
MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIANO - CAMPUS RIO VERDE

# CLASSIFICAÇÃO DE DADOS DE COVID-19 USANDO REDE NEURAL CONVOLUCIONAL

ADELSON NASCIMENTO LIMA NETO

Rio Verde

2023


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO - CAMPUS RIO VERDE
BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

# CLASSIFICAÇÃO DE DADOS DE COVID-19 USANDO REDE NEURAL CONVOLUCIONAL

ADELSON NASCIMENTO LIMA NETO

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde ligado ao Ministério da Educação, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Prof.ª Heyde Francielle do Carmo Franca

Rio Verde

2023

INSTITUTO FEDERAL
Goiano

Repositório Institucional do IF Goiano - RIIF Goiano
Sistema Integrado de Bibliotecas

# TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

## PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

### IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Adelson Nascimento Lima Neto

Matrícula:
2016102192010110

Título do trabalho:
CLASSIFICAÇÃO DE DADOS DE COVID-19 USANDO REDE NEURAL CONVOLUCIONAL

### RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 13 /03 /2023

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

### DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Guarulhos — Local
13 /03 /2023 — Data

Adelson Nascimento Lima Neto
Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

Heyde Francielle do Carmo França
Assinatura do(a) orientador(a)

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 3/2023 - GEPTNM-RV/DE-RV/CMPRV/IFGOIANO

**ATA DE DEFESA DE TRABALHO DE CURSO**

Ao(s) 28 do mês de fevereiro de 2023, às 18 horas, reuniu-se a banca examinadora composta pelos docentes: Heyde Francielle do Carmo França, Rafael Carvalho de Mendonca, Andrea Barboza Proto Sardi, para examinar o Trabalho de Curso intitulado "CLASSIFICAÇÃO DE DADOS DE COVID-19 USANDO REDE NEURAL CONVOLUCIONAL" do(a) estudante Adelson Nascimento Lima Neto, Matrícula nº 2016102192010110 do Curso de Ciência da Computação do IF Goiano – Campus Rio Verde. A palavra foi concedida ao(a) estudante para a apresentação oral do TC, houve arguição do(a) candidato pelos membros da banca examinadora. Após tal etapa, a banca examinadora decidiu pela APROVAÇÃO do(a) estudante. Ao final da sessão pública de defesa foi lavrada a presente ata que segue assinada pelos membros da Banca Examinadora.

*(Assinado Eletronicamente)*

Heyde Francielle do Carmo França

Orientador(a)

*(Assinado Eletronicamente)*

Rafael Carvalho de Mendonca

Membro

*(Assinado Eletronicamente)*

Andrea Barboza Proto Sardi

Membro

Documento assinado eletronicamente por:

- **Rafael Carvalho de Mendonca**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/03/2023 09:18:00.
- **Heyde Francielle do Carmo Franca**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 28/02/2023 18:58:12.

Este documento foi emitido pelo SUAP em 28/02/2023. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 471333
Código de Autenticação: 1ac4bdbd3a

Documento assinado digitalmente
gov.br
ANDREA BARBOZA PROTO SARDI
Data: 10/03/2023 13:18:48-0300
Verifique em https://verificador.iti.br

Dedico esse trabalho a Deus e minha família, que me deu todo o suporte e apoio para concluir essa longa jornada desde 2012.

## AGRADECIMENTOS

Gostaria de agradecer, em primeiro lugar, a Deus pela força que me deu nesses últimos meses para que eu conseguisse finalizar a escrita desse projeto e meu trabalho.

À minha família que me aguentou esse tempo e nunca me deixou sem suporte, mesmo na perda da minha mãe e meu bloqueio na escrita. Além do meu esposo que não cansava de me ver batendo a cabeça no teclado para ver se saia uma frase.

E um agradecimento especial a minha orientadora Heyde que teve uma paciência divina e não desistiu de mim.

*Transportai um punhado de terra todos os dias e fareis uma montanha (Confúcio).*

## RESUMO

NASCIMENTO L NETO, Adelson. CLASSIFICAÇÃO DE DADOS DE COVID-19 USANDO REDE NEURAL CONVOLUCIONAL. 2023. 37 f. Trabalho de Conclusão de Curso – Bacharelado em Ciência da Computação, Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde. Rio Verde, 2023.

No ano 2020 o mundo foi surpreendido com o surgimento de uma pandemia, identificada como Covid-19. Começando na China e se espalhando pelo globo em pouco tempo com sintomas parecidos com uma gripe, sua rápida transmissão e evolução dos casos surpreenderam a comunidade médica. Durante estes dois anos, muitos médicos e cientistas se esforçaram para compreender os padrões da doença e um dos métodos utilizados foi o uso de redes neurais que utiliza técnicas de inteligência artificial para identificar os padrões do vírus. Para contribuir com as pesquisas desenvolvidas durante a pandemia, este estudo visa avaliar, por via de um estudo de caso, uma rede neural convolucional 1D para a classificação de pacientes infectados ou não pelo vírus. Utilizando uma base de dados disponibilizada pelo Hospital Sírio Libanês de São Paulo. Os objetivos específicos são contextualizar o cenário da pandemia de Covid-19, introduzir e conceitualizar uma aprendizagem profunda, descrever e explicar o que é uma rede neural e o uso de redes neurais convolucionais 1D, e finalmente aplicar a rede neural convolucional 1D à classificação dos dados Covid-19 para a previsão de resultados. Após o treinamento do modelo de rede neural desenvolvido foi possível perceber que a classificação obteve uma acurácia acima de 90%, porém quando se analisa a classe minoritária esse valor se aproxima de 0, isso se deve ao grande desbalanceamento dos dados. No entanto, ao final do desenvolvimento, foi constatado que o uso dessa base de dados, que possui uma maior quantidade de dados negativos do que positivos, não atingiu uma taxa de acerto suficiente para os dados positivos, ficando evidente a necessidade de obtenção de mais dados dessa classe para poder melhorar os resultados de classificação do modelo.



**Palavras-chave**: Redes Neurais Convolucionais. Aprendizado de Máquina. Classificação de dados de COVID.

**ABSTRACT**

NASCIMENTO L NETO, Adelson. COVID DATA CLASSIFICATION BASED USING CONVOLUTIONAL NEURAL NETWORK. 2023. 37 f. Trabalho de Conclusão de Curso – Bacharelado em Ciência da Computação, Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde. Rio Verde, 2023.

In the year 2020 the world was surprised by the emergence of a pandemic, identified as Covid-19. Starting in China and spreading across the globe in a short period of time with flu-like symptoms, its rapid transmission and evolution of cases surprised the medical community. During these two years, many doctors and scientists have struggled to understand the patterns of the disease, and one of the methods used was the use of neural networks that use artificial intelligence techniques to identify the patterns of the virus. To contribute to the research developed during the pandemic, this study aims to evaluate, via a case study, a 1D convolutional neural network for the classification of patients infected or not by the virus. Using a database made available by Hospital Sırio Libanes de Sao Paulo. The specific objectives are to contextualize the Covid-19 pandemic scenario, introduce and conceptualize deep learning, describe and explain what a neural network is and the use of 1D convolutional neural networks, and finally apply the 1D convolutional neural network to the classification of Covid-19 data for outcome prediction. After training the neural network model developed, it was possible to see that the classification obtained an accuracy above 90%, but when analyzing the minority class this value approaches 0, this is due to the large unbalance of the data. However, at the end of the development, it was found that the use of this database, which has a larger amount of negative data than positive, did not reach a sufficient hit rate for positive data, making it is evident that it is necessary to obtain more data of this class in order to improve the model's classification results.
**Keywords**:Convolutional Neural Networks. Machine Learning. Classification of COVID data.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ADAM | Adaptive Moment Estimation |
| AUC | Area under the curve |
| CNN | Convolutional Neural Network |
| FN | False Negative |
| FP | False Positive |
| GPU | Graphical Processing Units |
| IA | Inteligência Artificial |
| KNN | K-Nearest Neighbor |
| MLP | Multi-Layer Perceptron |
| OMS | Organização Mundial de Saúde |
| TN | True Negative |
| TP | True Positive |

**SUMÁRIO**

# 1 INTRODUÇÃO

Ao passar dos anos, o mundo foi acometido por desastres naturais e doenças que eram impossíveis de se prever ou proteger por conta da tecnologia existente nas suas respectivas épocas. Desde a peste bubônica até a gripe suína, houveram muitas mortes pela falta de tecnologias capazes de auxiliar no combate e prevenção (CHONG, 2022).

O mesmo aconteceu com a pandemia do Coronavírus no dia 31 de dezembro de 2019 (GOV, 2022). Mesmo com todos os avanços na área da medicina e computação, mais de 690 mil pessoas só aqui no Brasil perderam suas vidas (GOV, n.d.). As técnicas de aprendizado de máquina podem ser aplicadas a diferentes áreas por diferentes motivos, inclusive na área da saúde, como na pandemia de Covid-19, doença causada pelo coronavírus (SARS-CoV-2) e identificada pela primeira vez na China, em dezembro de 2019 (OLIVEIRA PATRICK FRANCISCO; C MARA, 2019). O interesse em trabalhar especialmente com esse tipo de dado foi justamente pelo impacto que esta doença teve em todo o mundo durante o século XXI, deixando um rastro de morte e mudanças impactantes na estrutura de vida das pessoas.

Com toda a urgência e problemas estruturais da saúde, tornou-se um problema a coleta de dados dos pacientes completamente ou eficiente. Muitos dados incompletos ou com dados brancos, ou inválidos, além de alguns casos não haviam dados sequer dos pacientes. Mesmo assim houve tentativas de diagnóstico, via sintomas e características clínicas coletadas, na tentativa de identificar se os pacientes estavam contaminados pelo vírus utilizando a inteligência artificial e redes neurais (WANG et al., 2021).

Entre as várias áreas que se beneficiaram com as novas pesquisas em rede neurais, a medicina foi uma delas. Sendo utilizada para identificar tumores, entre outras doenças (NETO, 2017). Presentemente, em que enfrentamos a pandemia da Covid-19, os pesquisadores observaram que a inteligência artificial poderia aprender os padrões dessa nova doença para auxiliar no rápido diagnóstico da mesma.

A aplicação de redes neurais convolutivas 1D para classificação dos dados de doentes suspeitos de infecção pela Covid pode ser uma ferramenta importante para auxiliar na identificação do padrão de resultados de testes de outras doenças e em possíveis emergências futuras, como a pandemia da Covid-19.

A finalidade deste estudo consiste em avaliar o desempenho de uma rede neural convolucional 1D (CNN) na classificação de pacientes com Covid-19, utilizando uma base de dados clínicos e dos sintomas de 5643 indivíduos. Serão utilizados conceitos de *Deep Learning*, mais especificamente redes neurais convolutivas, na tentativa de medir o desempenho do modelo proposto através das métricas escolhidas e seus resultados.

O restante desse trabalho está organizado da seguinte forma: no segundo capítulo serão descritos os principais conceitos importantes para o entendimento da proposta. Em seguida, no terceiro capítulo, descrição de três trabalhos relacionados com o uso de CNNs em dados tabulares e/ou da saúde. No quarto capítulo as ferramentas e métodos utilizados para o desenvolvimento da rede neural. Para que, no quinto, haja a descrição completa da rede e seus resultados obtidos, e no sexto e último capítulo finaliza com a conclusão do trabalho.

# 2 REVISÃO DE LITERATURA

## 2.1 A PANDEMIA COVID-19

O que começou como uma epidemia, limitada inicialmente à China, tornou-se uma pandemia global. Em outubro de 2020, a Organização Mundial de Saúde (OMS) comunicou quase 45 milhões de casos confirmados e mais de 1 milhão de mortes por Covid-19. A doença foi detectada em mais de 200 países e territórios, tendo os Estados Unidos, Itália e Espanha registrado os surtos mais intensos fora da China. O verdadeiro número de infecções e mortes está propenso a ser espantosamente superior no final da pandemia (MORENS, 2020).

O governo chinês enfrentou o primeiro surto, colocando Wuhan e cidades vizinhas sob quarentena de fato, cobrindo cerca de 50 milhões de pessoas na região de Hubei (LE, 2020). As autoridades chinesas notificaram a OMS no início de janeiro sobre uma "pneumonia grave de etiologia desconhecida" — a ciência fala de uma misteriosa nova doença respiratória — havia sido descoberta em Wuhan, capital da província de Hubei, com 11 milhões de habitantes (ACTER, 2020).

Em janeiro de 2020, foi relatado o primeiro caso europeu na França. Este caso tinha um registro de viagens à China. No mesmo mês, a Alemanha também comunicou casos, ligados a uma pessoa de visita da China. Neste período a OMS declarou o primeiro surto do novo Coronavírus como sendo uma Emergência de Saúde Pública de preocupação internacional (ACTER, 2020).

Em março desse mesmo ano, a OMS reconheceu a existência da alta taxa de contaminação do novo vírus, decretando assim que o mundo se encontrava em uma pandemia (MORENS, 2020). No segundo semestre de 2020, as vacinas foram desenvolvidas mundialmente e a população começou finalmente a ser imunizada, resultando numa diminuição gradual do número de casos confirmados (LE TUNG THANH; CRAMER, 2020).

## 2.2 INTELIGÊNCIA ARTIFICIAL

O termo Inteligência Artificial (IA) surgiu logo após a Segunda Guerra Mundial, proposto em pesquisas realizadas nas áreas de ciências e engenharia (RUSSEL STUART; NORVIG, 2009). Hoje ela cobre uma enorme variedade de áreas do conhecimento, sejam elas desde a aprendizagem e percepção até jogar xadrez, provar teoremas matemáticos e auxiliar no diagnóstico de doenças.

Para auxiliar no entendimento dos conceitos de IA, (RUSSEL STUART; NORVIG, 2009) apresentam quatro categorias. Sendo as duas primeiras destinadas a medirem o sucesso em termos de fidelidade ao desempenho humano, enquanto as outras medem o sucesso em termos de uma medida de desempenho ideal, chamada racionalidade.

A primeira categoria remete ao Teste de Turing e seu esforço de criar uma máquina que possa ser comparada ao modo humano de pensar. Sendo hoje um dos testes mais utilizados a mais de 70 anos de sua criação. Na obra de Kurzweil (1990), ele define que a IA é "A arte de criar máquinas que executam funções que requerem inteligência quando executadas por pessoas". Pensamento esse sendo acompanhado por outros pesquisadores da época.

A segunda envolve a criação de sistemas computacionais que possuem o discer-

nimento de tomar decisões baseadas na forma geral do humano pensar através de vários estudos da psique humana e, consequentemente, suas ações. O ideal consiste em transferir o pensamento humano e atividades como aprendizagem e tomadas de decisões a um sistema computacional. Surgindo assim o campo da ciência cognitiva que reúne os modelos de IA e técnicas da psicologia (RUSSEL STUART; NORVIG, 2009).

A terceira consiste na transformação de todas as formas de conhecimento, formal ou informal, em notações lógicas para assim construir um programa que se torne a base de um sistema inteligente com a habilidade de resolver qualquer problema lógico, na prática. Na obra de Winston (1992), ele propôs "O estudo dos cálculos que tornam possível perceber, raciocinar e agir". Sendo estudado pelo campo da lógica e seus logicistas, esses estudos derivam-se dos estudos de Aristóteles (RUSSEL STUART; NORVIG, 2009).

A última representa a investigação de como a IA auxiliaria no desenvolvimento de agentes (módulos que atuam) capazes de executar atividades autônomas e conservar seu estado por muito tempo. Para poderem realizar suas tarefas e sejam considerados agente racionais, é necessário que as soluções para os problemas abordados possam ser deduzidos ao ponto de se aproximarem ao máximo do objetivo esperado. Por isso, a reprodução dos conhecimentos adquiridos e o ato de raciocinar, possibilitam a melhor tomada de decisão e compreensão da linguagem natural utilizada na sociedade (RUSSEL STUART; NORVIG, 2009).

Ao final dos anos 80, Yan Lecun construiu uma rede neural convolutiva, que possuía uma estrutura multicamada cuja arquitetura de ligações é inspirada pela do córtex visual dos mamíferos. Por exemplo, cada elemento está ligado apenas a um pequeno número de elementos vizinhos na camada anterior (O'SHEA KEIRON; NASH, 2015).

Os estudos de Yan Lecun foi utilizada na construção de um sistema automático de leitura de cheques que se espalhou por todo o mundo quase uma década depois de sua criação. No final dos anos 90, este sistema estava sendo utilizado para ler entre 10% e 20% de todos os cheques emitidos nos Estados Unidos. No entanto, o método era difícil de ser implementado com os computadores da época, e apesar do sucesso da automação, as redes convolucionais e as redes neurais, em geral, foram abandonadas pela comunidade científica entre 1997 e 2012 (O'SHEA KEIRON; NASH, 2015).

Ao final desse período ocorreram três eventos que alteraram subitamente a situação. Primeiro, as GPUs (*Graphical Processing Units*), capazes de mais de um trilião de operações por segundo, tornaram-se disponíveis a um preço acessível. Estes poderosos processadores, especializados em renderizar gráficos em videogames, provaram conseguir processar algoritmos de rede neural. Em segundo lugar, experiências realizadas simultaneamente na Microsoft, Google e IBM com a ajuda do laboratório de Geoff Hinton, mostraram que o uso de redes neurais profundas no reconhecimento de fala poderia reduzir pela metade as taxas de erro. O último evento estava relacionado as redes neurais convolucionais que quebraram vários recordes no reconhecimento de imagens (AGHDAM HAMED HABIBI; HERAVI, 2017).

O acontecimento mais significativo foi a vitória da equipe de Toronto no concurso de reconhecimento de objetos "ImageNet" com a impressionante taxa de erro 50% menor que a do segundo colocado, tornando-se um ponto de virada para as redes neurais. em virtude disso, a maioria das equipes de pesquisa de fala e visão abandonaram os seus métodos preferidos e mudaram para redes neurais convolucionais e outras redes neurais. Causando um imediato investimento em pesquisa e desenvolvimento de aprendizagem profundo pela comunidade (OLIVEIRA PATRICK FRANCISCO; C MARA, 2019).

### 2.2.1 APRENDIZADO SUPERVISIONADO vs NÃO SUPERVISIONADO

Para treinar uma rede neural para que ela consiga executar alguma tarefa, existem algumas abordagens de aprendizado de máquina. Primeiramente há o aprendizado supervisionado, no qual as características de entrada e saída são conhecidas pelo modelo e visto na figura 1. A conclusão do aprendizado se dá quando a diferença de erro, entre o que foi previsto pelo modelo e o que é realmente o valor correto, é menor que um valor mínimo pré-estabelecido (RAJ, n.d.).

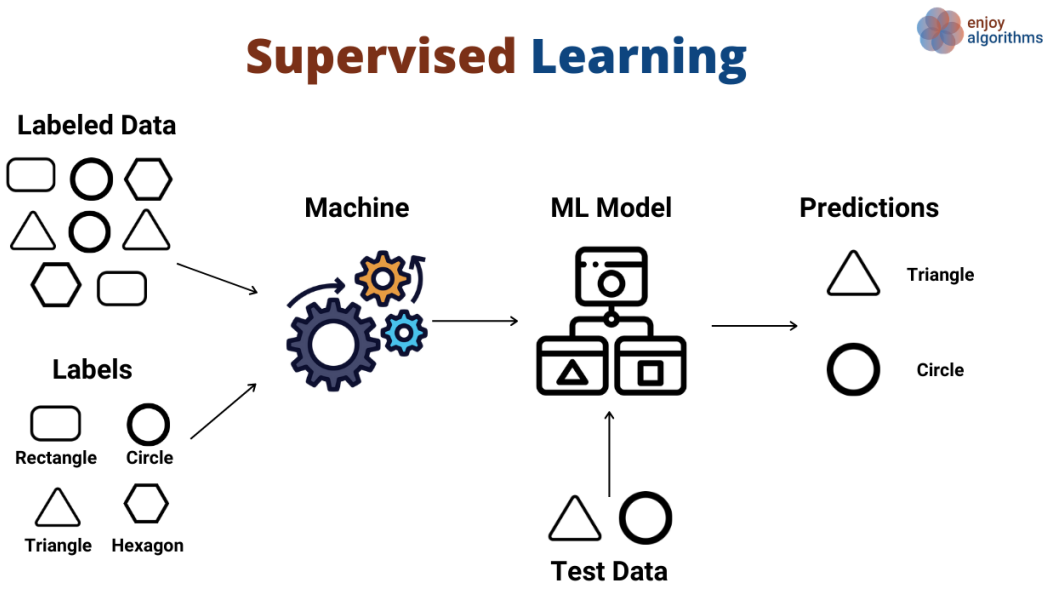



Figura 1 – Exemplo de aprendizado supervisionado (RAJ, n.d.)

Já o não supervisionado, mostrado na figura 2, se diferencia no quesito das características de saída. Apenas as de entrada são de conhecimento do modelo, esperando assim que os resultados previstos pelo modelo sejam semelhantes entre si (RAJ, n.d.). Geralmente é utilizado em bases com dados brutos não nominados para que seus resultados sejam agrupados e a análise seja realizada em seguida. Além da tentativa de descobrir se há padrões nos dados da base (UNSUPERVISED, 2022).

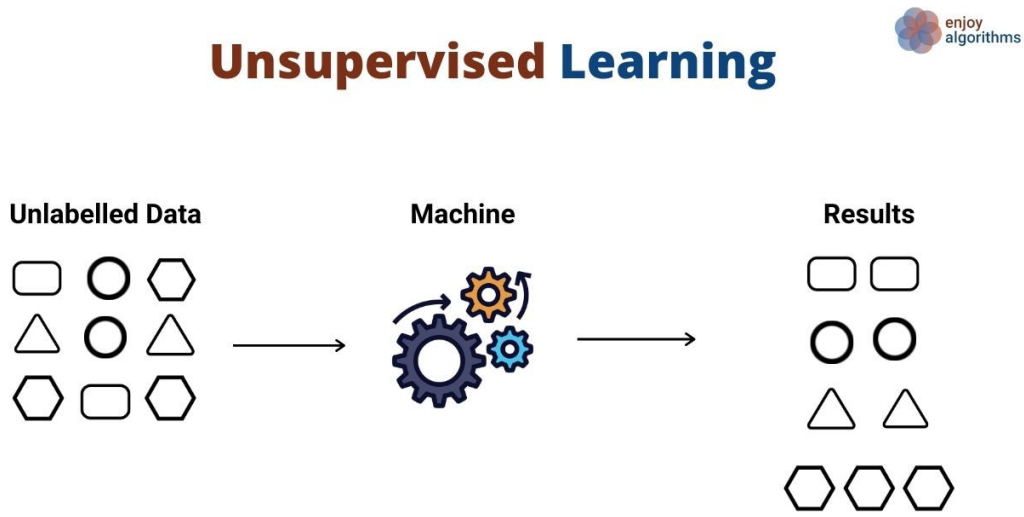



Figura 2 – Exemplo de aprendizado não supervisionado (RAJ, n.d.)

## 2.3 REDES NEURAIS ARTIFICIAIS

O conceito de redes neurais artificiais teve origem em 1943, proposto por Warren McCullough, um neuro físico, e Walter Pitts, um matemático. Ambos eram pesquisadores da Universidade de Chicago e a pesquisa foi apresentada pela primeira vez na revista Journal Brain Theory. Neste artigo os pesquisadores explicam que a unidade básica de ativação cerebral é o estímulo dos neurônios (FLECK, 2016).

Ao desenvolver esta teoria, os pesquisadores começaram a ver a possibilidade de recriar o funcionamento dos neurônios. O ano de 1957 foi outro marco no avanço da pesquisa de inteligência artificial com a invenção do perceptron, o qual é um algoritmo de aprendizagem de classificadores binários que representa o mais antigo algoritmo de aprendizagem de máquinas. A partir deste algoritmo, as máquinas poderão reconhecer objetos em imagens (CASTRO F. C. C. C.; DE CASTRO, 2001).

Hoje, o termo perceptron também se refere a uma rede de propagação direta de camada única. Na época, os computadores não eram suficientemente potentes para processar o volume de dados necessários para executar a rede neural artificial. Entretanto, o caminho para ele era conhecido e era apenas uma questão de tempo até que os pesquisadores colocassem o processo de aprendizagem em funcionamento. Devido à falta de recursos, a pesquisa em aprendizagem profunda estagnou por vários anos (RAUBER, 2005).

Com o surgimento do *Big Data*, alcançou-se a capacidade de criação de redes neurais complexas e estruturadas. Tendo a habilidade de superar o desempenho humano de reconhecer imagens com precisão. Com esses avanços, é possível perceber que a variedade de áreas onde as redes neurais podem atuar são inúmeras. O que aumenta as chances dos estudiosos de terem avanços significativos em suas áreas (FLECK, 2016).

Uma rede neural é uma rede computacional (ou algoritmo) cuja saída reproduz ou prevê o mais próximo possível o comportamento de qualquer processo, conforme os fatores que determinam esse comportamento. Por "processo" entendemos qualquer sistema, natural ou artificial, e por "fator" qualquer quantidade que influencie o processo.

Assim, uma rede neural consegue prever a evolução da temperatura de um forno em função da intensidade da corrente elétrica na resistência de aquecimento, a velocidade

de um veículo em função do deslocamento do pedal do acelerador e da inclinação da estrada, o preço de um título em função das taxas de câmbio do dólar e do iene, assim como seus preços passados (OSÓRIO FERNANDO S.; BITTENCOURT, 2000).

O cálculo realizado por uma rede neural envolve números chamados parâmetros ou pesos de rede, que devem ser calculados para tornar a imitação e a previsão o mais precisa possível. Para fazer este ajuste, são usados exemplos do comportamento a ser imitado: isto é chamado de "aprender" a rede neural a partir de exemplos cuidadosamente escolhidos. Uma vez concluída a aprendizagem, a rede deve ser capaz não apenas de reproduzir os exemplos de comportamentos aprendidos, mas sobretudo de prever com precisão os comportamentos não aprendidos: esta é a capacidade de "generalização" (FLECK, 2016).

As redes neurais, portanto, resolvem um problema estatístico: fundamentalmente, o problema que elas abordam não difere do dos institutos de pesquisa que, após questionar uma amostra representativa da população, generalizam as respostas para toda a população (RAUBER, 2005).

A capacidade das redes neurais de imitar e prever o comportamento, após aprender pelo exemplo, permite que elas tenham aplicações em uma grande variedade de campos, entre uma das particularidades da rede está entre simular e reconhecer como apresentado na lista abaixo (OSÓRIO FERNANDO S.; BITTENCOURT, 2000):

- Simular em um computador e assim prever o comportamento de processos complexos, sejam eles industriais (robôs, plantas industriais, circuitos eletrônicos, etc.), sejam eles relacionados à vida cotidiana (carros, sistemas de ar condicionado, etc.), à atividade (marketing, investimento, seleção de informações, etc.), ou governem ambientes naturais, ou sociais (ecossistemas, sistemas sociais ou políticos, etc.), etc.;
- Reconhecer formas, sinais, eventos (leitura automática de códigos postais ou cheques, reconhecimento de imagem, reconhecimento de voz, etc.).

Estas aplicações, apesar de sua diversidade, têm um fundamento matemático e estatístico comum: o objetivo é sempre obter a capacidade de generalização mais satisfatória possível, dada a qualidade e a relevância dos exemplos utilizados no curso do estudo. As redes neurais permitem resolver este tipo de problema eficientemente e precisa; no entanto, seria um exagero afirmar que elas representam uma forma de inteligência (FLECK, 2016).

Cada neurônio tem um valor particular que determina as informações que podem ser transmitidas para o sistema. A função de ativação permite que o valor de saída de cada neurônio seja calculado. É este cálculo que determina quantos neurônios devem ser ativados para resolver o problema. Um algoritmo é então criado. Ele atribui uma saída a cada uma das entradas (CASTRO F. C. C. C.; DE CASTRO, 2001).

O algoritmo permite que o computador aprenda com as novas informações que recebe. A rede neural é responsável pela análise dos exemplos, a fim de poder realizar uma tarefa. Estes exemplos estão rotulados. Este processo tornou os computadores capazes de reconhecer objetos em imagens, às vezes melhor do que o próprio cérebro humano. Como o cérebro humano, as redes neurais artificiais não podem ser programadas diretamente, mas devem ser aprendidas através do estudo e análise de exemplos (RAUBER, 2005).

## 2.4 DEEP LEARNING

O aprendizado profundo *(Deep Learning)* é um subcampo da IA. Este termo se refere ao conjunto de técnicas de aprendizagem de máquinas, ou seja, uma forma de aprendizagem baseada em abordagens matemáticas, usada para modelar dados. Para entender melhor estas técnicas, devemos voltar às origens da inteligência artificial em 1950, quando Alan Turning se interessou por máquinas capazes de pensar (GEOFFREY, 2015).

Esse interesse de Turing levou ao início da aprendizagem da máquina (*Machine Learning*), uma máquina que se comunica e se comporta conforme as informações armazenadas. O *Deep Learning* é um sistema avançado baseado no cérebro humano, que inclui uma grande rede de neurônios artificiais. Estes neurônios estão interligados para processar e memorizar informações, comparar qualquer problema ou situação com situações similares anteriores, analisar as soluções e resolver o problema da melhor maneira possível (PACHECO CéSAR AUGUSTO RODRIGUES; PEREIRA, 2018).

Esse tipo de aprendizado envolve o estudo de experiências vividas ou, no caso das máquinas, de informações gravadas. E uma de suas utilidades, no mundo da tecnologia de informação e comunicação, são em áreas como: reconhecimento de imagem, tradução automática, carros autônomos, diagnóstico médico, recomendações personalizadas, moderação automatizada de redes sociais, previsão financeira e comércio automatizado, identificação de peças defeituosas, detecção de malware ou fraude, chatbots (agentes de conversação), exploração espacial, robôs inteligentes (KELLEHER, 2019).

Também é utilizada em sistemas de reconhecimento facial e de voz embutidos em alguns celulares, e em robótica para permitir que equipamentos inteligentes tenham a reação esperada em uma determinada situação (por exemplo, um refrigerador inteligente que emite um sinal de alarme se detectar uma porta aberta ou uma temperatura anormal nos compartimentos).

Estas tecnologias também estão presentes em sistemas de tradução automática, em automóveis e outros veículos autônomos, na medicina para estabelecer um diagnóstico a partir de um teste de imagem (rádio, ressonância magnética, *scanner*), na física para buscar partículas, e no campo artístico para reproduzir uma obra de arte (AARON, 2016).

Fundamentado pelo conceito de uma rede de neurônios artificiais inspirados no cérebro humano, esta rede é composta por dezenas ou mesmo centenas de "camadas" de neurônios e cada uma das quais recebe e interpreta informações da camada anterior. Por exemplo, o sistema aprenderá a reconhecer as letras antes de atacar as palavras em um texto ou determinar se há um rosto em uma foto antes de descobrir quem é (KELLEHER, 2019). Em cada etapa, as respostas "erradas" são eliminadas e enviadas para níveis anteriores para ajustar o modelo matemático. Com o tempo, o programa reorganiza as informações em blocos mais complexos. Os dados iniciais são essenciais: quanto mais o sistema acumular diferentes experiências, melhor será o seu desempenho (PACHECO CéSAR AUGUSTO RODRIGUES; PEREIRA, 2018).

Em comparação com outras estruturas algorítmicas, as redes neurais têm duas vantagens principais: primeiro, sua estrutura baseada no empilhamento de funções não lineares lhes dá uma enorme capacidade. Demonstrou-se que uma rede com apenas duas camadas pode representar qualquer função matemática conectando entradas e saídas. Entretanto, por razões de treinamento e capacidade computacional, as arquiteturas modernas incluem um número maior de camadas (GEOFFREY, 2015).

A outra vantagem, as redes atuam como algoritmos de classificação e extratores de características. Na verdade, na grande maioria dos casos, são os próprios dados (valores de píxel no caso de imagens) fornecidos como entrada para uma rede e não características pré-definidas (AARON, 2016).

Os algoritmos de retro propagação e otimização permitem explorar a grande capacidade da rede, restringindo os pesos da rede a um conjunto de valores que levam a um bom desempenho de classificação: as características são aprendidas especificamente para a tarefa em consideração. As características extraídas, simples se consideradas nas primeiras camadas, tornam-se gradualmente mais complexas sobre as operações realizadas pelas camadas. É esta dimensão de camadas empilhadas, que permite a criação de características

complexas, referida quando se fala de aprendizagem "profunda". Isto leva a características que podem ser extremamente refinadas e sutis, muito mais do que um humano ou um algoritmo típico de extração de características pode conseguir (GEOFFREY, 2015).

Entretanto, a alta capacidade das redes pode trazer um grande inconveniente: a aprendizagem excessiva. As redes neurais têm que lidar com esta dificuldade. De fato, o número de pesos de uma rede é em muitos casos muito maior do que o número de parâmetros do hiperplano que permite que os dados de entrada sejam separados de acordo com suas etiquetas. Os chamados métodos de regularização são então aplicados para combater este aprendizado excessivo. No caso do "brinquedo" de regressão polinomial, a solução é escolher um modelo de menor grau, adaptado ao grau da função geradora (AARON, 2016).

A solução para obter um modelo útil é então aumentar ao máximo o número de dados para restringir o limite da decisão e evitar áreas "vazias", onde as previsões do modelo provavelmente serão discrepantes (PACHECO CéSAR AUGUSTO RODRIGUES; PEREIRA, 2018).

Portanto, algoritmos de aprendizado profundo requerem uma quantidade extremamente grande de dados para proporcionar um desempenho adequado. Além disso, esses dados devem ser anotados, ou seja, devem receber um rótulo. Dependendo da aplicação, a anotação pode ser extremamente demorada ou cara. Isto é particularmente verdadeiro em áreas onde esta anotação deve ser conduzida por especialistas, como agronomia ou medicina, e/ou para tarefas mais finas, como a segmentação (GEOFFREY, 2015). Na medicina, isto pode ser, por exemplo, a detecção de doenças por imagem. Para delinear uma lesão, a região em questão precisa ser delineada manualmente, o que leva muito tempo (KELLEHER, 2019).

### 2.4.1 REDES NEURAIS CONVOLUCIONAIS

As redes neurais convolucionais (ConvNet, ou CNN) designam uma subcategoria de redes neurais e são atualmente um dos modelos mais eficientes de classificação de imagens. Seu modo de operação consiste em que o usuário forneça como entrada uma imagem na forma de uma matriz de píxeis, que possui os tamanhos(AGHDAM HAMED HABIBI; HERAVI, 2017):

- Duas dimensões para uma imagem em escala de cinzentos;
- Uma terceira dimensão, profundidade 3, para representar as cores fundamentais (vermelho, verde e azul).

Ao contrário de um modelo clássico MLP (*Multi-Layer Perceptron*) que contém apenas uma camada de classificação, já a arquitetura da CNN tem uma camada convolutiva *upstream* portanto, duas camadas distintas (OLIVEIRA PATRICK FRANCISCO; C MARA, 2019):

- Uma parte convolutiva: Seu objetivo final é extrair características específicas de cada imagem, comprimindo-as para reduzir seu tamanho inicial. Em resumo, a imagem fornecida como entrada passa por uma sucessão de filtros, criando, simultaneamente, novas imagens chamadas de mapas de convolução. Finalmente, os mapas de convolução resultantes são concatenados em um vetor de características chamado código CNN;
- Uma parte de classificação: O código CNN obtido na saída da parte convolutiva é fornecido como entrada para uma segunda parte, que consiste em camadas conectadas totalmente chamadas MLP. A função desta parte é combinar as características do código CNN para classificar a imagem.

Convolução é uma operação matemática geralmente utilizada para o processamento e reconhecimento de imagens (O'SHEA KEIRON; NASH, 2015). Em uma imagem, seu efeito é semelhante ao da filtragem, que funciona da seguinte forma (OLIVEIRA PATRICK FRANCISCO; C MARA, 2019):

- Primeiro, o tamanho da janela do filtro no canto superior esquerdo é ajustado;
- A janela do filtro, representando a característica, move-se progressivamente da esquerda para a direita via um número de caixas pré-definidas (o degrau) até chegar ao final da imagem;
- Para cada parte da imagem encontrada, é feito um cálculo de convolução, resultando em um mapa de ativação ou mapa de características indicando onde as características estão localizadas na imagem: quanto maior o mapa de características, maior a parte da imagem digitalizada que se assemelha à característica.

Uma rede neural para classificação de imagens recebe e gerencia matrizes de píxeis como entradas; como os valores de píxel são RGB, três matrizes são gerenciadas para cada imagem, uma para cada componente de cor (AGHDAM HAMED HABIBI; HERAVI, 2017). Sua entrada, em alguns casos, é bidimensional, uma matriz, mas também pode ser modificada para ser unidimensional, permitindo uma representação interna de uma sequência unidimensional a ser desenvolvida (GU, 2018).

Permitindo que a CNN seja utilizada de forma mais geral em outros tipos de dados com uma relação espacial. Por exemplo, existe uma relação ordenada entre as palavras em um documento de texto. Há uma relação ordenada nas etapas temporais de uma série temporal (NADER, 2016). Embora não seja especificamente desenvolvida para dados sem imagem, as CNNs alcançam resultados de última geração em tópicos como classificação de documentos usados na análise de sentimentos e tópicos relacionados (AGHDAM HAMED HABIBI; HERAVI, 2017).

As redes neurais convolucionais seguem a estrutura clássica de uma rede genérica: consistem de uma camada de entrada, uma ou mais camadas intermediárias (chamadas camadas ocultas) e uma camada de saída. Cada camada é composta de neurônios, com um conjunto de entradas, uma saída e uma função de ativação (neurônio "ligado" ou "desligado"). Os neurônios são conectados mutualmente via conexões ponderadas, que recebem um valor w para cada conexão de entrada, fornecendo uma medida da importância desta entrada para o neurônio (O'SHEA KEIRON; NASH, 2015).

Durante a fase de treinamento, a rede aprende atualizando estes pesos a cada iteração. O método mais comumente utilizado é o Gradiente descendente, uma técnica de retro propagação. Após ter propagado a entrada através das diferentes camadas, a rede produz uma saída, com base na qual faz previsões sobre os dados (fase de propagação direta). Neste ponto, é calculada uma função de perda L (X i, w), que quantifica o quanto a saída produzida na i-ésima iteração difere da desejada. Este erro pode ser reduzido através da modificação dos pesos w na direção oposta ao gradiente de L (regra derivada da cadeia). Na verdade, o gradiente indica a direção de um maior crescimento de uma função em várias variáveis e o movimento na direção oposta reduz o erro (GU, 2018).

Entretanto, como o nome sugere, as CNNs adotam a convolução, um conceito matemático usado no Processamento Digital de Sinais que consiste em combinar duas funções (variáveis no tempo) coerentemente (AGHDAM HAMED HABIBI; HERAVI, 2017). A operação no domínio discreto para duas variáveis é definida como se segue:

$$(f * g)(x, y) \infty X u = -\infty f(u, v) g(x - u, y - v)$$

O processo de convolução é inspirado por processos biológicos de análise visual

em organismos vivos. A camada de convolução dos neurônios divide a imagem em vários fragmentos sobrepostos, os quais são analisados para identificar padrões de caracterização, transferindo as informações para a camada seguinte na forma de um mapa de características (O'SHEA KEIRON; NASH, 2015).

Basicamente, o algoritmo tenta detectar características na imagem, ou seja, características que distinguem os diferentes elementos: se, por exemplo, uma rede recebeu como entrada diferentes imagens de rostos de pessoas, ela tentaria encontrar características comuns, tais como dois olhos, duas orelhas, um nariz, uma boca. Um dos principais componentes da rede são os filtros (ou núcleos de convolução): do ponto de vista prático, eles correspondem a pequenas matrizes (em relação ao tamanho das imagens) contendo valores (GU, 2018). O nível de convolução, entretanto, introduz (NADER, 2016):

- complexidade, pois analisa pequenas partes da entrada, muitas das quais se sobrepõem e, consequentemente, o volume de saída é muito maior que a entrada e, nível por nível, a dimensionalidade se torna intratável;
- *overfitting*, pois o modelo pode aprender características que são específicas apenas do conjunto de treinamento, mas que não se refletem nos outros casos.

Estes problemas são minimizados pela introdução de uma camada de agrupamento logo após à camada convolutiva. E existem alguns tipos de camadas diferentes para usar dependendo da situação e do problema (OLIVEIRA PATRICK FRANCISCO; C MARA, 2019). Na figura 3 temos a camada de *Pooling*, responsável pela compressão dos dados que passam por ela e possui três outputs diferentes: *Max*, retira o maior valor de cada janela (Pool); *Min*, retira o menor valor; e *Average*, faz a média dos valores no *Pool*.

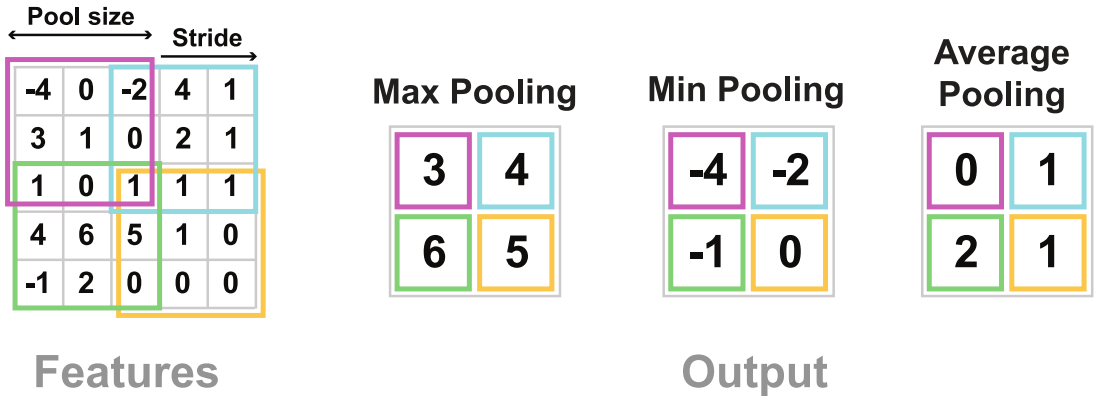



Figura 3 – Camada de Pooling (SPHINX, 2021)

Em seguida há a camada de Convolução, sendo essa a base das redes neurais convolucionais capazes de encontrar as *features* mais marcantes dos dados recebidos via filtros. Reduzindo a matriz para o tamanho do filtro utilizado, como mostrado no exemplo abaixo apresentado na figura 4 de 5×5 para uma 3×3.

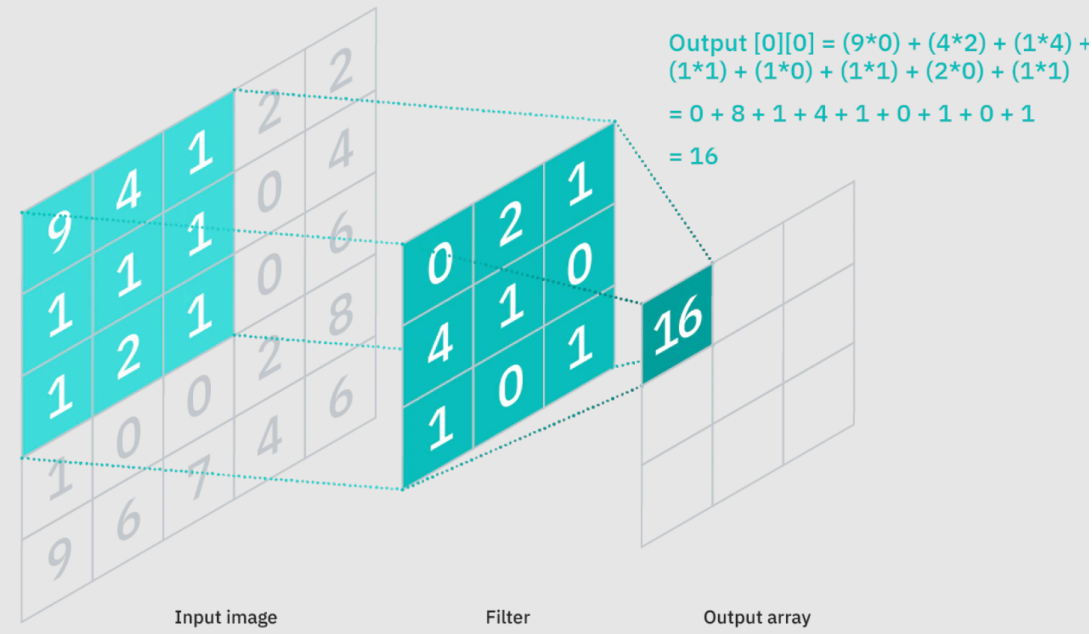



Figura 4 – Camada de Convolução (KUMAR, 2021)

Outra camada bastante usada é a *Flatten*. Caracterizada como a divisão entre a parte de extração de características e a da rede neural em si, normalmente utilizada depois de uma camada de *Pooling* para inserção na rede neural, transformando uma matriz em um *array* de dados, exemplo na figura 5.

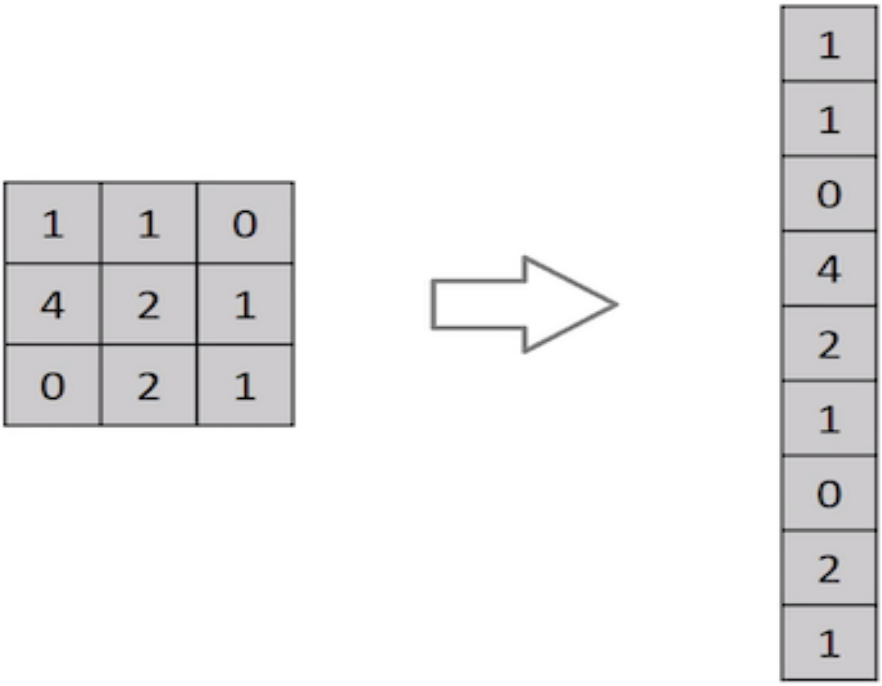



Figura 5 – Camada de Flatten (SAHA, 2018)

Finalizando a primeira parte da rede neural convolucional responsável pela extração dos dados, a segunda parte é responsável pela classificação dos mesmos. Chamada de *Dense Layer* (Camada Densa), ela pode ser composta por inúmeros neurônios (nós) que

classificarão os dados recebidos da camada de *Flatten*. Sendo constituída por um conjunto de 1 ou mais camadas densas como mostra a seguir a figura 6.

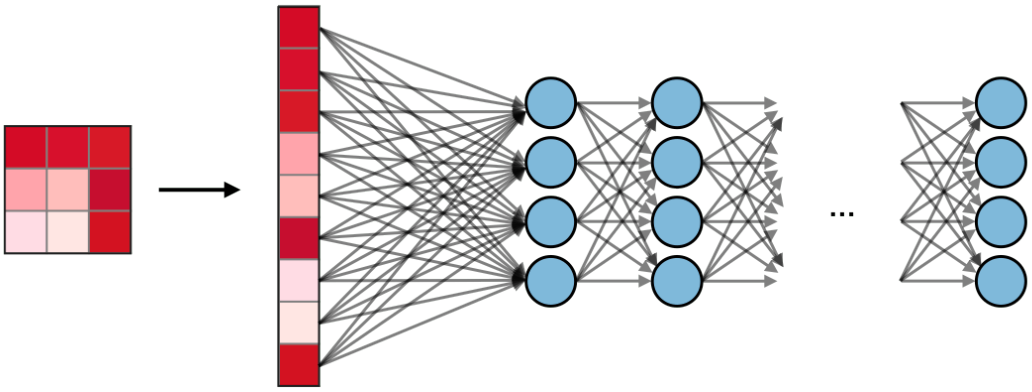


Figura 6 – Camada Densa (AMIDI; AMIDI, 2019)

Juntando as camadas convolutivas, com a de *pooling* e de *flatten* para extrair as características dos dados a serem classificados pela camada densa, segue um exemplo ilustrado na figura 7 de uma arquitetura de CNN completa.

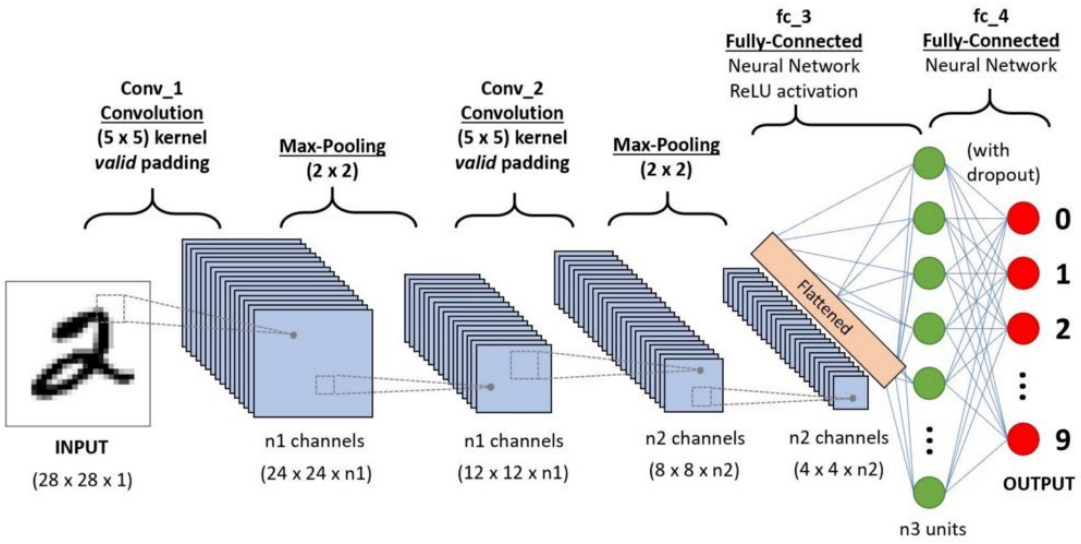



Figura 7 – CNN completa (SAHA, 2018)

# 3 TRABALHOS RELACIONADOS

Nesta seção serão apresentados alguns exemplos de pesquisas relacionadas à proposta deste trabalho. Os trabalhos a seguir serão apresentados em suma, focando nas características principais de cada solução e evidenciando as falhas ou melhorias possíveis. Não serão citados os trabalhos utilizando imagens como dados de entrada, pois já é sabido que para esse tipo de dado as redes neurais convolucionais são uma ótima solução e tem apresentado resultados bastante satisfatórios.

Vários trabalhos utilizando métodos tradicionais de *machine learning*, para seleção de características, classificação e predição, abordagens utilizando SVM, *Random Forest*, Árvores de decisão, KNN, AdaBoost e algoritmos genéticos (KHORSHED; MOUSTAFA; RAFEA, 2020) são citados. Porém, aqui o foco são os trabalhos utilizando redes neurais convolucionais em dados tabulares e/ou dados de Covid-19.

Um exemplo de trabalho realizado utilizando uma rede neural convolucional em auxílio a diagnóstico, foi desenvolvido por (MOHAMMAD M R KHAN MAMUN; ALI, 2020) para detecção do nível do problema de coração que o paciente poderia ter, utiliza uma combinação de dois algoritmos, o *Firefly* e uma CNN, que aumentaram o desempenho do modelo comparado com seus resultados separados. O primeiro foi um algoritmo chamado *Firefly*, originalmente proposto por Xin-She Yang, se inspira na forma que os vaga-lumes (*firefly*) se movem e se agrupam na natureza. Sempre na procura do que parece mais brilhante/atraente, o algoritmo busca essa atração entre os dados via cálculos de intensidade de luz e seu movimento. Pegando os resultados que mais possuem valor e direcionando a resposta para ele ou quando for um valor muito pequeno, o direcionamento vai para um valor aleatório.

Esse algoritmo foi utilizado antes de a base ser utilizada na CNN como um seletor das *features* mais importantes da base (das 180 existentes, 24 foram selecionadas). No trabalho referido não foi mostrada a arquitetura da rede CNN, apenas do projeto todo proposto como apresentado na figura 8. Mas a CNN é composta por duas camadas convolucionais, duas de max pooling e duas densas.

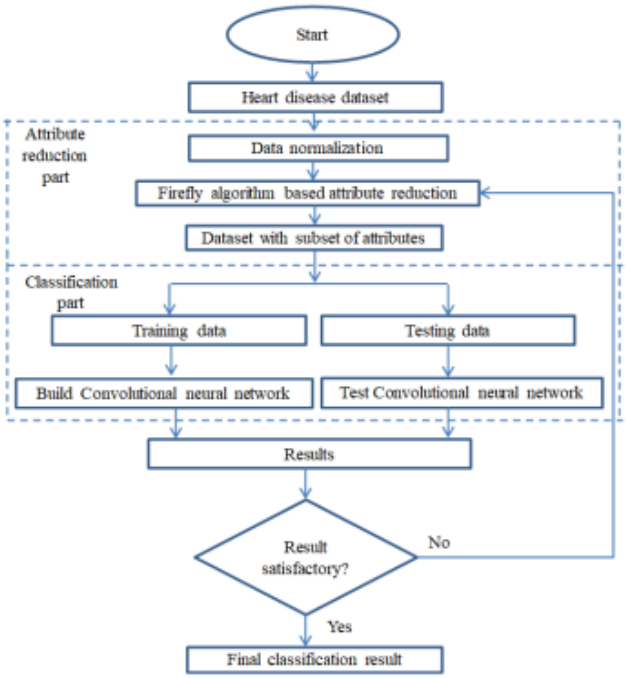



Figura 8 – Arquitetura do modelo proposto - Fonte: (MOHAMMAD M R KHAN MAMUN; ALI, 2020)

Na tabela 9 há o comparativo das métricas da rede proposta com outras técnicas de rede neural, uma delas é execução do modelo sem o uso do algoritmo do *firefly* (FA). Seus resultados se mantiveram em alta com o modelo proposto, indicando bons parâmetros da rede para os resultados esperados.

| Classifier | Precision | Recall | F-measure |
|---|---|---|---|
| Proposed method | 0.853 | 0.867 | 0.860 |
| Proposed method without FA | 0.828 | 0.832 | 0.830 |
| Random Forest | 0.78 | 0.805 | 0.792 |
| ANN | 0.813 | 0.814 | 0.813 |
| Logistic regression | 0.774 | 0.783 | 0.778 |

Figura 9 – Resultados obtidos das métricas da CNN - Fonte: (MOHAMMAD M R KHAN MAMUN; ALI, 2020)

No segundo trabalho (LIN et al., 2020) utiliza da combinação de três algoritmos diferentes para avaliação de pacientes que poderiam estar com depressão e seu nível de severidade. Para dados em formato de texto foi utilizado um modelo chamado *Bidirectional Long Short-Term Memory with Attention* (BiLSTM) que utiliza recorrência e aprendizado de mão dupla, ou seja, mesmo estando num passo a frente da iteração e não finalizado a rede consegue aprender tanto com passos anteriores quanto os que ainda estão por vir. Selecionando assim as características que demonstram o estado do paciente pelo que foi transcrito. Já a CNN 1D utilizada possui duas camadas convolucionais, duas de *pooling* e três camadas densas com um *dropout*, que retira de forma aleatória alguns valores da massa, antes da última camada densa.

| Features | Models | Classification | | | Regression | |
|---|---|---|---|---|---|---|
| | | F1 Score | Recall | Precision | MAE | RMSE |
| Audio | CNN-Augm [32] | 0.67 | 0.58 | 0.78 | - | - |
| | Williamson et al. [17] | 0.50 | - | - | 5.36 | 6.74 |
| | Ma et al. [33] | 0.52 | 1.00 | 0.35 | - | - |
| | Alhanai et al. [19] | 0.63 | 0.56 | 0.71 | 5.13 | 6.50 |
| | Yang et al. [50] | - | - | - | 4.63 | 5.52 |
| | Haque et al. [20] | - | - | - | 5.78 | - |
| | SVM | 0.40 | 0.50 | 0.33 | 5.93 | 6.74 |
| | Decision Tree | 0.57 | 0.50 | 0.57 | 5.70 | 6.79 |
| | Proposed 1D CNN model | **0.81** | **0.92** | 0.73 | **4.25** | **5.45** |

Figura 10 – Resultados obtidos das métricas da CNN - Fonte: (LIN et al., 2020)

Os dados passados vieram de dados de áudio focados no eixo da frequência. Juntando os dois resultados obtidos dos dois modelos, a rede composta por uma camada densa serviu como um ponto de fusão dos dados de áudio e texto para encontrar os pacientes que possuem depressão e a severidade do caso. Os resultados mostrados na figura 10, comparando o modelo proposto com outros modelos, foram superiores e rendendo bons dados para o trabalho realizado.

**Table 1.** Parameter settings of the 1D CNN model.

| Layer Name | Parameter Settings (Classification/Regression) | Activation |
|---|---|---|
| Conv1 | Kernel: [1, 7]<br>Stride: 1<br>Filter size: 32 | ReLU |
| Max-pool1 | Kernel: [4, 3]<br>Stride: [1, 3] | |
| Conv2 | Kernel: [1, 7]<br>Stride: 1/2<br>Filter size: 32 | ReLU |
| Max-pool2 | Kernel: [1, 3]<br>Stride: [1, 3] | |
| FC1 | Out features: 128 | ReLU |
| FC2 | Out features: 128 | ReLU |
| dropout | 0.5 | |
| FC3 | Out features: 2/1 | Softmax/Linear |

Figura 11 – Arquitetura 1D CNN - Fonte: (LIN et al., 2020)

Já o terceiro trabalho (LELLA; PJA, 2021) utiliza apenas de dados sonoros de voz, tosse e respiração dos pacientes para detecção de infectados pela Covid-19. Seu processamento utiliza um algoritmo de codificação de áudio para melhorar a extração dos dados da CNN 1D utilizada. Comparada com outros pré-processamentos, esse foi o que gerou melhor resultado como dado de entrada. O gráfico 12 mostra a arquitetura da CNN do trabalho que possui três camadas de convolução, três de *pooling* e uma única densa. Os dados inseridos foram dados não nomeados para que a rede em si os classifique e dê o retorno final.

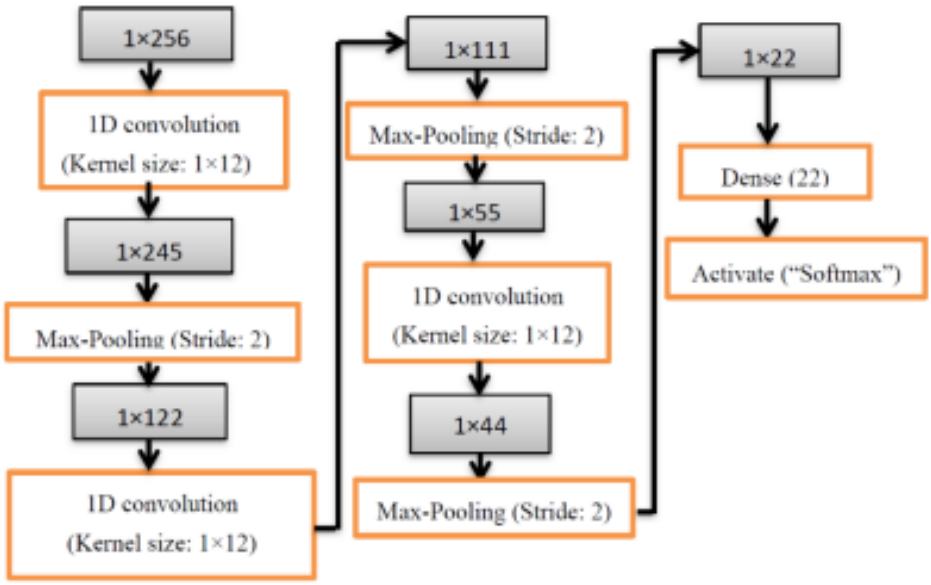



Figura 12 – Arquitetura 1D CNN - Fonte: (LELLA; PJA, 2021)

| Model | Dataset | Accuracy |
|---|---|---|
| SVM + PCA [23] | COVID-19 Crowdsourced Data | Task-I: 80% |
| | | Task-II: 82% |
| | | Task-III: 80% |
| SVM with Augmentation [23] | COVID-19 Crowdsourced Data | Task-II: 87% |
| | | Task-III: 88% |
| 1D CNN with Augmentation and DDAE (Proposed) | COVID-19 Crowdsourced Data | Task-I: 90% |
| | | Task-II: 88% |
| | | Task-III: 88% |
| | | Task-IV: 84% |
| | | Task-V: 86% |

Figura 13 – Resultados obtidos das métricas da CNN - Fonte: (LELLA; PJA, 2021)

Ao verificar os resultados do modelo na figura 13, a acurácia da CNN, em comparação com outros tipos de rede, a rede proposta obteve melhores resultados para que continuasse o projeto e boas classificações na avaliação preventiva da Covid19.

# 4 FERRAMENTAS E MÉTODOS

## 4.1 EQUIPAMENTOS

Para que o experimento pudesse ser realizado, foi utilizada uma máquina composta pelo processador Intel (Intel® Core™ i5-10400F CPU @2.90GHz), memória RAM de 16 GB (2×8 GB, DDR4 e 2666MHz), placa de vídeo NVIDIA (GTX 1650, 4 GB, GDDR6) e placa mãe da Asus (TUF Gaming B460M-Plus).

## 4.2 SOFTWARE

Com o intuito do trabalho a distância fosse monitorado e acompanhado pela orientadora, foi desenvolvido na ferramenta do Google chamada Google Colab. Com sua integração com o Drive, foi possível a manipulação e teste dos códigos tanto pela orientadora quanto do aluno.

## 4.3 BASE DE DADOS

A base de dados utilizada possui resultados de exames clínicos, exceto exames de imagem, de pessoas testadas positivas e negativas da Covid19, fornecidos pelo hospital Sírio Libanês e estão no formato CSV no Drive, acessado pelo Colab.

Possuindo 5643 indivíduos, sendo 558 positivos (sendo 1 seu valor representativo binário) e 5085 negativos (sendo 0 seu valor representativo binário). Além de existir 91 *features* ou características clínicas a serem utilizadas pela rede neural na detecção. Sendo as seguintes algumas utilizadas na classificação do modelo: Total Co, timeCreatinine, DDimer, pH, Ferritin, Sodium. Segue o mapa de calor abaixo 14 mostrando as *features* consideradas mais importantes (CARDOSO, 2022):

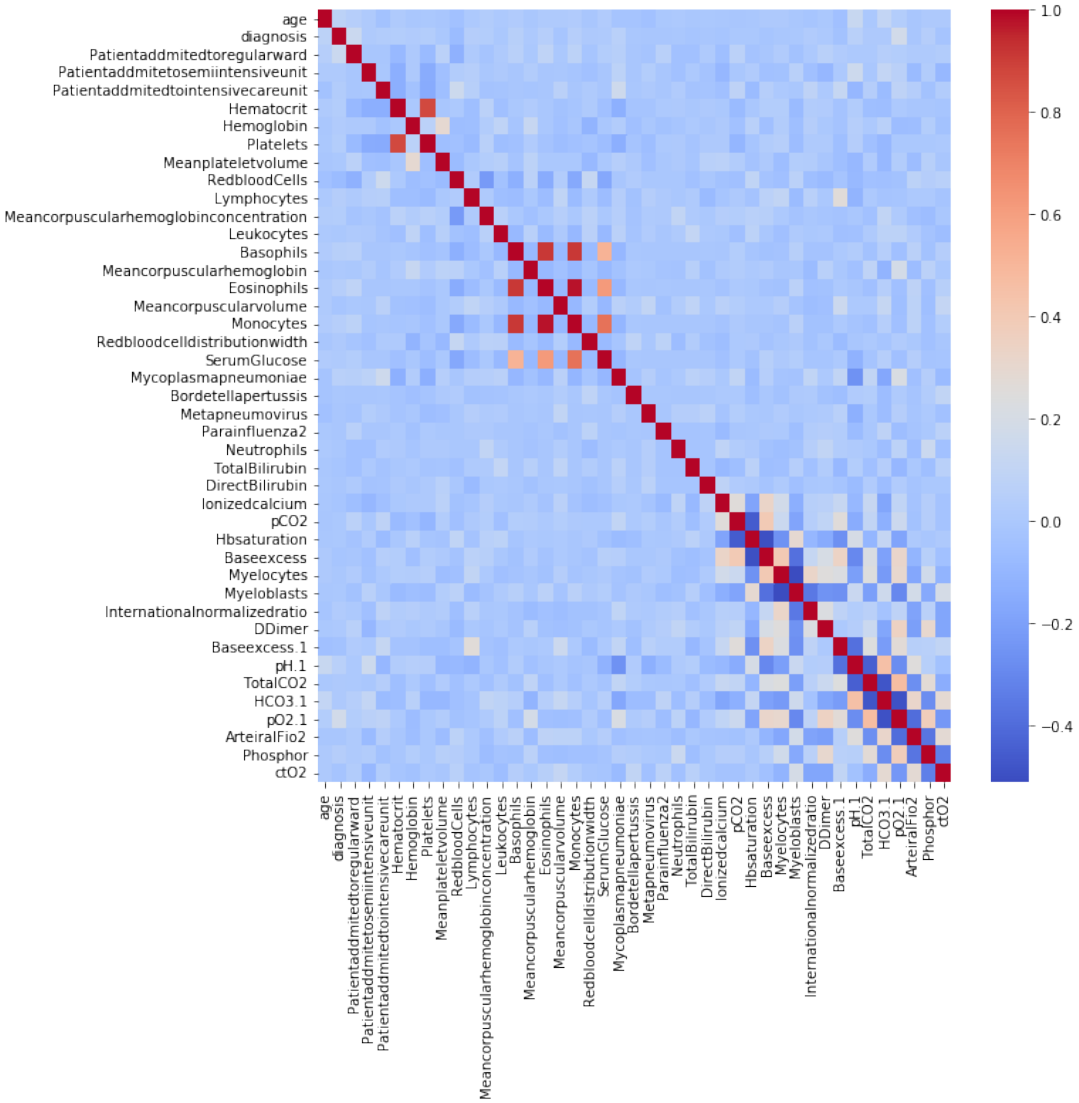



Figura 14 – Mapa de calor das *features* mais importantes - Fonte: próprio autor

No entanto, as informações contidas nessa base de dados apresentam alguns problemas fundamentais: dados vazios - muitos dos exames sem nenhum valor; inválidos - alguns possuem texto ao invés de números; e desbalanceados - o número de casos positivos equivale a 9.88% de todos os casos existentes, como mostradona figura abaixo 15. Atrapalhando a classificação utilizando a CNN deste projeto.

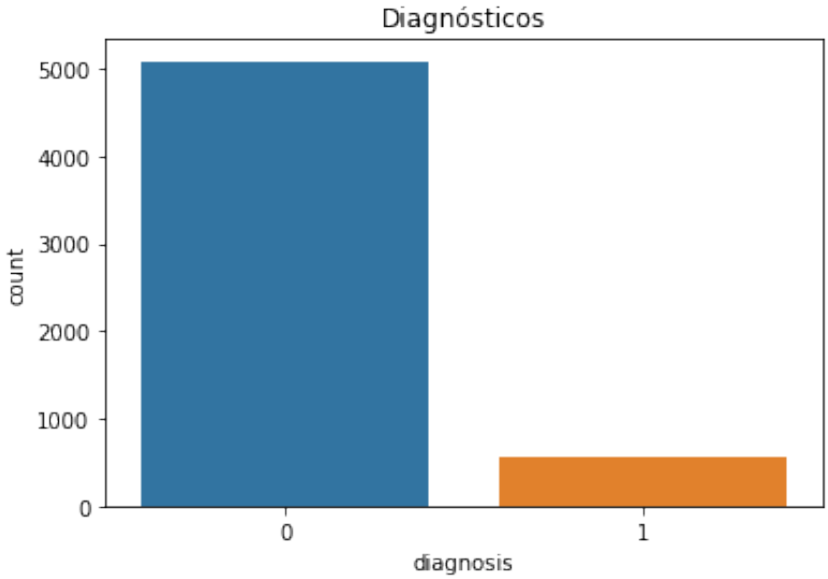



Figura 15 – Gráfico com a diferença entre dados negativos e positivos - Fonte: próprio autor

## 4.4 MÉTRICAS DE AVALIAÇÃO

Ao utilizar as diferentes métricas de avaliação para medir o desempenho de um modelo, é possível melhorar o poder preditivo geral do mesmo antes de colocá-lo, na prática, com dados considerados imprevisíveis, dados não observados pelo modelo (AGRAWAL, 2021). A não realização de uma avaliação adequada do modelo usando diferentes métricas de avaliação, e confiando apenas na acurácia, pode levar a previsões ruins. Neste trabalho serão avaliadas e analisadas as seguintes métricas:

A acurácia (A) é definida, informalmente, como a fração de previsões corretas do modelo. E formalmente, como a divisão entre o número de previsões corretas e o total de previsões feitas (COURSE, n.d.). Sua fórmula matemática descrita abaixo:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

A precisão (P) demonstra quantos dos casos previstos corretamente se revelaram realmente positivos. A precisão é útil nos casos em que os falsos positivos são mais preocupantes do que os falsos negativos (AGRAWAL, 2021). Descrita abaixo como:

$$P = \frac{TP}{TP + FP}$$

A sensibilidade, também chamada de *Recall* (R), explicita quantos dos casos positivos reais o modelo foi capaz de prever corretamente. É uma métrica útil nos casos em que os falsos positivos possuem mais importância do que os falsos negativos (AGRAWAL, 2021). Descrita abaixo:

$$R = \frac{TP}{TP + FN}$$

Nas fórmulas anteriores, TP (*True Positive* ou verdadeiro positivo) refere aos dados positivos previstos corretamente; já FP (*False Positive* ou falso positivo) refere aos dados considerados positivos, mas a predição errou. O termo TN (*True Negative*

ou verdadeiro negativo) refere aos dados negativos previstos corretamente e FN (*False Negative* ou falso negativo) aos dados considerados negativos, mas a predição errou.

Outra métrica utilizada foi a F1, ou *F-score*, definida como média harmônica entre a precisão e a sensibilidade (*recall*). E Por ser a média entre essas duas métricas, o peso dado para elas é de igual valor (KORSTANJE, 2021):

$$F1 = 2\frac{P * R}{P + R}$$

O Coeficiente de determinação, também chamado de $R^2$, possui valores de 0 a 100% sendo definido como a proporção entre a variância dos dados de entrada e saída. Em outras palavras, caso o valor seja alto, as variáveis possuem correlação entre si, sem variância alguma. Caso contrário, há uma divergência encontrada nos dados podendo indicar falha no modelo (ROWE, 2018). Segue sua fórmula na figura abaixo 16:

$$R2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})}, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Figura 16 – Fórmula do $R^2$ (Coeficiente da Determinação) - Fonte: (TAYO, 2021)

A matriz de confusão é uma medida muito utilizada em problemas de classificação tanto binários quanto multi-classes. Seus valores mostrados são classificados entre TP, TN, FP e TN (KULKARNI; CHONG; BATARSEH, 2020). Com os dados aplicados na Precisão, *Recall* e *F-score*, pode-se utilizar a matriz de confusão e comparar os resultados encontrados para avaliar a eficácia do modelo.

Uma das métricas utilizadas no treino da rede e responsável por definir o quanto a rede consegue modelar os dados, é a função de perda ou *Loss Functions*. Ela calcula a distância entre a predição feita pelo modelo e o valor real da classe alvo. Algumas dessas funções foram utilizadas neste trabalho. A *Binary cross-entropy* é uma função utilizada para classificação de dados binários (0 e 1) que gera uma boa generalização dos dados (PERE, 2020). Já a *Mean square error* (MSE) ou quadrado da média dos erros demonstra o tamanho da diferença entre os dados previstos pela rede e os valores reais (ROWE, 2018).

Outra das métricas utilizada foi o *Mean Absolute Error* (MAE) ou média dos erros absolutos. Conhecida como regressão L1, calcula a diferença a absoluta entre o real e o esperado e a divide pelo número de dados da saída. Tendo como objetivo minimizar as diferenças absolutas dos dados (ARYA, 2022). Já o *Mean Absolute Percentage Error* (MAPE) ou Erro Percentual Absoluto Médio calcula qual a porcentagem dos erros do modelo, de forma parecida com o MAE, com a diferença que há uma divisão no MAPE pelo valor absoluto do dado que está sendo calculado (JúNIOR, 2021).

$$MAE\,(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

(a) Fórmula da MAE - Fonte: (JúNIOR, 2021)

$$MAPE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{max(\epsilon, |y_i|)}$$

(b) Fórmula da MAPE - Fonte: (JúNIOR, 2021)

Figura 17 – Fórmulas das funções de erro

## 4.5 LINGUAGEM DE PROGRAMAÇÃO E BIBLIOTECAS

Na parte do desenvolvimento da rede neural, a linguagem utilizada foi o Python. Sendo uma linguagem de sintaxe menos complexa e próxima da natural, ou seja, de alto nível, é utilizada por muitos cientistas de dados por possuir tipagem dinâmica, ser orientada a objetos, vasto arsenal de bibliotecas e uma comunidade ativa em seu aprimoramento (MELO, 2021). As bibliotecas de uso da rede são apresentadas na figura 18:

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold
import csv
import matplotlib.pyplot as plt
import numpy as np
import keras.utils
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Conv1D, AveragePooling1D, MaxPooling1D, Flatten, Dense
from sklearn.metrics import confusion_matrix, accuracy_score, ConfusionMatrixDisplay
from sklearn.metrics import precision_score, recall_score, f1_score, r2_score
import time
df = pd.read_csv('/content/drive/MyDrive/Orientação Adelson-CC/covid/datasetIntSurinaOld.csv',
                 encoding='ISO-8859-1', sep=',')
```

Figura 18 – Bibliotecas - Fonte: próprio autor

As bibliotecas utilizadas foram as seguintes: pandas (utilizada para a estruturação dos dados lidos pela biblioteca anterior), csv (para leitura do arquivo da base de dados), matplotlib.pyplot (criação de gráficos), numpy (organização dos dados numéricos e seu formato para a análise), keras.utils (conjunto de diferentes funções do Keras, no trabalho o plotmodel foi utilizado para gerar uma foto da arquitetura da rede), keras.models (grupo de modelos que disponível no Keras, o utilizado no trabalho foi o *Sequential* que agrupa as camadas do modelo linear), keras.optimizers (funções utilizadas no treinamento da rede, o Adam foi escolhido para o trabalho), keras.layers (agrupamento de todas as possíveis camadas de uma rede neural, no trabalho as utilizadas foram Conv1D, AveragePooling1D, MaxPool1D, *Flatten* e *Dense*), e sklearn.metrics (uso de 6 métricas de avaliação citadas anteriormente juntamente do gráfico da matriz de confusão).

Já o StratifiedKFold da biblioteca keras.model-selection é uma função que reorganiza os dados da base em X partes (mínimo 2), separando e balanceando os dados em grupos de teste e grupos de treino de, no mínimo, uma parte em cada grupo. Os grupos de treino e teste são uma parte muito importante em relação ao treino e predição do modelo, pois pode dar problema na hora da classificação caso a separação seja feita de forma desbalanceada (em uma das partes há 150 dados e na outra 130, dando erro na hora de execução do modelo). A base de treino que fica em função de treinar o modelo a acertar/errar os dados da base, já a de teste é utilizada após o treinamento para as predições e comparações com a classe final.

# 5 PROPOSTA DE TRABALHO

Para o desenvolvimento deste estudo de caso, foi utilizada uma base de dados de exames clínicos de dados de pacientes com suspeita de Covid-19 disponibilizada pelo hospital Sírio Libanês de São Paulo, como mencionada anteriormente, desbalanceada e com problemas de dados faltantes. Como técnica de aprendizado de máquina foi selecionada a rede neural convolucional para classificar esses dados e avaliar a existência de padrões que possibilitasse a identificação dos pacientes com a presença e/ou ausência da doença. Para que a classificação desses dados fosse o mais precisa possível, foi realizada uma seleção de características para selecionar aquelas que melhor representassem a base de dados.

Dito isso, será apresentado os detalhes desse desenvolvimento, evidenciando as etapas de pré-processamento dos dados, esmiuçando a arquitetura do modelo desenvolvido e explicando cada camada e os parâmetros utilizados.

Por conta dos problemas mencionados na base, algumas medidas foram necessárias para substituir os dados nulos, vazio ou com formato errado. A estrutura apresentada nas figuras 19 a 26 demonstra todo o processo de tratamento de dados utilizando as seguintes funções: df.replace (troca o valor mencionado no primeiro parâmetro pelo segundo parâmetro e o terceiro parâmetro garante que não será criado outro DataFrame) e fillna (troca todos valores 'NaN' pela média dos outros valores que estão corretos).

```
df.replace("negative", 0, inplace = True)
df.replace("positive", 1, inplace = True)
df.replace("not_detected", 0, inplace = True)
df.replace("detected", 1, inplace = True)
df.replace("not_done", 0, inplace = True)
df.replace("not_defined", 0, inplace = True)
```

Figura 19 – Tratamento de dados 1 - Fonte: próprio autor

```
df['Albumin'].fillna(pd.to_numeric(df['Albumin'],errors='coerce').mean(),inplace = True)
df['TotalCO2'].fillna(df['TotalCO2'].mean(),inplace = True)
df['ArteiralFio2'].fillna(df['ArteiralFio2'].mean(),inplace = True)
df['Phosphor'].fillna(df['Phosphor'].mean(),inplace = True)
df['Hbsaturation'].fillna(df['Hbsaturation'].mean(),inplace = True)
df['age'].fillna(df['age'].mean(),inplace = True)
df['diagnosis'].fillna(df['diagnosis'].mean(),inplace = True)
df['Patientaddmitedtoregularward'].fillna(df['Patientaddmitedtoregularward'].mean(),inplace = True)
df['Monocytes'].fillna(df['Monocytes'].mean(),inplace = True)
df['Patientaddmitedtointensivecareunit'].fillna(df['Patientaddmitedtointensivecareunit'].mean(),inplace = True)
df['ctO2'].fillna(df['ctO2'].mean(),inplace = True)
```

Figura 20 – Tratamento de dados 2 - Fonte: próprio autor

```
df['Partialthromboplastin time'].fillna(pd.to_numeric(df['Partialthromboplastin time'],errors='coerce').mean(),inplace = True)
df['Relationship'].fillna(pd.to_numeric(df['Relationship'],errors='coerce').mean(),inplace = True)
df['Internationalnormalizedratio'].fillna(df['Internationalnormalizedratio'].mean(),inplace = True)
df['LacticDehydrogenase'].fillna(pd.to_numeric(df['LacticDehydrogenase'],errors='coerce').mean(),inplace = True)
df['ProthrombintimeActivity'].fillna(pd.to_numeric(df['ProthrombintimeActivity'],errors='coerce').mean(),inplace = True)
df['VitaminB12'].fillna(pd.to_numeric(df['VitaminB12'],errors='coerce').mean(),inplace = True)
df['Creatinephosphokinase'].fillna(pd.to_numeric(df['Creatinephosphokinase'],errors='coerce').mean(),inplace = True)
df['Ferritin'].fillna(pd.to_numeric(df['Ferritin'],errors='coerce').mean(),inplace = True)
df['ArterialLacticAcid'].fillna(pd.to_numeric(df['ArterialLacticAcid'],errors='coerce').mean(),inplace = True)
df['Lipasedosage'].fillna(pd.to_numeric(df['Lipasedosage'],errors='coerce').mean(),inplace = True)
df['DDimer'].fillna(df['DDimer'].mean(),inplace = True)
```

Figura 21 – Tratamento de dados 3 - Fonte: próprio autor

```
df['Hbsaturation'].fillna(pd.to_numeric(df['Hbsaturation'],errors='coerce').mean(),inplace = True)
df['Baseexcess'].fillna(pd.to_numeric(df['Baseexcess'],errors='coerce').mean(),inplace = True)
df['pO2'].fillna(pd.to_numeric(df['pO2'],errors='coerce').mean(),inplace = True)
df['Fio2'].fillna(pd.to_numeric(df['Fio2'],errors='coerce').mean(),inplace = True)
df['Total CO2'].fillna(pd.to_numeric(df['Total CO2'],errors='coerce').mean(),inplace = True)
df['pH'].fillna(pd.to_numeric(df['pH'],errors='coerce').mean(),inplace = True)
df['HCO3'].fillna(pd.to_numeric(df['HCO3'],errors='coerce').mean(),inplace = True)
df['Rods'].fillna(pd.to_numeric(df['Rods'],errors='coerce').mean(),inplace = True)
df['Segmented'].fillna(pd.to_numeric(df['Segmented'],errors='coerce').mean(),inplace = True)
df['Promyelocytes'].fillna(pd.to_numeric(df['Promyelocytes'],errors='coerce').mean(),inplace = True)
df['Metamyelocytes'].fillna(pd.to_numeric(df['Metamyelocytes'],errors='coerce').mean(),inplace = True)
```

Figura 22 – Tratamento de dados 4 - Fonte: próprio autor

```
df['InfluenzaBrapidtest'].fillna(pd.to_numeric(df['InfluenzaBrapidtest'],errors='coerce').mean(),inplace = True)
df['InfluenzaArapidtest'].fillna(pd.to_numeric(df['InfluenzaArapidtest'],errors='coerce').mean(),inplace = True)
df['Alaninetransaminase'].fillna(pd.to_numeric(df['Alaninetransaminase'],errors='coerce').mean(),inplace = True)
df['Aspartatetransaminase'].fillna(pd.to_numeric(df['Aspartatetransaminase'],errors='coerce').mean(),inplace = True)
df['Gammaglutamyltransferase'].fillna(pd.to_numeric(df['Gammaglutamyltransferase'],errors='coerce').mean(),inplace = True)
df['TotalBilirubin'].fillna(df['TotalBilirubin'].mean(),inplace = True)
df['DirectBilirubin'].fillna(df['DirectBilirubin'].mean(),inplace = True)
df['IndirectBilirubin'].fillna(pd.to_numeric(df['IndirectBilirubin'],errors='coerce').mean(),inplace = True)
df['Alkalinephosphatase'].fillna(pd.to_numeric(df['Alkalinephosphatase'],errors='coerce').mean(),inplace = True)
df['Ionizedcalcium'].fillna(df['Ionizedcalcium'].mean(),inplace = True)
df['Magnesium'].fillna(pd.to_numeric(df['Magnesium'],errors='coerce').mean(),inplace = True)
df['pCO2'].fillna(pd.to_numeric(df['pCO2'],errors='coerce').mean(),inplace = True)
```

Figura 23 – Tratamento de dados 5 - Fonte: próprio autor

```
df['Coronavirus229E'].fillna(pd.to_numeric(df['Coronavirus229E'],errors='coerce').mean(),inplace = True)
df['CoronavirusOC43'].fillna(pd.to_numeric(df['CoronavirusOC43'],errors='coerce').mean(),inplace = True)
df['InfAH1N12009'].fillna(pd.to_numeric(df['InfAH1N12009'],errors='coerce').mean(),inplace = True)
df['Bordetellapertussis'].fillna(df['Bordetellapertussis'].mean(),inplace = True)
df['Metapneumovirus'].fillna(df['Metapneumovirus'].mean(),inplace = True)
df['Parainfluenza2'].fillna(df['Parainfluenza2'].mean(),inplace = True)
df['Neutrophils'].fillna(df['Neutrophils'].mean(),inplace = True)
df['Urea'].fillna(pd.to_numeric(df['Urea'],errors='coerce').mean(),inplace = True)
df['ProteinaCreativa mgdL'].fillna(pd.to_numeric(df['ProteinaCreativa mgdL'],errors='coerce').mean(),inplace = True)
df['Creatinine'].fillna(pd.to_numeric(df['Creatinine'],errors='coerce').mean(),inplace = True)
df['Potassium'].fillna(pd.to_numeric(df['Potassium'],errors='coerce').mean(),inplace = True)
df['Sodium'].fillna(pd.to_numeric(df['Sodium'],errors='coerce').mean(),inplace = True)
```

Figura 24 – Tratamento de dados 6 - Fonte: próprio autor

```
df['RespiratorySyncytialVirus'].fillna(pd.to_numeric(df['RespiratorySyncytialVirus'],errors='coerce').mean(),inplace = True)
df['InfluenzaA'].fillna(pd.to_numeric(df['InfluenzaA'],errors='coerce').mean(),inplace = True)
df['InfluenzaB'].fillna(pd.to_numeric(df['InfluenzaA'],errors='coerce').mean(),inplace = True)
df['Parainfluenza1'].fillna(pd.to_numeric(df['Parainfluenza1'],errors='coerce').mean(),inplace = True)
df['CoronavirusNL63'].fillna(pd.to_numeric(df['CoronavirusNL63'],errors='coerce').mean(),inplace = True)
df['RhinovirusEnterovirus'].fillna(pd.to_numeric(df['RhinovirusEnterovirus'],errors='coerce').mean(),inplace = True)
df['Mycoplasmapneumoniae'].fillna(df['Mycoplasmapneumoniae'].mean(),inplace = True)
df['CoronavirusHKU1'].fillna(pd.to_numeric(df['CoronavirusHKU1'],errors='coerce').mean(),inplace = True)
df['Parainfluenza3'].fillna(pd.to_numeric(df['Parainfluenza3'],errors='coerce').mean(),inplace = True)
df['Chlamydophilapneumoniae'].fillna(pd.to_numeric(df['Chlamydophilapneumoniae'],errors='coerce').mean(),inplace = True)
df['Adenovirus'].fillna(pd.to_numeric(df['Adenovirus'],errors='coerce').mean(),inplace = True)
df['Parainfluenza4'].fillna(pd.to_numeric(df['Parainfluenza4'],errors='coerce').mean(),inplace = True)
```

Figura 25 – Tratamento de dados 7 - Fonte: próprio autor

```
df['Hematocrit'].fillna(df['Hematocrit'].mean(),inplace = True)
df['Hemoglobin'].fillna(df['Hemoglobin'].mean(),inplace = True)
df['Platelets'].fillna(df['Platelets'].mean(),inplace = True)
df['Meanplateletvolume'].fillna(df['Meanplateletvolume'].mean(),inplace = True)
df['RedbloodCells'].fillna(df['RedbloodCells'].mean(),inplace = True)
df['Lymphocytes'].fillna(df['Lymphocytes'].mean(),inplace = True)
df['Meancorpuscularhemoglobinconcentration'].fillna(df['Meancorpuscularhemoglobinconcentration'].mean(),inplace = True)
df['Leukocytes'].fillna(df['Leukocytes'].mean(),inplace = True)
df['Basophils'].fillna(df['Basophils'].mean(),inplace = True)
df['Meancorpuscularhemoglobin'].fillna(df['Meancorpuscularhemoglobin'].mean(),inplace = True)
df['Eosinophils'].fillna(df['Eosinophils'].mean(),inplace = True)
df['Meancorpuscularvolume'].fillna(df['Meancorpuscularvolume'].mean(),inplace = True)
df['Monocytes'].fillna(df['Monocytes'].mean(),inplace = True)
df['Redbloodcelldistributionwidth'].fillna(df['Redbloodcelldistributionwidth'].mean(),inplace = True)
df['SerumGlucose'].fillna(df['SerumGlucose'].mean(),inplace = True)
```

Figura 26 – Tratamento de dados 8 - Fonte: próprio autor

O tipo de rede neural desenvolvida foi a Rede Neural Convolucional 1D por conta da base de dados ser unidimensional, mas o *input* da rede necessita de dados bidimensionais. Para isso foi necessário reformatar os dados e criar uma dimensão que não afetasse o resultado. Será detalhado abaixo a arquitetura da rede, a função de cada uma de suas camadas e sua estrutura apresentada nas figuras 27 e 28:

```
def create_model(input_shape):
  global model
  model = Sequential([
    AveragePooling1D(4,2, input_shape=input_shape, padding='same'),
    Conv1D(64, 2, activation='relu', padding='same'),
    Conv1D(32, 2, activation='relu', padding='same'),
    MaxPooling1D(),
    Flatten(),
    Dense(256, activation='sigmoid'),
    Dense(1, activation='softmax')
  ])
  model.compile(optimizer='adam',
                loss=keras.losses.BinaryCrossentropy(),
                metrics=['accuracy','mse', 'mae', 'mape'])
```

Figura 27 – Estrutura da rede no código - Fonte: próprio autor

- Sequential: Configuração do modelo da rede, juntando suas camadas linearmente;
- AveragePooling1D: responsável pela diminuição dos dados de entrada utilizando a média como parâmetro;

- Duas camadas de Conv1D(64 e 32): as duas camadas convolucionais principais da rede com 64 e 32 filtros cada, criando um conjunto de diferentes mapas de características a serem processados nas camadas seguintes;
- MaxPool1D: utilizando o mesmo princípio da primeira camada com a diferença que essa utiliza o maior valor da janela de dados gerada;
- *Flatten*: a camada que reduz a dimensão dos dados de bidimensional para unidimensional, sem afetar o tamanho do *batch*;
- Camada *Dense* 256: camadas de decisões interconectadas com 256 neurônios e com a função de ativação *sigmoid*. A função *sigmoid* é uma função logística que possui uma curva em formato de S.
- Camada *Dense* 1: camadas de decisões interconectadas com 1 neurônio e a função de ativação *softmax*. A função *softmax* utiliza de probabilidades para fazer com que a predição da rede resulte em valores entre 0 e 1. Geralmente utilizado em conjunto com o emphBinaryCrossentropy como função de perda.

Ao compilar o modelo, é preciso passar três novos parâmetros na função model.compile como mostra a figura 27:

- *optimizer*: sua função é atribuir pesos e velocidade de aprendizagem da rede, dependendo do otimizador utilizado. Na rede foi escolhido o Adam (Adaptive Moment Estimation), pois foi uma das últimas modificações realizadas na rede e pelos estudos foi o que mais combinou com os dados. Sendo essa função a combinação de duas metodologias de gradiente descendente, Momentum e Root Mean Square Propagation (RMSP), utilizando os pontos positivos dos métodos e optimizando o gradiente descendente;
- *loss*: método responsável por dizer a rede o quanto deve buscar minimizar os valores de perda. De todos que existem, apenas o *BinaryCrossentropy* funcionou para os dados da base por conta de sua dimensão e valores (0 ou 1);
- *metrics*: são as métricas utilizadas para medir o desempenho da rede neural e aqui foram utilizadas a acurácia, *Mean Absolute Error*, *Mean Absolute Percent Error* e *Mean Squared Error*.

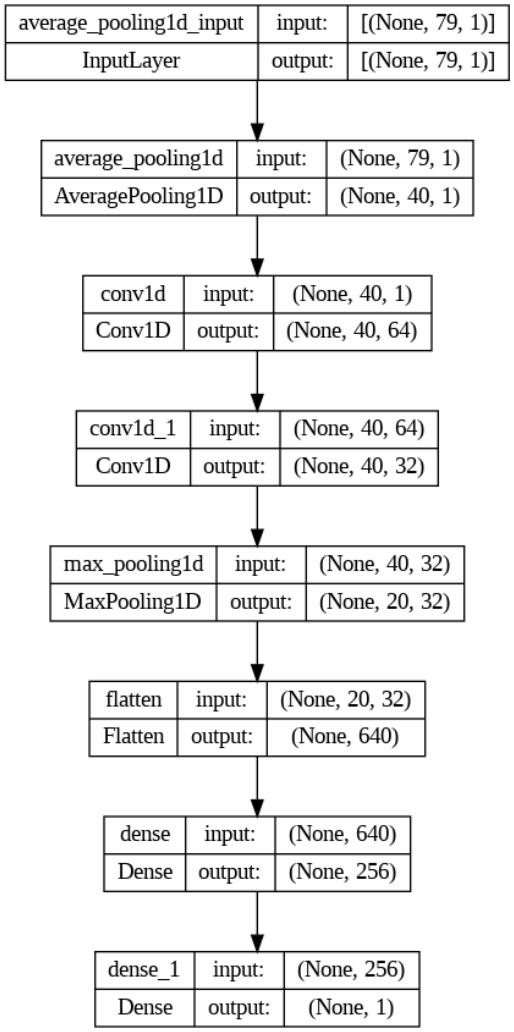



Figura 28 – Arquitetura da CNN 1D - Fonte: próprio autor

Passado o tratamento dos dados e a criação do modelo, foi criada outra função responsável por avaliar o tempo que a CNN criada leva para ser treinada. Na figura 29 temos a função secondsToMinutesSeconds recebendo um parâmetro sec (segundos), que será dividido por 3600 e seu resto armazenado na mesma variável. Em seguida é feita uma divisão inteira (o valor é truncado para o menor inteiro possível) e resulta no valor dos minutos. Finalizando com a divisão da variável dos segundos por 60 e seu resto são os segundos que restam.

```
def secondsToMinutesSeconds(sec):
    sec %= 3600
    min = sec // 60
    sec %= 60
    return "Evaluated in %02d:%02d" % (min, sec)
```

Figura 29 – Função de timer - Fonte: próprio autor

Realizadas todas as definições das bibliotecas, da criação das funções do modelo e do tempo de treino, da importação da base de dados e seu tratamento, o programa está

preparado para executar o treinamento do modelo. Começando, como mostrado na figura 30, com a criação das seis listas que utilizadas para o armazenamento das massas de treino que o SKF separar. O splitDados divide a quantidade de massas de treino/teste separadas pelo SKF que mantêm a proporção de dados de cada classe em cada um dos conjuntos de dados e o *target* armazena os rótulos classe alvo para a separação.

```
acc, f1s, prec, rec, r2s, cml = ([] for i in range(6))
splitDados = 2 #2 5 9
skf = StratifiedKFold(n_splits=splitDados)
target = df.loc[:,'diagnosis']
```

Figura 30 – Criação das listas das métricas e definição do StratifiedKFold - Fonte: próprio autor

Parâmetros e listas definidas, é necessário utilizar a função skf.split que recebe a base dados e o *target* como parâmetros. Por conta de serem mais de uma massa de treino/teste, precisa que a função seja coloca em um *loop* for e retirar da função os índices de treino e teste para realizar algumas formatações a seguir. Ainda na figura 31 há duas variáveis importantes, X e y. Sendo elas as 81 melhores *features* selecionadas e a classe final, respectivamente.

```
for i, (train_index, test_index) in enumerate(skf.split(df, target)):
    X = ['ID', 'age','Patientaddmitedtoregularward', 'CoronavirusNL63', 'pCO2',
         'SerumGlucose', 'RespiratorySyncytialVirus', 'Adenovirus', 'Albumin',
         'Coronavirus229E', 'Mycoplasmapneumoniae', 'Parainfluenza3', 'pCO2',
         'InfluenzaB', 'Meancorpuscularhemoglobin', 'RhinovirusEnterovirus',
         'Redbloodcelldistributionwidth', 'Lymphocytes', 'Relationship', 'Fio2',
         'ProthrombintimeActivity', 'InfluenzaArapidtest', 'Hemoglobin', 'pH',
         'LacticDehydrogenase', 'Meancorpusculavolume', 'Platelets', 'HCO3',
         'Ionizedcalcium', 'Meanplateletvolume', 'Hbsaturation', 'Phosphor',
         'Leukocytes', 'Total CO2', 'Metamyelocytes',
         'InfluenzaBrapidtest', 'Parainfluenza4', 'InfAH1N12009', 'DDimer',
         'Chlamydophilapneumoniae', 'Myeloblasts', 'ArteiralFio2', 'pO2', 'pH',
         'Hematocrit', 'Partialthromboplastin time', 'VitaminB12','CoronavirusOC43',
         'CoronavirusHKU1', 'Meancorpuscularhemoglobinconcentration', 'ctO2',
         'Aspartatetransaminase', 'Patientaddmitedtointensivecareunit', 'HCO3',
         'InfluenzaA', 'Segmented', 'Eosinophils', 'Alkalinephosphatase',
         'Baseexcess', 'Basophils', 'Rods', 'Internationalnormalizedratio',
         'ProteinaCreativa mgdL', 'Patientaddmitetosemiintensiveunit',
         'Promyelocytes', 'RedbloodCells', 'Creatinephosphokinase', 'Urea',
         'Metapneumovirus', 'ArterialLacticAcid', 'Bordetellapertussis',
         'Myelocytes', 'Ferritin', 'Magnesium', 'Hbsaturation', 'Sodium',
         'Parainfluenza2', 'DirectBilirubin', 'Neutrophils']
    y = ['diagnosis']
```

Figura 31 – Início do loop das massas de dados separados pelo SKF e as 81 features selecionadas - Fonte: próprio autor

Para utilizar esses valores de X, precisam passar por algumas etapas de formatação. Na figura 32 as variáveis de treino e teste utilizarão seus índices para localizar na base de dados seus valores e armazenar nas suas respectivas variáveis para que as outras variáveis como X_train/y_train e X_test/y_test sejam criadas e transformadas pela função to_numpy (convertendo o formato DataFrame do pandas para um array do NumPy).

Por conta dos dados serem tabulares, a rede CNN não consegue processar sua dimensão corretamente. Utilizando a função reshape do numpy serão acrescentados nos dados das features (X) uma nova dimensão e depois a função *flatten* organiza os dados da classe final (y) linearmente no array. Ao finalizar é preciso garantir que todos os dados possuam os mesmos tipos de valores, com a função astype os dados de todas as massas X são convertidos para float32 e as y para int32.

```
train = df.loc[train_index,:]
test = df.loc[test_index,:]
X_train = train[X].to_numpy()
y_train = train[y].to_numpy()
X_test = test[X].to_numpy()
y_test = test[y].to_numpy()
X_train = np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
y_train = y_train.flatten()
y_test = y_test.flatten()
X_train = X_train.astype('float32')
y_train = y_train.astype('int32')
X_test = X_test.astype('float32')
y_test = y_test.astype('int32')
```

Figura 32 – Construção e formatação das massas de treino e teste - Fonte: próprio autor

Concluídas as formatações segue o programa com a criação da variável de entrada da rede utilizando os valores das duas últimas dimensões da base de treino. Passando para a função create_model esses valores, o modelo será criado e compilado. Para medir o tempo de treino mencionado anteriormente, a função perf_counter começa um cronômetro antes da função de treino da rede (fit) e finaliza esse *timer* após sua execução. Na figura 33 a função fit recebe os seguintes parâmetros:

- X_train: dados de treino com as *features* selecionadas;
- y_train: dados de treino da classe final ao qual a rede deve acertar;
- *epochs*: quantas vezes os dados devem passar pela rede, no trabalho o número escolhido foi 200;
- *batchsize*: para cada vez que os dados passaram na rede, devem possuir um tamanho n de dimensão. O valor escolhido foi 32.
- *verbose*: mostrar (1) ou não (0) a execução da rede linha por linha. Nesse caso foi escolhido não (0) mostrar a execução toda no resultado.

```
input_shape = (X_train.shape[1],X_train.shape[2])
create_model(input_shape)
tic = time.perf_counter()
history = model.fit(X_train, y_train, epochs=200, batch_size=32, verbose=0)
toc = time.perf_counter()
y_pred = model.predict(X_test)
```

Figura 33 – Execução, avaliação e treino do modelo - Fonte: próprio autor

Para finalizar com a função de predição (*predict*) dos dados que foram separados com a massa de teste, diferente do método *fit* que treina a rede. Com esse resultado, o

valor é passado para as métricas de avaliação, como mostra a figura 34. Nela encontra-se as seis métricas utilizando as seguintes funções: r2_score, accuracy_score, precision_score, recall_score, f1_score e confusion_matrix. Após seu cálculo, os dados são adicionados as suas respectivas listas para serem plotados em um gráfico ao final da execução do *loop*.

```
#r2score
R2Score = r2_score(y_test, np.int_(y_pred.flatten()))
r2s.append(R2Score)
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test,np.int_(y_pred.flatten()))
acc.append(accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, np.int_(y_pred.flatten()), average='weighted')
prec.append(precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, np.int_(y_pred.flatten()))
rec.append(recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, np.int_(y_pred.flatten()))
f1s.append(f1)
#confusion matrix
cm = confusion_matrix(y_test, np.int_(y_pred.flatten()))
cml.append(cm)
```

Figura 34 – Métricas escolhidas para avaliação do modelo - Fonte: próprio autor

Com os dados do *timer* e das métricas da rede guardados em suas variáveis, foi construído um gráfico que demonstra a curva de aprendizado da rede durante o treino e seu tempo médio acima na figura 35. Cada gráfico plotado representa uma métrica de treino utilizada para verificar seu aprendizado, só que nesse caso o gráfico gerado foi linear para cada uma das métricas. Demonstrando que os problemas existentes na base, mesmo os dados sendo tratados, não foi possível treinar a rede para todos os dados. Realçando o problema em seu aprendizado, não importando se há 1 ou 200 *epochs*.

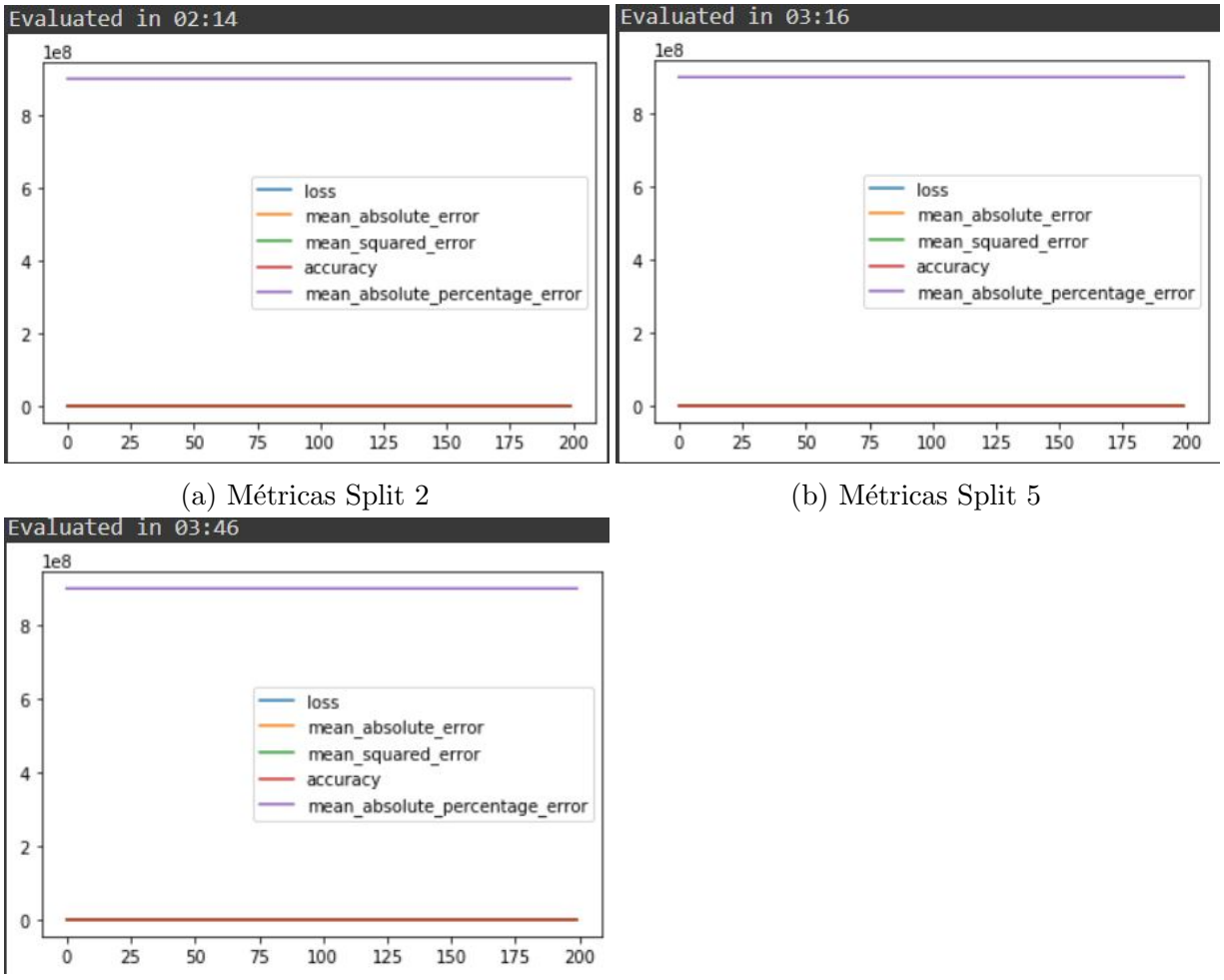
```
print(secondsToMinutesSeconds(toc - tic))

line1, = plt.plot(history.history['loss'], label='loss')
line2, = plt.plot(history.history['mae'], label='mean_absolute_error')
line3, = plt.plot(history.history['mse'], label='mean_squared_error')
line4, = plt.plot(history.history['accuracy'], label='accuracy')
line5, = plt.plot(history.history['mape'],
                  label='mean_absolute_percentage_error')
leg = plt.legend(loc='center right')
plt.show()
```

Figura 35 – Captura de tela do tempo que o modelo levou para ser treinado e o gráfico de suas métricas das 200 iterações (*epochs*) - Fonte: próprio autor

Finalizada a iteração no SKF o gráfico das métricas serão gerados utilizando o *plot*, mostrado na figura 36, assim como as métricas de treino anteriores as utilizam. Na figura 37 há três gráficos com seus respectivos tempos de execução e métricas de treino para cada divisão de massas. As figuras 37a, 37b e 37c representam os gráficos dos dados do splitDados de tamanho 2, 5 e 9, respectivamente.

```
line1, = plt.plot(r2s, label='R2_score')
line2, = plt.plot(acc, label='Accuracy')
line3, = plt.plot(prec, label='Precision')
line4, = plt.plot(rec, label='recall')
line5, = plt.plot(f1s, label='f1_score')
leg = plt.legend(loc='center right')
plt.xticks(np.arange(0, splitDados, 1))
plt.yticks(np.arange(-0.2, 1.1,0.1))
plt.figure()
plt.show()
```

Figura 36 – Criação do gráfico com as métricas finais da predição final do modelo - Fonte: próprio autor



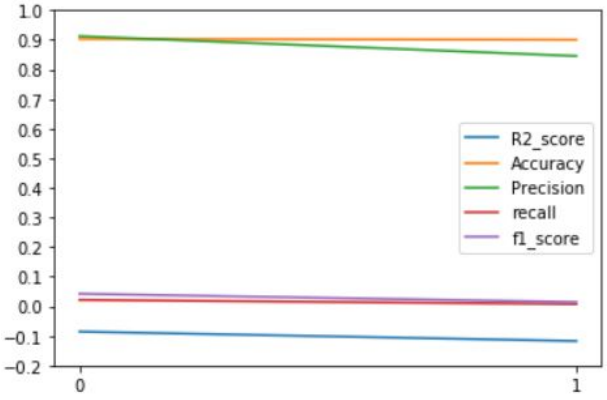

(a) Métricas Split 2

(b) Métricas Split 5

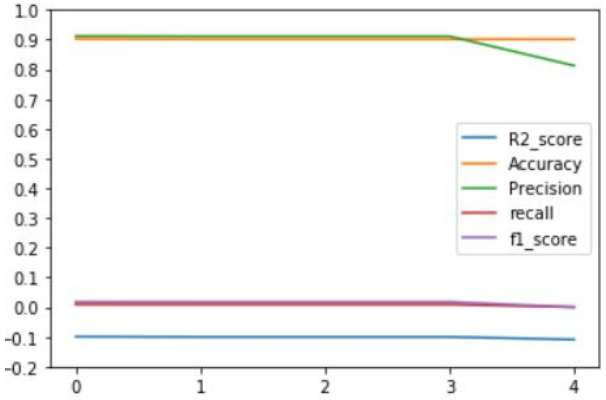
(c) Métricas Split 9

Figura 37 – Gráfico com as métricas e tempo médio de treino para de cada Split que o SKFold realizou ao separar os dados de teste e treino - Fonte: próprio autor

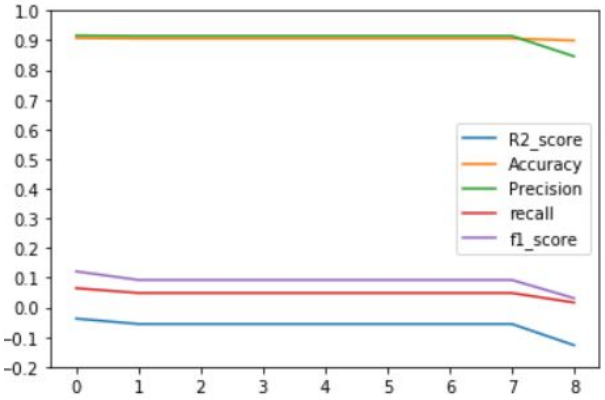
Os resultados das métricas de avaliação para cada massa estão demonstrados na figura 38. Analisando os resultados dos gráficos 38a, 38b e 38c é possível verificar que a acurácia ficou em 90% e a precisão começou um pouco maior e finalizou em quase 80%, demonstrando assim o desbalanceamento dos dados para todas as divisões de massas. As outras métricas (R2score, *recall* e f2Score) obtiveram um padrão semelhantes que as outras, começa com um valor mais alto e, na última iteração, finaliza com um valor mais baixo. Apenas no último gráfico consegue notar uma variância maior nos valores das massas divididas em 9 partes. Mantendo o padrão, alto na primeira massa e baixo na última massa.




(a) Métricas de Avaliação pós-predição Split 2



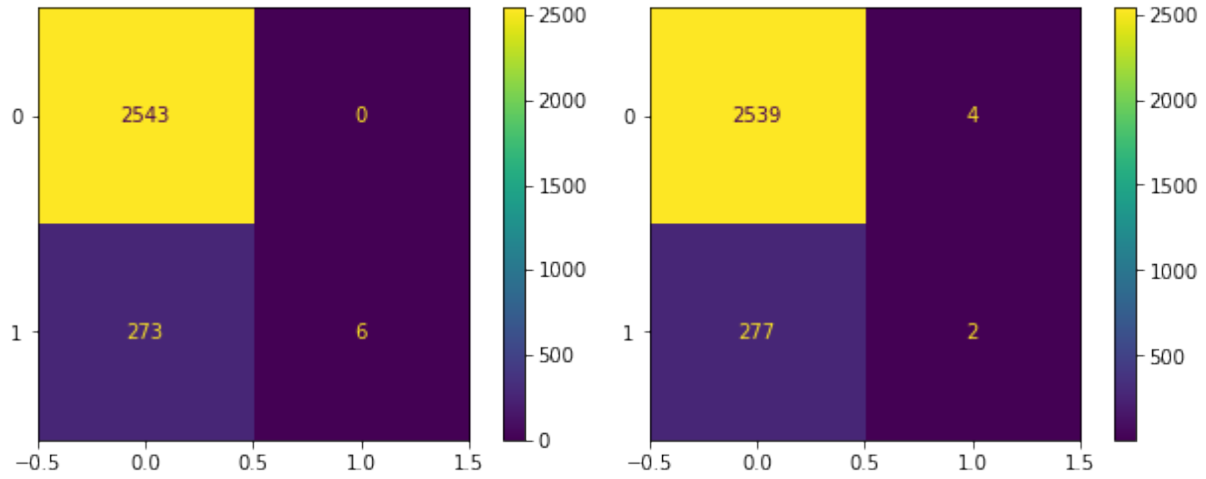

(b) Métricas de Avaliação pós-predição Split 5




(c) Métricas de Avaliação pós-predição Split 9

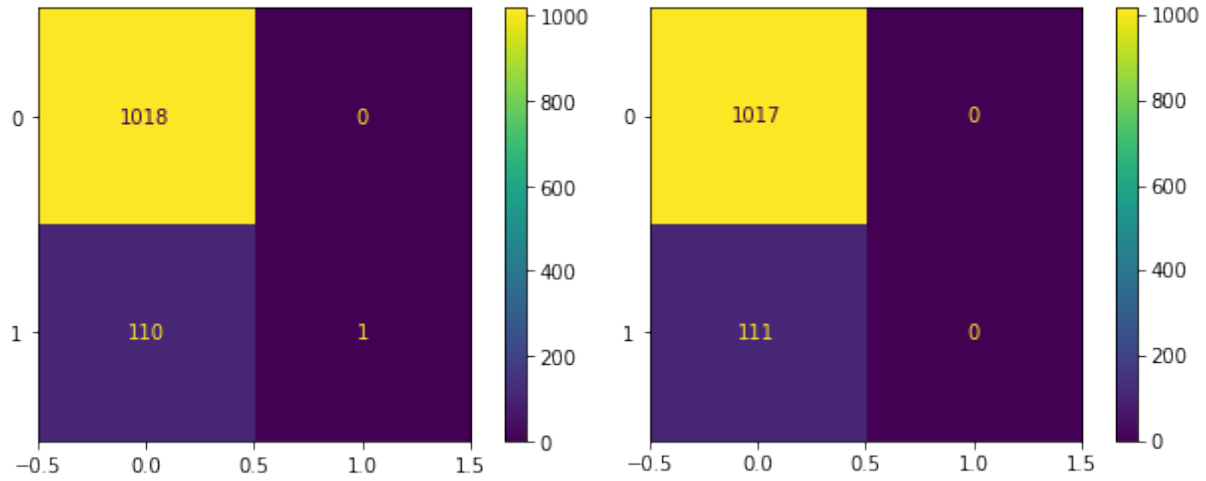
Figura 38 – Gráfico com os resultados de cada métrica após a predição dos dados - Fonte: próprio autor

A última métrica a ser avaliada é a matriz de confusão, mostrada nas figuras 39, 40 e 41. No primeiro conjunto temos duas massas avaliadas, o primeiro gráfico 39a mostra que teve 2543 TN e 6 TP e no segundo 39b temos 2539 TN e 2 TP. No split 5, as quatro primeiras massas demonstradas no gráfico 40a possuem os mesmos erros e acertos com 1018 TN e 1 TP e no gráfico 40b da última massa não acertou nenhum verdadeiramente positivo e aumentou o erro dos negativos. Por último no split9, o gráfico 41a tem como acertos TN 566 e 4 TP e a última massa do gráfico 41c tem 566 TN e 1 TP. Já nas massas 2 a 7 do gráfico 41b temos um comportamento semelhante ao do split 5, todas as massas obtiveram os mesmos resultados 565 TN e 3 TP.



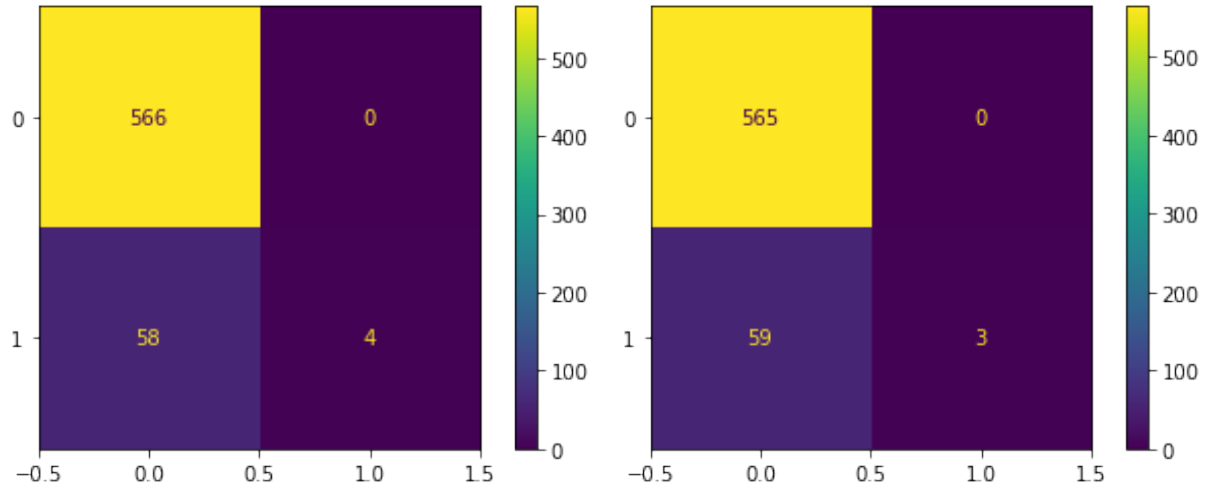

(a) Split 1 (b) Split 2

Figura 39 – Matriz de confusão Split2 - Fonte: próprio autor



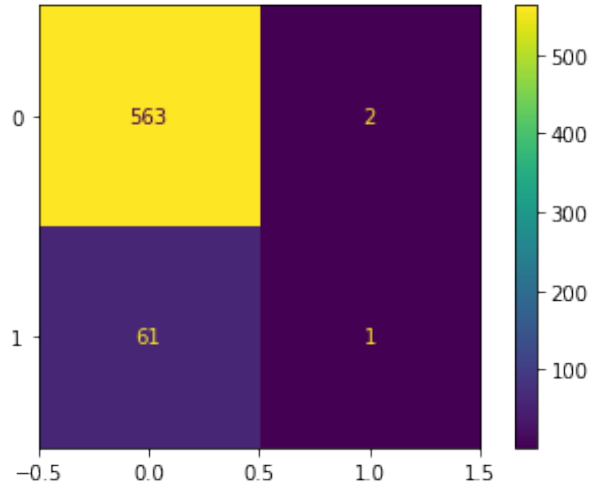

(a) Split 1 a 4 (b) Split 5

Figura 40 – Matriz de confusão Split5 - Fonte: próprio autor




(a) Split 1

(b) Split 2 a 7




(c) Split 9

Figura 41 – Matriz de confusão Split 9 - Fonte: próprio autor

Com as métricas geradas e avaliadas, pode-se comparar os resultados das matrizes de confusão com os gráficos das métricas. Quando a divisão de massas aumenta, o *split* é maior, menor é a diferença nos acertos e erros do modelo.

Em comparação com os dados encontrados pelo algoritmo desenvolvido pelo (CARDOSO, 2022), conseguimos ver algumas:

Tabela 1 – Tabela comparativa - Fonte: (CARDOSO, 2022) e próprio autor

| | Mediana | CNN |
|---|---|---|
| Acurácia | 0,9 | 0,90 |
| Precisão | 0,91 | 0,88 |
| F1-score | 0,03 | 0,03 |

# 6 CONCLUSÃO

A realização deste trabalho iniciou durante o auge da pandemia da Covid-19, em um momento que o mundo todo estava tentando descobrir técnicas e ferramentas para diagnosticar pacientes e realizar o tratamento correto. Visando criar uma rede neural para classificação de dados de Covid-19 baseados em dados clínicos e de exames de pacientes que foram ao hospital com suspeita da doença. Essa base de dados possui uma abundância de pacientes, porém continham valores nulos, vazios, com formatos diferentes e ainda desbalanceados (mais dados da classe negativos do que positivos).

Mesmo com esses problemas na base, algumas medidas de transformação de dados foram utilizadas para que fosse possível uma análise inicial dos dados. O uso dos métodos de replace do DataFrame foram essenciais para que a base fosse estudada pela rede. Pois o objetivo principal do projeto era verificar a possibilidade de utilizar a base de dados, com todos seus problemas citados, para treinar o modelo de CNN 1D desenvolvido para classificação dos pacientes doentes e saudáveis.

O resultado da rede neural na métrica de precisão foi em média 90% e precisão em média de 80%. No entanto, nas outras métricas, os resultados não foram satisfatórios, ficando muito próximo de 0. A CNN não foi capaz de aprender com essa base de dados, e o principal problema foi o desbalanceamento dos dados, gerando resultados altos apenas para os dados que estavam em maior quantidade (dados negativos). Por isso a importância de analisar várias métricas, principalmente ao ter uma base de dados tão desbalanceada como a utilizada nesse estudo de caso. Ao comparar com os trabalho do (CARDOSO, 2022) que também utilizou essa base de dados, porém com uma técnica baseada em árvore de decisão e também apresentou as mesmas dificuldades. Fica claro que seria necessário um aumento na quantidade de pacientes que possuem a doença, para possibilitar que a rede consiga aprender sobre essa classe também.

Mesmo os resultados não tendo sido viáveis para as duas classes, o projeto foi muito importante no desenvolvimento tecnológico do próprio aluno, conseguindo modelar, treinar e avaliar uma rede neural convolucional, bem como realizar o pré-processamento da base de dados. Neste ponto de vista o trabalho foi um sucesso.

## Referências


AARON, G. I. B. Y. C. *ADeep learning*. 2016. Citado 2 vezes nas páginas 7 e 8.

ACTER, T. e. a. *Evolution of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) as coronavirus disease 2019 (COVID-19) pandemic: A global health emergency*. 2020. 138996 p. Citado na página 2.

AGHDAM HAMED HABIBI; HERAVI, E. J. *Guide to convolutional neural networks*. 2017. 51 p. Citado 3 vezes nas páginas 3, 8 e 9.

AGRAWAL, S. K. *Metrics to Evaluate your Classification Model to take the right decisions*. 2021. <https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/>. Accessed: 22-01-2023. Citado na página 19.

AMIDI, A.; AMIDI, S. *Convolutional Neural Networks cheatsheet*. 2019. <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>. Citado 2 vezes nas páginas  e 12.

ARYA, N. *Loss Functions: An Explainer*. 2022. <https://www.kdnuggets.com/2022/03/loss-functions-explainer.html#:~:text=Mean%20Absolute%20Error%20Loss,by%20the%20number%20of%20output.> Citado na página 20.

CARDOSO, I. TÉcnicas de otimizaÇÃo e mÉtricas de avaliaÇÃo aplicadas a machine learning. 2022. Disponível em: <https://repositorio.ifgoiano.edu.br/handle/prefix/2712>. Citado 4 vezes nas páginas , 17, 33 e 34.

CASTRO F. C. C. C.; DE CASTRO, M. C. F. D. *Redes neurais artificiais*. 2001. Citado 2 vezes nas páginas 5 e 6.

CHONG, A. L. Z. *Technology and Epidemics.* 2022. Citado na página 1.

COURSE, G. *Classification: Accuracy*. n.d. <https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=en>. Accessed: 22-01-2023. Citado na página 19.

FLECK, L. e. a. *Redes neurais artificiais: Princípios básicos*. 2016. 47-57 p. Citado 2 vezes nas páginas 5 e 6.

GEOFFREY, L. Y. B. Y. H. *Deep learning*. 2015. 436-444 p. Citado 3 vezes nas páginas 6, 7 e 8.

GOV, i. *COVID-19 NO BRASIL*. n.d. <https://infoms.saude.gov.br/extensions/covid-19_html/covid-19_html.html>. Accessed: 22-01-2023. Citado na página 1.

GOV, O. *Histórico da COVID-19*. 2022. <https://www.paho.org/pt/covid19/historico-da-pandemia-covid-19#:~:text=Em%2031%20de%20dezembro%20de,identificada%20antes%20em%20seres%20humanos>. Accessed: 22-01-2023. Citado na página 1.

GU, J. e. a. *Recent advances in convolutional neural networks*. 2018. 354-377 p. Citado 2 vezes nas páginas 9 e 10.

JúNIOR, C. d. O. *Prevendo Números: Entendendo as métricas $R^2$, MAE, MAPE, MSE e RMSE*. 2021. <https://www.kdnuggets.com/2022/03/loss-functions-explainer.html#:~:text=Mean%20Absolute%20Error%20Loss,by%20the%20number%20of%20output.> Citado na página 20.

KELLEHER, J. D. *Deep Learning*. 2019. Citado 2 vezes nas páginas 7 e 8.

KHORSHED, T.; MOUSTAFA, M. N.; RAFEA, A. Deep learning for multi-tissue cancer classification of gene expressions (genexnet). *IEEE Access*, v. 8, p. 90615–90629, 2020. Citado na página 13.

KORSTANJE, J. *The F1 score*. 2021. <https://towardsdatascience.com/the-f1-score-bec2bbc38aa6>. Accessed: 22-01-2023. Citado na página 20.

KULKARNI, A.; CHONG, D.; BATARSEH, F. A. 5 - foundations of data imbalance and solutions for a data democracy. In: BATARSEH, F. A.; YANG, R. (Ed.). *Data Democracy*. Academic Press, 2020. p. 83–106. ISBN 978-0-12-818366-3. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128183663000058>. Citado na página 20.

KUMAR, A. *Real-World Applications of Convolutional Neural Networks*. 2021. <https://vitalflux.com/real-world-applications-of-convolutional-neural-networks/>. Citado 2 vezes nas páginas  e 11.

LE, T. T. e. a. *Evolution of the COVID-19 vaccine development landscape*. 2020. 667-668 p. Citado na página 2.

LE TUNG THANH; CRAMER, J. P. e. a. *Evolution of the COVID-19 vaccine development landscape*. 2020. <https://www.nature.com/articles/d41573-020-00151-8>. Accessed: 22-01-2023. Citado na página 2.

LELLA, K. K.; PJA, A. Automatic covid-19 disease diagnosis using 1d convolutional neural network and augmentation with human respiratory sound based on parameters: cough, breath, and voice. *AIMS Public Health*, 2021. Citado 3 vezes nas páginas , 15 e 16.

LIN, L. et al. Towards automatic depression detection: A bilstm/1d cnn-based model. *Applied Sciences*, v. 10, n. 23, 2020. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/10/23/8701>. Citado 3 vezes nas páginas , 14 e 15.

MELO, D. *O que é Python? [Guia para iniciantes]*. 2021. <https://tecnoblog.net/responde/o-que-e-python-guia-para-iniciantes/>. Accessed: 22-01-2023. Citado na página 21.

MOHAMMAD M R KHAN MAMUN; ALI, A. *FA-1D-CNN Implementation to Improve Diagnosis of Heart Disease Risk Level*. 2020. Citado 3 vezes nas páginas , 13 e 14.

MORENS, D. M. e. a. *The origin of COVID-19 and why it matters*. 2020. 955 p. Citado na página 2.

NADER, V. A. C. G. P. A. V. C. *Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres*. 2016. Citado 2 vezes nas páginas 9 e 10.

NETO, S. A. d. L. H. C. *Reconhecimento de tumores cerebrais usando redes neurais convolucionais*. 2017. 68 p. Citado na página 1.

OLIVEIRA PATRICK FRANCISCO; C MARA, C. E. *Análise de desempenho de um algoritmo desenvolvido para solução de deep learning utilizando redes neurais convolucionais para análise de contraste de imagens*. 2019. 84-105 p. Citado 5 vezes nas páginas 1, 3, 8, 9 e 10.

O'SHEA KEIRON; NASH, R. *An introduction to convolutional neural networks*. 2015. Citado 3 vezes nas páginas 3, 9 e 10.

OSÓRIO FERNANDO S.; BITTENCOURT, J. R. *Sistemas inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens*. 2000. Citado na página 6.

PACHECO CéSAR AUGUSTO RODRIGUES; PEREIRA, N. S. *ADeep learning conceitos e utilização nas diversas Áreas do conhecimento*. 2018. 34-49 p. Citado 2 vezes nas páginas 7 e 8.

PERE, C. *What are Loss Functions?* 2020. <https://towardsdatascience.com/what-is-loss-function-1e2605aeb904>. Citado na página 20.

RAJ, R. *Supervised, Unsupervised and Semi-supervised Learning with Real-life Usecase*. n.d. <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>. Citado 3 vezes nas páginas , 4 e 5.

RAUBER, T. W. *Redes neurais artificiais*. 2005. Citado 2 vezes nas páginas 5 e 6.

ROWE, W. *Mean Square Error  R2 Score Clearly Explained*. 2018. <https://www.bmc.com/blogs/mean-squared-error-r2-and-variance-in-regression-analysis/>. Citado na página 20.

RUSSEL STUART; NORVIG, P. *Artificial Intelligence - A Modern Approach*. 2009. Citado 2 vezes nas páginas 2 e 3.

SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Citado 3 vezes nas páginas , 11 e 12.

SPHINX, E. *Pooling (CNN)*. 2021. <https://epynn.net/Pooling.html>. Citado 2 vezes nas páginas  e 10.

TAYO, B. O. *What Really is R2-Score in Linear Regression?* 2021. <https://benjaminobi.medium.com/what-really-is-r2-score-in-linear-regression-20cafdf5b87c>. Citado 2 vezes nas páginas  e 20.

UNSUPERVISED, S. vs. *Supervised vs Unsupervised Learning Explained*. 2022. <https://www.seldon.io/supervised-vs-unsupervised-learning-explained>. Citado na página 4.

WANG, L. et al. Artificial intelligence for covid-19: A systematic review. *Frontiers in Medicine*, v. 8, 2021. ISSN 2296-858X. Disponível em: <https://www.frontiersin.org/articles/10.3389/fmed.2021.704256>. Citado na página 1.