ERCEMAPI

Escola Regional de Computação do Ceará, Maranhão e Piauí

LIVRO DE MINICURSOS

X EDIÇÃO/2022

ORGANIZADORES

ARIEL S. TELES

DANILO B. DA SILVA

GUILHERME A. R. M. ESMERALDO

REALIZAÇÃO

SBC Sociedade Brasileira de Computação

ORGANIZAÇÃO

Instituto Federal Maranhão

UFMA Universidade Federal do Maranhão

Universidade Estadual do Maranhão

UNDB Centro Universitário

APOIO

FAPEMA Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão

at Hand Tecnologia Corporativa

NCA Núcleo de Computação Aplicada

Golden Shopping Calhau É fácil ser feliz aqui

DComp Departamento de Computação

FOCO

Loggi

Fábrica de inovação

pulse by Grupo Mateus

X Escola Regional de Computação Ceará, Maranhão e Piauí

28 a 30 de setembro de 2022

LIVRO DE MINICURSOS

ERCEMAPI 2022

Organização do Livro

ARIEL SOARES TELES (IFMA)

DANILO BORGES DA SILVA (UESPI)

GUILHERME ALVARO RODRIGUES MAIA ESMERALDO (IFCE)

Coordenação Geral

DANIEL LIMA GOMES JUNIOR (IFMA)

JOÃO OTÁVIO BANDEIRA DINIZ (IFMA)

Sociedade Brasileira da Computação

Porto Alegre

2022

Dados Internacionais de Catalogação na Publicação (CIP)

E74 Escola Regional de Computação do Ceará, Maranhão e Piauí (10. : 28 – 30 set. 2022 : São Luís)
Minicursos da ERCEMAPI 2022 [recurso eletrônico] / organização: Ariel Soares Teles ; Danilo Borges da Silva ; Guilherme Alvaro Rodrigues Maia Esmeraldo. Dados eletrônicos. – Porto Alegre: Sociedade Brasileira de Computação, 2022.
130 p. : il. : PDF ; 10.7MB

Modo de acesso: World Wide Web.
Inclui bibliografia
ISBN 978-85-7669-514-1 (e-book)

1. Computação – Brasil – Evento. 2. Dados espaciais. 3. Machine learning. 4. Dispositivos eletroeletrônicos. 5. Redes neurais profundas. 6. Saúde pública. I. Teles, Ariel Soares. II. Silva, Danilo Borges da. III. Esmeraldo, Guilherme Alvaro Rodrigues Maia. IV. Sociedade Brasileira de Computação. VI. Instituto Federal do Maranhão. VII. Título.

CDU 004(063)

Ficha catalográfica elaborada por Jéssica Paola Macedo Müller – CRB-10/2662

Biblioteca Digital da SBC – SBC OpenLib

Índices para catálogo sistemático:

1. Ciência e tecnologia dos computadores : Informática – Publicação de conferências, congressos e simpósios etc. ... 004(063)

Sociedade Brasileira de Computação – SBC

Contato

Av. Bento Gonçalves, 9500

Setor 4 - Prédio 43.412 - Sala 219

Bairro Agronomia

91.509-900 – Porto Alegre RS

CNPJ: 29.532.264/0001-78

http://www.sbc.org.br

SINOPSE EM PORTUGUÊS

O Livro de Minicursos da ERCEMAPI 2022 aborda conteúdos relacionados à ciência de dados, inteligência artificial, visão computacional e informática em saúde. No primeiro capítulo, intitulado “Análise Exploratória de Dados Espaciais com Python” os autores abordam a análise exploratória de dados espaciais, associando a teoria à prática com a linguagem Python. No segundo capítulo, “Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional”, os autores apresentam um método para a utilização de técnicas de visão computacional no controle de dispositivos eletrônicos. O terceiro capítulo “Explainability e auditability: interpretando e validando modelos de machine learning” está situado em uma área de bastante crescimento nos últimos anos, a inteligência artificial explicável. O quarto capítulo “Introdução às Redes Neurais Profundas com Python” introduz não somente os conceitos, mas também exemplos práticos sobre o aprendizado profundo com redes neurais. No quinto capítulo, “Desenvolvimento Ágil e Informatização da Saúde Pública no Brasil”, os autores apresentam práticas ágeis utilizadas para o desenvolvimento de sistemas de grande porte para o Ministério da Saúde do Brasil. O sexto e último capítulo, com o título “Desenvolvimento de uma aplicação web para o diagnóstico da COVID-19 usando conceitos, técnicas e ferramentas de Inteligência Artificial”, discorre sobre o desenvolvimento de uma aplicação web para o diagnóstico da COVID-19, por meio de imagens de radiografia do tórax, usando conceitos, técnicas e ferramentas de inteligência artificial. Os seis capítulos deste livro possuem metodologias e ferramentas para a área de tecnologia da informação, sendo uma obra útil para pessoas que querem iniciar nas respectivas áreas abordadas e também ganhar conhecimento em tópicos de pesquisa bastante atuais.

SINOPSE EM INGLÊS

The ERCEMAPI 2022 Book of Mini-courses covers content related to data science, artificial intelligence, computer vision and health informatics. In the first chapter, entitled “Análise Exploratória de Dados Espaciais com Python”, the authors approach the exploratory analysis of spatial data, associating theory with practice using the Python language. In the second chapter, “Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional”, the authors present a method for using computer vision techniques to control electronic devices. The third chapter “Explainability e auditability: interpretando e validando modelos de machine learning” is situated in an area of considerable growth in recent years, the explainable artificial intelligence. The fourth chapter “Introdução às Redes Neurais Profundas com Python” introduces not only the concepts, but also practical examples on deep learning with neural networks. In the fifth chapter, “Desenvolvimento Ágil e Informatização da Saúde Pública no Brasil”, the authors present agile practices used for the development of large systems for the Ministry of Health in Brazil. The sixth and final chapter, entitled “Desenvolvimento de uma aplicação web para o diagnóstico da COVID-19 usando conceitos, técnicas e ferramentas de Inteligência Artificial”, discusses the development of a web application for the diagnosis of COVID-19, using chest radiography images, with artificial intelligence concepts, techniques and tools. The six chapters of this book have methodologies and tools for the area of information technology, and is a useful work for people who want to start in the respective areas covered, as well as acquire knowledge in very current research topics.

Mensagem da Coordenação Geral

Podemos começar afirmando que a ideia da organização de escolas regionais é brilhante, pois aproxima pessoas que estão próximas, mas muitas vezes distantes. E nessa linha de pensamento, a Escola Regional de Computação Ceará, Maranhão e Piauí cumpre um papel fundamental para o despertar científico de alunos desses estados.

Essa afirmação foi exemplificada na última palestra do evento de 2022 (do prof. Dr. Rafael Ferreira, da UFRPE), em que ele afirmou que seu interesse científico foi "despertado" ao participar de um evento organizado por uma escola regional, ainda como aluno de graduação. É exatamente esse nosso papel enquanto professores, proporcionar essas experiências aos nossos alunos.

Vale ressaltar que, em geral, nas edições realizadas no estado do Maranhão, a Universidade Federal do Maranhão (UFMA), na qual está instituída a Secretaria Regional do Maranhão, tem sido a instituição organizadora, e sempre fez isso em um alto nível de organização. No entanto, no ano de 2022, a convite dos professores Anselmo Cardoso de Paiva e Davi Viana (atual secretário regional do MA), fomos convidados em nome do IFMA para o desafio de organizar a X ERCEMAPI.

Hoje, ao escrever essa mensagem, muito nos orgulha dizer que fizemos o melhor possível, afinal, estamos saindo de uma pandemia, e durante muito tempo da preparação do evento pairava a dúvida se o evento seria apenas remoto ou se tentaríamos um formato híbrido. Acreditamos que tomamos a decisão correta e tivemos a coragem em realizar o evento com as atividades presenciais.

O melhor de tudo tem sido o feedback de alguns alunos que participaram das ações presenciais. Alunos que nunca tinham participado de um evento acadêmico por falta de oportunidade e recursos, e que estavam muito felizes com os conteúdos apresentados. Alguns desses feedbacks foram registrados no instagram do evento (@ercemapi2022.oficial) em formato de stories. E esse retorno dos alunos nos faz acreditar que a 10ª edição foi um sucesso, pois o esforço dos professores e da comissão organizadora tinha o objetivo de fazer o melhor para os nossos alunos.

Além disso, academicamente falando, tivemos uma excelente participação de diversas instituições de pesquisa e um bom número de submissões de artigos e de minicursos, o que consolida o excelente trabalho realizado pelos comitês definidos, em especial nesse livro, agradecemos ao comitê de minicursos, organizado pelo prof. Dr. Ariel Soares Teles (IFMA), prof. Dr. Guilherme Alvaro Rodrigues Maia Esmeraldo (IFCE) e prof. Me. Danilo Borges da Silva (UESPI).

E esse crescimento do interesse em publicar na ERCEMAPI, sugere que estamos no caminho certo. Que o evento possa crescer a cada ano e que as publicações possam ajudar no crescimento e divulgação da computação.

Que tenham uma excelente leitura.

Coordenação Geral da X ERCEMAPI
Prof. Dr. Daniel Lima Gomes Júnior
Prof. Dr. João Otávio Bandeira Diniz

Mensagem da Coordenação de Minicursos

O Livro de Minicursos da ERCEMAPI 2022, mais uma vez, busca fortalecer a missão da Escola Regional de Computação do Ceará, Maranhão e Piauí, que é difundir conhecimentos técnico-científicos e tecnologias de ponta nas diversas áreas da Computação. Em sua 10ª edição, o livro com os minicursos abordam conteúdos relacionados à ciência de dados, inteligência artificial, visão computacional e informática em saúde, como forma de atualizar os conhecimentos da comunidade acadêmica e profissional, de uma forma didática e de amplo acesso ao público. Os seis minicursos foram selecionados através de um processo de revisão das propostas por pares do tipo "*blind*", em que os revisores tiveram a oportunidade de saber quem eram os autores proponentes e analisar seus currículos.

Os seis capítulos deste livro discorrem sobre metodologias, técnicas e ferramentas para a área de Tecnologia da Informação e Comunicação, sendo uma excelente oportunidade para familiarização dos interessados com novos temas de pesquisa que podem ser úteis em suas vidas profissionais e acadêmicas.

Agradecemos a todos os revisores que compuseram o Comitê de Programa de Minicursos, por suas valiosas contribuições para os trabalhos e dedicação no processo de revisão. Em especial, agradecemos aos coordenadores gerais da ERCEMAPI 2022, o prof. Dr. Daniel Lima Gomes Júnior (IFMA) e o prof. Dr. João Otávio Bandeira Diniz (IFMA), por todo o suporte durante o evento e a produção deste livro.

Finalmente, desejamos que todos os leitores aproveitem o conteúdo apresentado.

Coordenação de Minicursos da X ERCEMAPI
Prof. Dr. Ariel Soares Teles
Prof. Me. Danilo Borges da Silva
Prof. Dr. Guilherme Alvaro Rodrigues Maia Esmeraldo

Comitê de Programa de Minicursos

Atslands Rego da Rocha (UFC)
Carina Teixeira de Oliveira (IFCE)
Davi Viana dos Santos (UFMA)
Diego Rocha Lima (IFCE)
Fabbio Anderson Silva (UESPI)
Francisco José da Silva e Silva (UFMA)
João Otávio Bandeira Diniz (IFMA)
Luciano Reis Coutinho (UFMA)
Omar Andres Carmona Cortes (IFMA)
Raimundo Valter Costa Filho (IFCE)

Secretarias Regionais SBC

Danielo Gonçalves Gomes (UFC / SR Ceará)
Davi Viana (UFMA / SR Maranhão)
Rodrigo Augusto Rocha Souza Baluz (UESPI / SR Piauí)

Sumário

Capítulo 1

Análise Exploratória de Dados Espaciais com Python

Gesiel Rios Lopes, Karina Jorge Pelarigo, Alexandre C. B. Delbem, Joélcio Braga de Sousa

Abstract

The sharp development of technologies for data analysis in the geographic space that occurred in recent years has offered innovative possibilities to understand the influence of spatial effects in explaining various phenomena and constitutes an excellent challenge for several areas of scientific knowledge. Exploratory Spatial Data Analysis (ESDA) is the extension of Exploratory Data Analysis (EDA) to the problem of detecting spatial properties of data sets where there is locational data for each attribute value. This location data references the point or area to which the attribute refers. ESDA is a set of techniques that aim to describe and visualize spatial distributions, identify atypical or spatial outliers, discover spatial association patterns and clusters, and suggest spatial regimes or other forms of spatial heterogeneity. This chapter aims to align theory and practice, presenting some concepts and techniques associated with ESDA with python, focusing on tabular vector data.

Resumo

O desenvolvimento acentuado de tecnologias para análise de dados no espaço geográfico ocorrido nos últimos anos tem oferecido possibilidades inovadoras ao entendimento da influência dos efeitos espaciais na explicação de vários fenômenos e constitui um grande desafio para diversas áreas do conhecimento científico. A análise exploratória de dados espaciais (AEDE) é a extensão da análise exploratória de dados (AED) para o problema de detecção de propriedades espaciais de conjuntos de dados onde, para cada valor de atributo, existe um dado locacional. Este dado de localização referencia o ponto ou a área à qual o atributo se refere. AEDE é um conjunto de técnicas que visa descrever e visualizar distribuições espaciais, identificar locais atípicos ou discrepantes espaciais, descobrir padrões de associação espacial, clusters e sugerir regimes espaciais ou outras formas de heterogeneidade espacial. O objetivo deste minicurso é alinhar a teoria e a prática, apresentando alguns dos conceitos e técnicas associadas à AEDE com python, com foco em dados vetoriais tabulares.

1.1. Introdução

A compreensão da influência dos efeitos espaciais na explicação de vários fenômenos constitui um grande desafio para diversas áreas do conhecimento, seja em saúde, em geologia, em agronomia, computação ou entre tantas outras. Tais estudos vêm se tornando cada vez mais comum, devido o aumento significativo na disponibilidade de dados com informações geoespaciais (tempo e espaço) e o rápido desenvolvimento de tecnologias para análise desses dados [Monteiro et al. 2004, Almeida 2012, Andrade et al. 2007, Lopes et al. 2021, Domingues et al. 2020].

A Análise Exploratória de Dados Espaciais (AEDE) é a coleção de técnicas para descrever e visualizar distribuições espaciais, identificar localidades atípicas (*outliers* espaciais), descobrir padrões de associação espacial (*clusters* espaciais) e sugerir diferentes regimes espaciais e outras formas de instabilidade espacial [Almeida 2012, Anselin 2005].

AEDE permite considerar o contexto espacial em que ocorrem a maioria dos eventos geograficamente referenciados. Esses eventos não podem ser vistos como desconexos e espacialmente independentes, caso contrário, as conclusões sobre os dados e, mais importante, as considerações teóricas sobre os processos sociais que ocorrem no espaço seriam negligenciadas [Haining et al. 1998].

As técnicas de AEDE são geralmente divididas em dois grupos principais: ferramentas para analisar autocorrelação espacial global e local. O primeiro considera a tendência geral que a localização dos valores segue e possibilita afirmações sobre o grau de agrupamento no conjunto de dados. Os valores geralmente seguem um padrão particular em sua distribuição geográfica? Os valores semelhantes estão mais próximos de outros valores semelhantes do que esperaríamos por puro acaso? Estas são algumas das questões que as ferramentas de autocorrelação espacial global permitem responder.

As ferramentas para autocorrelação espacial local, focam na instabilidade espacial: a partida de partes de um mapa da tendência geral. A ideia aqui é que, embora haja uma determinada tendência para os dados em termos da natureza e força da associação espacial, algumas áreas particulares podem divergir substancialmente do padrão geral. Independentemente do grau geral de concentração nos valores, podemos observar bolsões de valores anormalmente altos ou baixos próximos a outros valores altos ou baixos, no que chamaremos de pontos quentes ou frios. Além disso, também é possível observar alguns valores altos ou baixos cercados por valores baixos, ou altos, e chamaremos esses de "*outliers* espaciais". A principal técnica que revisaremos nesta sessão para explorar a autocorrelação espacial local são os Indicadores Locais de Associação Espacial (do inglês, *Local Indicators of Spatial Association* - LISA).

1.1.1. Por que usar python ?

Python é uma linguagem de programação interpretada de uso geral, ao contrário de R ou Matlab, centrada na legibilidade do código, o que a torna uma linguagem fácil de aprender. Python é uma linguagem dinâmica sendo adequada para desenvolvimento interativo e prototipagem rápida com o poder de suportar o desenvolvimento de grandes aplicativos, também é uma linguagem amplamente usada para aprendizado de máquina e ciência de dados devido ao excelente suporte à biblioteca. Ela rapidamente se tornou uma das

plataformas dominantes para praticantes de aprendizado de máquina e ciência de dados [Lopes et al. 2019]

Para o desenvolvimento deste minicurso, será utilizado o ambiente do *Jupyter Notebook*, uma interface de programação literária interativa[1] muito utilizada para prototipar e compartilhar *scripts* de soluções em ciência de dados [Kluyver et al. 2016]. Em [Pimentel et al. 2021] e [Lopes et al. 2019] pode ser encontrado um vasto material de como utilizar o *Jupyter Notebook*.

1.1.2. Biblioteca Python para Análise Exploratória de Dados Espaciais

Para este minicurso, usaremos a biblioteca de análise espacial Python *PySAL*, uma biblioteca multiplataforma de código aberto para ciência de dados geoespaciais com ênfase em dados vetoriais escrita em Python [Rey and Anselin 2007]. *Pysal* suporta o desenvolvimento de aplicações de alto nível para análise espacial, como:

- detecção de *clusters* espaciais, *hot-spots* e *outliers*;
- construção de gráficos a partir de dados espaciai;
- análise exploratória de dados espaço-temporais, dentre outras funcionalidades.

PySAL é uma família de pacotes para ciência de dados espaciais, dividido em quatro componentes principais:

- **Lib:** Estruturas de dados espaciais, IO de arquivo. Construção e edição interativa de matrizes e gráficos de pesos espaciais. Formas alfa, índices espaciais e relações espaço-topológicas.
- **Explore:** Módulos para realizar análises exploratórias de dados espaciais e espaço-temporais.
- **Model:** Estimativa de relações espaciais em dados com uma variedade de modelos lineares, lineares generalizados, aditivos generalizados e não lineares
- **Viz:** Visualize padrões em dados espaciais para detectar *clusters*, *outliers* e pontos de acesso.

Uma visão geral dos principais componentes do *PySAL* é apresentada na Figura 1.1. Ele é organizado em seis categorias principais de funcionalidade que lidam com operações básicas de dados, como a construção e manipulação de pesos espaciais e funções essenciais de geometria computacional, exploração de dados como métodos de agrupamento e análise exploratória de dados espaciais, e modelagem espacial, como dinâmica espacial e econometria espacial [Rey and Anselin 2007].

[1] O paradigma de programação literária visa ajudar na comunicação de programas através da alternância de texto em linguagem natural formatada, pedaços de código executáveis, e resultados de computações [Pimentel et al. 2021]

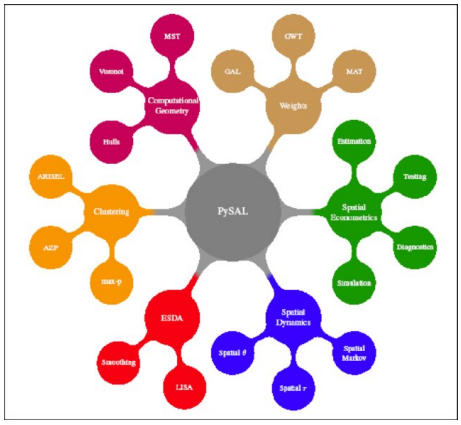


Figura 1.1. Componentes do *PySal* [Rey and Anselin 2007]

1.2. Dados Espaciais

Dados espaciais, dados geográficos ou dados geoespaciais têm duas características muito úteis. Primeiro, os dados geográficos são onipresentes. Tudo tem uma localização no espaço-tempo e essa localização pode ser usada diretamente para fazer melhores previsões ou inferências. Mas, assim como o "tempo" é mais que a posição do relógio, a geografia é mais que a posição da Terra: a localização permite entender as relações entre as observações. Muitas vezes são as relações úteis na ciência de dados porque nos permitem contextualizar nossos dados. Como argumenta o geógrafo Waldo Tobler em [Tobler 1970], *coisas próximas tendem a ser mais relacionadas do que coisas distantes, tanto no espaço quanto no tempo*, também conhecida como Primeira lei da geografia [Almeida 2012]. Portanto, se aprendermos adequadamente com essas informações contextuais, poderemos construir modelos melhores.

Assim como um modelo estatístico, um mapa é uma representação do processo geográfico subjacente, mas não é o processo. Apesar do fato de que essas representações não são exatamente corretas em algum sentido, elas são úteis para entender o que é importante sobre um processo geográfico [Almeida 2012, Andrade et al. 2007, Câmara et al. 2004].

Dados espaciais podem ser definidos como dados ou informações associados às coordenadas geográficas que representam a superfície terrestre. Os tipos mais comuns de dados espaciais são dados *raster* ou matriciais e dados vetoriais.

- Dados *raster* ou matriciais: essa estrutura de dados armazena informações geográficas por meio de grades ou matrizes. Um exemplo clássico de dados *raster* são as imagens de satélite obtidas por meio de sensoriamento remoto, nas quais, cada pixel da imagem estará sempre associado a coordenadas geográficas que representam determinada região da superfície terrestre.

- Dados vetoriais: essa estrutura de dados é representada por informações geográficas associadas a pontos, linhas e polígonos. Os limites de uma propriedade, a localização de um empreendimento e delimitações de estradas, biomas e bacias hidrográficas são exemplos de dados de natureza vetorial.

Neste minicurso, por simplicidade, o foco será em dados vetoriais através das tabelas geográficas do tipo `GeoDataFrame`, uma subclasse do `pandas.DataFrame` capaz de armazenar colunas geométricas e realizar operações espaciais [Lopes et al. 2021]. O primeiro passo, antes de ler alguns dados geoespaciais, é declarar o uso da biblioteca `GeoPandas` (Figura 1.2). Com a declaração do `GeoPandas`, usaremos `%matplotlib inline`, linha 1, uma configuração do `matplotlib` para permitir que os mapas apareçam diretamente no nosso *notebook*, ao invés de serem exibidos em uma janela diferente.

```
1 %matplotlib inline
2 import geopandas as gpd
```

Figura 1.2. Declaração do GeoPandas.

Precisamos agora ter acesso a um conjunto de dados geográficos e existem diversas plataformas governamentais que disponibilizam esses dados gratuitamente, como, por exemplo:

1. INDE (Infraestrutura Nacional de Dados Espaciais),
2. IBGE (Instituto Brasileiro de Geografia e Estatística),
3. ANA (Agência Nacional das Águas), etc.

Assumindo que temos um arquivo contendo dados espaciais, podemos lê-lo facilmente usando a função $gpd.read\_file$, que detecta automaticamente o tipo de arquivo e cria um `GeoDataFrame`. Para criar nosso primeiro mapa, utilizaremos a malha de setores censitários do estado de São Paulo utilizados no CENSO de 2010 do IBGE que pode ser baixado no seguinte endereço: `https://shorturl.at/klAF7`.

A Figura 1.3 apresenta os dois registros do `GeoDataFrame` da malha de setores censitários do estado de São Paulo.

```
1 setores_censitarios_sp = gpd.read_file(
2     'sp_setores_censitarios.zip'
3 )
4 setores_censitarios_sp.head(2)
```

	ID	CD_GEOCODI	TIPO	CD_GEOCODS	NM_SUBDIST	CD_GEOCODD	NM_DISTRIT	CD_GEOCODM	NM_MUNICIP	NM_MICRO	NM_ME
0	98237.0	354100005000009	URBANO	35410000500	None	354100005	PRAIA GRANDE	3541000	PRAIA GRANDE	SANTOS	METROPOLITA DE SÃO PAU
1	98232.0	354100005000004	URBANO	35410000500	None	354100005	PRAIA GRANDE	3541000	PRAIA GRANDE	SANTOS	METROPOLITA DE SÃO PAU

Figura 1.3. Malha de setores censitários do estado de São Paulo.

Cada linha desta tabela é um único setor censitário do estado de São Paulo. Cada setor possui um conjunto de informações como código do setor (CD_GEOCODI), nome do município (NM_MUNICIP), dentre outras. A geometria do limite dos setores censitários é armazenada na coluna de `geometry`. Assim como em outras estruturas de

dados baseadas em tabela em Python, cada linha e coluna tem um índice que as identifica exclusivamente e é renderizada em negrito no lado esquerdo da tabela.

No `geopandas`, assim como em outros pacotes que representam dados geográficos, a coluna de `geometry` possui características especiais que uma coluna "normal" não possui. Por exemplo, quando plotamos o `dataframe`, a coluna `geometry` é usada como a forma principal para plotar, conforme mostrado na Figura 1.4.

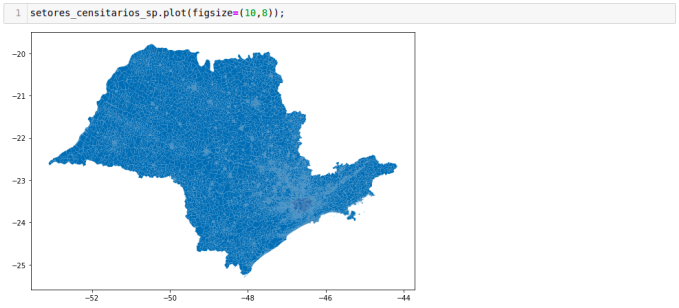


Figura 1.4. Mapa com a malha de setores censitários do estado de São Paulo.

Por ser um objeto derivado do `DataFrame`, é possível manipular o `GeoDataFrame` da mesma forma que manipulamos um `DataFrame`, por exemplo, é possível selecionar apenas a malha de setores censitários da cidade de São Paulo através da função `query`, linha 1 da Figura 1.5.

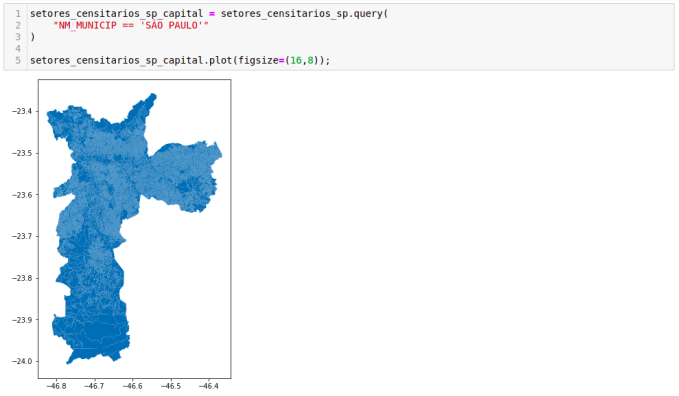


Figura 1.5. Mapa com a malha de setores censitários da cidade de São Paulo - SP.

Em muitos casos, as tabelas geográficas terão geometrias de um único tipo; todos os registros serão `Point` ou `LineString`, por exemplo. No entanto, não há nenhum requisito formal de que uma tabela geográfica tenha geometrias que sejam todas do mesmo tipo.

Ao longo do capítulo usaremos tabelas geográficas extensivamente, armazenando polígonos, mas também pontos e linhas. Maiores detalhes sobre como manipular essas outras formas geométricas com `geopandas` podem ser encontrado em [Lopes et al. 2021].

1.3. Matrizes de Ponderação Espacial

Matrizes de Ponderação Espacial ou "pesos espaciais" são uma maneira de representar gráficos em ciência de dados geográficos e estatísticas espaciais. São construções amplamente utilizadas que representam relações geográficas entre as unidades observacionais em um conjunto de dados referenciado espacialmente. Implicitamente, pesos espaciais conectam objetos em uma tabela geográfica mutualmente usando as relações espaciais entre eles. Ao expressar a noção de proximidade geográfica ou conectividade, os pesos espaciais são o principal mecanismo pelo qual as relações espaciais nos dados geográficos são levadas a efeito na análise subsequente [Almeida 2012, Andrade et al. 2007].

1.3.1. Pesos de Contiguidade

A matriz de pesos espaciais pode ser construída em consonância com a ideia de vizinha baseada na contiguidade, em que duas regiões são vizinhas, caso elas partilhem de uma fronteira física comum [Almeida 2012]. A ideia é que duas regiões contíguas possuem uma maior interação espacial, dessa forma, podemos definir uma matriz de pesos da seguinte forma:

$$w_{ij} = \begin{cases} 1 & \text{se } i \text{ e } j \quad \text{são contíguos} \\ 0 & \text{se } i \text{ e } j \quad \text{não são contíguos} \end{cases} \tag{1}$$

Convencionalmente, é presumido que $w_{ij} = 0$, ou seja, a região não é vizinha dela mesma, implicando que a matriz de contiguidade possua a sua diagonal principal composta por valores nulos. À primeira vista isso parece simples, no entanto, na prática, isso acaba sendo mais complicado. A primeira complicação é que existem diferentes maneiras pelas quais os objetos podem "compartilhar uma borda comum". Começaremos com o exemplo de uma grade de três por três (Figura 1.6), que resulta na grade mostrada na Figura 1.7.

```python
import numpy as np
import matplotlib.pyplot as plt
from shapely.geometry import Polygon

# Get points in a grid
l = np.arange(3)
xs, ys = np.meshgrid(l, l)
# Set up store
polys = []
# Generate polygons
for x, y in zip(xs.flatten(), ys.flatten()):
    poly = Polygon([(x, y), (x+1, y), (x+1, y+1), (x, y+1)])
    polys.append(poly)
# Convert to GeoSeries
polys = gpd.GeoSeries(polys)
gdf = gpd.GeoDataFrame(
    {
        'geometry': polys,
        'id': ['P-%s'%str(i).zfill(2) for i in range(len(polys))]
    }
)
```

Figura 1.6. Código para construção de uma grade 3x3.

Uma forma comum de expressar as relações de contiguidade/adjacência surge de uma analogia com os movimentos legais que diferentes peças de xadrez podem fazer. A contiguidade é considerada como *torre* (*rook*) caso apenas as fronteiras físicas com a extensão diferente de zero entre as regiões sejam consideradas. Se além das fronteiras com extensão diferente de zero puderem ser considerados os vértices como contíguos, a contiguidade é considerada como *rainha* (*queen*) se apenas os vértices forem considerados para definir a contiguidade, a convenção é demoninada *bisbo* (*bishop*) [Almeida 2012].

```
# Plot grid geotable
ax = gdf.plot(facecolor='w', edgecolor='k')

# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )

# Remove axes
ax.set_axis_off()
plt.show()
```

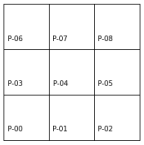
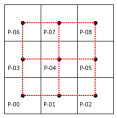


Figura 1.7. Plot da grade 3x3.

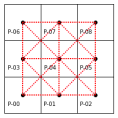
Na Figura 1.8(a) e 1.8(b) temos exemplos das convenções mais utilizadas na literatura, respectivamente, *Rook* e *Queen*.

```
# Build a regular 3x3 lattice and draw it here
wr = weights.contiguity.Rook.from_dataframe(gdf)

# Set up figure
f,ax = plt.subplots(1,1, subplot_kw=dict(aspect='equal'))
# Plot grid
gdf.plot(facecolor='w', edgecolor='k', ax=ax)
# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )
# Plot weights connectivity
wr.plot(gdf, edge_kws=dict(color='r', linestyle=':'), ax =ax)
# Remove axes
ax.set_axis_off()
```



(a)

```
# Build a regular 3x3 lattice and draw it here
wr = weights.contiguity.Queen.from_dataframe(gdf)

# Set up figure
f,ax = plt.subplots(1,1, subplot_kw=dict(aspect='equal'))
# Plot grid
gdf.plot(facecolor='w', edgecolor='k', ax=ax)
# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )
# Plot weights connectivity
wr.plot(gdf, edge_kws=dict(color='r', linestyle=':'), ax =ax)
# Remove axes
ax.set_axis_off()
```



(b)

Figura 1.8. Contiguidade para uma grade 3x3: (a) *rook* e (b) *queen*.

Outra forma de critério de proximidade na definição dos pesos espaciais é a distância geográfica. A ideia por trás é que duas regiões mais próximas geograficamente têm uma maior interação espacial. Uma matriz w muito adotada na literatura é a matriz dos k vizinhos mais próximos (*KNN*). Na Figura 1.9 temos um exemplo com três vizinhos para cada região (parâmetro $k = 3$ na linha 2).

Vamos agora definir matrizes de vizinhança utilizando os critérios *queen*, *rook* e *knn* para os setores censitários urbanos da cidade de São Bernardo do Campo, no interior de São Paulo. Primeiro devemos selecionar os respectivos setores censitários urbanos da cidade (linhas de 1 a 3 da Figura 1.10) e em seguida criar as matrizes de pesos para os respectivos critérios (linhas de 6 a 8 da Figura 1.10). De posse dos setores e das matrizes podemos plotar os mapas. A Figura 1.11 temos o código para gerar os respectivos mapas e na Figura 1.12) temos os mapas dos setores censitários de São Bernado do Campo com os critérios de vizinhança *queen*, *rook* e *knn*, respectivamente.

1.4. Mapas coropléticos

Agora que entendemos os processos geográficos e os dados que os mensuram, apresentaremos a AEDE. A AEDE aumenta a análise exploratória de dados de Tukey [Tukey 1977]

```
# Build a regular 3x3 lattice and draw it here
wr = weights.KNN.from_dataframe(gdf, k=3)

# Set up figure
f,ax = plt.subplots(1,1, subplot_kw=dict(aspect='equal'))
# Plot grid
gdf.plot(facecolor='w', edgecolor='k', ax=ax)
# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )
# Plot weights connectivity
wr.plot(gdf, edge_kws=dict(color='r', linestyle=':'), ax =ax)
# Remove axes
ax.set_axis_off()
```

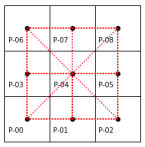


Figura 1.9. Contiguidade *knn* com três vizinhos para uma grade 3x3.

```
sao_bernardo_campo = setores_censitarios_sp.query(
    "NM_MUNICIP == 'SÃO BERNARDO DO CAMPO' and TIPO =='URBANO'"
)
sao_bernardo_campo.reset_index(drop=True, inplace=True)

w_queen = weights.contiguity.Queen.from_dataframe(sao_bernardo_campo)
w_rook = weights.contiguity.Queen.from_dataframe(sao_bernardo_campo)
w_knn = weights.distance.KNN.from_dataframe(sao_bernardo_campo, k=4)
```

Figura 1.10. Código para seleção dos setores censitários urbanos da cidade de São Bernardo do Campo com a definição da matriz de vizinhança *queen*, *rook* e *knn*

e envolve uma grande coleção de técnicas usadas para "orientar-se" (encontrar estrutura) dentro de seu conjunto de dados. Para problemas geográficos, isso geralmente envolve entender se nossos dados exibem um padrão geográfico.

Mapas coropléticos são mapas geográficos que exibem informações estatísticas codificadas em uma paleta de cores. Os mapas coropléticos desempenham um papel proeminente na ciência de dados geográficos, pois nos permitem exibir atributos ou variáveis não geográficas em um mapa geográfico e é a forma usual de apresentação de dados agregados por áreas [Câmara et al. 2004].

A eficácia de um mapa coroplético depende na maioria do propósito do mapa. Qual mensagem você deseja comunicar moldará quais opções são preferíveis em relação a outras. Podemos considerar três dimensões sobre as quais colocar o pensamento intencional valerá a pena. Os mapas coropléticos gira em torno de: primeiro, selecionar um número de grupos menor que n em que todos os valores em nosso conjunto de dados serão mapeados; segundo, identificar um algoritmo de classificação que execute tal mapeamento, seguindo algum princípio alinhado ao nosso interesse; e terceiro, uma vez que sabemos em quantos grupos reduziremos todos os valores em nossos dados, qual cor é atribuída a cada grupo para garantir que ele codifique as informações que queremos refletir. Em termos gerais, o esquema de classificação define o número de classes, bem como as regras de atribuição; enquanto uma boa simbolização transmite informações sobre a diferenciação de valor entre as classes.

```
# Set up figure and axis
f, axs = plt.subplots(1, 3, figsize=(16, 10))

# Queen
ax = axs[0]
sao_bernardo_campo.plot(ax=ax)
w_queen.plot(
    sao_bernardo_campo,
    edge_kws=dict(linewidth=1, color='orangered'),
    node_kws=dict(marker='*'),
    ax=ax
)
ax.set_axis_off()
ax.set_title('Queen')

# Rook
ax = axs[1]
sao_bernardo_campo.plot(ax=ax)
w_rook.plot(
    sao_bernardo_campo,
    edge_kws=dict(linewidth=1, color='orangered'),
    node_kws=dict(marker='*'),
    ax=ax
)
ax.set_axis_off()
ax.set_title('Rook')

# KNN
ax = axs[2]
sao_bernardo_campo.plot(ax=ax)
w_knn.plot(
    sao_bernardo_campo,
    edge_kws=dict(linewidth=1, color='orangered'),
    node_kws=dict(marker='*'),
    ax=ax
)
ax.set_axis_off()
ax.set_title('KNN 4');
```

Figura 1.11. Código para a geração dos mapas com os setores censitários urbanos da cidade de São Bernardo do Campo com os critérios de vizinhança *queen*, *rook* e *knn*

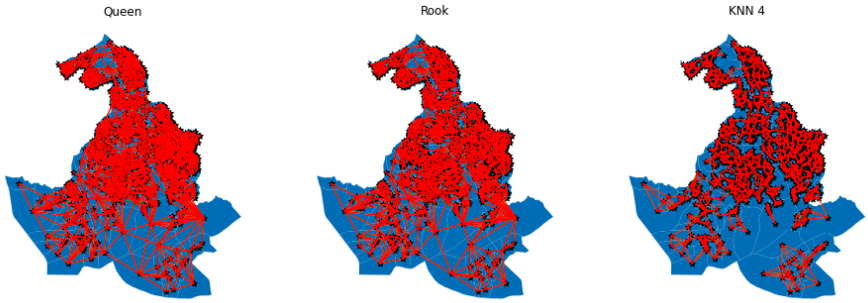


Figura 1.12. Mapas com os setores censitários urbanos da cidade de São Bernardo do Campo com critérios de vizinhança *queen*, *rook* e *knn*

1.4.1. Classificação de dados quantitativos

Selecionar o número de grupos aos quais queremos atribuir os valores em nossos dados e como cada valor é atribuído a um grupo seja um problema de classificação. A classificação dos dados considera o problema de particionar os valores dos atributos em grupos mutuamente exclusivos e exaustivos. A maneira precisa como isso é feito será função da escala de medição do atributo em questão. Para atributos quantitativos (escalas ordinais, intervalares, de razão) as classes terão uma ordenação explícita. Mais formalmente, o problema de classificação é definir limites de classe de tal forma que:

$$c_j < u_i \leq c_{i+1} \forall y_i \in C_j \tag{2}$$

onde y_i é o valor do atributo para localização espacial i, j é um índice de classe, e c_j representa o limite inferior do intervalo j. Diferentes esquemas de classificação são obtidos a partir de sua definição dos limites de classe. A escolha do esquema de classificação

deve levar em consideração a distribuição estatística dos valores dos atributos, bem como o objetivo do nosso mapa (por exemplo, destacar *outliers* versus retratar com precisão a distribuição dos valores).

Existem diversos algoritmos de classificação disponíveis para classificação que seguem critérios que podem ser de interesse em diferentes contextos, pois focam em diferentes prioridades. Abaixo, vamos nos concentrar em alguns deles. Para calcular a classificação, vamos basear no `mapclassify` da família *Pysal*.

Para testar os diferentes algoritmos de classificação disponibilizados no `mapclassify` vamos criar um `GeoDataFrame` com as malhas dos municípios brasileiros disponibilizadas pelos IBGE, com os casos confirmados de covid-19 disponibilizados em `https://brasil.io/dataset/covid19/caso_full/`. O código com a criação desse `GeoDataFrame` pode ser observado na Figura 1.13.

```
municipios_br = gpd.read_file('BR_Municipios_2021.zip')
municipios_br['CD_MUN'] = municipios_br['CD_MUN'].astype(int)

casos_confirmados_covid_br = pd.merge(
    casos_confirmados_covid,
    municipios_br,
    left_on='city_ibge_code',
    right_on='CD_MUN',
)

casos_confirmados_covid_br = gpd.GeoDataFrame(
    casos_confirmados_covid_br,
    crs = municipios_br.crs
)

casos_confirmados_covid_br.head()
```

	city_ibge_code	new_confirmed	CD_MUN	NM_MUN	SIGLA	AREA_KM2	geometry
0	1100015	4089	1100015	Alta Floresta D'Oeste	RO	7067.127	POLYGON ((-62.19465 -11.82746, -62.19332 -11.8...
1	1100023	22657	1100023	Ariquemes	RO	4426.571	POLYGON ((-62.53648 -9.73222, -62.52765 -9.736...
2	1100031	918	1100031	Cabixi	RO	1314.352	POLYGON ((-60.37119 -13.36655, -60.37134 -13.3...
3	1100049	14904	1100049	Cacoal	RO	3793.000	POLYGON ((-61.00080 -11.29737, -61.00103 -11.3...
4	1100056	2570	1100056	Cerejeiras	RO	2783.300	POLYGON ((-61.49976 -13.00525, -61.49809 -13.0...

Figura 1.13. Criação do GeoDataFrame com os casos de covid-19 agregados pela malha dos municípios brasileiros.

1.4.1.1. Intervalos iguais

A abordagem de Freedman-Diaconis [Freedman and Diaconis 1981] fornece uma regra para determinar a largura e, por sua vez, o número de caixas para a classificação. Este é um caso especial de um classificador mais geral conhecido como "intervalos iguais", onde cada uma das caixas tem a mesma largura no espaço de valor. Para um determinado valor de k, a classificação de intervalos iguais divide o intervalo do espaço de atributo em k intervalos de comprimento igual, com cada intervalo tendo uma largura $w = \frac{x_0 - x_{n-1}}{k}$. Assim, a classe máxima é $(x_{n-1} - w, x_{n-1}]$ e a primeira classe é $(-\infty, x_{n-1} - (k-1)w]$.

Intervalos iguais têm as vantagens duplas de simplicidade e facilidade de interpretação. No entanto, essa regra considera apenas os valores extremos da distribuição e, em alguns casos, isso pode resultar em uma ou mais classes esparsas.

Este é claramente o caso em nosso conjunto de dados de covid-19, criado na Figura 1.13, pois a maioria dos valores é colocada na primeira classe, deixando as demais classes bastante esparsas (Figura 1.14).

```
ei5 = mapclassify.EqualInterval(casos_confirmados_covid_br['new_confirmed'], k=5)
ei5
```

```
EqualInterval

       Interval          Count
------------------------------
[    10.00, 195250.80] |  5560
(195250.80, 390491.60] |     7
(390491.60, 585732.40] |     1
(585732.40, 780973.20] |     1
(780973.20, 976214.00] |     1
```

Figura 1.14. Distribuição dos casos de covid-19 pela abordagem de intervalos iguais.

Observe na Figura 1.14 que cada um dos intervalos tem igual largura de $w = 195240,8$. Deve-se notar também que a primeira classe é fechada no limite inferior, em contraste com a abordagem geral definida acima.

1.4.1.2. Quantis

Para evitar o problema potencial de classes esparsas, os *quantis* da distribuição podem ser usados para identificar os limites das classes. De fato, cada classe terá aproximadamente $\left\lfloor \frac{n}{k} \right\rfloor$ observações usando o classificador de quantil. Se $k = 5$ os quintis da amostra são usados para definir os limites superiores de cada classe, como pode ser observado na Figura 1.15.

```
q5 = mapclassify.Quantiles(casos_confirmados_covid_br['new_confirmed'], k=5)
q5
```

```
Quantiles

       Interval          Count
------------------------------
[    10.00,    440.00] |  1118
(   440.00,    809.00] |  1113
(   809.00,   1487.80] |  1111
(  1487.80,   3379.00] |  1115
(  3379.00, 976214.00] |  1113
```

Figura 1.15. Distribuição dos casos de covid-19 pela abordagem de *quantis*.

Observe que na Figura 1.15, embora os números de valores em cada classe sejam aproximadamente iguais, as larguras dos quatro primeiros intervalos são bastante diferentes. Embora os *quantis* evitem a armadilha das classes esparsas, essa classificação não está livre de problemas. As larguras variáveis dos intervalos podem ser marcadamente diferentes, o que pode levar a problemas de interpretação. Um segundo desafio enfrentado pelos *quantis* surge quando há um grande número de valores duplicados na distribuição, de modo que os limites para uma ou mais classes se tornam ambíguos. Por exemplo, se alguém tivesse uma variável com $n = 20$ mas 10 das observações assumiram o mesmo valor que foi o mínimo observado, então para valores de $k > 2$, os limites de classe tornam-se mal definidos, pois uma simples regra de divisão n/k no valor observado classificado dependeria de como os empates são tratados na classificação.

Geraremos uma variável sintética com essas características (Figura 1.16).

E agora executaremos a classificação com a abordagem de *quantil* (Figura 1.15).

Nesse caso, é formado uma classificação com um número menor de classes usando pseudo quantis, ou quantis definidos nos valores únicos da amostra.

```
# Set seed for reproducibility
np.random.seed(12345)
# Generate a variable of 20 values randomly
# selected from 0 to 10
x = np.random.randint(0,10,20)
# Manually ensure the first ten values are 0 (the
# minimum value)
x[0:10] = x.min()
x
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 7, 6, 0, 2, 9, 1, 2, 6, 7])
```

Figura 1.16. Código com a geração de um conjunto com valores repetidos.

```
ties = mapclassify.Quantiles(x, k=5)
ties
```

```
Quantiles

  Interval     Count
--------------------
[0.00, 0.00] |    11
(0.00, 1.40] |     1
(1.40, 6.20] |     4
(6.20, 9.00] |     4
```

Figura 1.17. Distribuição pela abordagem de *quantis* para o conjunto com valores repetidos.

1.4.1.3. Jenks Caspall

Originalmente proposto por [Jenks and Caspall 1971], aborda o desafio da classificação de uma perspectiva heurística, e não determinística, visando minimizar a soma dos desvios absolutos em torno das médias das classes. A abordagem começa com um número pré-especificado de classes e um conjunto inicial arbitrário de quebras de classe - por exemplo, usando *quintis*. O algoritmo tenta melhorar a função objetivo considerando o movimento de observações entre classes adjacentes. Por exemplo, o maior valor no *quintil* mais baixo seria considerado para o movimento para o segundo quintil, enquanto o valor mais baixo no segundo *quintil* seria considerado para um possível movimento para o primeiro *quintil*. O movimento candidato que resultar na maior redução na função objetivo seria feito, e o processo continua até que nenhum outro movimento de melhoria seja possível. O algoritmo Jenks Caspall é o caso unidimensional do algoritmo *K-Means* amplamente utilizado para agrupamento. Na Figura 1.18 temos a aplicação dessa abordagem para o conjunto de dados de casos de covid-19.

```
jc5 = mapclassify.JenksCaspall(casos_confirmados_covid_br['new_confirmed'], k=5)
jc5
```

```
JenksCaspall

       Interval          Count
------------------------------
[    10.00,   1675.00] |  3517
(  1675.00,   6209.00] |  1417
(  6209.00,  22031.00] |   470
( 22031.00,  80865.00] |   135
( 80865.00, 976214.00] |    31
```

Figura 1.18. Distribuição dos casos de covid-19 pela abordagem Jenks Caspall.

1.4.1.4. Fisher Jenks

O segundo algoritmo ótimo adota uma abordagem de programação dinâmica para minimizar a soma dos desvios absolutos em torno das medianas de classe. Em contraste com

a abordagem Jenks-Caspall, o Fisher-Jenks garante a produção de uma classificação ideal para um número pré-especificado de classes. Na Figura 1.19 temos a aplicação dessa abordagem para o conjunto de dados de casos de covid-19.

```
fj5 = mapclassify.FisherJenks(casos_confirmados_covid_br['new_confirmed'], k=5)
fj5
```

```
FisherJenks

      Interval          Count
------------------------------
[    10.00,  21053.00] |  5401
( 21053.00,  98093.00] |   145
( 98093.00, 278370.00] |    21
(278370.00, 597871.00] |     2
(597871.00, 976214.00] |     1
```

Figura 1.19. Distribuição dos casos de covid-19 pela abordagem Fisher Jenks.

1.4.1.5. Comparando esquemas de classificação

Como um caso especial de agrupamento, a definição do número de classes e os limites de classe representam um problema para o projetista do mapa. Para a classificação de mapas, um critério de otimização comum é uma medida de ajuste. No `mapclassify`, o "desvio absoluto em torno das medianas de classe" (do inglês *absolute deviation around class medians* - ADCM) é calculado e fornece uma medida de ajuste que permite a comparação de classificadores alternativos para o mesmo valor de k. O ADCM nos dará uma noção de quão "compacto" é cada grupo. Para ver isso, podemos comparar diferentes classificadores para k sobre os dados de casos de covid-19 (Figura 1.20.

```
# Bunch classifier objects
class5 = ei5, q5, msd, bp, jc5, fj5
# Collect ADCM for each classifier
fits = np.array([ c.adcm for c in class5])
# Convert ADCM scores to a DataFrame
adcms = pd.DataFrame(fits)
# Add classifier names
adcms['classifier'] = [c.name for c in class5]
# Add column names to the ADCM
adcms.columns = ['ADCM', 'Classifier']
ax = sns.barplot(
    y='Classifier', x='ADCM', data=adcms, palette='Pastel1'
)
```

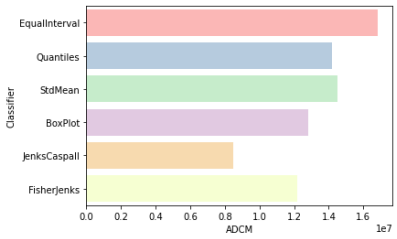


Figura 1.20. ADC dos classificadores usados na base de dados de casos de covid-19.

É possível observar na Figura 1.20 que o classificador Jenks-Caspall domina todos os outros classificadores para $k = 5$ com um ADCM de $0.848 \times e^7$ (lembre-se, quanto menor, melhor).

1.5. Autocorrelação Espacial Global

A noção de autocorrelação espacial refere-se à existência de uma "relação funcional entre o que acontece em um ponto do espaço e o que acontece em outro" [Anselin 1988].

A autocorrelação espacial, portanto, tem a ver com o grau onde a similaridade de valores entre observações em um conjunto de dados está relacionada à similaridade nas localizações de tais observações. Isso é semelhante à ideia tradicional de correlação entre duas variáveis, que nos informa sobre como os valores de uma variável mudam em função dos da outra, embora com algumas diferenças. De maneira semelhante, a autocorrelação espacial também está relacionada (mas distinta) à contraparte temporal, a autocorrelação temporal, que relaciona o valor de uma variável em um determinado momento com os de períodos anteriores. Em contraste com essas outras ideias de correlação, a autocorrelação espacial relaciona o valor da variável de interesse em um determinado local, com valores da mesma variável em outros locais. Uma forma alternativa de entender o conceito é como o grau de informação contido no valor de uma variável em um determinado local sobre o valor dessa mesma variável em outros locais [Almeida 2012, Andrade et al. 2007, Anselin et al. 2013, Anselin 2005].

Para entender melhor a noção de autocorrelação espacial, é útil começar considerando como é o mundo na sua ausência. Uma ideia chave neste contexto é a da aleatoriedade espacial: uma situação onde a localização de uma observação não fornece nenhuma informação sobre o seu valor. Em outras palavras, uma variável é espacialmente aleatória se sua distribuição não segue um padrão espacial discernível. A autocorrelação espacial pode assim ser definida como a "ausência de aleatoriedade espacial" [Almeida 2012, Andrade et al. 2007, Anselin et al. 2013, Anselin 2005].

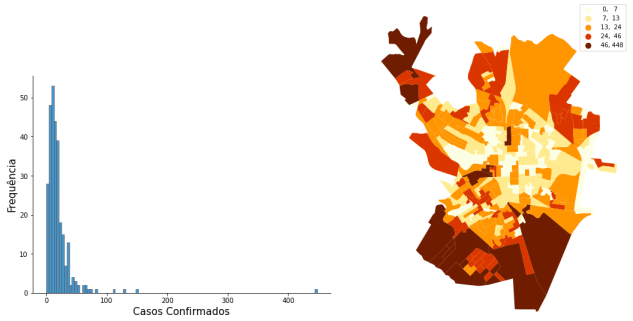
Para ilustrar a noção de autocorrelação espacial e suas diferentes variantes, utilizaremos as notificações confirmadas de casos de dengue do município de São Carlos, localizado na região central do estado de São Paulo, no ano de 2019, disponibilizados pela vigilância epidemiológica do município. Antes, importaremos todas as bibliotecas relevantes que usaremos ao longo desta seção (Figura 1.21).

```
# Graphics
import matplotlib.pyplot as plt
import seaborn
from pysal.viz import splot
from splot.esda import plot_moran
import contextily
# Analysis
import geopandas
import pandas
from pysal.explore import esda
from pysal.lib import weights
from numpy.random import seed
```

Figura 1.21. Importação das bibliotecas necessárias para análise da correlação espacial.

Para gerar essa base de dados de casos de dengue, foram considerados todos os casos confirmados de dengue notificados no Sistema de Informação de Agravos de Notificação - Sinan Dengue/Chikungunya dos residentes da área urbana do município de São Carlos-SP no período de 1 de janeiro à 31 de dezembro do ano de 2019. Essas notificações foram georreferenciadas e agregadas aos setores censitários por meio de uma função de *join spatial*. Na Figura 1.22(a) temos o histograma com a frequência de casos confirmados de dengue nos setores censitários e na Figura 1.22(b) é apresentado distribuição espacial dos casos confirmados de dengue, classificados segundo o algoritmo de *Jenks Caspall*.

Antes de nos aprofundarmos na autocorrelação, precisamos gerar a matriz de pesos espaciais. Usaremos oito vizinhos mais próximo, vale ressaltar que outros critérios



(a) (b)

Figura 1.22. (a) Histograma com a frequência de casos confirmados e (b) Representação coroplética da distribuição espacial dos casos confirmados de dengue.

também podem ser utilizados. Na Figura 1.23 temos o código necessário para gerar a nossa matriz de pesos espacial.

```
# Generate W from the GeoDataFrame
w = weights.KNN.from_dataframe(dengue_sc, k=8)
# Row-standardization
w.transform = 'R'
```

Figura 1.23. Código necessário para gerar a nossa matriz de pesos espacial.

Ao analisar um mapa como o da Figura 1.22(b), estamos buscando padrões do ponto de vista espacial e um dos aspectos necessários que temos que responder, como já mencionado, é verificar se o evento em estudo e os fatores relacionados a ele possuem distribuição espacialmente condicionada (dependência espacial), ou seja, se os valores do atributo em estudo em uma determinada região depende ou não dos valores deste atributo nas regiões vizinhas, tarefa extremamente difícil de se realizar visualmente [Almeida 2012].

1.5.1. Estatística *I* de Moran

[Moran 1948] propôs a elaboração de um coeficiente de autocorrelação espacial, usando a média de autocovariância na forma de produto cruzado. Assim surgia o primeiro coeficiente de autocorrelação espacial, denominado de *I* de Moran, definido pela Equação 3.

$$I = \frac{n}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij} z_i z_j}{\sum_i z_i^2} \tag{3}$$

em que n é o número de observações, ${}_z i$ é o valor padronizado da variável de interesse na região i, e ${}_{wij}$ é a célula correspondente à i-ésima linha da j-ésima coluna da matriz de pesos espaciais.

A estatística *I* de Moran é a estatística mais difundida e constitui em uma espécie de coeficiente de autocorrelação, ou seja, é a relação da autocovariância do tipo produto cruzado pela variância dos dados [Andrade et al. 2007, Almeida 2012].

Uma abordagem alternativa para entender a intuição por trás de sua matemática, é através de uma interpretação gráfica e essa interpretação gráfica pode ser efetu-

ada através do diagrama de dispersão de Moran, também conhecido como Moran Plot. O Moran Plot é uma maneira de visualizar um conjunto de dados espaciais para explorar a natureza e a força da autocorrelação espacial. É essencialmente um gráfico de dispersão tradicional em que a variável de interesse é exibida em relação à defasagem espacial da variável de interesse. Para poder interpretar valores como acima ou abaixo da média, a variável de interesse geralmente é padronizada subtraindo sua média [Andrade et al. 2007, Almeida 2012].

A Figura 1.24 mostra a relação entre a porcentagem padronizada casos confirmados de dengue e sua defasagem espacial que, devido à padronização por linha de w, pode ser interpretado como a densidade padronizada média dos casos confirmados de dengue na vizinhança de cada observação. A fim de orientar a interpretação do gráfico, um ajuste linear também é incluído. Essa linha representa o melhor ajuste linear ao gráfico de dispersão ou, em outras palavras, qual é a melhor forma de representar a relação entre às duas variáveis como uma linha reta.

```
# spacial lag
dengue_sc['dengue-2019_lag'] = weights.spatial_lag.lag_spatial(
    w, dengue_sc['dengue-2019']
)

dengue_sc['dengue-2019_std'] = ( dengue_sc['dengue-2019'] - dengue_sc['dengue-2019'].mean() )
dengue_sc['dengue-2019_lag_std'] = ( dengue_sc['dengue-2019_lag'] - dengue_sc['dengue-2019_lag'].mean() )

# Moran plot
f, ax = plt.subplots(1, figsize=(6, 6))
sns.regplot(
    x='dengue-2019_std',
    y='dengue-2019_lag_std',
    ci=None,
    data=dengue_sc,
    line_kws={'color':'r'}
)
ax.axvline(0, c='k', alpha=0.5)
ax.axhline(0, c='k', alpha=0.5)
ax.set_title('Moran Plot - % Leave')
plt.show()
```

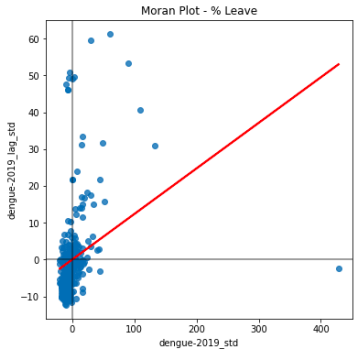


Figura 1.24. Gráfico de dispersão de Moran.

O gráfico da Figura 1.24 mostra uma relação positiva entre ambas as variáveis. Isso indica a presença de autocorrelação espacial positiva: valores semelhantes tendem a se localizar próximos uns dos outros. Isso significa que a tendência geral é que os valores altos estejam próximos de outros valores altos e que os valores baixos sejam cercados por outros valores baixos. Isso, no entanto, não significa que este seja o único caso no conjunto de dados: é claro que pode haver situações particulares em que valores altos são cercados por valores baixos e vice-versa. Mas isso significa que, se tivéssemos que resumir o padrão principal dos dados em termos de quão agrupados são os valores semelhantes, a melhor maneira seria dizer que eles são positivamente correlacionados, portanto, agrupados no espaço.

Para calcular a estatística *I* de Moran em nosso conjunto de dados, podemos chamar uma função específica do pacote `esda` diretamente. Na Figura 1.25 temos o cálculo da estatística *I* de Moran através do pacote `esda` com seu respectivo valor.

```
1 moran_dengue_2019 = esda.moran.Moran(dengue_sc['dengue-2019'], w)
2 moran_dengue_2019.I
```

```
0.12375745247599824
```

Figura 1.25. Cálculo da estatística *I* de Moran e seu respectivo valor.

A outra informação que pode ser extraída da estatística *I* de Moran está relacionada à inferência estatística, ou seja, qual a probabilidade do padrão que observamos no mapa e que a estatística *I* de Moran captura em seu valor ser gerado por um processo inteiramente aleatório? Se considerássemos a mesma variável, mas embaralhemos suas localizações aleatoriamente, obteríamos um mapa com características semelhantes? Para obter *insights* sobre essas questões, o pacote `esda` realiza uma simulação e retorna uma medida de certeza sobre a probabilidade de obter um padrão como o que observamos em um processo espacialmente aleatório. Isso está resumido no atributo `p_sim` (Figura 1.26).

```
1 moran_dengue_2019.p_sim
```

```
0.001
```

Figura 1.26. Valor do *p_sim* para os casos confirmados de dengue.

O valor é calculado como um $p-valor$ empírico que representa a proporção de realizações na simulação sob aleatoriedade espacial que são mais extremas do que o valor observado. Um valor de p suficientemente pequeno associado à estatística *I* de Moran de um mapa permite rejeitar a hipótese de que o mapa é aleatório. Em outras palavras, podemos concluir que o mapa apresenta mais padrão espacial do que esperávamos se os valores tivessem sido alocados aleatoriamente para um local.

O `pysal` disponibiliza um módulo de visualização que combina o Moran Plot (direita) com um gráfico do teste empírico que realizamos para obter `p_sim` (esquerda), chamado de `plot_moran` e sua utilização pode ser observado na Figura 1.27.

1.5.2. Outros índices globais

A estatística *I* de Moran é provavelmente a estatística mais utilizada para autocorrelação espacial global, porém não é a única. Apresentamos duas medidas adicionais comuns no trabalho aplicado. Embora todos considerem a autocorrelação espacial, diferem na forma como o conceito é abordado na especificação de cada teste.

Índice *C* de Geary

A razão de contiguidade, proposto por [Geary 1954], é dado pela Equação 4.

$$C = \frac{(n-1)}{2\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(y_i - y_j)^2}{\sum_i (y_i - \bar{y})^2} \tag{4}$$

```
plot_moran(
    moran_dengue_2019,
    zstandard=True,
    figsize=(10,5)
);
```

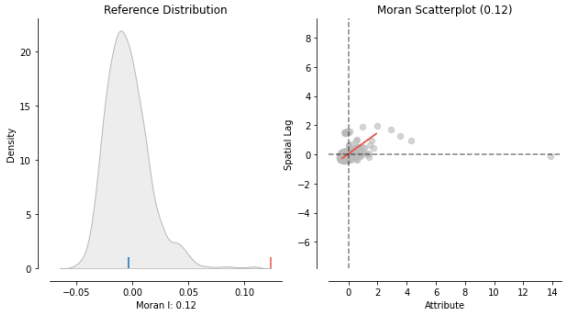


Figura 1.27. *plot_moran* módulo de visualização disponibilizado pelo *pysal*.

onde n é o número de observações, w_{ij} é a célula em uma matriz binária W expressando se i e j são vizinhos ($w_{ij} = 1$) ou não ($w_{ij} = 0$), y_i é a i-ésima observação da variável de interesse, e $\bar{y}$ é sua média amostral. Quando comparado à estatística I de Moran, fica evidente que ambas as medidas comparam a relação de Y dentro da vizinhança local de cada observação para aquela em toda a amostra. No entanto, também existem diferenças sutis. Enquanto a estatística I de Moran usa produtos cruzados nos valores padronizados, o índice C de Geary usa diferenças nos valores sem qualquer padronização.

Computacionalmente, o índice C de Geary é mais exigente, mas pode ser facilmente calculado usando o pacote `esda` do `pysal`, a Figura 1.28 mostra a sua utilização.

```
geary = esda.geary.Geary(dengue_sc['dengue-2019'], w)
geary.C
```

```
0.7905555768777744
```

Figura 1.28. Determinação do índice *C* de Geary através do pacote *esda* do *pysal*.

A inferência é realizada semelhantemente estatística I de Moran. Podemos realizar uma simulação que nos permite traçar uma distribuição empírica da estatística sob a nulidade de aleatoriedade espacial e depois compará-la com a estatística obtida ao usar a distribuição geográfica observada de os dados. Para acessar o pseudo valor-p, calculado como no caso de Moran, podemos chamar `p_sim` (Figura 1.29).

```
geary.p_sim
```

```
0.035
```

Figura 1.29. Valor do *p_sim* do índice *C* de Geary para os casos confirmados de dengue.

Nesse caso, o índice C de Geary aponta na mesma direção que a estatística I de Moran, ou seja, há uma clara indicação de que a estatística que calculamos no conjunto de dados observado difere que seria esperado em uma situação de pura aleatoriedade espacial. Assim, a partir desta análise, podemos concluir também que a autocorrelação espacial está presente.

Índice *G* de Getis e Ord

Originalmente proposto por [Getis and Ord 2010], o índice G é uma medida de autocorrelação espacial de natureza global. Na primeira versão dessa estatística, ela é computada para apenas valores positivos de uma variável por definir um conjunto de vizinhos para cada região como aquelas observações que fiquem dentre de uma distância de corte (*cutt-off*) fiza (d) da região [Almeida 2012]. O índice G de Getis e Ord é definida pela Equação 5.

$$G(d) = \frac{\sum_i \sum_j w_{ij}(d) y_i y_j}{\sum_i \sum_j y_i y_j} \tag{5}$$

onde w_{ij} é o peso binário atribuído na relação entre as observações i e j seguindo um critério de distância.

Para ilustrar seu cálculo, vamos calcular uma matriz de distância binária W. Para garantir que cada observação tenha pelo menos um vizinho, usaremos o método *min_threshold_distance* e projetaremos o conjunto de dados no *Ordnance Survey* CRS (código EPSG 27700), expresso em metros, esse procedimento pode ser observado na Figura 1.30.

```
1 dengue_sc_osgb = dengue_sc.to_crs(epsg=27700)
2 pts = dengue_sc_osgb.centroid
3 xys = pd.DataFrame({'X': pts.x, 'Y': pts.y})
4 min_thr = weights.util.min_threshold_distance(xys)
5 min_thr
```

```
2143.9486593791926
```

Figura 1.30. Determinação da matriz distância binária W.

Para que cada setor censitário tenha um vizinho, a faixa de distância deve ser de, pelo menos, cerca de 2143 metros. Essas informações podem ser passadas para no construtor `DistanceBand` (linha 1 da Figura 1.31).

```
1 w_dengue_sc = weights.DistanceBand.from_dataframe(dengue_sc_osgb, min_thr)
2 gao = esda.getisord.G(dengue_sc['dengue-2019'], w_dengue_sc)
3 print("Getis & Ord G: %.3f | Pseudo P-value: %.3f"%(gao.G, gao.p_sim))
```

```
Getis & Ord G: 0.123 | Pseudo P-value: 0.088
```

Figura 1.31. Determinando a distância miníma para os setores censitários e o índice *G* de Getis e Ord.

O acesso à estatística (`gao.G`) e atributos adicionais podem ser obtidos da mesma forma que com as estatísticas anteriores.

Da mesma forma, a inferência também pode ser realizada por meio de simulações computacionais que replicam várias instâncias de aleatoriedade espacial usando os valores da variável de interesse, mas embaralhando suas localizações. Neste caso, o pseudo valor p calculado sugere um claro afastamento da hipótese de não concentração.

1.6. Autocorrelação Espacial Local

As estatísticas globais de autocorrelação espaciais, vista na Seção anterior, fornecem padrões de associação linear espacial, ou seja, o grau em que o conjunto dos dados está agrupado, disperso ou distribuído aleatoriamente [Almeida 2012, Fotheringham et al. 2000].

A presença de autocorrelação espacial tem implicações importantes para análises estatísticas subsequentes. De uma perspectiva substantiva, a autocorrelação espacial poderia refletir a operação de processos que geram associação entre os valores em localidades próximas. Isso pode representar transbordamentos, onde os resultados em um local influenciam outros locais ou pode indicar contágio, onde os resultados em um local influenciam causalmente outros locais.

Apesar de sua importância, as medidas globais de autocorrelação espacial são estatísticas de "mapa inteiro". Eles fornecem um único resumo para um conjunto de dados inteiro, ou seja, as medidas pode nos dizer se os valores em nosso mapa se agrupam (ou se dispersam) em geral, mas não nos informará sobre onde estão os agrupamentos específicos (ou valores discrepantes) [Almeida 2012, Fotheringham et al. 2000].

1.6.1. *I* de Moran local

Proposto por [Anselin 1995], os Indicadores Locais de Associação Espacial (do inglês, *Local Indicators of Spatial Association* - LISA), visa identificar casos em que o valor de uma observação e a média de seus arredores são mais semelhantes (alto-alto, do inglês *high-high* - HH ou baixo-baixo, do inglês *low-low* - LL no gráfico de dispersão de Moran) ou diferentes (alto-baixo, do inglês *high-low* - HL ou baixo-alto, do inglês *low-high* - LH) do que esperaríamos do puro acaso. O mecanismo para fazer isso é semelhante ao do índice *I* de Moran global, mas aplicado neste caso a cada observação. Isso resulta em tantas estatísticas quanto as observações originais. A representação formal da estatística pode ser escrita pela Equação 6.

$$I_i = \frac{z_i}{m_2} \sum_j w_{ij} z_j ; m_2 = \frac{\sum_i z_i^2}{n} \tag{6}$$

onde m_2 é o segundo momento (variância) da distribuição de valores nos dados, $z_i = u_i - \bar{y}, w_{ij}$, é o peso espacial para o par de observações i e j, e n é o número de observações.

LISA é amplamente utilizada em muitos campos para identificar agrupamentos geográficos de valores ou encontrar discrepâncias geográficas. É uma ferramenta útil que pode retornar rapidamente as áreas em que os valores estão concentrados e fornecer evidências sugestivas sobre os processos que podem estar em ação. Por esses motivos, eles têm um lugar privilegiado na caixa de ferramentas da ciência de dados geográficos. Entre muitas outras aplicações, LISA têm sido usadas para identificar aglomerados geográficos de pobreza [Dawson et al. 2018], delinear áreas de atividade econômica particularmente alta/baixa [Torres-Preciado et al. 2014] ou identificar aglomerados de doenças contagiosas [Zhang et al. 2020]. É possível determinar os valores de LISA através da chamada da função `Moran_Local` do `esda`, conforme mostra na linha 1 da Figura 1.32. Precisamos passar a variável de interesse - casos confirmados de dengue da cidade de São Carlos-SP e os pesos espaciais que descrevem as relações de vizinhança entre as diferentes áreas que compõem o conjunto de dados. Isso cria um objeto LISA (lisa) que possui vários atributos de interesse. Os próprios indicadores locais estão no atributo `Is`, aonde se pode ter uma noção de sua distribuição.

O objeto LISA (linha 1 da Figura 1.32) contém as estatísticas *I* de Moran para cada setor censitário, com seus respectivos níveis de significância e essa copiosa quantidade

```
lisa = esda.moran.Moran_Local(dengue_sc['dengue-2019'], w)

# Draw KDE line
ax = sns.kdeplot(lisa.Is)
# Add one small bar (rug) for each observation
# along horizontal axis
sns.rugplot(lisa.Is, ax=ax);
```

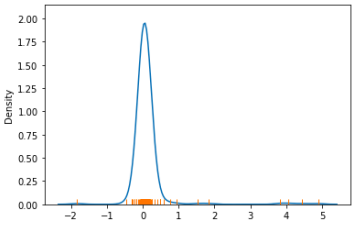


Figura 1.32. Determinação dos índices LISA para os casos confirmados de dengue da cidade de São Carlos-SP.

de informações pode confundir o pesquisador, se colocada em tabelas [Almeida 2012]. Uma forma mais eficiente de apresentar este conjunto de estatísticas é mapeá-las. Na Figura 1.33, o mapa de significância LISA exibe as regiões com estatística I local de Moran significativos para os setores censitários de casos confirmados de dengue para a cidade de São Carlos-SP.

```
dengue_sc['LISA'] = 'não significante'
dengue_sc.loc[lisa.p_sim < 0.05, 'LISA'] = 'p = 0.05'
dengue_sc.loc[lisa.p_sim < 0.01, 'LISA'] = 'p = 0.01'

ax = dengue_sc.plot(
    figsize=(8,8),
    column='LISA',
    legend=True,
    edgecolor='black',
    cmap='gray_r',
)
ax.set_axis_off();
```

Figura 1.33. Mapa de significância LISA para os setores censitários com casos confirmados de dengue da cidade de São Carlos-SP.

O mapa de *clusters* LISA combina a informação do diagrama de dispersão de Moran e a informação do mapa de significância das médias de associação local I_i. O *Pysal* disponibiliza a função `lisa_cluster` do módulo `splot.esda` com essa finalidade, agrupando os *clusters* em HH, HL, LH, LL e ns (não significante). A Figura 1.34 apresenta os *clusters* que passaram no teste de significância estatística ndo I de Moran local.

```
lisa_cluster(
    lisa,
    dengue_sc,
    p=0.05,
    figsize=(6,6),
);
```

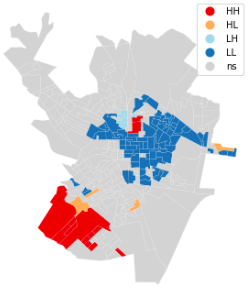


Figura 1.34. Mapa de *clusters* LISA para os setores censitários com casos confirmados de dengue da cidade de São Carlos-SP.

1.6.2. Estatística local de Getis e Ord

Semelhante ao caso global, há mais indicadores locais de correlação espacial do que o índice I de Moran local. O *pysal* inclui estatística local de Getis e Ord do tipo G_i. Estes são um tipo diferente de estatística local que são comumente usados em duas formas: a estatística G_i, que omite o valor em um site em seu resumo local, e a estatística G_i^*, que inclui o próprio valor do site no resumo local. A forma de calculá-los também segue padrões semelhantes às estatísticas do Moran Local.

1.7. Considerações finais

Este capítulo mostrou que as técnicas de análise exploratória de dados espaciais podem ampliar consideravelmente a capacidade de compreender os padrões espaciais associados a dados de área que apresentam autocorrelação espacial global e local. Técnicas exploratórias como os indicadores de Moran e os gráficos de dispersão de Moran são muito úteis para mostrar as agregações espaciais e indicar áreas prioritárias do fenômeno em estudo.

A Seção 1.2, foi apresentado alguns conceitos e características sobre dados espaciais e como manipulá-los através da linguagem Python. Já a Seção 1.3 apresentou os conceitos-chave sobre uma matriz de ponderação espacial, uma matriz quadrada de dimensão n por n que representa o grau de conexão entre as regiões segundo algum critério de proximidade, destacando a influência da região j sobre a região i. A Seção 1.4 apresentou alguns aspectos relacionados aos mapas coropléticos, a forma usual de apresentação de dados agregados por áreas. Na Seção 1.5 apresentamos as estatísticas globais de auto correlação espacial, técnicas utilizadas na tarefa de descobrir se os dados são aleatoriamente distribuídos através do espaço, isto é, se os dados estão correlacionados espacialmente. Por fim, a Seção 1.6 as estatísticas utilizadas para a identificação de padrões locais de autocorrelação espacial.

Disponibilidade de dados e código

O código-fonte utilizado no texto do capítulo, bem como o conjunto de dados estão disponíveis no repositório `shorturl.at/kY129`.

Agradecimentos

Este trabalho foi realizado com apoio financeiro da CAPES (PROEX), CNPQ (processo 312605/2018-8), C4AI, FAPESP (processo 2020/16578-5). Agradecemos também ao ICMC-USP e ao LCR por oferecerem a infraestrutura necessária para o desenvolvimento deste trabalho.

Referências

[Almeida 2012] Almeida, E. (2012). Econometria espacial. *Campinas–SP. Alínea*.

[Andrade et al. 2007] Andrade, A. L., Monteiro, A. M. V., Barcellos, C., Lisboa, E., Acosta, L. M. W., Almeida, M. C. d. M., Brito, M. R. V., Carvalho, M. S., Santos, M. A. d., Cruz, O., et al. (2007). Introdução à estatística espacial para a saúde pública.

[Anselin 1988] Anselin, L. (1988). *Spatial econometrics: methods and models*, volume 4. Springer Science & Business Media.

[Anselin 1995] Anselin, L. (1995). Local indicators of spatial association—lisa. *Geographical analysis*, 27(2):93–115.

[Anselin 2005] Anselin, L. (2005). Exploring spatial data with geodatm: a workbook. *Center for spatially integrated social science*, pages 165–223.

[Anselin et al. 2013] Anselin, L., Florax, R., and Rey, S. J. (2013). *Advances in spatial econometrics: methodology, tools and applications*. Springer Science & Business Media.

[Câmara et al. 2004] Câmara, G., Monteiro, A. M., Fucks, S. D., and Carvalho, M. S. (2004). Análise espacial e geoprocessamento. *Análise espacial de dados geográficos. Brasília: EMBRAPA*, pages 21–54.

[Dawson et al. 2018] Dawson, T., Sandoval, J. O., Sagan, V., and Crawford, T. (2018). A spatial analysis of the relationship between vegetation and poverty. *ISPRS International Journal of Geo-Information*, 7(3):83.

[Domingues et al. 2020] Domingues, A., Silva, F., Santos, L., Souza, R., Coimbra, G., and Loureiro, A. A. F. (2020). Dados geoespaciais: Conceitos e técnicas para coleta, armazenamento, tratamento e visualização. *Sociedade Brasileira de Computação*.

[Fotheringham et al. 2000] Fotheringham, A. S., Brunsdon, C., and Charlton, M. (2000). *Quantitative geography: perspectives on spatial data analysis*. Sage.

[Freedman and Diaconis 1981] Freedman, D. and Diaconis, P. (1981). On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476.

[Geary 1954] Geary, R. C. (1954). The contiguity ratio and statistical mapping. *The incorporated statistician*, 5(3):115–146.

[Getis and Ord 2010] Getis, A. and Ord, J. K. (2010). The analysis of spatial association by use of distance statistics. In *Perspectives on spatial data analysis*, pages 127–145. Springer.

[Haining et al. 1998] Haining, R., Wise, S., and Ma, J. (1998). Exploratory spatial data analysis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(3):457–469.

[Jenks and Caspall 1971] Jenks, G. F. and Caspall, F. C. (1971). Error on choroplethic maps: definition, measurement, reduction. *Annals of the Association of American Geographers*, 61(2):217–244.

[Kluyver et al. 2016] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016). *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016.

[Lopes et al. 2019] Lopes, G. R., Almeida, A. W. S., Delbem, A., and Toledo, C. F. M. (2019). Introdução à análise exploratória de dados com python. *Minicursos ERCAS ENUCMPI*, 2019:160–176.

[Lopes et al. 2021] Lopes, G. R., Delbem, A. C., and de Sousa, J. B. (2021). Introdução à análise de dados geoespaciais com python. *Sociedade Brasileira de Computação*.

[Monteiro et al. 2004] Monteiro, A. M. V., Câmara, G., Carvalho, M., and Druck, S. (2004). Análise espacial de dados geográficos. *Brasília: Embrapa*.

[Moran 1948] Moran, P. A. (1948). The interpretation of statistical maps. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):243–251.

[Pimentel et al. 2021] Pimentel, J. F., Oliveira, G. P., Silva, M. O., Seufitelli, D. B., and Moro, M. M. (2021). Ciência de dados com reprodutibilidade usando jupyter. *Sociedade Brasileira de Computação*.

[Rey and Anselin 2007] Rey, S. J. and Anselin, L. (2007). PySAL: A Python Library of Spatial Analytical Methods. *The Review of Regional Studies*, 37(1):5–27.

[Tobler 1970] Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240.

[Torres-Preciado et al. 2014] Torres-Preciado, V. H., Polanco-Gaytan, M., and Tinoco-Zermeño, M. Á. (2014). Technological innovation and regional economic growth in mexico: a spatial perspective. *The Annals of Regional Science*, 52(1):183–200.

[Tukey 1977] Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley, 1st edition.

[Zhang et al. 2020] Zhang, X., Rao, H., Wu, Y., Huang, Y., and Dai, H. (2020). Comparison of spatiotemporal characteristics of the covid-19 and sars outbreaks in mainland china. *BMC infectious diseases*, 20(1):1–7.

Capítulo 2

Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional

Marcos Melo Ferreira, Caio Eduardo Falcão Matos

Abstract

Computer Vision has been widely used in developing industry, medicine, automation, and public safety tools. Due to the current relevance of the topic, a mini-course on the theme "Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional"was accepted to be presented at the tenth Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI), held between September 28 and 30, 2022 in São Luis Maranhão. This mini-course aims to present a method that allows computer vision techniques to control electronic devices. The mini-course is expected to arouse participants' interest in developing projects in which knowledge in Artificial Intelligence and Electronics is used.

Resumo

A Visão Computacional tem sido amplamente utilizada no desenvolvimento de ferramentas utilizadas na indústria, medicina, automação e segurança pública. Devido a relevância atual do tema foi proposto o minicurso "Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional", aceito para ministração na décima Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI), realizada entre os dias 28 e 30 de Setembro de 2022 em São Luís, Maranhão. Este minicurso tem como objetivo principal apresentar um método que possibilite a utilização de técnicas de visão computacional para controle de dispositivos eletroeletrônicos. Espera-se que o minicurso desperte o interesse dos participantes em desenvolver projetos em que sejam empregados conhecimentos das áreas de Inteligência Artificial e Eletroeletrônica.

2.1. Introdução

Recentemente, a Inteligência Artificial tem sido amplamente empregada por meio de diversas aplicações em diferentes áreas, que vão desde a criação de ferramentas para auxílio de diagnóstico de doenças por meio exames de imagem [Ferreira et al., 2020],

[Shang et al., 2019], [Raman et al., 2019] a mecanismos de análise de sentimentos de usuários de redes sociais [Zhang et al., 2018] e detecção de *fake news* presentes em textos postados diariamente na Internet [Kaliyar et al., 2021]. Uma das sub-áreas da Inteligência Artificial é a Visão Computacional, que estuda métodos para processamento de imagens do mundo real por um computador. Por meio da visão computacional é possível detectar e diferenciar objetos presentes em uma imagem, classificar exames de imagem, determinando quais possuem maior probabilidade de pertencerem a um paciente que está acometido por algum tipo de doença. Uma das diversas possibilidades de aplicações, é o controle de dispositivos eletroeletrônicos por meio da visão computacional. O controle é realizado pela detecção de mãos, presentes em uma imagem capturada por meio de uma webcam. Caso alguma mão seja detectada, o computador irá calcular parâmetros que serão enviados a uma placa eletrônica responsável por acionar os dispositivos. Utilizando esta tecnologia pode ser possível desenvolver aplicações de automação residencial e de tecnologias assistivas. Deste modo é interessante que alunos de cursos da área de Tecnologia da Informação sejam apresentados e possuam noções sobre Inteligência Artificial e uma de suas sub-áreas, a Visão Computacional.

2.2. Visão Computacional

A visão computacional é uma das áreas da Inteligência Artificial em que mais pesquisas têm sido feitas, com o propósito de cada vez mais elevar a acurácia de métodos que possibilitem às máquinas a capacidade de interpretar visualmente informações, ou seja, enxergar. Esta capacidade de enxergar significa que as máquinas podem transformar dados de uma imagem ou de um vídeo em uma nova representação, utilizada para que algum objetivo seja alcançado, como detecção e identificação de objetos presentes em uma imagem [Kaehler and Bradski, 2016]. Essa capacidade possibilitou o desenvolvimento de aplicações em diferentes áreas, como na indústria onde são realizadas verificações não invasivas em embalagens e inspeção de maquinário e estruturas, e na medicina, onde informações são extraídas de exames de forma automática para auxiliar os médicos na obtenção de diagnósticos de forma mais rápida e precisa. Como é um campo que tem atraído a atenção de pesquisadores devido ao grande número de possibilidades de aplicação, diversas tecnologias vem sendo desenvolvidas rapidamente.

A realização do processamento de imagens consiste inicialmente na extração de características das imagens. Estas características são utilizadas por algoritmos de aprendizado de máquina, que podem ser configurados para obtenção de coordenadas de *bounding boxes*, classificação das imagens, segmentação semântica de um alvo, por exemplo veias presentes em olhos, dentre outros [Prince, 2012]. Anteriormente, métodos manuais (*handcrafted*) eram utilizados para obtenção das características. Recentemente, métodos baseados em redes neurais alcançaram resultados superiores nas tarefas de processamento de imagens. Estas redes, especificamente as convolucionais, conseguem realizar a extração de características de forma automática, e melhoram este processo durante o seu treinamento, onde ocorre o aprendizado de quais características são mais relevantes para que determinada tarefa seja realizada.

2.3. Redes Neurais Convolucionais

As redes neurais convolucionais (CNN, do inglês *Convolutional Neural Network*) são um modelo de rede neural profunda. Enquanto redes neurais simples são formadas por poucas camadas, redes profundas são formadas por várias camadas, uma camada de entrada, uma de saída e várias camadas escondidas. Outra característica deste modelo de rede é ser do tipo *feedforward*, ou seja, não existe realimentação. Os dados percorrem a rede em uma única direção, da entrada para a saída [Goodfellow et al., 2016].

A principal características deste modelo é a realização da operação matemática denominada convolução. (Equação 1).

$$s(t) = (x * w)(t) \tag{1}$$

onde x representa o valor de uma variável em um instante t, e w representa uma função de ponderação. Em redes convolucionais, o primeiro argumento da equação refere-se a um dado de entrada, por exemplo os pixels que formam uma região de uma imagem, e o segundo refere-se a um filtro (*kernel*). O resultado é geralmente denominado mapa de características (*feature maps*). Redes Convolucionais são redes neurais que utilizam a operação matemática convolução em pelo menos uma de sua camadas [Goodfellow et al., 2016]. Estas camadas são denominadas convolucionais, e fazem parte do extrator de características visuais. Como os computadores utilizam dados discretos, pode-se definir a convolução discreta como sendo a Equação 2.

$$s(t) = (x * w)(t) = \sum_{a=0}^{N} x(a)w(t-a) \tag{2}$$

onde x e w serão definidos apenas para valores inteiros de t.

A Figura 2.1 ilustra a operação de convolução entre uma região de uma imagem e um filtro. Esta região da imagem é chamada de campo receptivo local [Academy, 2019]. Este campo receptivo é deslizado por toda a imagem, a um passo medido em *pixels* denominado *stride length*. Apesar da figura ilustrar a convolução sendo realizada com os pixels variando entre 0 e 255, antes do cálculo acontece uma normalização. Cada pixel é normalizado para um valor entre zero e um.

Em redes neurais que utilizam multiplicação de matrizes, a saída de um neurônio é utilizado como entrada para todos os neurônios da próxima camada [Haykin, 1999]. Como essa conectividade "total" entre neurônios não acontece em uma camada convolucional, essas redes possuem interações esparsas. Isso acontece devido ao filtro possuir dimensões menores que a entrada [Goodfellow et al., 2016], como representado na Figura 2.1. Com isso é necessário a realização de menos operações matemáticas diminuindo o custo computacional. Através da convolução é possível detectar pequenas, porém relevantes características que serão utilizadas para classificação das imagens. Além disso, em uma rede convolucional existe o compartilhamento de pesos. Um mesmo filtro é utilizado para realizar a convolução em toda a imagem, o que não acontece em uma rede totalmente conectada, onde existe um peso para cada conexão entre o neurônio de uma camada e os neurônios da camada seguinte. Isso implica em dizer que existem menos parâmetros para serem analisados e ajustados, o que diminui o custo computacional. Outra vantagem

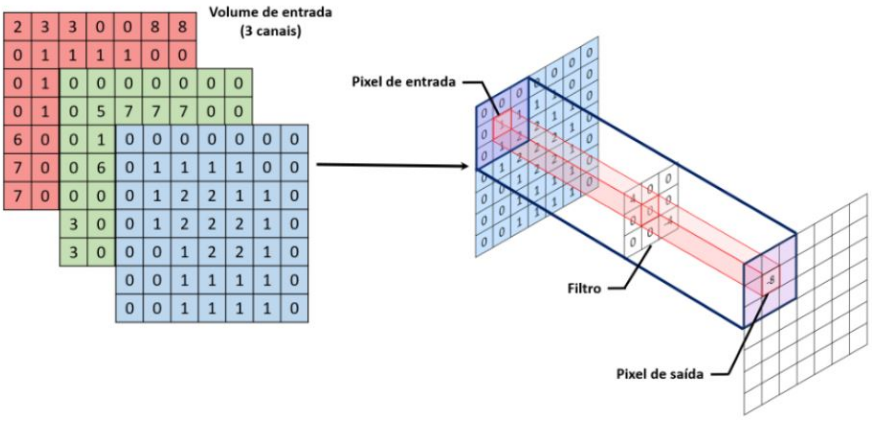


Figura 2.1. Convolução Discreta. [Araújo et al., 2017]

dessas redes é o fato de necessitarem de pouco pré-processamento. Enquanto em outros métodos o design dos filtros é feito por algum especialista, com o treinamento necessário, as Redes Convolucionais são capazes de aprender a ajustar os filtros [Saha, 2018].

As redes convolucionais possuem, além de camadas que realizam a operação de convolução, camadas que realizam uma operação denominada *pooling*. Em geral, o extrator de características deste tipo de rede é formado por blocos que contém duas ou mais camadas de convolução seguidas de uma camada de *pooling*. Esta função substitui a saída de uma camada por um índice estatístico. Por exemplo, a operação *Max Pooling*, predominantemente utilizada, reporta a saída máxima dentre as saídas existentes em uma vizinha retangular [Goodfellow et al., 2016]. Assim como a camada de convolução, a camada de *Max Pooling* também reduz o tamanho de uma imagem, uma vez que somente retorna um valor de uma determinada região da imagem de entrada. Além disso, a utilização do *Max Pooling* produz invariância à pequenas translações da imagem de entrada. Mesmo que haja alguma variação deste tipo, sendo que são imagens de um mesmo objeto, essa invariância evita que a rede classifique as variações de uma mesma imagem como imagens diferentes.

Em CNNs utilizadas para classificação de imagens, após os blocos formados pelas camadas de Convolução/Max Pooling, que tem por finalidade extrair características de imagens de entrada, são utilizadas camadas que irão utilizar estas características para gerar o resultado final da classificação. Os classificadores dos modelos são formados por camadas totalmente conectadas, denominadas camadas densas, exatamente iguais a uma rede MLP (do inglês, *Multi Layer Perceptron*) [Araújo et al., 2017]. Isto significa que todos os neurônios de uma camada estão conectados aos neurônios da camada seguinte [Goodfellow et al., 2016]. Em termos matemáticos, um neurônio pode ser descrito pelas Equações 3 e 4

$$h_j = \sum_{i=1}^{m} w_{ij} x_i \tag{3}$$

$$y_j = \varphi(h_j + b_j) \tag{4}$$

onde x_i são os m sinais de entrada, w_{ij} são os pesos sinápticos do neurônio j, e b_j corresponde ao *bias*, responsável por realizar o deslocamento da função de ativação definida por φ [Araújo et al., 2017].

Para que seja possível utilizar os mapas de características como entrada do classificador é necessário alterar o formato matricial dos dados para vetorial. Esse processo é denominado *flattening*. Em geral, as primeiras camadas densas utilizam a função ReLu (*Rectified Linear Unit*) Equação 5, como função de ativação, e a última camada do classificador utiliza a função de ativação *softmax* [Bishop, 2006], definida pela Equação 6

$$f(x) = max(0,x) \tag{5}$$

$$\sigma(Z)_i = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zj}} \tag{6}$$

onde z é um vetor de números reais que é normalizado em uma distribuição de K probabilidades. Esta função tem como saída uma distribuição de probabilidades, que é utilizada para classificação das imagens, e o número de neurônios da camada é igual ao número de classes em que as imagens podem ser classificadas. Outro parâmetro importante dos classificadores é o percentual de *dropout*, que consiste na remoção aleatória de um percentual de neurônios a cada iteração de treinamento, com o propósito de evitar *overfitting* e reduzir o tempo de treinamento [Goodfellow et al., 2016]. A arquitetura básica de uma CNN utilizada para classificação de imagens é ilustrada na Figura 2.2.

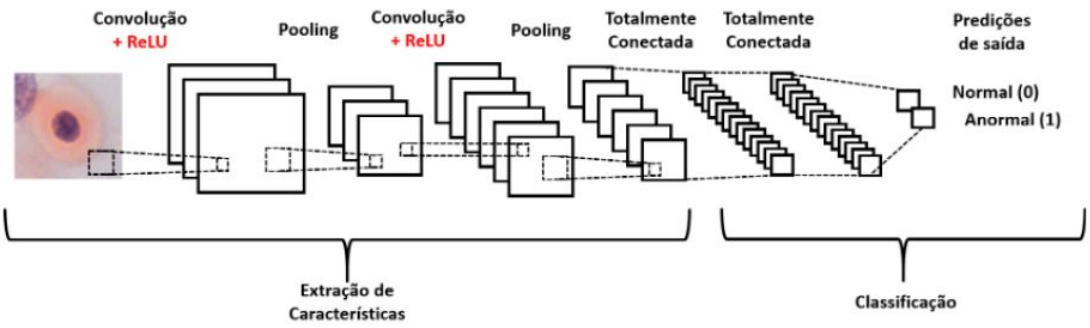


Figura 2.2. Arquitetura de uma rede convolucional. [Araújo et al., 2017].

2.4. Detecção de Objetos

A classificação de imagens somente faz sentido para imagens que possuem apenas um objeto, podendo serem citados os exames que contém imagens de um determinado órgão do corpo. Para o caso de imagens que possuem vários objetos é necessário localizar e classificar cada objeto presente na imagem. Esta tarefa é denominada detecção de objetos. Modelos computacionais utilizados para esta finalidade têm como saída não apenas a probabilidade de um objeto pertencer a uma classe, mas também as quatro coordenadas espaciais que serão utilizadas para traçado da caixa delimitadora (*Bounding Box*), retângulo que é utilizado para ilustrar a localização de um objeto em uma imagem. Deste modo, o

modelo computacional terá que prever a classe do objeto, coordenadas do canto superior esquerdo, altura e largura do retângulo. Uma forma comum de lidar com este problema consiste no uso de janelas deslizantes. Para realizar a detecção, as imagens são percorridas por janelas de tamanho fixo, que servem de entrada para um classificador. Algumas abordagens têm sido empregadas para este fim, como a Detecção de Objetos utilizando Histograma de Gradientes Orientados [Dalal and Triggs, 2005] e as Redes Neurais Convolucionais. Uma vez que o custo computacional das redes neurais profundas é maior que o de outras abordagens de aprendizado de máquina, é inviável alimentar uma rede com todas as regiões de uma imagem, com escalas diferentes. Sendo assim foram propostas as Redes Neurais Convolucionais Baseadas em Regiões (R-CNN) [Girshick et al., 2015], que trouxeram uma solução denominada busca seletiva (*Selective Search*), onde um algoritmo realiza uma busca na imagem para encontrar regiões com maior probabilidade de conter objetos, diminuindo o número de bounding boxes que serão utilizados como entrada para a rede neural. No entanto, o processo de detecção com CNNs ainda era lento, sendo que outras abordagens que tornaram o processo mais rápido foram propostas por [He et al., 2015], [Girshick, 2015], [Ren et al., 2015].

Outra abordagem existente para o solução do problema de detecção foi proposta por [Redmon et al., 2016], que apresentou a Rede Yolo (do inglês, *You Only Look Once*). Esta abordagem difere das anteriores porque a detecção não é baseada em regiões da imagem, logo não existe o *pipeline* para escolha de regiões da imagem que possuem um maior probabilidade de existência de objetos. Ao contrário das outras redes, a YOLO recebe como entrada a imagem completa e a divide em um *grid* de tamanho SxS. A rede então prevê os *bounding boxes* e coeficiente de confiança (*confidence score*) para cada um, sendo que para a maioria este valor será muito baixo e o bouding box será descartado. Além disso, a rede também prevê a probabilidade de cada divisão do *grid* pertencer a uma determinada classe. Devido ao fato da rede ter como entrada apenas uma imagem a detecção é realizada de forma muito rápida e pode ser executada em tempo real. No entanto, como a rede prevê apenas um tipo de classe para cada divisão do *grid*, esta rede apresenta como limitação a dificuldade de detectar objetos muito pequenos. A rede é formada por camadas de convolução, geralmente sendo escolhida alguma CNN utilizada para classificação de imagens como *backbone*, seguida de duas camadas totalmente conectadas (*fully connected*). A arquitetura e o formato da saída da rede são mostrados na Figura 2.3.

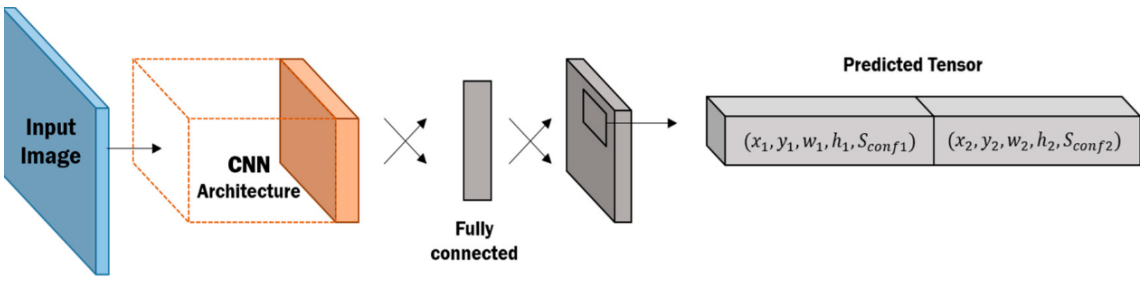


Figura 2.3. Arquitetura da Rede YOLO. [Redmon et al., 2016].

2.5. Placa Arduino Uno

A placa microcontroladora Arduino tem sido amplamente utilizada para o desenvolvimento de diversas aplicações em áreas como automação, robótica e tecnologias assistivas [Kondaveeti et al., 2021]. A placa pode ser entendida como um pequeno computador que pode ser programado para processar entradas e saídas, que são componentes externos conectados a ele [McRoberts, 2018]. Devido à uma maior facilidade para utilização e construção de protótipos, quando comparada à outras tecnologias, esta plataforma passou a ser bastante utilizada por professores e pesquisadores, e mais recentemente por profissionais de áreas citadas anteriormente. Existem diversos tipos de Arduino, sendo um dos mais utilizados, o Arduino Uno, que tem o microcontrolador ATmega328P produzido pela empresa Atmel. A placa, Figura 2.4, possui catorze pinos de E/S (Entrada e Saída), sendo seis que podem realizar a leitura de sinais analógicos e seis com função PWM (do inglês, *Pulse Width Modulation*), porta para comunicação serial, pino para alimentação, dentre outros periféricos.

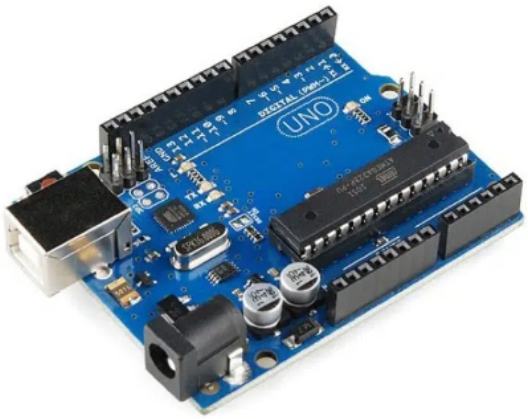


Figura 2.4. Placa Arduino Uno.

Uma das grandes vantagens da utilização de placas Arduino é a existência de módulos de expansão, que em geral são compatíveis com qualquer tipo de placa com necessidade de pouca ou nenhuma adaptação. Existem diferentes módulos, desde sensores de presença, aferição de temperatura e umidade e sensores ultrassônicos, diversos tipos de display, módulos para comunicação via protocolo TCP/IP, dentre outros. Estes módulos facilitam o desenvolvimento de aplicações um pouco mais complexas, que necessitam de sensores ou atuadores específicos. A Figura 2.5 mostra alguns exemplos de módulos de expansão, em sua maioria sensores digitais ou analógicos, juntamente com uma placa Arduino Uno.

2.6. Experimentos

Os experimentos utilizados neste minicurso têm como objetivo principal apresentar uma possibilidade de desenvolvimento de aplicações que utilizem de forma integrada, conceitos de visão computacional e de eletrônica, tendo em vista que grandes avanços têm sido alcançados recentemente nesta área da inteligência artificial, e que a mesma vem sendo cada vez mais explorada para desenvolvimento de produtos que possam ser utilizados

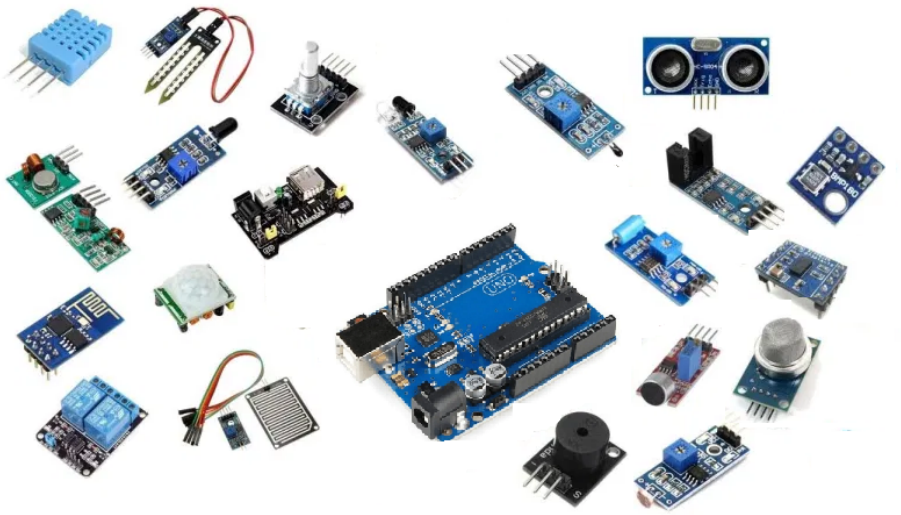


Figura 2.5. Módulos de Expansão.

para trazer melhorias à vida das pessoas, aumento de produtividade de empresas, dentre outros.

Os experimentos apresentam uma forma que possibilita o controle de circuitos eletrônicos através de sinais feitos com as mãos. Para tanto, é necessário um computador equipado com uma webcam, uma placa Arduino, módulos relés, lâmpadas ou outros dispositivos eletrônicos, além de fios e conectores elétricos. Nestes experimentos, as imagens capturadas pela webcam são os sinais de entrada do circuito de controle. As imagens capturadas são processadas para que sejam traçados os *bounding boxes*, permitindo que pontos de referência (*landmarks*) sejam encontrados. Por meio da localização destes landmarks, são geradas interpretações, por exemplo, qual mão foi detectada (direita ou esquerda), quantos dedos estão levantados e o cálculo do valor da distância entre dois dedos. Uma vez que a detecção é capaz de gerar diferentes saídas, basta utilizá-las para enviar diferentes códigos binários (caracteres ou números decimais) para o controlador do circuito eletrônico. A placa controladora Arduino Uno foi utilizada como "cérebro do circuito de controle". A placa recebe esses dados através de sua comunicação serial, e os utiliza como entrada para controle de suas saídas (dispositivos eletrônicos). Uma visão geral do recebimento e envio dos dados de entrada e saída dos experimentos é ilustrada na Figura 2.6.

2.7. Módulo Detector de Mãos

Com os recentes avanços no campo da visão computacional, diversas aplicações têm sido desenvolvidas com propósito de oferecer soluções para diversos problemas do cotidiano. Sistemas de visão têm sido implementados na robótica, veículos autônomos, câmeras, dentre outros. Em geral, estes sistemas buscam detectar um ou mais objetos presentes em imagens. Redes Neurais como a YOLO são treinadas para detectar e classificar objetos presentes em imagens captadas por uma câmera. Várias soluções de campos da Inteligência Artificial são disponibilizadas para desenvolvedores por meio de bibliotecas de alguma linguagem de programação como Python, R ou Matlab. Um exemplo é a biblio-

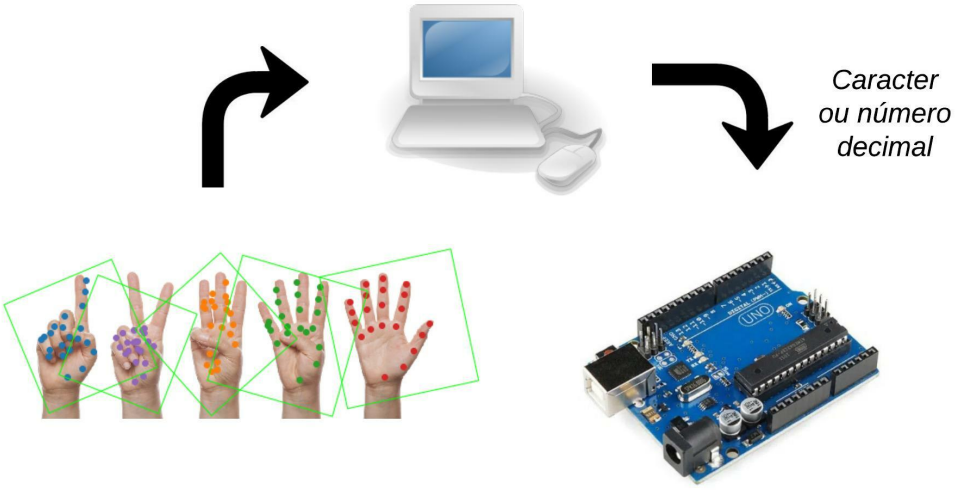


Figura 2.6. Experimento proposto.

teca *opencv*, que é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão Computacional.

Uma biblioteca disponível para desenvolvedores que utilizam a linguagem de programação python que vem sendo bastante utilizada em aplicações de visão computacional, é a CVzone (*Computer Vision Zone*), solução criada por [Hassan, 2020], que simplifica a utilização de módulos presentes na biblioteca *opencv*. Estão disponíveis tutoriais, cursos, livros e projetos, que são utilizados como ferramentas para o ensino de diversas aplicações como programação de drones, detecção de textos, detecção de objetos, robótica e realidade aumentada. Através da biblioteca são disponibilizadas diversas ferramentas e módulos que podem ser utilizados para desenvolvimento de aplicações, como a apresentada na Figura 2.7, em que objetos presentes em uma imagem são detectados e são rotulados de acordo com a maior probabilidade de pertencerem à uma classe, dentro de um número *n* de classes possíveis. O uso desta biblioteca facilita a realização de pesquisas para desenvolvimento de sistemas de reconhecimento de gestos e sinais, como o projeto apresentado por [Testa, 2020], em que foi desenvolvido um sistema de reconhecimento da Língua Brasileira de Sinais (LIBRAS) aplicado a robôs de serviço.

Um dos módulos existentes na biblioteca é o módulo detector de mãos. Este módulo fornece uma ferramenta capaz de detectar mãos que estão presentes em uma imagem, contar quantos dedos estão levantados e calcular distância entre dedos ou entre mãos. Cada mão detectada em uma imagem recebe o rótulo de esquerda (left) ou direita (right) e a marcação dos 21 pontos chaves (*landmarks*) utilizados para identificação, que são mostrados na Figura 2.8. Por meio destes pontos é possível realizar o cálculo de distâncias específicas, além do reconhecimento de sinais realizados com as mãos. A Figura 2.9 apresenta um exemplo de detecção realizada por meio do módulo *handDetector*.

A parte inicial do primeiro código-fonte utilizado para controle de dispositivos eletroeletrônicos é mostrado na Figura 2.10. Neste exemplo, são utilizados como sinais de entrada para o circuito de controle, o total de dedos levantados em uma mão, que tem

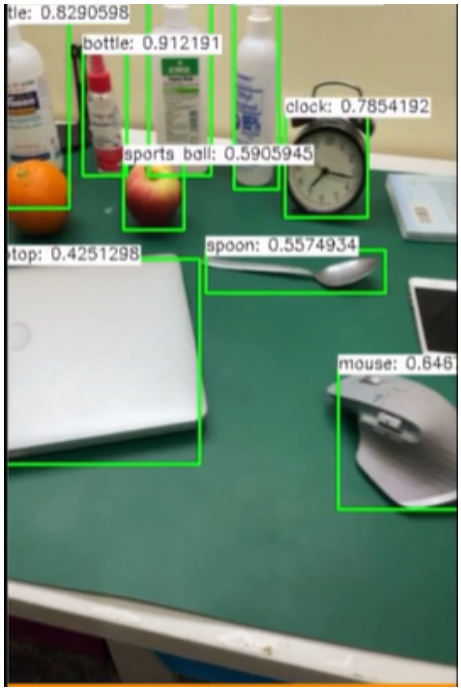


Figura 2.7. Exemplo de aplicação de Visão Computacional.

sua imagem capturada por uma webcam. Deste modo, é possível controlar o acionamento de até 5 dispositivos. Inicialmente, é feita a instalação de algumas bibliotecas (linhas 1 a 6), que são comumente utilizadas e algumas que são requisitos para funcionamento do código. É importante que sejam utilizados ambientes virtuais para execução dos códigos. Estes ambientes permitem que projetos, dependências e bibliotecas sejam isolados em um único local. Deste modo, elimina-se a possibilidade de que a instalação das dependências de um projeto interfira em outros projetos, já que é relativamente comum que no desenvolvimentos de projetos sejam utilizadas versões diferentes de uma mesma biblioteca. O editor de código-fonte escolhido para criação dos projetos foi o Visual Studio Code.

Após a instalação das dependências (bibliotecas), é iniciada a captura de vídeo via webcam (linha 9), seguida da inicialização da comunicação serial (linhas 11 e 12). É importante especificar de forma correta a porta que será utilizada para comunicação serial com a placa Arduino. Para verificar qual porta está sendo utilizada, basta acessar o ícone Ferramentas/Porta na IDE (do inglês, *Integrated Development Environment*) Arduino. A seguir, é necessário configurar a captura do detector. Dois parâmetros são essenciais, o limite de confiança para detecção de objetos (detectionCon) e o número máximo de mãos que será detectado (maxHands). Além disso, é utilizada uma variável que irá armazenar o total de dedos que estarão levantados. A parte final do código, Figura 2.11, é composta for comandos utilizados para contabilizar o total de dedos levantados e pelo envio deste dado à placa Arduino. Uma vez que estes comandos precisam ser executados de forma ininterrupta, é utilizada a instrução *while*. Inicialmente, é realizada a captura de um quadro de vídeo (linha 15), seguido do ajuste do tamanho da janela em que as imagens serão exibidas (linha 16). A próxima instrução (linha 17) é utilizada para obtenção dos landmarks de cada mão detectada na imagem, sendo que esses dados são salvos em uma lista,

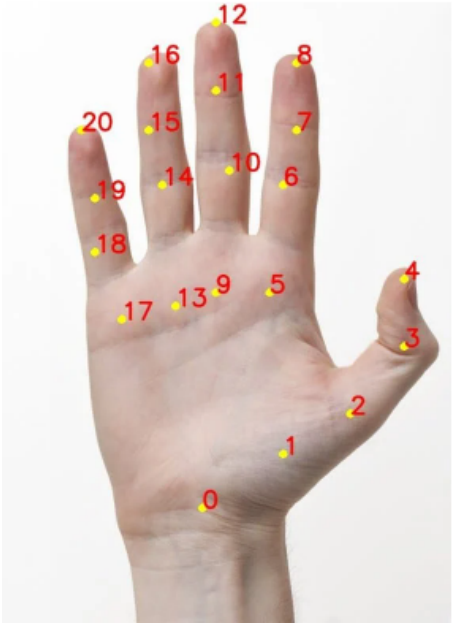


Figura 2.8. Pontos Utilizados para Identificação das Mãos.

denominada *hands*. A seguir, é utilizado o método *fingersUp* (linha 23) que retorna uma lista contendo 5 bits indicando o estado de cada dedo, sendo 1 utilizado para indicar que um determinado dedo está levantado. Por exemplo, se a lista retornada contiver os valores [0,0,1,1,1] significa que os dedos polegar e indicador estão abaixados e os demais estão levantados. A parte final do código contém as instruções utilizadas para contabilizar quantos dedos estão levantados, enviar este dado para a placa Arduino e para escrever a informação na janela que está exibindo a captura feita pela webcam.

O segundo exemplo consiste em controlar um dos sinais PWM da placa Arduino, utilizando como sinal de entrada a distância entre dois pontos chave (landmarks). É possível utilizar a distância entre dois pontos presentes na mesma mão (Figura 2.12) ou em mãos diferentes. A distância calculada será enviada a placa, que utilizará este valor como referência para controle de algum dispositivo conectado a uma de suas saídas analógicas. Utilizando a saída PWM é possível controlar a velocidade de rotação de um motor, a intensidade luminosa de uma lâmpada, assim como outras grandezas analógicas. As instruções utilizadas para que o módulo calcule a distância entre dois pontos são mostradas na Figura 2.13. É necessário obter a lista com os 21 pontos (linha 21), e depois utilizar o método *findDistance* (linha 22), que tem como parâmetros a especificação de quais pontos chave serão utilizados como referência para o cálculo. Também foram utilizadas instruções para desenhar um gráfico de barra, que mostra como o percentual de ciclo de trabalho do sinal de PWM deve variar de acordo com a distância entre os dedos polegar e indicador. Para isso, foram desenhados dois retângulos (linhas 29 e 30), um que tem altura fixa e outro com altura variável, além do valor percentual. É importante ajustar o valor da distância calculada à posição do gráfico desenhado na tela, sendo para isso utilizado o método *interp* da biblioteca *numpy*.

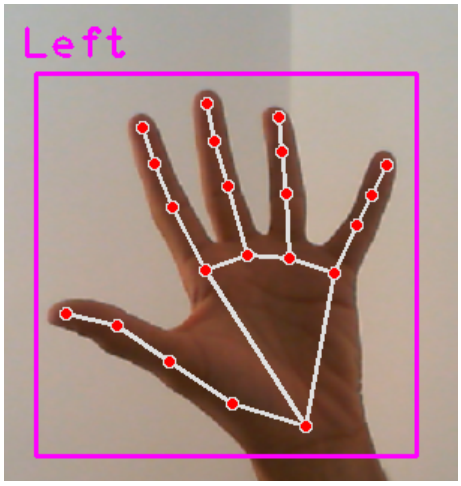


Figura 2.9. Detecção Realizada com o Módulo *HandDetector.*

```
1  import numpy
2  import serial
3  import cvzone
4  import cv2
5  from cvzone.HandTrackingModule import HandDetector
6  import matplotlib.pyplot as plt
7
8  #inicia a captura via webcam
9  cap = cv2.VideoCapture(0)
10 #inicia a comunicação com Arduino
11 serialcomm = serial.Serial('COM16', 9600)
12 serialcomm.timeout = 1
13 #configuração do detector
14 detector = HandDetector(detectionCon=0.8,maxHands=1)
15 #inicializa contador
16 w=0
```

Figura 2.10. Parte Inicial do Código Utilizado no Primeiro Experimento.

2.8. Programação da Placa Arduino

A programação de uma placa Arduino é realizada utilizando-se a linguagem C++ adaptada, e os códigos-fonte são escritos, compilados e enviados para a placa utilizando-se a IDE Arduino. A organização Arduino disponibiliza para consulta alguns tutoriais, últimas atualizações de placas e bibliotecas, além de outros serviços. Todos os serviços são gratuitos, uma vez que a iniciativa Arduino tem como objetivo a disseminação de conhecimento para o maior número de pessoas possível.

As instruções utilizadas para escrita dos algoritmos são em geral simples e intuitivas. Também é possível instalar bibliotecas que permitem utilizar de forma mais simples módulos de expansão. Diariamente, desenvolvedores disponibilizam versões de bibliotecas para tornar cada vez mais simples o desenvolvimento de projetos com a plataforma. Cada código fonte Arduino é composto obrigatoriamente de duas funções: *void setup() e void loop()*. A primeira é formada por comandos que irão configurar os pe-

```
14 while True:
15     success, img = cap.read() #leitura da imagem
16     img=cv2.resize(img,(800, 600)) #dimensiona janela
17     #Detecta mao(s) presentes na imagem
18     hands, img = detector.findHands(img,draw=True)
19     if hands:
20         #escolhe a 1ª mão detectada
21         hand1 = hands[0]
22         #armazena em um vetor quais dedos estão levantados
23         f = detector.fingersUp(hand1)
24         w=0
25         e='\n'#delimitador de dado transmitido
26         for q in f:
27             w +=q
28         #envia valor para arduino
29         serialcomm.write(e.encode())
30         serialcomm.write(str(w).encode())
31     cv2.putText(img, "Dedos: " + str(int(w)), (10,70),
32     cv2.FONT_HERSHEY_PLAIN, 3, (0,0,255), 3)
33     cv2.imshow("Image", img)
34     cv2.waitKey(10)
```

Figura 2.11. Parte Final do Código Utilizado no Primeiro Experimento.

riféricos da placa, como pinos de entrada e saída, comunicação serial, além de ser utilizada inicialização de alguns dispositivos como displays e relógios. A segunda contém as instruções que serão executadas de forma cíclica, durante o tempo em que a placa estiver alimentada.

O código necessário para que a placa receba dados de um computador de forma serial é bastante simples, uma vez que os comandos utilizados para comunicação são nativos da linguagem, além da IDE conter um monitor para comunicação serial. A parte inicial do código, Figura 2.14, contém as variáveis que serão utilizadas. A linha 1 contém a variável que irá armazenar a string recebida pelo Arduino, seguida da definição de conexão das lâmpadas nos pinos da placa (linha 3). A seguir a comunicação serial é iniciada com taxa de transmissão de dados de 9600 pbs, e os pinos onde serão conectadas as lâmpadas são definidos como saídas.

A parte principal, Figura 2.15 do código contém as instruções responsáveis pelo recebimento dos dados enviados pelo computador e pelo acionamento dos dispositivos eletroeletrônicos. Inicialmente, é verificada se a comunicação serial foi estabelecida (linha 18). A seguir a string enviada pelo computador é armazenada (linha 21) e convertida para um valor do tipo inteiro (linha 23). Agora basta que esse valor seja testado para que seja definido de que forma as lâmpadas serão acionadas. Optou-se pela escolha do comando switch para seleção das ações que serão tomadas pelo controlador. É necessário observar que para acionar uma lâmpada utilizou-se o comando *digitalWrite* com argumento LOW (nível lógico baixo), e para apagar o argumento HIGH foi utilizado. Isto deve-se devido ao módulo relé utilizado ser acionado com nível lógico baixo.

Além de ser possível controlar o acionamento de saídas digitais, a placa arduino também permite que saídas sejam controladas de forma analógica. Um exemplo, é o

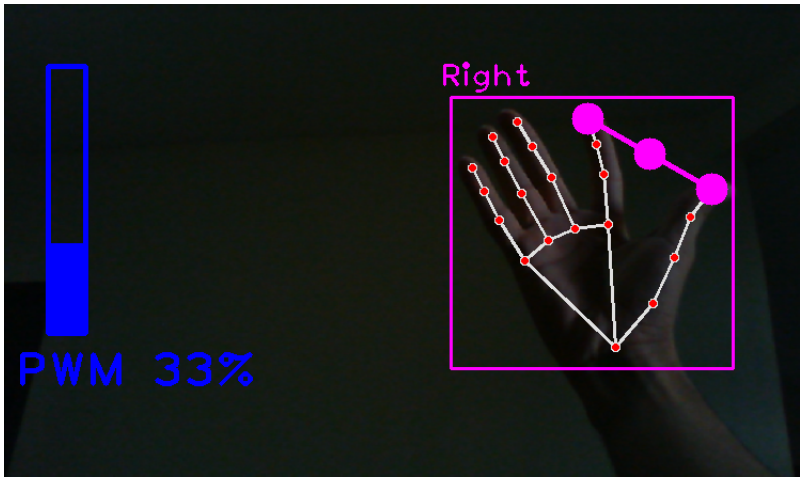


Figura 2.12. Distância entre dois pontos utilizada para controle de um sinal PWM.

controle da velocidade de um motor de corrente contínua ou o controle da intensidade luminosa de um LED ou de uma lâmpada. Para isso, é necessária a utilização de uma das saídas PWM da placa arduino. Para encontrar estes pinos basta procura pelo símbolo semelhante à uma onda. Utilizando-se um sinal PWM, é possível controlar o valor de potência entregue à uma carga. Uma vez que o módulo de detecção de mãos consegue realizar a medição da distância entre dois dedos, ou entre duas mãos, o valor dessa distância é enviado para a placa arduino para ser utilizado como valor de referência para o PWM. O código para este tipo de acionamento é mostrado na Figura 2.16. O código é relativamente simples. Após o recebimento da string e transformação da mesma em um número do tipo inteiro (linha 10), o valor recebido deve ser restringido a um intervalo que varia entre 0 e 255 (linha 13), uma vez que são os valores mínimo e máximo utilizados para configuração do PWM. A seguir, basta enviar este valor para o pino onde está conectado o dispositivo que será controlado de forma analógica (linha 14).

2.9. Circuito de Acionamento

A placa Arduino, assim como outros dispositivos de controle, emitem um sinal de saída capaz de acionar pequenas cargas, como LEDs (Light Emitting Diodes). Em geral os pinos operam em 3,3 ou 5 V, fornecendo uma corrente máxima que costuma variar entre 25 e 40 mA [Banzi and Shiloh, 2022]. Neste caso para que a placa Arduino possa controlar dispositivos alimentados com valores mais elevados de corrente e tensão elétrica é necessário utilizar algum circuito de acionamento ou circuito de potência. Este tipo de circuito é amplamente utilizado na indústria e em outros setores de serviços, uma vez que em uma instalação elétrica existem circuitos que operam com diferentes níveis de tensão [Segundo and Rodrigues, 2015].

Existem diferentes componentes elétricos que podem ser utilizados para permitir a conexão da placa arduino com uma uma carga que opere com níveis de potência mais elevado, ou que sejam alimentados com tensão alternada. Podemos citar o Retificador Controlado de Silício (SCR, do inglês *Silicon Controlled Rectifier*), TRIAC (do inglês, *Triode for Alternating Current*), optoacopladores ou acopladores ópticos, relés do estado sólido,

```python
while True:
    success, img = cap.read()
    img=cv2.resize(img,(800, 600))
    hands, img = detector.findHands(img,draw=True)
    if hands:
        hand1 = hands[0]
        #lista de 21 landmarks - pontos detectados em uma mão
        lmlist1 = hand1["lmList"]
        length, info, img = detector.findDistance(lmlist1[4], lmlist1[8],img)
        e='\n'
        # envia para o arduino
        serialcomm.write(e.encode())
        serialcomm.write(str(length).encode())
    bar = np.interp(length, [50, 300], [400, 150])
    bar_per = np.interp(length, [50, 300], [0, 100])
    cv2.rectangle (img, (50,150), (85,400), (255,0,0), 3)
    cv2.rectangle (img, (50,int(bar)), (85,400), (255,0,0), cv2.FILLED)
    cv2.putText(img, f'PWM {int(bar_per)}%', (20,450),
    cv2.FONT_HERSHEY_PLAIN, 3, (255,0,0), 3)
    cv2.imshow("Image", img)
    cv2.waitKey(10)
```

Figura 2.13. Código python Utilizado no Segundo Experimento (PWM).

transistores, dentre outros. Devido ao fato de não haver componentes mecânicos em sua construção estes dispositivos são denominados chaves estáticas. O símbolo elétrico de alguns destes dispositivos é mostrado na Figura 2.17. A escolha de qual componente deve ser utilizado deve levar em consideração o custo, vida útil do componente, nível de proteção necessária, disponibilidade de mercado. No experimento descrito, é utilizado o relé eletromecânico devido ao seu menor custo e maior disponibilidade. Seu símbolo elétrico e o módulo utilizado são mostrados na Figura 2.18.

A conexão da placa Arduino ao módulo relé e do módulo à lâmpada são mostrados na Figura 2.19. O módulo relé permite que o Arduino controle dispositivos eletro-eletrônicos que são alimentados com tensão superior à tensão de operação dos pinos de saída da placa. Neste caso, a placa envia um sinal para o módulo relé energizando sua bobina, realizando o acionamento do dispositivo conectado a seus terminais alimentados com tensão elétrica mais elevada. O terminal fase do circuito de alimentação deve ser conectado ao terminal comum (COM) do relé. A lâmpada deve ter seus terminais conectados da seguinte forma: um conectado ao contato normalmente aberto do relé (NA, do inglês *Normally open*) e o outro deverá ser conectado diretamente ao terminal Neutro da alimentação. Desta forma, o relé realizará o chaveamento através do terminal da fase, procedimento recomendado pela norma NBR5410 da ABNT (Associação Brasileira de Normas Técnicas) [ABNT, 2004]. É necessário ter atenção com relação a conexão dos terminais do receptáculo da lâmpada, para que os terminais sejam conectados de forma correta, para uma conexão segura. Além do baixo custo, o relé tem como vantagem o fato de seus terminais de tensão baixa (que são conectados diretamente ao Arduino) serem totalmente isolados dos terminais de tensão mais elevada, onde são conectados os dispositivos que têm seu acionamento controlado pela placa Arduino.

```
String dado_byte;
//Pinos onde as lâmpadas serão conectadas
int L_1=4,L_2=5,L_3=6,L_4=7,L_5=8;
int num;
void setup( )
{
//Inicializa porta serial
Serial.begin(9600);
//configura pinos como saídas
pinMode(L_1,OUTPUT);
pinMode(L_2,OUTPUT);
pinMode(L_3,OUTPUT);
pinMode(L_4,OUTPUT);
pinMode(L_5,OUTPUT);
}
```

Figura 2.14. Experimento 1: Parte Inicial do código-fonte para programação da placa Arduino.

2.10. Considerações Finais

Neste capítulo foi descrito o minicurso intitulado “Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional”que foi aceito para ser ministrado na décima Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI). O curso apresenta de que forma é possível controlar o acionamento de dispositivos eletroeletrônicos, tanto de forma digital quanto de forma analógica, utilizando Visão Computacional, campo de estudo da Inteligência Artificial em que são pesquisadas e desenvolvidas formas que possibilitem aos computadores interpretar dados para extrair informações de imagens. Recentemente, grandes avanços foram alcançados como classificação automática de imagens, e detecção automática de objetos presentes em imagens e reconhecimento facial.

Inicialmente foram apresentados conceitos básicos de Visão Computacional, com uma breve explicação sobre os métodos utilizados para processamento de imagens. O tópico seguinte trouxe de forma detalhada a fundamentação teórica sobre as Redes Neurais Convolucionais, que são os modelos computacionais mais utilizados em tarefas de visão computacional. Foi explicado de que forma uma rede convolucional extrai características presentes em uma imagem para realizar a sua classificação, além de alguns parâmetros que devem ser ajustados para realização do treinamento da rede. A seguir, foi apresentada a placa Arduino Uno, utilizada para confecção dos circuitos elétricos, juntamente com uma visão geral dos experimentos. Detalhes sobre o processo de captura e detecção de mãos presentes em imagens também foram apresentados, assim como figuras que mostram exemplos de detecção realizada com métodos existentes nas bibliotecas opencv e cvzone, sendo que as principais instruções que fazem parte do código foram explicadas. A parte final traz o código-fonte utilizado para programação da placa Arduino e o esquema elétrico do circuito de acionamento.

O curso tem como objetivo principal apresentar uma possibilidade para desenvolvimento de projetos de inovação tecnológica e científica que apliquem de forma integrada conhecimentos das áreas de microeletrônica e inteligência artificial, especificamente da

```
16 void loop( )
17 {
18 if(Serial.available( ) > 0)
19  {
20     //armazena dado recebido
21     dado_byte = Serial.readStringUntil( '\n');
22     //converte para um valor do tipo inteiro
23     num = arrivingdatabyte.toInt();
24     switch (num){
25       case 1:
26       //Aciona Lampada 1
27       digitalWrite(L_1,LOW);
28       digitalWrite(L_2,HIGH);
29       digitalWrite(L_3,HIGH);
30       digitalWrite(L_4,HIGH);
31       digitalWrite(L_5,HIGH);
32       break;
```

Figura 2.15. Experimento 1: Parte Principal do código-fonte para programação da placa Arduino.

```
 7 void loop()
 8 {
 9 while (Serial.available() > 0){
10 int red = Serial.parseInt();
11 if (Serial.read() == '\n')
12   {
13    red = constrain(red, 0, 255);
14    analogWrite(redPin, red);
15    Serial.print(red);}
16   }
17 }
```

Figura 2.16. Experimento 2: Parte Principal do código-fonte para programação da placa Arduino.

área da Visão Computacional , que têm sido amplamente utilizadas em diversas aplicações atualmente. Espera-se que futuramente possam ser desenvolvidos projetos em áreas da automação, e principalmente que possam ser desenvolvidos serviços e recursos de tecnologia assistiva, que possam prover alguma assistência e melhoria a vida de pessoas que possuam alguma necessidade específica.

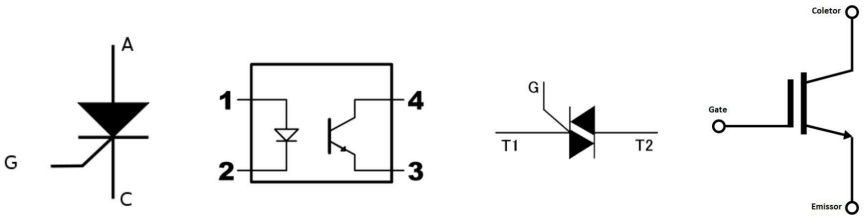


Figura 2.17. Chaves Estáticas de Potência (SCR, Optoacoplador, TRIAC, Transistor).

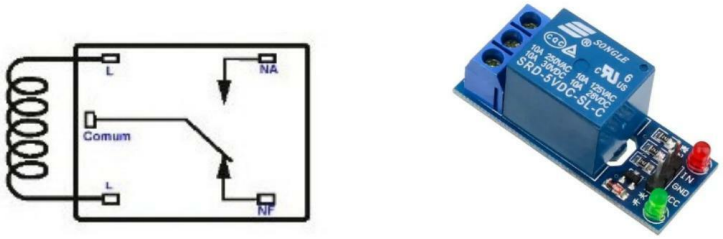


Figura 2.18. Relé: Símbolo elétrico e Módulo de expansão.

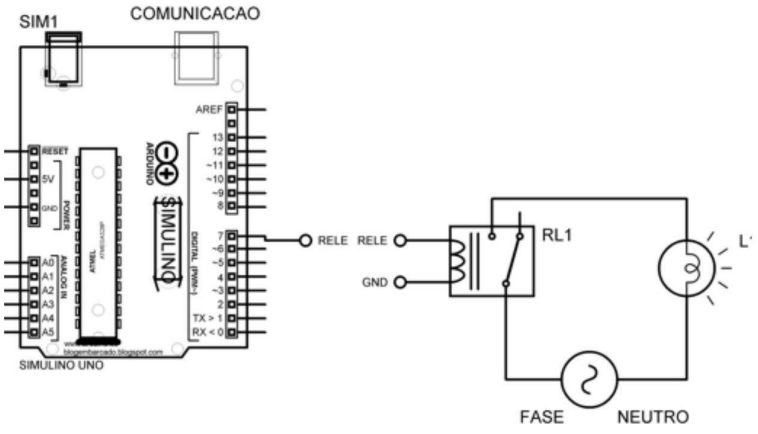


Figura 2.19. Conexão do circuito de acionamento.

Referências

[ABNT, 2004] ABNT (2004). *ABNT NBR 5410: instalações elétricas de baixa tensão*.

[Academy, 2019] Academy, D. S. (2019). Deep learning book. http://deeplearningbook.com.br. Disponível; acessado em 05-Junho-2022.

[Araújo et al., 2017] Araújo, F., Carneiro, A., Silva, R., Medeiros, F., and Ushizima, D. (2017). Redes neurais convolucionais com tensorflow:teoria e prática. 1:382–406.

[Banzi and Shiloh, 2022] Banzi, M. and Shiloh, M. (2022). *Getting started with Arduino*. Maker Media, Inc.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

[Ferreira et al., 2020] Ferreira, M. M., Esteve, G. P., Junior, G. B., de Almeida, J. D. S., de Paiva, A. C., and Veras, R. (2020). Multilevel cnn for angle closure glaucoma detection using as-oct images. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 105–110. IEEE.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[Girshick et al., 2015] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Hassan, 2020] Hassan, M. (2020). Computer vision zone: Cv zone. https://www.computervision.zone. Disponível; acessado em 05-Junho-2022.

[Haykin, 1999] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.

[Kaehler and Bradski, 2016] Kaehler, A. and Bradski, G. (2016). *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. "O'Reilly Media, Inc.".

[Kaliyar et al., 2021] Kaliyar, R. K., Goswami, A., and Narang, P. (2021). Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia Tools and Applications*, 80(8):11765–11788.

[Kondaveeti et al., 2021] Kondaveeti, H. K., Kumaravelu, N. K., Vanambathina, S. D., Mathe, S. E., and Vappangi, S. (2021). A systematic literature review on prototyping with arduino: Applications, challenges, advantages, and limitations. *Computer Science Review*, 40:100364.

[McRoberts, 2018] McRoberts, M. (2018). *Arduino básico*. Novatec Editora.

[Prince, 2012] Prince, S. (2012). *Computer Vision: Models, Learning, and Inference*. Computer Vision: Models, Learning, and Inference. Cambridge University Press.

[Raman et al., 2019] Raman, R., Srinivasan, S., Virmani, S., Sivaprasad, S., Rao, C., and Rajalakshmi, R. (2019). Fundus photograph-based deep learning algorithms in detecting diabetic retinopathy. *Eye*, 33(1):97–109.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

[Saha, 2018] Saha, S. (2018). A comprehensive guide to convolutional neural networks. http://www.towardsdatascience.com. Disponível; acessado em 20-Junho-2022.

[Segundo and Rodrigues, 2015] Segundo, A. K. R. and Rodrigues, C. L. C. (2015). Eletrônica de potência e acionamentos elétricos. *Ouro Preto: Ifmg-Instituto Federal de Ciencia, Educação e Tecnologia de Minas Gerais*.

[Shang et al., 2019] Shang, Q., Zhao, Y., Chen, Z., Hao, H., Li, F., Zhang, X., and Liu, J. (2019). Automated iris segmentation from anterior segment oct images with occludable angles via local phase tensor. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4745–4749, Berlin, Germany. IEEE.

[Testa, 2020] Testa, B. (2020). Sistema de reconhecimento de língua brasileira de sinais aplicado a robôs de serviço.

[Zhang et al., 2018] Zhang, L., Wang, S., and Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Capítulo 3

Explainability e *auditability*: interpretando e validando modelos de *machine learning*

Wendley Souza da Silva, Victória Tomé Oliveira, Sayonara Santos Araújo, Dario Vieira e Miguel Franklin de Castro

Abstract

Machine learning (ML) models and solutions have many applications in our daily lives, with the real possibility to improve products, processes, and techniques; however, in some situations, the systems do not sufficiently or convincingly explain their predictions. Understanding decision-making in highly sensitive areas such as healthcare or finance is of paramount importance to provide transparency, security, and trust to the users of intelligent systems. This short course introduces the participant, in theory and practice, to strategies and techniques for making machine learning models and their decisions interpretable and auditable.

Resumo

Os modelos e soluções de aprendizado de máquina (ML) possuem diversas aplicações no nosso cotidiano, com real possibilidade de melhorar produtos, processos e técnicas; entretanto, em algumas situações, os sistemas não explicam de forma suficiente ou convincente as suas previsões. Entender a tomada de decisões em áreas altamente sensíveis, como saúde ou finanças, é de suma importância para oferecer transparência, segurança e confiança aos usuários dos sistemas inteligentes. Este minicurso apresenta ao participante, na teoria e prática, as estratégias e técnicas para tornar os modelos de aprendizado de máquina e suas decisões interpretáveis e auditáveis.

3.1. Introdução

A precisão dos modelos atuais de Inteligência Artificial (IA) é notável, mas tal atributo não é o único aspecto de significativa importância. Para domínios sensíveis, uma compreensão detalhada do modelo e das saídas também é importante. Os algoritmos de aprendizado de máquina e aprendizado profundo constroem modelos complexos que são opacos

para humanos burkart2021survey. Holzinger et al. (2019) afirmam que o domínio médico está entre os maiores desafios para a IA. Para áreas como saúde, onde uma compreensão profunda da aplicação de IA é crucial, a necessidade de Inteligência Artificial Explicável (XAI, em inglês, *explainable artificial intelligence*) torna-se notável.

A explicabilidade é importante em muitos domínios, mas não exatamente em todos os domínios. Já mencionamos áreas em que a explicabilidade é importante, como a saúde. Por outro lado, em outros domínios como o de colisões de aeronaves, os algoritmos operam sem interação humana e sem dar explicações há anos. A explicabilidade é necessária quando há algum grau de completude (Burkart, Huber, 2021).

De acordo com Lipton (2018), a explicabilidade é exigida sempre que o objetivo para o qual o modelo de previsão foi construído difere do uso real quando o modelo está sendo implantado. Em outras palavras, a necessidade de explicabilidade surge devido ao descompasso entre o que um modelo pode explicar e o que um tomador de decisão deseja saber. De acordo com Martens et al. (2008), a explicabilidade é essencial sempre que um modelo precisa ser validado antes de ser implementado e implantado. Os domínios que demandam explicabilidade são caracterizados pela tomada de decisões críticas que envolvem, por exemplo, uma vida humana, mais especificamente, na área da saúde.

Nesse contexto, o renovado Regulamento Geral de Proteção de Dados da UE (GDPR) (Europa.eu, 2017) poderá exigir que os provedores de IA forneçam aos seus usuários explicações sobre os resultados da tomada de decisão automatizada com base em seus dados pessoais. O GDPR substituiu a Diretiva de Proteção de Dados de 1995 e esse novo requisito afeta grande parte da indústria. O Parlamento Europeu revisou os regulamentos que dizem respeito à coleta, armazenamento e uso de informações pessoais. O GDPR pode restringir ou até mesmo levar à proibição do uso de modelos opacos que são usados para determinadas aplicações, por exemplo, para sistemas de recomendação que funcionam com base em dados pessoais. Goodman, Flaxman (2017) chamam isso de direito de explicação para cada sujeito (pessoa). Isso provavelmente afetará instituições financeiras, redes sociais e o setor de saúde. A tomada de decisão automatizada usada por instituições financeiras para monitorar o risco de crédito ou lavagem de dinheiro precisa ser transparente, interpretável e responsável.

Em maio de 2017, a DARPA (Gunning, 2017) lançou o programa XAI que visa fornecer modelos explicáveis e altamente precisos. XAI é um termo genérico para qualquer pesquisa que tente resolver o problema da caixa preta para a IA. Uma vez que existem muitas abordagens diferentes para resolver este problema, cada uma com suas próprias necessidades e objetivos individuais, não há uma única definição comum do termo XAI. A ideia-chave, no entanto, é permitir que os usuários entendam a tomada de decisão de qualquer modelo.

Explicabilidade e interpretabilidade também são aspectos importantes para modelos de aprendizado profundo, onde uma decisão depende de uma enorme quantidade de pesos e parâmetros. Aqui, os parâmetros são muitas vezes abstratos e desconectados do mundo real, o que dificulta a interpretação e explicação dos resultados dos modelos profundos (Angelov, Soares, 2020). Por muito tempo, os modelos de ML eram universalmente vistos como caixas pretas porque humanos não podiam explicar o que acontecia com os dados entre a entrada e a saída. A partir disto, surgiu a explicabilidade, que será

detalhada a seguir. Dessa forma, este minicurso apresenta ao participante, na teoria e prática, as estratégias e técnicas para tornar os modelos de aprendizado de máquina e suas decisões explicáveis e interpretáveis, alcançando a auditabilidade.

3.2. Conceitos e técnicas de *explainability* e *auditability*

3.2.1. Explainability

Para explicar as previsões de um modelo de aprendizado de máquina, pode-se utilizar algum método de explicação, que na prática é um algoritmo que gera explicações. Uma explicação geralmente relaciona os valores de recursos de uma instância à previsão do modelo de uma maneira humanamente compreensível. A explicabilidade no aprendizado de máquina significa que você pode explicar o que acontece em seu modelo da entrada à saída, tornando os modelos mais transparentes e resolvendo, ou minimizando, o problema da caixa preta (Burkart, Huber, 2021).

A IA explicável (XAI) é a maneira mais formal de descrever isso e se aplica a toda a inteligência artificial. XAI significa métodos que ajudam especialistas humanos a entender as soluções desenvolvidas pela IA. "Explicabilidade" e "interpretabilidade" são frequentemente usados de forma intercambiável, apesar de terem o mesmo objetivo que é entender o modelo (Burkart, Huber, 2021).

Em seu livro, "*Interpretable Machine Learning*", Christoph Molnar (Molnar, 2020) define interpretabilidade como o grau em que um ser humano pode entender a causa de uma decisão ou o grau em que um ser humano pode prever consistentemente os resultados do modelo de ML.

A IA explicável trata de entender melhor os modelos de ML. Como eles tomam decisões e por quê. Os três aspectos mais importantes da explicabilidade do modelo são:

- Transparência;
- Capacidade de questionar;
- Facilidade de entendimento.

Pode-se abordar a explicabilidade de duas maneiras:

Globalmente Esta é a explicação geral do comportamento do modelo. Ele nos mostra uma visão geral do modelo e como os recursos nos dados afetam coletivamente o resultado;

Localmente Isso nos informa sobre cada instância e recurso nos dados individualmente (como explicar as observações vistas em determinados pontos do modelo) e como os recursos afetam individualmente o resultado.

3.2.1.1. Razões para a Explicação

Os sistemas automatizados de tomada de decisão não são amplamente aceitos, pois os humanos querem entender uma decisão ou pelo menos querem obter uma explicação

para certas decisões. Sendo assim, a confiança, então, é um dos aspectos motivadores da explicabilidade. Outros aspectos motivadores são causalidade, transferibilidade, informatividade, tomada de decisão justa e ética (Lipton, 2018), prestação de contas, ajustes e funcionalidade de proxy.

Confiança Confiança e aceitação do modelo de previsão são necessários para a implantação do modelo de previsão. Compreender e conhecer os pontos fortes e fracos do modelo de previsão é um pré-requisito para a confiança humana e, portanto, para a implantação do modelo.

Causalidade Explicação, por exemplo na forma de importância do atributo, transmite uma sensação de causalidade ao grupo-alvo do sistema. Este conceito de causalidade só pode ser apreendido quando o sistema aponta a relação insumo-produto subjacente.

Transferibilidade O modelo de previsão precisa transmitir uma compreensão do comportamento futuro para um tomador de decisão humano para usar o modelo de previsão com dados não vistos. Somente quando o decisor souber que o modelo generaliza bem ou quando souber em que contexto ele generaliza bem, o modelo de previsão será encarregado de tomar decisões.

Informatividade Para ser implantado como um sistema, é necessário saber se o sistema realmente atende aos propósitos do mundo real para o qual foi projetado, em vez de apenas servir aos propósitos para os quais foi treinado. Se essas informações forem fornecidas, o sistema poderá ser implantado.

Tomada de decisão justa e ética conhecer as razões de uma determinada decisão é uma necessidade da sociedade e provavelmente será um direito oficial dos cidadãos da UE (Goodman, Flaxman, 2016). Esse direito à explicação exige que os tomadores de decisão apresentem seus resultados de forma compreensível para perceber a conformidade com os padrões éticos. Cada pessoa afetada por uma decisão automatizada pode fazer uso desse direito à explicação.

Responsabilidade Um objetivo de incorporar explicabilidade no processo de tomada de decisão é tornar um algoritmo responsável por suas ações. Para que um sistema seja responsável, ele deve ser capaz de explicar e justificar suas decisões. Além disso, o problema do deslocamento de dados pode ser direcionado com sistemas interpretáveis, tornando-os mais responsáveis por suas ações (Freitas, 2014).

Ajustes Compreender o modelo de previsão e os fatores subjacentes permite que especialistas de domínio comparem o modelo de previsão com o conhecimento de domínio existente. A explicabilidade é um pré-requisito para a capacidade de ajustar o modelo de previsão incorporando o conhecimento do domínio. De acordo com (Selvaraju et al., 2017), a explicabilidade dos modelos de previsão pode ensinar os humanos, especialmente os especialistas de domínio que usam esses modelos de previsão, como tomar melhores decisões. Além disso, quando observada de um ponto de vista algorítmico, a explicabilidade permite que os projetistas de sistemas façam alterações no modelo de previsão, por exemplo, ajustando parâmetros.

A explicabilidade também é útil para desenvolvedores, pois pode ser usada para identificar modos de falha.

Funcionalidade de proxy quando a explicabilidade é fornecida por um sistema, ela também pode ser examinada com base em outros critérios que não podem ser facilmente quantificados, como segurança, não discriminação, privacidade, robustez, confiabilidade, usabilidade, justiça, verificação e causalidade (Doshi-Velez, Kim, 2017). Nesse caso, a explicabilidade serve como proxy.

3.2.1.2. Modelos explicáveis

Alguns modelos em ML têm a propriedade característica de explicabilidade, ou seja, transparência, facilidade de compreensão e capacidade de questionar. Sendo eles:

1. **Modelos lineares** - Modelos lineares como regressão linear, SVMs com kernel linear, etc seguem o princípio da linearidade de que duas ou mais variáveis podem ser somadas para que sua soma também seja uma solução. Por exemplo, y = mx + c.

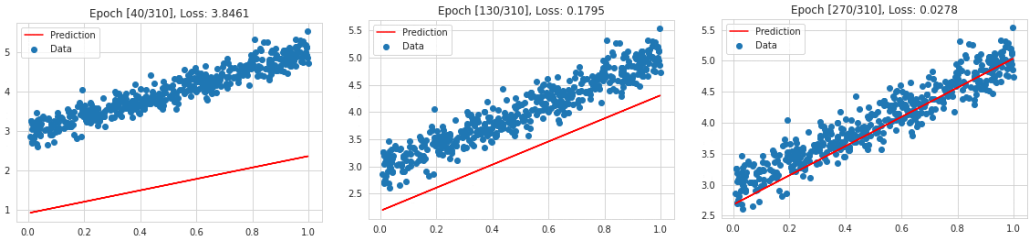


Figura 3.1. Modelo linear [Autor].

2. **Algoritmos de Árvore de Decisão** - Os modelos que usam árvores de decisão são treinados aprendendo regras de decisão simples obtidas de dados anteriores. Como eles seguem um conjunto específico de regras, entender o resultado depende simplesmente de aprender e entender as regras que levaram ao resultado.

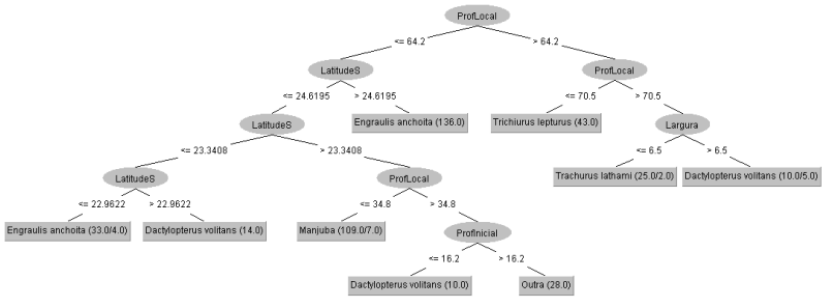


Figura 3.2. Algoritmos de Árvore de Decisão (Costa, 2019).

3. **Modelos Aditivos Generalizados (GAM)** - GAMs são modelos onde a relação usual entre variáveis preditivas e variável dependente (resposta) é substituída por funções suaves lineares e não lineares para modelar e capturar a não linearidade nos dados. GAMs são modelos lineares generalizados com função de suavização devido à sua natureza viciante, cada variável contribui para a saída. Assim, podemos explicar a saída de um GAM simplesmente entendendo as variáveis preditivas.

O problema com os modelos mais explicáveis é que eles na maioria das vezes não capturam a complexidade de alguns problemas do mundo real e podem ser inadequados. Além disso, porque um modelo é simples ou linear, isso não garante explicabilidade. Redes neurais ou modelos de conjunto, etc, são modelos complexos. Assim, para modelos complexos, usamos técnicas e ferramentas para torná-los explicáveis. Existem duas abordagens principais:

Abordagem Modelo-Agnóstica Técnicas/ferramentas independentes de modelo podem ser usadas em qualquer modelo de aprendizado de máquina, não importa o quão complicado seja. Esses métodos agnósticos geralmente funcionam analisando os pares de entrada e saída de recursos. Um bom exemplo é o LIME.

Abordagem Específica do Modelo As técnicas/ferramentas específicas do modelo são específicas para um único tipo de modelo ou um grupo de modelos. Eles dependem da natureza e das funções do modelo específico, por exemplo, intérpretes de árvore.

Existem inúmeras técnicas explicabilidade em ML, sendo elas: Parcelas de Dependência Parcial (PDP), Gráficos de expectativas de condição individual (ICE), Deixar uma coluna de fora (LOCO), Efeitos Locais Acumulados (ALE), Explicações agnósticas do modelo interpretável local (LIME), Âncoras, SHapley Additive exPlanations (SHAP), Recursos importantes de aprendizado profundo (DeepLIFT), Propagação de relevância em camadas (LRP), Método de Explicações Contrastivas (CEM) entre outros.

3.2.2. Auditability

Ao mesmo tempo que a Inteligência Artificial traz inúmeras facilidades às tarefas humanas, ela também acarreta desafios referentes à segurança, robustez e confiabilidade. Uma das abordagens relevantes para enfrentar esses desafios é utilizar mecanismos que permitam auditar sistemas de IA, como ferramentas, padrões e estratégias de avaliação (Berghoff et al., 2021).

As complexas interligações entre os componentes dos sistemas, os diversos fluxos de dados e muitas vezes, o grande número de pessoas envolvidas no processo de treinamento, implantação e manutenção dos modelos, faz com que seja necessário a utilização de alguma forma de auditoria para garantir a integridade das operações (Senft, Gallegos, 2008). Auditoria é basicamente uma análise imparcial que compara o produto desejado e o produto obtido, cujo objetivo é averiguar se o planejamento desse produto foi realizado e como foi desempenhado (Araújo, 2001).

A auditoria fornece garantia às várias partes interessadas de que o objeto alvo da auditoria está de acordo com sua missão e objetivo. Em Aprendizado de Máquina, a

auditabilidade fornece uma maior transparência aos modelos, enquanto que a explicabilidade interpreta os resultados obtidos (Toreini et al., 2020). Dessa forma, as duas técnicas contribuem para que os usuários deixem de enxergar as soluções de ML como uma caixa-preta misteriosa.

Mais especificamente, a auditabilidade avalia a influência dos dados de entrada na saída do modelo. Em outras palavras, auditabilidade diz respeito a qualquer informação que possa ser utilizada para descrever os dados e processos, como por exemplo: a natureza dos dados, o que inclui sua origem, destino e as dependências associadas a eles; as etapas de processamento; e outras informações contextuais, como aspectos de proteção de dados, configuração do sistema, ações da equipe, etc (Singh et al., 2018).

Auditar a IA não é muito diferente de auditar qualquer outra tecnologia emergente (e.g. computação em nuvem), exceto que a IA tem o potencial de impactar desproporcionalmente grupos já marginalizados, com por exemplo, o viés inerente aos conjuntos de dados utilizados no treinamento de novos modelos. Uma das ferramentas que pode ser utilizada no processo de auditabilidade é o *framework* proposto por Hummer et. al. chamado de ModelOps, que visa fornecer as ferramentas necessárias para o *deploy*, governança e monitoramento de modelos de IA (Hummer et al., 2019).

Embora a auditabilidade não seja um conceito novo, ela possui um potencial enorme por ser capaz de permitir que as soluções de IA sejam vistas como confiáveis pelos usuários. Assim, é importante desenvolver técnicas e ferramentas que visem melhorar tal área, pois só com modelos mais confiáveis de IA são a chave para o crescimento dessa tecnologia.

3.3. *Local Interpretable Model-Agnostic Explanations* (LIME)

O *Local Interpretable Model-Agnostic Explanations* (LIME) foi desenvolvido por pesquisadores da Universidade de Washington para ver o que acontece dentro de um algoritmo capturando interações de recursos. O LIME é uma estrutura mais geral que visa tornar as previsões de "qualquer" modelo de aprendizado de máquina mais interpretável (Ribeiro et al., 2016a).

O LIME explica as previsões do modelo no nível da amostra de dados. Ele permite que os usuários finais interpretem essas previsões e executem ações com base nelas, como mostra a Figura 3.3.

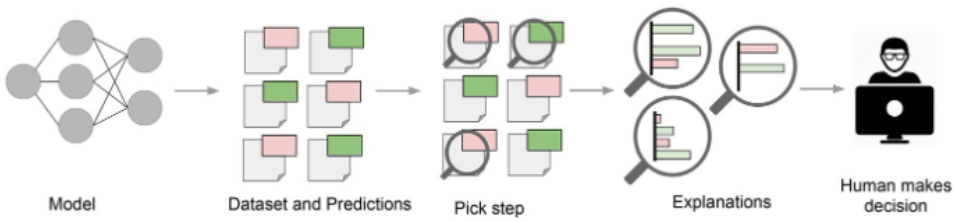


Figura 3.3. Explicando um modelo para um decisor humano (Ribeiro et al., 2016b).

O LIME é independente de modelo, o que significa que pode ser aplicado a qualquer modelo de aprendizado de máquina. A técnica tenta entender o modelo perturbando

a entrada de amostras de dados e entendendo como as previsões mudam. O LIME executa várias perturbações de vários recursos em torno de uma previsão específica e mede os resultados, ele também lida com entradas irregulares. Isso acaba sendo um benefício em termos de interpretabilidade, porque pode-se perturbar a entrada alterando componentes que fazem sentido para humanos, por exemplo, palavras ou partes de uma imagem, mesmo que o modelo esteja usando componentes muito mais complicados como recursos (ex., incorporação de palavras) (Ribeiro et al., 2016a).

O LIME assume um modelo de aprendizado de máquina de caixa preta e investiga a relação entre entrada e saída, representada pelo modelo, como pode-se observar na Figura 3.4.

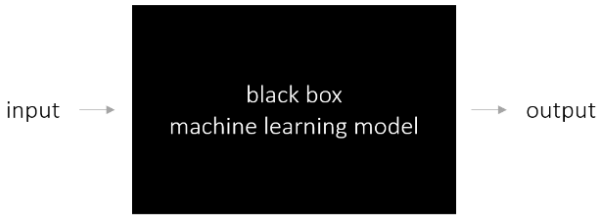


Figura 3.4. Modelo de aprendizado de máquina de caixa preta [Autor].

As abordagens específicas do modelo visam entender o modelo de aprendizado de máquina do modelo preto analisando os componentes internos e como eles interagem. Em modelos de aprendizado profundo, é possível, por exemplo, investigar unidades de ativação e vincular ativações internas de volta à entrada. Isso requer uma compreensão completa da rede e não se adapta a outros modelos (Hulstaert, 2018).

O LIME fornece interpretabilidade do modelo local. O LIME modifica uma única amostra de dados ajustando os valores do recurso e observa o impacto resultante na saída. Muitas vezes, isso também está relacionado ao que os humanos estão interessados ao observar a saída de um modelo. A pergunta mais comum é provavelmente: por que essa previsão foi feita ou quais variáveis causaram a previsão? (Hulstaert, 2018).

Outras técnicas de interpretabilidade do modelo apenas respondem à pergunta acima da perspectiva de todo o conjunto de dados. A importância do recurso explica em um nível de conjunto de dados quais recursos são importantes. Ele permite verificar hipóteses e se o modelo está superajustado ao ruído, mas é difícil diagnosticar previsões específicas do modelo (Hulstaert, 2018). O LIME tenta desempenhar o papel de "explicador", explicando as previsões para cada amostra de dados, como mostra a Figura 3.5.

3.3.1. Intuição por trás do LIME

Um requisito fundamental para a LIME é trabalhar com uma representação interpretável do input, que seja compreensível para os humanos. Exemplos de representações interpretáveis são, por exemplo, um vector BoW para PNL, ou uma imagem para visão por computador. Por outro lado, se o LIME trabalhar com incrustações densas ou não interpretáveis, provavelmente não melhorará a interpretabilidade (Ribeiro, 2021).

O output do LIME é uma lista de explicações, refletindo a contribuição de cada

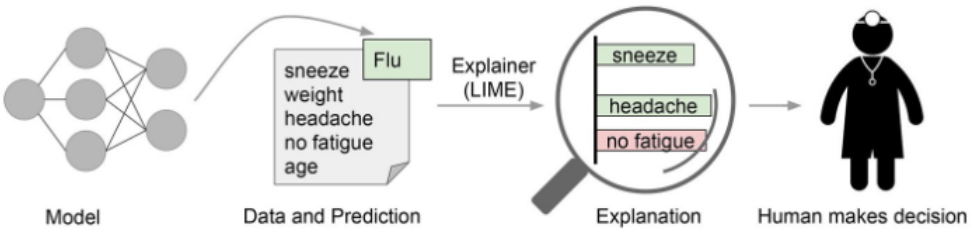


Figura 3.5. Explicando previsões individuais para um decisor humano (Ribeiro et al., 2016b).

recurso para a previsão de uma amostra de dados. Isso fornece interpretabilidade local e também permite determinar quais alterações de recursos terão mais impacto na prediction.output do LIME, é uma lista de explicações, refletindo a contribuição de cada recurso para a previsão de uma amostra de dados. Isso fornece interpretabilidade local e também permite determinar quais alterações de recursos terão mais impacto na previsão (Ribeiro, 2021).

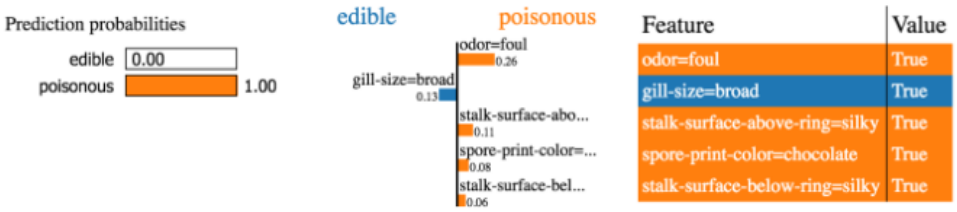


Figura 3.6. Um exemplo de LIME aplicado a um problema clássico de classificação (Ribeiro et al., 2016c).

Uma explicação é criada aproximando o modelo subjacente localmente por um modelo interpretável. Modelos interpretáveis são, por exemplo, modelos lineares com forte regularização, árvores de decisão, etc. Os modelos interpretáveis são treinados em pequenas perturbações da instância original e devem fornecer apenas uma boa aproximação local. O "conjunto de dados" é criado, por exemplo, adicionando ruído a recursos contínuos, removendo palavras ou ocultando partes da imagem. Ao aproximar apenas a caixa preta localmente (na vizinhança da amostra de dados), a tarefa é significativamente simplificada (Ribeiro, 2021).

3.3.2. Implementações de LIME

Usa-se o LIME para explicar uma infinidade de classificadores, como florestas aleatórias, máquinas de vetor de suporte (SVM) e redes neurais, nos domínios de texto e imagem. Implementações dessas técnicas tornaram-se disponíveis recentemente à medida que bibliotecas de programação específicas foram desenvolvidas (Ribeiro et al., 2016a; Alber et al., 2019).

Para explicações de imagem, uma determinada imagem é segmentada algoritmicamente em superpixels, e a relevância de cada superpixel para uma determinada classificação é determinada usando um modelo linear. O algoritmo é independente de modelo, pois requer apenas as saídas do classificador para imagens diferentes. De fato,

o LIME pode ser usado para qualquer sistema de classificação de imagens, não apenas redes neurais, pois não emprega procedimentos específicos de retropropagação ou etapas específicas para qualquer tipo de modelo individual (Ribeiro et al., 2016b).

A Figura 3.7 mostra um exemplo de como o LIME funciona para classificação de imagens. Imagine que queremos explicar um classificador que prevê a probabilidade de a imagem conter um sapo. Pegamos a imagem à esquerda e a dividimos em componentes interpretáveis, ou seja, superpixels contíguos (Ribeiro et al., 2016b).

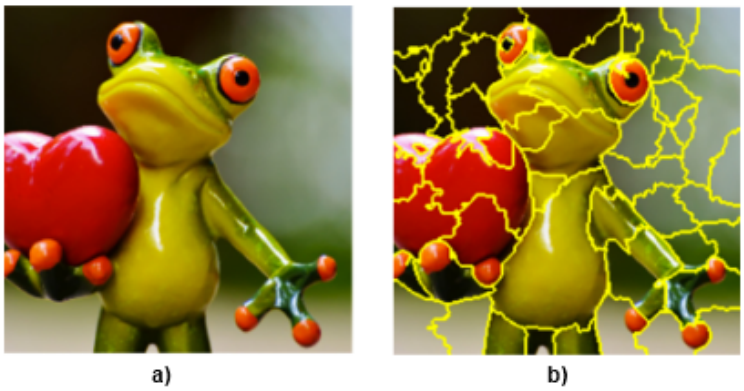


Figura 3.7. Transformando uma imagem em componentes interpretáveis. a) Imagem Original. b) Componentes Interpretáveis (Ribeiro et al., 2016b).

Gera-se um conjunto de dados de instâncias perturbadas desligando alguns dos componentes interpretáveis, neste caso, tornando-os cinza, como mostra a Figura 3.8. Para cada instância perturbada, obtem-se a probabilidade de que uma perereca esteja na imagem de acordo com o modelo. Em seguida, aprendem-se um modelo simples, linear, nesse conjunto de dados, que é ponderado localmente, ou seja, preocupa-se mais em cometer erros em instâncias perturbadas que são mais semelhantes à imagem original. No final, apresenta-se os superpixels com maiores pesos positivos como explicação, esmaecendo todo o resto (Ribeiro et al., 2016b).

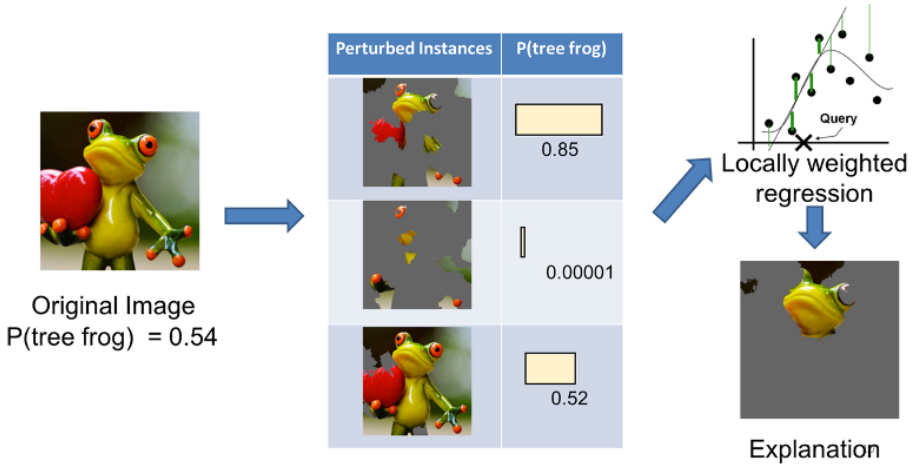


Figura 3.8. Previsão com LIME (Ribeiro et al., 2016b).

No contexto de classificação de texto, isso significa que algumas das palavras

são, por exemplo, substituídas, para determinar quais elementos da entrada afetam as previsões.

3.3.3. Desvantagens do LIME

Embora a ideia geral do LIME pareça fácil, existem algumas desvantagens em potencial. Na implementação atual, apenas modelos lineares são usados para aproximar o comportamento local. Até certo ponto, essa suposição está correta quando se observa uma região muito pequena ao redor da amostra de dados. Ao expandir essa região, no entanto, é possível que um modelo linear não seja poderoso o suficiente para explicar o comportamento do modelo original. A não linearidade em regiões locais acontece para os conjuntos de dados que exigem modelos complexos e não interpretáveis. Não ser capaz de aplicar o LIME nestes cenários é uma armadilha significativa (Hulstaert, 2018).

Em segundo lugar, o tipo de modificações que precisam ser realizadas nos dados para obter as explicações adequadas geralmente são específicos do caso de uso. Segundo Ribeiro et al. (2016a) um modelo que prevê imagens com tom sépia a serem retro não pode ser explicado pela presença ou ausência de super pixels.

Muitas vezes, simples perturbações não são suficientes. Idealmente, as perturbações seriam impulsionadas pela variação que é observada no conjunto de dados. Dirigir manualmente as perturbações do outro provavelmente não é uma ótima ideia, pois provavelmente introduziria viés nas explicações do modelo (Hulstaert, 2018).

3.4. *SHapley Additive exPlanations* (SHAP)

O SHAP é um método desenvolvido por Lundberg e Lee, em 2017, para explicar predições individuais, baseando-se nos valores de Shapley da teoria dos jogos. O método tenta interpretar de maneira mais direta as decisões dos modelos de inteligência artificial (IA), diferente das soluções caixa preta. Para isso, SHAP explica uma saída específica calculando a contribuição de cada característica (atributo) da entrada para predição (instância) (Burkart, Huber, 2021). Assim, SHAP indica as influências negativas ou positivas das características para um resultado do modelo. A Figura 3.9 ilustra o método, onde os atributos idade, gênero, pressão arterial e índice de massa corporal (IMC) têm valores de contribuições +0.4, -0.3, +0.1 e +0.1 para a saída apresentada.

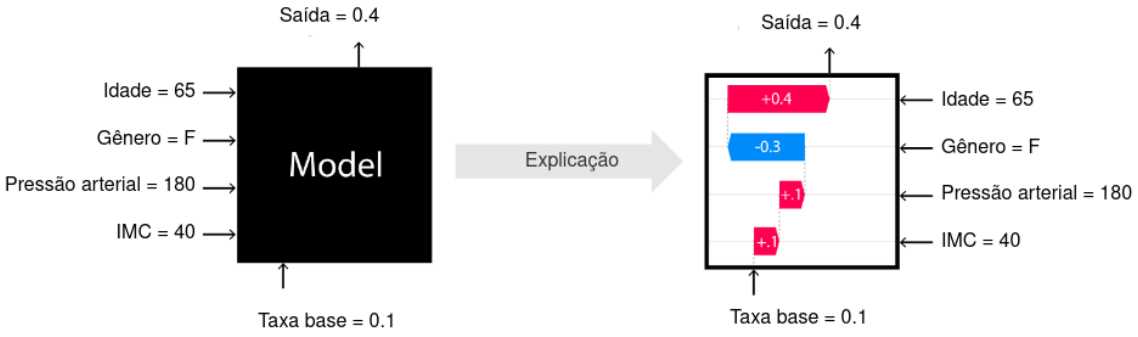


Figura 3.9. Ilustração da explicação com SHAP. Adaptação de Mitchell et al. (2020).

3.4.1. Base do SHAP: valores de Shapley

Para entender SHAP, é interessante observar como funciona o cálculo dos valores de Shapley. Esse método determina a contribuição de uma característica verificando a diferença entre a predição de um modelo com e sem essa característica. Como a ordem que o modelo analisa as características pode variar as contribuições dos atributos, faz-se necessário verificar todas as combinações (coalizões) possíveis. Então, um valor de Shapley é a contribuição marginal média de um atributo de uma instância entre todas as coalizões possíveis (López, 2021).

Na Figura 3.10, mostramos um exemplo prático do cálculo do valor de Shapley para o atributo A1. Nesse exemplo, organizamos seis coalizões e agrupamos essas coalizões sempre variando a presença de A1. Com a diferença das coalizões, obtemos a contribuição marginal (δ) de cada grupo. Por fim, calculamos a contribuição marginal média (ϕ), o valor de Shapley de A1.

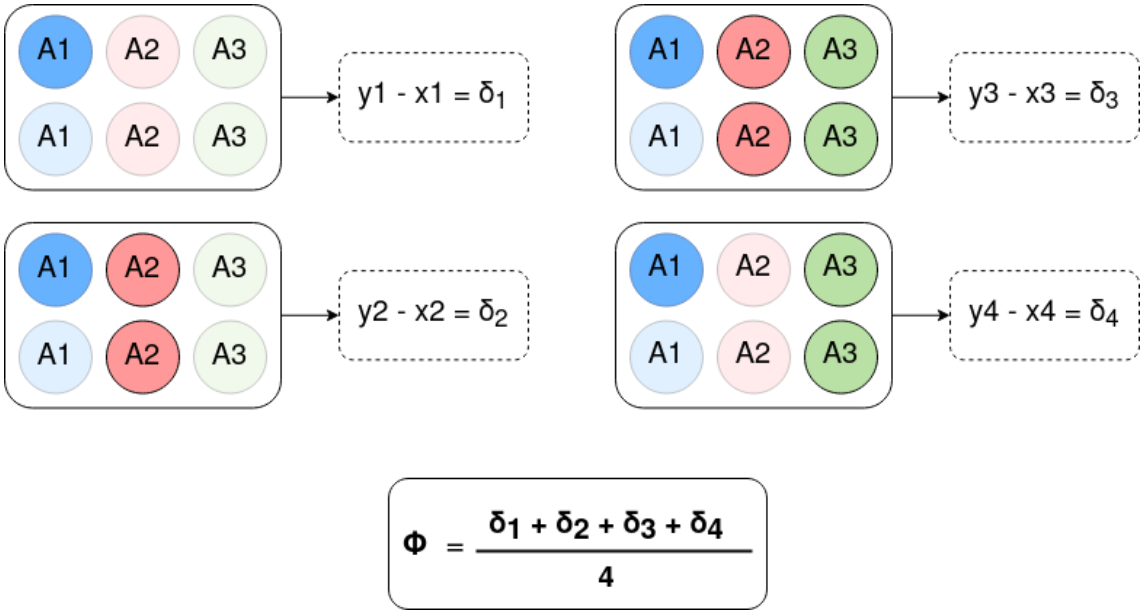


Figura 3.10. Exemplo: obtenção do valor de Shapley para o atributo A1.

A desvantagem dessa abordagem é calcular o valor exato de cada atributo, pois seu custo cresce exponencialmente de acordo com a quantidade de atributos (Bhatt et al., 2019). Uma das maneiras de otimizar esse cálculo é utilizando o SHAP.

3.4.2. Funcionamento do SHAP

O intuito do SHAP é explicar uma previsão feita a partir de uma instância, descobrindo o valor de Shapley de cada característica que resultou na predição. Essas características podem ser, por exemplo, um valor individual para dados tabulares ou um grupo de valores de característica para uma imagem. Além disso, SHAP cumpre três propriedades, já presente no método clássico de valores de Shapley, desejáveis em métodos de atribuição aditiva de atributos (Lundberg, Lee, 2017):

- Precisão local: determina a mesma saída para o modelo aproximado e o modelo

original;

- Ausência: garante nenhum impacto na saída para recursos ausentes;
- Consistência: afirma que, se um recurso do modelo aumenta ou permanece o mesmo independentemente dos outros recursos, o impacto desse recurso não deve diminuir.

Como vimos anteriormente, encontrar os valores de Shapley para um grande número de atributos pode ser um trabalho oneroso. Para evitar isso, Lundberg e Lee propuseram o Kernel SHAP, um método adaptado do LIME linear para calcular os valores de Shapley (Lundberg, Lee, 2017). O Kernel SAHP consegue obter esses valores com algumas amostras de coalizão, utilizando dados de treinamento do modelo e uma regressão linear ponderada onde os coeficientes da solução são os valores de Shapley.

Para cada amostra de coalizão, adquire-se sua predição e calculam-se os pesos W_C com a forma do *kernel* descrita na equação 1, onde C é índice da coalizão (López, 2021). Com essa equação, obtemos os pesos de uma regressão linear, os valores de Shapley.

$$W_C = \frac{N^o \text{ de atributos}}{\binom{N^o \text{ de coalizões}}{\text{do tamanho de } C} \times \left(N^o \text{ de atributos de } C\right) \times \binom{N^o \text{ de atributos}}{\text{ausentes em } C}} \tag{1}$$

Com esses pesos, entendemos a influência dos atributos. Quanto maior seu valor absoluto, mais importantes para predição. Para ter uma visão geral global dos resultados, calcula-se a média dos valores absolutos de Shapley.

Por fim, é importante acrescentar que o Kernel SHAP é agnóstico para o cálculo dos valores de Shapley, podendo ser aplicado a qualquer modelo de Aprendizado de Máquina. Contudo, existem outras versões de SHAP que possuem otimizações específicas para certos modelos, como os métodos Tree SHAP, Deep SHAP, Low-Order SHAP, Linear SHAP e Max SHAP (López, 2021).

3.5. Outras técnicas XAI

Atualmente, existem inúmeras técnicas para explicabilidade em Machine Learning, como: Anchors (Ribeiro et al., 2018), Partial Dependence Plots (PDP), Individual Condition Expectations plots (ICE) (Goldstein et al., 2015), Leave One Covariate Out (LOCO) (Molnar et al., 2020), Accumulated Local Effects (ALE) (Galkin et al., 2018), Deep Learning Important Features (DeepLIFT) (Shrikumar et al., 2017), Layer-wise relevance propagation (LRP) (Montavon et al., 2019), Contrastive Explanations Method (CEM) (Dhurandhar et al., 2018a), ProfWeight (Dhurandhar et al., 2018b), Permutation Feature Importance, entre outras. A seguir, uma visão geral de algumas dessas técnicas.

3.5.1. Anchors

A abordagem anchors foi construída pelos mesmos criadores do LIME. O método anchors explica as previsões individuais de um modelo usando regras IF-THEN facilmente compreensíveis - ”âncoras- que suportam (ancoram) as previsões suficientemente bem. Para

encontrar âncoras, os autores usam técnicas de reforço em combinação com um algoritmo de busca em grafo para explorar os conjuntos de perturbações em torno dos dados e seus efeitos nas previsões. Este método é agnóstico de modelo.

No artigo original (Ribeiro et al., 2018), os autores compararam o LIME com o Anchors e visualizaram como eles processam um modelo de classificador binário complexo (+ ou –) para chegar a um resultado. Conforme a Figura 3.11, a explicação do LIME funciona aprendendo um limite de decisão linear que melhor se aproxima do modelo, com alguma ponderação local, enquanto Âncoras adapta sua cobertura ao comportamento do modelo e deixa seus limites claros.

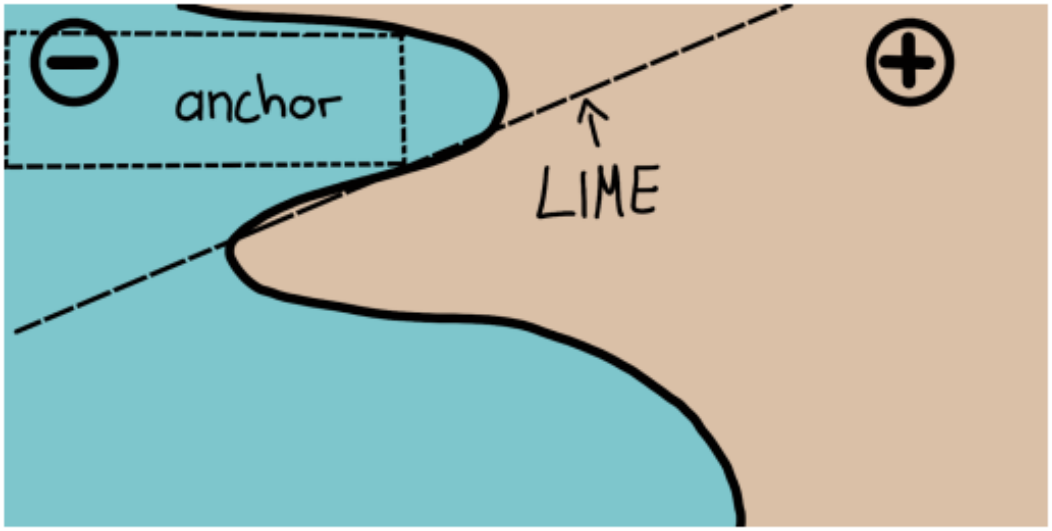


Figura 3.11. Comparação do método Anchors com o método LIME (Ribeiro et al., 2016c).

As âncoras também foram testadas em uma variedade de tarefas de Machine Learning, como classificação, geração de texto e previsões estruturadas.

3.5.2. Partial Dependence Plots (PDP)

O PDP obtém uma representação visual global de como um ou dois recursos influenciam o resultado previsto do modelo, com outros recursos mantidos constantes. O PDP informa se a relação entre o destino e o recurso escolhido é linear ou complexo. O PDP é independente de modelo.

3.5.3. Individual Conditional Expectations plots (ICE)

O ICE fornece uma representação visual local do efeito de um recurso no modelo em relação ao recurso de destino. Ao contrário do PDP, o ICE mostra previsões separadas da dependência do recurso com uma linha por amostra. Também é agnóstico de modelo (Goldstein et al., 2015).

3.5.4. Leave One Covariate Out (LOCO)

O Leave One Covariate Out é uma abordagem muito simplista. O LOCO deixa uma coluna de fora, treina novamente o modelo e, em seguida, calcula as diferenças de cada

modelo LOCO para a pontuação de previsão do modelo original. Se a pontuação mudar muito, a variável que ficou de fora deve ser importante. Dependendo da largura do modelo (quantidade de recursos), essa abordagem pode ser demorada (Molnar et al., 2020).

Existem algumas desvantagens que PDP, ICE e LOCO compartilham:

- Capturam diretamente as interações dos recursos;
- Podem ser muito aproximados, o que é potencialmente problemático para dados categóricos e codificação one-hot que é frequentemente usada no processamento de linguagem natural.

3.5.5. Layer-wise relevance propagation (LRP)

A propagação de relevância em camadas é semelhante ao DeepLIFT, ele faz a propagação para trás usando um conjunto de regras de propagação projetadas propositadamente a partir da saída, identificando os neurônios mais relevantes dentro da rede neural até você retornar à entrada. Assim, você obtém todos os neurônios (por exemplo, pixels que realmente contribuem para a saída. LRP funciona bem em CNNs e pode ser usado para explicar LSTMs (Montavon et al., 2019).

Confira esta demonstração interativa para ver como o LRP funciona.

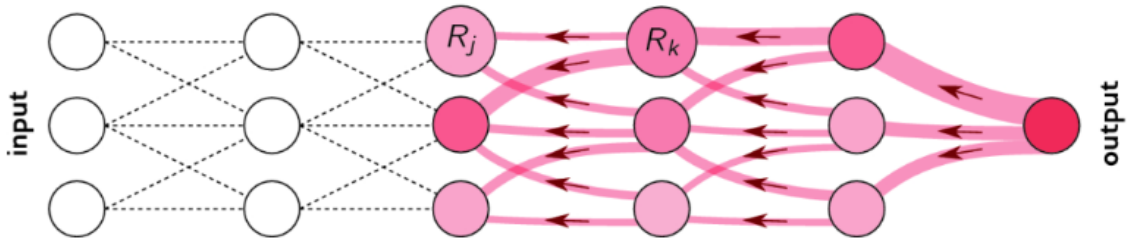


Figura 3.12. Representação visual de como o LRP faz retropropagação do nó de saída através dos neurônios da camada oculta até a entrada, identificando os neurônios que tiveram impacto na saída do modelo (Montavon et al., 2019).

3.5.6. ProfWeight

Em 2018, (Dhurandhar et al., 2018b) fez um artigo "Improving Simple Models with Confidence Profiles". O artigo propôs o método ProfWeight para explicabilidade do modelo. ProfWeight transfere a alta precisão de teste de uma rede neural profunda pré-treinada para uma rede rasa com baixa precisão de teste, como é visto na Figura 3.13.

Assim como um professor transferindo conhecimento para um aluno, ProfWeight utiliza sondas (pesos na amostra de acordo com a dificuldade da rede) para transferir conhecimento.

ProfWeight pode ser resumido em quatro etapas principais:

1. Anexe e treine sondas em representações intermediárias de uma rede neural de alto desempenho;

2. Treine um modelo simples no conjunto de dados original;

3. Aprenda pesos para exemplos no conjunto de dados em função do modelo simples e das sondas;

4. Treine novamente o modelo simples no conjunto de dados ponderado final.

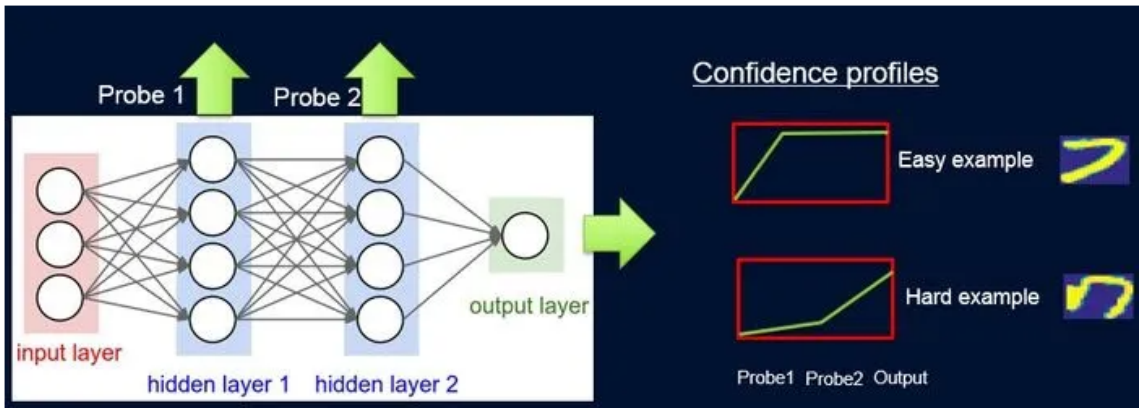


Figura 3.13. Representação visual do ProfWeight (Dhurandhar et al., 2018b).

3.6. Ferramentas e práticas

Nesta seção serão apresentadas as principais abordagens e trechos de código-fonte necessários para algumas implementações de técnicas XAI. Os trechos de código foram baseados nos algoritmos apresentados em [1] e [2]. Os autores criaram um *notebook* público no ambiente Google Colaboratory com todos os códigos abaixo exibidos, além de outros não listados neste documento. Para a execução, basta executar as células em sequência. A permissão do *notebook* é de visualização, mas podem ser realizadas cópias e, a partir de então, essas cópias poderão ser editadas e modificadas. Os autores recomendam que façam esse procedimento, pois é de suma importância que realizem diversas modificações nos códigos para compreender como que os algoritmos atuam à medida que trocamos parâmetros, valores *etc*. O link para o notebook está em [3].

3.6.1. Partial Dependence Plots - PDP

Gráficos de dependência parcial (*Partial Dependence Plots* - PDP) e gráficos de expectativa condicional individual (*Individual Conditional Expectation* - ICE) são imagens que podem ser utilizadas para facilitar a compreensão dos dados e analisar a interação entre a resposta alvo e um conjunto de recursos de entrada de interesse.

[1] https://scikit-learn.org/stable/modules/partial_dependence.html

[2] https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html

[3] https://colab.research.google.com/drive/1Fh8xmXzqMRR6aDR5lWLztC63VCuFH1t6?usp=sharing

Dessa forma, pode-se dizer que o PDP é uma técnica que fornece uma representação visual global de como um ou dois recursos influenciam o resultado previsto do modelo. A técnica também informa se a relação entre o destino e o recurso escolhido é linear ou complexo.

Os módulos de inspeção de aprendizado do Scikit fornecem uma função para gráfico de dependência parcial chamada *plot_partial_dependence* que cria um gráfico de dependência parcial unidirecional e bidirecional. No trecho de código a seguir é criado uma grade de gráficos de dependência parcial: dois PDPs unidirecionais para os recursos 0 e 1 e um PDP bidirecional entre os dois recursos. Como base de dados, são utilizados os dados de habitação da Califórnia-EUA. Esse conjunto de dados consiste em 20.640 quarteirões de casas em toda a Califórnia em 1990, e o objetivo é prever o logaritmo natural do preço médio das casas a partir de 8 recursos diferentes:

- MedInc - renda mediana no grupo de blocos
- HouseAge - idade média da casa no grupo de quarteirões
- AveRooms - número médio de quartos por família
- AveBedrms - número médio de quartos por domicílio
- Population - população do grupo de blocos
- AveOccup - número médio de membros da família
- Latitude - latitude do grupo de blocos
- Longitude - longitude do grupo de blocos

O código adiante carrega o banco de dados do conjunto habitacional de California-EUA:

```
import pandas as pd
import shap
import sklearn

# a classic housing price dataset
X,y = shap.datasets.california(n_points=1000)

X100 = shap.utils.sample(X, 100) # 100 instances for use as the
    ↪ background distribution

# a simple linear model
model = sklearn.linear_model.LinearRegression()
model.fit(X, y)
```

Em seguida, podemos examinar alguns coeficientes dos dados. A maneira mais comum de entender um modelo linear é examinar os coeficientes aprendidos para cada característica. Esses coeficientes nos dizem o quanto a saída do modelo muda quando alteramos cada um dos recursos de entrada:

```
print("Model coefficients:\n")
for i in range(X.shape[1]):
    print(X.columns[i], "=", model.coef_[i].round(5))
```

```
OUTPUT:

Model coefficients:

MedInc = 0.45769
HouseAge = 0.01153
AveRooms = -0.12529
AveBedrms = 1.04053
Population = 5e-05
AveOccup = -0.29795
Latitude = -0.41204
Longitude = -0.40125
```

Para entender a importância de um determinado recurso em um modelo, é necessário compreender como a alteração desse recurso afeta a saída do modelo e também a distribuição dos valores desse recurso. Para visualizar isso para um modelo linear, podemos construir um gráfico de dependência parcial clássico e mostrar a distribuição de valores de recursos como um histograma no eixo x. O código adiante faz esse processo:

```
shap.partial_dependence_plot(
    "MedInc", model.predict, X100, ice=False,
    model_expected_value=True, feature_expected_value=True
)
```

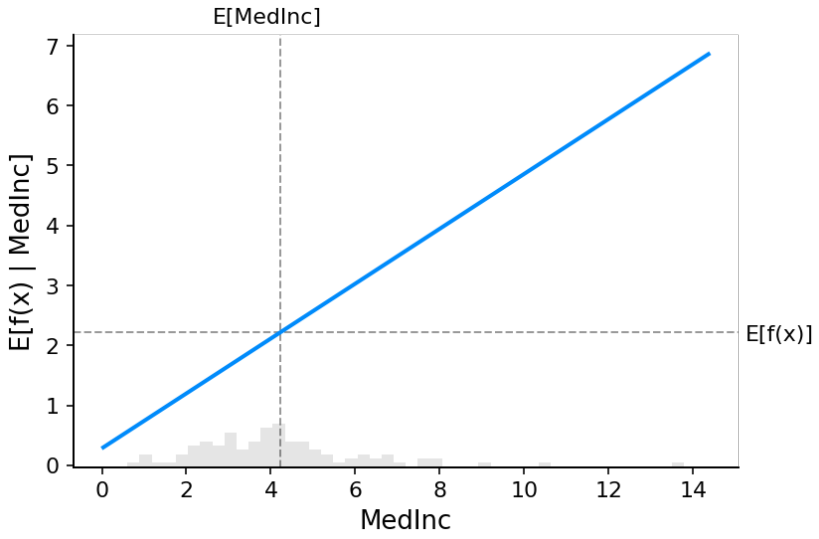


Figura 3.14. PDP do modelo em análise

3.6.2. SHAP

Para manipularmos na prática a técnica SHAP, inicialmente precisamos instalar o SHAP. O modo padrão para realizar tal procedimento no Python é conforme abaixo:

```
pip install shap
```

Uma das propriedades fundamentais dos valores de Shapley é que eles sempre se resumem à diferença entre o resultado do "jogo" quando todos os jogadores estão presentes e o resultado do jogo quando nenhum jogador está presente. Para modelos de aprendizado de máquina, isso significa que os valores SHAP de todos os recursos de entrada sempre somarão a diferença entre a saída do modelo de linha de base (esperada) e a saída do modelo atual para a previsão explicada. A maneira mais fácil de ver isso é por meio de um gráfico em cascata que começa na expectativa anterior de fundo para um preço de casa e, em seguida, adiciona recursos um de cada vez até atingirmos a saída do modelo atual. O trecho de código adiante produz a imagem esperada (Figura 3.15).

```
# the waterfall_plot shows how we get from shap_values.base_values to
    ↪ model.predict(X)[sample_ind]
shap.plots.waterfall(shap_values[sample_ind], max_display=14)
```

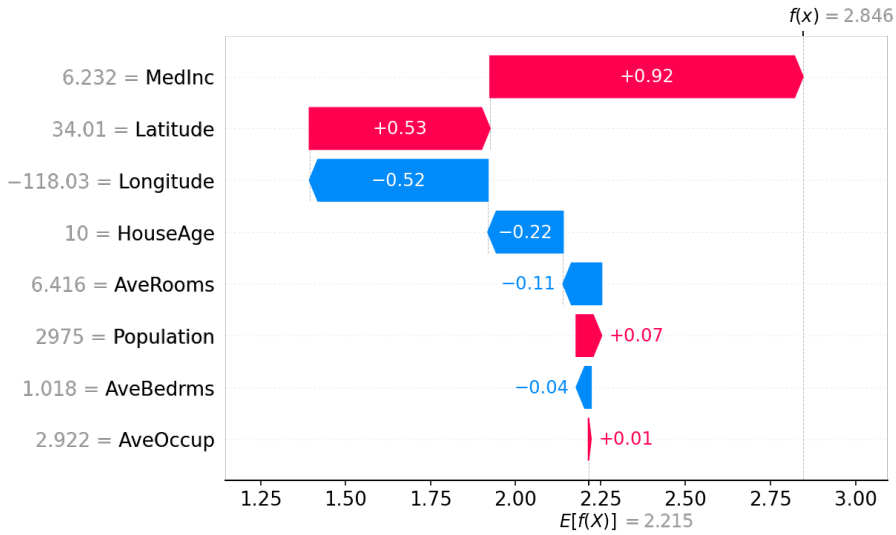


Figura 3.15. SHAP do modelo de casas - Califórnia

A partir de agora, usaremos o conhecido conjunto de dados de espécies Iris para ilustrar como o SHAP pode explicar a saída de muitos tipos de modelos diferentes. O conjunto de dados possui apenas 150 amostras e foram utilizados 130 de um conjunto aleatório para treinamento e 20 para teste dos modelos. Esta prática foi adaptada de [4]. Inicialmente, será realizada a coleta do banco de dados para subsequente preparo.

```
import sklearn
from sklearn.model_selection import train_test_split
import numpy as np
import shap
import time
```

[4] https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/model_agnostic/Iris%20classification%20with%20scikit-learn.html

```
X_train,X_test,Y_train,Y_test = train_test_split(*shap.datasets.iris(),
    ↪  test_size=0.2, random_state=0)

# rather than use the whole training set to estimate expected values,
    ↪ we could summarize with
# a set of weighted kmeans, each weighted by the number of points they
    ↪ represent. But this dataset
# is so small we don't worry about it
#X_train_summary = shap.kmeans(X_train, 50)

def print_accuracy(f):
    print("Accuracy = {0}%".format(100*np.sum(f(X_test) == Y_test)/len(
        ↪ Y_test)))
    time.sleep(0.5) # to let the print get out before any progress bars

shap.initjs()
```

Em seguida, será apresentado o resultado da técnica SHAP para o modelo K-nearest neighbors aplicado aos dados da íris.

```
knn = sklearn.neighbors.KNeighborsClassifier()
knn.fit(X_train, Y_train)

print_accuracy(knn.predict)
```

Na Figura 3.16 é apresentado visualmente o efeito que a previsão do KNN gerou no modelo.

```
explainer = shap.KernelExplainer(knn.predict_proba, X_train)
shap_values = explainer.shap_values(X_test.iloc[0,:])
shap.initjs()
shap.force_plot(explainer.expected_value[0], shap_values[0], X_test.
    ↪ iloc[0,:])
```

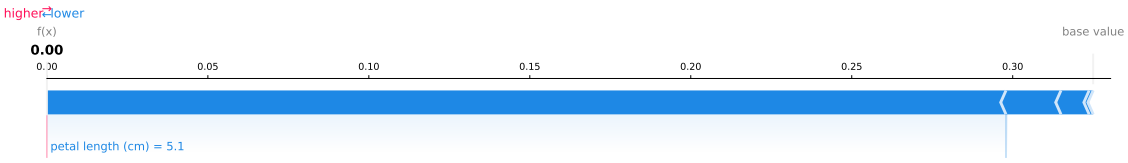


Figura 3.16. SHAP do modelo Íris - explicando uma única previsão do conjunto

Para identificarmos os efeitos combinados dos atributos, com resultados dinâmicos à medida que movemos o mouse sobre a imagem gerada pelo SHAP (imagem obtida diretamente no terminal), podemos fazer uso do método `shap.force_plot`. Na Figura 3.18 é apresentada uma imagem estática do que seria essa interação dinâmica que a técnica proporciona.

```
shap_values = explainer.shap_values(X_test)
shap.initjs()
shap.force_plot(explainer.expected_value[0], shap_values[0], X_test)
```

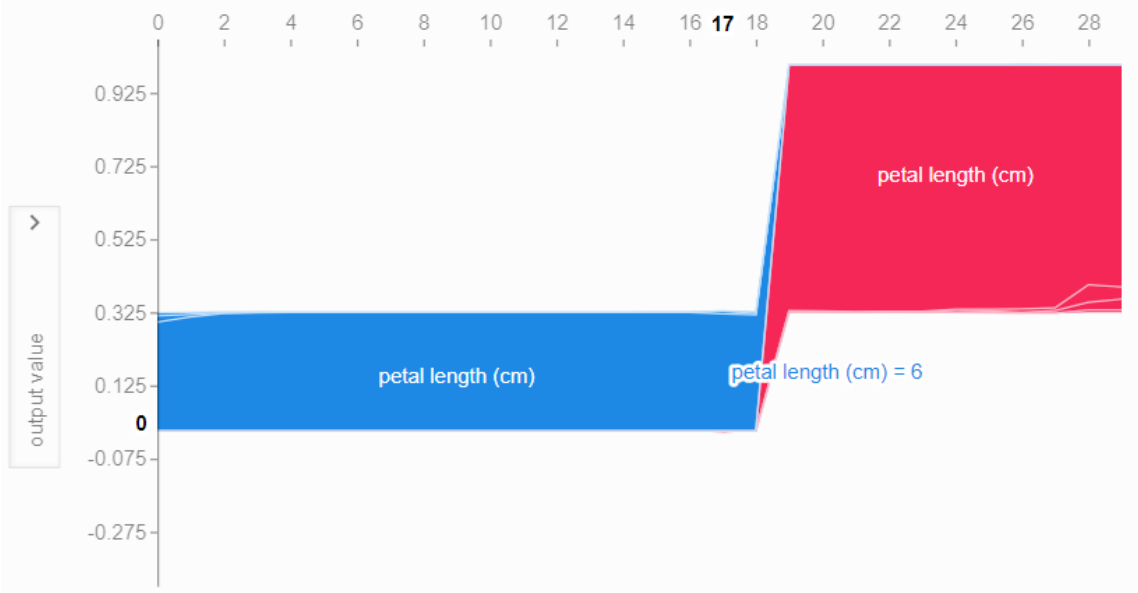


Figura 3.17. SHAP do modelo Íris - KNN

Agora veremos o efeito do modelo Support vector machine com um kernel linear sobre os dados da íris. Para tal, podemos utilizar o seguinte trecho de código:

```
svc_linear = sklearn.svm.SVC(kernel='linear', probability=True)
svc_linear.fit(X_train, Y_train)
print_accuracy(svc_linear.predict)

# explain all the predictions in the test set
explainer = shap.KernelExplainer(svc_linear.predict_proba, X_train)
shap_values = explainer.shap_values(X_test)
shap.force_plot(explainer.expected_value[0], shap_values[0], X_test)
```

3.7. Conclusão

O aprendizado de máquina tem um grande potencial para melhorar produtos, processos e pesquisas, entretanto, os sistemas computacionais por si geralmente não explicam suas previsões, o que é uma barreira para a adoção do aprendizado de máquina por diversos setores. Este minicurso tratou de forma introdutória sobre como tornar os modelos de aprendizado de máquina e suas decisões interpretáveis.

Depois de explorar os conceitos de interpretabilidade, são discutidos modelos simples e interpretáveis, como árvores de decisão, regras de decisão e estratégias para interpretar modelos de caixa preta, como a importância de recursos e efeitos locais acumulados, e explicar previsões individuais com valores de Shapley e LIME.

Referências

Alber Maximilian, Lapuschkin Sebastian, Seegerer Philipp, Hägele Miriam, Schütt Kristof T, Montavon Grégoire, Samek Wojciech, Müller Klaus-Robert, Dähne Sven, Kindermans Pieter-Jan. iNNvestigate neural networks! // J. Mach. Learn. Res. 2019. 20,

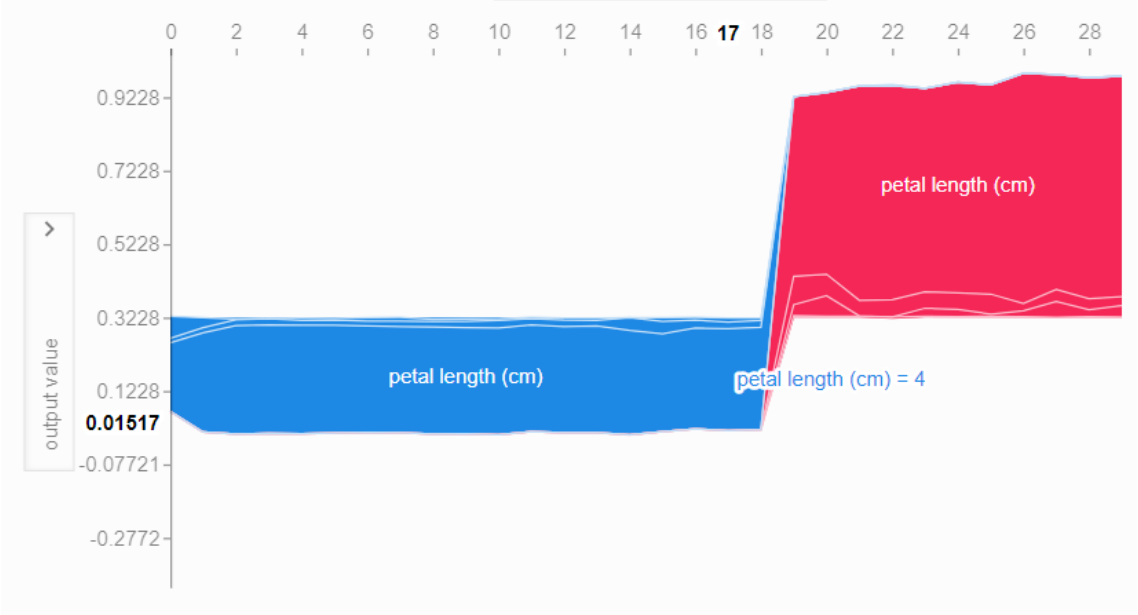


Figura 3.18. SHAP do modelo Íris com SVM

93. 1–8.

Angelov Plamen, Soares Eduardo. Towards explainable deep neural networks (xDNN) // Neural Networks. 2020. 130. 185–194.

Araújo Inaldo da Paixão Santos. Introdução à auditoria operacional. 2001.

Berghoff Christian, Biggio Battista, Brummel Elisa, Danos Vasilios, Doms Thomas, Ehrich Heiko, Gantevoort Thorsten, Hammer Barbara, Iden Joachim, Jacob Sven, Khlaaf Heidy, Komrowski Lars, Kröwing Robert, Metzen Jan, Neu Matthias, Petsch Fabian, Poretschkin Maximilian, Samek Wojciech, Schäbe Hendrik, Twickel Arndt, Vechev Martin, Wiegand Thomas. Towards Auditable AI Systems: Current status and future directions [White paper]. May 2021.

Bhatt Umang, Xiang Alice, Sharma Shubham, Weller Adrian, Taly Ankur, Jia Yunhan, Ghosh Joydeep, Puri Ruchir, Moura José M. F., Eckersley Peter. Explainable Machine Learning in Deployment. 2019.

Burkart Nadia, Huber Marco F. A survey on the explainability of supervised machine learning // Journal of Artificial Intelligence Research. 2021. 70. 245–317.

Costa Lucas Tubino Bonifácio Costa. Análise de modelos explicáveis de sistemas de classificação para registros hidroacústicos em ambientes marítimos. 2019.

Dhurandhar Amit, Chen Pin-Yu, Luss Ronny, Tu Chun-Chen, Ting Paishun, Shanmugam Karthikeyan, Das Payel. Explanations based on the missing: Towards contrastive explanations with pertinent negatives // Advances in neural information processing systems. 2018a. 31.

Dhurandhar Amit, Shanmugam Karthikeyan, Luss Ronny, Olsen Peder A. Improving simple models with confidence profiles // Advances in Neural Information Processing Systems. 2018b. 31.

Doshi-Velez Finale, Kim Been. Towards a rigorous science of interpretable machine learning // arXiv preprint arXiv:1702.08608. 2017.

Europa.eu . Official journal of the european union:Regulations. 2017.

Freitas Alex A. Comprehensible classification models: a position paper // ACM SIGKDD explorations newsletter. 2014. 15, 1. 1–10.

Galkin Fedor, Aliper Aleksandr, Putin Evgeny, Kuznetsov Igor, Gladyshev Vadim N, Zhavoronkov Alex. Human microbiome aging clocks based on deep learning and tandem of permutation feature importance and accumulated local effects // BioRxiv. 2018. 507780.

Goldstein Alex, Kapelner Adam, Bleich Justin, Pitkin Emil. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation // journal of Computational and Graphical Statistics. 2015. 24, 1. 44–65.

Goodman Bryce, Flaxman Seth. Eu regulations on algorithmic decision-making and a" right to explanation" // arXiv preprint arxiv:1606.08813. 2016.

Goodman Bryce, Flaxman Seth. European Union regulations on algorithmic decision-making and a "right to explanation" // AI magazine. 2017. 38, 3. 50–57.

Gunning David. Explainable artificial intelligence (XAI), Defense Advanced Research Projects Agency (DARPA), 2017. 2017.

Holzinger Andreas, Langs Georg, Denk Helmut, Zatloukal Kurt, Müller Heimo. Causability and explainability of artificial intelligence in medicine // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2019. 9, 4. e1312.

Hulstaert Lars. Understanding model predictions with LIME. VII 2018.

Hummer Waldemar, Muthusamy Vinod, Rausch Thomas, Dube Parijat, El Maghraoui Kaoutar, Murthi Anupama, Oum Punleuk. Modelops: Cloud-based lifecycle management for reliable and trusted ai // 2019 IEEE International Conference on Cloud Engineering (IC2E). 2019. 113–120.

Lipton Zachary C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. // Queue. 2018. 16, 3. 31–57.

Lundberg Scott M., Lee Su-In. A Unified Approach to Interpreting Model Predictions // Proceedings of the 31st International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc., 2017. 4768–4777. (NIPS'17).

López Fernando. SHAP: Shapley Additive Explanations. 2021. Online; acessado em Jul 2022.

Martens David, Baesens BB, Van Gestel Tony. Decompositional rule extraction from support vector machines by active learning // IEEE Transactions on Knowledge and Data Engineering. 2008. 21, 2. 178–191.

Mitchell Rory, Frank Eibe, Holmes Geoffrey. GPUTreeShap: Massively Parallel Exact Calculation of SHAP Scores for Tree Ensembles. 2020.

Molnar C, Gruber S, Kopper P. Limitations of interpretable machine learning methods. 2020.

Molnar Christoph. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. 2020.

Montavon Grégoire, Binder Alexander, Lapuschkin Sebastian, Samek Wojciech, Müller Klaus-Robert. Layer-wise relevance propagation: an overview // Explainable AI: interpreting, explaining and visualizing deep learning. 2019. 193–209.

Ribeiro Marco Tulio. LIME. VII 2021.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. "Why should i trust you?"Explaining the predictions of any classifier // Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016a. 1135–1144.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. Local Interpretable Model-Agnostic Explanations (LIME): An Introduction. VIII 2016b.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. Model-agnostic interpretability of machine learning // arXiv preprint arXiv:1606.05386. 2016c.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. Anchors: High-precision model-agnostic explanations // Proceedings of the AAAI conference on artificial intelligence. 32, 1. 2018.

Selvaraju Ramprasaath R, Cogswell Michael, Das Abhishek, Vedantam Ramakrishna, Parikh Devi, Batra Dhruv. Grad-cam: Visual explanations from deep networks via gradient-based localization // Proceedings of the IEEE international conference on computer vision. 2017. 618–626.

Senft Sandra, Gallegos Frederick. Information technology control and audit. 2008.

Shrikumar Avanti, Greenside Peyton, Kundaje Anshul. Learning important features through propagating activation differences // International conference on machine learning. 2017. 3145–3153.

Singh Jatinder, Cobbe Jennifer, Norval Chris. Decision provenance: Harnessing data flow for accountable systems // IEEE Access. 2018. 7. 6562–6574.

Toreini Ehsan, Aitken Mhairi, Coopamootoo Kovila, Elliott Karen, Zelaya Carlos Gonzalez, Van Moorsel Aad. The relationship between trust in AI and trustworthy machine learning technologies // Proceedings of the 2020 conference on fairness, accountability, and transparency. 2020. 272–283.

Capítulo 4

Introdução às Redes Neurais Profundas com Python

Josenildo C. da Silva, Raimundo Osvaldo Vieira

Abstract

Deep network are one of the most important subfield of machine learning. Everyday we learn about new applications or new architectures attracting a lot of interest from industry and academy as well. This lecture is a gentle introduction on deep neural networks for developers who want to give the first steps in this field. This text contains the fundamental concepts, main architectures and libraries. It is complemented by code examples covering each concept discussed.

Resumo

As redes neurais profundas representam uma das mais importantes subáreas da aprendizagem de máquina. Todos os dias surgem novas aplicações ou novas arquiteturas tornando esta tecnologia um tópico de muito interesse tanto para o mercado como para academia. Este minicurso é uma introdução às redes neurais profundas para desenvolvedores que desejam dar os primeiros passos nesta área. O texto apresenta os conceitos básicos, principais arquiteturas e bibliotecas e é complementado com exemplos de código para demonstrar na prática cada tópico.

4.1. Introdução

O desenvolvimento de algoritmos tradicionais é ponto central de estudo da ciência da computação. O programador tem a tarefa de compreender um dado problema e criar um programa que produz a saída requerida, de acordo com as restrições do problema. Isto funciona sempre que o problema é matematicamente bem definido, tais como programas para área comercial, aplicações bancárias, jogos, editores de texto, sistemas operacionais, etc. Entretanto, há algumas situações onde as regras não são claras ou o objetivo é descobri-las. Neste caso, a abordagem tradicional não se aplica e o desenvolvimento de um algoritmo torna-se inviável. Nestes casos, utilizamos a aprendizagem de máquina.

Uma das aplicações mais tradicionais de aprendizagem de máquina é a análise de crédito para solicitação de empréstimos ou cartão de crédito. Alguns exemplos mais recentes são aplicativos ou websites recomendadores automáticos (de livros, filmes, músicas, etc), detectores de *spam* nos aplicativos de email, reconhecedores de voz nos telefones celulares, identificadores de rosto no aplicativo de fotos Picasa, ou no Facebook. O maior exemplo, no entanto, talvez seja o mecanismo de busca da Google. Na realidade, todos os produtos da Google observam atentamente nossos movimentos e utilizam aprendizagem para fazer as propagandas do Google AdWords, ou manter o ranking das páginas. Atualmente, somos usuários de aprendizagem de máquina diariamente, mesmo sem nos darmos conta disto.

Uma importante subárea da aprendizagem de máquina é chamada de redes neurais profundas. A partir dos primeiros trabalhos sobre neurônios artificiais, na década de 1940, esta linha de pesquisa tem produzido muitos avanços nas ultimas décadas. Mas foi somente na década de 2000 que as redes neurais profundas começaram a chamar a atenção. Dois fatores contribuíram para o sucesso destas redes: o surgimento das unidades de processamento gráfico (GPUs) e o aumento da quantidade de dados disponíveis. Os primeiros sucessos das redes neurais profundas foram na área de processamento de imagem. Problemas tais como detecção de objetos, segmentação de imagem e até mesmo geração de legenda para imagens são exemplos de aplicação mais conhecidos. As redes profundas também aceitam outros tipos de dados, tais como áudio, vídeo ou texto. Os resultados mais recentes são de redes geradoras de imagens a partir de texto, DALL-E[1] e Imagen[2].

Este é um minicurso introdutório, voltado para desenvolvedores que desejam dar os primeiros passos na área de deep learning. Assumimos conhecimento prévio de álgebra básica e algoritmos clássicos de aprendizagem de máquina. Também é desejável que você tenha conhecimento básico da linguagem Python, o que pode facilitar o entendimento dos exemplos apresentados. Nosso objetivo é dar uma visão geral sobre redes neurais profundas, apresentando os conceitos fundamentais bem como exemplos em linguagem Python. Ao final deste minicurso, o aluno deve ter uma clara ideia sobre os componentes básicos de redes neurais profundas, seu potencial e indicações de como iniciar uma trilha de estudos nessa área.

Este texto faz parte do material do minicurso, juntamente com os códigos em Python que serão disponibilizados aos participantes. O objetivo é apresentar os conceitos básicos no texto e apresentar os aspectos avançados através de discussão com código. Nas seções seguintes, discutiremos os fundamentos da aprendizagem de máquina, redes neurais clássicas, redes neurais profundas e redes neurais recorrentes.

4.2. Fundamentos de Aprendizagem de Máquina

A visão mais atual de aprendizagem de máquina é a de construção de modelos. A tarefa de criar um software capaz de fazer predições é equivalente a criar um modelo de uma função desconhecida a partir dos dados disponíveis. A primeira etapa desta tarefa é definir uma família de modelos, como, por exemplo, funções lineares, árvores de decisão, ou mesmo redes neurais. O trabalho do algoritmo de aprendizagem pode ser visto como uma

[1] `https://openai.com/blog/dall-e/`
[2] `https://imagen.research.google/`

busca por uma instância de modelo específico que melhor represente o conjunto de dados [Hastie et al. 2009].

De modo geral, pode-se definir este problema como uma busca em espaço de estados, onde cada estado é um modelo possível. Para guiar o processo de navegação no espaço de estados, normalmente os algoritmos utilizam alguma **função de custo** além de **função de avaliação** do modelo, para decidir se o modelo atual é bom o suficiente para representar a função desconhecida ou se é necessário continuar a busca. Como se trata de uma busca local, ou seja, não importa o caminho percorrido até o modelo escolhido, tem-se um problema de otimização. Se a função de avaliação é convexa, pode ser possível encontrar uma **formula fechada** para produzir o modelo. Se a função de avaliação não for bem comportada, com vários máximos locais, é necessário utilizar alguma técnica de **otimização** avançada.

Para exemplificar, considere a regressão linear simples. O modelo produzido é uma combinação linear de polinômios. Cada estado é uma combinação em particular definida pelos valores dos coeficientes. A função de avaliação mais utilizada é a soma dos quadrados das diferenças entre previsão e valor correto de y (SSE, sum of squared errors). O algoritmo *least square regression* (LSR) utiliza uma fórmula fechada para calcular os coeficientes do modelo a partir dos dados, sem ter que navegar pelo espaço de estados.

Um outro exemplo é o algoritmo ID3. O modelo produzido pelo ID3 é simbólico, consiste em uma árvore formada pelo nome dos atributos e valores de decisão a cada nível. Nas folhas há o valor de cada classe. A construção é iterativa e a navegação é guiada por uma função heurística que define qual atributo escolher para a raiz da próxima subárvore.

4.2.1. Tipos de Aprendizagem

Um pressuposto importante na aprendizagem de máquina é que existe uma relação (desconhecida) entre os conjuntos de dados que estamos interessados. O objetivo da aprendizagem é construir um modelo que se comporte de modo similar à relação desconhecida.

Há três configurações básicas para aprendizagem automática, de acordo com o formato das informações disponíveis: aprendizagem supervisionada, aprendizagem não-supervisionada e aprendizagem por reforço [Géron 2017].

Na **aprendizagem supervisionada** há um conjunto X representando observações e um conjunto Y representando valores associados a cada elemento de X. Assim, pode-se dizer que, na aprendizagem supervisionada, para cada $x \in X$ sabe-se qual é o $y \in Y$ correto. Em outras palavras, existe uma coleção de pares $\{(x,y) \mid x \in X, y \in Y\}$, ou seja, uma amostra da relação verdadeira de modo enumerado. O objetivo do algoritmo de aprendizagem é encontrar uma aproximação mais concisa para esta relação.

Na **aprendizagem não-supervisionada** há um conjunto X representando as observações, mas não há um conjunto Y. Deste modo, não há valores corretos a priori e, portanto, não há relações para se descobrir. Entretanto, há várias propriedades dos dados que são de interesse. Por exemplo, pode-se investigar as estatísticas básicas das observações (média, desvio padrão, etc.), saber quantas modas existem nos dados, se há regiões mais densas que outras, etc.

Na **aprendizado por reforço** o objetivo é encontrar a melhor ação a, a ser escolhida

em um dada situação x, com a recompensa y. Além disso, não há conjuntos X, Y ou A à disposição, mas temos acesso a cada par (a,x,y) um por vez. O algoritmo constrói um modelo de modo incremental, à medida que chegam novas observações, que representa uma política de escolhas de ações de modo a maximizar a recompensa.

4.2.2. Utilização do Conhecimento

Uma vez que se tenha um modelo construído, há pelo menos duas coisas que se pode fazer: *prever o futuro* ou *explicar o passado*. De modo mais técnico, as principais tarefas de aprendizagem, de acordo com o uso do modelo, são: predição e descrição.

Predição é a tarefa de calcular valores Y para tuplas futuras. Por exemplo, se temos um modelo $f(\cdot)$ que foi treinado com os pares $\{(2,4),(9,81),(3,9),(10,100),(4,16)\}$, podemos utilizá-lo para predizer qual o valor y quando $x = 7$, ou seja, $y = f(7)$, ou ainda $(7,y)$. Duas formas de predição mais conhecidas são a regressão e a classificação. Fala-se de *regressão* quando o conjunto Y é continuo. Por outro lado, quando o conjunto Y é discreto, fala-se de *classificação*. A predição, pressupõe que haja um conjunto Y com os valores corretos ou categorias. Portanto, a predição é aprendizagem supervisionada.

Descrição é a tarefa de construir modelos que sintetizam as principais características do conjunto de observações. O objetivos é tentar descobrir se há alguma estrutura subjacente no conjunto de observações ou se é possível descrever o processo de geração dos dados. Duas formas de descrição mais conhecidas são a estimação de densidade e a análise de grupos. Como não há indicação de grupos a priori, trata-se de uma forma de aprendizagem não-supervisionada.

4.3. Redes Neurais Artificiais Clássicas

Redes Neurais Artificiais representam uma técnica supervisionada de predição que gera um modelo não linear representando um grafo de unidades neuronais e os pesos de suas conexões [Coppin and Valério 2015]. Os algoritmos mais conhecidos são Perceptron, Adaline e Multilayer Perceptron (MLPs). As redes neurais profundas são o desenvolvimento mais recente desta família de modelos.

4.3.1. Neurônio biológico, Perceptrons e MLPs

As redes neurais naturais são formadas por um intricado grafo de conexões entre os neurônios biológicos. Um neurônio recebe impulsos elétricos de outros neurônios e se a soma de todos os impulsos de entrada ultrapassam um determinado limiar, ele transfere o impulso para outros neurônios a ele conectado (veja Figura 4.1).

Um **perceptron** é uma descrição matemática de um neurônio biológico. Os perceptrons realizam uma soma ponderada dos valores de entrada, seguido de uma função de ativação, que se comporta de modo similar ao neurônio natural (Figura 4.2).

A saída será ativada se a soma atingir um valor de limiar.

$$y = \begin{cases} 1, \text{se } \mathbf{W}\mathbf{x} + b > 0 \\ 0, \text{caso contrário.} \end{cases} \tag{1}$$

Uma rede neural artificial clássica é uma estrutura de camadas onde vários percep-

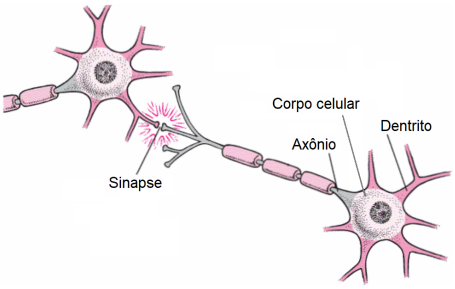


Figura 4.1. Neurônio biológico.

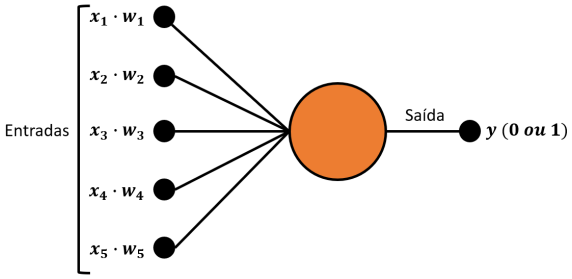


Figura 4.2. Perceptron.

trons são interconectados. Uma rede com esta arquitetura é denominada **perceptron de múltiplas camadas**, ou MLPs (da sigla em inglês). Nesta formato, as camadas internas da rede, que não sejam entrada ou saída, são chamadas de camadas escondidas (veja Figura 4.3).

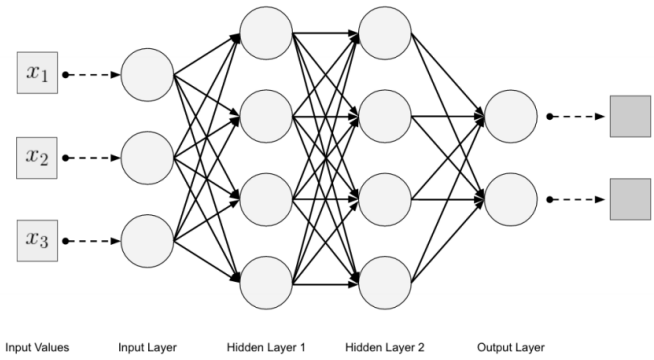


Figura 4.3. Exemplo de uma MLP com duas camadas escondidas.

As MLPs são chamadas de redes de alimentação adiante, ou *feedforward networks*, porque a informação é processada desde a camada de entrada até a camada de saída sem que haja retroalimentação entre as camadas ou unidades [Goodfellow et al. 2016].

4.4. Python, Tensorflow, Keras

4.4.1. Python

Python é uma das linguagens de programação mais populares atualmente. Aparece em terceiro lugar no TIOBE Index em setembro de 2019[3] tendo crescido muito o interesse por ela nos três últimos anos. Python tem se tornado uma das linguagens mais utilizadas para análise de dados junto com R e Júlia.

Python é uma linguagem de sintaxe simples, código aberto, que conta com uma grande comunidade de colaboradores. Isto torna a linguagem particularmente acessível e adequada para atacar vários tipos de problemas em domínios diferentes. As bibliotecas (packages) mais relevantes para a área de aprendizagem de máquina são: numpy, pandas, matplotlib, jupyter, entre outros.

A linguagem Python foi criada por Guido van Rossum em 1991 com o objetivo de ser clara, concisa e explícita. Para programadores com alguma experiência em outras linguagens, alguns aspectos do Python são muito estranhos, como a utilização de espaços em branco.

A tipagem em Python é dinâmica, o que significa que não precisamos declarar tipos ao criar uma variável. Entretanto, todo valor atribuído a uma variável possui um tipo associado. Recentemente, foi acrescentado a possibilidade de indicar o tipo de variáveis, parâmetros e retornos de funções, chamado de *type hints*, ou dicas de tipo (em tradução livre).

Python permite trabalhar com vários paradigmas. Oferece a possibilidade de se trabalhar orientado à objetos, mas também aceita que um programa seja baseado em funções. Além disso, por não ser compilado, permite prototipagem rápida ideal para área de análise de dados e aprendizagem de máquina.

4.4.2. Tensorflow

Tensorflow (TF) é uma plataforma de desenvolvimento de modelos de aprendizagem de máquina em larga escala [Abadi et al. 2015]. TF é desenvolvido em python pela Google para permitir operações com tensores de modo eficiente. De certo modo, o TF é muito similar ao Numpy. Entretanto, o TF consegue trabalhar com CPU e GPU, permite a distribuição de operações entre várias máquinas, além de poder ser exportado para outras plataformas tais como C++, Javascript ou Tensoflow Lite.

Tensorflow representa internamente a execução de uma função através de um grafo dirigido. Cada nodo representa uma operação, por exemplo multiplicação, ou soma. A informação flui através do grafo de computação na forma de vetores de dimensão arbitrária chamados de tensores.

O TF realiza diferenciação automática de funções utilizando um grafo específico para isso (veja Figura 4.5). Portanto, se um determinado nó do grafo representa uma operação $f(x)$, o TF calcula também sua derivada df/dx no gráfico de derivadas. Essa abordagem facilita a implementação de algoritmos tais como *backpropagation*

[3] `https://www.tiobe.com/tiobe-index/`

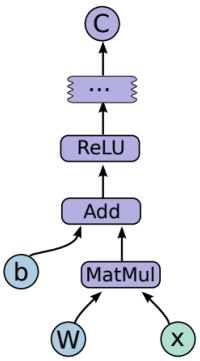


Figura 4.4. Grafo computacional do TensorFlow. Cada nó representa uma operação [Abadi et al. 2015].

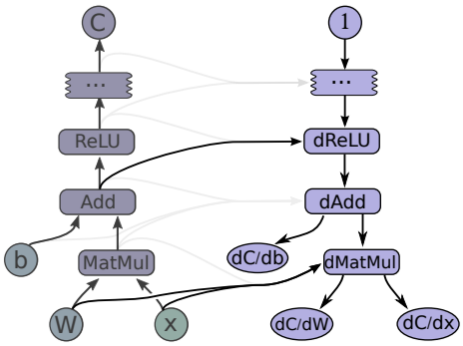


Figura 4.5. Grafo para derivação automática [Abadi et al. 2015].

A implementação do TF inclui vários pacotes com funções específicas tais como processamento de E/S, otimização, deployment, visualização com tensorboard, além de uma extensa lista de operações matemáticas (veja Figura 4.6). TF possui ainda pacotes para processamento de sinais, tratamento de imagens, e datasets de exemplos prontos para uso.

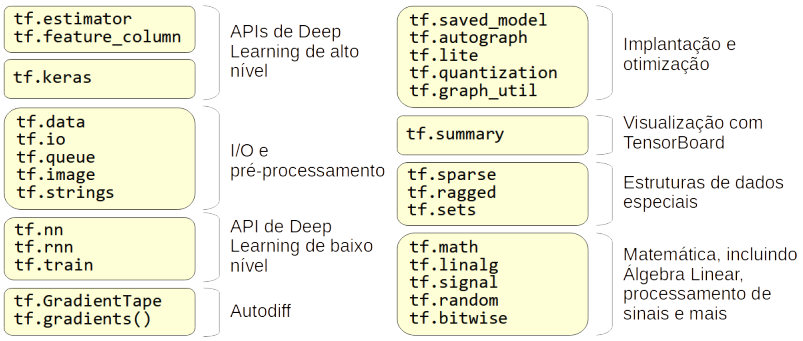


Figura 4.6. API do Tensorflow em Python [Géron 2017].

4.4.3. Keras

Keras é uma API de alto nível desenvolvido para facilitar o uso de tensorflow (veja figura 4.7). Keras foi lançado em 2015, e inicialmente foi desenvolvido sobre o Theano, outra biblioteca para manipulação de tensores que também tem diferenciação automática de funções. Em 2018 o Keras foi escolhido pelo Tensoflow como sua API de alto nível oficial.

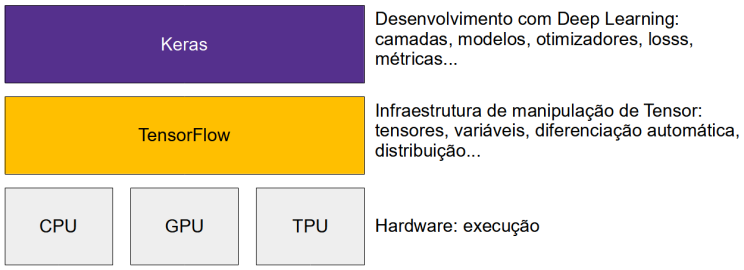


Figura 4.7. Keras é uma API desenvolvida a partir do Tensorflow [Chollet 2021].

Os principais componentes e etapas do Keras são: camadas, modelos, compilação e treinamento.

Camadas (Layers). Camadas são unidades de processamento no Keras que recebem um tensor de entrada e produzem outro tensor de saída. Os pesos de uma camada são aprendidos durante o treinamento da rede.

Modelos (Models). Uma rede neural em Keras é um grafo de camadas represendado pela classe Model. O modelo mais simples é chamado de Sequential, mas o Keras permite a definição de modelos com topologia complexa, não-sequencial.

Compilação. O método `compile()` configura o processo de treinamento. Geralmente são informados o otimizador, a função de custo e as métricas a serem utilizados no treinamento.

Treinamento. O método `fit()` executa o treinamento. Seus principais parâmetros são os dados de treinamento, o número de épocas e o tamanho do subset de treino (batch).

4.4.4. Exemplo: rede MLP simples

No código a seguir, mostramos como implementar uma MLP em tensorflow. Para isso vamos importar o pacote `tensorflow`.

```
import tensorflow as tf
```

Usaremos o dataset de dígitos manuscritos MNIST, que é distribuído no tensorflow. Há uma versão mais recente deste dataset, chamado fashion MNIST, que possui imagens de itens de moda. Ele pode ser obtido com `fashion_mnist` no lugar de `mnist`.

```
mnist = tf.keras.datasets.mnist
```

Separamos os dados de treino e de teste e também normalizamos os valores para que estejam no intervalo de 0 a 1.

```
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

O modelo tem como entrada um tensor de 784 neurônios definido pela camada Flatten, que recebe um tensor de rank-1. O valor 784 é o número de pixeis em cada imagem de dimensões 28x28. Nesta MLP há uma camada escondida de 128 neurônios e uma camada de saída com 10 neurônios.

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

O modelo definido é compilado com um otimizador `adam` e uma função de custo `entropia cruzada categórica`, que é a função de custo mais utilizada para treinar um classificador. Além disso, o modelo será avaliado com a métrica `acurácia`.

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Finalmente, iniciamos o treinamento chamando o método `fit()` do modelo compilado. Neste exemplo são definidos apenas 5 épocas de treinamento. Na prática, pode ser necessário um número muito maior de épocas para treinar um modelo.

```
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

4.5. Redes Neurais Convolucionais

Uma rede neural profunda [Goodfellow et al. 2016] é uma MLP, ou seja, uma rede de alimentação para frente (feedforward network) que possui várias camadas escondidas, tais como discutidas na seção 4.3. Estas redes são inspiradas no cortex visual do cérebro, onde os sinais exteriores são processados por várias camadas. Atualmente, o número de camadas pode chegar a centenas, dependendo da arquitetura. A AlexNet [Krizhevsky et al. 2012] por exemplo, possui 8 camadas. A VGGNet possui versões com 16 a 19 camadas [Simonyan and Zisserman 2014]. Já a ResNet pode chegar a 152 camadas [He et al. 2016]. As redes neurais profundas utilizam muitas operações matriciais e portanto só ganharam atenção com o desenvolvimento de GPUs (*graphic processor units*), que permitiram a execução paralela de um grande número de operações.

Talvez o ponto mais importantes das redes neurais profundas seja não apenas o número de camadas, mas a introdução de vários operadores que não eram utilizados em MLPs. Nas seções seguintes, vamos discutir os principais destes operadores: convolução, pooling e flattening. Devido à utilização de camadas de convolução, estas arquiteturas são geralmente chamadas de **redes neurais convolucionais**, ou **convolutional neural networks** (CNN) no original em inglês.

4.5.1. Principais Componentes das CNNs

Convolução Convolução é uma operação que aplica repetidamente um filtro (kernel) sobre os dados de entrada para gerar um mapa de atributos (features). O kernel geralmente tem dimensões pequenas e representa padrões locais a serem detectados. Matematicamente, a operação consiste em uma multiplicação do kernel com uma fatia dos dados de entrada. A operação utilizada em redes profundas é uma correlação cruzada (cross-correlation), mas o nome convolução continua sendo amplamente utilizado.

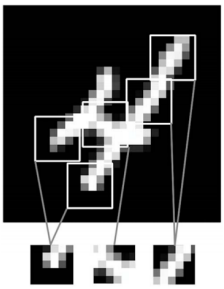


Figura 4.8. Kernels representam padrões locais aprendidos [Chollet 2021].

Ao contrário de um filtro utilizado em outros domínios, por exemplo tratamento de imagens, os valores do kernel em redes profundas são aprendidos durante o treinamento. O kernel representa um padrão local um atributo aprendido e é similar aos filtros de borda em imagens, linhas, ou outros padrões mais complexos (cf. Figura 4.8). Quando a característica representada pelo kernel está presente em uma região dos dados de entrada, o resultado da operação indica um valor maior que das regiões onde nada foi detectado. A utilização de operação tem como saída um mapa de resposta ou mapa de características (*feature map*), que geralmente é menor que os dados originais e que serve de entrada para as camadas seguintes da rede.

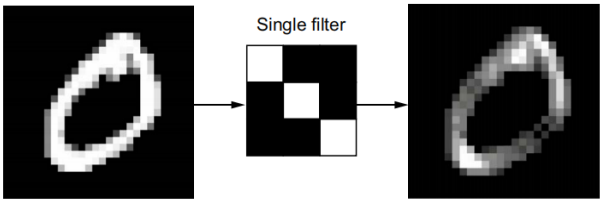


Figura 4.9. Mapa de resposta (*feature map*) resultante da aplicação de um filtro sobre uma imagem [Chollet 2021].

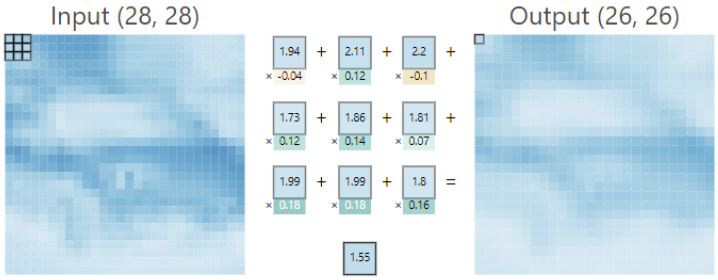


Figura 4.10. Convolução: multiplica-se cada elemento do filtro por um elemento equivalente na fatia da imagem e depois todos os termos são somados [Wang et al. 2020].

Há duas característica importantes das redes baseadas em convolução. Primeiro, os padrões aprendidos por convolução são independentes de localização. Desse modo, um padrão aprendido pode ser detectado em qualquer posição da imagem. Portanto, diz-se que os padrões aprendidos são *invariante de translação*. Segundo, os padrões aprendidos nas primeiras camadas são utilizados como elementos para compor padrões mais complexos em camadas subsequentes. Isto forma uma hierarquia de padrões[4].

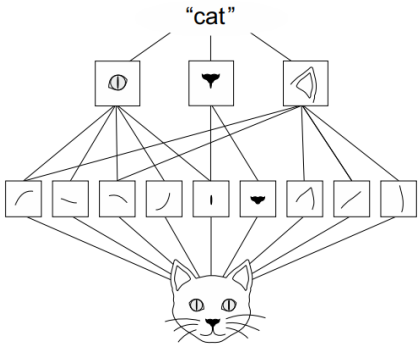


Figura 4.11. Hierarquia de padrões aprendidos em uma rede neural profunda [Chollet 2021].

A operação de convolução possui três hiperparâmetros: padding, kernel size e stride. Vamos discutir cada um deles.

[4]Um dos primeiros trabalhos a incluir todas estas características foi a rede Neurocognitron em 1980. O trabalho do Prof. Fukushima é reconhecido como o precursor das Redes Neurais Convolucionais.

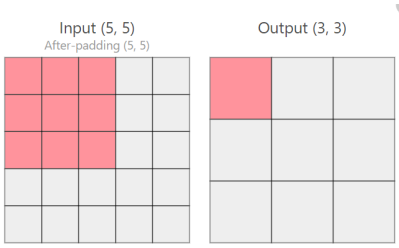


Figura 4.12. Operação de covolução com um kernel de 3x3 em uma imagem de 5x5, com padding = 0 e stride = 1.

kenel size: Define o tanho do filtro que será utilizado. Tamanhos típicos são 2x2, 3x3, ou 5x5.

padding: Ao aplicar o kernel, pode-se preencher a entrada com zeros para que o filtro seja aplicado em mais áreas limites. Um valor típico de padding é 0. Em Keras, usa-se o valor ”valid”para indicar padding = 0, and ”same”para usar um valor de padding automático.

stride: O filtro é aplicado em forma de janela deslizante. O stride define o passo em que o filtro será aplicado. Valor típico é 1.

Pooling. A operação de pooling aplica uma função matemática em uma subárea do input e substitui todos os valores pelo resultado da função. Geralmente usa-se a função max, mas também é possível utilizar a função média. Um tamanho típico de pooling é 2x2. Neste caso, uma região de 2x2 será substituída pelo valor máximo encontrado na região 4.13.

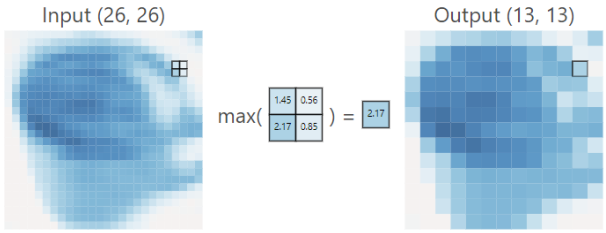


Figura 4.13. Max pooling de 2x2 aplicado a um input de 26x26. A Saída será 13x13.

Um dos efeitos do pooling é a redução das dimensões da entrada. Quanto maior o tamanho da camada de pooling, maior será a redução.

Flattening. Quando as operações de convolução e de pooling são realizadas em um tensor de rank-2 ou maior, ou seja, em uma entrada de duas ou mais dimensões, é necessário mudar o formato da camada antes de passar para etapa de classificação, geralmente um MLP. Por isso, a operação de flattening transforma os dados para um tensor de rank-1 (veja Figura 4.14).

Normalmente, a operação de flattening é aplicada sobre o mapa de características produzidas pelas camadas convolutiva anteriores. O tensor rank-1 resultante é utilizado

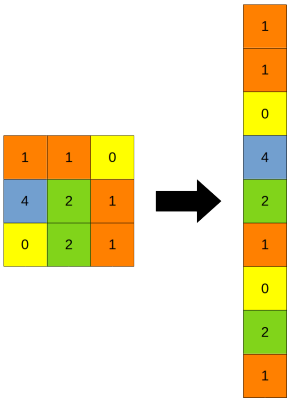


Figura 4.14. Operação de flattening.

como entrada em uma camada totalmente conectada, geralmente chamada de camada densa na literatura de redes profundas.

Funções de ativação. As funções de ativação mais conhecidas ao se construir uma MLP são a sigmoide e a tangente hiperbólica. Entretanto, a maioria das funções de

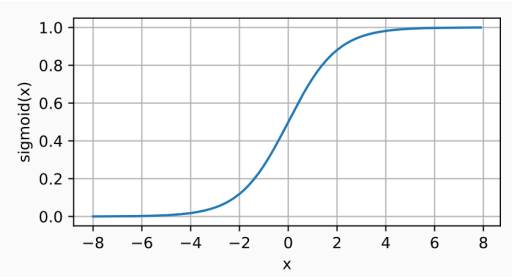


Figura 4.15. Função sigmoide [Zhang et al. 2021].

ativação podem fazer com que o gradiente fique próximo de zero em redes profundas, devido à grande quantidade de camadas onde o gradiente tem que ser multiplicado durante o processo de *backpropagation*. Devido à isso, várias outras funções de ativação são utilizadas em redes neurais profundas, dentre elas a *rectified linear unit*, ou ReLU.

$$ReLU(x) = \max(0, x) \tag{2}$$

A função ReLU é utilizada após a operação de convolução. O resultado é que apenas os valores positivos são preservados (veja Figura 4.16. Além disso os valores positivos não são restridos, como na sigmoide (veja Figura 4.15). Logo, não há saturação quando x é positivo. A ReLU pode ser utilizada em qualquer camada escondida logo após uma camada de convolução.

Existem várias outras alternativas à função ReLU, tais como Leaky ReLU, GeLU, e assim por diante. Todas buscam corrigir algum problema da ReLU. Ainda assim, ela continua sendo largamente utilizada na maioria das arquiteturas de redes profundas.

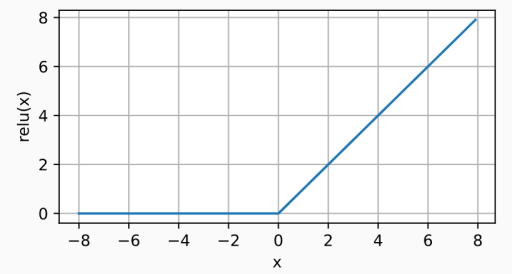


Figura 4.16. Função ReLU [Zhang et al. 2021].

Os valores de saída de uma ReLU não são restritos a nenhum valor superior. Para saber qual a classe indicada por uma rede, seria suficiente tomar o valor máximo na saída da rede. Contudo, geralmente é útil expressar a saída da rede em forma de probabilidade, onde todos os valores são normalizados para somarem 1. A função softmax faz isso. Além disso, ela é derivável, portanto pode ser utilizada diretamente no algoritmo de backpropagation.

A função softmax é definida como:

$$Softmax(\mathbf{z}) = \frac{\exp(z_i)}{\sum_{j=0}^{K} \exp(z_j)} \tag{3}$$

Por exemplo, considere que uma determinada classe tem o valor final, antes da softmax, de 8.65. A softmax vai somar todos os valores para outras classes e somar, e usar esse total para normalizar o valor 8.65. Veja a imagem na Figura 4.17.

A função softmax é utilizada como ativação na ultima camada de uma rede neural.

$$\frac{\exp(8.65)}{\begin{array}{l}(\exp(-4.26) + \exp(2.97) + \exp(-0.38) + \exp(5.24) + \\ \exp(-7.58) + \exp(-3.43) + \exp(8.65) + \exp(2.63) + \\ \exp(6.31) + \exp(0.69))\end{array}} = 0.8808$$

Figura 4.17. Exemplo numérico da aplicação de softmax.

4.5.2. Exemplo: arquitetura LeNet

Uma das primeiras redes que utilizam operação de convolução foi proposta por Yann LeCun [Lecun et al. 1998]. A LeNet foi aplicada para reconhecimento de dígidos manuscritos e ajudou a despertar o interesse para as arquiteturas profundas.

LeNet apresenta sete camadas, todas elas com parâmetros modificáveis durante o treino. As quatro primeira camadas apresentam blocos de Convolução seguida de Pooling. Em seguida é aplicada uma operação de Flattening que produz um vetor de entrada para três camadas Densas finais. A última camada tem como saída uma função de ativação softmax que produz a probabilidade da imagem de entrada ser um dos dígitos de 0 a 9.

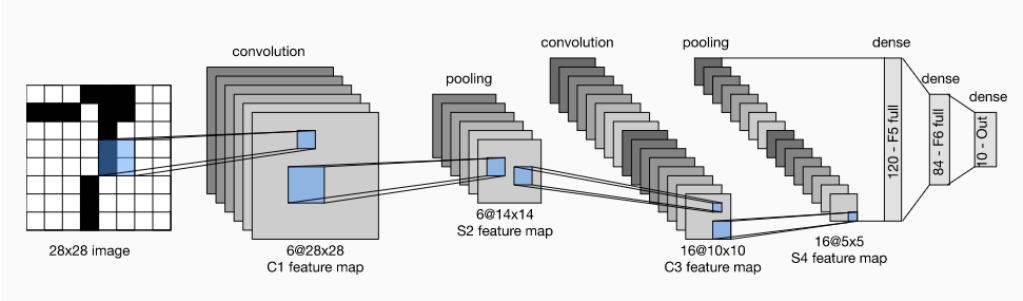


Figura 4.18. Arquitetura da rede LeNet (reproduzido de [Zhang et al. 2021]).

4.5.3. Exemplo: classificador de dígitos manuscritos

O exemplo a seguir foi extraído da documentação oficial do Keras. A arquitetura produzida é muito similar a uma rede LeNet. O objetivo é apresentar um exemplo mínimo mas funcional em Keras.

Os pacotes necessários para este exemplo são numpy, keras e layers.

```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
```

O dataset MNIST já vem disponível no Keras. As imagens são em escala de cinza, apenas um canal, com dimensões 28x28. No Keras o dataset já está dividido em treino e teste. Na prática, quando você utilizar um outro dataset, esta divisão terá que ser realizada explicitamente pelo programador.

```
# Model / data parameters
num_classes = 10
input_shape = (28, 28, 1)
# Load the data and split it between train and test sets
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

Os valores das imagens precisam ser normalizadas para valores entre 0 e 1. Originalmente estão em escala de cinza com valores de 0 a 255.

```
# Scale images to the [0, 1] range
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255
# Make sure images have shape (28, 28, 1)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train shape:", x_train.shape)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")
```

As classes são convertidas para uma codificação vetorial que facilita a saída de uma rede com 10 neurônios. Essa representação algumas vezes é chamada de *one-hot encoding*. Como exemplo, a classe 0 pode ser representada como [1,0,0,0,0,0,0,0,0,0]. A classe 1 pode ser representada como [0,1,0,0,0,0,0,0,0,0]. E assim por diante. Cada posição do vetor representa a classe da imagem.

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Esta rede possui uma entrada de 28x28. Em seguida, uma camada de convolução com 32 kernels de 3x3 é aplicada sobre a entrada, gerando os mapas de características. Uma função ReLU é utilizada para retificar os mapas de características. Em seguida uma camada de pooling de 2x2 e aplicada tendo como efeito a redução das camadas de features para metade do seu tamanho. A mesma sequencia é repetida, desta vez com 64 kernels 3x3, ReLU e pooling.

Finalmente, os mapas de caracteristicas são tranformados em um tensor rank-1 e passados como entrada para uma camada MLP (densa) com saída de 10 neurônios. A função de ativação na saída é uma softmax.

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

O Keras produz um resumo da arquitetura construída com o método `summary()` do modelo.

```
model.summary()
```

Este exemplo utiliza subconjuntos de 128 imagens para cada etapa do treinamento (batchs). O treinamento total será feito em 15 épocas.

```
batch_size = 128
epochs = 15
```

O modelo vai ser treinado com a função de custo ”categorical crossentropy”, muito utilizada para problemas de classificação. O otimizador utilizado é o Adam e a qualidade do modelo final será determinada utilizando acurácia.

```
model.compile(loss="categorical_crossentropy", optimizer="adam",
                                    metrics=["accuracy"])
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
                                    validation_split=0.1)
```

O método `evaluate()` permite examinar os valores finais da função de custo e da função de qualidade, neste caso a acurácia.

```
score = model.evaluate(x_test, y_test, verbose=0)
```

```
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

4.5.4. Aplicações de redes convolucionais

As redes convolucionais tem sido aplicadas em vários domínios diferentes e com uma vasta gama de tipos de dados. Para reconhecimento de objetos em imagens, detecção de objetos, segmentação de imagens. As CNNs tem sido ainda utilizadas para detecção de padrões em aúdio, reconhecimento de voz, tratamento de linguagem natural, análise de vídeo, detecção de padrões de sono em sinais de EEG.

4.6. Redes Recorrentes

Sequências são um tipo de dados onde a ordem é importante, o tamanho pode variar e os elementos podem se repetir. Por exemplo, letras em uma palavra pode ser visto como uma sequência. Palavras em um texto também são sequências. Notas musicais, frames de vídeos, valores de demandas de produtos, preços de ações em bolsa de valores, registro de consumo de energia.

As redes neurais convolucionais (CNNs) vistas na Seção 4.5 não conseguem representar dados sequenciais. Uma CNN recebe um tensor X de entrada e calcula uma saída y. Entretanto, a arquitetura CNN não consegue manter a ordem dos dados, ou lidar com sequências de tamanho variado ou muito grandes.

Mas, então, como modelar dados em sequência? A seguir, vamos discutir algumas abordagens, limitações e introduzir as redes neurais recorrentes e suas variações.

4.6.1. Redes Neurais Recorrentes (RNNs)

Uma primeira forma de modelar uma sequência $\mathbf{X} = (x_1, x_2, \ldots, x_t)$ poderia ser calcular a **probabilidade da sequência completa**, a partir da probabilidade de cada elemento. Assim, teremos:

$$p(\mathbf{X}) = \prod_{t=1}^{T} p(x_t) \tag{4}$$

Mas esse modelo assume independência entre o elementos da sequência, o que não é verdadeiro para sequências.

Um melhor modelo seria **condicionar a probabilidade** do próximo elemento aos elemento passados, ou seja, $p(x_t|x_{t-1}, x_{t-2}, \ldots, x_0)$. Entretanto, essa abordagem não é escalável. Mesmo considerando apenas um elemento anterior, para modelar a palavras em uma língua, teríamos que calcular um número gigante de probabilidades $p(x_t|x_{t-1})$, da ordem de $|\Sigma|^2$, onde Σ é o vocabulário da língua em questão.

Vetorizar o contexto é uma técnica que recebe $x_{t-1}, x_{t-2}, \ldots, x_0$ e devolve um valor h que representa um sumário dessa sequência de contexto. Desse modo:

$$p(x_t|x_{t-1}, x_{t-2}, \ldots, x_0) \approx p(x_t|h) \tag{5}$$

A **redes neuronais recorrentes** (*recurrents neural networks*, RNNs) codificam o contexto h através da multiplicação de uma matriz $\mathbf{W}$ de pesos aprendidos com o contexto anterior. Há outra matriz de pesos aprendidos $\mathbf{U}$ que é aplicada a entrada atual no cálculo do novo contexto h [Goodfellow et al. 2016]:

$$h_t = \tanh\left(\mathbf{W}^\top \mathbf{h}_{t-1} + \mathbf{U}^\top x_t\right) \tag{6}$$

Na literatura de RNNs, h é geralmente chamado de *hidden state*.

Assim, pode-se calcular $p(x_{t+1}|h_t)$ como:

$$p(x_{t+1}|h_t) = softmax(\mathbf{V}^\top \mathbf{h}_t) \tag{7}$$

4.6.2. LSTMs e GRUs

As RNNs sofrem de alguns problemas graves. Primeiro, não são capazes de trabalhar com sequências de tamanho variado. Além disso, devido ao número de matrizes intermediárias no cálculo de h, o gradiente pode tender a zero, tornando o aprendizado ineficiente. Este problema é chamado de vanishing gradients. As RNNs também não conseguem manter um contexto muito longo, pois o contexto é transformado por várias multiplicações de matrizes ao longo da construção do contexto.

As **long short term memory** (LSTMs) resolvem estes problemas através da introdução dos chamados portões (*gates*) e também do contexto de longo prazo c. O *forget gate* controla quais informações serão removidas do contexto de longo prazo. O *input gate* controla o que será adicionado ao contexto de longo prazo. O *output gate* utiliza a entrada x_t, o contexto de curto prazo h_{t-1} e o contexto de longo prazo c_{t-1} para produzir a saída h_t.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t]) \tag{8}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t]) \tag{9}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t]) \tag{10}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_{t-1} \tag{11}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t]) \tag{12}$$
$$h_t = o_t * \tanh(C_t) \tag{13}$$

As **gated recurrent unit** (GRUs) são uma versão simplificada das LSTMs, que trabalha com somente dois *gates*. Ela combina o forget e o input gate em um só e também combina contextos de longo e curto prazo.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{14}$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{15}$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{16}$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_{t-1} \tag{17}$$

4.6.3. Aplicações de Redes Recorrentes

As redes recorrentes são utilizadas em vários cenários distintos. Os principais tipos de tarefas quando se modela sequências são:

muitos-para-um: o contexto h representa vários elementos da sequência e a predição será para apenas um elemento x_{t+1}. Exemplo: texto de um comentário é o contexto e o sentimento é a saída calculada.

um-para-muitos: o contexto h representa um elemento da sequência e a predição será para vários elementos $x_{t+1}, x_{t+2}, \ldots$. Exemplo: uma imagem é a entrada e um texto de legenda é a saída gerada.

muitos-para-muitos: o contexto h representa vários elementos da sequência e a predição será para vários elementos $x_{t+1}, x_{t+2}, \ldots$. Exemplo: tradução automática de uma língua para outra.

4.7. Conclusão e Próximos Passos

Este minicurso apresentou uma breve introdução às redes neurais profundas. Nosso foco foi as arquiteturas voltadas para aprendizagem supervisionada, uma vez que isso já representa muito material a ser discutido. Todo o código usado está disponibilizado como material suplementar que acompanha esse texto.

Esperamos que este minicurso sirva para despertar o interesse do leitor para a área de redes neurais profundas. A partir desta introdução, o aluno interessado pode escolher uma trilha de estudos sobre aspectos fundamentais das redes, tais como funções de ativação ou arquiteturas avançadas, ou ainda uma trilha de aplicações.

Mesmo com o grande número de trabalhos que são publicados na área a cada dia, acreditamos que as bases são as mesmas e este minicurso pode servir de arcabouço para os estudos futuros. Recomendamos ao estudante os livros listados na seção de referência bibliográfica. Alguns são mais téoricos (ex.: [Goodfellow et al. 2016]) e outros mais práticos (ex.: [Géron 2017] ou [Zhang et al. 2021]). Também recomendamos fortemente as playlists de vídeos produzidas pela UCL e pelo MIT sobre o mesmo tema.

Referências

[Abadi et al. 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Chollet 2021] Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.

[Coppin and Valério 2015] Coppin, B. and Valério, J. (2015). *Inteligência Artificial*. Grupo Gen – LTC.

[Géron 2017] Géron, A. (2017). *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

[Goodfellow et al. 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[Hastie et al. 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer.

[He et al. 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Krizhevsky et al. 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Lecun et al. 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Simonyan and Zisserman 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Wang et al. 2020] Wang, Z. J., Turko, R., Shaikh, O., et al. (2020). Cnn explainer: Learning convolutional neural networks with interactive visualization. https://poloclub.github.io/cnn-explainer/.

[Zhang et al. 2021] Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2021). Dive into deep learning. *arXiv preprint arXiv:2106.11342*.

Capítulo

5

Desenvolvimento Ágil e Informatização da Saúde Pública no Brasil

Raul Sidnei Wazlawick, Jades Fernando Hammes, Thaísa Cardoso Lacerda, Ranieri Alves dos Santos

Abstract

In this article we present a set of agile practices and people development techniques successfully used by the Bridge Laboratory at the Federal University of Santa Catarina (UFSC) for the development of large systems for the Brazilian Ministry of Health, especially primary health care, with one of the largest electronic medical records in the world with more than 600 thousand daily users, more than 150 million personal records and more than eight billion clinical records. Bridge is a research and innovation laboratory with more than 160 employees and winner of the Labutantes award as the best innovation laboratory in the Brazilian Southern Region in 2021.

Resumo

Neste artigo apresentamos um conjunto de práticas ágeis e de desenvolvimento de pessoas utilizadas com sucesso pelo Laboratório Bridge da Universidade Federal de Santa Catarina (UFSC) para o desenvolvimento de sistemas de grande porte para o Ministério da Saúde do Brasil, em especial à Atenção Primária em Saúde que conta com um dos maiores prontuários eletrônicos do mundo com mais de 600 mil usuários diários, mais de 150 milhões de cadastros de pessoas e mais de oito bilhões de registros clínicos. O Bridge é um laboratório de pesquisa e inovação com mais de 160 colaboradores e vencedor do prêmio Labutantes como melhor laboratório de inovação da Região Sul em 2021.

5.1. Introdução

O Laboratório Bridge[1] foi estabelecido na UFSC em 2013 inicialmente para a condução de um grande projeto de informatização da saúde pública no Brasil, na época denominado Sistema e-SUS AB (Atenção Básica) e hoje Sistema e-SUS APS (Atenção Primária em

[1] https://portal.bridge.ufsc.br/

Saúde)[2]. Este projeto vem sendo desenvolvido desde então sendo que em 2022 iniciou a sexta contratação sucessiva (e-SUS APS Etapa 6) por pelo menos mais quatro anos.

Com o tempo, outros projetos também significativos foram incorporados ao laboratório. O SISMOB (Sistema de Controle de Obras do Ministério da Saúde)[3], RNI (Registro Nacional de Implantes – ANVISA) [4] e SIGRESIDÊNCIAS (Sistema de Controle de Residências em Saúde no Brasil)[5] foram já finalizados e encontram-se em uso. O Laboratório atua ainda como parceiro no projeto RNDS (Rede Nacional de Dados em Saúde - DATASUS)[6] e leva essa experiência ao Ministério da Educação com o Projeto Rede Aprender, iniciado em 2022, desenvolvendo o aplicativo Jornada do Estudante[7].

Inicialmente atuando com uma equipe de profissionais de mercado que optaram pelo desenvolvimento segundo o Modelo Cascata, o Bridge foi evoluindo suas práticas e se tornando cada vez mais ágil, chegando hoje a um nível de sucesso comprovado por entregas constantes de qualidade dentro dos prazos e orçamentos e com qualidade de excelência.

Atualmente o Bridge organiza-se em onze equipes ágeis com forte participação de designers, com foco em *User eXperience / User Interface* (UX/UI) [GOTHELF and SEIDEN 2021] e analistas de qualidade.

A partir do amadurecimento das práticas ágeis do Bridge dois grandes subprodutos emergiram: o *design system Bold* e o modelo ágil de desenvolvimento *b_thinking*.

Neste texto será apresentado um pouco mais sobre os produtos do Bridge e seu impacto no Brasil bem como o processo que levou o grupo inicialmente baseado no Modelo Cascata para uma cultura ágil e amadurecida.

5.2. A Cultura Bridge

Nesta seção apresentamos a cultura organizacional e ética que evoluiu naturalmente dentro do laboratório. Nossa missão é contribuir para o bem das pessoas com tecnologia e inovação. Nossa visão de futuro consiste em ser a desenvolvedora das melhores soluções tecnológicas da gestão pública. Nossos valores são:

- *Ser a diferença*: temos orgulho e motivação em assumir a responsabilidade de melhorar a vida das pessoas. Para atender este valor, os processos seletivos buscam selecionar as pessoas que possuem maior alinhamento com a missão do laboratório, avaliando de forma criteriosa o *fit* cultural do candidato com o Bridge. As vagas dos processos seletivos são divulgadas em um evento externo, o *b_discovery*, onde os candidatos têm acesso às informações detalhadas das vagas e é possível apresentar o impacto do laboratório no bem-estar das pessoas por meio da tecnologia;
- *Fazer bem-feito, inclusive o café*: nós somos engajados em atingir padrões elevados de qualidade, atendendo e superando expectativas com agilidade até nas

[2] https://sisaps.saude.gov.br/esus/

[3] https://sismob.saude.gov.br/sismob2/#

[4] https://www.gov.br/anvisa/pt-br/assuntos/fiscalizacao-e-monitoramento/tecnovigilancia/rni

[5] https://sigresidencias.saude.gov.br/login

[6] https://www.gov.br/saude/pt-br/assuntos/rnds

[7] https://www.gov.br/mec/pt-br/jornadadoestudante

pequenas entregas. Para tanto, o Bridge possui uma série de iniciativas que buscam a qualidade dos produtos desenvolvidos no laboratório. Existe uma equipe dedicada aos processos de melhoria contínua, um plano de desenvolvimento individual, para desenvolver as competências dos colaboradores, pesquisa de clima organizacional, pesquisa e guia de referências, para buscar as pessoas que executam com excelência suas atividades, bem como mecanismos de feedback e indicadores em diversas áreas do laboratório;

- *Compartilhar dá mais XP (experiência)*: acreditamos que toda troca de conhecimento é uma oportunidade para subir de nível e maximizar resultados. Neste sentido, o Bridge possui um evento de mesmo nome, onde os membros do laboratório ou convidados externos trazem conhecimentos e experiências para compartilhar com os colegas;
- *Explorar novos mundos*: temos iniciativa e coragem em assumir o desafio de buscar soluções inovadoras que contribuam para os objetivos do Laboratório. Para tal valor, o laboratório possui a iniciativa *b_up*, um momento da carga horária de trabalho semanal de cada colaborador que pode ser dedicado a projetos inovadores;
- *Conectar pessoas a momentos*: somos uma equipe que valoriza a união e reconhece a importância de cada *bridger*[8]. Nosso dia a dia descontraído nos proporciona um ambiente de trabalho leve e flexível. Nosso nome, Bridge (ponte) caracteriza essa conexão. O laboratório possui diversos eventos recreativos internos, como Arraiá, Helloween, Bridge Day, entre outras confraternizações, tanto remotas quanto presenciais.

Esta cultura evoluiu até este ponto a partir de reuniões de planejamento estratégico realizadas com o grupo ao longo dos anos. Para a concepção desta cultura no laboratório, foram realizadas algumas dinâmicas. Para este projeto foram envolvidos profissionais do design, a equipe de gestão e mais alguns colaboradores selecionados, juntamente com um consultor externo.

As dinâmicas de construção de cultura tiveram como objetivo construir frases que gerassem identificação e comunicação a partir de jargões e protocolos que expressassem a coesão do grupo todo do laboratório a partir de valores compartilhados. Neste sentido, as dinâmicas giraram em torno da pesquisa por palavras comuns entre os membros do laboratório que expressassem seus sentimentos com relação ao dia a dia de trabalho e os objetivos do laboratório. Estas palavras compuseram as características culturais do laboratório e o conjunto destas palavras geraram as frases comunicativas que expressam a cultura Bridge, apresentadas anteriormente no texto.

5.3. Informatização da Saúde Pública

Nesta seção apresentamos os projetos do Bridge e seu impacto na gestão da saúde pública no Brasil.

5.3.1. e-SUS APS

O e-SUS APS (Sistema Único de Saúde Eletrônico – Atenção Primária à Saúde) é nosso maior projeto, iniciado em 2013 e já contratado até 2025, com possibilidade de continuar por mais alguns anos. Trata-se da informatização da saúde pública no Brasil iniciando pela atenção primária (Unidades Básicas de Saúde – UBS) [Lacerda et al. 2021].

[8] *Bridger* é como são chamados no dia a dia os membros do Laboratório Bridge.

O principal produto do projeto é o PEC (Prontuário Eletrônico do Cidadão), um sistema que vem sendo evoluído ao longo de quase dez anos e cobrindo cada vez mais as atividades da atenção primária e secundária à saúde. Além da informatização das UBS's, foi lançado recentemente o módulo voltado para o atendimento aos Centros de Especialidades Odontológicas (CEO) e está sendo desenvolvido o módulo de Atenção Ambulatorial Especializada (AAE), que envolve as Unidades de Pronto Atendimento (UPA), policlínicas e demais estabelecimentos.

O sistema foi pensado para ser útil em qualquer situação de informatização existente no território nacional, desde capitais com modernas redes de fibra até unidades básicas localizadas em barcos, que atendem comunidades indígenas ou ribeirinhas em locais onde até a energia elétrica é indisponível. Três diferentes perfis de instalação foram então definidos em função do grau de informatização das unidades de saúde, sendo que no estágio menos favorável as informações a serem coletadas podem ser escritas em fichas de papel que são depois digitadas em um sistema de Coleta de Dados Simplificada (CDS)[9]. Alguns números do e-SUS APS:

- mais de 43 mil UBS em funcionamento das quais 50% usam a solução PEC completa, 30% a solução CDS simplificada e 20% sistemas próprios com 100% de integração entre todos os sistemas;
- mais de 151 milhões de cidadãos cadastrados;
- mais de oito bilhões de registros clínicos e demográficos no sistema.

O sistema e-SUS APS ainda comporta os seguintes aplicativos mobile:

- e-SUS AD (Atenção Domiciliar)[10] para o programa "Melhor em Casa", no qual pessoas com necessidades não graves de atenção médica podem ser internadas na própria casa e receber a visita de profissionais de saúde regularmente para os exames e tratamentos necessários. Este programa inovador do Brasil liberou leitos em hospitais para os casos mais graves e promoveu melhor qualidade de vida aos internados que têm maior conforto e atendimento em suas próprias casas;
- e-SUS AC (Atividade Coletiva)[11] vencedor do Prêmio Nacional de Computação Aplicada à Saúde (SBCAS, 2019) [Gomes et al. 2019], que visa facilitar o trabalho das equipes de saúde em atividades coletivas, como por exemplo em escolas, nos consultórios de rua, na atenção à saúde prisional e nos núcleos de apoio à saúde da família;
- e-SUS Território[12] para os agentes comunitários de saúde e agentes de combate a endemias;
- Gestão e-SUS APS[13] para uso dos gestores municipais ou de UBS, permitindo ao gestor maior conhecimento sobre a situação e a evolução da saúde em sua área de atuação;
- e-SUS Vacinação[14], para registro da aplicação de vacina na APS, menção honrosa no Prêmio Luiz Fernando Gomes Soares (WEBMEDIA, 2021). Este aplicativo foi idealizado especialmente para o processo de campanhas de vacinação, sendo projetado para uso *offline-first* e com um processo mais simples e rápido de registro que considera

[9] https://cgiap-saps.github.io/Manual-eSUS-APS/docs/PEC/PEC_07_cds/

[10] https://www.gov.br/pt-br/apps/e-sus-ad

[11] https://www.gov.br/pt-br/apps/e-sus-ab-atividade-coletiva

[12] https://www.gov.br/pt-br/apps/e-sus-territorio

[13] https://portal.bridge.ufsc.br/2022/06/02/gestao-e-sus-aps-pec-conheca-o-aplicativo-que-facilita-o-acesso
-a-relatorios-da-saude/

[14] https://cgiap-saps.github.io/Manual-eSUS-APS/docs/VACINACAO

que nas campanhas de vacinação usualmente apenas uma vacina é disponibilizada e que diversas pessoas receberão a vacina de um mesmo lote em sequência.

Todos esses sistemas enviam dados a um Centralizador Nacional (e-SUS CN), sendo essa a maior e mais importante base de dados da saúde pública hoje no Brasil.

Além disso, o laboratório, visando promover ainda mais o sucesso da iniciativa e-SUS APS, disponibiliza aos usuários um serviço de suporte (e-SUS Suporte)[15] com atendimento por profissionais de saúde e tecnologia. A Figura 5.1 apresenta um resumo dos principais produtos do projeto e-SUS APS.

Figura 5.1. Produtos e aplicativos do projeto e-SUS APS.

Observa-se que apesar de o sistema e-SUS APS ser disponibilizado gratuitamente pelo ministério aos municípios, um número significativo de municípios ainda usa sistemas próprios por diversas razões. Inclusive, o número de municípios com sistemas próprios tem crescido, embora em proporção menor do que a do PEC, isso se dá pela possibilidade de estes sistemas de terceiros comunicarem seus dados ao ministério através do envio de arquivos Apache thrift ou XML para uma instalação do PEC. A Figura 5.2 mostra a evolução no número de municípios que usam a solução PEC, CDS ou sistema próprio. Enfatizamos que com a crescente informatização das UBS's a tendência é que o número de municípios usando a versão CDS tenda a zero e que essa solução seja descontinuada.

[15] https://esusaps.freshdesk.com/support/login

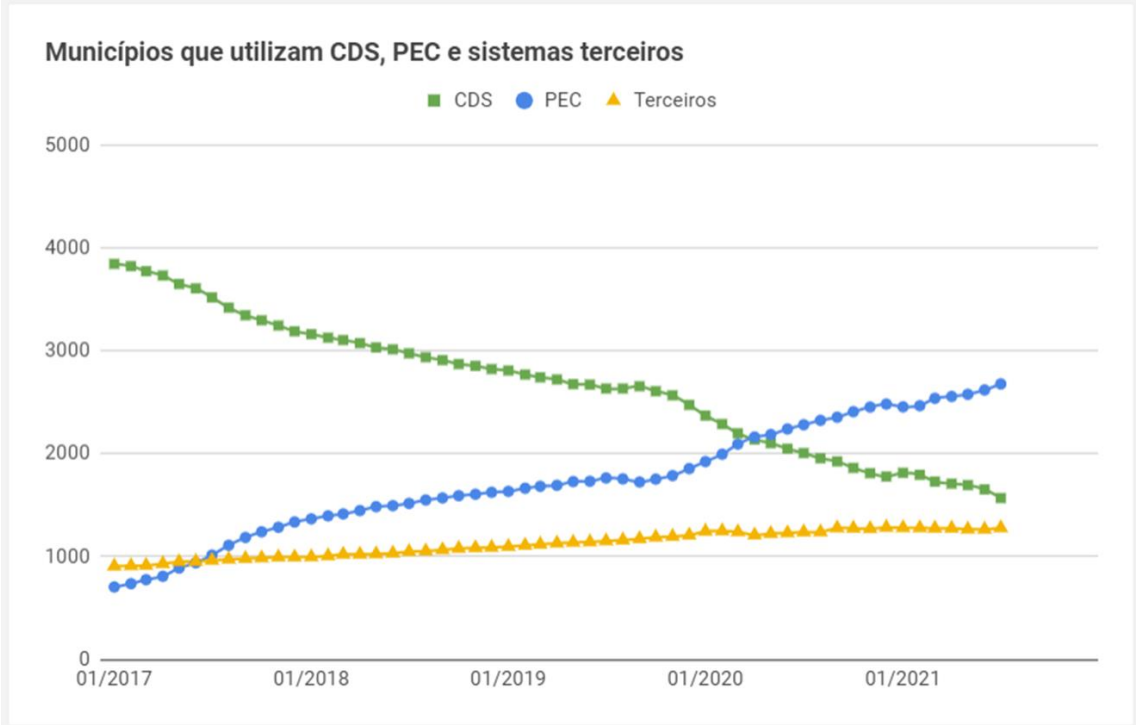


Figura 5.2. Número de municípios usando PEC, CDS ou sistema próprio na atenção primária à saúde.

O Projeto SIGRESIDÊNCIAS cuida de todo o processo de acompanhamento de residentes em saúde no Brasil contando com mais de 22 mil matrículas ativas e mais de 12 mil bolsistas cadastrados, sendo que os erros em pagamentos que antes eram de 80% caíram para zero. Dentro deste projeto foi também desenvolvida em tempo recorde a estratégia "o Brasil Conta Comigo" no início da pandemia de 2020, sendo este objeto de publicação no Boletim da OMS em 2022 [Celuppi et al. 2022]. Trata-se do maior banco de dados autodeclarado de profissionais de saúde do Brasil contando com mais de um milhão de cadastros.

O Projeto RNI foi desenvolvido especialmente para combater a máfia das próteses, aumentando o controle da ANVISA sobre próteses e órteses de quadril, joelho e stents coronários aplicados no Brasil.

O projeto SISMOB cuida da gestão, financiamento e execução de todas as obras em saúde financiadas pelo ministério, exceto hospitais. Além do controle facilitado para os gestores, o sistema também conta com uma versão web e mobile para que o cidadão possa acompanhar obras no seu município e denunciar irregularidades. Mais de onze bilhões de reais já foram repassados aos municípios a partir desse sistema. São mais de 27 mil obras inauguradas e mais de quatro mil em andamento.

5.4. Bold *Design System*

O Bold[16] foi desenvolvido pelos bridgers para melhorar e uniformizar a aparência e usabilidade dos produtos desenvolvidos no laboratório. O sistema tem foco na experiência do usuário (UX) e conta com paleta de cores, biblioteca de ícones, código executável e

[16] https://Bold.bridge.ufsc.br/pt/

especificações de uso. Acessível de forma completa e gratuita pela plataforma Sourceforge o Bold já foi utilizado por mais de 9 mil pessoas em mais de 130 países.

A necessidade para a construção de um novo *design system* surgiu a partir da carência por procedimentos padronizados na elaboração da camada visual dos produtos desenvolvidos pelo Bridge. Antes da elaboração do Bold, no início do projeto de um novo produto sempre eram elaborados novos estilos visuais para cada aplicação, dificultando a escalabilidade do design, visto que todo novo projeto possuía novos componentes com estilização distinta [Laboratório Bridge 2019b].

Ao estudar acerca dos produtos já desenvolvidos pelas equipes, foi verificado que os elementos visuais possuíam problemas relacionados com os padrões esperados pelas tecnologias utilizadas e que havia componentes inconsistentes. A partir do momento em que um componente visual era adicionado ao *front-end* da aplicação, em qualquer outro momento era possível que outro membro da equipe fizesse a sua alteração, modificando estilos e características visuais de elementos que deveriam ser padronizados, visto que possuíam a mesma funcionalidade no produto. Houve casos inclusive, em que havia componentes com 10 estilos distintos que possuíam a mesma funcionalidade.

Durante estes estudos, foi verificado também que os componentes visuais das aplicações não eram completamente acessíveis. Como se espera que um produto Web seja funcional para todas as pessoas, independente das suas capacidades pessoais ou do seu dispositivo, foi observada a necessidade de padronização visual dos produtos também sob o ponto de vista da acessibilidade [Laboratório Bridge 2019c].

A partir destas necessidades, foi definida a construção do que viria a ser o *design system* Bold. Para tanto, foi reunida uma equipe multidisciplinar com profissionais ligados às equipes de análise, design e desenvolvimento do Bridge, que construiu o Bold baseando-se em sete etapas [Laboratório Bridge 2019a]:

- levantamento de normas de usabilidade;
- listagem de componentes;
- levantamento de requisitos;
- pesquisa de referências;
- levantamento de requisitos de acessibilidade;
- construção dos componentes;
- documentação.

Além de atender as características esperadas de um *design system*, como possuir práticas compartilhadas condizentes com os objetivos do produto e ser composto por conjuntos de padrões interconectados entre si [Kholmatova 2017], o desenvolvimento do Bold procurou ser focado em usabilidade e na experiência do usuário. Se buscou também que ele fosse disponibilizado livremente para toda a comunidade e que fosse customizável e acessível para todos os usuários, independentemente das suas capacidades.

Para a implantação do Bold como *design system* oficial dos produtos do Laboratório Bridge, o seu uso foi iniciado durante o redesign do PEC (Prontuário Eletrônico do Cidadão). Esta foi a oportunidade de analisar a sua eficácia em produtos digitais. Desta forma, o Bold foi implantado no projeto enquanto o redesign do produto ainda estava em andamento. Assim, foi possível identificar as peculiaridades do *design system* frente a um produto real, facilitando a análise de pontos para a melhoria do Bold,

visando torná-lo o mais eficiente possível em termos de consistência e acessibilidade [Laboratório Bridge, 2019b].

5.5. Transformação ágil utilizando Scrumban e b_thinking

O Laboratório Bridge tem como visão ser a desenvolvedora das melhores soluções tecnológicas da gestão pública, por isso ele foca na qualidade dos produtos que desenvolve. Os atributos da qualidade dos produtos podem ser influenciados diretamente pela qualidade do processo de desenvolvimento [ISO/IEC 2010]. Embora não exista uma garantia de que um bom processo irá produzir um bom produto, geralmente a utilização de um bom processo resulta em produtos melhores do que utilização de um processo inadequado [Wazlawick 2019]. Existem várias vantagens em estabelecer o uso de um processo de desenvolvimento de software, entre elas, obter resultados mais previsíveis e de maior qualidade, redução do tempo de treinamento, produtos mais uniformizados, possibilidade de capitalizar experiências (atualizando o processo conforme boas práticas são identificadas) [Wazlawick 2019], dessa forma é de extrema importância definir e implementar processos adequados.

Existem diversos modelos de processos na Engenharia de Software, tanto prescritivos (que se baseiam na descrição de como as atividades devem ser feitas), quanto ágeis (que têm menos ênfase nas definições de atividades e mais ênfase na pragmática e nos fatores humanos do desenvolvimento). O método escolhido para ser utilizado deve ser adequado para o projeto e para a cultura da organização. Se a escolha for adequada, o processo será eficiente, caso contrário, o processo poderá gerar trabalho repetitivo e sua utilização será frustrante. Dessa forma, não há um modelo que seja melhor do que os outros em todos os contextos [Wazlawick 2019].

As próximas seções apresentam como o Laboratório Bridge definiu e implementou um processo de desenvolvimento ágil e centrado no usuário.

5.5.1. Experiência do Bridge até a adoção do Scrumban

Em 2013, quando o Laboratório Bridge iniciou suas atividades, os projetos eram desenvolvidos utilizando o Modelo Cascata. Esse método era refletido inclusive na estrutura física do Laboratório, que possuía salas diferentes para os papéis de analista de sistemas, programadores, designers e analistas de qualidade. Como resultado do trabalho desenvolvido em fases, os projetos não conseguiam fazer entregas frequentes e as soluções eram engessadas, uma vez que o método não facilita a realização de mudanças ao longo do desenvolvimento. A utilização do Cascata impactava também no desenvolvimento de soluções mais criativas/inovadoras, pois não promovia a troca entre os diferentes papéis do projeto, chegando ao ponto de surgirem filas para que os programadores pudessem conversar ou esclarecer dúvidas com os analistas. Nesse contexto foi percebida a necessidade de buscar soluções para tais problemas, foi quando o Laboratório decidiu começar uma transformação ágil, iniciando com a adoção de modelos ágeis de desenvolvimento.

Apesar dos modelos ágeis serem mais leves e menos burocráticos que métodos tradicionais, isso não significa que a sua adoção não apresente desafios. [DIGITAL.AI. 2021], isso porque, os hábitos e cultura das empresas costumam estar enraizados nos colaboradores e isso pode gerar resistência a mudanças. Além disso, durante o período de transição é comum que os processos e práticas entre as equipes fiquem inconsistentes,

quando as mudanças não são feitas de forma linear, o que pode aumentar a necessidade de gerenciamento das equipes. Outros desafios que podem surgir são a falta de colaboradores com experiência em métodos ágeis, participação insuficiente da liderança e treinamentos insuficientes.

5.5.2. Como o Laboratório Bridge superou esses desafios?

Em 2015, o Bridge reuniu líderes e referências de área em uma série de reuniões que tinham como objetivo estudar métodos ágeis e planejar as mudanças que seriam implementadas. O grupo identificou as características e boas práticas propostas em diversos métodos ágeis, e analisou aquelas que possuíam maior potencial para impactar positivamente os resultados dos projetos.

Em um momento inicial o objetivo do Laboratório não era instituir cerimônias ou outras práticas de algum framework específico, mas sim, melhorar a comunicação e aumentar a colaboração e entre bridgers de áreas diferentes e entre os bridgers e os clientes.

Para aumentar a qualidade do produto e realizar entregas mais frequentes: os bridgers foram alocados em equipes de desenvolvimento multidisciplinares e autossuficientes. Inicialmente foi feito um piloto com uma única equipe e, uma vez que foram observados resultados positivos, o modelo foi replicado nas demais equipes.

Para identificar e solucionar problemas mais cedo, foi apresentada uma nova proposta de comunicação aos clientes, a qual possibilitaria alinhamentos de forma diária. Dessa forma, seria possível que o cliente acompanhasse a evolução das atividades e comunicasse rapidamente caso identificasse algum problema.

O novo formato de trabalho, em equipes ágeis e novo modelo de comunicação com o cliente despertou o interesse em adotar um método ágil. Durante o período de 2016 a 2019 foram realizados treinamentos e a incorporação de cerimônias, papéis e práticas do Scrum e Kanban. Algumas das melhorias realizadas foram:

- capacitação sobre métodos ágeis para bridger e clientes;
- além das reuniões diárias, todas as equipes ágeis passaram a realizar reuniões de planejamento, retrospectiva e revisão;
- capacitação sobre o método Kanban;
- equipes começam a aplicar princípios do Kanban;
- equipes estabeleceram a rotina de monitoramento de métricas de performance;
- foram definidos os papéis de Scrum Master e Product Owner em todas as equipes ágeis.

Uma vez que o método Scrumban (uma combinação de Scrum e Kanban) estava estabelecido nas equipes, a equipe de Melhoria Contínua e os Scrum Masters do Laboratório identificaram a necessidade de mensurar a maturidade ágil das equipes. O objetivo da avaliação era identificar pontos de melhoria em cada uma das equipes. Após pesquisar na literatura sobre ferramentas de avaliação de maturidade ágil, foi escolhida a ferramenta Roda Ágil [GOMES SOARES 2017]. A Roda Ágil avalia 4 pilares: valor a todo instante (relacionado à geração de valor e satisfação do cliente), pessoas sensacionais (relacionado à autogestão da equipe), segurança (relacionado às práticas de desenvolvimento dos projetos) e experimente e aprenda rápido (relacionado ao uso das práticas ágeis). A aplicação da avaliação com Roda Ágil é realizada trimestralmente em

todas as equipes. Já foram realizados sete ciclos de avaliação. A partir dos resultados da avaliação cada equipe desenvolve planos de ação para o trimestre seguinte, criando-se assim, uma cultura de melhoria contínua dos processos utilizados nas equipes.

5.5.3. Utilização do b_thinking: modelo de referência de processo de UX

O processo de desenvolvimento pode influenciar diversas características de qualidade que impactam a Experiência do Usuário (UX), como usabilidade, adequação funcional, efetividade, satisfação, eficiência [ISO/IEC 2010]. Por isso, em 2019, o Laboratório Bridge iniciou um programa de melhoria com o intuito de amadurecer o processo de UX, ou seja, amadurecer os processos já existentes de levantamento de requisitos, análise e projeto.

Na primeira etapa foi realizada uma avaliação do processo de UX do Laboratório. A avaliação foi realizada em todos os projetos individualmente, por meio de reuniões, das quais participavam os analistas de sistemas e designers. O objetivo das reuniões era identificar pontos positivos e oportunidades de melhoria dos processos. Para realizar a avaliação foi utilizado o UPCASE [Lacerda et al. 2019], um método para autoavaliar a capacidade de processos de usabilidade em pequenas empresas. Ao final de cada avaliação, além de receber uma pontuação em relação à capacidade do processo, o UPCASE apresentava um resultado com que continha uma lista de indicadores sobre os aspectos do processo que poderiam ser melhorados.

Os resultados das avaliações mostraram que várias equipes já envolviam usuários no início do processo de concepção, analisavam o contexto das tarefas de forma sistemática, realizavam atividades de ideação cooperativa e coletavam feedback sobre as releases dos produtos. Por outro lado, foi verificado que essas atividades ainda não eram realizadas em todas as equipes. Foram identificadas as necessidades de promover a capacitação dos novos bridgers sobre conceitos básicos de design centrado no usuário e técnicas e ferramentas que podem ser usadas no processo de concepção dos produtos; e aumentar a integração das equipes ágeis neste processo. Com base nos resultados da avaliação foram definidas algumas ações:

- *criação de templates*: para facilitar a adoção de novas atividades, como entrevistas, testes de usabilidade, entre outros;
- *capacitação dos colaboradores*: para aproximar bridgers de todas as áreas sobre conceitos fundamentais do processo, como, maturidade de processo de UX, entrevista com usuários, UX metrics;
- *comunicação de atividades desenvolvidas*: para engajar mais equipes na utilização de novas ferramentas, foram definidos canais de comunicação para informar sobre atividades que estavam sendo realizadas nas equipes, como a realização de testes, entrevistas e seus resultados.
- *definição de um modelo de processo*: para promover um amadurecimento de processo uniforme no Laboratório foi criada uma iniciativa para desenvolver o *b_thinking*, um modelo de processo de UX, que poderia ser utilizado como referência pelas equipes. Para construir o modelo, um grupo de Product Owners e designers do Laboratório estudou diferentes processos, e fez pesquisa de *benchmarking* com empresas que são referências no mercado.

5.5.4. b_thinking: o modelo de referência

O modelo de referência foi construído levando em consideração a cultura e o contexto dos projetos desenvolvidos no Laboratório, e teve como meta ser um material didático, assim, até os bridgers mais novos na área poderiam aprender sobre o tema e nortear o seu trabalho conforme o modelo. Por isso, o b_thinking foi construído de forma colaborativa contando com a participação de diversos bridgers.

O objetivo do b_thinking é servir como um "mapa", que pode guiar as equipes durante o desenvolvimento de uma solução. Mas, o objetivo é orientar e não estabelecer um processo de trabalho engessado. Por isso, cada equipe tem flexibilidade para definir as etapas e métodos que precisa utilizar. Ele norteia o processo de criação dos produtos, auxiliando a:

- identificar e definir e o escopo de problemas complexos;
- criar hipóteses para resolução do problema identificado;
- desenvolver protótipos e validá-los, pautando assim as escolhas em dados empíricos;
- transicionar entre as etapas de descoberta e entrega, delimitando o problema e refinando-o com o time de desenvolvimento;
- entregar as funcionalidades, trazendo o time para o centro das principais decisões;
- avaliar as entregas para descobrir quais melhorias ainda precisam ser realizadas no sistema.
- criação de produtos que geram valor aos usuários e satisfazem os stakeholders envolvidos.

O *b_thinking* é composto dos seguintes componentes:

- *princípios*: Para desenvolver produtos que gerem valor, não basta apenas aplicar diferentes métodos e seguir etapas específicas de um método, é importante adotar um novo *mindset*, incorporando princípios do Lean UX [Gothelf 2021] para poder extrair o maior valor possível do processo. Esses princípios são:
    - *equipes multifuncionais, mas pequenas*: é importante ter colaboração entre diferentes disciplinas. Isso evita o efeito cascata, uma vez que as ideias são discutidas com todos os stakeholders. Equipes devem ser pequenas, para permitir uma comunicação mais eficaz e manter todos atualizados sobre mudanças e novos conhecimentos. As equipes recebem problemas para resolver e não funcionalidades para desenvolver;
    - *sem desperdícios, foco no resultado*: os resultados visam entregar valor aos stakeholders. Atividades que não agregam valor são despriorizadas. Foco no MVP (Mínimo Produto Viável) e entregas constantes incrementando o produto com base nas necessidades dos usuários;
    - *conhecimento é compartilhado, trabalho é colaborativo*: a equipe amadurece junto em relação ao produto, cliente e usuários. É essencial ter boa comunicação e expressar as ideias e conhecimentos com clareza. Gurus que querem brilhar sozinhos e não compartilham informação não têm espaço nessa equipe;
    - *testar antes, escalar depois*: é importante testar as hipóteses antes de escalá-las. Deve-se ter liberdade para experimentar soluções

sem ser penalizado se as ideias não tiverem sucesso. É criado apenas o MVP, garantindo economia de tempo e recursos.

- *etapas*: durante o desenvolvimento do produto, as etapas orientam as equipes a percorrer uma série de atividades que visam desde compreender o problema até identificar e avaliar as soluções que mais agregam valor para os clientes e usuários. Cada etapa possui saídas definidas e está associada a um conjunto de atividades. Conforme apresentado na Figura 5.3, o *b_thinking* possui as seguintes etapas:
  - *briefing*: identificar e definir o escopo da demanda, entregáveis, prazos, stakeholders, público-alvo e critérios de sucesso do produto;
  - *imersão*: pesquisar sobre o perfil dos usuários, quais tarefas realizam, quais as suas dores e necessidades, e qual o contexto que os cerca;
  - *definição*: analisar e sintetizar as descobertas priorizando os problemas do usuário e os padrões de comportamento que devem nortear o desenvolvimento do produto;
  - *ideação*: gerar ideias com a equipe para solucionar os problemas priorizados;
  - *prototipação de ideias*: produzir versões simplificadas do produto que permitam testar as ideias geradas;
  - *validação de ideias*: verificar por meio de experimentos quais ideias são mais adequadas para solucionar o problema;
  - *delimitação*: delimitar quais ideias serão implementadas de acordo com o seu valor, custo-benefício e prazo de entrega;
  - refinamento: detalhar os requisitos do sistema e produzir protótipos de alta fidelidade das soluções delimitadas;
  - *sprint*: implementar parte de uma solução que gere valor aos usuários;
  - *avaliação da release*: avaliar o uso do produto após a sua release para verificar se ele atendeu aos critérios de sucesso;
  - *melhoria contínua*: aprender com o *feedback* dos usuários e dados coletados, gerando ações de melhoria contínua do produto.

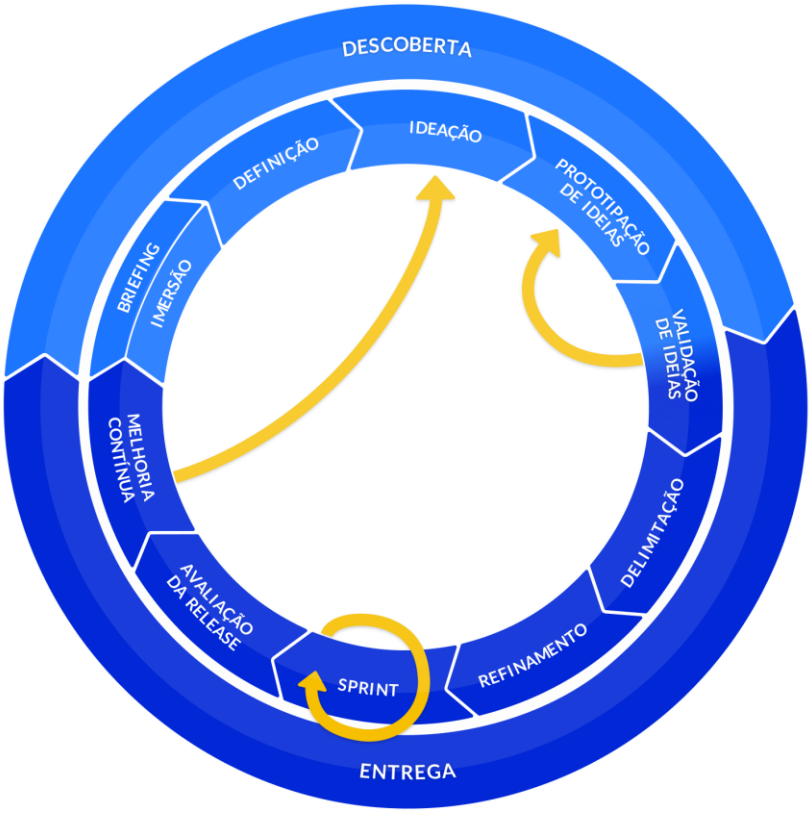


Figura 5.3. *b_thinking*

- *métodos e ferramentas*: para cada uma das suas etapas, o b_thinking descreve diversos métodos e ferramentas que podem ser utilizados, juntamente com suas respectivas instruções de uso. A lista de métodos e ferramentas pode ser consultada no site https://laboratoriobridge.github.io/bthinking/pt/intro. O material está sendo evoluído continuamente, conforme nossas equipes de produto experimentam novos métodos;
- *templates e guias*: para facilitar a utilização dos métodos e ferramentas, o b_thinking contém templates e guias.

O b_thinking foi construído com o objetivo de amadurecer o processo de UX do Laboratório Bridge. Por isso, além de servir como modelo de referências de processo, ele também é utilizado como uma ferramenta para aumentar o conhecimento de toda a equipe sobre UX. Observamos que o b_thinking:

- facilita o *onboarding* de novos bridgers;
- traz clareza para toda a equipe sobre o processo de criação dos produtos;
- facilita a comunicação sobre as atividades realizadas;
- cria uma base de conceitual comum entre bridgers e clientes;
- facilita a troca de conhecimento sobre o processo de UX;
- resulta no amadurecimento das nossas equipes.

Tudo isso leva ao amadurecimento da organização em relação ao processo de construção de produtos.

5.6. Ferramentas de Gestão Estratégica

Visando possibilitar o funcionamento ágil do Laboratório Bridge, outras ferramentas de gestão foram adotadas pelos setores e pelas equipes ágeis. As seções a seguir apresentam estas ferramentas e como elas são utilizadas dentro do laboratório.

5.6.1. Gestão de Pessoas

No Bridge o setor de gestão de pessoas faz parte do núcleo de gestão e é responsável pelos objetivos ligados à gestão do capital humano do laboratório, facilitando os processos visando que o ambiente, a convivência e a cultura organizacional sejam cada vez melhores. A gestão de pessoas atualmente não trata unicamente de processos operacionais, pois cada vez mais ela se envolve com aspectos estratégicos, visto que o fator humano nas organizações atualmente está cada vez mais valorizado [Silveira 2021].

Sendo assim, dentre as várias atribuições do setor de gestão de pessoas do laboratório, está a de atração e seleção de novos talentos para atuarem profissionalmente nas posições de trabalho do laboratório. Para tanto, a gestão de pessoas tem a atribuição de criar as definições de o que é ser um bridger e o que cada bridger fará em cada função, elencando os benefícios e atribuições deste profissional. É a gestão de pessoas que trabalha diretamente com a divulgação para a atração de talentos para as vagas. Além disso, todas as etapas dos processos seletivos para bolsistas e celetistas do laboratório são planejadas visando a aplicação dos testes, entrevistas, análises, dinâmicas, bem como os feedbacks para todos os candidatos.

Faz parte das atribuições da área de gestão de pessoas do laboratório, também manter os cuidados relacionados com o clima organizacional do Bridge. Este setor é o responsável pelas análises voltadas às pesquisas de clima organizacional e o eNPS (*Employee Net Promoter Score*) do Bridge, visualizando como está o ambiente atual de trabalho e o quanto os colaboradores indicariam o Bridge para outras pessoas. O setor também está diretamente envolvido com a organização dos eventos internos do laboratório, visando a conexão e a integração dos colaboradores. Dentre os eventos, está o Bridge Day, evento anual do laboratório, bem como outros eventos, torneios e demais comemorações [Silveira 2021].

O setor também elabora as iniciativas relacionadas com a qualidade de vida dos colaboradores, visto que a saúde mental e física dos profissionais influencia diretamente no resultado do trabalho de cada um. Sendo assim, regularmente o setor organiza eventos de promoção de saúde, como o projeto mensal *b_healthy*. Também são desenvolvidas parcerias com profissionais de saúde, academias, restaurantes, realização de dinâmicas, informativos e treinos coletivos.

O plano de desenvolvimento individual dos colaboradores, outra ferramenta sob responsabilidade da gestão de pessoas do laboratório, é o modo de auxiliar na condução da jornada do bridger dentro do seu trabalho. Neste plano, a partir do mapeamento prévio que o setor fez a partir das atividades relacionadas com diferentes disciplinas do seu perfil profissional, é possível compreender quais são as competências relacionadas com o trabalho de cada bridger. A partir disso, estas competências são avaliadas e discutidas constantemente com o colaborador e a sua pessoa de referência, auxiliando no seu desenvolvimento profissional dentro do laboratório [Silveira 2021].

5.6.2. Pesquisa de Clima Organizacional

Visando o acompanhamento do nível de satisfação dos membros do laboratório, o setor de gestão de pessoas elaborou um projeto de pesquisa de clima organizacional, com

periodicidade anual. A pesquisa de clima organizacional é um instrumento anônimo, onde os colaboradores podem reportar à gestão quais são as suas reais impressões acerca do ambiente de trabalho na organização [Lacerda e Soares 2022].

Desta forma, através deste tipo de pesquisa, fica possível o acesso a informações seguras sobre como está a percepção atual dos membros do laboratório acerca do dia a dia de trabalho e onde são necessários ajustes para potencializar o clima organizacional. No laboratório esta pesquisa possui 33 afirmativas, com respostas baseadas na escala de Likert, tendo a possibilidade de comentários textuais em cada uma [Lacerda e Soares 2022]:

- nunca é verdade;
- na maioria das vezes é verdade;
- às vezes é verdade, às vezes não é;
- na maioria das vezes é verdade;
- sempre é verdade.

Além das afirmativas, são realizadas três outras perguntas e também é mensurado o eNPS Interno, que avalia o quanto os colaboradores indicariam o Bridge para outras pessoas. Todas as questões envolvem aspectos gerais do laboratório, abrangendo pontos como os relacionamentos interpessoais, desenvolvimento, políticas do laboratório e transparência [Lacerda e Soares 2022].

Com base nos resultados da pesquisa, os dados são normalizados em índices de 0 a 100, que servem como parâmetros para a tomada de decisão no laboratório. Estes dados são avaliados com base nas pesquisas realizadas nos anos anteriores, visando identificar aumentos e quedas nos índices para a compreensão do clima interno do laboratório [Lacerda e Soares 2022].

5.6.3. Jornada Mapeada

Os bridgers podem ser bolsistas ou celetistas. Por ser um laboratório ligado à UFSC, os bolsistas são alunos da universidade que se dedicam por tempo parcial às atividades do Laboratório Bridge, no contraturno do período estudantil. Com o término do contrato, ou com a finalização dos seus cursos na UFSC, o aluno deixa de ser bolsista. Em alguns casos estes alunos rumam para atividades profissionais em outras organizações, ou são contratados efetivamente como celetistas no laboratório. Porém, há também a possibilidade da contratação direta de profissionais celetistas, neste caso, especialmente para profissionais já experientes.

Visando o desenvolvimento dos bridgers, em especial, dos que são bolsistas da universidade, o setor de gestão de pessoas implantou o conceito de *jornada mapeada*, uma forma de visualização do ciclo de experiência do estudante enquanto bolsista em suas atividades nas equipes do laboratório. A iniciativa foi desenvolvida a partir da percepção de que o bolsista apresentava algumas dúvidas durante a sua estada no laboratório.

Para a elaboração deste mapa da jornada do bolsista, foram realizadas pesquisas qualitativas com os bolsistas, que envolviam questionamentos acerca das suas experiências atuais no laboratório, procurando identificar como é o seu dia a dia, a sua rotina e as suas experiências no Bridge, sejam elas positivas ou negativas. Na pesquisa, também foram levantadas as possíveis experiências anteriores do bolsista, fora do laboratório, identificando quais pontos estes valorizavam e o que procuravam em uma vaga profissional. Por fim, nesta pesquisa qualitativa também foram levantadas informações sobre as perspectivas do bolsista para o futuro, como quais são os seus

objetivos, planos de vida e de carreira. Para compor o mapa também foram utilizados dados que já haviam sido levantados em outros instrumentos do laboratório, como na pesquisa de clima organizacional, pesquisas de desligamento de bolsistas e em perguntas destinadas ao Quick F5!, um evento de alinhamento entre todos os membros do Laboratório, realizado mensalmente.

Com base neste levantamento foi consolidado o padrão dos perfis dos bolsistas do Bridge e a partir de então foi desenhado o mapeamento da jornada destes no laboratório. A jornada, com três momentos, é dividida entre pré Bridge, jornada do bolsista e saída. O momento *pré Bridge* é antes da entrada do bolsista no laboratório, ele trata do processo seletivo relacionado com a vaga. A *jornada do bolsista*, segundo momento, se refere ao período de vigência do contrato com o laboratório, desde o seu *onboarding* até as suas atividades dentro das equipes. Já a *saída*, é o momento de finalização do contrato, onde o bolsista é desligado ou contratado como celetista.

O segundo momento desta jornada, é quando o bolsista está envolvido com o laboratório. Este momento é dividido em etapas. Inicialmente, as primeiras etapas são semanais. Na primeira semana, o bolsista recebe as orientações, conhece a equipe e participa das reuniões de onboarding e de integração. Na segunda semana, o bolsista se integra ao laboratório e inicia a compreensão acerca da sua dinâmica de trabalho. As próximas etapas são dos próximos três meses, onde o bolsista será direcionado sobre as suas tarefas, recebendo o suporte necessário.

Seguindo na jornada, aos seis meses como bolsista, este deve estar evoluindo em suas atividades e se sentindo mais autônomo em suas atividades. Ao final do primeiro ano como bolsista, este deve estar compreendendo as metodologias utilizadas no laboratório, estando adaptado a elas. Com um ano e seis meses de trabalho, o bolsista deve estar usufruindo das oportunidades com mais tranquilidade, executando as tarefas com mais conhecimento. Tendo dois anos como bolsista, o indivíduo já deve estar pensando no trabalho de forma mais estratégica, bem como no futuro da sua carreira. Todos os momentos e etapas da jornada estão mapeados com as expectativas, ações e pontos de contato. Além disso, o bolsista continua sendo orientado e acompanhado pelo setor de gestão de pessoas, bem como pela sua pessoa de referência.

5.6.4. Planejamento Estratégico

Com o crescimento do Laboratório Bridge, tanto em termos de pessoas, quanto em termos de projetos, surgiu a necessidade da concepção de um setor capaz de organizar áreas como gestão de pessoas, integração de colaboradores e pesquisas organizacionais. Para tanto, a partir do projeto SISMOB (Sistema de Monitoramento de Obras), em 2015 o Bridge iniciou sua percepção como organização. Desta forma, em 2016 foi realizado o primeiro planejamento estratégico, onde foram definidos pontos como a missão, visão e valores do laboratório [Lacerda 2022].

Com o ingresso de novos colaboradores e novos projetos, tendo ainda as mudanças aceleradas pelos acontecimentos da pandemia por Covid-19, surgiu em 2020 a necessidade de um novo planejamento estratégico, visando posicionar o laboratório frente a estes cenários. Sendo assim, em 2021 foi realizado um novo planejamento estratégico, com foco no triênio seguinte, a partir do alinhamento entre as lideranças e agora com um foco em objetivos e metas.

Este último planejamento estratégico foi realizado de forma totalmente remota, para tanto, contou com uma série de peculiaridades visando o engajamento e a participação dos envolvidos. Todo o processo iniciou com um planejamento inicial e com

a comunicação acerca do processo pelos canais de comunicação utilizados no laboratório. A execução do planejamento foi baseada em apresentações interativas e dinâmicas integrativas entre os envolvidos, utilizando ferramentas de quadros virtuais online, tendo ao final do processo, um foco grande na comunicação interna para todos do laboratório utilizando as ferramentas internas de comunicação e os eventos remotos síncronos [Hammes *et al*. 2021].

5.6.5. Sistema de Gestão por OKR

Desde o primeiro planejamento estratégico executado pelo Laboratório Bridge, foi observada a necessidade de um acompanhamento de objetivos e metas. Sem este tipo de metrificação havia dificuldades na mensuração do impacto das ações realizadas e se observavam impedimentos relacionados com as tomadas de decisão. Tudo isso dificultava o rastreamento do real estado do processo do planejamento estratégico [Lacerda 2022].

Para tanto, desde 2018 foi implantada a gestão de objetivos e resultados chave por OKR (*Objectives and Key-Results*). Este modelo de gestão de metas e objetivos é baseado em duas perguntas principais: "Aonde queremos chegar?" e "Como posso saber se estou chegando lá?". A primeira pergunta responde sobre qual objetivo está sendo traçado, já a segunda, delimita quais são as metas (resultados chave) que precisam ser atingidas para alcançar este objetivo. Com o processo de definição de objetivos e dos seus respectivos resultados chave, é possível desdobrar objetivos maiores, como as elencadas no planejamento estratégico, em objetivos bem definidos. A partir de então, visando a rastreabilidade destes objetivos, com a delimitação dos resultados chave (*key results*), é possível a visualização do quanto cada objetivo está próximo do seu pleno atendimento [Grove 2015].

O processo de implantação da gestão por OKRs foi baseado em um projeto de treinamentos internos dos colaboradores do laboratório, enquanto de forma simultânea as metas e objetivos do projeto eram delimitadas. Porém, a implantação da gestão por OKRs como um todo no laboratório se deu a partir de 2019, com a elaboração de OKRs para mais áreas do Bridge, treinamento intenso dos colaboradores na metodologia. Com o planejamento estratégico de 2021 do laboratório, novos objetivos e resultados chave foram definidos, com um OKR da visão, com prazo de 3 anos desdobrado em OKRs menores, com prazos de um ano ou três meses [Lacerda 2022].

Toda a estruturação do programa de gestão baseada em OKRs do laboratório foi desenvolvida pela equipe de melhoria contínua do Bridge, um setor integrante do núcleo de gestão. Neste sentido, esta equipe é quem organiza as etapas que as áreas do laboratório executam para a operacionalização da gestão por OKRs no laboratório. Por conta das suas particularidades de trabalho, cada área adota atividades distintas para a execução dos ciclos de OKRs, porém, há uma sequência básica de etapas que costuma ser adotada pelas áreas [Lacerda 2022].

A primeira etapa é a de planejamento, onde comumente, no evento Quick F5!, de forma síncrona é comunicado o momento de atualização da planilha com os OKRs e os planos de ação do ciclo atual. Na segunda etapa, a de execução e acompanhamento, onde as metas e objetivos são executados, as áreas acompanham as atualizações da planilha de OKRs e participam de uma pesquisa de implementação do ORK. E na terceira e última etapa, a de revisão do ciclo, os bridgers participam do evento de *review* do ciclo de OKR e de uma nova pesquisa de implementação do OKR. Durante as duas últimas etapas, a pesquisa de implementação do OKR serve para mensurar o engajamento dos

colaboradores no processo de gestão por OKRs, onde eles respondem perguntas acerca do quanto colaborou ou se sentiu motivado nos seus objetivos e metas, por exemplo [Lacerda 2022].

5.6.6. Comunicação

Visando facilitar os processos de comunicação, tanto de forma interna quanto externa, o Laboratório Bridge possui uma equipe de comunicação que elabora as estratégias de mídia para o dia a dia da organização.

Neste sentido, a equipe de comunicação trabalha diretamente com a criação de conteúdo para as mídias sociais, envolvendo a escrita, design e manutenção das redes sociais. Isso se aplica também nas divulgações externas de eventos e processos seletivos do Bridge, estudando o público-alvo, temas, identidades visuais e demais elementos, além do acompanhamento de métricas e de inscrições.

A equipe trabalha diretamente com a manutenção das linhas editoriais de conteúdo nas redes sociais e no blog[17] do portal do Bridge, onde são publicados textos divulgando produtos, processos e *cases* de sucesso. Os comunicados externos e internos também passam pela revisão textual do setor de comunicação, bem como toda identidade visual de peças com circulação externa e interna e as apresentações do laboratório são desenvolvidas e revisadas pelo setor.

5.6.7. Trabalho Remoto

Com a necessidade de distanciamento devido à pandemia de Covid-19, em março de 2020 foi necessário que o laboratório se adaptasse às formas de trabalho remoto, inicialmente, em caráter emergencial. Para tanto, as ferramentas de trabalho, produção e comunicação interna do laboratório precisaram ser potencializadas e algumas ações foram necessárias [Lacerda 2022].

Uma destas ações foram as relacionadas com os momentos em que os membros das equipes precisavam se reunir de forma síncrona. Porém, as ferramentas utilizadas até o momento suportavam apenas 20 bridgers ao mesmo tempo. Desta forma, foi necessário adequar as ferramentas para os mais de 100 colaboradores do laboratório. A partir de então, as reuniões e os eventos internos do laboratório como o Compartilhar dá +XP e o Quick F5! foram capazes de ser realizados mesmo remotamente.

Para as operações de produção das equipes, os quadros de Kanban, antes físicos, baseados em quadros e post-its, foram substituídos por ferramentas online de quadros, como o Trello. Toda a comunicação entre as equipes foi padronizada para o uso do Slack em todo o laboratório, dividindo equipes, projetos e necessidades por canais.

Tendo diversos profissionais trabalhando de modo remoto, foi importante deixar clara a cultura de trabalho autônomo de cada indivíduo e o modo de gestão auto-organizado de cada equipe. Como mesmo antes do período de isolamento já havia planos para oferecer modalidade remota de trabalho para os colaboradores do Bridge e a cultura de confiança mútua já estava presente no laboratório, foi trivial a mudança de foco do trabalho físico na sede do laboratório para as ferramentas como o Slack, Google Workspace, Github, entre outras [Lacerda 2022].

Porém, com a volta das atividades presenciais na universidade, a partir de abril de 2022 foi necessário organizar as políticas de trabalho no laboratório. Para tanto, conforme já estava definido, graças à experiência exitosa no período de distanciamento, o

[17] https://portal.bridge.ufsc.br/blog

laboratório assumiu como política oficial de trabalho o *remote first* (remoto primeiro). Esta modalidade de trabalho prioriza o trabalho remoto, mas não exclui o trabalho presencial. Sendo assim, o laboratório está preparado com ferramentas e métodos para suportar o trabalho totalmente remoto, porém, também continuará dispondo de uma sede física para quem optar pelo trabalho presencial.

Neste sentido, os bridgers podem escolher se atuarão de forma presencial ou remota. Isso facilita na organização do dia a dia de cada colaborador e permite o alcance de talentos que presencialmente não poderiam atuar em um posto de trabalho em Florianópolis. Os bridgers que optaram pela modalidade presencial possuem sua estação de trabalho na sede e trabalham todos os dias na sede do laboratório. Os que preferiram pelo modo remoto, podem atuar a partir de qualquer local, sendo possível atuar em até dois dias por semana presencialmente no laboratório, mediante agendamento prévio. Havendo necessidade de alternância entre as modalidades de trabalho, é possível trocar, visando adequar todos os colaboradores às suas realidades [Lacerda 2022].

5.6.8. Relacionamento com o Cliente

O Laboratório Bridge sempre esteve preocupado com o relacionamento com o cliente entendendo que isso é fundamental para o desenvolvimento e sucesso de um projeto. Desta forma, a relação com o cliente nunca esteve baseada unicamente no formato cliente e fornecedor, mas sim, na relação de colaboração e parceria, compreendendo que todos devem estar alinhados e trabalhando com sinergia para a evolução adequada, cumprimento das metas e entrega dos produtos.

Visando estabelecer esta relação, o bridge possui um membro do *C-Level* dedicado na garantia da experiência do cliente, o CXO (*Chief Experience Officer*). O papel deste profissional é o de manter contato frequente com representantes do cliente responsáveis pela gestão do projeto para mantê-los informados sobre o andamento das atividades, alterações no cronograma, desembolso de recursos, estabelecimento de estratégias, elaboração de documentos, processos burocráticos, relacionamento entre equipes e demais assuntos que possam impactar a transparência e satisfação do cliente em relação à execução do projeto.

Nesse contexto, o Bridge se dedica em estreitar o relacionamento com o cliente desde a concepção do projeto até seu encerramento. Sendo assim, todos os stakeholders interagem trocando informações e conhecimento ao longo do período de execução do projeto, dando celeridade e promovendo a integração das equipes. O Bridge pesquisou e implementou canais, métodos e instrumentos para facilitar a integração entre os envolvidos, garantir maior transparência nas ações e dar credibilidade aos indicadores de produção. O principal canal de comunicação preferido pelos clientes é o Whatsapp. Seu uso é amplo o que resulta em rápida adesão por todos os envolvidos no projeto. Um dos métodos implantados foi a realização de reuniões semanais e mensais para alinhamentos técnicos e de gestão.

Porém, independente destes canais, as equipes ágeis possuem total liberdade para entrar em contato com os clientes para agendar reuniões e discutir sobre questões técnicas necessárias para o andamento das atividades. Alguns instrumentos compartilhados com os clientes tratam da gestão das atividades de desenvolvimento, documentação do projeto e relatórios de gestão. A equipe de Gestão do Bridge ainda aplica a pesquisa “Satisfação Cliente Bridge - SCB”, para avaliar o que é necessário aprimorar no processo de trabalho.

5.6.9. Relatórios de Evolução

A equipe técnica do Bridge mantém atualizado e compartilhado um arquivo contendo todas as demandas previstas com seus respectivos prazos de entrega. Isso garante um acompanhamento diário em relação à execução das atividades já desenvolvidas, em desenvolvimento e as previstas para desenvolvimento. Porém, essas informações são detalhadas e técnicas e podem sofrer alterações a qualquer momento, dificultando o acompanhamento por parte dos gestores.

Nesse sentido, para facilitar a transparência e manter todos os gestores informados sobre a execução do projeto, envolvendo o cliente, a universidade e o laboratório, foi desenvolvida uma metodologia para gerar indicadores mensais de execução dos produtos e metas estabelecidas nos instrumentos de contrato. Esses indicadores são consolidados e os resultados alimentam o Relatório de Evolução Mensal, que apresenta os percentuais de execução do projeto, de cada meta e de cada produto referente a cada mês, além de gráficos para facilitar o entendimento e visualização das informações. Um exemplo é o gráfico de execução *burn-up*, mostrado na Figura 5.4.

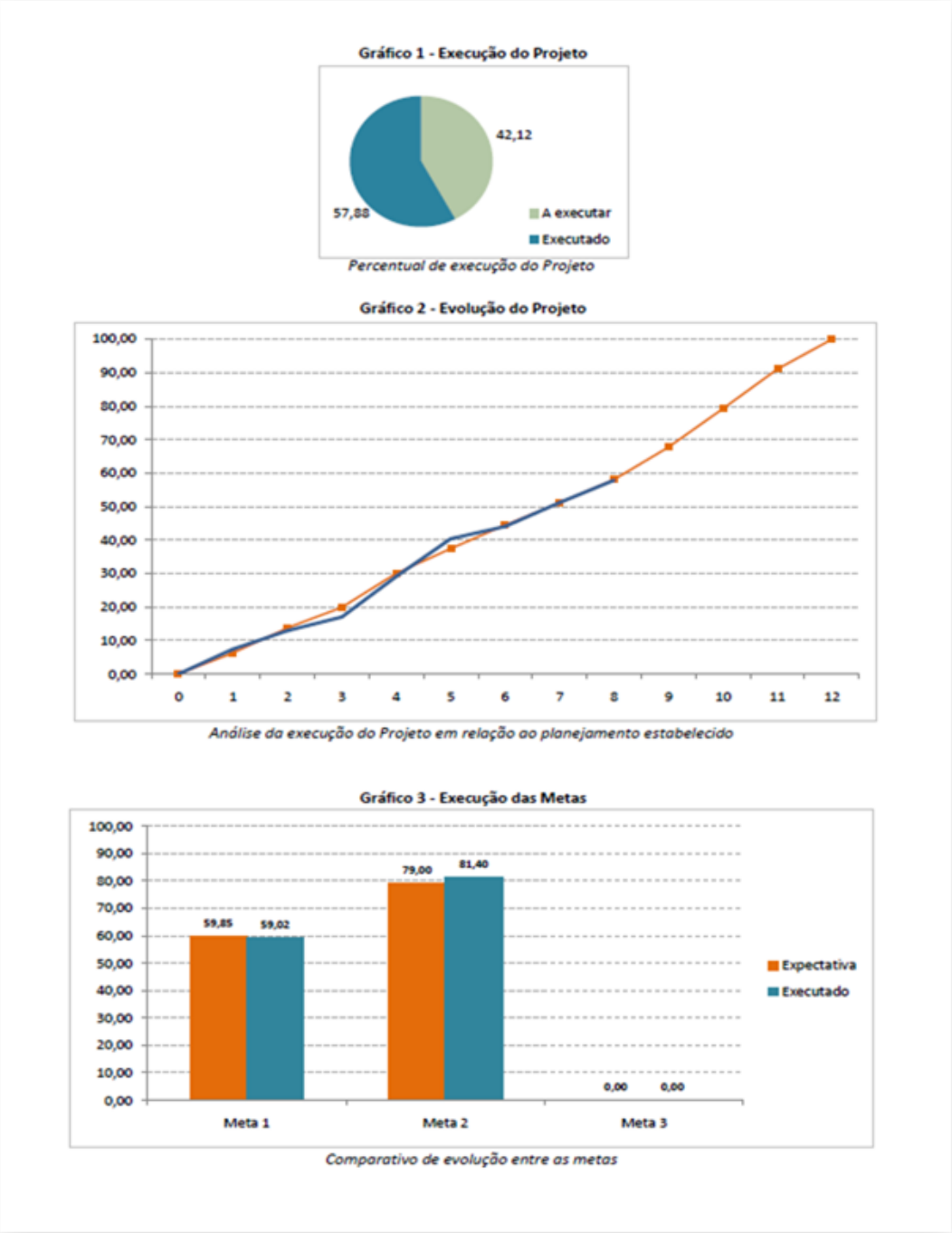


Figura 5.4. Gráficos de acompanhamento de execução de projeto

Esse relatório é enviado mensalmente para o fiscal do projeto na UFSC, para os membros do Conselho do Bridge (professores e gerentes) e técnicos indicados pelo cliente. Tal instrumento é utilizado para subsidiar solicitações de reembolso, reuniões e visa manter o histórico de evolução do projeto ao longo do seu desenvolvimento.

5.7. Considerações Finais

Este artigo apresentou o Laboratório Bridge, seus projetos e sua metodologia de trabalho. O amadurecimento desta cultura ágil e fortemente focada no usuário permitiu a criação de produtos com alto impacto social, excelente qualidade e previsibilidade em termos de custos e cronograma, realizando assim algumas das promessas da engenharia de software.

Referências

Celuppi, I. C. *et al*. (2021) Uma análise sobre o desenvolvimento de tecnologias digitais em saúde para o enfrentamento da COVID-19 no Brasil e no mundo. *Cadernos de Saúde Pública*, v. 37.

DIGITAL.AI. (2021) *15th annual state of agile report*.

Gothelf, J., Seiden, J. (2021) *Lean UX*. "O'Reilly Media, Inc.".

Gomes, I. C. *et al*. (2019) e-SUS AB Atividade Coletiva: aplicativo móvel para registro de atividades coletivas em serviços de Atenção Básica. In: *Anais Estendidos do XIX Simpósio Brasileiro de Computação Aplicada à Saúde*. SBC, 2019. p. 7-12.

Soares, A. P. G. (2022) *Roda ágil: um assessment para medir a agilidade do seu time ágil*. [S. l.], 2017. Disponível em: https://www.anagsoares.com/conheca-a-roda-agil-um-assessment-para-medir-a-agilidade-do-seu-time-agil/. Acesso em: 18 jul. 2022.

Grove, A. S. (2015) *High Output Management.* New York: Vintage.

Hammes, J. *et al*. (2021) *Como executar um Planejamento Estratégico de forma 100% remota.* Disponível em: https://portal.bridge.ufsc.br/2021/11/18/plano-estrategico. Acesso em: 27 jul. 2022.

ISO/IEC (2010) ISO/IEC 25010 - Systems and software engineering. In: *Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models,* Geneva, 2010.

Kholmatova, A. (2017) Design Systems: A practical guide to creating design languages for digital products. *Smashing Magazine.*

Laboratório Bridge (2019a) *Aprenda a construir um design system em 7 passos*. Disponível em: https://laboratoriobridge.medium.com/7-passos-para-construir-um-design-system-5d50cbe44a5. Acesso em: 26 jul. 2022.

Laboratório Bridge (2019b) *Porque decidimos construir um Design System*. Disponível em: https://laboratoriobridge.medium.com/porque-decidimos-construir-um-design-system-a7e3472fb5ea. Acesso em: 26 jul. 2022.

Laboratório Bridge (2019c) *Por que ter um design system acessível*? Disponível em: https://laboratoriobridge.medium.com/porqu%C3%AA-ter-um-design-system-acess%C3%ADvel-e-como-construir-um-5f85ba6bbf29. Acesso em: 26 jul. 2022.

Lacerda, L. C. (2022) *Quatro motivos que nos levaram a ser remote first*. Disponível em: https://portal.bridge.ufsc.br/2022/04/05/remote-first. Acesso em: 28 jul. 2022.

Lacerda, L. C. (2022) *Práticas para aumentar o engajamento no sistema de gestão por OKR em um laboratório de TI*. Disponível em: https://repositorio.ufsc.br/handle/123456789/232607. Acesso em: 27 jul. 2022.

Lacerda, L. C., Soares, M. (2022) *Como mensurar a satisfação dos colaboradores*? Disponível em: https://portal.bridge.ufsc.br/2022/07/22/como-mensurar-a-satisfacao-dos-colaboradores. Acesso em: 29 jul. 2022.

Lacerda, T.C. *et al*. (2020) e-SUS APS strategy: Case of success on Primary Care informatization in Brazil. *Journal of Health Informatics*, v. 12, n. 4.

Lacerda, T. C., von Wangenheim, C. G., Hauck, J. C. R. (2019) *UPCASE-A Method for Self-Assessing the Capability of the Usability Process in Small Organizations*. arXiv preprint arXiv:1902.07244.

Norde, G., Eing, L., Cellupi, I. C., Lacerda, T. C. (2021) *B_thinking: o modelo de processo de UX do Bridge, agora para todos*! Disponível em: https://laboratoriobridge.medium.com/b-thinking-por-que-decidimos-construir-um-modelo-de-processo-de-ux-pr%C3%B3prio-4c10aafe9ca5. Acesso em: 14 maio. 2022.

Silveira, T. (2021) *Cinco tarefas da gestão de pessoas que você provavelmente não conhecia.* Disponível em: https://laboratoriobridge.medium.com/5-tarefas-da-gest%C3%A3o-de-pessoas-que-voc%C3%AA-provavelmente-n%C3%A3o-conhecia-b7dd4c59f9a8. Acesso em: 28 jul. 2022.

Soares, M., Lacerda, T. C. (2021) *b_thinking: Por que decidimos construir um modelo de processo de UX próprio*? Disponível em: https://laboratoriobridge.medium.com/b-thinking-por-que-decidimos-construir-um-modelo-de-processo-de-ux-pr%C3%B3prio-4c10aafe9ca5. Acesso em: 14 maio. 2022.

Wazlawick, R. S. (2019) *Engenharia de software: conceitos e práticas*. Elsevier Editora Ltda. 2a ed.

Capítulo 6

Desenvolvimento de uma aplicação web para o diagnóstico da COVID-19 usando conceitos, técnicas e ferramentas de Inteligência Artificial

Giovanni L. F. da Silva, Emanuel L. C. de S. Filho, Paulo G. S. Magno, Camyla J. P. Santos, João O. B. Diniz, Jonnison L. Ferreira, Caio E. F. Matos e Aristófanes C. Silva

Abstract

The pandemic caused by COVID-19 has drastically affected the health and well-being of the world's population. However, early detection is essential not only for the treatment and cure of patients, but also for public health, ensuring the isolation of patients. So far, several screening methods have been introduced for the diagnosis of COVID-19, including the analysis of medical images. Recent studies prove that Artificial Intelligence can be useful for a fast and accurate diagnosis. Therefore, the mini-course aims to develop a web application for the diagnosis of COVID-19, through chest radiography images, using concepts, techniques and tools of Artificial Intelligence and Digital Image Processing, being able to provide a second opinion to specialists.

Resumo

A pandemia causada pela COVID-19 afetou drasticamente a saúde e o bem-estar da população mundial. No entanto, a detecção precoce é essencial não só para o tratamento e cura dos pacientes, mas também para a saúde pública, garantindo o isolamento dos pacientes. Até o momento, vários métodos de triagem foram introduzidos para o diagnóstico da COVID-19, incluindo a análise de imagens médicas. Estudos recentes comprovam que a Inteligência Artificial pode ser útil para um diagnóstico rápido e preciso. Portanto, o minicurso tem como objetivo desenvolver uma aplicação web para o diagnóstico da COVID-19, por meio de imagens de radiografia do tórax, usando conceitos, técnicas e ferramentas de Inteligência Artificial e Processamento de Imagens Digitais, podendo fornecer uma segunda opinião aos especialistas.

6.1. Introdução

Em dezembro de 2019, a Organização Mundial da Saúde (OMS) relatou problemas de uma pneumonia não identificada surgindo em Wuhan, China [Zhang et al. 2020]. Posteriormente, o vírus foi nomeado como síndrome respiratória aguda grave coronavírus 2 (SARS-CoV-2) e a doença ficou conhecida como COVID-19, tornando-se em pouco tempo uma pandemia de grandeza mundial, totalizando mais de 578 milhões de infectados e 6,4 milhões de óbitos [Chaudhary and Pachori 2021]. Os sintomas mais frequentes da COVID-19 incluem febre, tosse seca, dificuldade na respiração ou falta de ar, dor no peito, perda de movimentos e a perda de paladar ou olfato [Ghassemi et al. 2021, Chaudhary and Pachori 2021].

A detecção precoce da COVID-19 é um fator crucial não só para o tratamento e cura dos pacientes acometidos, mas também para a saúde pública, garantindo o isolamento dos pacientes e, desta forma, controlando a proliferação da doença [Ghaderzadeh and Asadi 2021]. Assim, as modalidades de imagem médica, como a radiografia do tórax (raio-X), podem desempenhar um papel importante no diagnóstico de pacientes com alta dúvida de infecção de acordo com os sintomas [Huang et al. 2021]. Os benefícios da imagem de raio-X, em comparação com outras modalidades de imagem, incluem fácil acesso, baixo custo e baixo risco de radiação, uma vez que são perigosos para a saúde humana [Ghassemi et al. 2021, Ghaderzadeh and Asadi 2021].

Com o avanço da tecnologia nos últimos tempos, principalmente com o uso de aplicações baseadas em Inteligência Artificial [da Silva et al. 2018, Souza et al. 2019, da Silva et al. 2020, Diniz et al. 2021], tornou-se possível o desenvolvimento de sistemas computacionais que auxiliam o especialista na tarefa de análise e interpretação de imagens, conhecidos como sistemas CAD/CADx [Dias et al. 2021, Chaudhary and Pachori 2021]. Uma vez que o computador consegue interpretar de maneira rápida e ininterrupta várias imagens, estes sistemas podem melhorar a eficiência e aumentar a precisão do diagnóstico da COVID-19 e, consequentemente, fornecer uma segunda opinião aos especialistas [Diniz et al. 2021, Huang et al. 2021].

Neste contexto, o objetivo principal deste minicurso é apresentar o desenvolvimento de uma aplicação web para o diagnóstico da COVID-19, por meio de imagens de radiografia do tórax, usando conceitos, técnicas e ferramentas de Inteligência Artificial. Para isso, a Seção 6.2 apresenta os trabalhos relacionados ao tema abordado disponíveis na literatura, Seção 6.1 descreve os materiais e o método proposto para o diagnóstico da COVID-19 por meio de base de imagens de raio-X públicas. Em seguida, a Seção 6.4 detalha o desenvolvimento da aplicação com a utilização da linguagem de programação Python e apresenta os resultados obtidos pelo método proposto, por fim, a Seção 6.5 retrata as considerações finais e propõe os trabalhos futuros.

6.2. Trabalhos relacionados

Os sistemas CAD/CADx baseados em técnicas de Inteligência Artificial são essenciais na facilitação da triagem da COVID-19 por meio das imagens médicas. Neste sentido, vários trabalhos foram recentemente publicados na literatura visando aumentar a precisão e a agilidade no diagnóstico dos pacientes acometidos pela doença. [Al-Bawi et al. 2020] apresenta um método automático usando a variação da rede neural convolucional

conhecida como VGG aplicada a imagens de raio-X, alcançando uma acurácia de 98,52%. Já [Ghassemi et al. 2021] propõe um método automático para o diagnóstico da COVID-19 usando a rede neural CycleGAN em imagens de tomografia computadorizada (TC), obtendo uma acurácia de 99,60%.

O trabalho proposto por [Dias et al. 2021] demonstra a aplicabilidade das *deep features* e o algoritmo XGBoost no diagnóstico automático da COVID-19 em imagens de raio-X, obtendo uma acurácia de 98,17%. [Chaudhary and Pachori 2021] propõe um método automático utilizando a decomposição baseada em expansão da série de Fourier-Bessel e a rede neural convolucional, alcançando uma acurácia de 100%. No geral, os trabalhos publicados apresentam resultados satisfatórios, tanto em imagens de raio-X quanto em TC, usando bases de imagens públicas na Internet. Uma vez que o objetivo do minicurso é apresentar o desenvolvimento de uma aplicação web para o diagnóstico da COVID-19, o método proposto neste trabalho utiliza técnicas convencionais do Processamento de Imagens Digitais e Inteligência Artificial.

6.3. Materiais e método

O método proposto neste minicurso para o desenvolvimento de uma aplicação web capaz de diagnosticar COVID-19 por meio de imagens de raio-X, ilustrado na Figura 6.1, está dividido em quatro etapas. Em resumo, a primeira etapa consiste na aquisição dos materiais utilizados, como a base de imagens públicas dos exames de radiografia do tórax. Na segunda etapa, a extração de característica é realizada usando descritores de textura, a terceira etapa apresenta o treinamento do modelo preditivo e, por fim, na última etapa, os resultados obtidos pelo método são avaliados.

Figura 6.1. Fluxograma do método proposto para o diagnóstico da COVID-19 em imagens de raio-X.

6.3.1. Materiais

A base de dados utilizada neste minicurso consiste em uma pequena amostra das bases de imagens *COVID-19 Radiography Database* e a *COVID-QU-Ex Dataset*, ambas disponíveis na Internet. A amostra utilizada consiste em 100 imagens de raio-X de pacientes acometidos pela COVID-19 e 100 imagens de raio-X de pacientes saudáveis, totalizando 200 imagens ao todo. Todas as imagens foram padronizadas para a resolução espacial 256 x 256. A Figura 6.2 apresenta exemplos de imagens utilizadas pelo método proposto.

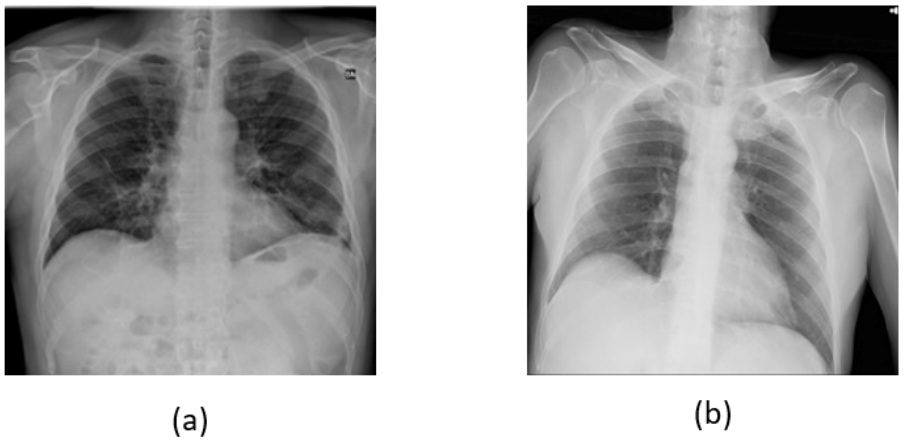


Figura 6.2. Exemplos de imagens de raio-X utilizadas no método proposto.

A radiografia do tórax costuma ser o primeiro exame de imagem usado para ajudar no diagnóstico de doenças e no tratamento de emergência, pois é rápido, fácil e de baixo custo. A partir do exame, pode ser revelado muitas coisas dentro do corpo, incluindo a condição dos pulmões, insuficiência cardíaca, vasos pulmonares, enfisema, COVID-19, o tamanho e o contorno do coração, fraturas, alterações pós-operatórias e posicionamento de dispositivos médicos [Huang et al. 2021].

6.3.2. Extração de características

Uma das tarefas mais complexas presentes na área de Processamento de Imagens Digitais está na definição de um conjunto de características capazes de descrever de maneira efetiva a imagem de entrada [Pedrini and Schwartz 2008]. Segundo [Haralick 1979], a textura pode ser definida como a característica de uma região relacionada a coeficientes de uniformidade, densidade, aspereza, regularidade, intensidade, entre outras características da imagem.

A análise de textura é importante em imagens médicas, pois concedem informações importantes ao realizar a discriminação [Pedrini and Schwartz 2008]. Uma forma clássica de quantificação da textura numa imagem em níveis de cinza é a abordagem estatística, a qual possibilita a descrição da textura através das regras estatísticas que regem tanto a distribuição quanto à relação entre os diferentes níveis de cinza presentes em uma imagem. Portanto, neste minicurso, foram extraídas 14 características de textura proposta por [Haralick 1979].

6.3.3. Modelo preditivo

A etapa de treinamento do modelo preditivo consiste na utilização de algoritmos de Aprendizado de Máquina para a tarefa de reconhecimento de padrões baseado em um conjunto de dados históricos [Janiesch et al. 2021]. Após a etapa de extração de características, os dados que antes estavam no formato não estruturado (imagens) passam a estar estruturado (tabular), portanto, estão aptos para a utilização de técnicas de Aprendizado de Máquina tradicionais, tais como a Máquina de Vetores de Suporte, a Árvore de Decisão, a Regressão Logística, K-Vizinhos Mais Próximos, entre outras.

Os tipos de aprendizado por partes dos algoritmos de Aprendizado de Máquina são compostos por dois grupos principais, supervisionado e não-supervisionado. O aprendizado supervisionado consiste em um processo prévio de treinamento de um classificador para entender os padrões desejados. Posteriormente, o classificador é capaz de identificar o rótulo de qualquer objeto desconhecido na mesma população de treinamento. No aprendizado não-supervisionado, não há informação prévia sobre os rótulos aos quais os padrões de amostras pertencem [Mitchell and Mitchell 1997].

Neste minicurso, o algoritmo XGBoost proposto por [Chen and Guestrin 2016] foi utilizado para o treinamento do modelo preditivo. A escolha pelo XGBoost foi devido ao seu desempenho, em comparação com outros algoritmos disponíveis. Além disso, o XGBoost tem como vantagens a velocidade de predição, escalabilidade do modelo preditivo, baixo consumo de memória e recursos de hardware, sendo um dos algoritmos mais utilizados na literatura atual, alcançando o estado da arte em diversas aplicações [Chen and Guestrin 2016].

O algoritmo XGBoost é baseado no algoritmo de árvores de decisão com aumento de gradiente, do inglês *gradient boosting decision tree*. O algoritmo *boosting* é uma técnica de *ensemble* onde novos modelos são adicionados sequencialmente para corrigir os erros obtidos pelos modelos anteriores, até que não haja melhora no resultado. O *boosting* é constituído essencialmente de três etapas:

1. A partir de um modelo inicial F_0, uma entrada de dados X e a classe y, o erro residual desse modelo é dado por:

$$E_0 = y - F_0(X) \tag{1}$$

   onde $F_0(X)$ é a saída do modelo inicial.

2. Um novo modelo h_1 é treinado então apresentando como rótulo o erro residual do modelo anterior (no caso E_0).

3. Assim, F_0 e h_1 são combinados para gerar F_1:

$$F_1(X) = F_0(X) + h_1(X) \tag{2}$$

   em que o novo erro será dado por:

$$E_1 = y - F_1(X) \tag{3}$$

Esse processo pode ser repetido para a inserção de vários modelos.

Já a técnica de aumento de gradiente, do inglês *gradient boosting*, utiliza um algoritmo de gradiente descendente para minimizar a função de perda ao adicionar novos modelos [Chen and Guestrin 2016]. No caso do algoritmo XGBoost, os modelos utilizados pelo algoritmo são árvores de decisão, porém, a implementação do algoritmo foi feita para ter a maior eficiência possível no uso de recursos de memória e processamento.

A função objetivo do XGBoost é dada por:

$$\ell^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{4}$$

onde $\ell^{(t)}$ é a função de objetivo na iteração t, $\hat{y}_i^{(t-1)}$ é a predição da i-ésima instância na iteração $t-1$, y_i é a classe, $\Omega(f_t)$ é o termo de regularização do modelo, $f_t(x_i)$ é a saída da árvore na iteração t, e g_i e h_i são os gradientes de primeira e segunda ordem da função de perda [Chen and Guestrin 2016].

6.3.4. Validação do modelo preditivo

Em um sistema CAD/CADx, os resultados obtidos pelo modelo preditivo em relação ao diagnóstico podem ser divididos em quatro grupos:

- O teste é positivo e o paciente tem a doença – Verdadeiro Positivo (VP);
- O teste é positivo e o paciente não tem a doença – Falso Positivo (FP);
- O teste é negativo e o paciente tem a doença – Falso Negativo (FN);
- O teste é negativo e o paciente não tem a doença – Verdadeiro Negativo (VN).

Para avaliar o desempenho do modelo preditivo, é comum utilizar o cálculo de algumas estatísticas como a acurácia, *recall*, especificidade e a precisão [Hossin and Sulaiman 2015], obtidos pela matriz de confusão ilustrada na Figura 6.3.

A acurácia representa a probabilidade de pacientes classificados corretamente sobre o total de pacientes existentes, como descrita na Equação 5.

$$ACU = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

O *recall* representa a probabilidade de verdadeiros positivos sobre o total de pacientes acometidos pela COVID-19 existentes, como descrita na Equação 6.

$$REC = \frac{VP}{VP + FN} \tag{6}$$

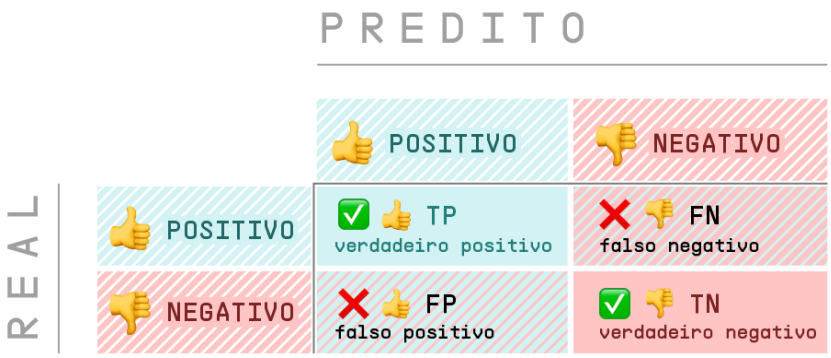


Figura 6.3. Matriz de confusão.

A especificidade representa a probabilidade de verdadeiros negativos sobre o total de pacientes saudáveis existentes, como descrita na Equação 7.

$$ESP = \frac{VN}{VN + FP} \tag{7}$$

A precisão representa a probabilidade de verdadeiros positivos sobre o total de pacientes classificados como acometidos pela COVID-19, como descrita na Equação 8.

$$PREC = \frac{VP}{VP + FP} \tag{8}$$

A acurácia, *recall*, especificidade e precisão foram empregadas neste minicurso para avaliar o desempenho da método proposto para o diagnóstico da COVID-19 por meio de imagens de raio-X, considerando pacientes acometidos pela COVID-19 corretamente classificados como verdadeiros positivos.

6.4. Resultados e discussão

Esta seção detalha o desenvolvimento da aplicação web para apresentação e interpretação dos resultados obtidos usando o método proposto descrito na Seção 6.1. Para isso, toda a aplicação foi desenvolvida na linguagem de programação Python [Millman and Aivazis 2011], usando as bibliotecas OpenCV [Bradski 2000], Mahotas [Coelho 2012], XGBoost [Chen and Guestrin 2016], Imblearn [Mishra 2017], LIME [Ribeiro et al. 2016] e Streamlit [Shukla et al. 2021].

Todo o projeto está disponível no repositório público [1], dividido em três pastas principais: etc, images e src. A pasta etc consiste nos arquivos gerados durante o minicurso, tais como o arquivo de características no formato csv e o modelo preditivo no formato dat. A pasta images contém as 200 imagens de raio-X organizadas em duas subpastas, normal e covid. Por fim, a pasta src contém o *scripts* necessários para o desenvolvimento da aplicação web para o diagnóstico da COVID-19.

[1] https://github.com/giiovannilucca/covid_diagnostic_app.git

A Figura 6.4 apresenta o passo-a-passo para reprodução do ambiente utilizado para o desenvolvimento do minicurso. O arquivo *environment.yml* contém todas as bibliotecas necessárias. Para o melhor entendimento da aplicação desenvolvida de ponta-a-ponta, todos os *scripts* são explicados linha por linha.

```
#Passo 1 - Clonar o repositório no Github

>> git clone https://github.com/giiovannilucca/covid_diagnostic_app.git

#Passo 2 - Instalar as bibliotecas usando o Anaconda

>> conda env create -f environment.yml

#Passo 3 - Ativar o ambiente covid no Anaconda

>> conda activate covid

#Passo 4 - Extrair as características de textura

>> python src/feat_extraction.py

#Passo 5 - Treinar o modelo preditivo

>> python src/model_training.py

#Passo 6 - Executar a aplicação web

>> streamlit run src/web_application.py
```

Figura 6.4. Passo-a-passo para reprodução do ambiente utilizado para o desenvolvimento do minicurso.

6.4.1. Script *feat_extraction.py*

A Figura 6.5 detalha a importação das bibliotecas necessárias para a extração das características das imagens de raio-X (linhas 1 à 5). Entre as bibliotecas mencionadas, destacam-se a Mahotas e a OpenCV. A biblioteca Mahotas disponibiliza os descritores de textura utilizados para descrever os padrões das imagens de raio-X e a OpenCV proporciona diversas operações em imagens digitais, desde a leitura até a aplicação de técnicas de melhoramento de imagens avançadas. A linha 7 inicializa uma lista com os nomes dos rótulos das imagens e a linha 9 inicializa uma lista vazia que armazenará as características extraídas de todas as imagens.

A Figura 6.6 descreve o processo de leitura das imagens e extração das características de textura. O primeiro laço (linha 11) itera sobre a lista dos rótulos, portanto, na primeira iteração lê todas as imagens de raio-X rotuladas como "Normal"e, em seguida, lê todas as imagens de raio-X rotuladas como "COVID". Ao todo, o minicurso foi desenvolvido com 200 imagens, sendo 100 imagens de raio-X de pacientes saudáveis e 100 imagens de raio-X de pacientes acometidos pela COVID-19.

Ainda na Figura 6.6, a linha 19 detalha a leitura da imagem de raio-X usando a biblioteca OpenCV. Para isso, foram passados dois parâmetros para o método *cv.imread*, o primeiro consiste no caminho relativo da imagem de entrada e o segundo consiste no formato de leitura, sistema de cores original ou escala de cinza. O valor zero indica que

```
1 import mahotas
2 import pandas
3 import numpy
4 import glob
5 import cv2
6
7 labels = ['normal', 'covid']
8
9 features_list = list()
```

Figura 6.5. Importação das bibliotecas necessárias para a extração das características.

a imagem foi lida em escala de cinza. A linha 23 ilustra a extração das características de textura usando a biblioteca Mahotas. Por fim, a linha 26 adiciona o rótulo da imagem de entrada, ou seja, o diagnóstico comprovado clinicamente com as demais características de textura e a linha 28 insere as características extraídas a lista que contém todas as demais.

A Figura 6.7 apresenta o processo de exportação da lista de características no formato csv. As linhas 29 e 31 viabilizam a criação do cabeçalho do arquivo csv, ou seja, detalha o nome dos descritores de características extraídas juntamente com a coluna para o rótulo. Posteriormente, as linhas 33 e 35 descrevem a exportação das características para o arquivo chamado *features* usando a biblioteca de análise de dados Pandas [Nelli 2018].

Ao final da execução desde *script*, espera-se que um arquivo chamado *features* no formato csv seja criado na pasta etc do projeto. O arquivo conterá as características para todas as 200 imagens de raio-X utilizadas no minicurso, além do cabeçalho com o nome de todos os descritores de textura. Ressalta-se que o quantitativo de imagens pode

```
11 for label in labels:
12
13     path = './images/' + label
14
15     images_list = glob.glob(path + '/*.png')
16
17     for image_path in images_list:
18
19         img = cv2.imread(image_path, 0)
20
21         label = 1 if 'covid' in image_path else 0
22
23         features_img = mahotas.features.haralick(img, compute_14th_feature = True,
24                                                  return_mean = True)
25
26         features_img = numpy.append(features_img, label)
27
28         features_list.append(features_img)
```

Figura 6.6. Extração das características de textura.

```
features_names = mahotas.features.texture.haralick_labels

features_names = numpy.append(features_names, 'Label')

df = pandas.DataFrame(data = features_list, columns = features_names)

df.to_csv('./etc/features.csv', index = False, sep = ';')
```

Figura 6.7. Exportação da lista de características no formato csv.

ser aumentado facilmente com a inserção de novas imagens nas pastas images/normal e images/covid.

6.4.2. Script *model_training.py*

A Figura 6.8 ilustra as bibliotecas necessárias para o treinamento do modelo preditivo usando o algoritmo XGBoost e a técnica de aumento de dados SMOTE [Chawla et al. 2002]. Para isso, foram utilizadas as bibliotecas Sklearn, Imblearn, Xgboost, Pandas e Pickle. A linha 1 indica a importação do método *train_test_split* necessário para separação do conjunto de treinamento e teste. A linha 2 ilustra a importação do método *confusion_matrix* fundamental para o cálculo das métricas de desempenho.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import SMOTE
import xgboost
import pandas
import pickle
```

Figura 6.8. Importação das bibliotecas necessárias para o treinamento do modelo preditivo.

Além disso, na Figura 6.8, a linha 3 indica a importação do método SMOTE usado para aumentar de forma sintética a quantidade de dados no conjunto de treinamento, visando resultados melhores no conjunto de teste. A linha 4 importa a biblioteca XGBoost para a construção do modelo preditivo, a linha 5 indica a importação da biblioteca Pandas para a leitura do arquivo no formato csv e, por fim, a linha 6 apresenta a importação da biblioteca Pickle para o armazenamento e leitura do modelo preditivo em disco local.

A Figura 6.9 apresenta uma função para o cálculo das métricas de desempenho do modelo preditivo descrita na Seção 6.3.4, tais como a acurácia, *recall*, especificidade e a precisão. Todas estas métricas podem ser obtidas com a utilização da matriz de confusão

(linha 9). Neste minicurso, a principal métrica de desempenho é o *recall*, pois representa a taxa de acertos do modelo preditivo para as imagens de raio-X de COVID-19 classificadas corretamente.

```
 8 def get_metrics(y_true, y_pred):
 9     vn, fp, fn, vp = confusion_matrix(y_true, y_pred).ravel()
10     accuracy = (vp + vn) / (vp + fp + fn + vn)
11     recall = vp / (vp + fn)
12     specificity = vn / (vn + fp)
13     precision = vp / (vp + fp)
14
15     return {
16         'accuracy': accuracy,
17         'specificity': specificity,
18         'recall': recall,
19         'precision': precision,
20     }
```

Figura 6.9. Função para o cálculo das métricas de desempenho do modelo preditivo.

A Figura 6.10 apresenta a etapa de separação dos conjuntos de treinamento e teste usando a técnica *hold-out* [Hossin and Sulaiman 2015]. A linha 22 descreve a leitura do arquivo *features* usando a biblioteca Pandas. As linhas 24 e 25 apresentam a separação das características e dos rótulos. Em seguida, a linha 27 separa de forma mutuamente exclusiva os dados de treinamento e teste, usando a proporção de 20% para o conjunto de teste. Por fim, as linhas 29 e 30 ilustram a aplicação da técnica SMOTE no conjunto de treinamento para aumento dos dados. O objetivo é criar novas instâncias afim de garantir uma melhor generalização no modelo preditivo.

```
22 df = pandas.read_csv('./etc/features.csv', delimiter = ';')
23
24 X = df.drop('Label', axis = 1)
25 y = df['Label']
26
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
28
29 balanced = SMOTE(random_state=42)
30 X_train, y_train = balanced.fit_resample(X_train, y_train)
```

Figura 6.10. Separação dos conjuntos de treinamento e teste.

A Figura 6.11 detalha a etapa de construção do modelo preditivo usando o algoritmo XGBoost. Para isso, a linha 32 apresenta os hiperparâmetros utilizados no XGBoost. A linha 36 apresenta o treinamento do modelo através do método *fit*, em seguida, a linha 38 armazena as predições após o treinamento do modelo preditivo para o conjunto de teste. A linhas 40 e 42 apresentam os resultados obtidos pelo método

proposto, alcançando uma acurácia de 85%, *recall* de 89,47%, especificidade de 80,95% e precisão de 80,95%.

```
32 model = xgboost.XGBClassifier(objective = "binary:logistic", random_state = 42, max_depth = 9,
                                 colsample_bytree = 0.4033, min_child_weight = 6, gamma = 0.429,
33                                 eta = 0.5995, n_estimators = 1000, use_label_encoder=False,
                                     eval_metric='merror')
34
35 model.fit(X_train, y_train)
36
37 y_pred = model.predict(X_test)
38
39 metrics = get_metrics(y_test, y_pred)
40
41 print(metrics)
42
43 pickle.dump(model, open('./etc/xgb_model.dat', 'wb'))
```

Figura 6.11. Treinamento do modelo preditivo usando o algoritmo XGBoost.

6.4.3. Script *web_application.py*

A Figura 6.12 apresenta as bibliotecas necessárias para a aplicação web usando a biblioteca Streamlit e LIME. De bibliotecas novas, apenas as linhas 2, 3 e 4. O Streamlit é uma biblioteca de código-aberto que facilita a criação e o compartilhamento de aplicativos web personalizados e bonitos para as áreas de Inteligência Artificial e Ciência de Dados. A linha 3 indica a importação da biblioteca LIME para interpretação das predições do modelo preditivo.

```
1 from sklearn.model_selection import train_test_split
2 import streamlit.components.v1 as components
3 import lime.lime_tabular
4 import streamlit
5 import mahotas
6 import pickle
7 import pandas
8 import numpy
9 import cv2
```

Figura 6.12. Importação das bibliotecas necessárias para a aplicação web.

A Figura 6.13 ilustra duas funções auxiliares no desenvolvimento da aplicação web. A primeira consiste na função para o carregamento do modelo preditivo que está localizado na pasta etc (linhas 12 e 13). A criação do modelo preditivo é resultado da execução do *script* anterior, descrito na Seção 6.4.2. A segunda função consiste na conversão da imagem de raio-X de bytes para a biblioteca OpenCV (linhas 15 a 20).

```
12 def get_model():
13     return pickle.load(open('./etc/xgb_model.dat', 'rb'))
14
15 def convert_byteio_image(string):
16     array = numpy.frombuffer(string, numpy.uint8)
17     image = cv2.imdecode(array, flags=1)
18     image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
19
20     return image
```

Figura 6.13. Funções auxiliares na aplicação web.

A Figura 6.14 apresenta a configuração da barra lateral da aplicação web. A linha 22 expõe o título da aplicação e a linha 23 inicializa a barra lateral. Em seguida, a linha 26 apresenta a funcionalidade de *upload* da imagem de entrada, a imagem precisa estar no formato png e o *upload* não aceita mais de uma imagem por vez. Por fim, na linha 28 o modelo preditivo é carregado em memória para a predição dos novos dados, ou seja, as novas imagens de raio-X.

```
22 streamlit.markdown("<h1 style='text-align: center; color: black;'>Web application for the diagnosis of
                       COVID-19 using X-ray images</h1>", unsafe_allow_html=True)
23
24 streamlit.sidebar.title('Settings')
25
26 uploaded_image = streamlit.sidebar.file_uploader("Choose an image", type = 'png',
                                                    accept_multiple_files = False)
27
28 model = get_model()
```

Figura 6.14. Configuração da barra lateral da aplicação web.

A Figura 6.15 demonstra a conversão da imagem em bytes para OpenCV e, em seguida, a extração de características de textura. A linha 30 verifica se alguma imagem foi carregada durante o *upload*, caso tenha acontecido, a imagem passa pelo processo de conversão nas linhas 32 e 34. Posteriormente, a linha 36 verifica a resolução espacial da imagem de entrada, caso ela seja diferente de 256 x 256, a imagem é redimensionada, conforme ilustrado na linha 37. Finalmente, a linha 39 apresenta o processo de extração de características.

A Figura 6.16 apresenta a predição do modelo preditivo (linhas 41 e 43) usando as características de textura extraídas na linha 39. Além disso, as linhas 47 e 49 exibem a imagem de raio-X centralizada, possibilitando a visualização do exame para um especialista. Por último, as linhas 51 e 53 formatam a predição do modelo preditivo em probabilidades, ou seja, além de informar a predição final, o modelo preditivo é capaz de indicar o quão convicto está em sua predição.

```
30 if uploaded_image is not None:
31
32     bytes_data = uploaded_image.getvalue()
33
34     image = convert_byteio_image(bytes_data)
35
36     if (image.shape != (256, 256)):
37         image = cv2.resize(image, (256, 256))
38
39     features = mahotas.features.haralick(image, compute_14th_feature = True,
40                                          return_mean = True).reshape(1, 14)
```

Figura 6.15. Leitura da imagem de entrada e extração de características.

O último trecho do *script* da aplicação web, ilustrado na Figura 6.17, demonstra a utilização da biblioteca LIME para o entendimento dos fatores que levaram ao modelo preditivo tomar aquela decisão em sua predição, ou seja, quais características contribuíram de forma positiva e negativa para a predição do modelo preditivo (linhas 68 a 73). Uma vez que o algoritmo LIME precisa do conjunto de treinamento para a sua interpretação local, foi necessário a leitura do conjunto de treinamento novamente (linhas 61 a 66).

A Figura 6.18 apresenta a interface gráfica da aplicação web desenvolvida com a parte da visualização da imagem de raio-X. Após o carregamento da imagem de entrada, todo o processo de extração de característica e predição do modelo preditivo é iniciado em *background*, assim, que o processo estiver concluído, um texto abaixo da imagem de entrada indica a predição juntamente com a probabilidade assegurada pelo modelo preditivo. Por fim, a Figura 6.19 demonstra a interpretação da predição obtida usando a biblioteca LIME, no qual, as características ressaltadas com a cor laranja indica uma

```
41 pred = model.predict(features)
42
43 probs = model.predict_proba(features)
44
45 streamlit.markdown("<h3 style='text-align: center; color: black;'>Image</h3>",
                        unsafe_allow_html=True)
46
47 col1, col2, col3 = streamlit.columns([0.2, 5, 0.2])
48
49 col2.image(image, use_column_width=True)
50
51 pred_output = "Patient affected by COVID-19 with {:.2%} certainty".format(probs[0]
                 [1]) if pred[0] == 1 else "Patient within the normal range with
                     {:.2%} certainty".format(probs[0][0])
52
53 streamlit.markdown("<h4 style='text-align: center; color: black;'>" + pred_output + "
                        </h4>", unsafe_allow_html=True)
```

Figura 6.16. Predição e visualização da imagem de entrada.

```
55 if streamlit.sidebar.button("Explain prediction"):
56
57     streamlit.markdown("<h3 style='text-align: center; color:
           black;'>Interpretation</h3>", unsafe_allow_html=True)
58
59     with streamlit.spinner('Calculating...'):
60
61         df = pandas.read_csv('./etc/features.csv', delimiter = ';')
62
63         X = df.drop('Label', axis = 1)
64         y = df['Label']
65
66         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                             random_state = 42)
67
68         explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values,
                        feature_names = X.columns,
69                      class_names = ['Normal', 'COVID-19'],
70                      feature_selection = 'lasso_path',
71                      discretize_continuous = True)
72
73         exp = explainer.explain_instance(features.reshape(14,), model.predict_proba,
                                             num_features = 14)
74
75         components.html(exp.as_html(predict_proba = False), height = 800)
76
```

Figura 6.17. Interpretação das predições do modelo preditivo.

contribuição positiva para o rótulo COVID-19. Por outro lado, a cor azul indica uma contribuição positiva para o rótulo normal.

6.5. Considerações finais

Este capítulo de livro apresentou o desenvolvimento de uma aplicação web para o diagnóstico da COVID-19 por meio de imagens de raio-X usando conceitos, técnicas e ferramentas de Inteligência Artificial. Com esse intuito, foram utilizadas imagens de raio-X públicas, disponíveis na Internet. O método proposto usando os descritores de textura de Haralick e o algoritmo XGBoost apresentou resultados satisfatórios, obtendo uma acurácia de 85%, *recall* de 89,47%, especificidade de 80,95% e precisão de 80,95%.

Uma vez que o objetivo principal do minicurso foi apresentar o desenvolvimento de ponta-a-ponta de uma aplicação web baseada em Inteligência Artificial, não foi necessário um grande quantitativo de imagens para validação do método proposto e nem um pipeline de técnicas inovadoras. No entanto, como trabalhos futuros, destacam-se: 1) aplicação de técnicas de melhoramento de imagens, 2) utilização de outros descritores de textura, juntamente com uma seleção de características, 3) execução de outros algoritmos de Aprendizado de Máquina e, por fim, 4) *deploy* da aplicação em um servidor web.

Referências

[Al-Bawi et al. 2020] Al-Bawi, A., Al-Kaabi, K., Jeryo, M., and Al-Fatlawi, A. (2020). Ccblock: an effective use of deep learning for automatic diagnosis of covid-19 using

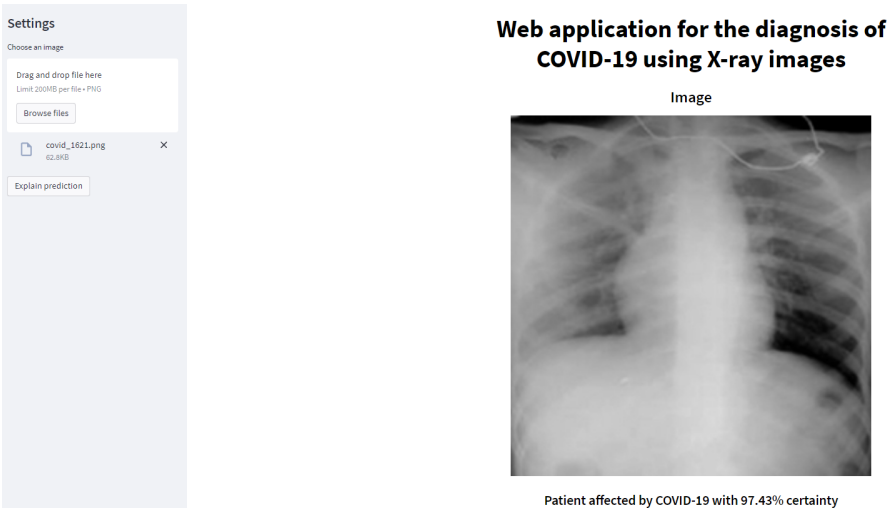


Figura 6.18. Interface gráfica da aplicação web com a visualização da imagem de raio-X.

Figura 6.19. Interface gráfica da aplicação web com a interpretação da predição obtida.

x-ray images. *Research on Biomedical Engineering*, pages 1–10.

[Bradski 2000] Bradski, G. (2000). The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123.

[Chaudhary and Pachori 2021] Chaudhary, P. K. and Pachori, R. B. (2021). Fbsed based automatic diagnosis of covid-19 using x-ray and ct images. *Computers in Biology and Medicine*, 134:104454.

[Chawla et al. 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[Chen and Guestrin 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

[Coelho 2012] Coelho, L. P. (2012). Mahotas: Open source software for scriptable computer vision. *arXiv preprint arXiv:1211.4907*.

[da Silva et al. 2020] da Silva, G. L., Diniz, P. S., Ferreira, J. L., Franca, J. V., Silva, A. C., de Paiva, A. C., and de Cavalcanti, E. A. (2020). Superpixel-based

deep convolutional neural networks and active contour model for automatic prostate segmentation on 3d mri scans. *Medical & Biological Engineering & Computing*, 58(9):1947–1964.

[da Silva et al. 2018] da Silva, G. L. F., Valente, T. L. A., Silva, A. C., de Paiva, A. C., and Gattass, M. (2018). Convolutional neural network-based pso for lung nodule false positive reduction on ct images. *Computer methods and programs in biomedicine*, 162:109–118.

[Dias et al. 2021] Dias, D. A. J., da Cruz, L. B., Diniz, J. O. B., da Silva, G. L. F., Junior, G. B., Silva, A. C., de Paiva, A. C., Nunes, R. A., and Gattass, M. (2021). Automatic method for classifying covid-19 patients based on chest x-ray images, using deep features and pso-optimized xgboost. *Expert Systems with Applications*, 183:115452.

[Diniz et al. 2021] Diniz, J. O., Quintanilha, D. B., Santos Neto, A. C., da Silva, G. L., Ferreira, J. L., Netto, S., Araújo, J. D., Da Cruz, L. B., Silva, T. F., da S Martins, C. M., et al. (2021). Segmentation and quantification of covid-19 infections in ct using pulmonary vessels extraction and deep learning. *Multimedia Tools and Applications*, 80(19):29367–29399.

[Ghaderzadeh and Asadi 2021] Ghaderzadeh, M. and Asadi, F. (2021). Deep learning in the detection and diagnosis of covid-19 using radiology modalities: a systematic review. *Journal of healthcare engineering*, 2021.

[Ghassemi et al. 2021] Ghassemi, N., Shoeibi, A., Khodatars, M., Heras, J., Rahimi, A., Zare, A., Pachori, R. B., and Gorriz, J. M. (2021). Automatic diagnosis of covid-19 from ct images using cyclegan and transfer learning. *arXiv preprint arXiv:2104.11949*.

[Haralick 1979] Haralick, R. M. (1979). Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804.

[Hossin and Sulaiman 2015] Hossin, M. and Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1.

[Huang et al. 2021] Huang, S., Yang, J., Fong, S., and Zhao, Q. (2021). Artificial intelligence in the diagnosis of covid-19: Challenges and perspectives. *International Journal of Biological Sciences*, 17(6):1581.

[Janiesch et al. 2021] Janiesch, C., Zschech, P., and Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3):685–695.

[Millman and Aivazis 2011] Millman, K. J. and Aivazis, M. (2011). Python for scientists and engineers. *Computing in Science & Engineering*, 13(2):9–12.

[Mishra 2017] Mishra, S. (2017). Handling imbalanced data: Smote vs. random undersampling. *Int. Res. J. Eng. Technol*, 4(8):317–320.

[Mitchell and Mitchell 1997] Mitchell, T. M. and Mitchell, T. M. (1997). *Machine learning*, volume 1. McGraw-hill New York.

[Nelli 2018] Nelli, F. (2018). The pandas library—an introduction. In *Python Data Analytics*, pages 87–139. Springer.

[Pedrini and Schwartz 2008] Pedrini, H. and Schwartz, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações*. Cengage Learning.

[Ribeiro et al. 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

[Shukla et al. 2021] Shukla, S., Maheshwari, A., and Johri, P. (2021). Comparative analysis of ml algorithms & stream lit web application. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 175–180. IEEE.

[Souza et al. 2019] Souza, J. C., Diniz, J. O. B., Ferreira, J. L., da Silva, G. L. F., Silva, A. C., and de Paiva, A. C. (2019). An automatic method for lung segmentation and reconstruction in chest x-ray using deep neural networks. *Computer methods and programs in biomedicine*, 177:285–296.

[Zhang et al. 2020] Zhang, J., Xie, Y., Li, Y., Shen, C., and Xia, Y. (2020). Covid-19 screening on chest x-ray images using deep learning based anomaly detection. *arXiv preprint arXiv:2003.12338*, 27.