece em um certo nível. Quando a curva de acurácia no conjunto de treinamento é muito maior do que a curva acurácia no conjunto de validação, também ocorre o superajuste.

3.7.4 Arquitetura da rede neural

A rede neural ficou assim configurada:

- **Quantidade de camadas ocultas:** 2
- **Quantidade de neurônios por camada oculta:** 16
- ***Learning Rate*:** 0,0001
- **Função de ativação das camadas ocultas:** ReLU
- **Função de ativação da camada de saída:** Sigmoid
- **Funções de otimização:** Adam
- ***Batch Size*: 64**
- ***Epoch*: 20**

A Figura 3.9 mostra a arquitetura da rede neural. Importante notar que a *input layer* foi visualmente simplificada na imagem, já que ela recebe as 66 variáveis preditoras como entrada. As camadas são totalmente conectadas (*fully connected*).

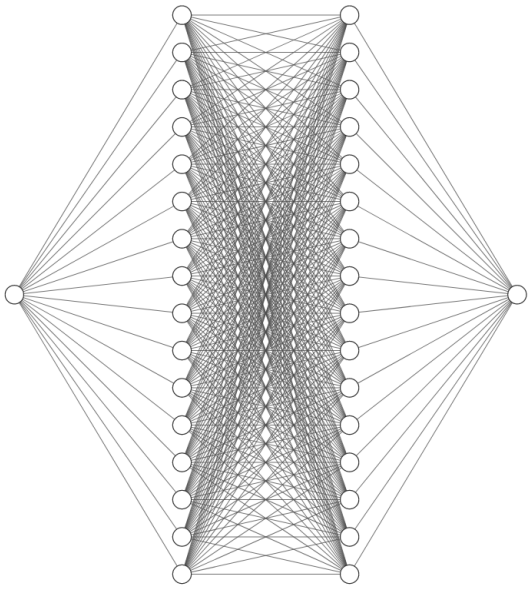

Figura 3.9: *Arquitetura da Rede Neural*

Capítulo 4

Resultados Obtidos

Na etapa de pré-processamento, o conjunto de dados foi divido em três: conjunto de treinamento, conjunto de validação e o conjunto de teste. O método de otimização de hiperparâmetros *Random Search* utilizou o conjunto de treinamento para aprender padrões e o conjunto de validação para adequar o modelo e evitar superajuste.

Em seguida, o conjunto de teste é usado para avaliar o desempenho (ou seja, generalização) do modelo, através das métricas citadas na introdução teórica: *accuracy*, *recall*, *precision*, *F1-score* e *AUROC*.

4.1 Avaliação do modelo

A Tabela 4.1 mostra as métricas obtidas ao utilizar o modelo proposto, a Figura 4.1 mostra *AUROC* e a Figura 4.2 a matriz de confusão.

Métrica	Valor
Accuracy	0.77
Recall	0.81
Precision	0.37
F1-score	0.51
AUROC	0.86

Tabela 4.1: Desempenho do modelo

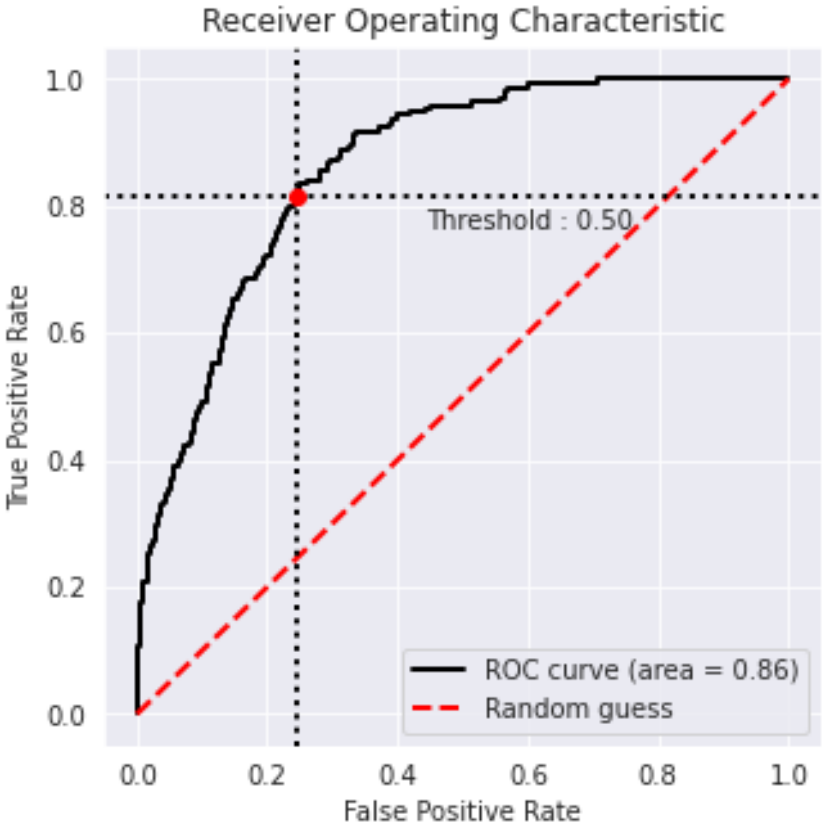

Figura 4.1: *AUROC*

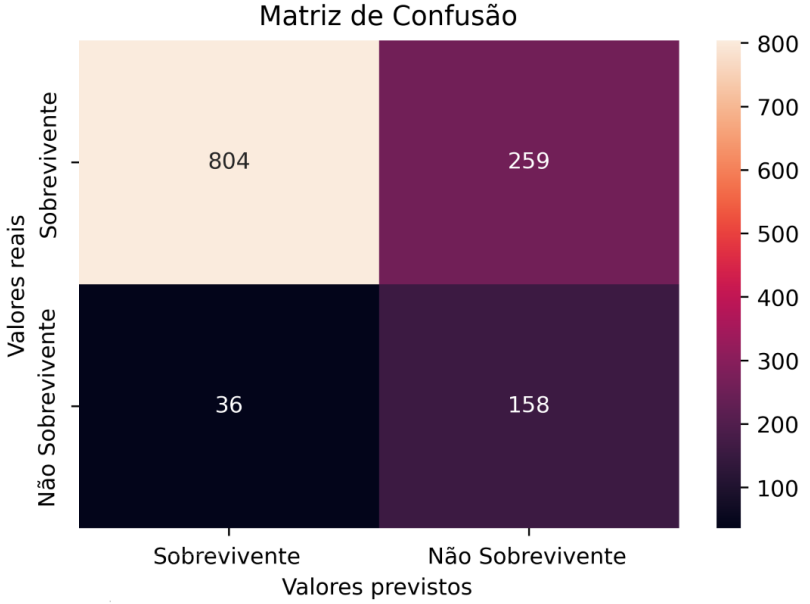

Figura 4.2: *Matriz de confusão*

O modelo apresentou um alto valor de *recall*, considerado como uma das mais importantes métricas para os estudos médicos, uma vez que se deseja errar o menor número possível de casos positivos, o que se traduz em um *recall* alto [50]. No entanto, o modelo apresentou um valor médio de *precision*.

Normalmente, *recall* e *precision* estão em "tensão", ou seja, aumentar *precision* tipicamente reduz *recall*, e vice-versa. É possível alterar seus valores mudando o *Threshold* para obter um melhor equilíbrio.

O equilíbrio entre os dois deve ser baseado no caso de uso médico e nos requisitos associados. Quando o objetivo é identificar doenças graves ou mortalidade, um valor alto de *recall* é desejado.

4.2 Comparação com outros trabalhos

O objetivo dessa seção é apresentar o resultado de outros trabalhos relacionados à predição de mortalidade de pacientes, comparando com o modelo aqui apresentado.

- ***Using machine learning methods to predict in-hospital mortality of sepsis patients in the ICU*** [51]. Utilizando o MIMIC-III, esse estudo considerou 86 variáveis preditórias, consistindo de dados demográficos, exames laboratoriais e comorbidades, com o objetivo de predizer a mortalidade hospitalar de pacientes com sepse. Para isso, selecionaram os algoritmos *least absolute shrinkage and selection operator (LASSO), random forest (RF), gradient boosting machine (GBM)* e *logistic regression (LR)*, obtendo *AUROC* de 0.829, 0.829, 0.845 e 0.833, respectivamente. O *AUROC* do modelo aqui proposto foi superior.
- ***Early hospital mortality prediction using vital signals*** [52]. Também utilizando o MIMIC-III, esse estudo propôs um método para prever a mortalidade usando características extraídas dos sinais cardíacos de pacientes na primeira hora de admissão na UTI. As variáveis preditórias alimentaram oito classificadores: *decision tree, linear discriminant, logistic regression, support vector machine (SVM), random forest, boosted trees, Gaussian SVM* e *K-nearest neighborhood (K-NN)*. O classificador *decision tree* obteve os melhores resultados, com *F1-score* e *AUROC* iguais a 0.91 e 0.93, respectivamente. Portanto, esse trabalho obteve uma melhor capacidade preditória em relação ao modelo aqui proposto.
- ***A Comparison of Intensive Care Unit Mortality Prediction Models through the Use of Data Mining Techniques*** [53]. Este estudo foi feito para desenvolver um modelo de previsão de mortalidade em UTI construído com dados

do *University of Kentucky Hospital (UKH)* e avaliar se o desempenho de várias técnicas de mineração de dados, como *artificial neural network (ANN), support vector machine (SVM)* e *decision tree (DT)*, superam o modelo estatístico convencional de regressão logística. O algoritmo *decision tree (DT)* obteve o melhor resultado, com *AUROC* de 0.892, seguido do *support vector machine* (*AUROC* de 0.876) e *artificial neural network* (*AUROC* de 0.874). O desempenho dos modelos desse estudo se assemelham ao do modelo aqui proposto (*AUROC* de 0.867).

- ***Intensive Care Unit Mortality Prediction: An Improved Patient-Specific Stacking Ensemble Model*** [54]. Também utilizando o MIMIC-III, o estudo utilizou diversos classificadores, incluindo *discriminant analysis, decision tree (DT), multilayer perceptron, k-nearest neighbor*, e *logistic regression (LR)*. Em seguida, um classificador foi construído e otimizado com base na fusão desses cinco classificadores. Os resultados indicam que o modelo superou as abordagens de última geração em termos de acurácia (94,4%), *F1-score* (93,7%), *precision* (96,4%), *recall* (91,1%) e *AUROC* (93,3%). Esse trabalho obteve um dos melhores resultados na literatura.

Capítulo 5

Conclusões

Usando o banco de dados MIMIC-III, esse trabalho desenvolveu com sucesso um modelo de rede neural com boa capacidade para a previsão de mortalidade de pacientes internados na UTI, modelo este que pode ser aprimorado e refinado para ser utilizado como forma de auxílio das equipes médicas para tomada de decisões.

A maior carga do trabalho recaiu sobre a seleção, extração e pré-processamento do conjunto de dados. As variáveis preditoras escolhidas são facilmente obtidas e podem ser utilizadas para outros objetivos, procurando auxiliar no prognóstico ou diagnóstico de pacientes.

Por fim, a comparação com resultados e desempenho relatados na literatura esclarece que nenhum algoritmo em específico supera os demais em termos de previsão. Portanto, é um desafio fornecer uma previsão precisa da mortalidade hospitalar.

5.1 Trabalhos futuros

O trabalho aqui descrito sofreu várias simplificações para que fosse viável sua realização no tempo disponível e com a quantidade de pessoas disponível. Porém, mesmo sendo simplificado, os resultados não devem ser descartados, pois servem para motivar ou guiar trabalhos mais complexos.Um trabalho futuro possível seria a captura de mais *features*, incluindo as notas escritas por enfermeiros, que descrevem com mais detalhes a condição do paciente.

Além disso, outra possibilidade seria a utilização de outros algoritmos de *Machine Learning* no mesmo conjunto de dados, possibilitando a comparação de seus desempenhos para a escolha do modelo mais adequado.

Outro estudo possível seria utilizar os mesmos métodos, porém mudando a janela de tempo analisada, como sinais medidos nas primeiras 12 horas ou nas primeiras 48 horas.

Por fim, um outro trabalho futuro interessante seria o desenvolvimento de uma interface web disponibilizada para os profissionais que trabalham na UTI, possibilitando que eles insiram as variáveis necessárias e recebam como resposta a probabilidade de morte do paciente em questão.

Referências

[1] Pollard T. J. Shen L. Lehman L. H. Feng M. Ghassemi M. Moody B. Szolovits P. Celi L. A. Mark R. G. Johnson, A. E. W. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 2016. vi, vii, 18

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012. 1

[3] Richard Socher, Eric Huang, Jeffrey Pennin, Christopher D Manning, e Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in neural information processing systems*, 24, 2011. 1

[4] Warren S. Mcculloch e Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Journal of Symbolic Logic*, 9(2):49–50, 1943. 1

[5] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, e Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017. 1

[6] Jie-Zhi Cheng, Dong Ni, Yi-Hong Chou, Jing Qin, Chui-Mei Tiu, Yeun-Chung Chang, Chiun-Sheng Huang, Dinggang Shen, e Chung-Ming Chen. Computer-aided diagnosis with deep learning architecture: Applications to breast lesions in us images and pulmonary nodules in ct scans. *Scientific reports*, 6(1):24454–24454, 2016. 1

[7] Você sabe o que é uma uti e sua importância na covid-19? `https://coronavirus.saude.mg.gov.br/blog/61-o-que-e-uma-uti#:~:text=Esse%20tipo%20de%20instala%C3%A7%C3%A3o%20hospitalar,pacientes%20que%20precisam%20de%20respiradores.` Accessed: 2022-05-09. 1

[8] Resolução cfm nº 2.271/2020. `https://sistemas.cfm.org.br/normas/visualizar/resolucoes/BR/2020/2271`. Accessed: 2022-05-09. 1

[9] Jean-Roger Le Gall, Stanley Lemeshow, e Fabienne Saulnier. A New Simplified Acute Physiology Score (SAPS II) Based on a European/North American Multicenter Study. *JAMA*, 270(24):2957–2963, 12 1993. 1

[10] William A Knaus, Elizabeth A Draper, Douglas P Wagner, e Jack E Zimmerman. Apache ii: a severity of disease classification system. *Critical care medicine*, 13(10):818–829, 1985. 1

[11] William A Knaus, Douglas P Wagner, Elizabeth A Draper, Jack E Zimmerman, Marilyn Bergner, Paulo G Bastos, Carl A Sirio, Donald J Murphy, Ted Lotring, Anne Damiano, et al. The apache iii prognostic system: risk prediction of hospital mortality for critically iii hospitalized adults. *Chest*, 100(6):1619–1636, 1991. 1

[12] Jack E Zimmerman, Andrew A Kramer, Douglas S McNair, e Fern M Malila. Acute physiology and chronic health evaluation (apache) iv: hospital mortality assessment for today's critically ill patients. *Critical care medicine*, 34(5):1297–1310, 2006. 1

[13] Phil Simon. *Too big to ignore: the business case for big data*, volume 72. John Wiley & Sons, 2013. 3

[14] Machine learning, explained. `https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained`. Accessed: 2022-05-09. 3

[15] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009. 3

[16] The perceptron. `https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Neuron/index.html#:~:text=Neural%20Networks%20%2D%20Neuron&text=The%20perceptron%20is%20a%20mathematical,are%20represented%20as%20numerical%20values.` Accessed: 2022-05-09. 3

[17] The concept of artificial neurons (perceptrons) in neural networks. `https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc`. Accessed: 2022-05-09. 4

[18] Everything you need to know about "activation functions" in deep learning models. `https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84` Accessed: 2022-05-09. 4

[19] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, e Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018. 5

[20] Fantastic activation functions and when to use them. `https://towardsdatascience.com/fantastic-activation-functions-and-when-to-use-them-481fe2` Accessed: 2022-05-09. 5

[21] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 6

[22] Redes neurais, perceptron multicamadas e o algoritmo backpropagation. https://medium.com/ensina-ai/redes-neurais-perceptron-multicamadas-e-o-algoritmo-backpropagation-eaf89778f5b8. Accessed: 2022-05-09. 6

[23] Loss and loss functions for training deep learning neural networks. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. Accessed: 2022-05-09. 7

[24] Krzysztof C Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical programming*, 90(1):1–25, 2001. 7

[25] Leslie N Smith. Cyclical learning rates for training neural networks. Em *2017 IEEE winter conference on applications of computer vision (WACV)*, páginas 464–472. IEEE, 2017. 8

[26] Ilya Sutskever, James Martens, George Dahl, e Geoffrey Hinton. On the importance of initialization and momentum in Deep Learning. Em *International conference on machine learning*, páginas 1139–1147, 2013. 9

[27] Moritz Hardt, Benjamin Recht, e Yoram Singer. Train faster, generalize better: stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015. 9

[28] Aurélien Géron. Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligentsystems. Paperback, April 2017. 9

[29] Diederik P Kingma e Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 9

[30] David E. Rumelhart e James L. McClelland. *Learning Internal Representations by Error Propagation*, páginas 318–362. 1987. 10

[31] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004. 10

[32] Underfitting. https://www.ibm.com/cloud/learn/underfitting/. Accessed: 2022-05-09. 11

[33] Lutz Prechelt. *Early Stopping - But When?*, páginas 55–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. 11

[34] Machine learning. https://www.ibm.com/cloud/learn/machine-learning#toc-machine-le-K7VszOk6. Accessed: 2022-03-21. 12

[35] Ian Goodfellow, Yoshua Bengio, e Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 12

[36] David Martin Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011. 14

[37] The pandas development team. pandas-dev/pandas: Pandas, February 2020. 17

[38] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, e Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. 17

[39] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 17

[40] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. 17

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, e E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 17

[42] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, e Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 17

[43] François Chollet et al. Keras. `https://keras.io`, 2015. 17

[44] The glasgow structured approach to assessment of the glasgow coma scale. `https://www.glasgowcomascale.org/`. Accessed: 2022-03-30. 23

[45] L. O. Hall N. V. Chawla, K. W. Bowyer e W. P. Kegelmeye. Smote: Syn- thetic minority over-sampling technique. *Jounal of Artificial Intelligence Re- search*, 2002. 31

[46] I. Tomek. Two modifications of cnn. *IEEE Transactions on Systems Man and Communications*, 1976. 31

[47] Miroslav Kubat e Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. Em *In Proceedings of the Fourteenth International Conference on Machine Learning*, páginas 179–186. Morgan Kaufmann, 1997. 31

[48] Gustavo Batista, Ana Bazzan, e Maria-Carolina Monard. Balancing training data for automated annotation of keywords: a case study. páginas 10–18, 01 2003. 31

[49] James Bergstra e Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. x, 33

[50] Steven A. Hicks, Inga Strümke, Vajira Thambawita, Malek Hammou, Michael A. Riegler, Pål Halvorsen, e Sravanthi Parasa. On evaluation metrics for medical applications of artificial intelligence. *Scientific Reports*, 2022. 38

[51] Guilan Kong, Ke Lin, e Yonghua Hu. Using machine learning methods to predict in-hospital mortality of sepsis patients in the icu. *BMC medical informatics and decision making*, 20(1):1–10, 2020. 38

[52] Reza Sadeghi, Tanvi Banerjee, e William Romine. Early hospital mortality prediction using vital signals. *Smart Health*, 9-10:265–274, 2018. CHASE 2018 Special Issue. 38

[53] Park Rae Woong Kim Sujin, Kim Woojae. A comparison of intensive care unit mortality prediction models through the use of data mining techniques. *hir*, 17(4):232–243, 2011. 38

[54] Nora El-Rashidy, Shaker El-Sappagh, Tamer Abuhmed, Samir Abdelrazek, e Hazem M. El-Bakry. Intensive care unit mortality prediction: An improved patient-specific stacking ensemble model. *IEEE Access*, 8:133541–133564, 2020. 39