# X SINFO

Simpósio de Sistemas de Informação

ANAIS ELETRÔNICOS SINFO 2019
MINICURSOS

ORGANIZAÇÃO GERAL
Glauber Dias Gonçalves

COMITÊ CIENTÍFICO
Deborah Maria Vieira Magalhães
Flávio Henrique Duarte de Araújo
Romuere Rodrigues Veloso e Silva

REALIZAÇÃO
Universidade Federal do Piauí


Picos. PI
2019

# MINICURSOS

**1. Análise de Dados com Python: Tratando dados com Pandas, NumPy e IPython**
Rafael Luz Araujo, Romuere Veloso e Silva, Daniel de Sousa Luz, José Anatiel G. S. Landim, Thiago Jose B. lima, Nonato R. de Sales Carvalho, Pablo de Abreu Vieira, Kelly M. da Silva Oliveira

**2. Introdução a machine learning em python**
Thiago J. B. Lima, Flávio H. D. Araújo, Pablo A. Vieira, Nonato R. de S. Carvalho, Rafael L. Araújo, Laécio A. Rodrigues

**3. Deep Learning From ScratchWithout Framework**
Pablo de Abreu Vieira, Romuere Rodrigues Veloso Silva, Thiago José Barbosa Lima, Nonato Rodrigues de Sales Carvalho, Rafael Luz Araújo

**4- Técnicas de Extração de Atributos Aplicadas ao Processamento de Imagens**
Patrick Ryan Sales dos Santos, Vitória de Carvalho Brito e Antonio Oseas de Carvalho Filho

**5. Alocação de Recursos no contexto de Network Slicing as a Service baseado em Algoritmos Genéticos: Teoria e Prática**
Rayner Gomes Sousa

**6. Processamento de Dados do Twitter para Monitoramento e Recomendação de Rotas em Cenários de Desastres Naturais**
Mateus P. Garcia, Orrana L. V. de Sousa, Ana C. de A. Alves, Deborah M.V. Magalhães

**7. Construindo Modelo de Site com Fundamentos em Bootstrap**
Gabriell Oliveira Paes Landim, Luis Guilherme Sousa e Silva, Antônio Oseas de Carvalho Filho e Rafael Luz Araújo

**8. Mapeamento sistemático feito com o auxílio de ferramenta computacional**
Ana Beatriz Barbosa Maia da Silva, Thales José Sousa Bezerra

**9. Construindo um mini instagram em 4 horas com NodeJS**
Marcos Paulo S. Barreto, Lucas B. M. Souza, Thaliane R. Gomes

**10. Automatizando Tarefas Web com o Framework Selenium e Python**
Lucas V. S. dos Santos, Rubenilson S. Lima, Samuel P. B. D. Lélis, Cícero N.A. do Ó, Carvalho Filho A. O. e Rafael Luz Araújo

# Chapter 1

# Análise de Dados com Python: Tratando dados com Pandas, NumPy e IPython

Rafael Luz Araujo, Romuere Veloso e Silva, Daniel de Sousa Luz, José Anatiel G. S. Landim, Thiago Jose B. lima, Nonato R. de Sales Carvalho, Pablo de Abreu Vieira, Kelly M. da Silva Oliveira

***Abstract***

*Data analysis emerged from the knowledge potential gained through data, making it essential for the maintenance and success of any company. The daily life of the professional data scientist involves complex problem solving and it is always necessary to see information in new ways in order to turn data into gains for the company and its customers. Due to its simplicity and practicality, the Python language has become popular and has been widely used in this area. In this chapter we will introduce the basic teachings for data analysis and manipulation using Python and some of its main libraries.*

***Resumo***

*A análise de dados surgiu a partir do potencial de conhecimento obtido por meio de dados, tornando-se essencial para manutenção e sucesso de qualquer empresa. O dia a dia do profissional cientista de dados envolve resolução de problemas complexos e é necessário sempre enxergar as informações de novas formas afim de transformar dados em ganhos para a empresa e seus clientes. Por sua simplicidade e praticidade, a linguagem Python popularizou-se e vem sendo muito utilizada nessa área. Neste capitulo serão introduzidos os ensinamentos básicos para análise e manipulação de dados utilizando Python e algumas de suas principais bibliotecas.*

## 1.1. Introdução

Segundo o estudo "A Universe of Opportunities and Challenges", desenvolvido pela consultoria EMC Corporation, em 2020 o mundo terá criado aproximadamente 44 zettabytes, ou 44 trilhões de gigabytes de dados [Gantz and Reinsel 2012], ou seja, um volume jamais alcançado. Este grande volume de dados abre novos caminhos para tomada de decisões a partir de padrões que se revelam nestes dados.

A análise de dados é a ciência que torna possível a extração de informações deste imenso volume de dados. A partir disto é possível encontrar padrões, produzir informações úteis e predizer situações futuras com taxas de acerto bastante aceitáveis.

O presente trabalho aborda a análise de dados e do tratamento feito sobre estes dados descrevendo o processo de extração de informações supracitado bem como as ferramentas utilizadas para esta tarefa, tais como: a linguagem de programação Python e as bibliotecas Pandas e NumPy, além das plataformas de desenvolvimento IPython e Jupyter Notebook.

Estes são alguns dos temas que serão apresentados no decorrer do texto sobre o processo de descoberta de conhecimento em bases de dados: Coleta e pré-processamento, mineração de dados, ferramentas utilizadas e aprendizado de máquina detre outros que também serão tratados objetivando introduzir os conhecimentos básicos a respeito da análise e manipulação de dados utilizando Python e algumas de suas principais bibliotecas.

## 1.2. Análise de Dados

Até 2020 os investimentos em dispositivos conectados e no ecossistema digital devem aumentar em aproximadamente 40% no mundo [Gantz and Reinsel 2012]. Tal avanço nos investimentos abrem caminho para a produção da dados cada vez maior e faz com que um outro termo seja apontado: Big Data. Este termo diz respeito à análise de grandes quantidades de dados brutos para a extração de informações relevantes a algum negócio.

Esta análise e extração de dados é feita com base em linguagens e ferramentas de programação que auxiliam os pesquisadores e programadoras na tarefa de navegar por imensas quantidades de dados não estruturados e a conseguir categorizar, estruturar e dar sentido a estes dados.

### 1.2.1. Coleta e Pré-processamento

A coleta conhecida também como entendimento do domínio, compreende inicialmente entrevistas com o especialista de domínio. Sendo um ator essencial com capacidade de fornecer ao analista de dados informações com o intuito de facilitar o entendimento da aplicação e dos recursos disponíveis como:

- Pessoal e Dados;
- Hardware e Softwares;
- Existência de conhecimento prévio.

Segundo [de Castro Luiz and Martins 2015] a etapa de pré-processamento é responsável por captar, organizar e tratar os dados. Entre as principais funções podemos destacar:

1. Seleção de Dados ou Redução: após o entendimento do domínio do problema é necessária uma análise e seleção de informações que são consideradas relevantes ao longo do processo.

2. Limpeza de Dados: é a função que cuida do tratamento dos dados selecionados ao qual podem existir inconsistências, como campos sem preenchimento, ou preenchidos erroneamente assegurando qualidade.

3. Codificação de Dados: função responsável por codificar os dados para um formato que possa ser utilizado como entrada dos algoritmos utilizados nas etapas seguintes.

4. Enriquecimento de Dados: função que trabalha para obter o máximo de informações que possam ser agregadas aos dados e registros já existentes.

### 1.2.2. Mineração de Dados

Para [Fayyad et al. 1996] minerar dados inclui procurar padrões de interesse em uma forma representacional específica ou em um conjunto dessas representações, incluindo regras ou árvores de classificação, regressão, *clustering*, modelagem de sequência, dependência e análise de linha.

Em particular, os algoritmos de mineração de dados consistem basicamente em uma combinação específica de três componentes:

- Modelo: Contém parâmetros que devem ser determinados a partir dos dados através de seus fatores mais relevantes: função do modelo e forma representacional.

- Critério de preferência: Base para a preferência de um modelo ou conjunto de parâmetros em relação a outro, dependendo dos dados fornecidos.

- Algoritmo de busca: A especificação de um algoritmo para encontrar modelos e parâmetros específicos, dados fornecidos, um modelo (ou família de modelos) e um critério de preferência.

As funções de modelo mais comuns na prática atual de mineração de dados incluem:

- Classificação: mapeia (ou classifica) um item de dados em uma das várias classes categóricas predefinidas.

- Regressão: mapeia um item de dados para uma variável de previsão de valor real.

- *Clustering*: mapeia os dados em uma das várias classes categóricas (subconjuntos de *clusters*) nas quais as classes devem ser determinadas a partir dos dados.

- Sumarização: fornece um resumo (descrição) para um subconjunto de dados.

Quanto as representações mais populares de modelos de mineração estão árvores e regras de decisão, modelos lineares, modelos não lineares (por exemplo, redes neurais), métodos baseados em exemplos, modelos probabilísticos de dependência gráfica (por exemplo, redes bayesianas) e modelos de atributos relacionais.

### 1.2.3. Avaliação e Interpretação dos Resultados

Esta fase requer a visualização, a análise e a interpretação do modelo de conhecimento. É comum nesta etapa o especialista em descoberta de conhecimento juntamente com o especialista de domínio avaliar os resultados obtidos e propor novas possibilidades de investigação. O autor [Goldschmidt and Passos 2005] indica algumas operações possíveis no pós-processamento:

- Simplificação de modelo de conhecimento: operação responsável por remover detalhes do modelo de conhecimento com o objetivo de torná-lo menos complexo mas, sem perder informações relevantes. Um exemplo seria a representação de conhecimento com regras que em grande quantidade pode prejudicar a interpretação.
- Transformação de modelo de conhecimento: esse tipo de método na conversão da forma de representação de conhecimento de um modelo para outra forma de representação do mesmo modelo. Um ótimo exemplo é a conversão de árvores de decisão em regras ou vice-versa.
- Organização e apresentação de resultados: as representações de modelos de conhecimento podem acontecer de várias formas, conforme mencionado na subseção anterior. Dentre elas árvores, regras, gráficos em duas ou três dimensões, planilhas, tabelas entre outras úteis na representação do conhecimento. As técnicas de visualização em geral, buscam impulsionar a percepção e a inteligência humana com o objetivo de ampliar o entendimento e associação de novos padrões.

## 1.3. Ferramentas Utilizadas

A linguagem de programação Python possui bibliotecas amplamente testadas e necessárias para resolver efetivamente um amplo conjunto de problemas de análise de dados [McKinney 2012]. Python é a linguagem mais utilizada por cientistas ao redor do mundo para análise de dados [KDnuggets 2016].

Duas das principais bibliotecas do Python para análise de dados são a NumPy e a Pandas. Ambas possuem uma série de recursos e algoritmos essenciais para agilizar o tratamento dos dados, o que torna esta tarefa menos árdua.

O NumPy é o pacote fundamental para a computação científica com Python. Para dados numéricos, as matrizes NumPy são uma maneira muito mais eficiente de armazenar e manipular dados do que as outras estruturas de dados Python internas. Além disso, as bibliotecas escritas em uma linguagem de baixo nível, como C ou Fortran, podem operar nos dados armazenados em uma matriz NumPy sem copiar nenhum dado [McKinney 2012].

Outra biblioteca existente no Python e de extrema relevância para a ciência de dados é a Pandas. Fornece estruturas e funções de dados avançadas, projetadas para tornar o trabalho com dados estruturados rápido, fácil e expressivo. É um dos ingredientes críticos que permitem ao Python ser um ambiente de análise de dados poderoso e produtivo [McKinney 2012].

Para trabalhar com a linguagem Python dispondo de todos os seus recursos e bibliotecas, tais como as anteriormente citadas, uma das ferramentas mais bem aceitas pela comunidade científica é Jupyter Notebook. Uma plataforma de desenvolvimento intuitiva que facilita a tarefa da análise de dados bem como o aprendizado das ferramentas utilizadas para tal serviço.

O Jupyter Notebook é um aplicativo de código aberto que permite criar e compartilhar documentos que contêm código ativo, equações, visualizações e texto narrativo. Os usos incluem: limpeza e transformação de dados, simulação numérica, modelagem estatística, visualização de dados, aprendizado de máquina e muito mais [Jupyter 2019].

## 1.4. Numpy

Para apresentar os comandos do NumPy, serão utilizados dados do curso da Software Carpentry [Carpentry 2019]. Os dados referem-se a inflamação em pacientes que receberam um novo tratamento para artrite, serão analisados a primeira dúzia de conjuntos de dados. Os conjuntos de dados são armazenados em valores separados por vírgula no formato CSV. Cada linha contém informações para um único paciente, e as colunas representam dias sucessivos. O primeiro passo é carregar os dados, o Código Fonte a seguir apresenta o carregamento e exibe os dados carregados.

```
data = numpy.loadtxt(fname='inflammation-01.csv', delimiter=',')
print(data)
```

Resultado do Código Fonte:

```
[[0. 0. 1. ... 3. 0. 0.]
 [0. 1. 2. ... 1. 0. 1.]
 [0. 1. 1. ... 2. 1. 1.]
 ...
 [0. 1. 1. ... 1. 1. 1.]
 [0. 0. 0. ... 0. 2. 0.]
 [0. 0. 1. ... 1. 1. 0.]]
```

Inicialmente, pode-se verificar o tipo dos dados com o comando *type* que retorna a classe do tipo 'numpy.ndarray'. Para visualizar a forma dos dados utiliza-se *data.shape* que retorna as dimensões, que são 60 linhas e 40 colunas. Como foi apresentado, cada linha representa um paciente, e cada coluna representa um dia sucessivo.

```
# tipo
print('Tipo: ', type(data))

# dimensões
print('Dimensão:', data.shape)
```

Para obter um valor basta informar o índice do mesmo. No código a seguir é capturado o valor do paciente 0 no dia 0.

```
data[0, 0]
```

Também, é possível capturar um intervalo de valores utilizando dois pontos entre os índices. O Código Fonte a seguir apresenta a captura dos valores do paciente 0 até o 4, nos dias 0 até 10.

```
data[0:4, 0:10]
```

Outros comandos muito úteis que podem ser encontrados no NumPy são a média dos valores, o maior valor, o menor e o desvio padrão. O Código Fonte a seguir apresenta a tuilização dessas funções.

```
print ('Mean inflammation:', data.mean())
print ('maximum inflammation:', data.max())
print ('minimum inflammation:', data.min())
print ('standard deviation:', data.std())
```

Algumas vezes, é necessário especificar qual eixo quer utilizar no NumPy. O eixo 0 representa as linhas e o 1 as colunas. Supondo que deseja-se obter a inflamação média por paciente em todos os dias. Como é visto no Código Fonte a seguir, pode-se utilizar a função mean definindo-se o eixo = 1.

```
print (data.mean(axis=1))
```

O resultado do Código Fonte apresenta a inflamação média de cada paciente em todos os dias.

```
[5.45  5.425 6.1   5.9   5.55  6.225 5.975 6.65  6.625 6.525 6.775 5.8
 6.225 5.75  5.225 6.3   6.55  5.7   5.85  6.55  5.775 5.825 6.175 6.1
 5.8   6.425 6.05  6.025 6.175 6.55  6.175 6.35  6.725 6.125 7.075
   5.725
 5.925 6.15  6.075 5.75  5.975 5.725 6.3   5.9   6.75  5.925 7.225 6.15
 5.95  6.275 5.7   6.1   6.825 5.975 6.725 5.7   6.25  6.4   7.05  5.9
    ]
```

### 1.4.1. Plotando Informações

Durante a análise de dados, diversas vezes faz-se necessário a apresentação dos dados ou dos resultados de forma gráfica, para facilitar o entendimento e simplificar a visualização. Para plotar dados de forma gráfica pode-se utilizar a biblioteca *matplotlib*. A seguir tem-se a importação do mesmo.

```
%matplotlib inline
from matplotlib import pyplot
```

Para visualizar os dados de inflamação dos pacientes utiliza-se a função imshow apresentada a seguir.

```
pyplot.imshow(data)
pyplot.show()
```

A plotagem dos dados pode ser vista na Figura 1.1, onde nas regiões escuras neste mapa de calor são valores baixos, enquanto o centro que é claro representa valores altos. A inflamação aumenta e cai em um período de 40 dias.

O Código Fonte 1.1 apresenta como exibir um gráfico de linha dos valores. Para isso serão apresentadas a média, o máximo e o mínimo por dia em todos os pacientes.

```
plt.figure(figsize=(10.0, 3.0))

plt.subplot(1, 3, 1)
```

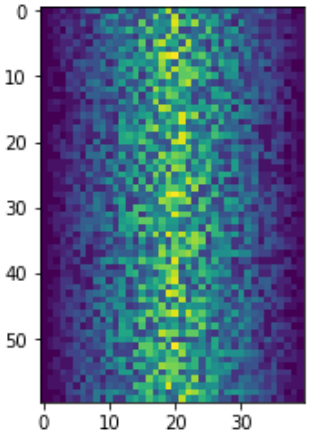



**Figure 1.1. Inflamações dos Pacientes**

```
plt.ylabel('average')
plt.plot(data.mean(0))

plt.subplot(1, 3, 2)
plt.ylabel('max')
plt.plot(data.max(0))

plt.subplot(1, 3, 3)
plt.ylabel('min')
plt.plot(data.min(0))

plt.tight_layout()
plt.show()
```

**Código Fonte 1.1. Inflamação média**

O resultado da média é aproximadamente um aumento linear e queda, o gráfico do valor máximo ficar na forma de um triangulo, já o mínimo forma escadas como pode ser visto na 1.2.

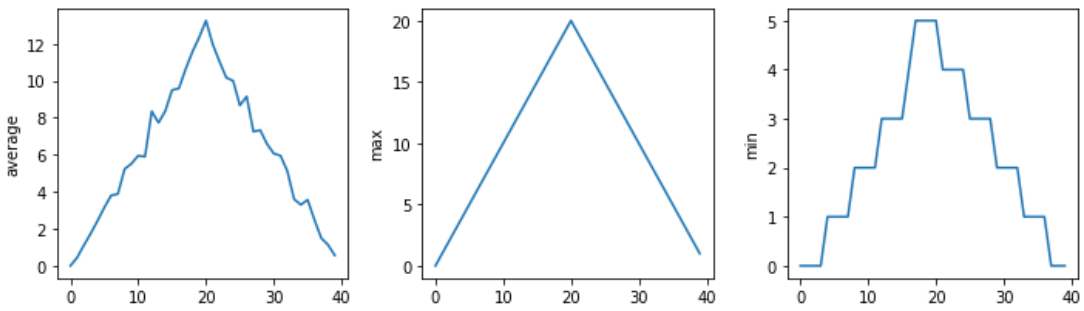



**Figure 1.2. Inflamação média, máxima e mínima por dia**

## 1.5. Pandas

O Pandas é uma biblioteca python para análise de dados que facilita a manipulação, leitura e visualização dos dados. Para utiliza-lo primeiro é necessário realizar a sua importação.

```
import pandas as pd
```

O Pandas trabalha com dois tipos principais de estruturas de dados: Series e DataFrames que serão apresentadas a seguir.

### 1.5.1. Series e DataFrame

Uma Series é igual a um Array unidimensional, ou uma lista. Possui um índice que dá rótulos a cada elemento da lista, como o Código Fonte a seguir que apresenta a criação de uma Series contendo os itens mouse, teclado, monitor e hd.

```
produtos = pd.Series(['mouse', 'teclado', 'monitor', 'hd'])
produtos
```

Saída do Codigo Fonte:

```
0      mouse
1    teclado
2    monitor
3         hd
dtype: object
```

Pode-se observar na saída do Código Fonte que cada item possui um índice que vai de 0 a n. Com isso, é possivel capturar qualquer elemento da Series apenas informando o seu indice, como no exemplo abaixo que retorna o 'mouse'.

```
produtos[0]
```

Em sua essência a Series é uma coluna e a DataFrame é uma tabela multidimensional composta por uma coleção de Séries. O código a seguir apresenta a criação de um DataFrame com informações de pessoas, ele possui a propriedade *data* que representa os dados que serão inseridos em cada linha e *columns* que representa as colunas.

```
DADOS = [['Rafael',27],['Daniel',25],['Nonato',35],['Kelly',24]]
COLUNAS = ['Nome','Idade',]
pessoas = pd.DataFrame(data=DADOS,columns=COLUNAS)
pessoas
```

A saída do Código Fonte é apresentada a seguir e segue a forma de uma tabela. Cada linha possui um índice e os dados podem ser acessados pelo valor de seu índice.

```
   Nome  Idade
0 Rafael  27
1 Daniel  25
2 Nonato  35
3 Kelly 24
```

Também é possível acessar todos os dados de uma coluna através de seu nome, como é apresentado no código a seguir que obtém como retorno apenas a coluna 'Nome'.

```
pessoas['Nome']
```

### 1.5.2. Analisando dados de compras

A análise é uma tarefa rotineira na vida de um cientista de dados. Como exemplo disso, pode-se imaginar um cenário hipotético de um cientista de dados que, em seu trabalho, recebeu a tarefa de analisar um conjunto de dados sobre compras realizadas em um *e-commerce*.

O *e-commerce* possuí uma área pessoal, em que o cliente deve autenticar-se para acessá-la, e pode realizar a aquisição de produtos que são oferecidos. Cada um dos produtos oferecidos possuí um valor para venda. Diversos dados do cliente, como sexo e idade, foram coletados e estão disponíveis no conjunto de dados.

Os dados presentes no conjunto citado, estão em um arquivo de formato JSON (*JavaScript Object Notationnde*). Tal formato facilita o armazenamento, transmissão e processamento dos dados, uma vez que este possuí uma estrutura bem simples, se comparado até mesmo com a estrutura de outros formatos com este mesmo fim, como o XML (*Extensible Markup Language*) recomendado pelo W3C (*World Wide Web Consortium*).

O trabalho do cientista, neste caso, é entregar uma análise comportamental de compra dos consumidores do *e-commerce*. Que no futuro será utilizado para alimentar um modelo de *Machine Learning*, onde serão realizadas diversas previsões sobre comportamentos futuros.

Inicialmente o cientista deve realizar a aquisição dos dados que serão analisados. Neste caso, os dados do comércio eletrônico hipotético foram disponibilizados por [Academy 2019] e serão utilizados ao longo desta subseção.

Após a aquisição do *dataset*, é hora carregá-lo na memória, já que ele será utilizado em toda a análise. Para isso, é necessário a alocação de uma variável chamada "endereco" que receberá o caminho do arquivo onde estão os dados, e será passada juntamente com um segundo parâmetro *(orientation)*, que indica o formato de sequência JSON esperado, à função responsável por realizar a leitura de arquivos do tipo JSON e que pertencente a biblioteca pandas.

```
# Endereco do arquivo
endereco = '/minicurso/dados_compras.json'
# Carregar arquivo
dados_compras = pd.read_json(endereco, orient = "values")
#Para ver toda a tabela de valores basta remover o head()
dados_compras.head()
```

**Código Fonte 1.2. Carregando os Dados de Compras**

O retorno do passo anterior é um *DataFrame* pandas, uma estrutura de dados bidimensional tabular que possuí todos os atributos do arquivo carregado. Esses dados são mostrados logo em seguida através do comando head, conforme ilustrado na figura 1.3.

A partir do *DataFrame* é possível acessar diversos dados pelo nome de colunas (*columns*) e/ou linhas (*rows*). No código 1.3, por exemplo, será mostrado o login do primeiro cliente, utilizando a posição 0 do aquivo índice do *Dataframe* carregado. Além disso, também é possível utilizar índices de outras formas para manipular os dados. Para exemplificar este último caso, é realizada uma busca utilizando o índice exclusivo

| | Idade | Item ID | Login | Nome do Item | Sexo | Valor |
|---|---|---|---|---|---|---|
| **0** | 38 | 165 | Aelalis34 | Bone Crushing Silver Skewer | Masculino | 3.37 |
| **1** | 21 | 119 | Eolo46 | Stormbringer, Dark Blade of Ending Misery | Masculino | 2.32 |
| **2** | 34 | 174 | Assastnya25 | Primitive Blade | Masculino | 2.46 |
| **3** | 21 | 92 | Pheusrical25 | Final Critic | Masculino | 1.36 |
| **4** | 23 | 63 | Aela59 | Stormfury Mace | Masculino | 1.27 |

**Figure 1.3. DataFrame Contendo os Atributos do Arquivo JSON**

*unique* em duas situações: retornar o número de compradores e obter o número de itens exclusivos, ambos os casos também são ilustrados no 1.3. Uma das vantagens de utilizar o índice *unique* no pandas é a sua velocidade, significativamente mais rápido que o "numpy.unique".

```
#Para acessar as posições do Dataframe
dados_compras['Login'][0]
#Mostrando o número total de copradores
print("Número total de compradores")
print(dados_compras['Login'].nunique())
```

**Código Fonte 1.3. Acessando Posições do *Dataframe***

A realização de operações matemáticas é outra tarefa corriqueira quando se analisa dados. O pandas disponibiliza diversas destas operações em suas funções. É possível realizar operações como: média, soma e contagem de itens utilizando poucos comandos.

No código 1.4 são verificados o preço médio de compras, o número total de compras, e o rendimento total das compras através de operações matemáticas utilizando o pandas.

```
# Preço médio de compra
print("Preço médio de compra")
print(dados_compras['Valor'].mean())

# Número total de compras
print("Numero total de compras")
print(dados_compras['Valor'].count())

# Rendimento total
print("Rendimento total")
print(dados_compras['Valor'].sum())
```

**Código Fonte 1.4. Operações Matemáticas com Pandas *Dataframe***

No processo de descoberta de conhecimento, mais especificamente na etapa de análise, é essencial a exibição da informação de forma amigável e detalhada. Diante disso, pode-se plotar os dados para que o valor das informações expressem o que realmente demonstram. Neste sentido, uma informação significativa, no cenário do comércio eletrônico, é o percentual de compradores pelo sexo do cliente. No código 1.5 é demon-

strado a construção de um gráfico em pizza que ilustra o percentual citado, o mesmo gráfico está plotado na 1.4.

```
# graficamente exibir o percentual de compradores pelo sexo
contagem = pd.value_counts(dados_compras['Sexo'])
contagem.plot(kind='pie', autopct='%.2f%%')
```

**Código Fonte 1.5. Gráfico de Clientes por Sexo *Dataframe***

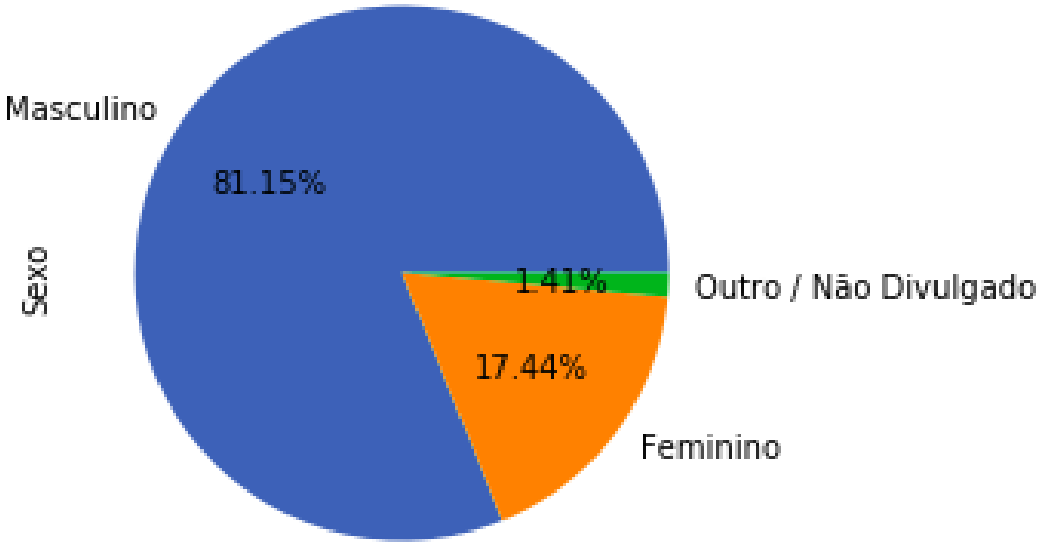



**Figure 1.4. Gráfico Pizza com Informações sobre Clientes por Sexo**

Dentre as informações importantes que devem ser extraídas dos dados do *e-commerce*, estão os cliente que mais realizam compras. O código 1.6 apresenta os cinco clientes que mais compraram, o resultado dessa instrução é ilustrado na 1.5.

```
# 5 consumidores que mais fizeram compras

logins = dados_compras.groupby(by="Login")
compras_numero = logins['Login'].aggregate(len).sort_values(ascending=
    False).head(5).to_frame('Top 5 numero de compras')
compras_numero
```

**Código Fonte 1.6. Cinco Clientes que Mais Realizaram Compras *Dataframe***

Por fim, será mostrado através do código 1.7, os produtos que mais tem dado lucro ao negócio na figura 1.6.

```
#####** Identificando os 5 itens mais lucrativos pelo valor total de
    compra e, em seguida, listando: **
####   ID do item   Nome do item   Número de compras   Preço do item
    Valor Total de Compra

### Pegar valor total de compra
```

Out[59]:

| Login | Top 5 numero de compras |
|---|---|
| Undirrala66 | 5 |
| Hailaphos89 | 4 |
| Mindimnya67 | 4 |
| Qarwen67 | 4 |
| Sondastan54 | 4 |

**Figure 1.5. Cinco Clientes que Mais Realizaram Compras**

```
itens_lucrativos = pd.DataFrame(dados_compras.groupby('Item ID')['Valor'].sum())

# Ordenar
itens_lucrativos.sort_values('Valor', ascending = False, inplace = True)

# Pegar apenas os 5 primeiros
itens_lucrativos = itens_lucrativos.iloc[0:5][:]

### Pegar o numero de compras
numero_de_compras = pd.DataFrame(dados_compras.groupby('Item ID')['Item ID'].count())

### concatenar a soma dos valores com o numero de compras
itens_lucrativos = pd.merge(itens_lucrativos, numero_de_compras, left_index = True, right_index = True, how = 'left')

# Pegar dados unicos
no_dup_items = dados_compras.drop_duplicates(['Item ID'], keep = 'last')

### Pegar os atributos que faltam com o no_dup_items
top5 = pd.merge(itens_lucrativos, no_dup_items, left_index = True, right_on = 'Item ID', how = 'left')

# Pegar atributos que queremos / <obs>: as colunas almejadas estão com os identificadores Item ID_x(quantidade), Valor_y(valor), Valor_x(soma total)
top5 = top5[['Item ID', 'Nome do Item', 'Item ID_x', 'Valor_y','Valor_x']]

# colocar para mostrar no index o id do item
top5.set_index(['Item ID'], inplace=True)

# Renomear os titulos das colunas
top5.rename(columns = {'Item ID_x': 'Número de compras', 'Valor_y': 'Item Valor', 'Valor_x': 'Valor Total de Compra'}, inplace = True)

# Formatar os valores para exibir com o simbolo do real (R$)
```

```
35 top5.style.format({'Item Valor': 'R$ {:.2f}', 'Valor Total de Compra':
     'R$ {:.2f}'})
```

**Código Fonte 1.7. Listando os Itens Mais Lucrativos *Dataframe***

| Item ID | Nome do Item | Número de compras | Item Valor | Valor Total de Compra |
|---|---|---|---|---|
| **34** | Retribution Axe | 9 | R$ 4.14 | R$ 37.26 |
| **115** | Spectral Diamond Doomblade | 7 | R$ 4.25 | R$ 29.75 |
| **32** | Orenmir | 6 | R$ 4.95 | R$ 29.70 |
| **103** | Singed Scalpel | 6 | R$ 4.87 | R$ 29.22 |
| **107** | Splitter, Foe Of Subtlety | 8 | R$ 3.61 | R$ 28.88 |

**Figure 1.6. Lista de Itens Mais Lucrativos**

## 1.6. Aprendizado de Máquina

Segundo [Norvig and Russell 2014] a IA (Inteligência Artificial) é um dos campos mais recentes em ciências e engenharia, iniciada logo após a segunda guerra mundial se popularizou e está presente em diversos tipos de estudos. De forma geral, envolve aprendizagem e percepção e pode estar até em atividades específicas como jogos de xadrez, criação de poesia e diagnóstico de doenças.

O aprendizado de máquina é um ramo da IA que se fundamenta na ideia de que sistemas podem aprender com dados para identificar padrões e tomar decisões com pouca intervenção humana. Existem diversos paradigmas de aprendizado de máquina, alguns baseados em aprendizado evolutivo (algoritmos genéticos), aprendizado estatístico (algoritmos de agrupamento) e dentre eles tem-se o aprendizado conexionista (redes neurais artificiais). As redes neurais artificiais (RNA) foram inspiradas no funcionamento dos neurônios biológicos e são muito utilizadas no reconhecimento de padrões sendo um dispositivo não-linear [Treleaven et al. 1989]. A seguir, será utilizado aprendizado de máquina para realizar analise de dados.

### 1.6.1. Boston Dataset

O pacote sklearn.datasets incorpora alguns conjuntos de dados comumente usados pela comunidade de aprendizado de máquina. Dentre eles, temos o Boston Dataset que contem o preço médio das casas em um subúrbio de Boston em meados da década de 1970.

Será desenvolvida uma rede neural para uma tarefa de regressão dos preços desses imóveis. Problemas de regressão tentam prever valores contínuos, como os preços das casas. Já problemas de classificação, tentam prever um rótulo discreto. Após importar a base e carrega-la em uma variável, é possível visualizar informações detalhadas através da função "DESCR". Essas informações incluem por exemplo: número de instâncias, número de atributos, informações de atributos, etc. A seguir são apresentados alguns atributos da base.

- Taxa de criminalidade per capita.
- O número médio de quartos por habitação.
- Distâncias ponderadas para cinco centros de emprego em Boston.
- Dentre outros...

As redes neurais normalmente possuem duas entradas, os atributos e o objetivo, o objetivo pode ser valores contínuos para regressão e rótulos/classes para classificação. Neste caso, os atributos são as informações e o objetivo é os preços das casas. O Código Fonte 1.8 apresenta o processo de importação e carregamento da base. Em seguida, é realizada a separação dos atributos e dos preços das casas em dois Dataframes distintos. Para o Dataframe dos atributos foi definido que as colunas terão os nomes dos atributos. Por fim é exibido as cinco primeiro linhas do dataframe atributos.

```
# Importando o dataset boston do scikit-learn
from sklearn.datasets import load_boston
boston = load_boston()

# Mostrar detalhes da base
#print(boston.DESCR)

# Transformar os precos das casas em um DataFrame pandas
preco_casas = pd.DataFrame(boston.target)

# Transformar os atributos um DataFrame pandas
atributos_casas = pd.DataFrame(boston.data)
atributos_casas.head()

# Definir as colunas com os nomes dos atributos
atributos_casas.columns = boston.feature_names
atributos_casas.head()
```

**Código Fonte 1.8. Obtendo os atributos e os preços das casas na base Boston**

A figura 1.7 apresenta a saída do Código Fonte 1.8 que contem o Dataframe dos atributos, onde as linhas indicam as casas e as colunas os atributos.

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0.0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1.0 | 296.0 | 15.3 | 396.90 | 4.98 |
| **1** | 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396.90 | 9.14 |
| **2** | 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392.83 | 4.03 |
| **3** | 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394.63 | 2.94 |
| **4** | 0.06905 | 0.0 | 2.18 | 0.0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3.0 | 222.0 | 18.7 | 396.90 | 5.33 |

**Figure 1.7. DataFrame Contendo os Atributos**

### 1.6.2. Random Forest Regressor

O Random Forest (Floresta Aleatória) é um algoritmo de aprendizagem de máquina fácil de usar e que produz ótimos resultados tanto para regressão quanto para classificação, na maioria das vezes, mesmo com pouco ajuste de hiperparâmetros. Diferente da arvore de decisão convencional a Floresta Aleatória proposta por [Breiman 2001] combina o uso de varias arvores de decisão com uma técnica chamada bootstrap Aggregration que possui uma camada de aleatoriedade. Cada nó é dividido usando o melhor dentre um subconjunto de preditores escolhidos aleatoriamente nesse nó.

Para utilizar o modelo Random Forest Regressor contido no sklearn devemos realizar a sua importação. Em seguida devemos criar o modelo, treina-lo e por fim ele será capaz de fazer predições. Inicialmente no O Código Fonte 1.9 é realizada a importação do *trein test split* que serve para fatiarmos nosso conjunto de dados em duas partes, a de treino e a de teste. O *test size* indica o percentual da divisão, no exemplo tem-se 30% para teste e os outro 70 para treino. O atributo *random state* garantira que quem utilize o mesmo parâmetro terá seus dados divididos da mesma forma que o obtido neste trabalho.

Após essa divisão, é realizada a criação do modelo, para isso alguns parâmetros podem ser alterados afim de melhorar os resultados. Na documentação é possível encontrar todos os parâmetros e seus possíveis valores. Em seguida o modelo é treinado pelo método *fit* que recebe os atributos e o objetivo. O método *predict* serve para realizar as predições. Nó código encontram-se duas métricas de avaliação. O *score* que indica a precisão de acertos da rede e o *mean squared error* (MSE) que é a perda de regressão ao erro quadrático médio. O resultado obtido foi um score = 90.50 e um MSE = 8.69.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Dividir a base em treino e teste
X_train, X_test, y_train, y_test = train_test_split(atributos_casas,
    preco_casas, test_size = 0.3, random_state = 1)

# Criando o modelo
regr = RandomForestRegressor(max_depth=None, random_state=50,
    n_estimators=100)

# Treinando a rede
regr.fit(X_train, y_train)

# Realizando a Predicao com o modelo treinado
print("score", regr.score(X_test, y_test)*100)
print("Mean squared error", mean_squared_error(y_test,regr.predict(
    X_test)))
```

**Código Fonte 1.9. Criação e Treinamento do Modelo**

Para visualização de forma mais intuitiva das predições, o Código Fonte 1.10 apresenta como utilizar o modelo treinado para predizer o preço de todas as casas e como realizar o plote de forma gráfica da comparação entre os preços originais e os preços preditos pelo nosso modelo treinado.

```
# Predizendo todos os precos e plotando
plt.title("Preco original X Preco Predito com random forest")

# precos preditos pela rede (azul)
plt.plot(regr.predict(atributos_casas[0:50]),'bo')

#precos reais (vermelho)
plt.plot(preco_casas[0:50],'ro')

plt.axis([-1,50,0,40])
plt.ylabel('Preco')
plt.xlabel('Casa')
plt.show()
```

**Código Fonte 1.10. Preço original X Preço Predito com random forest**

A figura 1.8 apresenta de vermelho os preços originais das casas e de azul os preços preditos pelo nosso modelo treinado. Obtém-se da analise que a rede erra por uma margem muito baixa, tendo uma precisão de 90.5% de acerto durante a predição, mostrando o potencial de se usar aprendizado de maquina para esse tipo de atividade.

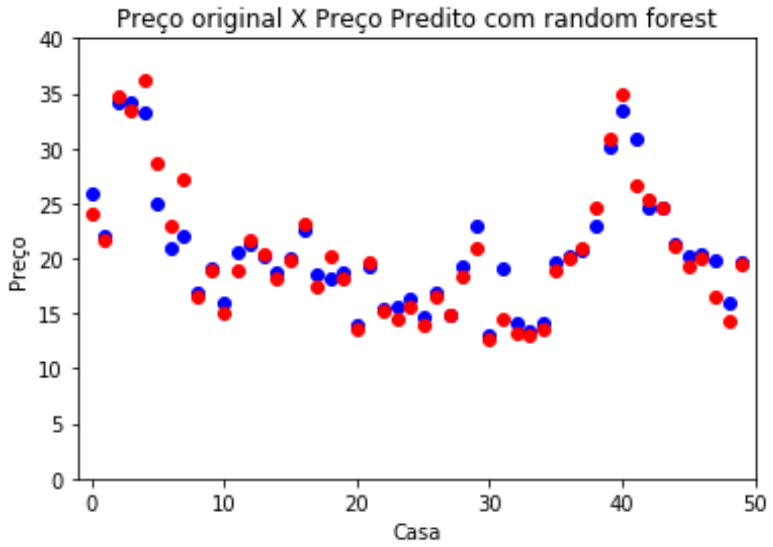



**Figure 1.8. Preço original X Preço Predito com random forest**

## 1.7. Conclusão

Neste capítulo foram mostrados conceitos sobre análise de dados utilizando a linguagem Python. Para isso, foram apresentadas as etapas do processo de descoberta de conhecimento em bases de dados desde a coleta dos dados até a avaliação e interpretação dos dados. Também, apresentou-se a linguagem de programação Python e algumas das principais bibliotecas utilizadas atualmente como Pandas e NumPy, além das plataformas de desenvolvimento IPython e Jupyter Notebook.

Percebe-se que a utilização das bibliotecas Pandas e NumPy fornecem enormes vantagens ao desenvolver sistemas que incluem maior agilidade e realização de tarefas

complexas de forma simples e com pouco código. Diversas operações foram realizadas como manipulações básicas de dados desde carregar, ver o tipo, ver o tamanho das dimensões, plotar graficamente, até operações mais complexas como a análise de dados reais. Ao final, foi apresentado como a aprendizagem de máquina pode ser utilizada para facilitar o processo de analise de dados.

Os trabalhos futuros poderão contemplar tarefas mais complexas, mostrando situações que exemplifiquem o dia-a-dia do profissional cientista de dados, bem como um maior possuir um maior foco na utilização da aprendizagem de maquina para resolução de diversos problemas.

## References

[Academy 2019] Academy, D. S. (2019). Python fundamentos para análise de dados. Python Fundamentos para Análise de Dados. https://www.datascienceacademy.com.br/course?courseid=python-fundamentos, note = Online; acessado 26 de Outubro 2019.

[Breiman 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Carpentry 2019] Carpentry, S. (2019). Programming with Python - Arthritis Inflammation. https://swcarpentry.github.io/python-novice-inflammation/. Online; acessado 24 de Outubro 2019.

[de Castro Luiz and Martins 2015] de Castro Luiz, A. R. J. and Martins, D. M. S. (2015). O uso da descoberta de conhecimento em banco de dados para extração e analise de informação: Estudo de caso. *Caderno de Estudos em Sistemas de Informação*, 2(2).

[Fayyad et al. 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

[Gantz and Reinsel 2012] Gantz, J. and Reinsel (2012). The Digital Universe In 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf. Online; acessado 24 de Outubro 2019.

[Goldschmidt and Passos 2005] Goldschmidt, R. and Passos, E. (2005). *Data mining: um guia prático*. Gulf Professional Publishing.

[Jupyter 2019] Jupyter (2019). Jupyter Home Page. https://jupyter.org/. Online; acessado 26 de Outubro 2019.

[KDnuggets 2016] KDnuggets (2016). Top 2016 KDnuggets Stories: Must-Know Data Science Interview QA, 10 Algorithms Machine Learning Engineers Need to Know. http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf. Online; acessado 24 de Outubro 2019.

[McKinney 2012] McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.".

[Norvig and Russell 2014] Norvig, P. and Russell, S. (2014). *Inteligência Artificial: Tradução da 3a Edição*, volume 1. Elsevier Brasil.

[Treleaven et al. 1989] Treleaven, P., Pacheco, M., and Vellasco, M. (1989). Vlsi architectures for neural networks. *IEEE micro*, 9(6):8–27.

# Capítulo 2

# Introdução a machine learning em python

Thiago J. B. Lima, Flávio H. D. Araújo, Pablo A. Vieira, Nonato R. de S. Carvalho, Rafael L. Araújo, Laécio A. Rodrigues

***Abstract***

*Machine Learning is one of the fields of Artificial Intelligence (AI) work that aims to use intelligent algorithms that are able to make computers learn through databases. This technique can be employed in different areas of the labor market, making the demand for qualified professionals increasingly high. This chapter will provide readers with theoretical and practical aspects of machine learning, focusing specifically on classification and regression tasks. The main techniques covered are: attribute selection, normalization, data standardization and evaluation metrics. At the end will be presented and solved two real-world problems (classification and regression).*

***Resumo***

*A área de Machine Learning (aprendizado de máquina) é um dos campos de trabalho da Inteligência Artificial (IA) que visa a utilização de algoritmos inteligentes que sejam capazes de fazer com que computadores aprendam por meio de base de dados. Essa técnica pode ser empregada em diferentes áreas no mercado de trabalho, tornando cada vez maior a procura por profissionais qualificados. Este capítulo irá proporcionar aos leitores aspectos teóricos e práticos da área de aprendizado de máquina, focando especificamente nas tarefas de classificação e regressão. As principais técnicas abordadas são: seleção de atributos, normalização, padronização de dados e métricas de avaliação. Ao final será realizada a apresentação e resolução de dois problemas do mundo real (classificação e regressão).*

## 2.1. Introdução

A área de aprendizado de máquina é um dos campos de trabalho da IA que visa a utilização de algoritmos inteligentes que sejam capazes de fazer com que computadores aprendam por meio de base de dados. Neste contexto, são utilizadas técnicas computacionais

que visam, de forma automática, resolver problemas simples e complexos de diferentes áreas.

O uso de técnicas de IA como sistemas de regressão e/ou classificação, possibilita automatizar tarefas em empresas, organização, entre outras. Com isso, é importante o estudo dessas técnicas para que sejam aplicadas de forma segura e eficiente. Essas técnicas podem ser empregadas em diferentes áreas no mercado de trabalho, como banco de dados autônomo, combate a fraudes em sistemas de pagamento, recomendação de conteúdo, previsão de preço de imóveis, entre outras. É importante salientar que os principais problemas encontrados podem ser interpretados como: problemas de classificação e regressão.

Este capítulo tem como objetivo apresentar uma visão geral da área de aprendizado de máquina, abordando aspectos teóricos e práticos, focando especificamente nas tarefas de classificação e regressão. Serão abordadas técnicas de seleção de atributos, normalização e padronização de dados, as principais métricas de avaliação e exemplos práticos de classificação e regressão.

## 2.2. Aprendizado de Máquina

Nos últimos 30 anos o aprendizado de máquina tornou-se multidisciplinar envolvendo disciplinas como teoria das probabilidades, estatística, teoria da oximação, analise convexa, complexidade computacional dentre outros. Aprendizado de máquina é um campo de trabalho da IA que visa a utilização de algoritmos inteligentes, ou seja, é um método de análise de dados que automatiza a construção de modelos analíticos. Sendo baseado na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana [Bishop 2006].

Um sistema de aprendizado é um programa de computador que toma decisões baseado em experiências acumuladas através da solução bem sucedida de problemas anteriores. Os diversos sistemas de aprendizado de máquina possuem características particulares e comuns que possibilitam sua classificação quanto à linguagem de descrição, modo, paradigma e forma de aprendizado utilizado. A Figura 2.1 apresenta um problema que poder ser resolvido com a aplicação de técnicas de aprendizado de máquina. O algoritmo tem como entrada **emails** e o objetivo é classificar como ***Spam*** ou **Não *Spam***. Este modelo pode ser aplicado em outros tipos problemas.

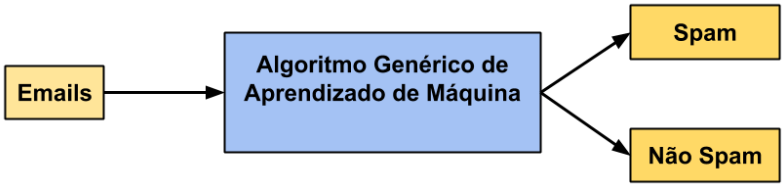



**Figura 2.1. Este algoritmo de aprendizagem de máquina é uma caixa preta que pode ser reutilizado em diferentes problemas de classificação[da Silva 2017].**

Os algoritmos de aprendizado de máquina podem ser organizados em vários tipos. Os tipos de algoritmo mais comuns são apresentados na Figura 2.2, onde, os que estão

em destaques serão abordados no **Problema proposto**:

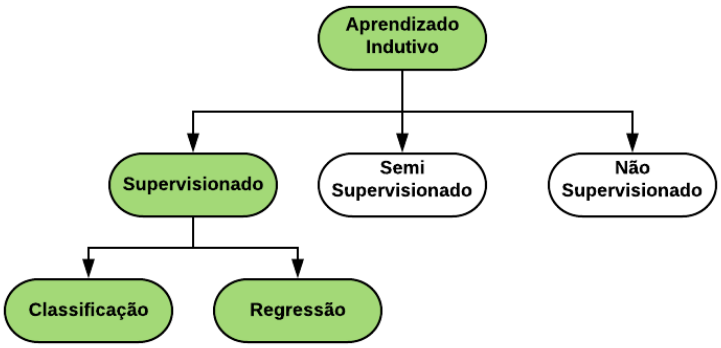



**Figura 2.2. Representação dos tipos de aprendizado de máquina.**

### 2.2.1. Inferência Indutiva

Na vida cotidiana, tem-se o pensamento que uma experiência anterior poderá ser importante quando se tem que aprender algo novo, que pode se relacionar com a experiência anterior. Por exemplo, estamos mais confiante em aprender um idioma semelhante ao nosso que um diferente. Esse tipo de expectativa, construída pela experiência da vida, nos leva a formular a seguinte pergunta: “Até que ponto a experiência adquirida em ‘aprender’ alguma classe de fenômenos C será útil para ‘aprender’ algumas outra classe’, relacionada de alguma forma a C?”. Para tentar responder essa pergunta, é preciso fazer referência a uma definição clara do “processo de aprendizagem” e a uma classe bem definida de objetos que é preciso “aprender”.

O aprendizado de máquina é algo complexo e bastante abstrato, o que torna o processo de generalizar e abstrair uma determinada "lei"algo que pode assumir *n* formas, onde podemos identificar todo o processo de aprendizado como inferência indutiva. Uma máquina de inferência indutiva M é qualquer dispositivo para calcular uma função recursiva parcial de N para N [Minicozzi 1976]. O problema de inferir uma função a partir de exemplos é complicado. Podemos pensar em pelo menos três problemas diferentes que podem surgir: **memória**, **calcular média** e **generalização**.

### 2.2.2. Supervisionado

O emprego de aprendizado supervisionado é comum em problemas de classificação, já que o objetivo é fazer com que o computador aprenda classificar dados. O reconhecimento de dígitos, é um exemplo de aprendizado de classificação. O aprendizado da classificação é apropriado para qualquer problema que a classificação seja fácil de determinar. Em alguns casos, pode não ser necessário atribuir classificações predeterminadas a todas as instâncias do problema. O aprendizado indutivo é o processo de aprender um conjunto de regras a partir de instâncias, que torna possível criar um método de classificação automática que pode ser usado para generalizar a partir de novas instâncias [Ayodele 2010].

### 2.2.3. Semi-Supervisionado

O aprendizado semi-supervisionado, assume que, juntamente com o conjunto de treinamento, há um segundo conjunto,não rotulado, também disponível durante o treinamento. Uma das metas do aprendizado semi-supervisionado é o treinamento de classificadores quando uma grande quantidade de exemplos não rotulados está disponível. A motivação para o aprendizado semi-supervisionado deve-se ao fato que, em muitas aplicações do mundo real, conjuntos de exemplos não rotulados são facilmente encontrados, quando comparados aos conjuntos de exemplos rotulados. Um outro fator é que exemplos não rotulados podem ser coletados de forma automática enquanto os rotulados necessitam de especialistas ou outros recursos de classificação [Ayodele 2010].

### 2.2.4. Não Supervisionado

O aprendizado não supervisionado é complexo: o objetivo é fazer o computador aprender a fazer algo que não é dito como fazer! Existem duas abordagens para o aprendizado não supervisionado. A primeira abordagem ensina ao agente não fornecendo categorizações explícitas, mas usa algum tipo de sistema de recompensa para indicar sucesso. Esse tipo de treinamento geralmente se encaixa na estrutura do problema de decisão pois o objetivo não é produzir uma classificação, mas tomar decisões que maximizem ou minimizem as recompensas. Essa abordagem funciona bem em problemas reais, onde os agentes podem ser recompensados por realizar determinadas ações e punidos por outras. O segundo é chamado de agrupamento. Nesse aprendizado, o objetivo não é maximizar uma função de utilidade, mas sim encontrar semelhanças nos dados de treinamento. A suposição é frequentemente que os *clusters* descobertos corresponderão razoavelmente bem a uma classificação intuitiva. Por exemplo, agrupar indivíduos com base na demografia pode resultar em agrupar os ricos em um grupo e os pobres em outro [Ayodele 2010].

## 2.3. Medida de desempenho

É preciso saber o quão bem o computador aprendeu. Algumas métricas nos ajudam a avaliar isso para que possamos comparar o desempenho entre diferentes modelos de aprendizado, diferentes parâmetros e diferentes conjuntos de treinamento. O benefício de separar o conjunto de dados em subconjuntos de treinamento e teste é que podemos testar o modelo em relação a dados nunca antes vistos durante a fase de treinamento e, assim, evitar erros de variação que ocorrem quando o modelo é excessivamente sensível aos dados com o qual foi treinado conhecido como *Overfiting* [Cárdenas-Montes 2006].

### 2.3.1. Conjunto de treinamento

Em processos de aprendizado de máquinas existe a necessidade que após o treinamento seja feito a validação do método, afim de garantir que o aprendizado realmente foi exitoso em conseguir abstrair e generalizar os dados de entrada. O conjunto de treinamento pode ser definido em uma porcentagem dos dados servindo para treinar um modelo. Na aprendizagem supervisionada, um algoritmo de aprendizado de máquina constrói um modelo examinando muitos exemplos e tentando encontrar um padrão que otimize a classificação ou regressão. Geralmente os dados são divididos em treino e teste de forma aleatória.

### 2.3.2. Conjunto de validação

O conjunto de validação é uma subdivisão do conjunto de treinamento, criando um conjunto menor, que será utilizado para verificar a eficiência do método quanto a sua capacidade de generalização durante o treinamento, e podendo ser empregado como critério de parada do treinamento.

### 2.3.3. Conjunto de teste

O conjunto de teste, é o restante dos dados que não foram utilizados, e serve para determinar a performance do aprendizado com dados que não foram previamente utilizados. A performance do método, medida nesta fase, é uma boa indicação de sua performance real. Ou seja,o método será testado com esse conjunto pra que seja possível verificar se as métricas foram suficientes para a obtenção de bons resultados.

## 2.4. Tecnologias utilizadas

Anaconda é uma distribuição de código aberto, considerada a maneira mais fácil de executar códigos em Python/R relacionados a ciência de dados e aprendizado de máquina, funcionando em diferentes sistemas operacionais como: *Windows* e *Mac OS X* e distribuições *Linux*. Com mais de 15 milhões de usuários em todo o mundo, é o padrão do setor para desenvolvimento, teste e treinamento em uma única máquina. A utilização do Anaconda possibilita a criação e configuração de diversos ambientes virtuais, como por exemplo ter dois o mais ambientes com versões diferentes da linguagem *Pyhton*[1].

Além do Anaconda este capitulo faz uso de outras tecnologias como: a linguagem de programação *Python* utilizada em muitos domínios como desenvolvimento *web*, científico e numérico, educação, desenvolvimento de *software*, aplicações de negócios, entre outros[2]. O *Jupyter Notebook* que é um aplicativo da *web* de código aberto que permite criar e compartilhar documentos que contêm código ativo, equações, visualizações e texto narrativo[3]. O *Pandas* que fornece estruturas de dados de alto desempenho e fáceis de usar fornecendo ferramentas de análise de dados para a linguagem de programação *Python*[4]. O *NumPy* é o pacote fundamental para a computação científica com *Python*. Ele contém um poderoso objeto de matriz N-dimensional, funções sofisticadas, ferramentas para integrar código C / C ++ e *Fortran*, entre outros[5]. O *Scikit-Learn*, uma ferramentas simples e eficientes para mineração e análise de dados acessível a todos e reutilizável em vários contextos[6].

O *Matplotlib* é uma biblioteca de plotagem 2D do *Python* que produz números de qualidade de publicação em vários formatos de cópia impressa e ambientes interativos entre plataformas[7]. Já O *Seaborn* é uma biblioteca de visualização de dados *Python* baseada no *matplotlib*. Ele fornece uma interface de alto nível para desenhar gráficos estatísticos

[1] https://www.anaconda.com/distribution/
[2] https://www.python.org/
[3] https://jupyter.org/
[4] https://pandas.pydata.org/
[5] https://numpy.org/
[6] https://scikit-learn.org/stable/
[7] https://matplotlib.org/

atraentes e informativos[8]. A Figura 2.3 apresenta um gráfico gerado com *Seaborn* a partir dos dados obtidos da base IRIS, onde, as especificações desta base serão apresentadas no **problema proposto**.

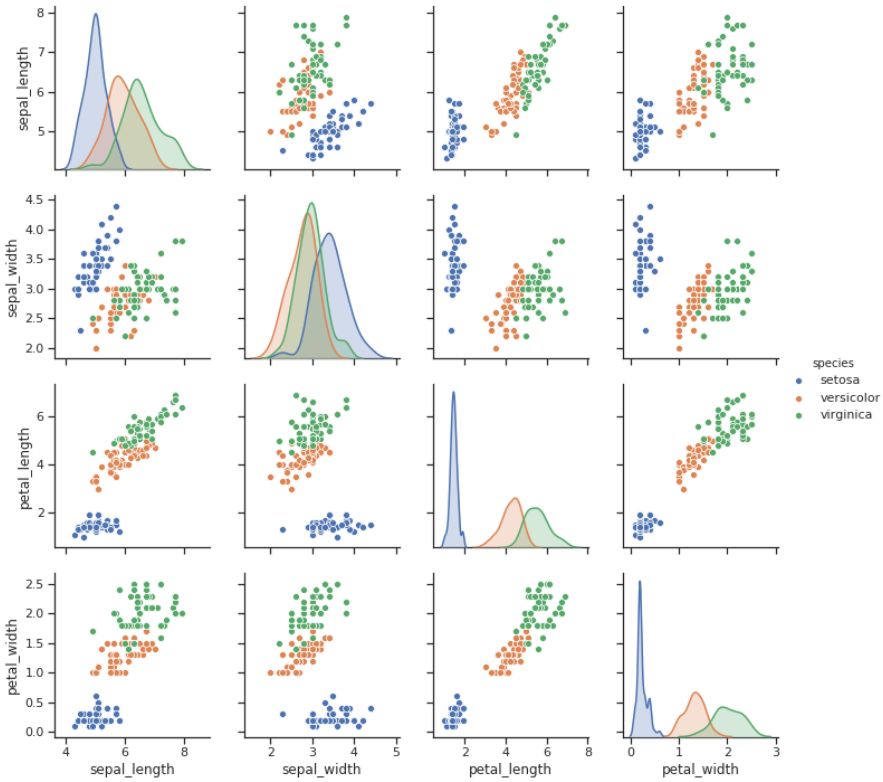



**Figura 2.3. Exemplo de uma plotagem de relacionamentos em pares em um conjunto de dados (Íris). Neste exemplo foi utilizada função *pairplot* do *Seaborn*.**

## 2.5. Tratamento dos dados

A qualidade dos dados é algo de suma importância para o aprendizado de máquinas, já que os algoritmos são frequentemente utilizados para extrair informações através dos dados de entrada. Uma vez que o aprendizado de máquina induz conhecimento estritamente a partir de dados, a qualidade do conhecimento extraído é determinada pela qualidade dos dados de entrada. O tratamento dos dados para o aprendizado de máquina é uma etapa fundamental para se obter sucesso, já que nem sempre os dados "brutos"de entrada estão preparados para a extração de informações.

Muitos aspectos influenciam no desempenho do aprendizado de máquinas já que esses algoritmos trabalham analisando uma grande carga de dados. Nas bases "reais"dois aspectos estão: **a)** relacionados com a presença de valores desconhecidos como os do tipo *NAN* e *string*, os quais devem ser tratados; **b)** a diferença dos números de exemplos ou registro de banco de dados, uma vez que essa diferença é grande os sistemas de aprendizado podem não conseguir generalizar as classes minoritárias [Batista 2003].

[8]seaborn.pydata.org

### 2.5.1. Seleção e criação de novos Atributos

A seleção dos atributos é uma etapa muito importante, já que geralmente a solução do problema começa a se resolver por ai. Essa etapa consiste em descobrir e encontrar um conjunto de atributos em que o aprendizado de máquinas poderá trabalhar. A necessidade de fazer uma seleção de atributos se da: Alguns algoritmos de aprendizado de máquina não conseguem trabalhar com grandes quantidades de atributos; Com um afunilamento da quantidade de atributos a generalização do aprendizado de máquina se torna maior; Alguns domínios tem alto custo de coletar dados, o que torna a utilização de métodos de seleção de atributos indispensáveis [Lee 2000].

Alguns atributos podem ser considerados inadequados, ou até mesmo irrelevante para o aprendizado dos algoritmos. Podemos atribuir níveis de relevância para os atributos: fracamente relevantes, condicionalmente relevantes ou inapropriados. Quando os atributos utilizados na classificação ou regressão são inapropriados, provavelmente sua classificação ou regressão serão imprecisas ou muito desafiadoras. Atributos de relevância fraca podem ser individualmente inadequados, porém a combinação desses atributos pode ser interessante trazendo bons resultados [Bloedorn 1998].

A transformação de dados tem como objetivo transformar a forma como os dados são representados superando as limitações existentes nos algoritmos de aprendizado, assim possibilitando e facilitando a extração de padrões. A transformação dos dados está ligada diretamente a qual proposito se tem e quais algoritmos de aprendizados serão empregados neste dados [Batista 2003].

### 2.5.2. Normalização e padronização

A normalização implica em transformar os atributos de seus intervalos naturais para um intervalo especifico, como por exemplo: [-1,1] ou [1,0]. Essa necessidade se da pois muitas vezes o intervalo dos dados são muito discrepantes, o que dificulta o aprendizado dos padrões. Esse tipo de transformação pode ser muito valiosa para algoritmos que calculam a distância entre os vizinho como o *KNN*. As redes neurais obtém melhores resultados quando são treinadas com atributos de valores pequenos. Porém quando se trata de métodos que utilizam árvores para a classificação ou regressão a normalização não traz uma grande contribuição, já que a normalização tende a diminuir algumas informações dos dados [Luai Al Shalabi 2006].

## 2.6. Classificação e Regressão

Em um problema de classificação é assumido que dados que possuem rótulos podem ser utilizados para criar regras que facilitem rotular novos dados que não possuem rótulos. A classificação pode associar o problema matematicamente. Por exemplo, um conjunto de dados no domínio D em Rl, indicando que os eventos no conjunto dependem de l características e formam um vetor de l dimensões. Funções matemáticas auxiliam a definir um classificador adequado para classificação de um determinado dado [Bell 2014].

Em estatística, a regressão é uma técnica que permite explorar e inferir a relação de uma variável dependente (variável de resposta) com variáveis independentes específicas (variáveis explicativas). A análise de regressão pode ser usada como um método

descritivo de análise de dados (por exemplo, ajuste de curva) sem exigir nenhuma suposição sobre os processos que geraram os dados. A regressão também designa uma equação matemática que descreve a relação entre duas ou mais variáveis [Izbicki and dos Santos ].

A regressão linear é uma abordagem linear para modelar o relacionamento entre uma resposta escalar (ou variável dependente ) e uma ou mais variáveis explicativas (ou variáveis independentes ). O caso de uma variável explicativa é chamado de regressão linear simples. Para mais de uma variável explicativa, o processo é chamado de regressão linear múltipla [Freedman 2009]. Uma linha de regressão linear tem uma equação da forma $Y = a + bX$ , onde $X$ é a variável explicativa e $Y$ é a variável dependente. A inclinação da reta é $b$ , e $a$ é a interceptação (o valor de $y$ quando $x = 0$).

A regressão não linear é uma forma de análise de regressão na qual os dados observacionais são modelados por uma função que é uma combinação não linear de parâmetros do modelo e depende de uma ou mais variáveis independentes. Os dados são ajustados por um método de aproximações sucessivas [Schittkowski 2013].

São inúmeras as técnicas para aprendizado de máquina, e nesta seção iremos abordar: *Multi-layer Perceptron* (MLP) e *Random Forest* (RF), que podem ser utilizados nas tarefas de classificação e regressão.

### 2.6.1. MLP

Rede neural artificial procura simular o cérebro humano, na biologia uma rede neural é composta por grupos de neurônios que se interconectam formando sinapses, que são estímulos feitos por pequenas cargas elétricas, o que possibilitam a comunicação entre os neurônios. Em uma rede neural biológica, os estímulos não possuem direções especificas para propagar entre os neurônios, o mesmo acontece coma rede neural artificial. Uma das redes artificias mais utilizadas pelos cientistas de dado é a MLP [Nicolas 2015].

Ao contrário de outras técnicas estatísticas, a MLP não faz suposições prévias sobre a distribuição dos dados. Ela pode modelar funções não-lineares e ser treinada para generalizar com precisão quando recebe dados não vistos antes. A MLP consiste num sistema de neurônios interconectados simples [Minsky 1954].

### 2.6.2. RF

A RF é um método de classificação supervisionado, que utiliza árvore de decisão como modelo de parametrização, integrado a uma técnica de amostragem e a abordagem de otimização, para construir o modelo classificador [Suthaharan 2016].

Uma árvore de decisão tem por objetivo construir um modelo que irá prever o valor de uma variável alvo, baseando-se no conjunto de variáveis de entrada. Cada árvore é composta de nós, que são associados a uma variável de entrada, estes nós possuem arestas que estão associadas a todos os valores possíveis para aquele nó. Uma folha representa o valor alvo possível para uma dada variável de entrada [Bell 2014]

O RF consiste numa combinação de árvores onde cada classificador é gerado usando um vetor aleatório independentemente do vetor de entrada. O RF utiliza o índice *Gini* como uma medida de seleção de atributo, medindo a impureza de um atributo em relação a classe [Ho 1995].

## 2.7. Métricas

Nesta seção apresentaremos as principais métricas mais utilizadas para modelos de classificação e/ou regressão considerando os conceitos já apresentados.

### 2.7.1. Acurácia

A medição constitui-se das operações necessárias para se determinar o valor de uma grandeza que se deseja conhecer. A medição baseia-se em princípios de medição: para medir a velocidade do ar, por exemplo, posso utilizar o princípio da diferença entre as pressões total e estática de um escoamento, recorrendo ao tubo de *Pitot*. Ou pode ser medido a variação da resistência elétrica de um fio muito fino que é resfriado pela corrente do fluido, assim os princípios são a variação da resistência elétrica do fio com a temperatura e a dissipação de calor do fio para a corrente do fluido, função de sua velocidade. O método de medição é a sequência de operações que realizo para obter o valor da grandeza ou mesurando [Unicamp 2019].

A acurácia (ou exatidão) e a precisão de um certo resultado de medição estarão sempre limitadas tanto pela sofisticação do equipamento utilizado, pela habilidade do sujeito que realiza a medida, pelos princípios físicos básicos tanto do instrumento de medida, quanto do fenômeno que gerou o experimento - princípio, método e procedimento de medição - e o conhecimento que se tem sobre o "valor verdadeiro"(definido ou consensual) da grandeza física [Unicamp 2019]. O cálculo da acurácia pode ser representado pela equação

$$acc = \frac{a}{a+e} \tag{1}$$

onde, a variável **acc** representa a acurácia calculada, a variável **a** representa o número de acerto e a variável **e** representa o número de erros. Logo, a acurácia é calculada a partir da divisão do número de acertos pelo total de amostras (acetos + erros).

### 2.7.2. R2_Score

O uso de $R^2$, o coeficiente de determinação, também chamado de coeficiente de correlação múltipla, está bem estabelecido na análise de regressão clássica [Rao et al. 1973]. Sua definição como a proporção da variância 'explicada' pelo modelo de regressão o torna útil como uma medida de sucesso na previsão da variável dependente a partir das variáveis independentes [Nagelkerke et al. 1991].

É desejável generalizar a definição de R 'para modelos mais gerais, para os quais o conceito de variação residual não pode ser facilmente definido, e a máxima probabilidade é o critério de ajuste. Enquanto os estatísticos tendem a não ter muito interesse em $R^2$, pesquisadores de outras disciplinas acham útil como uma maneira de descrever até que ponto um modelo estatístico se ajusta aos dados observados (uma medida da qualidade do ajuste) [Kramer 2005].

O $R^2$, diz o quanto meu modelo está prevendo corretamente. O cálculo dele pode ser representado pela Equação 2

$$R^2 = \frac{\sum (y-\bar{y})^2}{\sum (y-\hat{y})^2} \tag{2}$$

onde, $\sum (y-\bar{y})^2$ (Soma Total dos Quadrados): variação de y em torno da própria média. É o somatório das diferenças entre o valor alvo real e sua média elevado ao quadrado. E $\sum (y-\hat{y})^2$ (Soma dos Quadrados dos Resíduos): variação de Y que não é explicada pelo modelo elaborado. É o somatório das diferenças entre o valor predito e o valor real elevados ao quadrado [de Souza 2019b].

### 2.7.3. Erro Absoluto Médio - MAE

A medida mais simples da precisão da previsão é chamada de Erro Absoluto Médio (MAE). Erro absoluto médio é simplesmente, como o nome sugere, a média dos erros absolutos. O erro absoluto é o valor absoluto da diferença entre o valor previsto e o valor real. O erro absoluto médio mede a precisão de variáveis contínuas. O erro absoluto médio indica o tamanho médio de um erro que podemos esperar da previsão. O erro absoluto médio mede a magnitude média dos erros em um conjunto de previsões, sem considerar sua direção. O erro médio absoluto é a média na amostra de teste das diferenças absolutas entre previsão e observação real, em que todas as diferenças individuais têm peso igual. Tanto o erro absoluto médio quanto o erro do quadrado médio da raiz expressam o erro médio de previsão do modelo em unidades da variável de interesse. O erro médio absoluto e o erro quadrático médio da raiz podem variar de 0 a  e são indiferentes à direção dos erros [Wang and Lu 2018]. Como pode ser visto na Equação

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_{j-}\hat{y}_j\right| \tag{3}$$

onde, $\frac{1}{n}\sum_{j=1}^{n}$ é executado a operação de média e $(y_j-\hat{y}_j)^2$ se calcula o erro absoluto de cada dado.

### 2.7.4. Erro Quadrático Médio - MSE

Por mais de 50 anos, o Erro Quadrático Médio (MSE) tem sido a métrica de desempenho quantitativo dominante no campo do processamento de sinais. Continua sendo o critério padrão para avaliação da qualidade e fidelidade do sinal; é o método de escolha para comparar métodos e sistemas concorrentes de processamento de sinal e, talvez o mais importante, é a preferência quase onipresente de engenheiros de projeto que buscam otimizar os algoritmos de processamento de sinal. O objetivo de uma medida de fidelidade de sinal é comparar dois sinais, fornecendo uma pontuação quantitativa que descreve o grau de semelhança / fidelidade ou, inversamente, o nível de erro / distorção entre eles. Geralmente, supõe-se que um dos sinais seja um original original, enquanto o outro está distorcido ou contaminado por erros [Wang and Bovik 2009].

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left(y_j-\hat{y}_j\right)^2 \tag{4}$$

onde, $\frac{1}{n}\sum_{j=1}^{n}$ é executado a operação de média e $(y_j-\hat{y}_j)^2$ se calcula o erro quadrado de cada dado.

### 2.7.5. Matriz de Confusão

A matriz de confusão mostra as frequências de classificação para cada classe do modelo. Indica quantas classes foram classificadas corretamente e/ou incorretamente, mostrando as seguintes frequências: **Verdadeiro positivo (VP)**, ocorre quando no conjunto real, a classe que estamos buscando foi prevista corretamente. **Falso positivo (FP)**, ocorre quando no conjunto real, a classe que estamos buscando prever foi prevista incorretamente. **Falso verdadeiro (FV)**, ocorre quando no conjunto real, a classe que não estamos buscando prever foi prevista corretamente. **Falso negativo (FN)**, ocorre quando no conjunto real, a classe que não estamos buscando prever foi prevista incorretamente [de Souza 2019a].

Para ilustrar a matriz de confusão, iremos abordar um problema de classificação binária (com apenas duas classes) de nódulos pulmonares. Neste exemplo, nossas classes serão representadas por nódulos e não nódulo, considerado como supervisionado (as classes são conhecidas). Suponhamos que o problema é composto por um conjunto de imagens contendo N amostras. Hipoteticamente, foi realizada uma extração de características das imagens (nódulos e não nódulos) e em seguida aplicada uma classificação. Neste problema foi separada 10 amostras para validar nosso modelo treinado, contendo 5 nódulo e 5 não nódulos.

A Figura 2.4 ilustra uma matriz de confusão simulando a classificação binária dos nódulos pulmonar em: nódulo ou não nódulo. A Figura 2.4 (A) representa a matriz de confusão com dados e a Figura 2.4 (B) representa as categorias. A cor verde representa as decisões corretas e a cor vermelha representa as decisões incorretas. O primeiro caso (**Nódulo**) apresenta 4 acertos e 1 erro. O segundo caso (**Não Nódulo**) apresenta 3 acertos e 2 erros.

**(A)**

| | Nódulo | Não Nódulo |
|---|---|---|
| Nódulo | 4 | 1 |
| Não Nódulo | 2 | 3 |

**(B)**

| | Nódulo | Não Nódulo |
|---|---|---|
| Nódulo | VP | FP |
| Não Nódulo | FN | FV |

**Figura 2.4. Matriz de confusão simulando uma classificação binária (*Nódulo* ou *Não Nódulo*). (A) matriz de confusão usando dados e (B) matriz de confusão mostrando as categorias.**

## 2.8. Problema abordado

Com o intuito de prover aspectos teóricos e práticos, aqui iremos abordar dois tipos de problemas, um de classificação e outro de regressão. Em seguida apresentaremos a construção do algoritmo de classificação e regressão.

### 2.8.1. Classificação

No problema de classificação foi utilizada a base de dados **IRIS**[9], que é um dos banco de dados bastante conhecido e amplamente usado como um conjunto de dados para iniciantes para fins de aprendizado de máquina. O conjunto de dados contém 3 classes de 50 instâncias cada, em que cada classe se refere a um tipo de planta de íris.

Este é um tipo de problema classificado como supervisionado (onde as classes são conhecidos) podendo ser resolvido com algoritmo de classificação, pois a tarefa é classificar a espécie de planta. Abaixo serão apresentadas as informações sobre os atributos da base:

1. ***sepal_length***: comprimento da sépala em cm
2. ***sepal_width***: largura da sépala em cm
3. ***petal_length***: comprimento da pétala em cm
4. ***petal_width***: largura da pétala em cm
5. ***species***: setosa, versicolor e virgínica

A Tabela 2.1 apresenta um resumo estatístico sobre a base IRIS. Apresentamos as características de cada planta, relacionadas a elas temos: o valor mínimo, valor máximo, a média, o desvio padrão e a correlação da classe. Podemos observar que a correlação do comprimento da pétala e largura da pétala estão bem altos.

**Tabela 2.1. Resumo estatístico sobre a base IRIS, contendo as seguintes informações: valor mínimo, valor máximo, média, desvio padrão e correlação da classe.**

| | Mínimo | Máximo | Media | Desvio padrão | Correlação |
|---|---|---|---|---|---|
| comprimento da sépala | 4,3 | 7,9 | 5,84 | 0,83 | 0,7826 |
| largura da sépala | 2,0 | 4,4 | 3,05 | 0,43 | -0,4194 |
| comprimento da pétala | 1,0 | 6,9 | 3,76 | 1,76 | 0,9490 |
| largura da pétala | 0,1 | 2,5 | 1,20 | 0,76 | 0,9565 |

A implementação do código fonte para classificação da base IRIS, inicia-se com a importação das principais bibliotecas, que serão apresentadas no código 2.1. Após essa importação, iremos fazer a leitura da base utilizando o comando *read_csv* do pandas, como mostra no código 2.2.

```
1 import numpy as np
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.neural_network import MLPClassifier
5 from sklearn.ensemble import RandomForestClassifier
```

[9]https://archive.ics.uci.edu/ml/datasets/iris

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.datasets import base
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
%matplotlib inline
```

**Código Fonte 2.1. Principais pacotes utilizados.**

```
#Lendo a base de dados
dados = pd.read_csv("iris.csv")
#Listando os seis primeiros registros
dados.head(6)
```

**Código Fonte 2.2. Leitura da base e listagem dos seis primeiros registros.**

A Tabela 2.2 ilustra os seis primeiros registros extraídos da base IRIS. Foi utilizando o comando *head(6)* do pandas, que está presente no código 2.2. Podemos listar mais ou menos registros, basta alterar o valor parâmetro da função *head()*, que por padrão serão listados os cinco primeiros registros.

**Tabela 2.2. Imprimindo os seis primeiros registros da base IRIS, utilizando o comando *head(6)* do pandas.**

| sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|
| 5,1 | 3,5 | 1,4 | 0,2 | setosa |
| 4,9 | 3,0 | 1,4 | 0,2 | setosa |
| 4,7 | 3,2 | 1,3 | 0,2 | setosa |
| 4,6 | 3,1 | 1,5 | 0,2 | setosa |
| 5,0 | 3,6 | 1,4 | 0,2 | setosa |
| 5,4 | 3,9 | 1,7 | 0,4 | setosa |

Na etapa de tratamento de dados, podemos nos depara com inúmeros problemas. Esta base apresenta classes que são representadas por dados categóricos, então, utilizamos o comando *loc* do pandas para substituir esses dados por valores inteiros como, mostra o código 2.3. Os dados da coluna *species* (setosa, versicolor e virginica) foram substituídos por valores inteiros (0, 1 e 2), respectivamente. Vale ressaltar que o comando *loc* é primariamente baseado nas *labels* das colunas, também podemos utiliza-lo com um *array booleano*.

```
dados.loc[dados['species'] == 'setosa','species'] = 0
dados.loc[dados['species'] == 'versicolor','species'] = 1
dados.loc[dados['species'] == 'virginica','species'] = 2
```

**Código Fonte 2.3. Substituindo dados categóricos da coluna *species* por valores inteiros.**

Na etapa de visualização dos dados, podemos utilizar comandos específicos do *Seaborn* apresentado na Figura 2.3 como também podemos visualizar a distribuição dos dados em um gráfico 3D. O código 2.4 apresenta um *plot* 3D referente a distribuição dos dados da base IRIS em função de três atributos (*sepal_length*, *sepal_width* e *petal_length*).

```
feature_number1 = 0  # sepal_length
feature_number2 = 1  # sepal_width
feature_number3 = 2  # petal_length

# Define o tamanho da figura a ser plotada
fig = plt.figure(figsize=(6, 4))
# Define Axes3D
ax = Axes3D(fig, elev=-150, azim=110)
# Seta os valores dos atributos sepal_length (eixo X), sepal_width (
    eixo Y), petal_length (eixo Z)

scatter = ax.scatter(dados.iloc[0:150, feature_number1],
            dados.iloc[0:150, feature_number2],
            dados.iloc[0:150, feature_number3],
            # seta os valores do atributo species (classe)
            c=dados.iloc[0:150, 4],
            label = ['jkjk'],
            edgecolor='k', s=40)
# Título
ax.set_title("Distribuição dos dados (A)")
# Adiciona os nomes dos atribrutos em cada eixo correspondente
ax.set_xlabel(dados.columns[feature_number1])
ax.set_ylabel(dados.columns[feature_number2])
ax.set_zlabel(dados.columns[feature_number3])

# Adicionar legenda
elem_legenda = ax.legend(*scatter.legend_elements(),
                    loc="best", title="Classe")
# Alterar os valores da legenda
elem_legenda.get_texts()[0].set_text('setosa')
elem_legenda.get_texts()[1].set_text('versicolor')
elem_legenda.get_texts()[2].set_text('virginica')
plt.show()
```

**Código Fonte 2.4. Distribuição dos dados.**

A Figura 2.5 apresenta os gráficos gerados a partir do código 2.4 em função dos atributos (*sepal_length*, *sepal_width* e *petal_length*) representado pela Figura 2.5 **(A)** e em função dos atributos (*sepal_width*, *petal_length* e *petal_width*) representado pela Figura 2.5 **(B)**. Na Figura 2.5 **(A)** observamos que as classes versicolor e virgínica apresenta valores bem próximos para os atributos em questão. Já na Figura 2.5 **(B)** as três classes possuem valores bem próximos, neste cenário dificultará a classificação com relação aos atributos (*sepal_width*, *petal_length* e *petal_width*).

Para realizar a classificação será necessário separar a base em dois conjunto X e Y. O conjunto X é representado pelos atributos que representam as características das plantas

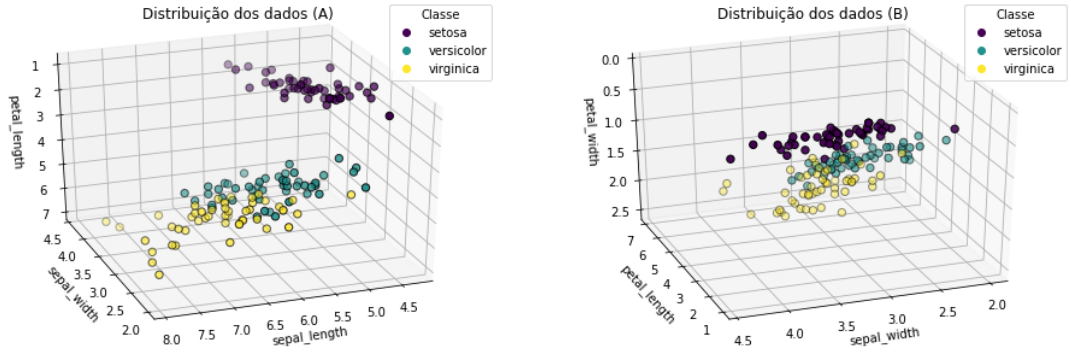



**Figura 2.5. Representação da distribuição dos dados da base IRIS em função dos atributos (*sepal_length*, *sepal_width* e *petal_length*) (A) e em função dos atributos (*sepal_width*, *petal_length* e *petal_width*) (B).**

(*sepal_length, sepal_width, petal_length, petal_width*). Já o conjunto Y contém os *labels* que indica a espécie de cada planta (*species*). Após essa divisão será necessário dividir os conjunto X e Y em treino (70%) e teste(30%). A função *train_test_split()* possibilita fazer essa divisão.

O código 2.5 contém os procedimentos para separar a base, dividir em treino e teste, como também exibe algumas informações importantes como: o *shape* de cada conjunto X e Y, e as quantidades de amostras selecionadas para os conjuntos de treino e teste. Um parâmetro importante da função*train_test_split()* é o *random_state*. Adicionando um valor inteiro a esse parâmetro, podemos embaralhar os dados e parâmetro garantir que sempre serão selecionados os mesmos dados. Pode ser utilizado em diferentes computadores, basta que o valor do *random_state* seja o mesmo para todos. A função garante que sempre serão selecionados o mesmos conjuntos dados para cada máquina. É importante salientar que a base de dados tem que ser a mesma.

```
# Obtendo todos os atributos, exceto a classe
X = dados.iloc[:, 0:-1]
# Obtendo apenas a classe
Y = dados.iloc[:, -1]
# Separando o conjunto em Treino(70%) e Teste(30%)
X_treino, X_teste, Y_treino, Y_teste = train_test_split(X, Y, test_size
    =0.3) # random_state=2
# Obtendo a quantidade de amostras do conjunto de Treino
qtd_treino = Y_treino.value_counts()
# Obtendo a quantidade de amostras do conjunto de Teste
qtd_teste = Y_teste.value_counts()
# Imprimindo algumas informações
print('Shape do conjunto X: ', X.shape)
print('Shape do conjunto Y: ', Y.shape)
print('Quantidade de amostras(TREINO):\n', qtd_treino)
print('Quantidade de amostras(TESTE):\n', qtd_teste)

# Resultado dos prints acima:
Shape do conjunto X:  (150, 4)
Shape do conjunto Y:  (150,)
```

```
Quantidade de amostras(TREINO):
1     36
0     36
2     33
Name: species, dtype: int64
Quantidade de amostras(TESTE):
2     17
1     14
0     14
Name: species, dtype: int64
```

**Código Fonte 2.5. Dividindo o conjunto em treino (70%) e teste (30%)**

O código 2.6 apresenta o treinamento do modelo MPL. Na linha 2, o modelo foi definido com duas camadas ocultas, contendo 10 neurônios em cada camada. Também foi definido 1.000 épocas, ou seja, a rede irá treinar por 1.000 iterações. Em seguida na linha 4 será realizado o treinamento com o conjunto de treino e na linha 6 será realizada a predição, utilizando o conjunto de teste. Após realizamos a predição, calculamos a matriz de confusão e acurácia, como é apresentado nas linhas 8 e 10, respectivamente. Os resultados da matriz de confusão e acurácia serão impressos posteriormente. Obtivemos uma acurácia de 97,77% com apenas um erro, onde, a classe real é 1 e o modelo classificou como sendo da classe 2.

```
1  # Definindo um modelo com duas camadas ocultas contendo 10 neurôneos em
       cada, e 1500 épocas
2  MLP = MLPClassifier(hidden_layer_sizes=(20,10), max_iter=1500, verbose=
       False)
3  # Realizando o treinamento
4  MLP.fit(X_treino, Y_treino)
5  # Realizando a predição
6  Y_predito = MLP.predict(X_teste)
7  # Calculando a matriz de confusão conjunto de teste
8  matriz_confusao = confusion_matrix(Y_teste, Y_predito)
9  # Calculando a acurácia do conjunto de teste
10 acuracia = accuracy_score(Y_teste, Y_predito)
11 # Imprimindo a matriz de confusão
12 print('Matriz de Confusão: \n',matriz_confusao)
13 # Imprimindo a acurácia
14 print('Acurácia = ', acuracia)
15
16 # Resultado da predição do conjunto de teste após realizar o
       treinamento com a MLP.
17 Matriz de Confusão:
18  [[14  0  0]
19  [ 0 17  1]
20  [ 0  0 13]]
21 Acurácia =  0.9777777777777777
```

**Código Fonte 2.6. Treinamento e teste com a MLP**

Utilizando o classificador *Random Forest* com os parâmetros *max_depth* igual a 40 e *n_estimators* igual a 50, obtivemos uma acurácia de 95,55%. Errou apenas para duas

amostras, que pertencia a classe 1 e classificou como sendo da classe 2.

```
# Definindo um modelo RandomForest com max_depth=40 e n_estimators=50
RF = RandomForestClassifier(max_depth=40, n_estimators=50)
# Realizando o treinamento
RF.fit(X_treino,Y_treino)
# Realizando a predição
Y_predito = RF.predict(X_teste)
# Calculando a matriz de confusão conjunto de teste
matriz_confusao = confusion_matrix(Y_teste, Y_predito)
# Calculando a acurácia do conjunto de teste
acuracia = accuracy_score(Y_teste, Y_predito)
# Imprimindo a matriz de confusão
print('Matriz de Confusão: \n',matriz_confusao)
# Imprimindo a acurácia
print('Acurácia = ', acuracia)

# Resultado da predição do conjunto de teste após realizar o
    treinamento com o \textit{Random Forest}.
Matriz de Confusão:
 [[14  0  0]
 [ 0 16  2]
 [ 0  0 13]]
Acurácia =  0.9555555555555556
```

**Código Fonte 2.7. Python example**

Além de calcular as métricas para o conjunto de teste, podemos fazer o mesmo procedimento no conjunto de treino. O código 2.8 ilustra uma função para imprimir as métricas acurácia e matriz de confusão do conjunto de treino e teste. Basta passar por parâmetro o modelo treinado e os conjuntos de treino e teste.

```
def metricas_treino_teste(modelo, X_treino, X_teste, Y_treino, Y_teste)
    :
    # Realizando a predição no conjunto de treino
    valores_preditos_treinamento = modelo.predict(X_treino)
    # Realizando a predição no conjunto de teste
    valores_preditos_teste = modelo.predict(X_teste)
    # Calculando a acurácia do conjunto de Teste
    acuracia_teste = accuracy_score(Y_teste, valores_preditos_teste)
    # Calculando a acurácia do conjunto de Treino
    acuracia_treinamento = accuracy_score(Y_treino,
        valores_preditos_treinamento)
    # Calculando a matriz de confusão do conjunto de Treino
    matriz_confusao_treino = confusion_matrix(Y_treino,
        valores_preditos_treinamento)
    # Calculando a matriz de confusão do conjunto de Teste
    matriz_confusao_teste = confusion_matrix(Y_teste,
        valores_preditos_teste)

    # Imprimir as métricas
    print('\nMatriz de Confusão - Treino: \n', matriz_confusao_treino)
```

```
    print('Acurácia treino = ', acuracia_treinamento)
    print('\nMatriz de Confusão - Teste: \n', matriz_confusao_teste)
    print('Acurácia teste = ', acuracia_teste)
```

**Código Fonte 2.8. Função para calcular e imprimir as métricas do conjunto de treino e teste.**

Para visualizarmos os cálculos da matriz de confusão e acurácia do conjunto de treino e teste, basta chamar a função *def metricas_treino_teste()* passando o modelo treinado os conjuntos de treino e teste. O código 2.9 mostra o resultado das métricas referentes ao treinamento e teste com classificador *Random Forest*.

```
# Executando a função metricas_treino_teste
metricas_treino_teste(RF, X_treino, X_teste, Y_treino, Y_teste)

# Resultados após executar a função metricas_treino_teste
Matriz de Confusão - Treino:
 [[36  0  0]
 [ 0 32  0]
 [ 0  0 37]]
Acurácia treino =  1.0

Matriz de Confusão - Teste:
 [[14  0  0]
 [ 0 16  2]
 [ 0  0 13]]
Acurácia teste =  0.9555555555555556
```

**Código Fonte 2.9. Executando a função metricas_treino_teste.**

### 2.8.2. Regressão

Na etapa de regressão abordaremos o seguinte problema: "Prevendo o preço de uma pizza de acordo com o seu diâmetro". Utilizamos as seguintes informações: o tamanho da pizza em diâmetro e o preço. Foram definidos dois *arrays*: um para representar o diâmetro da pizza (eixo X) e outro para representar o preço (eixo Y). No código 2.10 aprestamos as principais bibliotecas utilizadas para resolver este problema de regressão.

```
import numpy as np
from sklearn import linear_model
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, mean_absolute_error,
    r2_score
import matplotlib.pyplot as plt
%matplotlib inline
```

**Código Fonte 2.10. Importando as principais bibliotecas utilizadas.**

O código 2.11 define os atributos **diametros** e **precos**, como também a construção

de um gráfico em 2D utilizando a biblioteca *matplotlib*. Vale ressaltar que a distribuição dos preços das pizzas em relação ao tamanho do diâmetro será apresentado na Figura 2.6.

```
# Diâmetros (cm)
diametros = np.array([7, 10, 15, 30, 45])
# Mudando o formato do array onde cada linha representa um exemplo
diametros = np.reshape(diametros,(-1,1))
# Preços (R$)
precos = np.array([8, 11, 16, 38.5, 52])
# Definindo um plot
plt.figure()
# Rótulo do eixo X
plt.xlabel('Diâmetro em cm')
# Rótulo do eixo Y
plt.ylabel('Preço em reais')
# Adicionando os valores do diâmetros e preços ao plot.
plt.plot(diametros, precos,'o', label='Preço')
# Definindo os valores mínimo e máximo do gráfico (Eixos X e Y), neste
    caso de 0 a 60 para cada eixo.
plt.axis([0,60,0,60])
# Adicionando legenda
plt.legend()
# Adicionando grade
plt.grid()
plt.show()
```

**Código Fonte 2.11. Definião dos atributos (diametros e precos) e construção de um *plot* 2D para visualização dos dados.**

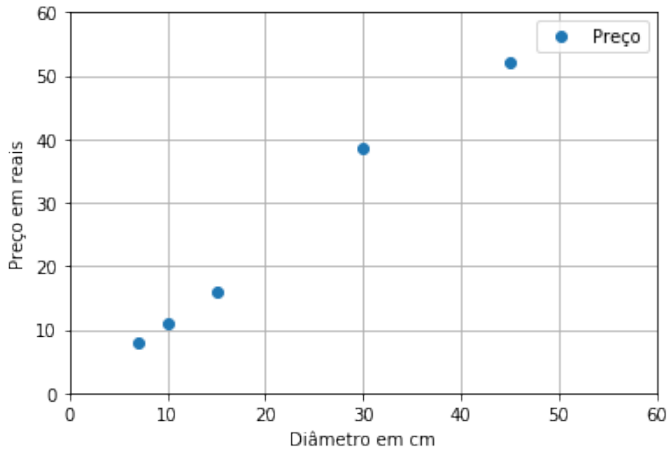



**Figura 2.6. Gráfico que representa a distribuição do preço da pizza em relação ao diâmetro.**

Aplicando uma regressão linear ao nosso problemas, podemos realizar um treinamento e predição. Como o objetivo deste problema proposto é predizer o valor da pizza passando apenas o diâmetro, utilizamos inicialmente o *LinearRegression*(), presente no . O código 2.12 ilustra a aplicação da regressão linear, onde, definimos o modelo (linha 2), realizamos o treinamento (linha 4), realizamos a predição a partir do modelo treinado

(linha 6), e calculamos e imprimimos as principais métricas (linhas 8, 9 e 10). Podemos observar que tivemos 99,07% de acerto, o MSE foi de 2,74 e MAE de 1,31.

```
1  # Definindo o modelo Linear Regression
2  modelo_LR = linear_model.LinearRegression()
3  # Realizando o treinamento
4  modelo_LR.fit(diametros,precos)
5  # Realizando a predição
6  precos_preditos_lr = modelo_LR.predict(diametros)
7  # Imprimindo métricas (MSE, MAE e SCORE)
8  print("MSE = ", mean_squared_error(precos, precos_preditos_lr))
9  print("MAE = ", mean_absolute_error(precos, precos_preditos_lr))
10 print("SCORE = ", r2_score(precos, precos_preditos_lr))
11
12 # Definindo um plot, adicionando os valores preditos
13 plt.figure()
14 # Rótulo do eixo X
15 plt.xlabel('Diâmetro em cm')
16 # Rótulo do eixo Y
17 plt.ylabel('Preço em reais')
18 # Adicionando os valores do diâmetros e preços ao plot.
19 plt.plot(diametros, precos,'o', label='Preço')
20 # Adicionando os valores preditos
21 plt.plot(diametros, precos_preditos_lr, linewidth=2, label='Preço
       Predito')
22 # Definindo os valores mínimo e máximo do gráfico (Eixos X e Y), neste
       caso de 0 a 60 para cada eixo.
23 plt.axis([0,60,0,60])
24 # Adicionar legenda
25 plt.legend()
26 # Adicionando grade
27 plt.grid()
28 plt.show()
29
30 # Resultado após executar o código (prints).
31 MSE =  2.7420035671819223
32 MAE =  1.3137931034482773
33 SCORE =  0.9907189156269228
```

**Código Fonte 2.12. Definindo um modelo *Linear Regression*.**

A Figura 2.7 mostra um gráfico com os dados do problema (diâmetros e preços) e ainda os preços preditos pelo modelo de regressão linear (apresentado no código 2.12). A Linha em laranja representa os valores preditos pelo modelo treinado.

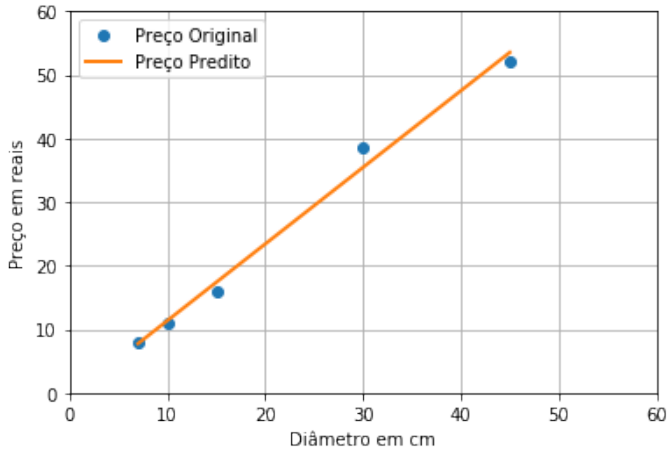



**Figura 2.7. Gráfico que representa a distribuição do preço da pizza em relação ao diâmetro e a adição dos preços preditos, que estão representados por uma reta.**

Na Regressão polinomial, mesmo sem saber a forma funcional do processo gerador de dados, podemos desenvolver uma técnica geral que funciona bem para problemas de relações não lineares. O código 2.13 apresenta a implementação de duas funções: a *treinameno_p()* para treinamento e predição, e **print_metricas()** usada para imprimir as principais métricas. A função *treinameno_p()* recebe por parâmetro o *degree*, diâmetros e preços, e retorna os preços preditos. Já a função print_metricas() recebe os preços reais e os preços preditos, e imprime as métricas: **MSE**, **MAE** e ***SCORE***.

```
1
2  def treinameno_p(degree, d, p):
3      # Definindo o modelo
4      modelo_polinomial = make_pipeline(PolynomialFeatures(degree), Ridge
           ())
5      # Realizando o trenamento
6      modelo_polinomial.fit(d, p)
7      # Retornando a predição
8      return modelo_polinomial.predict(d)
9
10 def print_metricas(precos, precos_preditos):
11     # Imprimindo métricas (MSE, MAE e SCORE)
12     print("MSE = ", mean_squared_error(precos, precos_preditos))
13     print("MAE = ", mean_absolute_error(precos, precos_preditos))
14     print("SCORE = ", r2_score(precos, precos_preditos))
```

**Código Fonte 2.13. Implementação das funções: *treinameno_p()* (função responsável pelo treinamento e predição) e print_metricas() (responsável por imprimir as métricas utilizadas).**

Com o código 2.14 podemos visualizar graficamente a distribuição dos valores preditos através da regressão polinomial. Alterando o valor do *degree* de 2 até 4, podemos perceber uma variação no resultado predito.

```
1
2  # Definindo um plot
3  plt.figure()
4  # Rótulo do eixo X
5  plt.xlabel('Diâmetro em cm')
6  # Rótulo do eixo Y
7  plt.ylabel('Preco em reais')
8  # Definindo um range de 2 até 4, que representará o valor do degree
9  for i in range(2, 5):
10     # Plotando uma linha para representar os preços preditos, variando
           apenas o degree (representado pelo valor da variável i)
11     plt.plot(diametros, treinameno_p(i, diametros, precos),linewidth=3,
           label='Degree = '+str(i))
12     # Definindo os valores mínimo e máximo do gráfico (Eixos X e Y),
           neste caso de 0 a 60 para cada eixo.
13     plt.axis([0,60,0,60])
14 # Adicionando os valores do diâmetros e preços (original) ao plot .
15 plt.plot(diametros,precos,'o', label='Peço Original')
16 # Adicionar legenda
17 plt.legend()
18 # Adicionando grade
19 plt.grid()
20 plt.show()
```

**Código Fonte 2.14. Plot variando o valor da variável *degree* de 2 até 4.**

A Figura 2.8 apresenta um gráfico com os preços preditos variando a variável *degree* de 2 até 4. O melhor resultado encontrado para este problema foi aplicando o *degree* igual a 5, que será apresentado na Tabela 2.3.

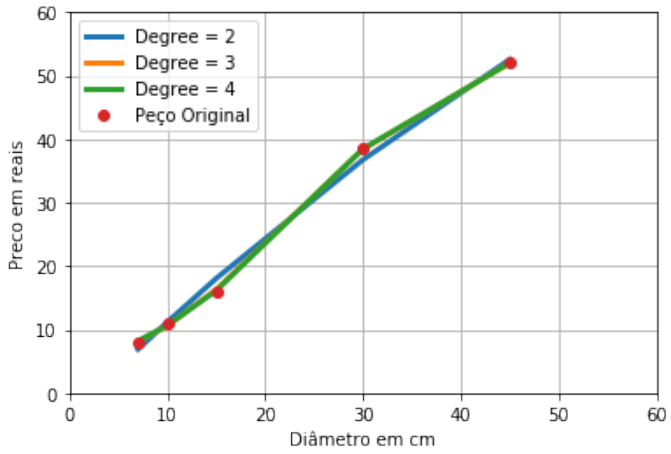



**Figura 2.8. Variação do parâmetro *degree* de 2 até 4.**

```
1  import pandas as pd
2  def print_metricas2(precos, precos_preditos):
3      # Retornando métricas (MSE, MAE e SCORE)
```

```
    return mean_squared_error(precos, precos_preditos),
        mean_absolute_error(precos, precos_preditos), r2_score(precos,
        precos_preditos)

def gerar_metricas_degree(quantidade):
    # Define uma lista vazia
    lista = []
    # Laço de repetição de 1 até quantidade informada
    for i in range(1, quantidade):
        # Executa a função print_métricas2, alterando o valor do degree
        mse, mae, sc = print_metricas2(precos, treinameno_p(i,
            diametros, precos))
        # Adiciona o valor do degree (i), mse, mae e score
        lista.append(([i, mse, mae, sc]))
    # Definindo valores com 4 casas decimais após a virgula
    lista_2 = np.round(lista,decimals=4)
    # Retorna um DataFrame (pandas)
    return pd.DataFrame(lista_2, columns=['DEGREE','MSE', 'MAE', 'SCORE
        '])

# Executando 10 vezes, com o valor do degree de 1 até 10
gerar_metricas_degree(10)
```

**Código Fonte 2.15. Implementação do código para alternar o valor do *degree*.**

A Tabela 2.3 apresenta as métricas relacionadas a cada execução referente a variação da variável *degree*, variando de 1 até 9. Observamos que o *degree* com valor 5 se destacou em todos os quesitos (MSE, MAE e *SCORE*) em relação aos demais.

**Tabela 2.3. Tabela.**

| DEGREE | MSE | MAE | SCORE |
|---|---|---|---|
| 1 | 2,7423 | 1,311 | 0,9907 |
| 2 | 1,7647 | 1,1284 | 0,994 |
| 3 | 0,0565 | 0,1884 | 0,9998 |
| 4 | 0,0373 | 0,1445 | 0,9999 |
| **5** | **0,0003** | **0,0143** | **1,0** |
| 6 | 0,2015 | 0,3148 | 0,9993 |
| 7 | 0,3412 | 0,3991 | 0,9988 |
| 8 | 0,4792 | 0,4635 | 0,9984 |
| 9 | 0,5962 | 0,5113 | 0,998 |

## Referências

[Ayodele 2010] Ayodele, T. O. (2010). Types of machine learning algorithms. In *New advances in machine learning*. IntechOpen.

[Batista 2003] Batista (2003). Préprocessamento de dados em aprendizado de máquina supervisionado. *Educational and psychological measurement*.

[Bell 2014] Bell, J. (2014). *Machine learning: handson for developers and technical professionals*. John Wiley & Sons.

[Bishop 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Bloedorn 1998] Bloedorn, M. (1998). data driven construtive induction.

[Cárdenas-Montes 2006] Cárdenas-Montes, M. (2006). Sobreajuste - overfitting. Technical report.

[da Silva 2017] da Silva, J. C. (2017). Aprendizagem de máquina é divertido! https://medium.com/machina-sapiens/aprendizagem-de-m

[de Souza 2019a] de Souza, E. G. (2019a). Entendendo o que é matriz de confusão com python. https://medium.com/data-hackers/entendendo-o-que-

[de Souza 2019b] de Souza, E. G. (2019b). Implementando regressão linear simples em python. https://medium.com/data-hackers/implementando-regress

[Freedman 2009] Freedman, D. A. (2009). *Statistical models: theory and practice*. cambridge university press.

[Ho 1995] Ho, T. K. (1995). Random decision forest.

[Izbicki and dos Santos ] Izbicki, R. and dos Santos, T. M. Machine learning sob a ótica estatística.

[Kramer 2005] Kramer, M. (2005). R 2 statistics for mixed models.

[Lee 2000] Lee, M. (2000). Applying knowledge-driven constructive induction: Some experimental results. *Educational and psychological measurement*.

[Luai Al Shalabi 2006] Luai Al Shalabi, Z. S. (2006). Normalization as a preprocessing engine for data mining and the approach of preference matrix.

[Minicozzi 1976] Minicozzi, E. (1976). Some natural properties of strong-identification in inductive inference. *Theoretical Computer Science*, 2(3):345–360.

[Minsky 1954] Minsky, M. (1954). Theory of neural-analog reinforcement systems and its application to the brain-model problem. (9438).

[Nagelkerke et al. 1991] Nagelkerke, N. J. et al. (1991). A note on a general definition of the coefficient of determination. *Biometrika*, 78(3):691–692.

[Nicolas 2015] Nicolas, P. R. (2015). *Scala for machine learning*. Packt Publishing Ltd.

[Rao et al. 1973] Rao, C. R., Rao, C. R., Statistiker, M., Rao, C. R., and Rao, C. R. (1973). *Linear statistical inference and its applications*, volume 2. Wiley New York.

[Schittkowski 2013] Schittkowski, K. (2013). *Numerical data fitting in dynamical systems: a practical introduction with applications and software*, volume 77. Springer Science & Business Media.

[Suthaharan 2016] Suthaharan, S. (2016). Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, 36:1–12.

[Unicamp 2019] Unicamp (2019). Medição de dados experimentais, incerteza e propagação de erro. Disponível em: http://www.fem.unicamp.br/ instmed/Incerteza$_{O}ld.htm.Acessadoem$ : 25 – 10 – 2019.

[Wang and Lu 2018] Wang, W. and Lu, Y. (2018). Analysis of the mean absolute error (mae) and the root mean square error (rmse) in assessing rounding model. In *IOP Conference Series: Materials Science and Engineering*, volume 324, page 012049. IOP Publishing.

[Wang and Bovik 2009] Wang, Z. and Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117.

# Capítulo 3

# Deep Learning From Scratch Without Framework

Pablo de Abreu Vieira, Romuere Rodrigues Veloso Silva, Thiago José Barbosa Lima, Nonato Rodrigues de Sales Carvalho, Rafael Luz Araújo

***Abstract***

*Deep Learning (DL) has close ties with Machine Learning and Artificial Intelligence, bringing major breakthroughs to society. DL is a powerful tool in helping us understand and control the environment around us by allowing multi-layer computer models to learn data representation at various levels of abstraction. Unlike other techniques DL makes no prior assumptions about data, nonlinear functions are modeled to be trained to generalize when receiving data never seen before accurately. DL, has been achieving great success in processing images, videos, audios, texts, new drugs among others. This chapter aims to make the reader create his 1st Neural Network from scratch, without framework, just using math, Python and its Numpy library to help create and manipulate arrays.*

***Resumo***

*Deep Learning (DL) tem laços estreitos com Machine Learning e Inteligência Artificial, trazendo grandes avanços à sociedade. A DL é uma poderosa ferramenta em nos ajudar a entender e controlar o ambiente ao nosso redor permitindo que modelos computacionais, compostos por várias camadas, aprendam representação de dados em vários níveis de abstrações. Diferente de outras técnicas a DL não faz suposições prévias sobre dados, funções não lineares são modeladas para serem treinadas para generalizar ao receberem dados nunca antes vistos com precisão. DL, vem alcançando grande sucesso no processamento de imagens, vídeos, áudios, textos, novas drogas dentre outros. Este capitulo visa fazer o leitor criar sua 1ª Rede Neural do zero, sem framework, apenas utilizando matemática, Python e sua biblioteca Numpy de auxilio a criação e manipulação de matrizes.*

## 3.1. Introdução

Embora o conhecimento sobre *Deep Learning* já exista a décadas sua popularização se deu apenas no início do século XXI, quando esta área de conhecimento passou a receber

maior atenção tanto acadêmica quanto profissional. As técnicas de *Deep Learning*, nos últimos anos, impactaram a sociedade numa ampla gama de trabalhos e pesquisas de processamento de dados como imagens, vídeos, áudios, textos, descoberta de novas drogas dentre outros.

*Deep Learning* tem laços estreitos com *Machine Learning* e Inteligência Artificial. A *Multilayer Perceptron* (MLP) forma um tipo de Rede Neural Artificial (RNA). Ao contrário de outras técnicas estatísticas, as MLP's não fazem suposições prévias sobre a distribuição dos dados podendo modelar funções não lineares e serem treinadas para generalizar com precisão quando recebem dados nunca antes vistos. A MLP consiste num sistema de neurônios interconectados simples dispostos em mais de uma camada, o que torna o aprendizado profundo [M. W. Gardner 1998].

Devido o grande apelo, tanto acadêmico quanto profissional, sobre o entendimento e a utilização de *Deep Learning* em sistemas de aprendizado de padrões, surgiu a proposta deste capitulo para que leitores possam não só entender os aspectos teóricos e técnicos como também desenvolver suas próprias *Deep Learnings* com a utilização da **matemática** e a linguagem de programação **Python** aliada a sua biblioteca **Numpy** que auxilia na criação e manipulação de matrizes possibilitando assim uma boa curva de aprendizado.

O presente capitulo inseri os aspectos teóricos de *Deep Learning* como também práticos. Aqui é descrito a base matemática, miníma, para o entendimento de como são feitos o processamento de dados e suas predições. É desenvolvido algoritmos na linguagem de programação *Python*, auxiliado de sua biblioteca *Numpy*, para assim poder aplicar o aprendizado teórico em praticas como: criação de classe, atributos, pesos, camadas, erros, interações, *backpropagation* e predições. O leitor poderá desenvolver sua primeira Rede Neural do zero, sem a utilização de *frameworks* prontos.

## 3.2. Deep Learning

Desde o início dos tempos, a humanidade vem construindo ferramentas e métodos cada vez mais sofisticadas para controlar e entender o ambiente ao nosso redor. Podemos citar como exemplos: a "domesticação"do fogo, armas, agricultura, roda, escrita, papel, avião, computadores e a internet dentre outros. O aprendizado profundo é o capitulo atual nessa longa história de inovações! Isso é extraordinário, uma vez que as *Deep Learnings* usam um algoritmo inspirado em nossos cérebros as Redes Neurais.

O cérebro humano é um "computador"complexo, não linear e paralelo, o que nos possibilita a visão, sensoriamento, lembrar, falar. O cérebro pode desenvolver regras através de experiências sensoriais como o calor, frio, doce e amargo. Consegue trabalhar bem sem regras explicitas percebendo e agindo. A informação é processada no cérebro através de uma rede de bilhões de neurônios. Cada neurônio recebe sinais (sinapses) de um grande número de outros neurônios para combina essas entradas, e envia o resultado dessa combinação a um grande número de outros neurônios. Existe um padrão de conexão entre os neurônios que parece incorporar o conhecimento requerido para realizar o processamento da informação [Dale Purves 2010].

Uma pergunta pode ser feita: O que é *Deep Learning*? DL é uma poderosa ferramenta em entender e controlar o ambiente ao nosso redor, o que permite modelos compu-

tacionais, compostos por várias camadas, aprendam a representação de dados em vários níveis de abstração. DL é utilizado para resolver tarefas praticas como: visão computacional (classificar imagens); processamento de linguagem natural (processamento de textos); reconhecimento automático de fala (processamento de áudio); biomedicina - farmácia (desenvolvimento de novas drogas) e etc. Resumindo: DL é um conjunto de métodos que utiliza principalmente Redes Neurais Artificiais para modelar funções não lineares, afim de treinarem para generalizar com precisão dados nunca antes vistos.

Mas o que é uma Rede Neural? São sistemas paralelos distribuídos composto por uma unidade de processamento simples (nodos) que calculam determinadas funções matemáticas (normalmente não lineares). Tais unidades são dispostas em uma ou mais camadas e são interligadas por um grande número de conexões, geralmente chamadas de unidades racionais. Na maioria dos modelos estas conexões estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por um neurônio da rede [Haykin 2007].

O neurônio biológico é composto da seguintes estruturas: **Dendritos** (entrada) que é responsável por receber os impulsos nervosos (sinapses) de outros neurônios e levar até o corpo celular; **Corpo celular** (processamento) é responsável por processar a informação e gerar novos impulsos nervosos; **Axônios** (saída) é responsável por transmitir os impulsos gerados no corpo celular para outros neurônios.A Figura 3.1 trás uma representação de um neurônio biológico [Dale Purves 2010].

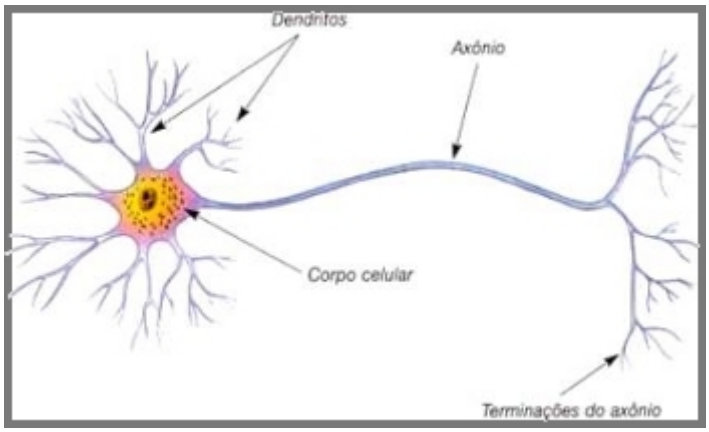



**Figura 3.1. Representação do neurônio biológico, [Hermano 2009]**

O neurônio artificial é composto das seguintes estruturas: Por um conjunto de sinapses, elos de conexão, caracterizados por pesos. Um sinal *Xj* na entrada da sinapse *j* conectado ao neurônio *K* é multiplicado pelo peso *Wk*; Um somador para somar os sinais de entrada, ponderados pelas sinapses do neurônio; Uma função de ativação para restringir a amplitude de saída de um neurônio, de onde saem as predições; O modelo neuronal também inclui uma *bias* (*Bk*) que tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação [Rosenblatt 1958].

Toda a infraestrutura, que simula uma neurônio biológico de forma artificial pode ser visualizada na Figura 3.2. Nela podemos observar que o neurônio artificial busca imitar tanto os componentes do neurônio biológico como também a forma de seu comporta-

mento, é por isso que se pode afirmar que as redes neurais tentam imitar o funcionamento do cérebro humano.

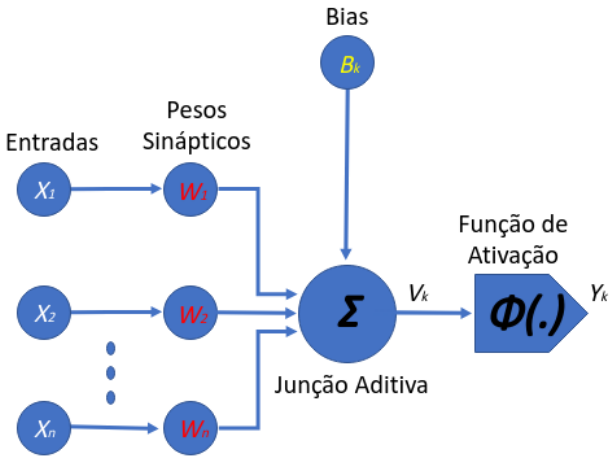



**Figura 3.2. Representação do neurônio artificial.**

### 3.2.1. Datas Importantes

Algumas contribuições são pontos chave para o aprendizado profundo se tornar tão eficiente na tarefa de nos ajudar a entender e controlar o ambiente ao nosso redor, através de computadores. A tabela 3.1 nos mostra as datas que ocorreram aquelas contribuições que são consideradas como importantes para o aprendizado profundo, ela também indica seus responsáveis.

**Tabela 3.1. Datas importantes.**

| ANO | AUTOR | EVENTO |
|---|---|---|
| 1941 | ALAN TURING | "INTELIGÊNCIA MECÂNICA" |
| 1943 | McCULLOCH & PITTS | "MODELO DE NEURÔNIO ARTIFICIAL" |
| 1947 | ALAN TURING | "INTELIGÊNCIA COMPUTACIONAL" |
| 1949 | HEBB | "REGRA DE APRENDIZAGEM DE UM NEURÔNIO" |
| 1950 | ALAN TURING | "INTELIGÊNCIA ARTIFICIAL" |
| 1954 | MINSKY | "MODELO-CEREBRAL BASE PARA MLP" |
| 1958 | ROSENBLATT | "PERCEPTRON" |
| 1969 | MINSKY & PAPERT | "XOR EM PERCEPTRON" |
| 1986 | RUMELHART et al | "RETROPROPAGAÇÃO DE ERRO" |
| 1986 | RUMELHART et al | "PERCEPTRON DE MULTIPLAS CAMADAS" |
| 1998 | YAN LECUN | "REDE NEURAL CONVOLUCIONAL" |

Foi Allan Turing quem inciou as pesquisas relacionadas a inteligência computacional [Turing 1992]. O primeiro neurônio artificial foi modelado por McCulloch e Pitts [Warren S. McCulloch 1943], antes mesmo do termo "inteligência artificial"ter sido cunhado por Turing em 1950. Em 1958 Rosenblatt criou a *Perceptron*, seu impacto foi muito grande na acadêmia [Rosenblatt 1958]. Esse período é conhecido como o período

do "*entusiasmo*". No entanto logo Minsky e Papert perceberam que ela só servia para resolver problemas linearmente separáveis, não conseguindo resolver problemas simples como *xor*, inciando um período nomeado de "*inverno*" [Marvin Minsky 1988].

Após a decepção das *Perceptron* em não conseguir resolver o problema do *xor* pesquisas relacionadas ao aprendizado de máquinas, aprendizado profundo e inteligência artificial quase que desapareceram, no entanto alguns entusiastas não desistiram destas áreas de atuação. Foi no período do "*inverno*"que Rumelhart utilizou o *Backpropagation* em redes neurais [Rumelhart 1986], e neste mesmo ano de 1986 Rumelhart e McClelland, através de estudos realizados em 1954 por Minsky [Minsky 1954], conseguiram implementar a *Multilayer Perceptron*, essa técnica conseguia resolver problemas não linearmente separáveis como o *xor* [Rumelhart et al. 1986], porém naquele período o poder computacional comprometia sua utilização, fator que contribuía para o "*inverno*".

Em 1998, Yan Lecun resouveu o problema das imagens *Minist* através de uma nova técnica as redes neurais convolucionais [Yann LeCun 1998], o que fez com que o "inverno"acabasse surgindo o "renascimento"em aprendizado profundo, algo que só foi possível ao poderio computacional e o largo acesso a internet que ocorreu desse período em diante, até os dias atuais.

### 3.3. Perceptron

Ao fim dos anos 50 Roseblatt, prosseguindo as ideias de McCulloch e Pitts, havia criado uma genuína rede de múltiplos neurônios do tipo descriminadores lineares e a nomeou de *Perceptron*. Sua unidade básica são os neurônios. A *Perceptron* consiste de uma única camada de *n* neurônios com pesos sinápticos e *bias* ajustáveis. Se os padrões dos dados de entrada forem linearmente separáveis, a convergência é garantida, existe a capacidade de se encontrar um conjunto de pesos que poderá classificar corretamente os dados. As *Perceptron* são redes neurais simples com capacidade de aprendizado [Warren S. McCulloch 1943] [Marvin Minsky 1988].

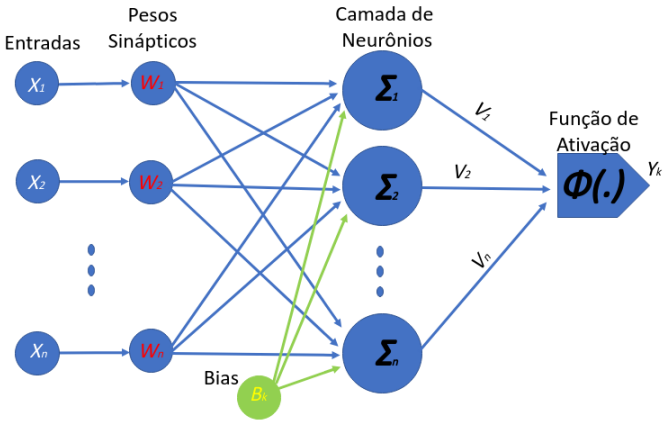



**Figura 3.3. Representação da perceptron, onde temos as entradas conectadas aos neurônios pelos pesos, incluindo o bias e a função de ativação.**

### 3.3.1. Predição com Múltiplas Entradas

Para entendermos melhor como uma rede neural faz uma predição, ilustraremos uma predição simples com uma *Perceptron*.

Uma *Perceptron* pode nos ajudar a descobrir se um time de futebol irá ganhar uma partida ou não, pra isso temos que passar algumas informações para a *Perceptron*, estas informações são 3 entradas, estas entradas serão multiplicadas por 3 pesos para ao final ser feita uma soma ponderada para obter a predição. Assumiremos duas suposições (nossas classes) *1* representa que o time ganhará e *0* que o time irá perder.

O time em questão possui três atributos que serão utilizados na predição: chutes ao gol na ultima partida = 8.5; média da posse de bola da ultima partida = 65% ; e gols marcados na ultima partida = 1. os pesos para cada entrada que as interligam a função de predição são: 0,1; 0,2; e 0.

De posse dos dados de entrada e os pesos a predição pode ser feita. A Figura 3.4 nos ilustra todos os passos para uma predição feita por uma *Perceptron* recebendo múltiplas entradas. As entradas são os dados fornecidos anteriormente, essas entradas são multiplicadas com os pesos: 8,50 * 0,1 = 85; 0,65 * 0,2 = 1,3; e 1.2 * 0 = 0;

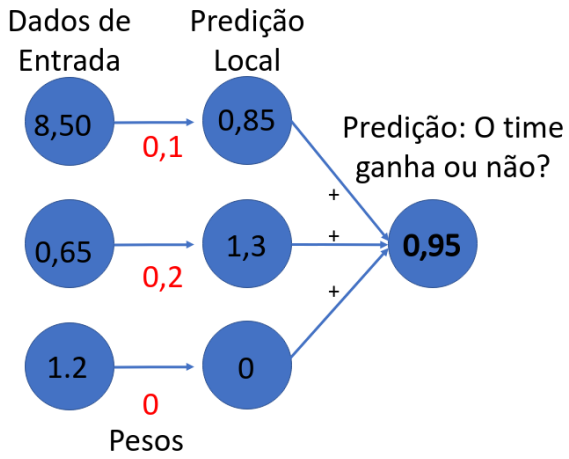



**Figura 3.4. Perceptron fazendo uma predição com múltiplas entradas.**

A primeira etapa consiste em fazer a multiplicação dos dados de entrada pelos pesos: 8,5 * 0,1; 0,65 * 1,3; 1,2 * 0. Com isso é obtido a predição local de cada entrada. Na segunda etapa é feito a soma ponderada, que consiste em somar todos os resultados das predições locais: 0,85 + 1,3 + 0 = 0,95, para obter a predição. Com o resultado de 0,95 notamos que o resultado está mais próximo de 1 que 0, ou seja, nossa *Perceptron* previu que estes conjuntos de dados relacionados com os conjuntos de pesos obtiveram resultado muito próximo a classe 1. A classe 1 se refere a vitória do time, já a classe 0 refere a derrota, com a predição podemos dizer que nosso querido time tem grandes chances de ganhar sua partida.

## 3.4. Matemática - Trabalhando com Matrizes

Os cálculos que foram utilizados na seção anterior, na nossa primeira predição, são muito simples. Eles continuaram simples, porém as redes neurais trabalham com matrizes (sistemas lineares). É necessário entender como são feitos os cálculos com matrizes, só assim será possível entender o funcionamento dos códigos de implementados neste capitulo. Já que os dados de entrada de uma rede neural são matrizes como também seus pesos e neurônios.

O produto de uma matriz por outra não é determinado por meio do produto dos seus respectivos elementos. Assim, o produto das matrizes $A = (aij)m\ x\ p$ e $B = (bij)p\ x\ n$ é a matriz $C = (cij)\ m\ x\ n$, em que cada elemento $cij$ é obtido por meio da soma dos produtos dos elementos correspondentes da i-ésima linha de A pelos elementos da j-ésima coluna B [Boldrini 1986].

Multiplicaremos as matrizes $A$ e $B$ para entendermos melhor como se obtém cada elemento $cij$. A Figura 3.5 ilustra as matrizes $A$ e $B$, as demais etapas da multiplicação podem ser conferidas a seguir:

$C(cij) = A(aij) * B(bij)$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ e } B = \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix}$$

**Figura 3.5. Matriz A e matriz B.**

O primeiro passo para se fazer a multiplicação entre duas matrizes é multiplicar a primeira linha de $A$ pela primeira coluna de $B$, onde cada elemento da primeira linha de $A$ multiplica o elemento correspondente a sua posição na primeira coluna de $B$, a Figura 3.6 nos mostra esse procedimento destacando as linhas e colunas desta etapa da multiplicação de vermelho, como também seu resultado.

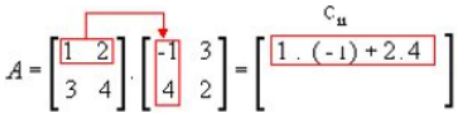



**Figura 3.6. 1ª linha de A vezes a 1ª coluna de B.**

A figura 3.7 nos mostra que a segunda etapa é multiplicar cada elemento da primeira linha de $A$ pelos elementos respectivos da segunda coluna de $B$.

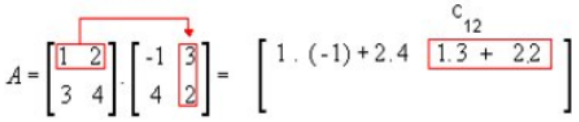



**Figura 3.7. 1ª linha de A vezes a 2ª coluna de B.**

A figura 3.8 mostra que a terceira etapa é multiplicar com a segunda linha de *A* a primeira coluna de *B*, calda elemento de *A* multiplica os elementos respectivos de *B*.

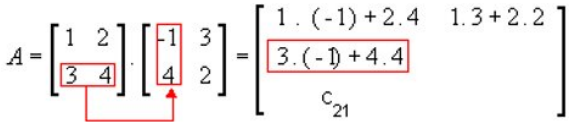



**Figura 3.8. 2ª linha de A vezes a 1ª coluna de B.**

A última etapa de nossa multiplicação é ilustrada na Figura 3.9, onde multiplicamos a segunda linha de *A* pela segunda coluna de *B*, elemento a elemento.

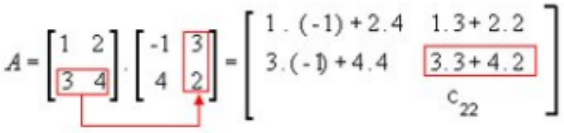



**Figura 3.9. 2ª linha de A vezes a 2ª coluna de B.**

O resultado da Multiplicação da matriz *A* por *B* esta na Figura 3.10.

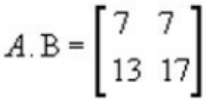



**Figura 3.10. Resultado da multiplicação da Matriz A por B.**

Uma importante observação deve ser feita sobre a multiplicação de matrizes: A propriedade *cumulativa* da multiplicação não se aplica a multiplicação de matrizes, ou seja ( *A x B*) é diferente de (*B* x *A*), como pode ser observado na Figura 3.11.

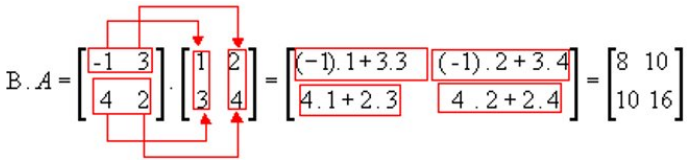



**Figura 3.11. (A x B) é diferente de (B x A).**

Para que ocorra a multiplicação de *A x B* é necessário que o número de linhas em *A* seja igual ao número de colunas em *B*, o que permite matrizes de tamanhos diferentes possam fazer multiplicações. É interessante ressaltar que as demais operações soma, subtração e divisão também seguem o mesmo padrão apresentado aqui na multiplicação.

## 3.5. Predição

Como é definido valores dos pesos para que uma rede preveja com precisão? Essa é talvez a pergunta mais importante relacionado ao aprendizado profundo.

A forma mais simples para se fazer o ajuste correto dos pesos é **comparar** para **aprender**. Comparando podemos ver o quanto a previsão "errou", medir o erro é algo fundamental para o aprendizado profundo. Neste capitulo focaremos no **Erro Médio Quadrático**. Quando se brinca de "quente"e "frio", você não entende o porque errou na brincadeira, em que direção errou ou qual movimento deve ser feito para corrigir o erro. O que fazer com o erro é função da próxima etapa, aprender. É no aprendizado que cada peso é informado como ele pode mudar para reduzir o erro. Aprender está relacionado a atribuição de erros ou a arte de descobrir como cada peso pode influenciar nos erros.

Porque o erro é elevado ao quadrado? Podemos pensar num atirador tentando acertar o centro do alvo na Figura 3.12: Quando uma bala atinge 4 linhas acima do centro, qual foi o erro? Quando uma bala atinge 4 linhas para baixo do centro, qual foi o erro? Nas duas situações os erros foram de 4 linhas. É por isso que é utilizado o erro quadrático, assim forçamos o erro ser sempre positivo, já que um erro negativo não faria sentido.

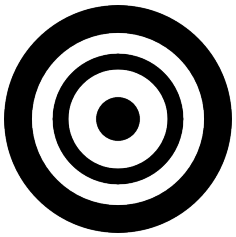


**Figura 3.12. Alvo.**

O erro médio quadrático é um método utilizado para estimar a quantidade de erro que é a diferença entre os valores predito - pela rede - a que classe realmente pertence. Este erro será sempre positivo. Quanto menor o valor, mais próximo de *0*, menor será a taxa de erro. A definição do erro médio quadrático pode variar de acordo com a necessidade do estimador. A Equação 1 pode ser usada para obter o erro quadrático médio de um conjunto ou *n* dados [Dennis Wackerly 2008].

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2 \tag{1}$$

Onde $\frac{1}{n}\sum_{j=1}^{n}$ é executado a operação de média e $(y_j - \hat{y}_j)^2$ se calcula o erro quadrado de cada dado.

### 3.5.1. Frio e Quente - Um Aprendizado Simples

Ajustar os pesos para "cima"e para "baixo"é como uma rede "aprende", isso é o que pode fazer o erro tender a *0*. Se temos uma entrada com valor de 8,5, e o peso 0,1, e a classe que estamos tentando predizer é a *1*, logo nosso erro médio quadrático será 0,0225; Com a mesma entrada e classe para predição e um peso de valor 0,11 obtemos um erro quadrático de 0.0042; Se nosso próximo peso for de 0,09 nosso resultado será de 0,0552. Numa rápida analise percebemos que o peso que traz a melhor predição é o de valor 0,11, pois ele proporciona um erro mais aproximo de *0*. A Figura 3.13 ilustra bem esse processo.

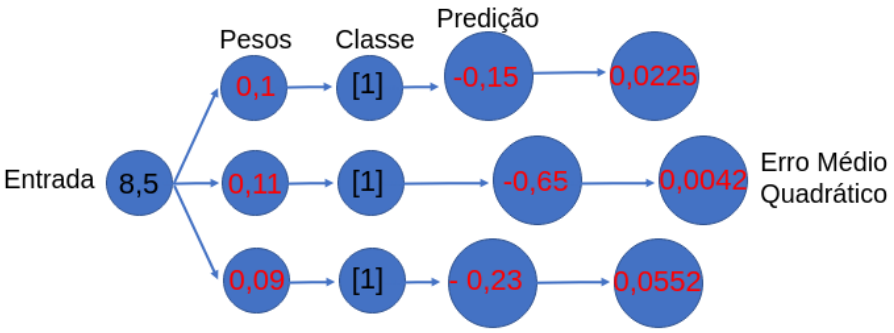



**Figura 3.13. Predição "Quente"e "Frio".**

Essa metodologia apesar de muito simples contém alguns problemas: Ineficiência, são necessárias múltiplas atualizações de pesos até se encontrar o peso mais apropriado; Em alguns casos pode ser impossível fazer uma predição exata, mesmo com milhares de tentativas; O maior problema é que mesmo que se saiba qual direção seguir, não será possível saber qual o tamanho da "passada"a se tomar.

### 3.5.2. Descida do Gradiente

Para se calcular a direção correta e o montante que o peso deve mudar é utilizado a **Descida do Gradiente**. Esse nome deve trazer "calafrios", porém vamos desmitificar a descida do gradiente. O objetivo da descida do gradiente é encontrar um mínimo dado alguma função. Em funções convexas é possível encontrar o mínimo global, porém em funções do "mundo real"não convexas pode haver $n$ mínimos e em sua maioria mínimos locais, em sua minoria sendo os tão procurados mínimos globais [Alexey Izmailov 2012].

Imagine a competição "cabo de guerra", nesta atividade existe uma corda que pode ligar por suas extremidades dois competidores, quando um competidor puxa a corda sua ação causa uma reação no outro competidor que é puxado, e o contrario também é verdade. O que a descida do gradiente faz é entender qual a relação entre duas variáveis, quando eu aumento uma o que ocorre na outra? No aprendizado profundo é definido uma relação exata entre erro e peso, a descida do gradiente ajuda encontrar qual peso melhor se adequada para o erro tender a 0.

A descida do gradiente é um método numérico usado em otimização. Para encontrar um mínimo (local) de uma função usa-se um esquema iterativo, onde em cada passo se toma a direção (negativa/contraria) do gradiente, que corresponde à direção de declive máximo. Pode ser encarado como o método seguido por um curso da água, na sua descida pela força da gravidade, a Formula 2 representa o inicio da descida do gradiente, a Formula 3 se verifica a condição de descida do método, já a Formula 4 é fincada a interação [David G. Luenberger 2008].

Começando com um vetor inicial $x_0$, visando alcançar um ponto de mínimo de **F**, consideramos a sucessão definida por $x_0, x_1, x_2, \ldots$ onde a pesquisa linear é dada pela direção de descida $b_n$.

$$x_{n+1} = x_n + w_n d_n. \tag{2}$$

No caso do método do gradiente a condição de descida verifica-se tomando

$$d_n = -\bigtriangledown F(x_n). \tag{3}$$

ficando a interação definida por

$$x_{n+1} = x_n - w_n \bigtriangledown F(x_n). \tag{4}$$

Se o valor de um peso for alterado em qualquer direção. Havendo uma relação exata entre erro e peso, é possível ponderá e calcular o quanto isso irá alterar o erro. E se existir a necessidade de mover o erro em uma direção específica? Isso poderia ser feito? A resposta é sim.

Para se conseguir uma direção especifica 1º se calcula o *delta* que é dado pela predição da rede menos o valor real (classe), a Figura 3.14 (a) ilustra essa etapa; A próxima etapa é calcular o delta do peso que é obtido com a multiplicação do delta pela entrada, ilustrado na Figura 3.14 (b).

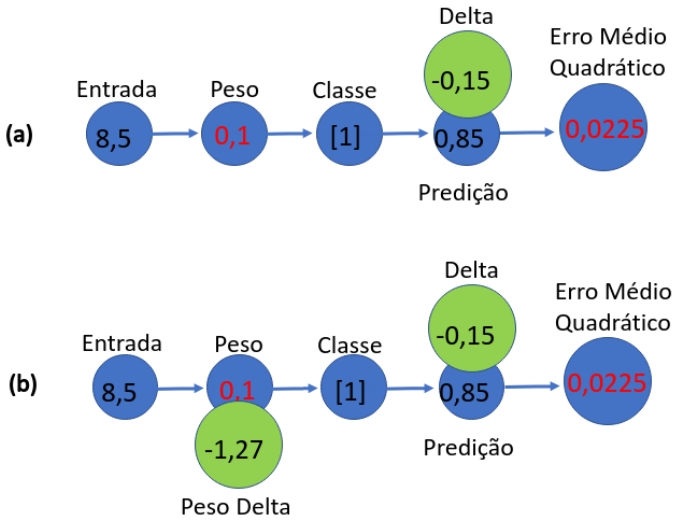



**Figura 3.14. Descida do Gradiente, (a) delta, (b) peso delta.**

### 3.5.3. Taxa de Aprendizagem (Alpha)

Você deve ter sentido falta da atualização do peso na última seção, o que ocorreu foi que descobrimos o *delta* e o *delta do peso*, porém faltou a atualização do peso. Até agora a descida do gradiente só nos forneceu qual direção seguirmos, e quanto devemos alterar o peso? A variável que pode nos fornecer quanto alterar o peso é o *Alpha* que é conhecido como a taxa de aprendizado. O *alpha* indica quanto nossa rede vai aprender a cada interação, quantos "passos"a rede deve "andar"no rumo do erro *0* [Chollet 2017].

Para a etapa de atualização do peso, o peso *delta* deve ser multiplicado pelo *alpha*, e em seguida decrementar o peso original pelo valor dessa multiplicação para assim o peso poder ser finalmente atualizado, a Figura 3.15 ilustra isso.

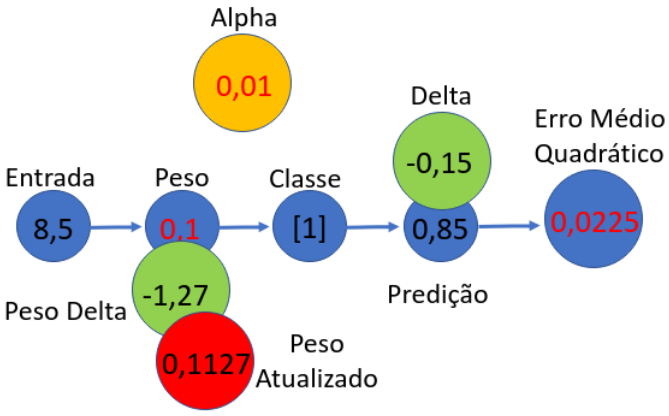



**Figura 3.15. Atualização do Peso.**

A escolha de um tamanho para o *alpha* deve ser feita com cuidado, já que ela pode definir se a rede terá ou não um bom aprendizado. A taxa de aprendizado alta faz com que a rede aprenda rápido, porém pode fazer ela passar do ponto ótimo. A Taxa de aprendizado baixo faz com que a rede aprenda lentamente, porém aumenta a probabilidade dela estacionar no ponto ótimo.

### 3.5.4. Ponto Ótimo

Utilizando derivadas de variáveis é possível levar o erro a *0*. É possível calcular a inclinação (*derivada/delta*) da linha para qualquer valor de peso. Essa inclinação pode ser usada para descobrir qual a direção que reduz o erro. Com esse cálculo é possível saber o quão longe se está do ponto ideal [Trask 2019].

Como na competição de "cabo de guerra"para qualquer função pode ser escolhido duas variáveis para se entender seu relacionamento. Para qualquer função duas variáveis podem ser plotadas em um plano cartesiano. Para qualquer função é possível escolher duas variáveis e calcular o quanto uma muda quando você altera outra. Para qualquer função é possível entender como alterar uma variável para poder mover a outra variável em uma direção. A Figura 3.16 ilustra o resultado - onde o ponto estaciona no plano cartesiano - entre a relação de duas variáveis (peso e erro).

Ao examinar a derivada em uma tabela de referência, você entenderá o que a derivada representa. O relacionamento entre duas variáveis em função, o quanto muda quando a outra muda? Podemos nomear isso de sensibilidade entre duas varia veis. Sensibilidade positiva é quando as variáveis se movem juntas; Sensibilidade negativa é quando as variáveis se movem em direções opostas; E sensibilidade zero é quando uma permanece fixa independente do que seja feita com a outra [Trask 2019].

A diferença entre o erro e a derivada do erro e peso é que o erro é a medida do quanto se perdeu, a derivada define a relação entre cada peso e o quanto errou. A inclinação de uma linha ou curva sempre aponta na direção oposta ao ponto mais baixo da linha ou curva. Se existe uma inclinação negativa, aumente o peso para encontrar o erro mínimo (ponto ótimo). Então o peso é movido na direção oposta da derivada para encontrar o menor peso. A rede neural aprende. Este é o método do gradiente descendente

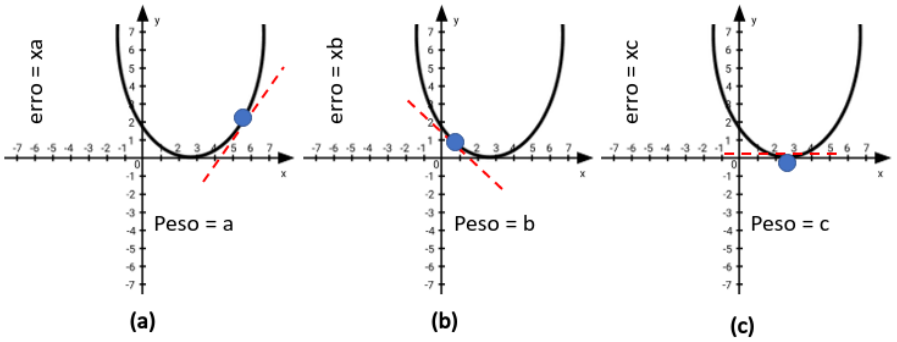



**Figura 3.16. Ponto ótimo, ponto resultante a); ponto resultante b); ponto resultante c).**

[David G. Luenberger 2008].

## 3.6. Função de Ativação ReLU

As funções de ativação é que são responsáveis por adicionarem o componente da não linearidade as redes neurais, isso possibilita com que elas possam aprender além das relações lineares entre as variáveis. Porém a utilização de funções de ativações fazem com as redes neurais tenham uma ativação não linear, o que faz com que a rede deixe de ser convexa, o que torna a otimização mais complexa [Xavier Glorot 2010].

A função de ativação *ReLU* possui relações com a restrição de não-negatividade, sendo a mais adotada por facilitar o processo de treinamento das redes neurais. A *ReLU* é simplesmente a função identidade para valores positivos. Por exemplo: as funções *sigmoidais* comprimem a saída para um intervalo curto, enquanto *ReLU* cancela todos os valores negativos, sendo linear para os positivos. Ao calcularmos a derivada da *ReLU*, seu gradiente terá sempre uma direção não-nula, enquanto no caso das *sigmoidais*, para valores longe da origem podemos ter gradiente nulo [Moacir A. Ponti 2017], a Figura 3.17 ilustra a função *ReLU*.

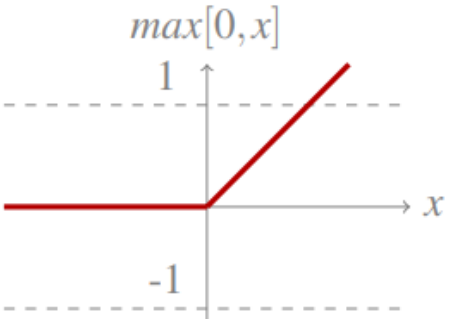



**Figura 3.17. Ilustração da função de ativação ReLU [Moacir A. Ponti 2017].**

## 3.7. Backpropagation - Retro-propagação

No período do "inverno"para pesquisas e desenvolvimentos na área de *deep learning* as pesquisas e trabalhos diminuíram porém não cessaram. Foi justamente nesta época que Rumelhart desenvolveu o método de retro-propagação de erros [David E. Rumelhart 1986]. A retro-propagação é um algoritmo utilizado no treinamento de redes neurais. A retro-propagação calcula o gradiente da função de perda em relação aos pesos para um exemplo de entrada e saída. É isto que possibilita a descida do gradiente ocorrer. O calculo é feito para cada neurônio uma camada por vez, começando da camada de saída até chegar na camada de entrada [Ian Goodfellow 2016], a Figura 3.18 ilustra o *backpropagataion*.

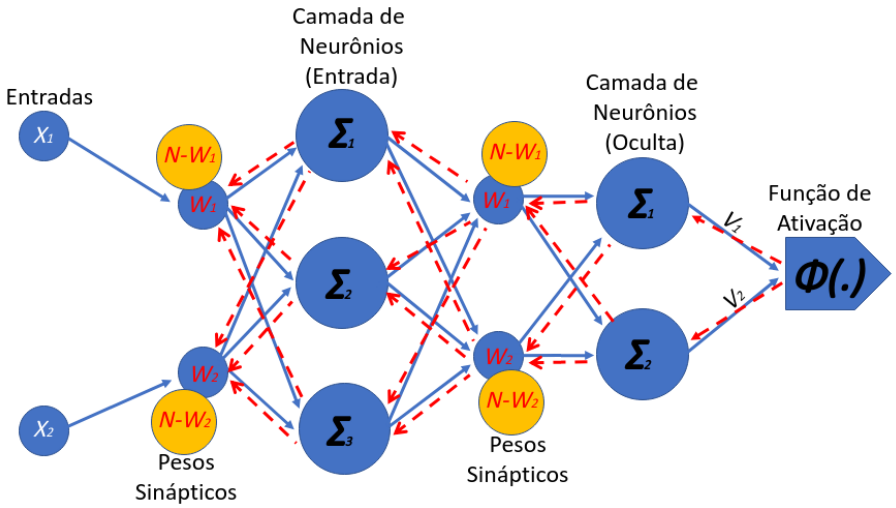



**Figura 3.18. Backpropagation, as setas tracejadas de vermelho representam a retro-propagação dos erros.**

A correlação dos erros pode ser feita utilizando outros métodos, em particular o cálculo da segunda derivada. Essa técnica consiste em corrigir os erros de acordo com a importância dos elementos que contribuíram precisamente para a realização desses erros. No caso de redes neurais, pesos sinápticos que contribuem para um erro significativo serão modificados mais significativamente do que os pesos que causaram um erro marginal. Esse princípio baseia os métodos do algoritmo de gradiente, que são utilizados em redes neurais multicamadas, como *Perceptron* multicamadas (MLP) [Ben Kröse 1996].

## 3.8. Multilayer Perceptron (MLP)

A *Multilayer Perceptron* forma um tipo de Rede Neural Artificial (RNA). Ao contrário de outras técnicas estatísticas, as MLP não fazem suposições prévias sobre a distribuição dos dados podendo modelar funções não lineares e serem treinadas para generalizar com precisão quando recebem dados nunca antes vistos. A MLP consiste num sistema de neurônios interconectados de forma simples, podendo ser constituída de uma ou mais camadas ocultas [M. W. Gardner 1998].

Utilizando um único neurônio não se pode fazer muitas coisas, porém ao combinarmos mais de uma neurônio em uma única camada as possibilidades aumentam o que torna possível fazer predições precisas em conjuntos de dados linearmente separáveis, a um custo computacional não tão alto.

Quando lidamos com problemas não linearmente separáveis uma *Perceptron* não é capaz de convergir. Para resolver este tipo de problema é necessário combinar os neurônios artificiais em *n* camadas. Quando uma rede tem mais de uma camada nomeamos a primeira - que recebe os dados - de camada de entrada e as demais de camadas ocultas. Ao adicionarmos camadas a mais em nossa rede estamos adicionando profundidade, o aprendizado se torna profundo, o que a torna capaz de aprender a fazer relações cada vez mais complexas, A Figura 3.19 nos mostra como as camadas de neurônios podem ser arranjadas em uma rede neural de aprendizado profundo.

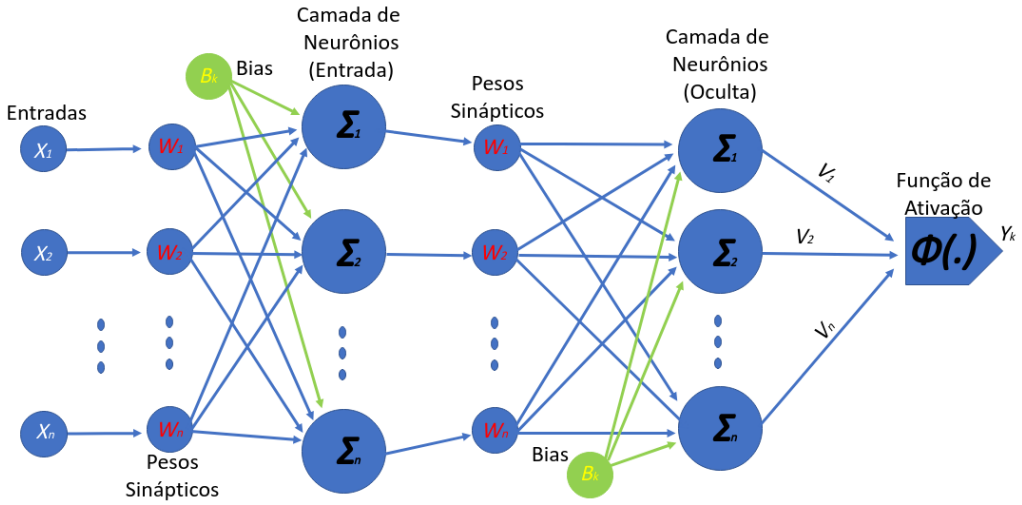



**Figura 3.19. MLP.**

Apesar das redes de múltiplas camadas de neurônios serem bastante poderosas e precisas em entender problemas complexos, elas geram um grande custo computacional. A cada camada e neurônio adicionados na rede o aprendizado pode melhorar, porém a cada aumento desses elementos o custo computacional também cresce. Quando é adicionado um neurônio, automaticamente um uma nova conexão com um peso deve ser feita. Quando se adiciona uma nova camada de neurônios, automaticamente devem ser criados novos pesos e novas conexões. cada conexão feita em uma rede representa um processo matemático que tem seu custo computacional.

Atualmente a boa disposições dos elementos, e suas quantidades, de uma rede neural é objeto de grandes estudos. Já existem redes neurais que tem o papel de tornar a utilização destes recursos mais otimizados, um bom exemplo é a *Morphnet* desenvolvida pelos laboratórios do Google. A *Morphnet* recebe uma rede neural existente como entrada e produz uma nova rede que é menor, mais rápida e produz um melhor desempenho sob medida para um determinado problema [Ariel Gordon 2018].

## 3.9. Codificação em Python

Nesta seção serão introduziremos os códigos necessários para se construir uma rede neural de aprendizado profundo. A linguagem que utilizaremos será o Python[1] que permite implementação rápida com uma curva de aprendizado muito boa. Para auxiliar na criação

[1]https://www.python.org/

e manipulação de matrizes e vetores utilizaremos o Numpy[2] que é um pacote fundamental para computação cientifica.

### 3.9.1. Codificação - Perceptron

A primeira codificação implementada é a da *Perceptron*, já que exige uma codificação mais simples, o que é essencial para o aprendizado. O que será feito? Será criado uma *Perceptron* para analisar alguns dados de uma entrada e dizer se eles pertencem ou não a uma classe. Criaremos o dado de entrada como uma matriz de atributos; as classe que o elemento pertence; criaremos o peso; taxa de aprendizado (*alpha*); faremos uma predição, Código 3.1 ilustra essa construção:

```
# Perceptron

# 1 - Importando a biblioteca Numpy:
imoprt numpy as np

# 2 - Etapa é criar os atributos de entrada, uma matriz com 5 vetores
    que possuem 3 dados de atributos cada:
entradas = np.array( [ [ 1, 0, 1 ],
                       [ 0, 1, 1 ],
                       [ 0, 0, 1 ],
                       [ 1, 1, 1 ],
                       [ 0, 1, 1 ],
                       [ 1, 0, 1 ] ] )

# 3 - Criando um vetor das classes que cada elemento da matriz de
    entrada pertence:
classes = np.array( [ [ 0 ],
                      [ 1 ],
                      [ 0 ],
                      [ 1 ],
                      [ 1 ],
                      [ 0 ] ] )

# 4 - Criando o vetor de pesos:
pesos = np.array([0.5, 0.48, -0.7])

# 5 - Pegando um único elemento de nossa matriz de atributos:
entrada = entradas[0]

# 6 - Fazendo uma predição:
# Utilizando a função .dot() do Numpy para fazer a soma ponderada.
predição = entrada.dot(pesos)

# 7 - Imprimindo a saída:
print(predicao)

Out: -0.19999999999999996
```

**Código Fonte 3.1. Descrição Perceptron.**

[2]https://numpy.org/

Como podemos conferir o resultado no Código 3.1, a predição foi de: -0.19, o objeto que foi feito a predição pertencia a classe 0. O leitor pode escolher os demais objetos e fazer outras predições.

A próxima codificação será a de uma *Perceptron* que irá interagir com todo os elementos da matriz de atributos, teremos desta vez 5 objetos diferentes entrando, os quais tentaremos predizer de qual classe cada um pertence, o Código 3.2 ilustra todo o processo de construção da *Perceptron*. A codificação mostra: o cálculo médio do erro; a taxa de aprendizado; calcularemos o *delta* para atualizarmos os pesos:

```
# Perceptron que prediz com múltiplas entradas:

# Importando a biblioteca Numpy:
imoprt numpy as np

# Setando os pesos:
pesos = np.array([0.5, 0.48, -0.7])

# Setando o Alpha:
alpha = 0.1

# Setando as entradas:
entradas =  np.array( [ [ 1, 0, 1 ],
                        [ 0, 1, 1 ],
                        [ 0, 0, 1 ],
                        [ 1, 1, 1 ],
                        [ 0, 1, 1 ],
                        [ 1, 0, 1 ] ] )

# Setando as classes respectivas a cada objeto:
classes = np.array( [ [ 0 ],
                      [ 1 ],
                      [ 0 ],
                      [ 1 ],
                      [ 1 ],
                      [ 0 ] ] )

# Fazendo uma interação de 10 épocas:
for iteracao in range(10):

    # Criando uma variável de apoio para armazenar o erro:
    # ao final de todas as predições.
    erro_para_todos_objetos = 0

    print('============================')
    # Imprimindo as épocas:
    print("Época: ", iteracao+1)

    # Fazendo a interação com todos os 5 objetos:
    for índice in range(len(classes)):

        # Acessando os objetos por ordem:
        entrada = entradas[indice]
```

```
44
45          # Pegando a classe a qual o objeto pertence:
46          classe_do_objeto = classes[indice]
47
48          # Fazendo a predição, usando a soma ponderada:
49          predicao = entrada.dot(pesos)
50
51          # Calculando o erro médio quadrático:
52          # Esse cálculo serve para ter uma noção do nível de acerto.
53          # Pra gerar o erro,
54          # Pega o que a rede 'chutou' e o que realmente é.
55          erro = (classe_do_objeto - predição) ** 2
56
57          # Atualizando a variável erro_para_todos_objetos:
58          erro_para_todos_objetos += erro
59
60          # Calculando o delta:
61          # O delta serve para apontar qual direção devemos ir.
62          delta = predicao - classe_do_objeto
63
64          # Atualizando os pesos:
65          # O Alpha indica qual a taxa de aprendizado, ou seja o quanto
                os pesos, devem ser alterados.
66          pesos = pesos - (alpha * (entrada * delta))
67
68          # Imprimindo as saídas:
69          print("Objeto:"+str(indice))
70          print("Predição:" + str(predicao))
71          print('******************************')
72      print('==============================')
73
74  Out: ==============================
75      Época:  1
76      Objeto:0
77      Predição:-0.19999999999999996
78      Objeto:1
79      Predição:-0.19999999999999996
80      Objeto:2
81      Predição:-0.5599999999999999
82      Objeto:3
83      Predição:0.6160000000000001
84      Objeto:4
85      Predição:0.17279999999999995
86      Objeto:5
87      Predição:0.17552
88      ==============================
89      .
90      .
91      .
92      ==============================
93      Época:  10
94      Objeto:0
95      Predição:0.04958243192113776
96      Objeto:1
97      Predição:0.9056327614440267
```

```
98      Objeto:2
99      Predição:-0.13768337501215525
100     Objeto:3
101     Predição:1.1250605910610996
102     Objeto:4
103     Predição:0.9132624284442169
104     Objeto:5
105     Predição:0.04653264583708144
106     ==============================
```

**Código Fonte 3.2. Descrição Perceptron com múltiplas entradas.**

Aos compararmos os resultados da primeira época com os da decima época, no Código 3.2, podemos ver que a *Perceptron* teve uma melhora substancial em seus resultados. Algumas alterações nos valores dos pesos, da taxa de aprendizado e quantidade e interações podem ser feitos para ver o que acontece, podendo trazer resultados diferentes.

### 3.9.2. Codificação - MLP

Neste momento foi obtido todo o ferramental necessário para a implementação da primeira rede neural com profundidade, ou seja, será implementado uma MLP. Com a implementação da MLP será conquistado o aprendizado profundo. O Código 3.3 ilustra uma MLP com: a descida do gradiente completa em lote atualizando pesos de um exemplo a cada vez a cada interação; uma função de ativação *RelU* que irá converter todos os números negativos em 0 (a *Perceptron* gerava previsões negativas); Criação de pesos de forma automatizada; utilização do *Backpropagation*:

```
1  # Aprendizado em profundidade (MLP):
2
3  # Importando a biblioteca Numpy:
4  imoprt numpy as np
5
6  # Utilizando a fincão seed():
7  # garantindo que os pesos iniciados aleatórios serão sempre os mesmos:
8  np.random.seed(1)
9
10 # Função de Ativação ReLU:
11 def funca_ativacao_relu(x):
12     # Se for menor que 0, retorne 0:
13     return (x > 0) * x
14
15 # Função de ativação Relu_Derivada:
16 def relu2deriv(output):
17     # Se a entrada for maior que 0, retorne 1:
18     return output>0
19
20 # Definindo Alpha:
21     alpha = 0.01
22
23 # Definindo a matriz de atributos:
24 entradas = np.array( [ [ 1, 0, 1 ],
25                        [ 0, 1, 1 ],
```

```
                        [ 0, 0, 1 ],
                        [ 1, 1, 1 ],
                        [ 0, 1, 1 ],
                        [ 1, 0, 1 ] ] )

# Definindo as classes de cada objeto da matriz de atributos:
classes = np.array([[0,1,0,1,1,0]]).T

# Pegando dados para definir o tamanho da camada de entrada:
# Quantidade de pesos e neurônios.
linhas,_ = entradas.shape
tamanho_oculto = linhas

# Setando o tamanho da camada oculta
# Quantidade de pesos e neurônios.
tamanho_oculto = 17

# Definindo o conjunto de pesos que ira conectar os dados de entrada a
    camada de entrada:
pesos_0_1 = 2 * np.random.random((colunas, tamanho_oculto)) - 1
# Definindo o conjunto de pesos que ira conectar a camada oculta a
    camada de saída:
pesos_1_2 = 2 * np.random.random((tamanho_oculto, 1)) - 1

# fazendo 100 interações:
for interacao in range(100):

    # Variável auxiliar que recebera os erros na camada 2:
    camada_2_error = 0

    # Fazendo a interação com todos os 5 objetos:
    for i in range(len(entradas)):

        # Passando os objetos para a camada de entrada:
        camada_0 = entradas[i:i+1]

        # Fazendo a soma ponderada dos objetos da camada_0 com os
            pesos_0_1:
        # Se os valores forem negativos a Relu transforma em 0.
        camada_1 = funca_ativacao_relu(np.dot(camada_0,pesos_0_1))

        # Recebendo a saida da camada oculta:
        # E aplicando a soma ponderada com os pesos_1_2:
        camada_2 = np.dot(camada_1,pesos_1_2)

        # Calculando o erro:
        camada_2_error += np.sum((camada_2 - classes[i:i+1]) ** 2)

        # Inicio do Backpropagation

        # Calculando o Delta para a camada 2:
        camada_2_delta = (classes[i:i+1] - camada_2)

        # Calculando o Delta para a camada 1:
        camada_1_delta = camada_2_delta.dot(pesos_1_2.T)*
```

```
            ativacao_relu_deriva(camada_1)
78
79          # Atualizando os pesos:
80          pesos_1_2 += alpha * camada_1.T.dot(camada_2_delta)
81          pesos_0_1 += alpha * camada_0.T.dot(camada_1_delta)
82
83
84      # Imprimindo a taxa de erro, para poder analisar se a rede está
            aprendendo:
85      if(interacao % 10 == 9): # Imprimindo somente a cada 10 épocas:
86          print("Época: ", interacao, "Error:" + str(camada_2_error))
87
88  Out:
89
90      Época:  0 Erro: 8.066440572813145
91      Época:  10 Erro: 0.06626703469166717
92      Época:  20 Erro: 0.021142459122806476
93      Época:  30 Erro: 0.011938369213078183
94      Época:  40 Erro: 0.007603673179313106
95      Época:  50 Erro: 0.0050333460316583254
96      Época:  60 Erro: 0.003376825513183101
97      Época:  70 Erro: 0.0022799737570888693
98      Época:  80 Erro: 0.0015465073842685039
99      Época:  90 Erro: 0.0010536221372978703
```

**Código Fonte 3.3. Descrição MLP.**

Analisando os resultados no Código 3.3, percebemos que a rede inicia com uma taxa de erro muito alta. Isso é normal já que devido a quantidade de pesos e neurônios dificilmente ela irá encontrar uma boa relação entre pesos e atributos de forma randômica. A partir da décima época notamos que a rede já encontrou uma boa relação entre seus pesos e os dados de entrada já que o resultado começa tender a *0*, o que nos mostra o quão poderoso é a descida do gradiente aliada ao *backpropagation*. Na nonagésima época o erro já está consideravelmente baixo.

Irei propor um desafio ao leitor. Tente encontrar um conjunto de: taxa de aprendizado, quantidade de neurônios, pesos e épocas de interação que possa obter melhor resultado com esse conjunto de dados. Mão na massa, afinal a melhor forma de aprender é praticando.

## 3.10. Deep Learning Considerações Finais

Técnicas de *Deep Learning* tem conseguido atingir resultados excepcionais. No entanto, é interessante ressaltar que existem limitações no uso de redes neurais em sua forma de aprendizado que é utilizado na matriz de entradas. Esse aprendizado é dado por um conjunto de pesos sinápticos, que são atualizados durante o treinamento para tentar converge com os dados de entrada. Todo o processo que envolve o aprendizado profundo tem um grande custo computacional envolvido, principalmente a parte do treinamento. Para a rede conseguir realizar o treinamento, as transformações que os dados de entrada sofrem devem ser passiveis de serem derivados [Chollet 2017].

Ao contrário da forma como o cérebro humano pode aprender. As redes neurais

necessitam de uma grande quantidade de dados para poder conseguir fazer a generalização e poder aprender alguns conceitos simples. Se a rede não for alimentada com grandes quantidades de dados pode ocorrer o *overfit* que é quando se tem a falsa sensação de aprendizado generalizado, mas na verdade a rede passa a entender muito bem a correlação dos dados de entrada utilizados no treinamento, mas se mostra ineficaz em prever novos resultados para dados nunca antes vistos [Cárdenas-Montes 2006].

O aprendizado profundo quando se trata de aprendizado de máquinas e problemas de classificação é o estado-da-arte. Desde o "renascimento", o aprendizado profundo nos mostrou o mais alto nível de desempenho em uma variedade de áreas como: reconhecimento automático de fala, visão computacional, bioinformática dentre outros. Para ajudar resolver esses problemas, várias estruturas de aprendizado profundo foram desenvolvidas como as redes neurais profundas, redes neurais convolucionais, redes neurais recorrentes. "*Deep Learning*"tornou-se um termo, ou uma nova forma de redes neurais.

Os conceitos teóricos e matemáticos desenvolvidos neste capitulo podem ser utilizados como os primeiros passos rumo ao entendimento de *Deep Learning*. Os códigos deste capitulo servem para introduzir no leitor o entendimento básico de como o aprendizado profundo realmente trabalha a nível de codificação, porém eles não foram planejados para ter um bom desempenho computacional, a codificação desenvolvida procura ser o mais didático possível.

A codificação foi baseada em estudos com os livros: [Haykin 2007], [Chollet 2017] e [Trask 2019], como também em pesquisas realizadas na *Internet* (especificamente em fóruns especializados sobre *Deep Learning*) dentre outros.

Para conhecimento mais aprofundado sobre "*Deep Learning*"é recomendado [Marvin Minsky 1988], [Ben Kröse 1996], [M. W. Gardner 1998], [Haykin 2007], [Ian Goodfellow 2016], [Chollet 2017], [Trask 2019].

## Referências

[Alexey Izmailov 2012] Alexey Izmailov, M. S. (2012). *Otimização*. IMPA.

[Ariel Gordon 2018] Ariel Gordon, Elad Elban, O. N. B. C. H. W. T.-J. Y. E. C. (2018). Morphnet: Fast simple resource-constrained structure learning of deep networks. Technical report, Google Research, Google brain, Energy-efficient multimedia systems group, MIT., Georgia Instititute of Technology.

[Ben Kröse 1996] Ben Kröse, P. v. d. S. (1996). *An Introduction Neural Networks*. The University of Amsterd.

[Boldrini 1986] Boldrini, L. J. (1986). *Álgebra Linear*. Harbra São Paulo Brasil.

[Chollet 2017] Chollet, F. (2017). *Deep Learning with Python*. Manning.

[Cárdenas-Montes 2006] Cárdenas-Montes, M. (2006). Sobreajuste - overfitting. Technical report.

[Dale Purves 2010] Dale Purves, George J. Agostinho, D. F. W. C. H. A.-S. L. A. J. O. M. L. E. W. (2010). *Neurociências*. ARTMED EDITORA SA.

[David E. Rumelhart 1986] David E. Rumelhart, Geoffrey E. Hinton, R. J. W. (1986). Learning representations by back-propagation erros. Technical report, Institute for Cognitive Science, Unervisty of California.

[David G. Luenberger 2008] David G. Luenberger, Y. Y. (2008). *Linear and Nonlinear Programming. International Series in Operations Research Management Science*. Springer.

[Dennis Wackerly 2008] Dennis Wackerly, William Mendenhall, R. S. (2008). *Estadistica Matematica con Aplicaciones*. Cengage Learning Editores S.A. de C.V.

[Haykin 2007] Haykin, S. (2007). *Redes Neurais: Princípios e Prática*. Artmed.

[Hermano 2009] Hermano (2009). Perceptron em perl. Technical report.

[Ian Goodfellow 2016] Ian Goodfellow, Yoshua Bengio, A. C. (2016). *Deep Learning*.

[M. W. Gardner 1998] M. W. Gardner, S. R. D. (1998). Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences. Technical report, Atmospheric Environment.

[Marvin Minsky 1988] Marvin Minsky, S. P. (1988). *Perceptrons - Expanded Edition*. The MIT Press.

[Minsky 1954] Minsky, M. (1954). Theory of neural-analog reinforcement systems and its application to the brain-model problem. Technical report, Princenton University.

[Moacir A. Ponti 2017] Moacir A. Ponti, G. B. P. d. C. (2017). *Como funciona o Deep Learning*. Tópicos em Gerenciamento de Dados e Informações, SBC, 1ª Edição.

[Rosenblatt 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Technical report.

[Rumelhart et al. 1986] Rumelhart, E., D., Mcclelland, J., and L., J. (1986). *Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations*.

[Rumelhart 1986] Rumelhart, David E.; Hinton, G. E. W. R. J. (1986). Learning representations by back-propagating errors. Technical report.

[Trask 2019] Trask, A. W. (2019). *Grokking Deep Learning*. Manning Shelter Island.

[Turing 1992] Turing, A. (1992). *Mechanical Intelligence, Volume 1 (Collected Works of A.M. Turing)*.

[Warren S. McCulloch 1943] Warren S. McCulloch, W. H. P. (1943). A logical calculus of th ideas immanent in nervous activity. Technical report, Atmospheric Environment.

[Xavier Glorot 2010] Xavier Glorot, Y. B. (2010). Understanding the difficulty of training deep feedforward neural networks. Technical report.

[Yann LeCun 1998] Yann LeCun, Léon Botton, Y. B. P. H. (1998). Gradient-based learning applied to document recognition. Technical report, ATT labs-Research.

# Capítulo 4

# Técnicas de Extração de Atributos Aplicadas ao Processamento de Imagens

Patrick Ryan Sales dos Santos, Vitória de Carvalho Brito e Antonio Oseas de Carvalho Filho

***Abstract***

*The feature extraction step is fundamental in Digital Image Processing, since at this stage that image properties can be represented using mathematical models to serve as input for pattern recognition and classification analysis. Given the context, knowledge of methods capable of characterizing the images is essential, since this phase directly influences the correct classification of the data. Therefore, this chapter explains about three relevant techniques in the extraction step: the Gray Levels Co-Occurrence Matrix, the Taxonomic Indexes and extraction by applying Transfer Learning using a Convolutional Neural Network architecture. Besides the theoretical approach, the chapter presents tutorials in python for the application of the mentioned techniques.*

***Resumo***

*A etapa de extração de características é fundamental no Processamento de Digital de Imagens, pois é nesta etapa que as propriedades da imagem podem ser representadas através de modelos matemáticos para servirem de entrada para a análise de reconhecimento e classificação de padrões. Diante do contexto, é imprescindível o conhecimento de métodos capazes de caracterizar as imagens, visto que esta fase influencia diretamente na classificação correta dos dados. Portanto, este capítulo explana sobre três técnicas relevantes na etapa de extração: a Matriz de Co-Ocorrência de Níveis de Cinza, os Índices Taxonômicos e a extração por meio da aplicação de Transfer Learning utilizando uma arquitetura de Rede Neural Convolucional. Além da abordagem teória, o capítulo apresenta tutoriais em python para a aplicação das técnicas citadas.*

## 4.1. Introdução

O Processamento Digital de Imagens (PDI) é definido como sendo o conjunto de técnicas computacionais que manipulam uma imagem de entrada em uma saída, sendo que,

na maioria dos casos, a saída é outra imagem digital. Dessa maneira, é possível, além de tratar aspectos visuais e estruturais, fornecer outros elementos que possibilitem a interpretação visual e computacional da imagem [Carvalho Filho et al. 2016]. Segundo [Gonzalez et al. 2002], um sistema clássico de PDI pode ser agrupado em cinco etapas: aquisição, pré-processamento, segmentação, extração de características e classificação. A Figura 4.1 ilustra esse processo.

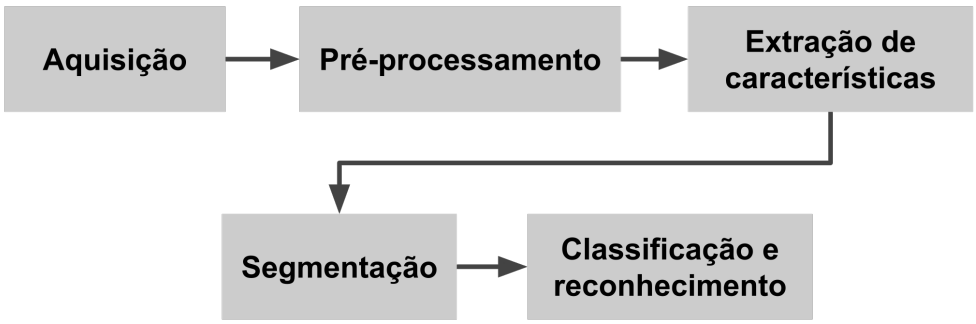



Figura 4.1: Etapas do Processamento Digital de Imagens.

Na primeira etapa, tem-se a aquisição de imagens, onde as mesmas são capturadas e representadas de forma computacional para serem interpretadas pela etapa seguinte. No pré-processamento, são realizados procedimentos capazes de proporcionar um melhoramento nos aspectos visuais e estruturais da imagem. A etapa de segmentação visa obter a região de interesse da imagem, separando-a do fundo. A representação e descrição, conhecida também por extração de características, tem como objetivo representar, através de valores, uma imagem ou partes dela. Por fim, na etapa de classificação os valores obtidos pela extração de características são os insumos para que uma técnica de aprendizado de máquina possa então discernir entre possíveis padrões contidos em um grupo de imagens [Carvalho Filho et al. 2016].

A análise de imagens é uma das principais tarefas envolvidas em um sistema de classificação. Seu objetivo é obter informações suficientes para distinguir entre diferentes regiões de interesse. Normalmente, o processo de classificação é baseado nos níveis de cinza ou cores, características de forma e textura. Diversas aplicações utilizam técnicas de classificação para reconhecimento de objetos em imagens. Assim, este capítulo apresenta algumas técnicas capazes de descrever as características das imagens, de modo que esses atributos possam servir de entrada para processos de mais alto nível.

## 4.2. Extração de Características

Como visto em [Pedrini e Schwartz 2008], uma das tarefas mais complexas na análise de imagens está na definição de um conjunto de características que possam descrever de maneira concreta cada região contida em uma imagem, de modo a serem utilizados em processos de mais alto nível. Em outras palavras, a etapa de caracterização e representação da imagem consiste em uma etapa de fundamental importância no modelo clássico do processamento de imagem digital proposto por [Gonzalez et al. 2002]. Nesta etapa, as propriedades da imagem podem ser representadas através de modelos matemáticos para

servirem de entrada para a análise de reconhecimento e classificação de padrões.

Em linhas gerais, a etapa de extração de características pode ser dividida em duas categorias de análises, sendo elas por: textura e forma. Em uma análise por textura, o objetivo geral é descrever aspectos da imagem no que diz respeito a suavidade, rugosidade e regularidade [Gonzalez et al. 2002]. Já em uma análise baseada na forma da imagem, o intuito é extrair informações que mensuram sobre propriedades morfológicas da imagem.

Diante do contexto, é imprescindível o conhecimento de métodos capazes de caracterizar as imagens, visto que esta fase influencia diretamente na classificação correta dos dados. Portanto, este capítulo explana sobre três técnicas relevantes na etapa de extração: a *GLCM*, que caracteriza a textura por meio distribuição conjunta de pares de pixels vizinhos dentro de uma imagem, considerando um determinado relacionamento espacial [Haralick et al. 1973]; os Índices Taxonômicos, que descrevem a diversidade e a distinção taxonômica entre as espécies de uma comunidade [CLARKE e WARWICK 1998]; e, por fim, a *VGG16* [Simonyan e Zisserman 2014a], uma evolução da *AlexNet*, com muitos filtros de tamanho de $3\times3$.

### 4.2.1. Matriz de Co-Ocorrência de Níveis de Cinza

A Matriz de Co-Ocorrência de Níveis de Cinza (ou *GLCM*, do inglês: *Gray Level Co-occurrence Matrix*) é uma técnica utilizada dentro da área de análise de texturas, desenvolvida na década de 70 por [Haralick et al. 1973]. Baseando-se em estatísticas, são analisadas as co-ocorrências existentes entre pares de pixels, levando em consideração também a distância entre os pixels em questão, bem como a direção de encontro dos mesmos.

A *GLCM* considera a relação entre dois pixels por vez, um chamado de pixel referência e o outro de pixel vizinho. O pixel vizinho escolhido pode, por exemplo, estar a leste (direita) de cada pixel referência. Isto pode ser expresso como uma relação (1,0): 1 pixel na direção x, 0 pixels na direção y. Cada pixel dentro da imagem torna-se o pixel referência, iniciando no canto superior esquerdo e procedendo até o inferior direito. Os pixels situados na margem direita não têm vizinhos da direita, então eles não são utilizados para esta contagem [Schwartz e Pedrini 2003].

A Figura 4.2 ilustra um exemplo de uma *GLCM* gerada para uma região de imagem, onde é possível observar que a relação entre os pixels é analisada com uma distância de 1 pixel e ângulo 0. Cada vez que o par analisado ocorre na imagem, a matriz *GLCM* é incrementada na posição correspondente ao valor dos pixels. Além disso, a Figura 4.2 também apresenta as possíveis direções a serem levadas em consideração durante a análise das ocorrências.

Para descrever as texturas, [Haralick et al. 1973] propuseram 14 medidas estatísticas, calculadas a partir da *GLCM*. No entanto, segundo [Baraldi e Parmiggiani 1995], apenas seis delas são as mais relevantes: O segundo momento angular, a entropia, o contraste, a variância, a correlação e a homogeneidade, definidas através das expressões descritas nos subtópicos seguintes.

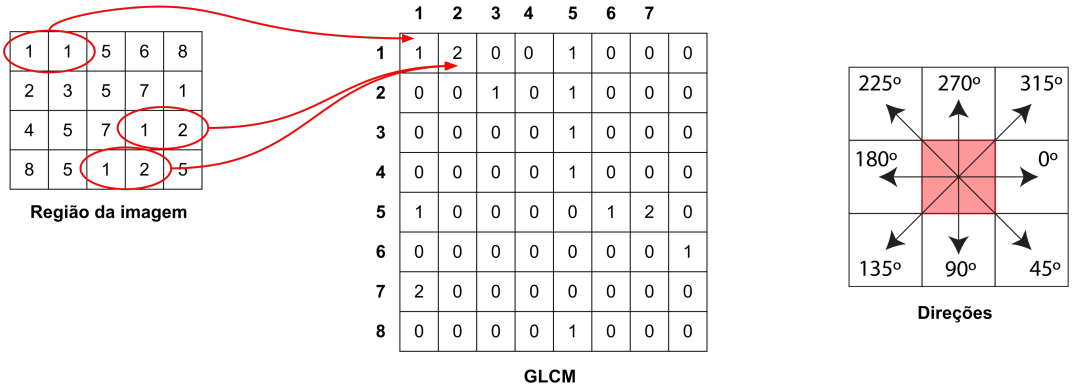



Figura 4.2: Exemplo de uma *GLCM* gerada para uma imagem e os ângulos de direção que podem ser utilizados.

#### 4.2.1.1. Segundo momento angular

O segundo momento angular, representado pela Equação 1, mede a uniformidade da textura, ou seja, a quantidade de repetições de pares de pixels. Essa medida terá valores altos quando os valores da matriz de co-ocorrência estiverem concentrados, ou seja, quando houver grande repetição na variação dos níveis de cinza. Assume valores positivos menor ou igual a 1, para a matriz normalizada.

$$sma = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} \{p(i,j)\}^2 \tag{1}$$

#### 4.2.1.2. Entropia

Este parâmetro mede, como o nome sugere, a desordem da imagem. Portanto, essa medida assume valores elevados quando a imagem possui textura não uniforme. A entropia está fortemente relacionada, porém de maneira inversa, com o segundo momento angular. A Equação 2 apresenta a fórmula para calcular esse parâmetro.

$$ent = -\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} p(i,j) log(p(i,j)) \tag{2}$$

#### 4.2.1.3. Contraste

A diferença entre o maior e o menor valor de um determinado conjunto contíguo de pixels é definida como frequência espacial. A expressão do contraste (Equação 3) indica que uma imagem com valor baixo nesse parâmetro não é necessariamente caracterizada por uma distribuição de tons de cinza estreita, mas sim por uma baixa frequência espacial, possuindo valores da matriz de co-ocorrência concentrados próximos à diagonal principal.

$$con = \sum_{n=0}^{Ng-1} n^2 \left\{ \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} p(i,j) \right\}_{|i-j|=n} \tag{3}$$

#### 4.2.1.4. Variância

A variância é uma medida da heterogeneidade da textura. O valor da variância (Equação 4) aumenta quando os valores dos tons de cinza diferem de sua média.

$$var = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} (1-\mu)^2 p(i,j) \tag{4}$$

#### 4.2.1.5. Correlação

A correlação, definida pela Equação 5, é a medida da dependência linear dos tons de cinza em uma imagem. Altos valores de correlação implicam uma relação linear entre os níveis de cinza dos pares de pixels.

$$cor = \frac{\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{5}$$

#### 4.2.1.6. Homogeneidade

A homogeneidade, definida pela Equação 6, é sensível a valores próximos à diagonal da matriz de co-ocorrência e, portanto, ao baixo contraste da textura. A Homogeneidade está inversamente relacionada com o Contraste e a Energia da imagem.

$$hom = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} \frac{1}{1+(i-j)^2} p(i,j) \tag{6}$$

### 4.2.2. Índices Taxonômicos

Filogenia é um ramo da biologia que estuda as relações evolutivas entre as espécies, sendo possível descrever a similaridade entre elas. Nas árvores filogenéticas, as espécies são representadas pelas folhas e os ancestrais comuns são representados pelos nós. A Figura 4.3 ilustra um exemplo de árvore filogenética, representando o relacionamento genético entre espécies de macacos e a espécie humana, onde é possível observar que, geneticamente, o homem e o chimpanzé são mais próximos que os demais pares existentes na árvore [Baxevanis e Ouellette 2004].

A diversidade filogenética é uma medida da diversidade de uma comunidade que incorpora as relações filogenéticas das espécies [Magurran 2013]. A combinação de abundância das espécies com a proximidade filogenética para gerar um índice de diversidade é

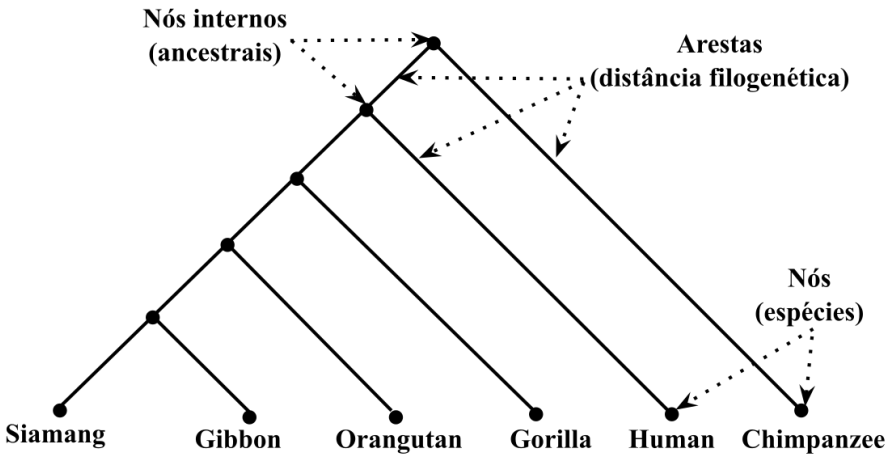



Figura 4.3: Exemplo de árvore filogenética para alguns primatas. Adaptada de [Baxevanis e Ouellette 2004].

denotada diversidade taxonômica. Para aplicar esses conceitos à caracterização das imagens, é necessário compreender a correspondência entre as definições da biologia e do Processamento de Imagens. A Tabela 4.1 descreve essa correspondência.

Tabela 4.1: Correspondência entre os termos da biologia e os do Processamento de Imagens.

| **Biologia** | **Imagem** |
|---|---|
| Comunidade | Região de Interesse (ROI) |
| Espécie | Valor do pixel do ROI |
| Indivíduos | Número de pixels em uma espécie |
| Características fenotípicas da espécie | Relação de textura entre os pixels |

A relação entre dois organismos escolhidos aleatoriamente em uma filogenia existente em uma comunidade é apresentada pelos índices de diversidade taxonômica ($\triangle$) e distinção taxonômica ($\triangle^*$) [CLARKE e WARWICK 1998]. Esses índices possuem três fatores essenciais para sua aplicação: número de espécies, número de indivíduos e a estrutura de ligação das espécies (número de arestas).

Na Figura 4.4 tem-se um exemplo de uma região da imagem onde há diversidade de espécies em relação aos pixels. Assim, é possível a identificação de padrões em imagens, utilizando os índices de diversidade taxonômica.

O valor de $\triangle$ fornece a média da distância filogenética entre os indivíduos de uma espécie [CLARKE e WARWICK 1998]. Para isso, esse índice leva em consideração a quantidade de indivíduos das espécies e o relacionamento taxonômico entre elas. A fórmula para calcular o $\triangle$ está definida pela Equação 7, onde $x_i$ (i = 1,...,S) representa a

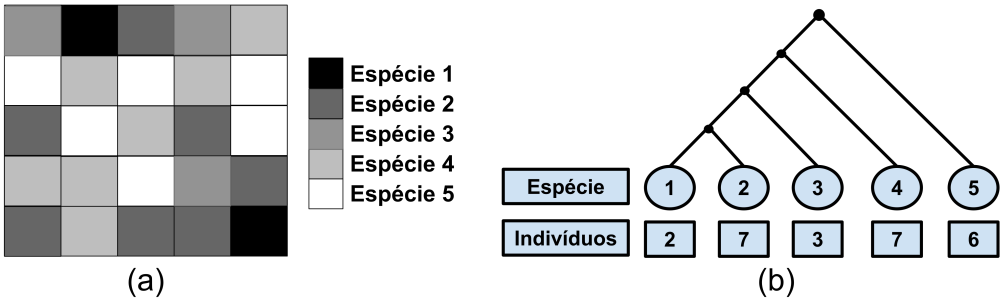



Figura 4.4: (a) Representação da região analisada; (b) árvore filogenética extraída.

abundância da $i_{th}$ espécie, $x_j$ (j = 1,...,S) representa a abundância da $j_{th}$ espécie, $S$ indica a quantidade total de espécies, $n$ denota a quantidade total de indivíduos e $w_{ij}$ expressa a distância taxonômica entre as espécies $i$ e $j$.

$$\triangle = \frac{\sum\sum_{i<j} w_{ij} x_i x_j}{[n(n-1)/2]} \tag{7}$$

Já o índice $\triangle^*$, definido pela Equação 8, expressa a média da distância taxonômica entre dois indivíduos, desde que as espécies sejam distintas [CLARKE e WARWICK 1998]. Neste cálculo, $x_i$ (i = 1,...,S) é a abundância da espécie $i_{th}$, $x_j$ (j = 1,...,S) é a abundância da espécie $j_{th}$, $S$ indica a quantidade total de espécies, $n$ denota a quantidade total de indivíduos e $w_{ij}$ expressa a distância taxonômica entre as espécies $i$ e $j$.

$$\triangle^* = \frac{\sum\sum_{i<j} w_{ij} x_i x_j}{\sum\sum_{i<j} x_i x_j} \tag{8}$$

### 4.2.3. Redes Neurais Convolucionais

O conceito de algoritmos de aprendizado profundo foi introduzido no final do século XX, permitindo que modelos computacionais compostos por várias camadas de processamento aprendam representações de dados com vários níveis de abstração. Em contraste com as abordagens tradicionais de aprendizado de máquina, as tecnologias de aprendizado profundo estão progredindo recentemente em aplicações para reconhecimento de fala, processamento de linguagem natural, recuperação de informações, visão computacional e análise de imagens [Liu et al. 2017].

As Redes Neurais Convolucionais (*CNNs*) fazem parte de uma categoria de algoritmos de aprendizado profundo que tornou-se o novo padrão na área de Visão Computacional, pois a facilidade de treinamento quando existe grandes quantidades de dados rotulares contribui para que as *CNNs* sejam utilizadas para classificar rótulos distintos. As *CNNs* são baseadas no processamento de dados visuais, capaz de aplicar filtros nesses dados, mantendo a relação de vizinhança entre os pixels da imagem ao longo do processamento da rede [LeCun et al. 1998]. Essa operação é conhecida como convolução, onde ocorre a somatória do produto ponto a ponto entre os valores de um filtro e cada posição da vizinhança do pixel de entrada.

Dentre as vantagens oferecidas pelas *CNNs*, estão a extração de atributos relevantes das imagens, através da aplicação de convoluções entre os *kernels*, e a redução da complexidade de parametrização, pois, como as unidades das camadas convolucionais não estão totalmente conectadas, como acontece nas Redes Neurais tradicionais, existem menos pesos a serem ajustados. A Figura 4.5 apresenta exemplos de reconhecimento de objetos, pessoas e animais em imagens cotidianas, por intermédio de *CNNs*.

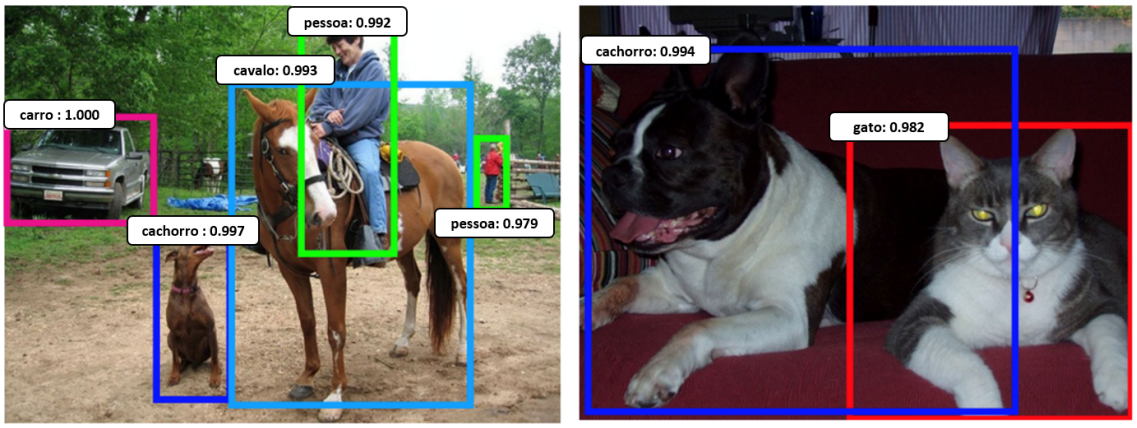



Figura 4.5: Exemplos de reconhecimento de objetos, pessoas e animais feitos por uma *CNN*. Fonte: [Araújo et al. 2017].

Como visto em [Araújo et al. 2017], existem diversas bibliotecas que oferecem implementações importantes das operações utilizadas pelas *CNNs*, algumas delas são: *Caffe* [Jia et al. 2014], *MatConvNet* [Vedaldi e Lenc 2015], *Theano* [Al-Rfou et al. 2016], *Torch* [Collobert et al. 2011] e *TensorFlow* [Abadi et al. 2015]. Todavia, a familiarização com as bibliotecas e a compreensão do perfeito funcionamento das *CNNs* com imagens são tarefas complexas que acabam exigindo alta dedicação. Diante disso, esta Seção tem por objetivo abordar os fundamentos básicos e principais componentes da *CNN*, a fim de proporcionar uma base teórica para o entendimento dos algoritmos.

#### 4.2.3.1. Camada convolucional

As camadas convolucionais são formadas por um conjunto de filtros que são iniciados com um arranjo 3D, muitas das vezes chamado de volume. Cada filtro tem uma dimensão reduzida, porém estende-se por toda a profundidade do volume de entrada. No processo de treinamento da rede, esses filtros são ajustados automaticamente para que possam extrair características relevantes, como orientação de bordas ou manchas de cores [Karpathy 2015]. A relevância é avaliada de uma forma que os resultados obtidos possam ter uma otimização em função de um conjunto de amostras previamente classificadas.

O volume de entrada é percorrido por cada um desses filtros, dando assim origem a uma estrutura conectada localmente. O processo de convolução é dado pelo somatório do produto ponto a ponto entre o valores presentes no filtro e cada posição do volume de entrada, essa operação é ilutrada na Figura 4.6. Os valores que são provenientes da operação de convolução passam por uma função de ativação, sendo a mais comum a

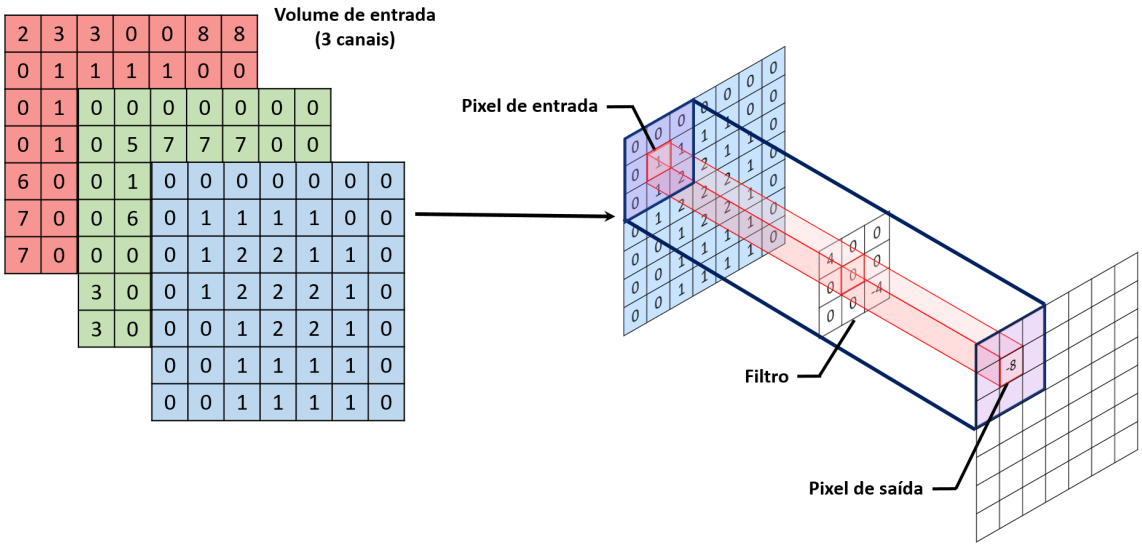



Figura 4.6: Ilustração da convolução entre um filtro 3x3 e o volume de entrada. Fonte: [Araújo et al. 2017].

função *ReLU* (Rectified Linear Units) [Krizhevsky et al. 2012]. Essa função pode ser calculada pela Equação 9

$$f(x) = max(0, x). \tag{9}$$

#### 4.2.3.2. Camada de *Polling*

O objetivo da camada de *pooling* é reduzir a dimensionalidade do volume de entrada, de modo geral esse progresso ocorre logo após uma camada convolucional, assim, diminuindo drasticamente o custo computacional da rede e o tempo de processamento, por consequência a camada de *pooling* tende a evitar o *overfitting*. No processo de *pooling*, cada valor da nova matriz é referente ao resultado de alguma métrica aplicada a uma região do mapa de convoluções, assim, um *kernel* do tamanho da matriz de *pooling* percorrerá toda a matriz de atributos resultantes da convolução para calcular os valores do *pooling*. Diversas métricas podem ser aplicadas no *pooling*, como o máximo valor da região, o mínimo ou a média, dependendo do problema abordado. Um exemplo de operação de *pooling* pode ser verificado na Figura 4.7, onde a métrica escolhida foi o maior valor da região. Essa métrica é conhecida como *max pooling* e é útil para eliminar valores desprezíveis, reduzindo a dimensão da representação dos dados e acelerando a computação necessária para as próximas camadas, além de manter a consistência da região, tornando as distorções locais mínimas [Araújo et al. 2017].

#### 4.2.3.3. Camada Totalmente Conectada

A imagem de entrada fornece para a camada convolucional e de *pooling* a possibilidade de extrair as características relevantes acerca dessa determinada imagem. O objetivo da

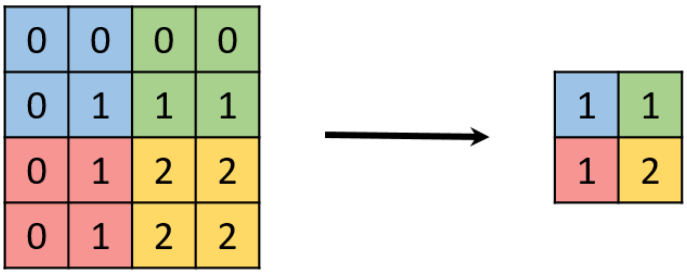



Figura 4.7: Operação de *max pooling* em uma imagem 4x4 utilizando um *kernel* 2x2. Além de reduzir o processamento para as próximas camadas, essa técnica também auxilia no tratamento de invariâncias locais. Fonte: [Araújo et al. 2017].

camada totalmente conectada é utilizar essas características para classificar a imagem em um rótulo pré-determinado, como é possível observar na Figura 4.8. As camadas responsáveis pela classificação das características extraídas das camadas convolucionais são exatamente como uma rede artificial convencional (*Multi Layer Perceptron* ou *MLP*) [Haykin et al. 2009].

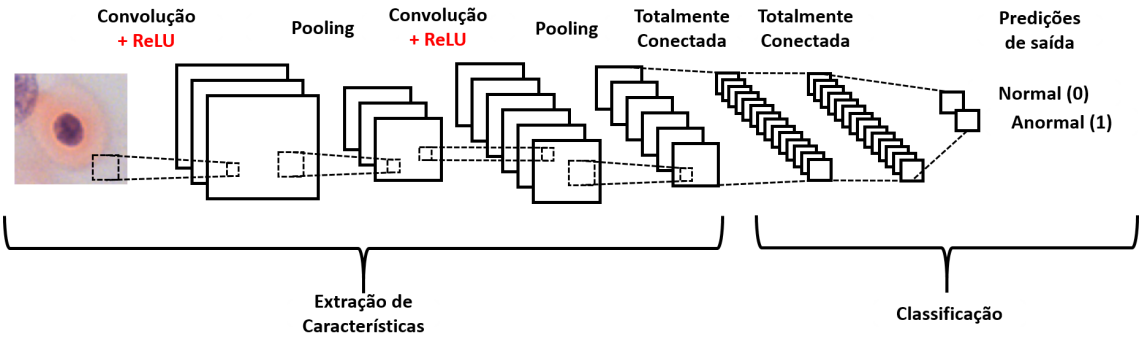



Figura 4.8: Ilustração da extração de características de uma imagem por uma CNN e sua posterior classificação. Fonte: [Araújo et al. 2017].

As camadas são formadas por neurônios, que são unidades de processamento. Na arquitetura os neurônios são totalmente conectados, pois todos os neurônios da camada anterior estão conectados a todos os neurônios da camada seguinte.

#### 4.2.3.4. VGG16

A *VGG16* é uma arquitetura de *CNN* proposta por [Simonyan e Zisserman 2014b] que alcança 92,7% de precisão no top 5 no grande conjunto de imagens *ImageNet*. Ela aprimora a *AlexNet* substituindo os filtros grandes por vários filtros 3×3 do tamanho de um núcleo, um após o outro.

A arquitetura é composta por uma pilha de camadas convolucionais, onde filtros 3×3 são aplicados para o processo de convolução, que tem o passo fixado em 1 pixel. O agrupamento espacial é realizado por cinco camadas de *pooling*, que seguem algumas das

camadas de convolução, pois não são aplicados a todas as convoluções da rede. O *max pooling* é realizado em uma janela de 2×2 pixels, com passo 2 [Simonyan e Zisserman 2014a]. A Figura 4.9 ilustra a organização das camadas na arquitetura da *VGG16*, onde é possível observar que a rede possui 16 camadas, sendo 13 convolucionais e 3 totalmente conectadas, além de 5 operações de *pooling*.

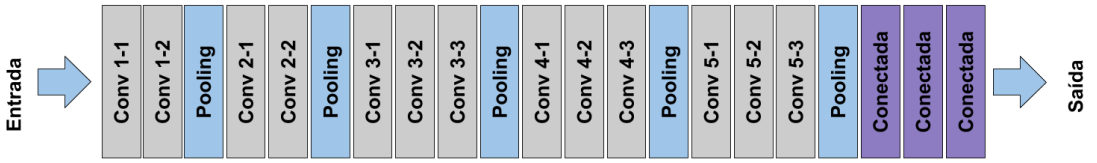



Figura 4.9: Ilustração da arquitetura da VGG16.

#### 4.2.3.5. Transferência de Aprendizado

Para o pleno treinamento de uma *CNN*, faz-se necessário uma quantidade elevada de dados e uma alta variabilidade de características. Neste contexto, a depender do caso, os dados podem ter um alto custo de aquisição ou até serem inviáveis de adquirir. Mesmo com todas as vantagens que as *CNNs* apresentam, elas são sensíveis às diferenças de iluminação, contraste ou rotação. Se na fase de treinamento essas variáveis forem desconsideradas, o classificador poderá não se adaptar a esses detalhes, reduzindo o desempenho da classificação. Dados indesejados, como partes do plano de fundo ou ruído, podem fazer com que o sistema se especialize apenas nos dados de treinamento, levando a uma generalização baixa [Godinho de Oliveira et al. 2018].

Segundo [Godinho de Oliveira et al. 2018], para que uma rede tenha um probabilidade de aprender a detectar características de baixo nível corretamente, a base tem que apresentar uma variabilidade suficientemente abrangente, como rotação dos objetos, contraste para representação de um mesmo objeto e diversas iluminações. As características de baixo nível apresentam os atributos mais básicos da imagem, como bordas, círculos, linhas e padrões de cores. Sabendo que as camadas intermediárias de uma *CNN* são comumente extratores de características que podem ser generalizadas para conjuntos de dados diferentes, uma solução comum para o treinamento é usar um conjunto de dados maior para pré-treinar a rede e, em seguida, retreinar a rede para reconhecer as classes da base de dados alvo, esse método é conhecido como transferência de aprendizado.

Esta técnica é amplamente utilizada almejando uma acurácia significativa ao treinar redes utilizando um conjunto de dados que antes seria insuficiente. Outra utilidade da transferência é adicionar novas imagens em uma rede treinada. Nesse caso, é possível retreinar apenas as camadas de reconhecimento com a nova base de dados.

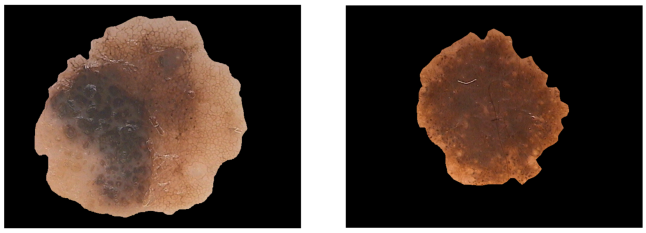
Nesse contexto, uma possível aplicação da transferência de aprendizado seria a utilização dos pesos de uma rede anteriormente treinada para uma base que tem uma quantidade de imagens consideravelmente grande, como no caso da *ImageNet* que possui mais de 1 milhão de imagens de 1000 classes, para inicializar e retreinar uma rede aplicada a outro problema [Karpathy 2015]. A transferência de aprendizado pode ser utilizada até

mesmo para a extração de características de imagens, como é o caso da demonstração prática apresentada neste capítulo.

Para utilizar a transferência de aprendizado na extração de características, basta remover a última camada totalmente conectada da rede (a camada que realiza a classificação dos dados) e utilizar a saída final da nova rede como características que descrevem a imagem de entrada. Com isso, as características obtidas podem servir de entrada para diversos classificadores, como Máquina de Vetor de Suporte (Support Vector Machine - SVM) [Cortes e Vapnik 1995], *Random Forest* (RF) [Breiman 2001] e K Vizinhos Mais Próximos (K Nearest Neighbors - KNN) [Aha e Kibler 1991].

## 4.3. Demonstração das Técnicas Apresentadas

Diversos problemas podem ser abordados com o uso de técnicas de extração de características para descrever os atributos relevantes das imagens para a classificação. Nesse capítulo, optamos por aplicar as técnicas apresentadas no problema de detecção de melanoma em imagens de lesões de pele. A base de imagens utilizada foi a PH2, que contém 200 imagens de lesões de pele, sendo 40 de melanoma e 160 de não melanoma [Mendonça et al. 2013]. A Figura 4.10 mostra exemplos das imagens de lesões disponíveis na PH2, que foram segmentadas a partir das máscaras disponibilizadas pelos especialistas na base. Os códigos 4.1, 4.2 e 4.3 apresentam a extração de características das lesões de pele com a *GLCM*, os índices taxonômicos e a *VGG16*, respectivamente. No caso da *VGG16*, os pesos aplicados na rede foram os da *ImageNet*, que possui mais de 1 milhão de imagens divididas em 1000 classes.



(a) Melanoma (b) Não melanoma

Figura 4.10: Exemplos de lesões da base PH2.

```
1
# Importando as bibliotecas
3 import glob as g
import numpy as np
5 from skimage.feature import greycomatrix, greycoprops
from skimage.io import imread
7 from skimage.color import rgb2gray

9 # Método para salvar o arquivo de características em libsvm
def geraSVMfile(rotulo, lista_feat, name_file, path_out, modo):
11     arquivo = open(path_out + name_file, modo)
    featureFile = str(rotulo) + " "
13     arquivo.write(featureFile)
    for i in range(len(lista_feat)):
```

```
        linha = str(str(i + 1) + ":" + str(lista_feat[i]) + " ")
        arquivo.write(linha)
    arquivo.write('\n')
    arquivo.close()

# Extraindo as características
def run_imagens(classe, lista, modo):
    for i in range(len(lista)):
        name = lista[i].split('/')[-1].split('.')[0].split('g')[0]

        img = imread(lista[i])
        img = rgb2gray(img)*255
        img = np.uint32(img)

        # Gerando a GLCM com distância 1 e angulo 90
        # levels: indica os níveis de cinza da imagem
        # symmetric: se True, a matriz de saída é simétrica. O padrão é False.
        # normed: se True, normaliza os dados da matriz de saída. O padrão é False.
        matrix_coocurrence = greycomatrix(img, [1], [90], levels=img.max()+1, normed=False, symmetric=False)

        # Adicionando as features na lista, que são os 6 atributos extraídos da GLCM
        feat = []
        feat.append(greycoprops(matrix_coocurrence, 'contrast')[0][0])
        feat.append(greycoprops(matrix_coocurrence, 'dissimilarity')[0][0])
        feat.append(greycoprops(matrix_coocurrence, 'homogeneity')[0][0])
        feat.append(greycoprops(matrix_coocurrence, 'energy')[0][0])
        feat.append(greycoprops(matrix_coocurrence, 'correlation')[0][0])
        feat.append(greycoprops(matrix_coocurrence, 'ASM')[0][0])

        # Salvando a lista de features no arquivo GLCM
        print('imagem ' + str(i) + ' da classe ' + str(classe) + ' foi processada ')
        name = 'glcm.libsvm'
        path_file = '../output/'
        geraSVMfile(rotulo=classe, lista_feat=feat, name_file=name, path_out=path_file, modo=modo)

# Obtendo as imagens da base
def getImagens():

    path_melanoma = '../input/ph2/melanoma/'
    path_nmelanoma = '../input/ph2/naomelanoma/'
    extensao = '*.bmp'

    lista_melanoma = g.glob(path_melanoma + extensao)
    lista_nmelanoma = g.glob(path_nmelanoma + extensao)

    run_imagens(classe=1, lista=lista_melanoma, modo='a')
```

```
    run_imagens(classe=0, lista=lista_nmelanoma, modo='a')

    #melanoma => classe 1; não melanoma => classe 0

# Chamando as funções
if __name__ == '__main__':
    getImagens()
```

Código Fonte 4.1: Descritor de Textura - Matriz de Co-Ocorrência de Níveis de Cinza.

```
# Importando as bibliotecas
import glob as g
from skimage.io import imread
from skimage.color import rgb2gray
import numpy as np

# Método para salvar o arquivo de características em libsvm
def geraSVMfile(rotulo, lista_feat, name_file, path_out, modo):
    arquivo = open(path_out + name_file, modo)
    featureFile = str(rotulo) + " "
    arquivo.write(featureFile)
    for i in range(len(lista_feat)):
        linha = str(str(i + 1) + ":" + str(lista_feat[i]) + " ")
        arquivo.write(linha)
    arquivo.write('\n')
    arquivo.close()

# Extraindo as características
def run_imagens(classe, lista, modo):

    somatorio = 0
    histograma = [0]*256
    especie_ = 0

    for i in range(len(lista)):

        name = lista[i].split('/')[-1].split('.')[0].split('g')[0]

        img = imread(lista[i])
        img = rgb2gray(img)*255
        img = np.uint32(img)

        min = img.min()
        sizeHistograma = 256

        for b in range(img.shape[0]):
            for c in range(img.shape[1]):
                value = img[b][c]
                if int(value) > 0:
                    histograma[value] += 1

        for d in range(0, sizeHistograma):
            if int(histograma[d]) > 0:
                especie_ += int(histograma[d])

```

```
        # diversidade taxonômica
        somatorioDelta = 0.0
        for x in range(0, sizeHistograma):
            somatorio_initial = 0.0
            for y in range(x + 1, sizeHistograma):
                val = histograma[x] * histograma[y] * ((y - x) + 1)
                somatorio_initial += val
            somatorioDelta += somatorio_initial
        deLta = somatorioDelta / (especie_ * (especie_ - 1) / 2)

        # distinção taxonômica
        somatorioDelta_ = 0.0
        somatorioDelta_q = 0.0
        for x in range(0, sizeHistograma):
            somatorio_initial_ = 0.0
            somatorio_initialq = 0.0
            for y in range(x + 1, sizeHistograma):
                val = histograma[x] * histograma[y] * ((y - x) + 1)
                valq = histograma[x] * histograma[y]
                somatorio_initial_ += val
                somatorio_initialq += valq
            somatorioDelta_ += somatorio_initial_
            somatorioDelta_q += somatorio_initialq
        delta_ = somatorioDelta_ / somatorioDelta_q

        # lista de características
        features = []
        features.append(deLta)
        features.append(delta_)

        # Salvando as características no arquivo libsvm
        print('imagem ' + str(i) + ' da classe ' + str(classe) + ' foi
    processada ')
        name = 'taxonomicos.libsvm'
        path_file = '../output/'
        geraSVMfile(rotulo=classe, lista_feat=features, name_file=name,
     path_out=path_file, modo=modo)

# Obtendo as imagens da base
def getImagens():

    path_melanoma = '../input/ph2/melanoma/'
    path_nmelanoma = '../input/ph2/naomelanoma/'
    extensao = '*.bmp'

    lista_melanoma = g.glob(path_melanoma + extensao)
    lista_nmelanoma = g.glob(path_nmelanoma + extensao)

    run_imagens(classe=1, lista=lista_melanoma, modo='a')
    run_imagens(classe=0, lista=lista_nmelanoma, modo='a')

    #melanoma => classe 1; não melanoma => classe 0

# Chamando as funções
if __name__ == '__main__':
```

```
100     getImagens()
```

Código Fonte 4.2: Descritores de textura - Índices Taxonômicos.

```
1
# Importando as bibliotecas
3 import matplotlib.pyplot as plt
%matplotlib inline
5 import os
from keras.applications.vgg16 import VGG16, preprocess_input
7 from keras.preprocessing import image
from skimage.io import imread, imsave
9 import numpy as np
from keras.models import Model
11 import pandas as pd
import csv
13 from glob import glob

15 # Carregando os pesos da VGG16 para a ImageNet
model = VGG16(weights='imagenet', include_top=True)
17
# Visualizando as camadas da VGG16
19 model.summary()

21 # Utilizando apenas os pesos da última camada como características
model = Model(input=model.input, output=model.get_layer('fc1').output)
23
# Adquirindo as imagens da base
25 path_melanoma = "../input/ph2/melanoma/"
path_naomelanoma = "../input/ph2/naomelanoma/"
27
lista_melanoma = glob(path_melanoma+'*.bmp')
29 lista_naomelanoma = glob(path_naomelanoma+'*.bmp')

31 # Aplicando os pesos da VGG16 nas imagens para obter as características
features = []
33
for i in range(len(lista_melanoma)):
35     img = image.load_img(lista_melanoma[i], target_size=(224, 224))
    x = image.img_to_array(img)
37
    # Expande o array, inserindo um novo eixo
39     x = np.expand_dims(x, axis=0)

41     # Adequa sua imagem ao formato exigido pelo modelo
    x = preprocess_input(x)
43
    # Aplica a imagem no modelo e retorna as características extraídas
45     feature = model.predict(x).reshape(-1)

47     # Transformando de np pra list
    feature = list(feature)
49
    # Adicionando label na lista
51     feature.append(1)
```

```
    # Adicionando esse vetor de features na matriz de features
    features.append(feature)

for i in range(len(lista_naomelanoma)):
    img = image.load_img(lista_naomelanoma[i], target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    feature = model.predict(x).reshape(-1)
    feature = list(feature)
    feature.append(0)
    features.append(feature)

# Salvando as características no arquivo libsvm
def geraSVMfile(rotulo, lista_feat, name_file, path_out, modo):
    arquivo = open(path_out + name_file, modo)
    featureFile = str(rotulo) + " "
    arquivo.write(featureFile)
    for i in range(len(lista_feat)-1):
        linha = str(str(i + 1) + ":" + str(lista_feat[i]) + " ")
        arquivo.write(linha)
    arquivo.write('\n')
    arquivo.close()

name = 'VGG16.libsvm'
path_file = '../output/'

for i in range(len(features)):
    classe = features[i][-1]
    geraSVMfile(rotulo=classe, lista_feat=features[i], name_file=name,
    path_out=path_file, modo='a')
```

Código Fonte 4.3: Extraindo características com a VGG16, usando Transfer Learning.

## Referências

[Abadi et al. 2015] Abadi, M. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. `http://tensorflow.org/`. Software disponível em tensorflow.org.

[Aha e Kibler 1991] Aha, D. e Kibler, D. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

[Al-Rfou et al. 2016] Al-Rfou, R. et al. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

[Araújo et al. 2017] Araújo, F. H., Carneiro, A. C., Silva, R. R., MEDEIROS, F. N., e USHIZIMA, D. M. (2017). Redes neurais convolucionais com tensorflow: Teoria e prática. *SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos*, 1:382–406.

[Baraldi e Parmiggiani 1995] Baraldi, A. e Parmiggiani, F. (1995). An investigation of the textural characteristics associated with gray level cooccurrence matrix statistical

parameters. *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING*, 33(2).

[Baxevanis e Ouellette 2004] Baxevanis, A. D. e Ouellette, B. F. (2004). *Bioinformatics: a practical guide to the analysis of genes and proteins*, volume 43. John Wiley & Sons.

[Breiman 2001] Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

[Carvalho Filho et al. 2016] Carvalho Filho, A. O. d. et al. (2016). Métodos para sistemas cad e cadx de nódulo pulmonar baseada em tomografia computadorizada usando análise de forma e textura.

[CLARKE e WARWICK 1998] CLARKE, K. R. e WARWICK, R. M. (1998). A taxonomic distinctness index and its statistical properties. *Journal of Applied Ecology*, 35(4):523–531.

[Collobert et al. 2011] Collobert, R., Kavukcuoglu, K., e Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. Em *BigLearn, NIPS Workshop*.

[Cortes e Vapnik 1995] Cortes, C. e Vapnik, V. (1995). Support-vector networks. Em *Machine Learning*, páginas 273–297.

[Godinho de Oliveira et al. 2018] Godinho de Oliveira, B. A., Pinho Ferraz, P., Machado-Coelho, T., Rungue, A., Antonio Dos Santos, W., Magalhães, F., Góes, L., Soares, G., e Martins, C. (2018). Avaliação de técnicas de transferência de aprendizado em cnns.

[Gonzalez et al. 2002] Gonzalez, R. C., Woods, R. E., et al. (2002). Digital image processing.

[Haralick et al. 1973] Haralick, R. M., Shanmugam, K., e Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621.

[Haykin et al. 2009] Haykin, S. S., Haykin, S. S., Haykin, S. S., e Haykin, S. S. (2009). *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA:.

[Jia et al. 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., e Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[Karpathy 2015] Karpathy, A. (2015). Transfer learning and fine-tuning convolutional neural networkss. `http://cs231n.github.io/transfer-learning/`.

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

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

[Liu et al. 2017] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., e Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

[Magurran 2013] Magurran, A. E. (2013). *Measuring biological diversity*. John Wiley & Sons.

[Mendonça et al. 2013] Mendonça, T., Ferreira, P. M., Marques, J. S., Marçal, A. R. S., e Rozeira, J. (2013). Ph2 - a dermoscopic image database for research and benchmarking. *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, páginas 5437–5440.

[Pedrini e Schwartz 2008] Pedrini, H. e Schwartz, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações*. Thomson Learning.

[Schwartz e Pedrini 2003] Schwartz, W. R. e Pedrini, H. (2003). Método para classificação de imagens baseada em matrizes de coocorrência utilizando características de textura. *III Colóquio Brasileiro de Ciências Geodésicas, CuritibaPR, Brasil*, páginas 1.

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

[Simonyan e Zisserman 2014b] Simonyan, K. e Zisserman, A. (2014b). Very deep convolutional networks for large-scale image regognition. Em *CVPR14*.

[Vedaldi e Lenc 2015] Vedaldi, A. e Lenc, K. (2015). Matconvnet – convolutional neural networks for matlab. Em *Proceeding of the ACM Int. Conf. on Multimedia*.

Capítulo
# 5

# Alocação de Recursos no contexto de *Network Slicing as a Service* baseado em Algoritmos Genéticos: Teoria e Prática

Rayner Gomes Sousa

## 5.1. Introdução

Redes do Futuro representa uma visão de redes que operam em perfeita integração entre os diversos sistemas heterogêneos, cujos parâmetros de QoS são específicos e distintos em conformidade com a tarefa a ser desempenhada por cada sistema respectivamente. Sobre estas redes, os aplicativos aproveitam a enorme quantidade de recursos computacionais difundidos e compartilhados pelos provedores de infraestrutura para prover novos serviços. Os aplicativos impõem alguns SLAs (*Service Level Agreement* - Acordos de Nível de Serviço) diversificados, que devem ser sustentados de ponta-a-ponta pelas infraestruturas de rede. Sob a perspectiva do usuário, todo o sistema funciona como uma entidade única e sem falhas [1].

A gama de novos aplicativos sobre as redes do futuro é uma consequência do rápido progresso da integração de circuitos digitais, da onipresença da rede de acesso sem fio (*wireless*) e do rápido processo de implantação de aplicativos em razão da maturidade de plataformas modernas, como *Cloud Computing* e metodologias de desenvolvimento ágeis fomentadas por tecnologias de virtualização leve (p.ex: contêiner) e estilo arquitetural como o de microsserviços [2].

*Internet of Things* (IoT ou *Internet* das Coisas) é um exemplo atual e distinto desses novos tipos de aplicativos. Atualmente, os aplicativos da IoT estão presentes em muitas áreas da atividade humana, como agricultura, saúde eletrônica, transporte, casas inteligentes e cidades inteligentes, entre outras áreas. Essas áreas distintas ilustram o quão diverso e heterogêneas são essas aplicações. Além disso, paradigmas como o *Machine-to-Machine* (M2M), *Device-to-Device* (D2D) e *Ad-Hoc Vehicular* (VANET) são exemplos de novos paradigmas com diferentes demandas de rede. Para dar suporte a esses aplicativos é amplamente reconhecido que uma nova arquitetura seguindo o modelo *"one-fit-all"* (uma única para todos) não é viável. O problema da inflexibilidade da *Internet*, conhecido na literatura como o problema de "ossificação", enfrentado pela arquitetura atual da *Internet*,

ensina que é extremamente difícil manter uma arquitetura de rede global e adaptada a diversas classes de aplicações [3].

O 5G (Quinta Geração de Telefonia Móvel), de forma a responder a demandas das novas aplicações, citadas no parágrafo anterior, além de tornar as Redes de Futuro factíveis, visa fornecer uma plataforma flexível para possibilitar novos casos de negócios (*business cases*) e integrar setores verticais (*vertical markets*) como o setor: automotivo; manufatura de larga escala; entretenimento; e a IoT. Assim, o *network slicing* surge como um modelo de serviço promissor para que uma infraestrutura de rede contemple os setores diferentes de negócios e indústrias [2, 4–6].

Uma fatia de rede (*network slice*), doravante simplesmente nomeada por fatia, é composta por uma coleção de recursos e serviços que adequadamente combinados, atendem aos requisitos de uma aplicação ou a um grupo de aplicações com SLA semelhantes, sendo eles de um *business case* específico, ou de forma mais ampla de um *vertical market* [7]. Um proprietário de uma fatia pode ser qualquer usuário que deseja alugar uma parte da infraestrutura da rede, também nomeado por substrato, sem nenhum conhecimento sobre como os recursos são mantidos. Portanto, uma fatia significa diminuir o custo total de propriedade (*Total Cost of Ownership* - TCO) para os usuários finais. Logo, o *network slicing* é um novo produto a ser adicionado ao portfólio de operadoras de rede. Essa solução de rede é oferecida através de um serviço conhecido na literatura como NSaaS (*Network Slicing as a Service*) [7–11].

O NSaaS gerencia o ciclo de vida das fatias [12]. Para viabilizar esse serviço, o provedor precisa planejar os requisitos de cada fatia juntamente com seu cliente, executar a criação de fatia pelo operador de rede, se necessário orquestrar a criação de fatia com outros operadores de rede, relação intra-domínio, com base no conceito ponto a ponto de uma fatia, além de gerenciar as requisições de cada fatia para evitar uma quebra de contrato.

## 5.2. Visões e Oportunidades

### 5.2.1. Visões

#### 5.2.1.1. Visão do Mercado: Fatias de rede como fonte de lucro

Além das fatias de rede serem uma maneira flexível de lidar com aplicativos complexos, é uma possível fonte de lucro para o provedor de infraestrutura. Por exemplo, o trabalho [11] analisou o impacto do fatiamento de rede no gerenciamento de recursos e na geração de lucro. O trabalho mostra que o gasto e a receita de uma rede podem ser estimados de acordo com as propriedades de fatia, como requisitos descritos por seus indicadores de performance *Key Performance Indicator - (KPI)* e os preços dos serviços a serem instalados em cada fatia. Portanto, evitar o fornecimento excessivo ou insuficiente de recursos é essencial para os mecanismos de mapeamento. O excesso de provisão de recursos implica menos lucro para o provedor quando novas fatias de rede não podem ser criadas devido ao esgotamento de recursos.

#### 5.2.1.2. Visão da Computação: Complexidade do Mapeamento

O fatiamento de rede usa uma classe de algoritmos comumente conhecida como VNE (*Virtual Network Embedding*) [13]. Um VNE é um processo de mapeamento de um conjunto de nós virtuais para um conjunto de nós reais e uma coleção de enlaces virtuais para um conjunto de enlaces reais. Os nós e os enlaces virtuais são formalmente requisitados em uma solicitação de rede virtual (*Virtual Network Requested* - VNR). Uma VNR também está associada a um SLA específico. Como o problema do VNE é de complexidade NP-difícil, uma vez que é um problema de combinação, encontrar uma abordagem ideal requer o uso de heurísticas que tendem a relaxar a questão original, concentrando-se em soluções quase ideais. Por exemplo, o trabalho [14] comprovou que há um dilema entre alocação ótima de nós e alocação ótima de enlaces.

#### 5.2.1.3. Redes de Futuro: Novos cenários, 5G e IoT

Apesar do VNE ter capturado a atenção dos pesquisadores ao longo do tempo, tais trabalhos não levam em consideração o cenário 5G em virtude do momento o qual eles foram projetados [15–17]. Além disso, as novas classes de aplicativos IoT exigem novos parâmetros de QoS, decorrendo em novos requisitos, logo invalidando trabalhos de outrora. A infraestrutura projetada para o 5G utiliza novos paradigmas, como SDN [1] (*Software Defined Networking*) e NFV [8] (*Network Function Virtualization*), que impõem novas restrições criando novas possibilidades de algoritmos VNE. Portanto, o VNE no momento atual está suscetível a novas abordagens diante dos desafios em aberto. Por exemplo, o primeiro trabalho que leva em consideração uma restrição básica do SDN, o tamanho da tabela de encaminhamento (denominada por *flowtable*), no processo de mapeamento foi realizado em [18]. E, somente, em 2019 o primeiro mapeamento levando em consideração a característica multi-domínio e multi-enlace das redes de acesso segundo o modelo do 5G foi considerado no projeto de [19].

#### 5.2.1.4. Realocação: Ofertas de fatias como um serviço dinâmico

Um mapeamento de VN (*Virtual Network* - Rede Virtual) pode se tornar subótimo ao longo do tempo devido à chegada e partida de outras VNs, bem como devido a alterações no substrato em virtude de *upgrades*, *downgrades* ou a falhas na rede [19]. Uma maneira de mitigar o impacto desse dinamismo da rede é realocar periodicamente recursos às VNs existentes. A realocação de VNE pode aumentar a receita do provedor de infraestrutura (*Infrastructure Provider* - InP), diminuindo o consumo de largura de banda e aumentando a possibilidade de aceitar futuros VNs [20]. Adicionalmente, a fragmentação de recursos pode ocorrer à medida que as solicitações de VN são atendidas e encerradas no decorrer do tempo, a alocação de recursos no substrato se torna ineficiente, com muitos recursos fragmentados podendo ser insuficientes para atender a uma nova VN [21].

Dois desafios são de grande importância no momento atual. Primeiro, a considerar a mudança dinâmica das instanciações das funções de rede. Paradigmas como NFV (*Network Function Virtualization*) [22] desacopla funções de respectivos serviços de rede

tradicionalmente encapsulados em um dispositivo e permite instanciá-los em diversas parte da rede. Logo, um serviço de rede pode ser fragmentado por uma sequencia específica de funções que podem ser instanciadas em locais distintos na rede. A disposição das funções tem como consequência a mudança do fluxo no encaminhamento de pacotes na rede. Este comportamento é na literatura conhecido como serviço dinâmico presente no paradigma SFC (*Service Function Chaining*) [23]. Como apontado por [5], tecnicamente o desenvolvimento dessa alocação ótima de recursos de uma fatia é muito desafiador, principalmente quando se considera um conjunto de requisitos funcionais (funções) que devem ser conhecidos antecipadamente.

O segundo desafio, também apontado por [5], se não houver recursos suficientes, p.ex: na ocorrência de falhas na rede, o gerenciador de fatias precisará buscar o menor impacto para todas as fatias implantadas. A eficiência de Pareto ou a otimização de Pareto é um estado de alocação de recursos a partir do qual é impossível realocar para melhorar qualquer critério individual ou de preferência, sem piorar pelo menos um critério individual ou de preferência.

### 5.2.2. Oportunidades

#### 5.2.2.1. 5G: Nova Geração da Telefonia Móvel

A arquitetura 5G precisa fornecer redes de alta flexibilidade, baixa latência e alta capacidade para suportar o aumento estimado de mil vezes do tráfego de dados móveis, fornecendo latência abaixo de milissegundos [24–26]. 5G é a primeira proposta de telecomunicações que, desde o início, incluiu os conceitos de IoT em seu projeto.

As promessas do 5G são ambiciosas, a proposta é que o 5G consista em um grande número de dispositivos com características variadas, redes diferentes, como macrocélula, célula pequena, WiFI e MIMO (*Multiple Input and Multiple Output*). Além disso, o 5G é composto por modelo de negócios complexos, como *Internet* das Coisas, M2M (*Machine to Machine*) e D2D (*Device to Device*). Para a rede estes complexos modelos trazem restrições combinatórias rígidas de impacto ao desempenho, largura de banda, latência e no uso de energia. Além disso, após alcançar a primeira fase da arquitetura do sistema [1], foi definido que os serviços de rede são entregues seguindo o modelo de rede como serviço (NSaaS). A Fig. 5.1 ilustra diferentes fatias compartilhando uma mesma infraestrutura física.

#### 5.2.2.2. IoT: *Internet of Things*

*Internet* das Coisas (*Internet of Things* - IoT) é um paradigma que visa o uso de dispositivos inteligentes que se conectam e interagem para criar um ambiente integrado de monitoramento, visando conectar pessoas com pessoas, pessoas com objetos e objetos com objetos, todos compartilhando milhões de informações via *Internet*. Entende-se por objetos qualquer dispositivo eletrônico embarcado em objetos físicos e com capacidades mínimas de processamento, comunicação e sensoriamento.

[1] http://www.3gpp.org/news-events/3gpp-news/1930-sys_architecture

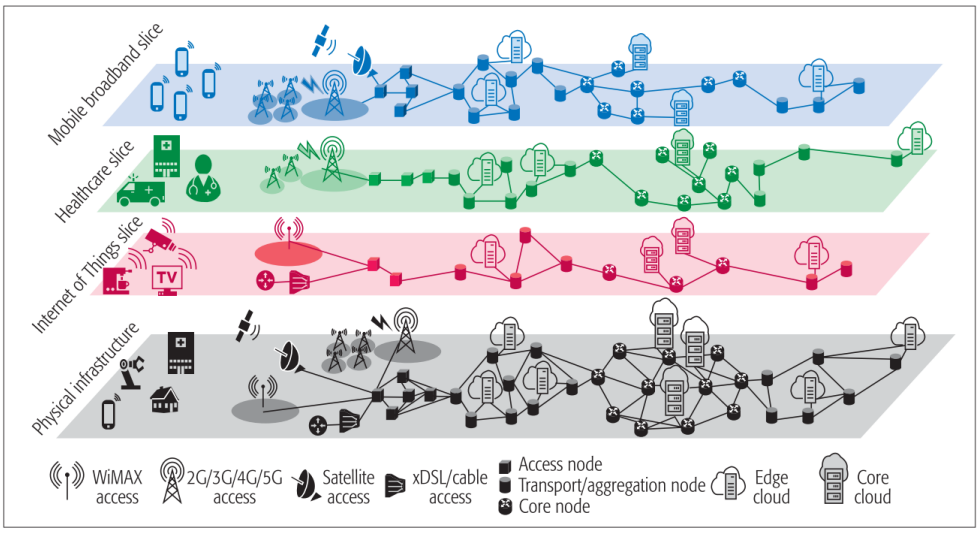



**Figura 5.1. *5G and Slices*, imagem original: [8].**

A IoT é um paradigma emergente e exige uma rede de computadores dinâmica para ajudar a atender a todos os requisitos dos novos tipos de aplicativos. Com o tempo, a *Internet* recebeu muitas melhorias para o incremento de sua escalabilidade, segurança, mobilidade e Qualidade de Serviço (QoS) [27–29]. Apesar de todos esses avanços, a *Internet* não está preparada para a magnitude de bilhões de objetos. Atualmente, a *Internet* precisa suportar muitos tipos de redes sem fio e muitos aplicativos, cada um dos quais com requisitos explícitos e rigorosos. Portanto, a *Internet* precisa de um novo modelo de arquitetura. É reconhecido que uma única arquitetura para todas os tipos de aplicações IoT, p.ex: *smartcity*, *smarthome*, *e-health*, transporte não é factível [12, 30], logo a abordagem de criar fatias para cada classe de aplicações se mostra mais realístico, de modo que a implantação de NSaaS se torna também crucial para implantação de diversas aplicações de IoT sobre uma infraestrutura de rede compartilhada.

#### 5.2.2.3. Redes Definidas por *Software*

Rede definida por *software* (*Software-Defined Networking - SDN*) é um paradigma de rede que quebra a rigidez das redes tradicionais, removendo a grande quantidade de recursos espalhados em cada equipamento e os posicionando em um ponto central do sistema. Essa característica permite incorporar novas funções à rede com menos complexidade e independentemente dos fornecedores de equipamentos. Nos últimos anos, as soluções de rede estão transformando cada dispositivo de rede em um dispositivo de rede sofisticado e especializado, contribuindo para tornar o sistema mais difícil de gerenciar e de incrementar novas funcionalidades.

A proposta apresentada pela SDN é desacoplar o encaminhamento feito pelo *hardware* do processo de decisão feito por *software*. Ela promete simplificar drasticamente o gerenciamento de rede e facilitar a inovação e evolução. Nas SDNs a inteligência fica centralizada em um componente especial chamado de controlador, responsável pelo plano de controle. Plano de controle no contexto das SDNs representa toda lógica das redes, p.ex:

roteamento, mobilidade, controle de acesso, gerenciamento de recursos, filtragem de pacotes e monitoramento. Os dispositivos de rede (roteadores, *switches* e *middlebox*) ficam responsáveis apenas pelo encaminhamento, o encaminhamento é baseado em tabelas (*flowtables* [31]) configuradas pelo controlador.

SDN é amplamente reconhecida como um pilar para implantação dinâmica de fatias de rede [15]. Assim, os algoritmos de mapeamento devem levar em consideração o paradigma SDN. Não existe uma padronização da forma de como o SDN deve ser usado. Recentemente, a *Open Network Foundation* [9] liberou uma arquitetura de referência para aplicação de SDN em conjunto com o conceito de virtualização de rede. O trabalho de [9] descreve os principais aspectos funcionais da arquitetura SDN que se aplicam à habilitação do conceito de virtualização de rede orientado a negócios, em conformidade com os principais conceitos do 5G, conforme previsto pela aliança NGMN [2] [32].

## 5.3. Referencial Teórico

### 5.3.1. *Virtual Network Embedding*

O fatiamento de rede (*network slicing*) usa uma classe de algoritmos conhecida como VNE (*Virtual Network Embedding*) [12, 13, 15]. Um VNE é um processo de mapeamento de um conjunto de nós virtuais para um conjunto de nós reais, e uma coleção de enlaces virtuais para um conjunto de enlaces reais. Os nós e os enlaces virtuais são formalmente requisitados através de um documento descritivo conhecido como *Virtual Network Request* (VNR) (ilustrado na Fig.5.2). Um VNR está associado a um SLA específico da VN a ser criada. Sob a visão das redes os recursos associados a uma VN fazem parte dos recursos assegurados a sua respectiva fatia. Como o problema do VNE é NP-Difícil (ou em inglês *NP-Hard*), ou seja, faz parte de uma classe de problemas combinatórios que a solução ótima não é calculada em tempo linear, apenas a checagem da solução pode ser feita em tempo polinomial em função da entrada. Uma vez que é um problema combinatório, encontrar uma abordagem eficiente e eficaz requer o uso de heurísticas que tendem a relaxar a questão original. Por exemplo, o trabalho [14] comprova que há um dilema (*tradeoff*) entre a alocação ótima de nós e alocação ótima de enlaces.

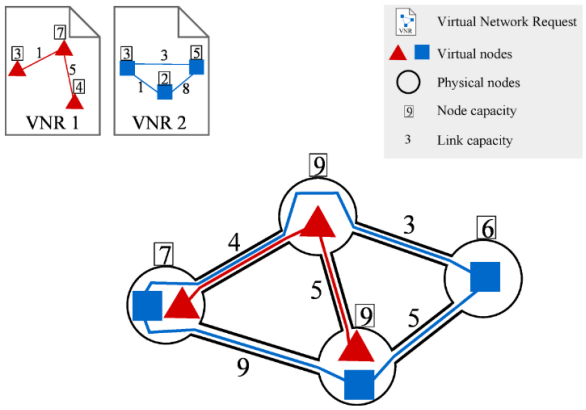



**Figura 5.2. Mapeamento de redes virtuais para uma rede física [13].**

[2] https://www.ngmn.org

O fatiamento de rede e a virtualização de rede são conceitos que usam algoritmos VNE, portanto, eles têm algumas semelhanças. Os seguintes pontos destacam as características de uma fatia [10]:

- Cada fatia deve ser isolada de outra instância.
- Uma fatia compreende recursos físicos e lógicos.
- Uma fatia pode ser compostas por sub-instâncias.
- Uma instância é formalmente definida através de um documento conhecido como *network slice blueprint* ou simplesmente por VNR.
- Políticas e regras são requeridas no momento da criação de uma fatia.
- O fatiamento da rede precisa ser implementado de maneira ponta a ponta (*end-to-end* - E2E) para atender a diversos requisitos de serviço.

### 5.3.2. Formalização Matemática

Formalmente, o problema VNE pode ser descrito como um grafo. Seja $SN = (S, L)$ o substrato da rede, onde $N$ representa o conjunto de nós do substrato e $L$ o conjunto de enlaces. Seja, $VNR_i = (N_i, L_i)$ um conjunto de $i = 1, ..., n$ requisições de redes virtuais (*Virtual Network Request*) onde $N_i$ e $L_i$ representa o conjunto de nós e enlaces virtuais de uma respectiva $VNR_i$. Adicionalmente, seja $R\prod_{j=1}^{m} R_j$ um vetor de espaço de recursos $R_1, ..., R_m$ e seja $cap : N \cup L \rightarrow R$ uma função que associa recursos disponíveis aos elementos do substrato da rede. Finalmente, para cada $VNR_i$, seja $dem_i : N_i \cup L_i \rightarrow R$ uma função que associa demandas de todos elemento de todas requisições.

Logo, VNE consiste de funções $f_i : N_i -> N$ e $g_i : L_i -> SN' \subseteq SN$ contida em SN para cada $VNR_i$ tal que $\forall n^i \in N^i : demi_i(n^i) \leq cap(f_i(n^i))$ e $\forall l^i \in L^i : \forall g_i(l^i) : dem_i(l^i) \leq cap(l)$. $f_i$ é então chamada de função de mapeamento de nós e $g_i$ é chamada função de mapeamento de enlace. Juntos elas formam uma mapeamento (*embedding*) para $VNR_i$. Não é requerido que estas duas funções sejam calculdas com uma única entidade, o calculo pode ser dividido em multiplas entidades. A Fig.5.3 resume as definições apresentadas nesta seção.

| Term | Description |
|---|---|
| $SN = (N, L)$ | $SN$ is a substrate network, consisting of nodes $N$ and links $L$ |
| $VNR^i = (N^i, L^i)$ | $VNR^i$ denotes the $i^{th}$ Virtual Network Request, consisting of nodes $N^i$ and links $L^i$ |
| $\dot{R} = \prod_{j=1}^{m} R_j$ | $\dot{R}$ contains resource vectors for all resources $R_1, ..., R_m$ |
| $cap : N \cup L \rightarrow \dot{R}$ | The function $cap$ assigns a capacity to an element of the substrate network (either node or link) |
| $dem_i : N^i \cup L^i \rightarrow \dot{R}$ | The function $dem_i$ assigns a demand to an element of $VNR^i$ (either a node or a link) |
| $f_i : N^i \rightarrow N$ | $f_i$ is the function that maps a virtual node of $VNR^i$ to a substrate node (VNoM) |
| $g_i : L^i \rightarrow SN' \subseteq SN$ | $g_i$ is the function that maps a virtual link of $VNR^i$ to a path in the substrate network (VLiM) |

**Figura 5.3. Terminologia usada na descrição de uma VNE [13].**

#### 5.3.2.1. Complexidade da VNE

Problemas de mapeamento (VNE) são considerados de complexidade *NP-Hard*, isto indica que a melhor solução não pode ser encontrada em tempo polinomial. A complexidade do mapeamento somente dos enlaces pode ser reduzidos a classe de problemas conhecidos como *unsplittable flow problem* que também tem a complexidade *NP-Hard*. Como provado pelo trabalho [14] há um *tradeoff* entre encontrar soluções ótimas quando tenta-se atender os requisitos dos nós e dos enlaces. Há cenários aonde não se pode encontrar ao mesmo tempo soluções ótimas de mapeamento dos nós e enlaces. Ademais, quando o mapeamento leva em consideração requisitos como custo monetário ou parâmetros de QoS o processo se torna mais complexo. Assim, atualmente o foco principal do trabalho dentro da comunidade de pesquisa são as abordagens heurísticas ou meta-heurísticas.

Resolver o problema do VNE é um *NP-hard*, pois está relacionado com a associação de um nó virtual a um nó real (ou nó físico). Dois nós virtuais de uma mesma requisição não podem ser associados ao mesmo nó físico. O problema de alocar de maneira otimizada um conjunto de enlaces virtuais para caminhos de um substrato depende da topologia da VN e da associação dos nós virtuais. Obter estas rotas consiste na resolução do *Unsplittable Flow Problem - UFP* [20]. Os problemas de UFP são *NP-Hard*. O problema da mochila clássico (*knapsack*) pode ser reduzido para UFP, considerando UFP com uma única aresta $e = (u, v)$ de capacidade $c(e) = W$ igual a capacidade $W$ da mochila. A demanda $d_i$ é igual ao peso $p_i$ para o item i da mochila e $w_i = 1$. Desta forma, não existe uma solução polinomial para o problema do UFP.

Portanto, soluções realmente ótimas só podem ser adquiridas para pequenas instâncias do problema. Assim, atualmente, o foco principal do trabalho dentro da comunidade de pesquisa são as abordagens heurísticas ou metaheurísticas.

## 5.4. Algoritmo Genético

As seções 5.1 e 5.2 destacam a importância da virtualização da rede como uma alternativa para maximizar os lucros dos provedores e de atender as demandas de aplicações heterogêneas. A seção 5.3.2.1 descreve a complexidade do processo de mapeamento. Como apontado na seção 5.3.1 as heurísticas são ferramentas para realizar o mapeamento em um tempo hábil, ou seja, procurar dentro do espaço de soluções uma solução aceitável.

Várias heurísticas, como a Colônia de Formiga [33], são fundamentadas em processos estocásticos, assim, o fator probabilísticos é incorporado, levando a possíveis e diferentes resultados com a mesma entrada de dados no algoritmo. Os dois trabalhos [13] e [34] são pesquisas amplas e apresentam dezenas de técnicas e heurísticas para realizar o mapeamento.

### 5.4.1. O princípio

Neste trabalho, realizar-se-á o mapeamento utilizando-se de algoritmo genético (AG). AG é um método de resolução de problemas de otimização que é baseado na seleção natural, o processo simula o processo da evolução biológica. A ideia é criar uma população, que representa um conjunto de possíveis soluções, ou seja, valores do espaço

de solução e repetidamente modificar a população. Cada indivíduo da população representa uma solução. E em cada interação, o algoritmo seleciona indivíduos aleatoriamente da população para ser os pais. Os pais são utilizados para gerar uma nova solução, este processo simula a reprodução. Qualquer indivíduo possui características de seus ancestrais. Os filhos produzidos constituem a nova geração da população futura, e a população anterior é descartada. Este constate processo de reprodução, geração de novos filhos e descarte da população antiga simula a evolução das espécies. O processo evolucionário do AG tem o princípio de a cada passo gerar filhos mais próximos da solução ótima. Por isto, que o algoritmo genético é classificado como uma heurística evolutiva [35, 36]. Todos elementos citados estão ilustrados na Fig.5.4

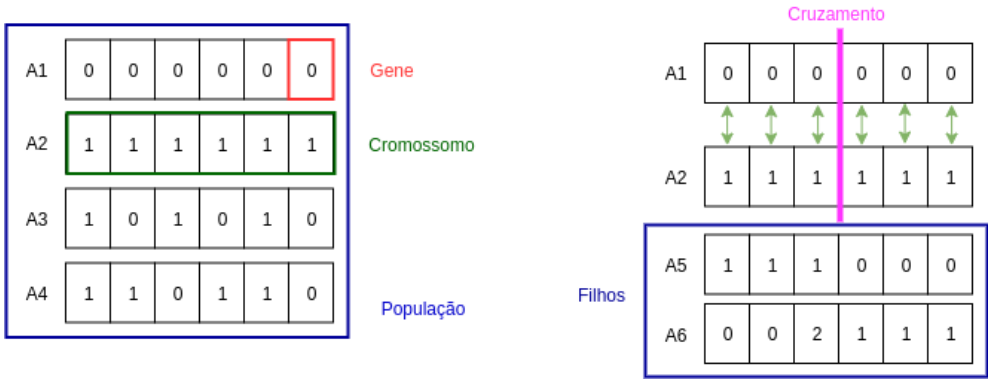



**Figura 5.4. Elementos do AG.**

O AG é uma técnica de busca, ou seja, ele não é um processo determinístico. A ideia básica é gerar um grupo de indivíduos que representa soluções válidas e variadas, e dentro deste grupo realizar operações genéticas como a reprodução e a mutação.

### 5.4.2. Dinâmica do AG

O algoritmo genético usa cinco tipos principais de regras em cada interação para criar a próxima geração de população. Todos os passos encadeados estão ilustrados na Fig.5.5, são eles:

1. Criar a população: Uma população tem $N$ indivíduos, sendo $N$ um valor inteiro e maior que zero. Os indivíduos são criados aleatoriamente, espera-se que o algoritmo gere indivíduos o mais diverso possível. P.ex, se o indivíduo tem um cromossomo tem 4 bits, logo é melhor gerar os indivíduos 0000,0001,0101 e 1010 do que 0000,0000,0000 e 0000.

2. Avaliação: Quanto maior o valor de $N$, sendo $N$ o tamanho da população, maior a possibilidade de gerar indivíduos distintos. A avaliação é uma função que tem como entrada o cromossomo do indivíduo e tem como resultado um valor que traduz o quão próximo o indivíduo está da solução ótima.

3. Seleção: Esta regra filtra indivíduos, chamados pais, que contribuem para nova população. A filtragem é realizada por meio de uma função que seleciona aleatoriamente os pais com maior chances de gerar filhos mais aptos.

4. Reprodução: Simula o processo reprodutório por acasalamento, esta regra combina dois cromossomos de pais distintos para formar um filho (novo indivíduo) da próxima geração. Após a reprodução, a geração de novos indivíduos, o individuo pode sofrer uma mutação. O disparo da mutação ocorre randomicamente, esta regra muda uma parte do valor do cromossomo do filho.

5. Passo final: Os passos 1,2,3 e 4 são repetidos *R* vezes, sendo *R* um valor inteiro. O valor de *R* não é exato para todas as aplicações. A cada desenvolvimento, este valor deve ser descoberto, assim, de acordo com os processos lógicos implementados nos passos 1,2,3 e 4 e após sucessivas avaliações, procura-se encontrar um valor fixo do *R*.

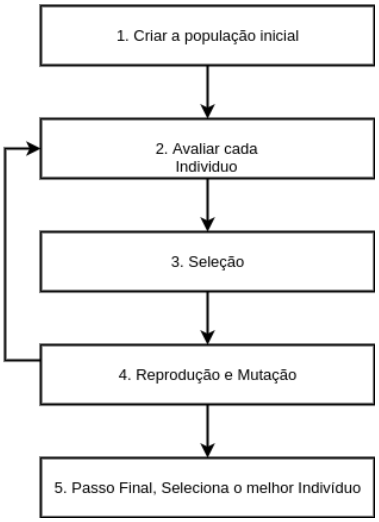



**Figura 5.5. Passos do AG.**

## 5.5. Uma Abordagem do uso de AG para realizar o VNE

A dezenas de heurísticas para realizar o mapeamento [12, 13, 37, 38]. Algoritmo Genético é uma das heurísticas que possui como vantagens [19,36,39]: a. permite a busca de solução em uma ampla área de busca; b. o processo de procura não fica restrito e preso aos espaços de buscas locais; c. é flexível o suficiente para ser adaptado em diversos cenários; d. o algoritmo é passível de ser paralelizado. A estrutura básica do algoritmo, seus componentes e dinâmica foram apresentados na seção 5.4. Esta seção apresenta uma abordagem distinta da aplicação do AG para realizar o mapeamento. Em particular, esta abordagem define a forma de representar um Indivíduo por meio de um cromossomo cuja estrutura represente uma possível forma de mapear e uma função de avaliação (*fitness*) para mensurar a qualidade de um indivíduo. Ressalta-se que problemas complexos, *NP-complexos* (também nomeados de *NP-Hard* ou *NP-difíceis*), não é conhecido a solução ótima a priori, o que se pretende é realizar buscas guiadas e estocásticas para aproximar de uma solução ótima.

A Fig.5.6 ilustra as principais classes usadas no desenvolvimento para modelar o sistema. A figura apresenta as propriedades das 9 classes. Em razão do limite de espaço deste documento todos os métodos foram omitidos.

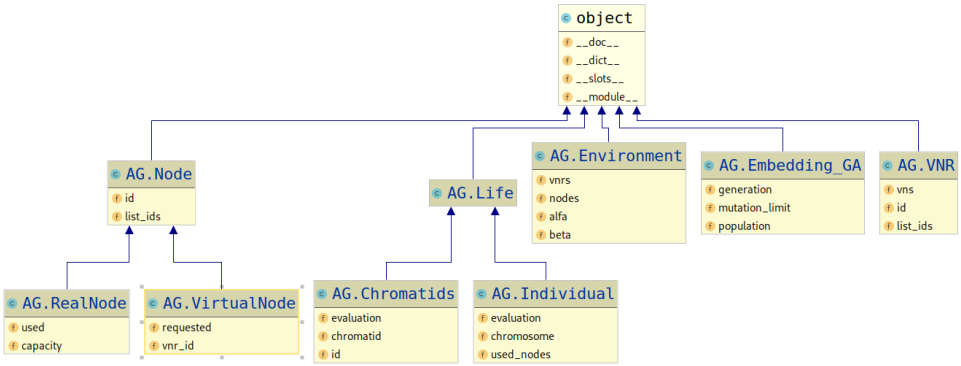



**Figura 5.6. Passos do Algoritmo do AG.**

A finalidade de cada classe ilustrada na Fig.5.6 é:

- ***Node***: Garante a identificação única de cada nó. Cada nó é identificado por um inteiro, a propriedade *id*. Se um nó real tiver dois *id* idênticos uma exceção é lançada.

- ***RealNode***: Classe descendente da *Node*. Especializa a classe *Node* adicionando as propriedades *used* e *capacity*. A primeira propriedade armazena a quantidade de recursos aprovisionados. A segunda, abstrai a quantidade total de recursos do nó. A modelagem é genérica, podendo ser estes valores representar um potencial de processamento, quantidade de memória ou outros recursos computacionais numericamente mensuráveis.

- ***VirtualNode***: Classe descendente da *Node*. Especializa a classe *Node* adicionando as propriedades *requested* e *vnr_id*. A primeira propriedade armazena a quantidade de recursos demandados por um nó virtual pertecente a uma requisição. A segunda identifica a requisição (VNR) a qual o objeto desta classe está vinculado.

- ***Life***: Tem a função simples de verificar se todos os recursos necessários para iniciar a execução do programa estão instanciados e configurados, p.ex: se as requisições e a topologia da rede foram criadas.

- ***Individual***: Representa uma possibilidade de mapeamento. Suas propriedades são: *evaluation*, *chromosome* e *used_nodes*. A primeira propriedade é o resultado da função de avaliação (*fitness*), ela é um valor numérico utilizado para comparação com outro indivíduo, isto é, a outro objeto da mesma classe. A segunda, a mais sofisticada de todo o sistema, é um vetor de objetos da classe *Chromatids*. O cromossomo é uma estrutura complexa e é detalhada na seção 5.5.1.

- ***Chromatid***: Esta classe modela cada parte de um cromossomo. Cada parte de um cromossomo representa um possível mapeamento para uma dada requisição (VNR). P.ex: se houver 10 VNRs a serem mapeadas, um indivíduo terá um cromossomo e cada cromossomo terá um conjunto de 10 *Chromatids*.

- ***Environment***: Seu objetivo é instanciar um grafo para representar o substrato. O substrato é formado por nós (representando os comutadores) e as arrestas (representando os enlaces). Associado aos nós e aos enlaces a um peso como atributo. No caso dos nós o peso representa a capacidade do nó e aos enlaces a capacidade de transmissão (*bandwidth*). As VNRs também são instanciadas nesta classe. A topologia e as VNRs após serem criadas são persistidas em um arquivo texto no formato JSON (*JavaScprit Object Notation*), a finalidade do salvamento das estruturas é que novas execuções podem ser realizadas mantendo o estado atual da execução anterior, logo alterações no AG podem ser comparadas.

- ***Embedding_GA***: Esta classe basicamente implementa o AG. Todos os elementos do AG visto na seção 5.4 são codificadas por esta classe. Logo, ela cria uma população inicial com $P$ indivíduos, sendo $P$ um valor inteiro e maior que 1. Próximo passo, ela reitera $R$ repetições, sendo $R$ um valor inteiro e maior do que 1. No final das $R$ iterações, a classe terá gerado a população $R+1$, contendo nela vários indivíduos. A solução é recuperada da população pelo indivíduo que tem a melhor avaliação.

- ***VNR***: Armazena uma requisição. Esta classe possui como propriedades *id*, *list_ids* e *vns*. A *id* é o identificador da requisição, ela é uma propriedade do objeto sendo utilizada por todo processo de mapeamento. A *list_id* é uma propriedade da classe sendo compartilhada por todos objetos, a sua finalidade é não permitir que duas requisições sejam criadas com o mesmo *id*. A última, *vns* é uma lista de tuplas no formado *(vn, requested, topo)*. *vn* é o identificador de um nó virtual, *requested* representa o quanto ele demanda de recursos. Por fim, *topo* é a topologia da rede virtual.

### 5.5.1. O Cromossomo

No AG a função de avaliação (*fitness*) e a estrutura do cromossomo concentram a parte mais sofisticada da adaptação do processo genérico da heurística a solução de um problema. A Fig. 5.7 ilustra a forma como o problema de mapeamento é representado pelo cromossomo (classe *Chromosome*).

No lado da esquerda da Fig.5.7 ilustra a estrutura de um cromossomo, um cromossomo é composto por uma sequencia de cromátides, cada qual um objeto da classe *Chromatid*. Uma cromátide é uma lista de mapeamentos de nós virtuais em nós reais. As regras para criar uma cromátide são:

1. não pode existir duas cromátides com o mesmo *id*.
2. o *id* de um cromátide representa o *id* da VNR o qual ela está abstraindo.
3. não pode existir em uma mesma cromátide dois mapeamentos com o mesmo nó real. Está regra é uma imposição do VNE.
4. um nó virtual não pode ser mapeado para um nó real cuja capacidade é menor que a demandada pelo nó virtual.
5. cada cromátide possui um valor de avaliação. Este valor é produzido pela função de avaliação (*fitness*) que mensura a qualidade do mapeamento.

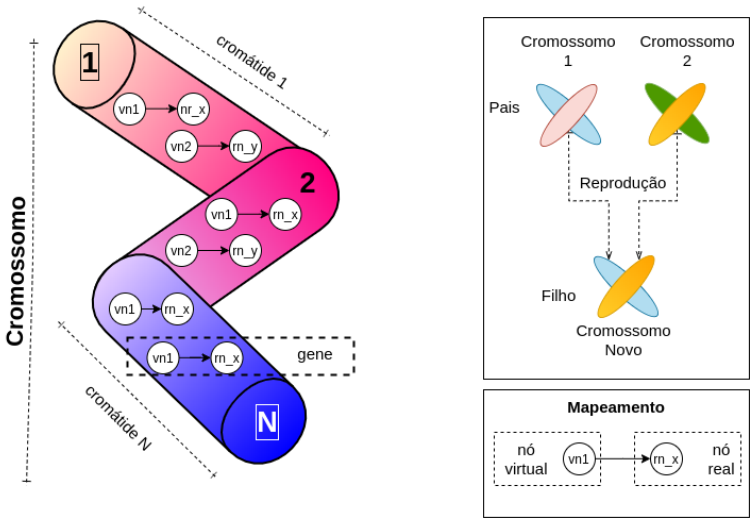



**Figura 5.7. Estrutura do Cromossomo.**

### 5.5.2. A Função de Avaliação

A função de avaliação (FA, em inglês *fitness*) é o que faz do AG adaptar-se a diferentes tipos de aplicações. Para a solução do mapeamento e segundo nossa abordagem a FA irá numericamente representar o quanto a alocação de recursos está espalhada por todo o substrato. Este mecanismo é inspirado pelo trabalho de [14]. Este mecanismo é uma forma genérica que objetiva evitar o *stress* na rede. O *stress* é um conceito definido por [14] que representa o quanto um nó ou um enlace está sobrecarregado. Logo, usando este conceito, a FA avalia o mapeamento com uma alocação uniformemente distribuída melhor do que uma alocação que concentra os mapeamentos em nós únicos. Este mecanismo evita criar pequenos espaços disponíveis para alocação que são tão pequenos que não servirá para nenhuma outro mapeamento, assim, fragmentando os recursos do substrato.

```
def run_evaluation(self):
    alfa = Environment.alfa
    beta = Environment.beta
    sum_resource_nodes = 0
    for nv, nr in self.chromatid:
        sum_resource_nodes += (Environment.nodes[nr].capacity -
                               Environment.vnrs[self.id].vns[nv].requested)
    sum_resource_links = 0
    len_chromatid =  len(self.chromatid)
    for i in range(len_chromatid):  # self.chromatid = [(vn,rn)...()]
        node_a = self.chromatid[i][1]
        node_b = self.chromatid[(i+1) % len_chromatid][1]
        sum_resource_links += Environment.links.get((node_a,node_b),0)
    return (alfa * sum_resource_nodes) + (beta * sum_resource_links)
```

**Figura 5.8. Fragmento do código da FA presente no método *run_evaluation()* da classe *Chromatid*.**

## 5.6. Execução e Resultados

### 5.6.1. Infraestrutura de execução

A tabela 5.6.1 especifica detalhadamente o *hardware* e o *software* utilizados no desenvolvimento, implementação, execução e simulação.

**Especificações**

| | | |
|---|---|---|
| *Hardware* | **CPU** | Intel Core i7-7700HQ 31 |
| | **GPU** | NVIDIA GeForce GTX 1060 (6GB GDDR5) |
| | **HDD** | 2TB 5400rpm. |
| | **SDD** | 500MB M.2 |
| | **RAM** | 8GB |
| *Software* | **SO** | Ubuntu Linux v. 19.10 |
| | **Linguagem** | Python 3.7 |
| | **Bibliotecas** | NetworkX |

#### 5.6.1.1. Topologia de Rede

A biblicoteca *NetworkX*[3] do *Python*[4] foi utilizada para criar a topologia do substrato. O modelo foi baseado no algoritmo de Barabasi[5]. O algoritmo de Barabasi modela o crescimento e a topologia de redes complexas [40]. Diversos trabalhos da area de *network slicing* tem usado o Barabasi para modelar uma rede complexa tal como a infraestrutura física do 5G [41, 42]. A Fig.5.9 apresenta o código, o uso da biblioteca *NetworkX*, atributos da rede, e o gráfico da rede criada com 20 nós e 6 arestas cada.

```
import  networkx as nx
import matplotlib.pyplot as plt
G = nx.read_gml("dataset.gml")
print(nx.info(G))
nx.draw(G)
plt.show()
```

```
Name:
Type: Graph
Number of nodes: 20
Number of edges: 75
Average degree:   7.5000
/usr/local/lib/python3.7/dist-packages/networkx/drawing/nx_py
The iterable function was deprecated in Matplotlib 3.1 and wi
  if not cb.iterable(width):
```

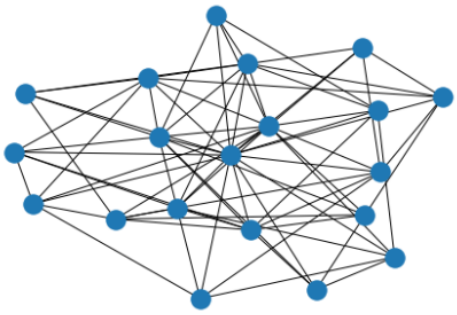


**Figura 5.9. Exemplo de uma Topologia Barabási com 20 nós e 6 enlaces cada.**

### 5.6.2. Resultados

Um exemplo de mapeamento usando AG e a abordagem apresentada neste documento é apresentada abaixo nos dois blocos de saída do programa. O algoritmo foi executado com os seguintes parâmetros: a. repetição = 100; b. tamanho da população =

[3]https://networkx.github.io/
[4]https://www.python.org/
[5]http://barabasi.com/

100; c. quantidade de nós físicos = 200; d. distribuição de conexões dos nós = 4, logo a quantidade máxima de arestas para cada nó é 4.

O bloco a seguir é uma parte da saída do programa. A linha 01 mostra que o indivíduo com a melhor avaliação tem o valor 6 e foi retirado da geração 101. A linha 02 até 04 mostra a estrutura crossomática do indivíduo. O valor [(1, 102), (2, 45), (3, 126)] mostra uma sequência de três tuplas, a primeira tupla (1,102) apresenta que o nó virtual 1 foi mapeado no nó físico 102.

```
01: Chromosome Generation: 101 ,Evaluation: 6.0
02: Chromatid: [(1, 102), (2, 45), (3, 126)]
03: Chromatid: [(1, 28), (2, 121), (3, 30)]
04: Chromatid: [(1, 39), (2, 66), (3, 183)]
```

Finalmente, a próxima parte da saída do programa mostra o tempo de: (linha 01): criar o teste; (linha 02): criar a primeira população; tempo da execução do AG. O valor da linha 03 é em segundo, ou seja, as geração de toda a população com 100 indivíduos, as 100 interações e todo o processo de geração de novos indivíduos levou em torno de 1.08 segundo.

```
01: ('Create the Test', 3.337860107421875e-06)
02: 'Create the first population', 0.2833988666534424)
03: ('Run', 1.0770583152770996)
```

## 5.7. Palavras Finais

### 5.7.1. Interessados em continuar (...)

O código completo está disponível no GitHub[6], pelo endereço [7]. A apresentação (*slides*) em PDF de todo o processo está disponível no endereço [8]. Todo este capítulo faz parte de uma área de estudo, pesquisa e desenvolvimento atual e de grande interesse da comunidade científica e industria. Caso tenha interesse em prosseguir, entre em contato.

### 5.7.2. O autor

Rayner Gomes Sousa (*e-mail* de contato: rayner@ufpi.edu.br) é professor pela Universidade Federal do Piauí (UFPI) desde 2009. Possui experiência como professor universitário desde 2001 ministrando disciplinas na área de redes de computadores, sistemas distribuídos, sistemas operacionais e programação orientada a objeto. Tem interesse em heurísticas de alocação *online* de recursos em redes de computadores aplicadas no contexto da 5G, IoT e *Network Slicing*. Participou do projeto *SCORPION* fruto da parceria entre a Universidade Federal do Ceará (UFC), *École d'ingénieurs Généraliste du Numérique (EFREI - France), Université de La Rochelle (ULR)* e a Nokia. Mestre em Ciência da Computação pela Universidade Federal de Uberlândia (UFU).

[6]https://github.com/
[7]https://bit.ly/34n1B8x
[8]https://bit.ly/36v7ouD

## Referências

M. Agiwal, A. Roy, N. Saxena, Next Generation 5G Wireless Networks: A Comprehensive Survey, IEEE Communications Surveys Tutorials 18 (3) (2016) 1617–1655 (2016). doi:10.1109/COMST.2016.2532458.

T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration, IEEE Communications Surveys and Tutorials 19 (3) (2017) 1657–1681 (2017). doi:10.1109/COMST.2017.2705720.

W. Ejaz, A. Anpalagan, M. A. Imran, M. Jo, M. Naeem, S. B. Qaisar, W. Wang, Internet of Things (IoT) in 5G Wireless Communications, IEEE Access 4 (January) (2016) 10310–10314 (2016). doi:10.1109/ACCESS.2016.2646120.

5G PPP Architecture Working Group, View on 5G Architecture, White paper (July) (2016). doi:10.13140/RG.2.1.3815.7049.

I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, H. Flinck, Network slicing and softwarization: A survey on principles, enabling technologies, and solutions, IEEE Communications Surveys and Tutorials 20 (3) (2018) 2429–2453 (2018). doi:10.1109/COMST.2018.2815638.

A. Gupta, R. K. Jha, A Survey of 5G Network: Architecture and Emerging Technologies, IEEE Access 3 (2015) 1206–1232 (2015). doi:10.1109/ACCESS.2015.2461602.

X. Zhou, R. Li, T. Chen, H. Zhang, Network Slicing as a Service : Enabling Enterprises ' Own Software-Defined Cellular Networks, IEEE Communications Magazine 677 (July) (2016) 146–153 (2016). arXiv:arXiv:1011.1669v3, doi:10.1016/S0140-6736(14)61695-0.
URL https://www.researchgate.net/publication/305339446

J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, J. Folgueira, Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges, IEEE Communications Magazine 55 (5) (2017) 80–87 (2017). doi:10.1109/MCOM.2017.1600935.

Open Networking Foundation, Applying SDN Architecture to 5G Slicing, Tech. Rep. 1, ONF (2016).
URL shorturl.at/dglrK

Description of Network Slicing Concept, Ngmn 5G P1 1 (September) (2016) 19 (2016).
URL shorturl.at/bxFH5

B. Han, S. Tayade, H. D. Schotten, Modeling profit of sliced 5G networks for advanced network resource management and slice implementation, Proceedings - IEEE Symposium on Computers and Communications (2017) 576–581 (2017). doi:10.1109/ISCC.2017.8024590.

S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, G. S. Paschos, The Algorithmic Aspects of Network Slicing, IEEE Communications Magazine 55 (8) (2017) 112–119 (2017). doi:10.1109/MCOM.2017.1600939.

A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, X. Hesselbach, Virtual network embedding: A survey, IEEE Communications Surveys and Tutorials 15 (4) (2013) 1888–1906 (2013). doi:10.1109/SURV.2013.013013.00155.

Y. Zhu, M. Ammar, Algorithms for assigning substrate network resources to virtual network components, Proceedings - IEEE INFOCOM (2006). doi:10.1109/INFOCOM.2006.322.

L. R. Bays, L. P. Gaspary, R. Ahmed, R. Boutaba, Virtual network embedding in software-defined networks, in: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, no. Noms, IEEE, 2016, pp. 10–18 (apr 2016). doi:10.1109/NOMS.2016.7502791.
URL http://ieeexplore.ieee.org/document/7502791/

C. Zhao, Z. Pu, Recent advances and trends in virtual network embedding, IEICE Transactions on Communications E99B (6) (2016) 1265–1274 (2016). doi:10.1587/transcom.2015EUP0001.

H. Cao, L. Yang, Z. Liu, M. Wu, Exact solutions of VNE: A survey, China Communications 13 (6) (2016) 48–62 (2016). doi:10.1109/CC.2016.7513202.

Online virtual links resource allocation in Software-Defined Networks, Proceedings of 2015 14th IFIP Networking Conference, IFIP Networking 2015 (2015).

L. Chen, S. Abdellatif, A. F. Simo Tegueu, T. Gayraud, Embedding and re-embedding of virtual links in software-defined multi-radio multi-channel multi-hop wireless networks, Computer Communications 145 (July) (2019) 161–175 (2019). doi:10.1016/j.comcom.2019.06.012.
URL https://doi.org/10.1016/j.comcom.2019.06.012

S. R. Chowdhury, R. Ahmed, N. Shahriar, A. Khan, R. Boutaba, J. Mitra, L. Liu, ReViNE: Reallocation of Virtual Network Embedding to eliminate substrate bottlenecks, Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management (2017) 116–124 (2017). doi:10.23919/INM.2017.7987271.

S. B. Masti, S. V. Raghavan, Simulated annealing algorithm for virtual network reconfiguration, 8th EURO-NF Conference on Next Generation Internet, NGI 2012 - Proceedings (i) (2012) 95–102 (2012). doi:10.1109/NGI.2012.6252170.

X. Foukas, G. Patounas, A. Elmokashfi, M. K. Marina, Network Slicing in 5G: Survey and Challenges, IEEE Communications Magazine 55 (5) (2017) 94–100 (2017). doi:10.1109/MCOM.2017.1600951.

P. Quinn, J. Guichard, Service function chaining: Creating a service plane via network service headers, Computer 47 (11) (2014) 38–44 (2014). doi:10.1109/MC.2014.328.

R. Vilalta, A. Mayoral, R. Casellas, R. Martínez, R. Muñoz, Sdn/nfv orchestration of multi-technology and multi-domain networks in cloud/fog architectures for 5g services, in: OptoElectronics and Communications Conference (OECC) held jointly with 2016 International Conference on Photonics in Switching (PS), 2016 21st, IEEE, 2016, pp. 1–3 (2016).

N. Bizanis, F. A. Kuipers, Sdn and virtualization solutions for the internet of things: A survey, IEEE Access 4 (2016) 5591–5606 (2016).

A. Manzalini, C. Buyukkoc, P. CHEMOUIL, S. Kuklinski, F. Callegati, A. Galis, M.-P. Odini, C.-L. I, J. Huang, M. Bursell, N. Crespi, E. Healy, S. Sharrock, Towards 5G software-defined ecosystems, Research report (2016).

R. Braden, D. Clark, S. Shenker, J. Wroclawski, Developing a next-generation internet architecture, White paper, DARPA (2000).

S. Paul, J. Pan, R. Jain, Architectures for the future networks and the next generation Internet: A survey, Computer Communications 34 (1) (2011) 2–42 (2011).

S. Wee, The next generation of networked experiences, in: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), Vol. 57, IEEE, 2014, pp. 29–35 (feb 2014). doi:10.1109/ISSCC.2014.6757326.

Demystifying network slicing: From theory to practice, Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management (2017) 1115–1120 (2017). doi:10.23919/INM.2017.7987450.

F. Ieee, C. E. Rothenberg, M. Ieee, S. Azodolmolky, S. M. Ieee, S. Uhlig, M. Ieee, Software-Defined Networking : A Comprehensive Survey, Proceedings of the IEEE 103 (1) (2015) 14 – 76 (2015). arXiv:1406.0440, doi:10.1109/JPROC.2014.2371999.
URL https://bit.ly/2WEpeal

NGMN Alliance, NGMN 5G White Paper, Tech. rep. (2015).
URL https://bit.ly/36mlCy0

Ant colony optimization, IEEE Computational Intelligence Magazine 1 (4) (2006) 28–39 (nov 2006). doi:10.1109/MCI.2006.329691.
URL http://ieeexplore.ieee.org/document/4129846/

H. Cao, H. Hu, Z. Qu, L. Yang, Heuristic solutions of virtual network embedding: A survey, China Communications 15 (3) (2018) 186–219 (2018). doi:10.1109/CC.2018.8332001.

D. E. Moriarty, A. C. Schultz, J. J. Grefenstette, Evolutionary Algorithms for Reinforcement Learning, Journal of Artificial Intelligence Research (1999). arXiv:1106.0221, doi:10.1613/jair.613.

B. Han, J. Lianghai, H. D. Schotten, Slice as an Evolutionary Service: Genetic Optimization for Inter-Slice Resource Management in 5G Networks, IEEE Access 6 (c) (2018) 33137–33147 (2018). arXiv:1802.04491, doi:10.1109/ACCESS.2018.2846543.

I. Fajjari, N. Ait Saadi, G. Pujolle, H. Zimmermann, VNE-AC: Virtual Network Embedding Algorithm Based on Ant Colony Metaheuristic, in: 2011 IEEE International Conference on Communications (ICC), IEEE, 2011, pp. 1–6 (jun 2011). doi:10.1109/icc.2011.5963442.
URL http://ieeexplore.ieee.org/document/5963442/

Rethinking virtual network embedding in reconfigurable networks, 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking, (2018) 1–9 (2018).

Z. Qin, G. Denker, C. Giannelli, P. Bellavista, N. Venkatasubramanian, IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined Worlddoi:10.1109/NOMS.2014.6838365.

S. Boccaletti, G. Bianconi, R. Criado, C. I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, M. Zanin, The structure and dynamics of multilayer networks, Physics Reports 544 (1) (2014) 1–122 (2014). arXiv:1407.0742, doi:10.1016/j.physrep.2014.07.001.
URL http://arxiv.org/abs/1407.0742

W. Guan, X. Wen, L. Wang, Z. Lu, Y. Shen, A service-oriented deployment policy of end-to-end network slicing based on complex network theory, IEEE Access 6 (c) (2018) 19691–19701 (2018). doi:10.1109/ACCESS.2018.2822398.

X. Yu, Y. Xue, Smart Grids: A Cyber-Physical Systems Perspective, Proceedings of the IEEE 104 (5) (2016) 1058–1070 (2016). doi:10.1109/JPROC.2015.2503119.

R. Morabito, N. Beijar, A Framework based on SDN and Containers for Dynamic Service Chains on IoT Gateways, Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems - HotConNet '17 (May) (2017) 42–47 (2017). doi:10.1145/3094405.3094413.
URL http://dl.acm.org/citation.cfm?doid=3094405.3094413

A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, T. Magedanz, Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges, IEEE Communications Magazine 55 (2) (2017) 216–223 (2017). doi:10.1109/MCOM.2016.1600219RP.

M. Yu, Y. Yi, J. Rexford, M. Chiang, Rethinking virtual network embedding: Substrate support for path splitting and migration, Computer Communication Review 38 (2) (2008) 19–29 (2008). doi:10.1145/1355734.1355737.

M. Mechtri, I. Houidi, W. Louati, D. Zeghlache, SDN for inter cloud networking, SDN4FNS 2013 - 2013 Workshop on Software Defined Networks for Future Networks

and Services (2013) 1–7 (nov 2013). doi:10.1109/SDN4FNS.2013.6702552.
URL https://bit.ly/3352EtA

Capítulo

# 6

# Processamento de Dados do *Twitter* para Monitoramento e Recomendação de Rotas em Cenários de Desastres Naturais

Mateus P. Garcia, Orrana L. V. de Sousa, Ana C. de A. Alves, Deborah M.V. Magalhães

***Abstract***

*Crowd behavior analysis in the natural disaster scenario plays an important role in disaster management as well as evacuation planning. In this context, social media play an important role as they provide real-time spatiotemporal data to guide evacuation policies. Extracting spatiotemporal data from social media such as time, date, and location is a complex task. In addition, a large amount of unstructured data needs to be filtered and structured in order to communicate useful information. This chapter will present the collection of a georeferenced database from the Twitter Application Programming Interface (API), as well as address the processing and analysis of acquired data, then evacuation routes will be recommended through simulations.*

***Resumo***

*A análise do comportamento de multidões no cenário de catástrofes naturais, exerce um papel importante no gerenciamento de desastres, bem como no planejamento da evacuação. Nesse contexto, as mídias sociais assumem um papel importante, pois fornecem dados espaço-temporais em tempo real para orientar políticas de evacuação. A extração de dados espaço-temporais a partir de mídias sociais, tais como, hora, data e localização, é uma tarefa complexa. Além disso, um grande volume de dados não estruturados precisa ser filtrado e estruturado de modo a comunicar informação útil. Nesse capítulo, será apresentada a coleta de uma base de dados georeferenciada proveniente do Twitter através de uma Interface de programação de aplicações (API), além de abordar etapas de processamento e análise dos dados adquiridos, em seguida rotas de evacuação serão recomendadas através de um sistema de simulação.*

## 6.1. Introdução

As mídias sociais são uma grande e rica fonte de dados, tais como textos, vídeos, imagens e dados de geolocalização, sendo reconhecidas como um dos principais canais de comunicação em situações de crises e desastres [Anbalagan e Valliyammai 2016]. A extração de dados espaço-temporais em cenários de desastres naturais é utilizada para analisar como as pessoas respondem ao desastre. Essas informações são imprescindíveis para a construção de políticas de planejamento de evacuação cujo fim é minimizar perdas de recursos humanos e financeiros.

Dessa forma, foram aplicadas técnicas de Processamento de Linguagem Natural (PNL)[Wibowo 2017], a fim de normalizar o texto contido no conjunto de dados. Além da análise de texto que fornecem métodos para encontrar padrões ou interpretar as informações [Arianto et al. 2018]. Nesse contexto, plataformas como o *Twitter* assumem um papel importante, pois permitem que usuários postem e atualizem opiniões, sentimentos e observações em tempo real. Esse conteúdo pode incluir informações espaço-temporais associadas a um evento de interesse [Hernandez-Suarez et al. 2019]. A mineração desses dados é eficaz no gerenciamento de emergências, incluindo realocação de operações de resgate, recomendação de rotas de fuga e distribuição de mantimentos e ajuda médica [Sadhukhan et al. 2018].

Diante disso, será apresentado o processo de mineração de dados georreferenciados utilizando a rede social *Twitter*. Esses dados possuem informações coletadas durante a ocorrência de um desastre natural no Havaí, devido à abalos sísmicos seguidos pela erupção do vulcão Kilauea em 2018. Além disso, iremos apresentar a modelagem e simulação desse cenário para recomendação de uma rota de evacuação a fim de apresentar a aplicabilidade prática dos dados coletados.

### 6.1.1. Estudo de caso

O estudo de caso ocorreu no Havaí como resultado de fissuras na lateral do sul do vulcão Kilauea, na zona leste do rifte, no qual consiste em grandes fraturas tectônicas, por onde escorre a lava, causando a abertura do solo. O terremoto e erupção do vulcão tiveram início no dia 3 de maio de 2018 e o vulcão permaneceu ativo até meados do mês de julho de 2018 [1].

A abertura do solo, a uma profundidade de 50 metros, foi contínua e esteve associada à atividade sísmica [Bulletin of Volcanology Editor-in-Chief 2018]. No dia 4 de maio a magnitude do terremoto chegou 6,9 graus na escala Richter na região próxima ao vulcão Kilauea. A escala Richter é usada por sismólogos para medir a energia sísmica liberada pelos tremores da terra. A escala Richter vai de 1 à 9, onde, o tremor igual ou inferior que 3,5 não é sentido, mas pode ser registrado.

Observado a Figura 6.1, é visto que a mancha vermelha representa o fluxo de lava que cobriu o Leste do Havaí, especialmente nas cidades de Kapoho que é um distrito de Puna e Leilani States que é uma região censo designada do estado do Havaí. As áreas afetadas já possuem histórico de fluxo de larva em 1840, 1955 e 1960. Portanto, a

[1] Por que o vulcão Kilauea continua em erupção no Havaí e os cientistas não sabem até quando vai durar: https://www.bbc.com/portuguese/geral-44884334

ocorrência de desastres naturais é recorrente nessa região, motivando o desenvolvimento deste capítulo.

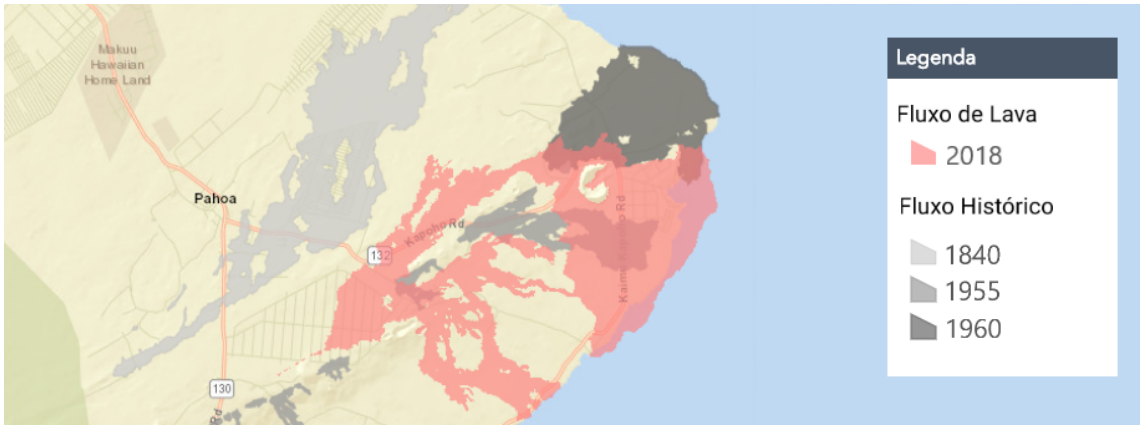



Figura 6.1: Fluxo de lava (em vermelho) durante a erupção do vulcão Kilauea em 2018 e o histórico do fluxo em anos anteriores destacado em escala de cinza. Modificado de: <https://hawaiicountygis.maps.arcgis.com>

## 6.2. Processamento de dados

Neste trabalho, nós coletamos, processamos e visualizamos informações como, geolocalização, horário e texto, pois são essências para o gerenciamento de desastres. A Figura 6.2 ilustra as etapas que vão desde a aquisição dos dados através da API do *Twitter* até a visualização dos mesmos, passando pelas etapas de limpeza e filtragem. Todas essas etapas serão detalhadas e demonstradas nas subseções a seguir.

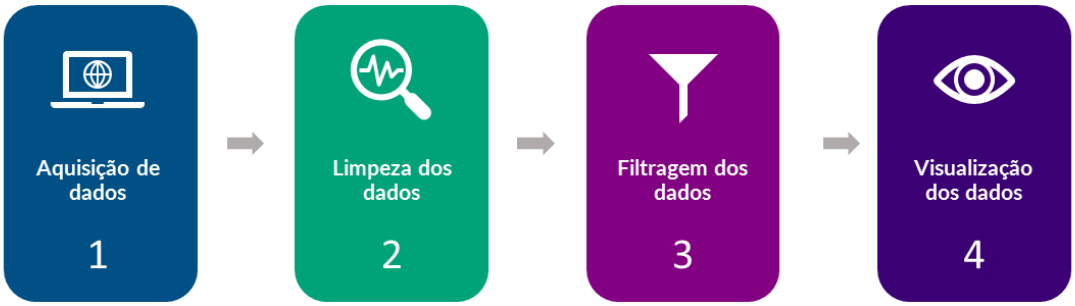



Figura 6.2: Fluxo de processamento dos dados

Uma quantidade imensa de dados é gerada diariamente nas mídias sociais, contudo, essa vasta quantidade é inútil sem qualquer método rápido e confiável para processá-los [Beniwal et al. 2018]. Assim, a extração de informações relevantes tem se tornado uma preocupação crescente. Para extrair informações significativas de dados não estruturados ou semiestruturados, técnicas de mineração de dados são amplamente utilizadas [Beniwal et al. 2018]. Os dados não estruturados não possuem padrões fixos. Os dados semi-estruturados não tem uma estrutura rígida, mas não são totalmente desestruturados. A mineração de dados visa construir uma base de conhecimento que contém informações úteis.

A disponibilidade de mídias sociais baseadas em localização com dados de redes sociais geomarcados, que significa que são dados com marcação geográfica, eles melhoraram significativamente a pesquisa e a prática de gestão de emergências, através do monitoramento de desastres e da reação das pessoas [Martín et al. 2017]. Além disso, pesquisas usando dados de redes sociais geomarcados e a identificação de padrões espaciais dos seus usuários são utilizadas em cenários de terremotos, incêndios florestais, ciclones tropicais, eventos de inundação, dentre outros. Portanto, o processo que estamos apresentando neste capítulo não se restringe à um tipo de cenário de desastre específico. E, pode ser extrapolado para outras áreas, como análise de sentimentos, por exemplo. Nessa plataforma há publicações constantes, possibilitando que os usuários compartilhem opiniões, sentimentos e observações [Hernandez-Suarez et al. 2019].

### 6.2.1. Aquisição dos dados

A aquisição de dados é uma etapa de grande importância e exige atenção aos requisitos para se obter dados confiáveis, como por exemplo: fonte dos dados, qualidade e formato. É comum nos depararmos com fontes que disponibilizam dados não estruturados e em diferentes formatos, então é necessário atenção para eliminar etapas de estruturação dos dados, o que pode levar bastante tempo. O *Twitter* oferece uma *API* de fácil acesso e com quantidade significativa de dados disponíveis, porém é importante ressaltar que os usuários que utilizam *Twitter* com geolocalização ativa é baixa, porém ainda assim bastante significativa. A fim de capturar os dados, é necessário definirmos alguns parâmetros importantes como: latitude, longitude, distância. Levando em conta esses parâmetros, podemos construir um filtro, onde somente *tweets* provenientes da área especificada do desastre posam ser coletados. A Figura 6.3 apresenta a área selecionada de acordo com os parâmetros: latitude: 19.420120, longitude: 155.253137 e distância: 200 quilômetros.


Figura 6.3: Área de coleta dos dados. Baseado no modelo: <http://bit.ly/31SypVv>

#### 6.2.1.1. Código fonte para coleta

O código fonte 6.1 é responsável pela coleta dos dados e a gravação em disco no formato *Comma-Separated Values* (CSV), amplamente utilizado para organizar dados. Neste código, nós fazemos uso da biblioteca *Tweepy*[2] para conexão com a *API* do *Twitter*. Todos os códigos apresentados neste trabalho podem ser acessados no repositório público em: `https://github.com/mpgxc/XSINFO-ICV.Disaster`, nele está contido todos os detalhes de implementação e arquivos de códigos exemplos extras.

```
import tweepy, json
#Lê as Tokens de acesso ao Twitter API
ENVS_KEYS = json.loads(
    open('ENVS.json', 'r').read()
)
#Autenticando Tokens
api = tweepy.API(
    tweepy.AppAuthHandler(
        ENVS_KEYS['API_KEY'],
        ENVS_KEYS['API_SECRET']
    ),
    wait_on_rate_limit=True,  wait_on_rate_limit_notify=True
)
#Pega o id do último tweet
def get_last_id(url_path):
    try:return list(reversed(
            [json.loads(x)['id'] for x in open(url_path, 'r')]
        ))[0]
    except:return None
if __name__ == "__main__":
    #Configurações gerais
    URL_PATH = "DATA.json"
    RESULTS_GEO = RESULT_ALL = RESULT_TWEETS = 0
    DATA_OUTPUT = open(URL_PATH, "a")
    LAST_ID = get_last_id(URL_PATH)
    GEO_PARAMS = "%f,%f,%dkm" % (
        ENVS_KEYS['LATITUDE'],
        ENVS_KEYS['LONGITUDE'],
        ENVS_KEYS['MAX_RANGE']
    )
    while True:
        try:
            for x in api.search(geocode=GEO_PARAMS,
                                count=100,
                                max_id=LAST_ID,
                                tweet_mode='extended'):
                if x.geo:
                    DATA_OUTPUT.write(str(json.dumps(x._json)) + "\n")
                LAST_ID = x.id
        except:
            print("\n Limite da API atingido!")
```

Código Fonte 6.1: Script de coleta conectando via API do *Twitter*

[2]**Tweepy**: Twitter for Python! Disponível em: <`http://docs.tweepy.org/en/latest/`>

#### 6.2.1.2. Autenticação

O código fonte 6.1 utiliza a biblioteca *tweepy* para conectar-se especificamente à *Standard search API*[3], da qual destacamos alguns aspectos importantes:

- Formato da resposta *JSON* (JavaScript Object Notation);
- Requer Autenticação;
- 180 solicitações a cada 15 minutos (autenticação por usuário);
- 450 solicitações a cada 15 minutos (autenticação por aplicativo);
- Cada requisição retorna um total máximo de 100 *tweets*;

Como pré requisito para execução do código 6.1, nós temos a criação de *APP* dentro da plataforma *Twitter Developer*[4], para todo *APP* criado dentro da plataforma são atribuídos *tokens* e *keys* de autenticação via *API Rest*[5]. Os *tokens* e *keys* são apresentadas da seguinte forma:

- Consumer API keys (autenticação por aplicativo);
- Access tokens + Consumer API keys (autenticação por usuário);

As principais diferenças dos modos de autenticação se dão pelo fato da autenticação por usuário permitir acesso direto aos dados do usuário da conta de desenvolvedor, tendo acesso as mensagens privadas, *timeline*, e-mails cadastrados, números de telefone e outros dados sensíveis, porém como foi visto, essa opção também tem maiores limitações de requisições à *API*. Para este trabalho não se faz necessário o acesso direto a esse tipo de dados privados, então será utilizado a autenticação por aplicativo que oferece uma maior cobertura, dado o seu limite superior de requisições à API, as diferenças de acesso podem ser observados seguir:

- Autenticação por usuário: 180 *requests* * 100 *tweets* = 18.000 *tweets*;
- Autenticação por aplicativo: 450 *requests* * 100 *tweets* = 45.000 *tweets*;

#### 6.2.1.3. Seleção de atributos

A seleção dos atributos permite reduzir a quantidade de dados coletados, evitando atributos irrelevantes e melhorando a compreensão dos dados. Para isso, deve ser utilizado o código fonte 6.2, que de maneira simples efetua a leitura dos dados coletados e os

[3]**Standard search API**. Disponível em: <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html>

[4]**Twitter Developer**. Disponível em: <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html>

[5]**What is REST**. Disponível em: < https://restfulapi.net/>

transcreve para um novo arquivo contendo apenas os atributos definidos no código (ver o código 6.2). Exemplo de atributos disponíveis: *id, username, time, coordinates* . Para executar o exemplo abaixo: ./parser.py nome-da-base-origem.json

```
import time
import csv
import json
import sys
import pandas as pd

data_output_json = open("DATA_FINAL.json", 'w')

if __name__ == "__main__":

    for tweet in open(sys.argv[1], "r"):

        item = json.loads(tweet)
        # pegando todas as hashtags sem o caractere '#'
        hashtags = (
            hash_word['text']
            for hash_word in item['entities']['hashtags']
        )
        lat, long = item['geo']['coordinates']

        # formatando date e time
        tweet_date,
        tweet_time = time.strftime('%Y-%m-%d %H:%M:%S', time.strptime(
            item['created_at'], '%a %b %d %H:%M:%S +0000 %Y'
        )).split(' ')

        data_output_json.write(
            str(json.dumps({
                'user_id': item['user']['id_str'],
                'screen_name': item['user']['screen_name'],
                'location': item['user']['location'],
                'followers_count': item['user']['followers_count'],
                'lang': item['lang'],
                'text': item['full_text'].replace('\n', ''),
                'date': tweet_date,
                'time': tweet_time,
                'hashtags': ' '.join(hashtags),
                'source_device': item['source'],
                'latitude': lat,
                'longitude': long
            })) + '\n'
        )
    data_output_json.close()
    dataframe_json = pd.read_json("DATA_FINAL.json", lines=True)
    dataframe_json.to_csv("DATA_FINAL.csv", index=None)
```

Código Fonte 6.2: Script para selecionar atributos específicos

### 6.2.2. Limpeza dos dados

Após a coleta dos dados foram aplicadas técnicas de Processamento de Linguagem Natural (NLP) a fim de normalizar o texto dos *tweets*. A técnicas utilizadas neste processo para remoção de ruídos foram: Tokenização, Remoção de *Stopwords* e a remoção de outros ruídos como números, pontuações e caracteres especiais. Inicialmente é aplicado o processo de *Tokenização* no qual consiste em decompor a mensagem inteira em termos/palavras, a fim de facilitar etapas futuras, como aferir a frequência de cada palavra do conjunto. Seguindo, é executado a remoção de números, caracteres especiais e pontuações, que são ruídos que não oferecem valor para o trabalho. Por fim, é executado a remoção de *Stopwords* ou palavras de parada que servem para compor uma frase lingando uma palavra a uma outra próxima, na língua portuguesa são exemplos palavras de parada: as, e, os, de, para, com, sem, foi; para o NLP *Stopwords* não oferecem valor ou sentido na análise de um texto. Segue o exemplo na tabela 6.1 de como fica um texto passado pelo processo de limpeza

| | **Processo de limpeza de um texto** |
|---|---|
| Antes | Monday-January-2018::15:14: An earthquake of magnitude 4.280000 shook 33km ESE of Pahala, Hawaii at a depth of 12.6 |
| Depois | monday january an earthquake magnitude shook km ese pahala hawaii depth |

Tabela 6.1: Processo de limpeza, antes e depois dos filtros.

O código fonte 6.3 responsável por todo processo de limpeza, contendo três funções com todos os pssos citados acima.

```
def remove_numbers(text):
    return re.sub(r'\d+', '', text)

def remove_dots(text):
    return  re.compile(f'[{re.escape(string.punctuation)}]').sub(' ', text)

def remove_stopwords(text):
    stop_words = set(stopwords.words('english'))
    make_tokenize    = TweetTokenizer()
    tokens  = make_tokenize.tokenize( remove_dots( remove_numbers(text) ) )
    return [w.lower() for w in tokens if not w in stop_words]
```

Código Fonte 6.3: Funções para as etapas de limpeza do texto

### 6.2.3. Filtragem dos dados

Esta fase conta com a construção de um dicionário de *keywords* para verificar a ocorrência das palavras nos tweets e assim filtrar somente conteúdo relacionado ao desastre. O dicionário foi construído inspirado no trabalho [Murthy e Gross 2017], onde um dicionário foi organizado classificando as palavras relativas ao desastre por categorias. Nós adaptamos tal dicionário conforme as especificidades deste trabalho. O dicionário foi organizado nas seguintes categorias: *Earthquake*; *Volcano*; *Locations*; *Weather*; *Emotion*. Cada categoria possui suas respectivas palavras-chave que possuem alta correlação com o evento de erupção do vulcão Kilauea.

abandon, active, active_fault, administration, administrators, adversity, afraid, aftershocks, aghast, aid, airfall, aliment, alimentchow, allowance, amplitude, andesite, annihilated, anxious, apartment, appeal, apprehensive, ash, ashes, ashfall, assistance, asthenosphere, asylum, bail_out, basalt, begging, bench, block, bomb, broken, calamity, caldera, careful, cataclysm, catastrophe, celsius, central_vent, central_volcano, charity, chow, cinder, cinder_cone, cinder kipuka, clearance, clinic, comestible, concerned, concerns, condo, conduit, cone, contribution, convulsion, coulee, crater, craving, crisis, crowd, dacite, danger, death, deformation, demolished, destroyed, devastated, disasters, difficulty, direction, directors, disaster, disintegrated, displace, distress, distressed, disturbed, donations, dwelling, earthquake, earthquake_fault, ejecta, emergency, emission, emptying, endowment, entreaty, epicentre, erupt, erupting, eruption, eruptions, erupts, evacuee, evacuate, evacuation, evacuee, exigency, extremity, fahrenheit, fallout, fault, feed, fidgety, fire, fissures, flow, fog, foodstuff, forsake, fumarole, gas, heavy, help, holocene, hut, imploring, insecurity, instability, institution, jittery, kilauea, killed, kipuka, lahar, larder, lava, laze, leadership, leave, leilani, lodging, lost, lua, luapele, macroseism, magma, magnitude, mainshock, meal, mess, microseism, migration, misadventure, mischance, misfortune, mishap, monogenetic, mount, move_out, movement, nervous, nervy, nourishment, nutriment, organization, orison, pahalacloud, pahoehoe, pali, path, paths, pele, peril, perilousness, petition, phreatomagmatic, pleading, plight, plug, prayers, probability, problem, pull_out, pumice, pyroclastic, quake, quaker, quit, ravaged, razed, refreshment, refuge, reliefsubsidy, removal, remove, repose, request, restless, rhyolite, risk, riskiness, road, route, ruined, safe, safety, sanctuary, scared, scoria, seismicity, seism, seismicity, seismism, seismograms, seismograph, seismology, service, shake, shaky, shattered, shelter, shelters, shifting, shock, shook, seismic, sismo, situation, slag, slip, smashed, smoke, solfatara, spooked, steaming, store, subsidy, subsistence, supervision, support, surface, surge, survive, sustenance, tears, tectonic, tectonics, temblor, tephra, terremoto, tilt, tragedy, transport, trembler, tremor, trouble, tuff, uneasy, uptight, urgent, vapor, vents, vents_steaming, volcanic, volcano, volcanoes, volcanowatch, vulcan, vulcanian, volcanoes, vulcao, vulcão, wasted, withdraw,

Figura 6.4: Dicionário de palavras no formato de entrada para filtragem de conteúdo relacionado

```
def read_words():
    arq = open('path_dic_words.txt').readlines()
    text = " ".join(arq).replace(',',' ').lower().split(' ')
    return sorted(set([x for x in text if len(x) > 1]))
```

Código Fonte 6.4: Código fonte para leitura do dicionário de palavras.

#### 6.2.3.1. Buscando conteúdo relacionado

Utilizando das *Keywords* do dicionário de palavras o código fonte 6.5 faz a intersecção entre o dicionário de palavras e o conjunto de dados coletados, então obedecendo a regra, temos que para todo dado do conjunto de *Keywords* tendo referência no conjunto de dados coletados, essa informação é considerada válida, ou seja é um dado relevante.

```
data['interest'] = 0
for index in data.index:
    text = data['text'][index]
    result = pd.Index(str(text).split()).intersection(pd.Index(words))
    if len(result) > 0:
        data.loc[index, 'interest'] = 1
```

Código Fonte 6.5: Código fonte para verificar a interseção entre dicionário de palavras e o conjunto de dados coletados.

### 6.2.4. Visualização dos dados

Após a filtragem dos *tweets* é possível plotar imagens referentes ao dados de forma a comunicar as informações obtidas. Dessa forma, foi gerado um mapa de calor a partir da geolocalização dos usuários que publicaram no *Twitter* durante a ocorrência do desastre no Havaí. Conforme visto na Figura 6.5, o mapa da esquerda representa a densidade de postagens no *Twitter* o que corresponde aos locais com maior concentração de pessoas. Isso é corroborado pelo mapa da direita que mostra que esses locais também apresentam maior infraestrutura.

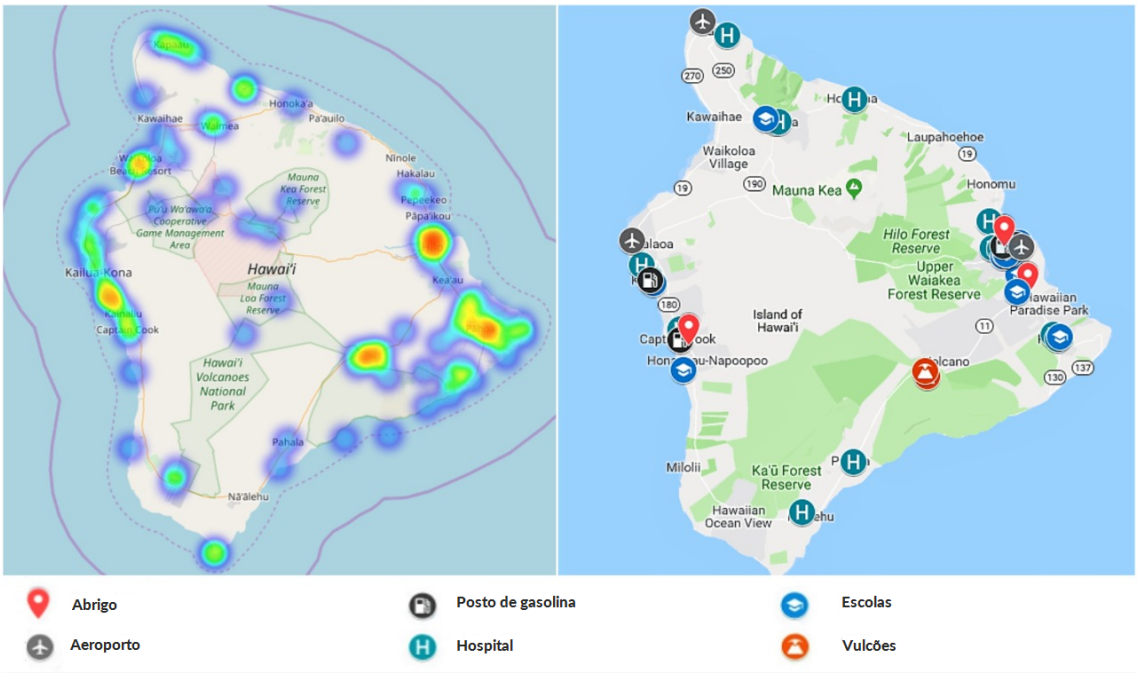



Figura 6.5: Mapa de calor ilustrando a concentração de postagens no *Twitter* na área especificada; localização dos locais comumente buscados durante um desastre.

A partir dos dados coletados, é possível realizar uma contagem da quantidade de *tweets* que foram postados por dia. Na Figura 6.6 podemos observar que os dias com maior quantidade de postagem no Havaí foram entre os dias 1 de maio e 9 de maio.

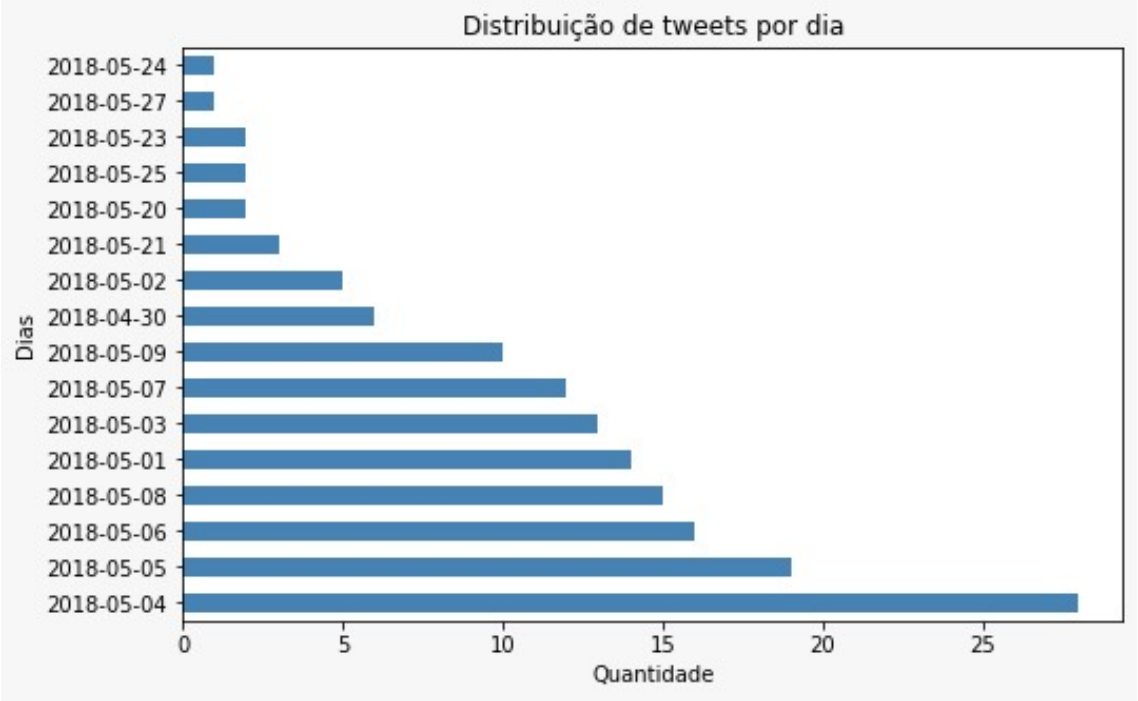



Figura 6.6: Distribuição dos *tweets* por dia.

Podemos também, realizar uma contagem das palavras mais publicadas por dia. Conforme visto na Figura 6.7 foram montados grupos de palavras referentes a datas determinadas pelo próprio algoritmo, no qual foi determinada a frequência das palavras.

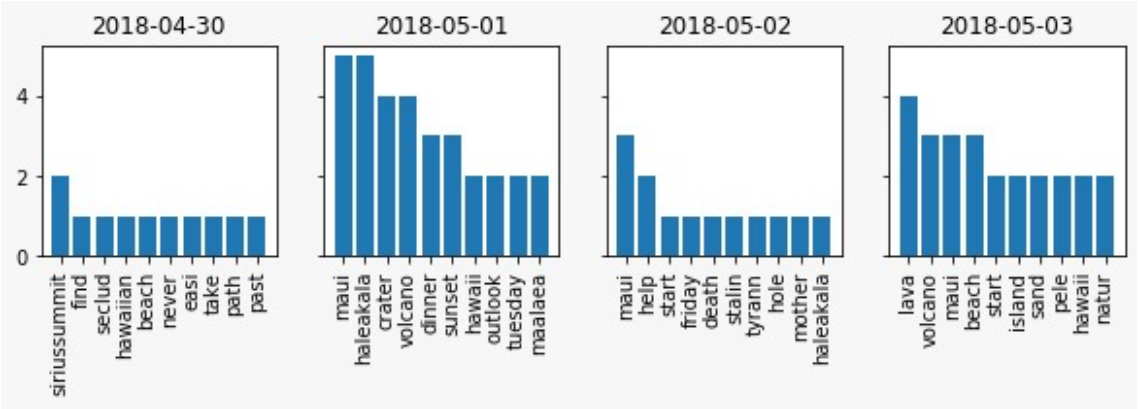



Figura 6.7: Palavras mais utilizadas por dia.

É possível, também, plotar as palavras que foram mais publicadas no *Twitter* durante a ocorrência do desastre. Conforme visto na 6.8, há algumas palavras do dialeto havaiano que aparecem nos *tweets*. Tais palavras foram adicionadas no dicionário pois são relacionadas ao desastre e consequentemente significativas para a base de dados. A palavra *Maui* que aparece em quarto lugar no gráfico é referente à segunda maior ilha do Havaí. A palavra *Kilauea* é referente à cratera vulcânica que entrou em erupção durante o desastre. A palavra *Pele*, que aparece em sétima posição, significa vulcão no dialeto

havaiano. *Leilani* é uma área do Havaí, que fica localizada na zona de risco e que foi bastante afetada pela erupção do vulcão. *Haleakala* é um vulcão que fica localizado em Maui e consiste na mais alta montanha da ilha. A palavra *Hana* é referente a uma região localizada em Maui.

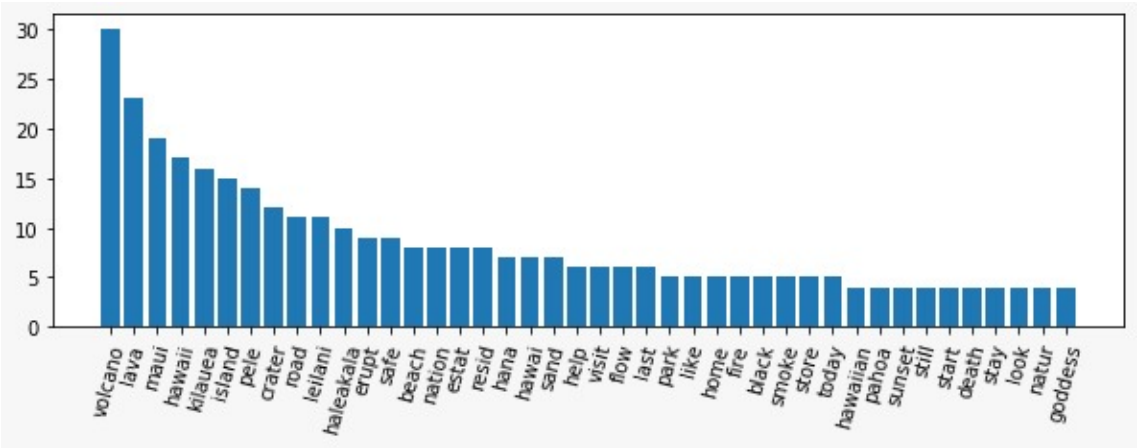



Figura 6.8: Palavras mais frequentes nos dados coletados.

## 6.3. Simulação

A simulação de cenários de desastre traz benefícios à pesquisadores e desenvolvedores na área, pois permite: (1) avaliação de políticas de evacuação antes de ser colocada em prática em um ambiente real e em larga escala e (2) a simulação de cenários realísticos em um ambiente controlado, onde é possível realizar modificações, repetições e adicionar novas características ao cenário.

### 6.3.1. *MATSim*

*MATSim*[6] é uma ferramenta que oferece uma estrutura para realizar simulações de transporte baseadas em agentes (entidades individuais ou coletivas, como organizações ou grupos) em larga escala. Cada agente (indivíduo) otimiza repetidas vezes seu cronograma diário de atividades ao mesmo tempo que compartilha espaços na infraestrutura de transporte com os demais agentes ao longo da simulação. Tal cronograma representa uma lista de atividades, como, por exemplo, estar em *casa* ou no *trabalho* e se deslocar de *carro* até o shopping). O cronograma conta ainda com com informações temporais, por exemplo, sair de casa às *7:00*, e informações adicionais, como a rota detalhada de casa pro shopping.

Em uma simulação típica do *MATSim*, os dados referentes a trajetória a qual se deseja simular são otimizados em uma rede de transporte definida, esse é um dos recursos fundamentais do simulador, tornando-o adequado para uso em estudos de políticas de evacuação. Durante o processo de simulação e otimização, cinco etapas principais são identificadas, conforme ilustrado na Figura 6.9: demanda inicial, execução, pontuação, replanejamento e análise.

[6]**MATSim**. Disponível em: <`https://matsim.org/`>

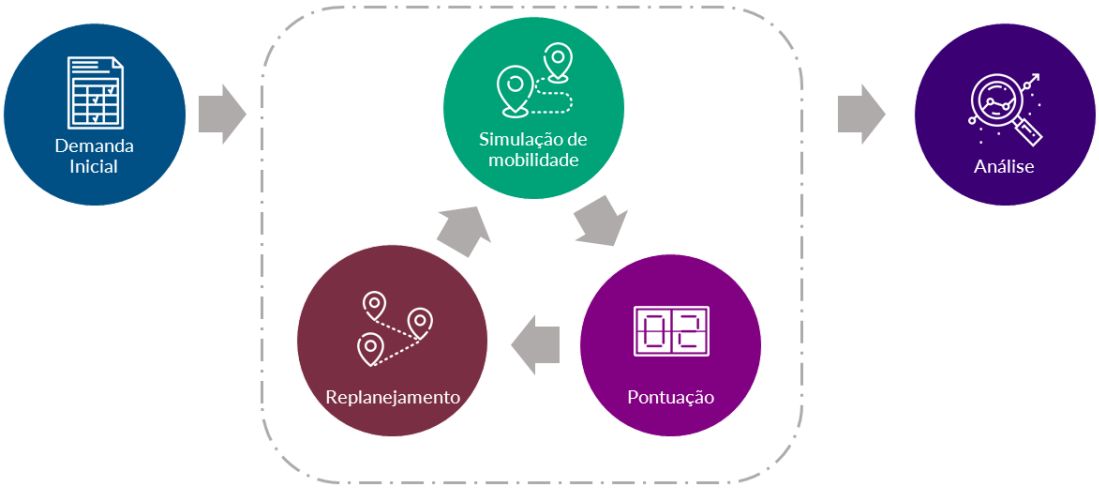



Figura 6.9: O ciclo de vida da simulação no *MATSim*. Adaptado de [Andreas Horni 2019]

Na primeira etapa, a *Demanda* inicial descreve o comportamento de mobilidade a ser simulado. Abrange uma lista completa de agentes com seus respectivos cronogramas diários. Na etapa de execução, também conhecida como *Simulação de Mobilidade* (*mobsim*) todos os planos dos agentes são executados juntos. Isto é, agentes e veículos são simulados na rede rodoviária do cenário. Desse modo, um agente pode impactar em outros agentes, podendo ocasionar, por exemplo, engarrafamentos se vários agentes decidem transitar pela mesma estrada.

Após a execução, dá-se início a etapa de *Pontuação*, onde cada plano é avaliado e recebe um determinado número de pontos. A função de pontuação é customizável mas, por padrão, o tempo gasto nas atividades aumenta a pontuação, enquanto o tempo gasto na viagem o diminui. Desse modo, agentes em engarrafamentos perdem pontos, enquanto agentes que têm viagens rápidas e curtas acumulam mais pontos, executando atividades por um maior período.

Como discutido anteriormente, agentes podem ser influenciados por outros, provocando congestionamentos em vias. Portanto, na etapa de *Planejamento*, os agentes podem modificar seus planos visando melhorar as suas respectivas pontuações. Dessa forma, horários de inicio e fim de atividades podem ser alterados na simulação de mobilidade a fim de evitar situações que levem a pontuações ruins.

Na etapa de *Análise*, as ações ou eventos realizados na simulação são agregados para avaliar qualquer medida na resolução desejada, como compartilhamentos de modo (carro, táxi, bicicleta), total de milhas percorridas pelos agentes, duração média das viagens e distância por modo e hora. Tais análises podem ser executadas automaticamente no final ou em uma etapa separada qualquer de pós-processamento.

#### 6.3.1.1. Arquivos de entrada

Para realizar uma simulação, os seguintes arquivos precisam necessariamente serem configurados no *MATSim*:

- *config.xml*, descreve, o comportamento da simulação;
- *network.xml*, descreve a rede rodoviária da área a ser simulada;
- e *population.xml*, descreve as demandas de viagem de acordo com o cenário que se deseja simular.

O *MATSim* é configurado através do arquivo *config.xml*, que constrói uma ligação entre o simulador e o usuário, controlando como a simulação será feita. Cada parâmetro faz parte de exatamente um grupo de configurações, como, por exemplo, o parâmetro *inputNetworkFile* mostrado no Código 6.6. Esse grupo coleta parâmetros semelhantes ou diferentes para uma parte específica do *MATSim*. Configurações como sistema de coordenadas, número de iterações e tipos de atividades são definidas neste arquivo.

```
<config>
	<module name="network">
		<param name="inputNetworkFile" value="network.xml" />
	</module>
	<module name="plans">
		<param name="inputPlansFile" value="plans.xml" />
	</module>
	<module name="controler">
		<param name="firstIteration" value="0" />
		<param name="lastIteration" value="0" />
	</module>
	<module name="planCalcScore">
			<parameterset type="activityParams">
				<param name="activityType" value="h" /> <!-- home -->
				<param name="priority" value="1" />
				<param name="typicalDuration" value="12:00:00" />
				<param name="minimalDuration" value="08:00:00" />
			</parameterset>
		</parameterset>
	</module>
</config>
```

Código Fonte 6.6: Exemplo de arquivo de configuração

O arquivo *network.html* descreve a rede em que agentes ou veículos se locomovem, como apresentado no Código 6.7. Consiste de nós e *links*, ou vértices e arestas na teoria dos grafos. Cada nó é representado por valores de coordenadas x e y, enquanto *links* têm um número maior de características. Dentre essas características podem-se citar atributos como *from* e *to* que descrevem a geometria da rede. Além disso, todos os links são unidirecionais. Se uma via puder ser percorrida nas duas direções, dois links devem ser estipulados com os atributos *from* e *to* contrários.

```
<network>
	<nodes>
		<node id="0" x="-200.0" y="200.0" > </node>
		<node id="1" x="0.0" y="200.0" > </node>
		<node id="2" x="200.0" y="200.0" > </node>
		<node id="3" x="400.0" y="400.0" > </node>
```

```
      </nodes>
      <links capperiod="01:00:00" effectivecellsize="7.5" effectivelanewidth="3.75">
            <link id="0_1" from="0" to="1" length="200.0" freespeed="222.22222222222223"
            capacity="3600.0" permlanes="1.0" oneway="1" modes="car" >
            </link>
            <link id="1_2" from="1" to="2" length="200.0" freespeed="222.22222222222223"
            capacity="3600.0" permlanes="1.0" oneway="1" modes="car" >
            </link>
            <link id="2_3" from="2" to="3" length="200.0" freespeed="3.33889816360601"
            capacity="1800.0" permlanes="1.0" oneway="1" modes="car" >
            </link>
      </links>
</network>
```

Código Fonte 6.7: Exemplo de arquivo de *network*

A demanda de viagens do *MATSim* é descrita pelos planos diários dos agentes presentes no arquivo *population.xml*, como exposto no Código 6.8. Por isso, o arquivo de população contém essencialmente uma lista de planos diários em uma estrutura hierárquica. A população contêm uma lista de pessoas, cada pessoa tem uma lista de planos, e cada plano tem uma lista de atividades. Apenas um plano é escolhido. O plano selecionado de cada agente é executado pelo *mobsim*, e pode ser substituído por outro durante a fase de replanejamento.

```
<population>
   <person id="car_home1_work1_0">
            <plan selected="yes">
                  <activity type="home" x="-1188391.227582054" y="114456.09067770051"
                  end_time="09:00:00" > </activity>
                  <leg mode="car"> </leg>
                  <activity type="work" x="4663945.2848932855" y="0.0"
                  end_time="17:00:00" >
                  </activity>
                  <leg mode="car"> </leg>
                  <activity type="home" x="-1188391.227582054" y="114456.09067770051"
                  end_time="09:00:00" > </activity>
            </plan>
      </person>
</population>
```

Código Fonte 6.8: Exemplo de arquivo de população

### 6.3.2. Criação da rede *MATSim*

Como discutido anteriormente, um dos arquivos principais para a realização de uma simulação no *MATSim* é o *network.xml*. O método mais comum para a sua geração é através da exportação de dados do OpenStreetMap(OSM) [7], que disponibiliza dados geoespaciais abertos e detalhados. Através dessa ferramenta pode-se criar redes *MATSim* mais reais e acuradas, possibilitando uma melhor visualização da área do estudo de caso.

#### 6.3.2.1. Extração de dados do *OpenStreetMap*

O *OpenStreetMap* é um projeto de mapeamento colaborativo para criar um mapa livre e editável do mundo. É constituído por dados abertos: qualquer pessoa tem a liberdade de usá-los para qualquer fim. Ele fornece dados a centenas de *sites* na *Internet*,

[7] **Open Street Map**. Disponível em: <https://www.openstreetmap.org/about>

aplicações de celular e outros dispositivos. Os mapas foram desenvolvidos e são mantidos com rigor por sua comunidade voluntária de mapeadores, que inserem e revisam dados recebidos de fotografias aéreas, imagens de satélite e outras fontes livres.

A exportação de mapas deste projeto não é uma tarefa complicada. Existem diversos serviços da *Web* que possibilitam a extração de dados para uma área de interesse, tais como: *GeoFabrik*, *BBBike* e *Overpass Turbo*. Além destes, existem outros *softwares* livres, como o *Quantum GIS* e *JOSM*, que têm *plugins* que são utilizados para este fim. Contudo, a forma mais comum é através do próprio site do *OpenStreetMap*, é possível buscar pelo mapa ou região que você deseja capturar, em seguida você pode exportar o mapa e baixá-lo para sua máquina local, conforme ilustrado na Figura 6.10.

Na etapa de exportação, é importante destacar a diferença entre exportar mapas pequenos pequenos de mapas grandes. No caso de mapas de cidades, povoados e bairros, a exportação pode ser realizada diretamente do servidor do OpenStreetMap. Já para mapas grandes, onde a área é extensa, como condados, estados e países, utiliza-se um espelho do banco de dados do *OpenStreetMap*, a *Overpass API*[8]. Com isso, um arquivo .osm é gerado. Este arquivo é utilizado para salvar informações de mapas de ruas na forma de "nós"(pontos), "caminhos"(conexões) e "relações"(propriedades de ruas e objetos, como *tags*).


Figura 6.10: Fluxo de extração de dados do *OpenStreetMap*

### 6.3.2.2. Criação do arquivo *network.xml*

Foi desenvolvido o Código 6.9 para a conversão de dados *.osm* em arquivos compatíveis com o simulador *MATSim*. Tal *script* é responsável por criar arquivos XML e SHP que poderão ser utilizados tanto para a simulação quanto para a visualização do cenário abordado.

```
1  class NetworkGeneratorHawaii {
2
3     public static final String HAWAIICoord = "EPSG:2782";
4
5     public static void main(String [] args) throws SAXException {
6
7        String osm = "network.osm";
8        Scenario sc = ScenarioUtils.createScenario(ConfigUtils.createConfig());
9        Network net = sc.getNetwork();
```

[8]**Overpass API**. Dinponível em: <https://wiki.openstreetmap.org/wiki/Overpass_API>

```
11        CoordinateTransformation ct = TransformationFactory.getCoordinateTransformation(
          TransformationFactory.WGS84, HAWAIICoord);
13
          OsmNetworkReader onr = new OsmNetworkReader(net,ct);
15        onr.setHighwayDefaults(1, "null", 1, 60/3.6, 1, 600);
          onr.setHighwayDefaults(1, "service", 1, 60/3.6, 1, 600);
17        onr.parse(osm);

19        new NetworkCleaner().run(net);
          String filename = "network.xml";
21        new NetworkWriter(net).write(filename);

23        FeatureGeneratorBuilderImpl builder = new FeatureGeneratorBuilderImpl(net,
          HAWAIICoord);
25        builder.setWidthCoefficient(0.01);
          builder.setFeatureGeneratorPrototype(PolygonFeatureGenerator.class);
27        builder.setWidthCalculatorPrototype(CapacityBasedWidthCalculator.class);
          new Links2ESRIShape(net,"network.shp", builder).write();
29
      }
31  }
```

Código Fonte 6.9: Criando a rede *MATSim*

### 6.3.3. Criação da população no *MATSim*

Para que o ciclo *MATSim* seja realizado é necessária uma demanda inicial, como mostrado na Figura 6.9. Existem comumente duas possibilidades para a geração dessa demanda: a forma prática e a exigente. A forma mais prática é usar um censo populacional completo na criação da demanda, enquanto que a exigente é gerar uma população sintética, que pode ser baseada em pesquisas de amostra ou estrutura.

Neste estudo de caso foram criadas duas populações sintéticas. Uma antes e outra após o desastre, para fins de visualização. Essas populações foram criadas a partir de dados minerados do *Twitter* na ilha do Havaí, conforme o estudo de caso apresentado na Seção 6.1.1. Como resultado da mineração, um arquivo *.csv* foi gerado contendo dados como data, *id*, texto, hora, latitude e longitude, como exemplificado na Figura 6.11. Utilizando a data como fator decisivo, esse arquivo foi dividido em dois: um com dados até a data 04/05/2018 (data de evacuação) [9] e outro com dados após esta data.

Com essas informações, 145 agentes foram criados na primeira população e 141 na segunda. Os agentes receberam como identificadores os *ids* dos *tweets* minerados e tiveram a atividade *home* (casa) formulada utilizando os dados de latitude e longitude de cada *.csv*. Além disso, outra atividade foi gerada para essas populações: a atividade *shelter* (abrigo). A atividade *shelter* foi criada a partir de coordenadas de 11 abrigos em potencial da ilha do Havaí. Foi então desenvolvido um *script* com uma lista de coordenadas desses abrigos que é chamado sempre que a atividade *shelter* de um agente é gerada.

[9] **Centenas de pessoas são evacuadas após erupção de vulcão no Havaí**; EXAME. Disponível em: https://exame.abril.com.br/mundo/centenas-de-pessoas-sao-evacuadas-apos-erupcao-de-vulcao-no-havai/

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | date | id | text | time | latitude | longitude |
| 2 | 2018-05-09 | 48119119108 | ıg Kapa | 3:44:09 | 209967 | -156653 |
| 3 | 2018-05-03 | 21151091051161 | n Hawai'i for a fev | 1:25:05 | 196346908 | -1.5599E+16 |
| 4 | 2018-04-30 | 711997105105981 | irits Sansei Waik | 23:31:26 | 1991383842 | -15587940562 |
| 5 | 2018-05-08 | 711997105105981 | afternoon strawb | 20:13:31 | 19723 | -156005 |
| 6 | 2018-05-02 | 711997105105981 | shotguinness tull | 9:01:52 | 1992545 | -15588433 |
| 7 | 2018-05-08 | 112117114101108 | you ready to cha | 22:50:39 | 2.09131E+16 | -15669244 |
| 8 | 2018-05-05 | 112117114101108 | u won't value you | 8:19:23 | 208861 | -156675 |
| 9 | 2018-05-01 | 081211031111161 | Pic looking towar | 5:10:07 | 2.08869E+16 | -1564774079 |
| 10 | 2018-05-08 | 100117101991011 | at hopupmagazin | 20:32:09 | 2068377868 | -1564417521 |
| 11 | 2018-04-30 | 039710897120101 | rapped RingsPur | 1:20:31 | 1964188 | -15599315 |

Figura 6.11: Algumas linhas do arquivo *.csv*

#### 6.3.3.1. CSV

```
public static String[][] getCSVData(String csvFile, int i, int j) {

      BufferedReader br = null;
      String line = "";
      String cvsSplitBy = ",";
      String[] position = null;
      String person[][] = new String[i][j];

      try {

         br = new BufferedReader(new FileReader(csvFile));
         int p = 0;
         while ((line = br.readLine()) != null) {

            position = line.split(cvsSplitBy);
            person[p][0]=position[1]; //coluna de ids
            person[p][1]=position[5]; //coluna de longitudes
            person[p][2]=position[4]; //coluna de latitudes

            p++;
         }

      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      } finally {
         if (br != null) {
            try {
               br.close();
            } catch (IOException e) {
               e.printStackTrace();
            }
         }
      }

      return person;

}
```

Código Fonte 6.10: Método para obter os *ids* e coordenadas dos agentes

#### 6.3.3.2. *ShelterCoord*

```
public static Coord getCoord(CoordinateTransformation ct) {

      Coord shelterHCES = new Coord (-155.084807, 19.721082);
      Coord coordHCES = ct.transform(shelterHCES);

      Coord shelterHSH = new Coord (-155.090788, 19.723933);
      Coord coordHSH = ct.transform(shelterHSH);

      Coord shelterUN = new Coord (-155.98662, 19.63223);
      Coord coordUN = ct.transform(shelterUN);

      Coord shelterUHHAF = new Coord (-155.04953, 19.65319);
      Coord coordUHHAF = ct.transform(shelterUHHAF);

      Coord shelterPACRC = new Coord (-155.04791, 19.73122);
      Coord coordPACRC = ct.transform(shelterPACRC);

      Coord shelterHCC = new Coord (-156.022302, 19.810874);
      Coord coordHCC = ct.transform(shelterHCC);

      Coord shelterCCECS = new Coord (-155.079334, 19.703855);
      Coord coordCCECS = ct.transform(shelterCCECS);

      Coord shelterTDKICP = new Coord (-155.08957, 19.6977);
      Coord coordTDKICP = ct.transform(shelterTDKICP);

      Coord shelterUHH = new Coord (-155.08146, 19.70101);
      Coord coordUHH = ct.transform(shelterUHH);

      Coord shelterHPA = new Coord (-155.698639, 20.029981);
      Coord coordHPA = ct.transform(shelterHPA);

      Coord shelterHPALM = new Coord (-155.673607, 20.024664);
      Coord coordHPALM = ct.transform(shelterHPALM);
      List<Coord> list = new ArrayList<>();
      list.add(coordHCES);
      list.add(coordHSH);
      list.add(coordUN);
      list.add(coordUHHAF);
      list.add(coordPACRC);
      list.add(coordHCC);
      list.add(coordCCECS);
      list.add(coordTDKICP);
      list.add(coordUHH);
      list.add(coordHPA);
      list.add(coordHPALM);
      Random rand = new Random();
      return list.get(rand.nextInt(list.size()));
}
```

Código Fonte 6.11: Método com a lista de coordenadas dos abrigos

#### 6.3.3.3. Criação dos arquivos de população

Utilizando os Códigos 6.10 e 6.11, o Código 6.12 foi feito para criar as populações referentes aos dados até o dia 04/05/2018 e após ele. Este *script* cria a quantidade de agentes existentes em cada arquivo *.csv* com suas respectivas atividades. Cada agente possui uma atividade *home* e uma *shelter*. A atividade *home* começa às 00:00 e termina às 07:00 hrs ao passo que a atividade *shelter* começa às 08:00 e não tem horário de fim.

```
public class PopulationHawaii{
```

```
    private static final String HAWAIICoord = "EPSG:2782";
    private static final Logger log = Logger.getLogger(PopulationHawaii.class);
    private static final String csvFile = "";
    public static void main(String [] args) throws IOException {
        CoordinateTransformation ct = TransformationFactory.getCoordinateTransformation(
        TransformationFactory.WGS84, HAWAIICoord);
        Scenario scenario = ScenarioUtils.createScenario(ConfigUtils.createConfig());

        int columns = 3, num = 145;
        createPersons(scenario, num, ct, columns);
        createActivities(scenario,ct);
        String popFilename = "population.xml";
        new PopulationWriter(scenario.getPopulation(), scenario.getNetwork()).write(
        popFilename);
        log.info("population written to: " + popFilename);
    }
    private static void createActivities(Scenario scenario, CoordinateTransformation ct){

        Population pop = scenario.getPopulation();
        PopulationFactory pb = pop.getFactory();

        for (Person pers : pop.getPersons().values()) {
            Plan plan = pers.getPlans().get(0);
            Activity homeAct = (Activity) plan.getPlanElements().get(0);
            homeAct.setEndTime(7*3600);

            Leg leg = pb.createLeg(TransportMode.car);
            plan.addLeg(leg);

            Point p = getShelterPointInFeature(ct);
            Activity shelt = pb.createActivityFromCoord("shelter", new Coord(p.getX(),
            p.getY()));
            double startTime = 8*3600;
            shelt.setStartTime(startTime);
            plan.addActivity(shelt);
        }
    }

    private static void createPersons(Scenario scenario, int number,
    CoordinateTransformation ct, int col){
        Population pop = scenario.getPopulation();
        PopulationFactory pb = pop.getFactory();
        String[][] position = new String[number][col];
        position = CSV.getCSVData(csvFile, number, col);
        int i = 0;

        for (; number > 0; number--) {
            Person pers = pb.createPerson(Id.create(position[i][0], Person.class));
            pop.addPerson(pers);
            Plan plan = pb.createPlan();
            Coord c = new Coord(Double.parseDouble(
            position[i][1]), Double.parseDouble(position[i][2]));
            Coord coord = ct.transform(c);
            Activity act = pb.createActivityFromCoord("home", new Coord(coord.getX(),
            coord.getY()));
            plan.addActivity(act);
            pers.addPlan(plan);
            i++;
        }
    }

    public static Point getShelterPointInFeature(CoordinateTransformation ct) {

        Coord c = ShelterCoord.getCoord(ct);
        return MGC.coord2Point(c);
    }
}
```

Código Fonte 6.12: Criando as populações *MATSim*

#### 6.3.3.4. Arquivo *config.xml* utilizado na simulação

```
<config>
	<module name="global">
		<param name="randomSeed" value="4711" />
		<param name="coordinateSystem" value="EPSG:2782" />
	</module>

	<module name="network">
		<param name="inputNetworkFile" value="artigo.xml.gz" />
	</module>

	<module name="plans">
		<param name="inputPlansFile" value="population_before.xml.gz" />
	</module>

	<module name="controler">
		<param name="outputDirectory" value="./output" />
		<param name="firstIteration" value="0" />
		<param name="lastIteration" value="5" />
		<param name="writeEventsInterval" value="1" />
	</module>

	<module name="qsim">
		<param name="startTime" value="06:00:00" />
		<param name="endTime" value="23:00:00" />
		<param name = "snapshotperiod" value = "07:00:00"/>
	</module>

	<module name="planCalcScore">
		<param name="learningRate" value="1.0" />
		<param name="BrainExpBeta" value="2.0" />
		<param name="lateArrival" value="-18" />
		<param name="earlyDeparture" value="-0" />
		<param name="performing" value="+6" />
		<param name="traveling" value="-6" />
		<param name="waiting" value="-0" />

		<param name="activityType_0" value="home" />
		<param name="activityPriority_0" value="1" />
		<param name="activityTypicalDuration_0" value="12:00:00" />
		<param name="activityMinimalDuration_0" value="06:00:00" />
		<param name="activityOpeningTime_0"  value="00:00:00"/>
		<param name="activityClosingTime_0" value="07:00:00" />

		<param name="activityType_1" value="shelter" />
		<param name="activityPriority_1" value="1" />
		<param name="activityTypicalDuration_1" value="16:00:00" />
		<param name="activityMinimalDuration_1" value="06:00:00" />
		<param name="activityOpeningTime_1" value="08:00:00" />
		<param name="activityLatestStartTime_1" value="09:00:00" />
		<param name="activityEarliestEndTime_1" value="" />
		<param name="activityClosingTime_1" value="" />
	</module>

	<module name="strategy">
		<param name="maxAgentPlanMemorySize" value="5" />
		<param name="ModuleProbability_0" value="0.9" />
		<param name="Module_0" value="BestScore" />
		<param name="ModuleProbability_1" value="0.1" />
		<param name="Module_1" value="ReRoute" />
	</module>
</config>
```

Código Fonte 6.13: Criando as populações *MATSim*

### 6.3.4. *Via*

O *Via* é um *software* independente que possibilita a visualização e análise de dados do *MATSim*, bem como conjuntos de dados espaciais e temporais genéricos. Os dados visualizados no *Via* são estruturados em camadas. Cada camada é responsável por visualizar um conjunto de dados. Assim, ele fornece camadas para visualizar redes, instalações, atividades e viagens de agentes, paradas de transporte público ou apenas pontos genéricos no espaço, como uma coordenada qualquer.

As simulações do estudo de caso foram feitas utilizando o Código Fonte 6.13 importado no *MATSim*. A simulação da primeira população levou 18 segundos para ser completada, enquanto que a segunda levou 19 segundos. Dessa forma, arquivos de *output* foram criados, viabilizando a visualização no *Via*. Além disso, arquivos de cada iteração foram gerados para um melhor entendimento do que acontece na simulação.

Utilizando os arquivos *output_network.xml.gz* e *output_events.xml.gz*, todo processo de simulação pôde ser visualizado. Durante a simulação agentes saíram de suas casas e dirigiram-se para o abrigo definido na etapa da criação da população. Às 09:55:42 todos os agentes chegaram a seus respectivos abrigos, terminando assim a simulação. Para uma melhor visualização foi utilizado o mapa *Microsoft Hybrid*.

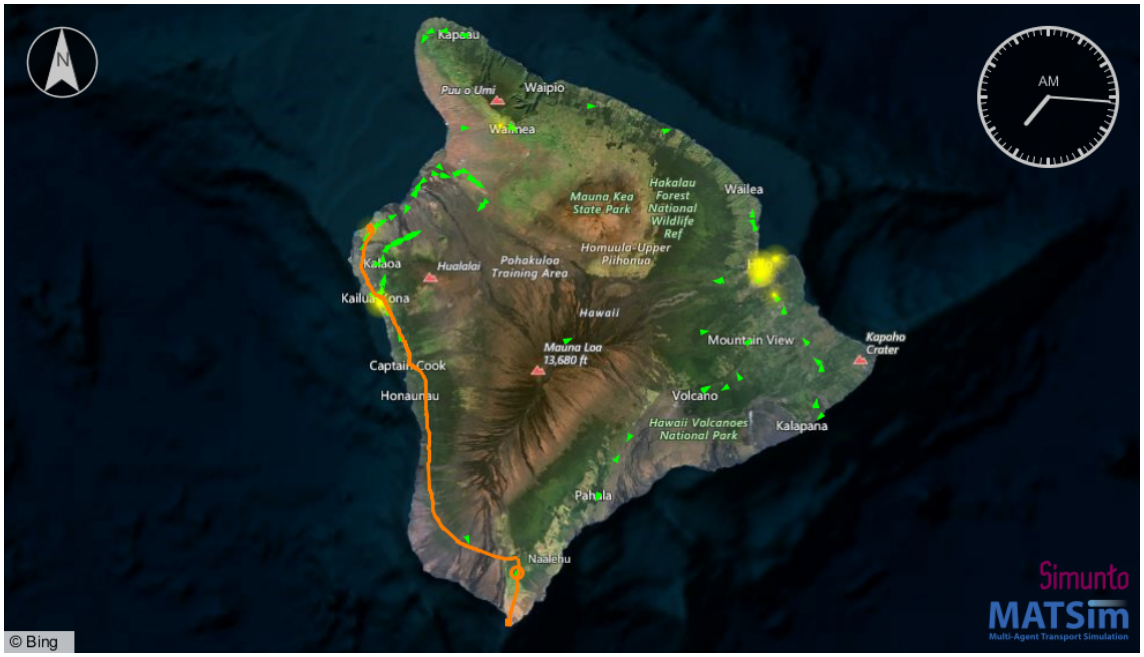



Figura 6.12: Representação da população antes do desastre.

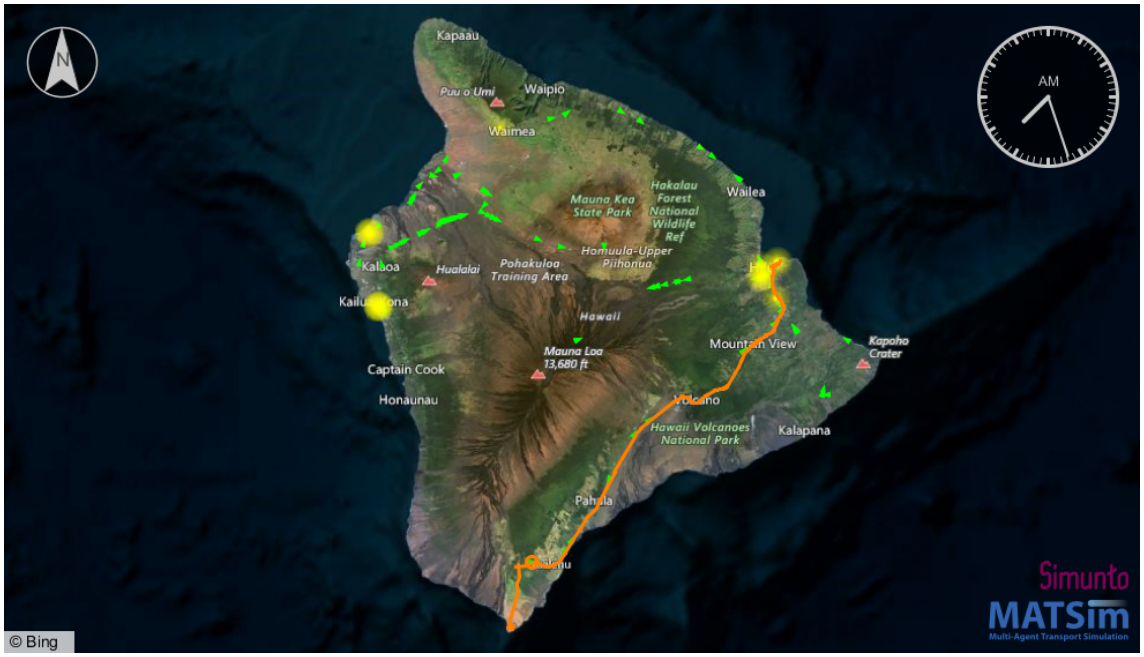



Figura 6.13: Representação da população depois do desastre.

## Referências

[Anbalagan e Valliyammai 2016] Anbalagan, B. e Valliyammai, C. (2016). #chennaifloods: Leveraging human and machine learning for crisis mapping during disasters using social media. Em *2016 IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW)*, páginas 50–59.

[Andreas Horni 2019] Andreas Horni, Kai Nagel, K. W. A. (2019). *The Multi-Agent Transport Simulation MATSim*.

[Arianto et al. 2018] Arianto, R., Warnars, H. L. H. S., Gaol, F. L., e Trisetyarso, A. (2018). Mining unstructured data in social media for natural disaster management in indonesia. Em *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, páginas 192–196.

[Beniwal et al. 2018] Beniwal, R., Gupta, V., Rawat, M., e Aggarwal, R. (2018). Data mining with linked data: Past, present, and future. Em *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, páginas 1031–1035.

[Bulletin of Volcanology Editor-in-Chief 2018] Bulletin of Volcanology Editor-in-Chief (2018). Eruption crisis at kilauea caldera (big island of hawaii, usa). *Bulletin of Volcanology*, 80(8):66.

[Hernandez-Suarez et al. 2019] Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, K., Perez-Meana, H., Portillo-Portillo, J., Luis, V. S., Javier García Villalba, L., et al. (2019). Using twitter data to monitor natural disaster social dynamics: a recurrent neural network approach with word embeddings and kernel density estimation. *Sensors*, 19(7):1746.

[Martín et al. 2017] Martín, Y., Li, Z., e Cutter, S. L. (2017). Leveraging twitter to gauge evacuation compliance: spatiotemporal analysis of hurricane matthew. *PLoS one*, 12(7):e0181701.

[Murthy e Gross 2017] Murthy, D. e Gross, A. J. (2017). Social media processes in disasters: Implications of emergent technology use. *Social Science Research*, 63:356 – 370.

[Sadhukhan et al. 2018] Sadhukhan, S., Banerjee, S., Das, P., e Kumar Sangaiah, A. (2018). Producing better disaster management plan in post-disaster situation using social media mining. páginas 171–183.

[Wibowo 2017] Wibowo, T. W. (2017). Spatial point data analysis of geolocated tweets in the first day of eid al-fitr 2017 in java island. *IOP Conference Series: Earth and Environmental Science*, 98:012032.

# Capítulo 7

# Construindo Modelo de Site com Fundamentos em Bootstrap

Gabriell Oliveira Paes Landim, Luis Guilherme Sousa e Silva, Antônio Oseas de Carvalho Filho e Rafael Luz Araújo

***Abstract***

*This meta-article describes how technologies used for web programming can be used for website building and how using the CSS Bootstrap framework automates the process of styling web pages. The algorithms and scripts that are implemented by HTML markup, CSS styling, and element animation are built in JavaScript, as they are high-level languages, with broad access to their simple syntax, making it pedagogical and easy understanding. These updated languages shape how websites are currently developed. The Bootstrap tool utilizes such aforementioned technologies, improving and streamlining implementation and quality in web development results.*

***Resumo***

*Este capítulo descreve como as tecnologias usadas para programação WEB podem ser usadas para a construção de sites e como o uso do framework CSS Bootstrap automatiza o processo de estilização das páginas WEB. Os algoritmos e scripts que são implementados pela marcação em HTML, a estilização se dá pelo CSS e a animação dos elementos é construída em JavaScript, dado que são linguagens de alto nível, com amplo acesso por sua sintaxe simples, tornando-a pedagógica e de fácil entendimento. Estas linguagens em sua versão atualizadas moldam a forma de como sites são desenvolvidos atualmente. A ferramenta Bootstrap se utiliza de tais tecnologias citadas anteriormente, melhorando e agilizando a implementação e a qualidade nos resultados do desenvolvimento WEB.*

## 1.1. Introdução

Nos dias atuais a maior parte do que é desenvolvido de aplicações é concentrado para a web. O desenvolvimento WEB é o termo utilizado para a criação de um *software*, que culmina na construção de páginas que podem variar de simples sites estáticos a aplicações

de grande porte, o que permite conectividade, integração entre diferentes programas e ações com resultados instantâneos, trazendo com si uma série de vantagens, segundo FERNANDES (2018) são elas:

- Acessibilidade: Um sistema integrado da web permite maior mobilidade para quem vai trabalhar com ele, pois pode ser acessado a partir de qualquer dispositivo conectado à Internet e acessado de qualquer lugar sem a necessidade de estar fisicamente no site.
- Custo-Benefício: Quando se tem um sistema web, não precisa pensar em muitas equipes para acessá-lo. Apenas um computador conectado à Internet. Um sistema da Web, incluindo uma parte traseira um pouco mais alta, fornece soluções para seus problemas em geral. Otimiza o gerenciamento para que você possa analisar seus dados mais rapidamente.
- Segurança: Os sistemas da Web possuem criptografias que impedem invasores com intenção maliciosa. Além disso, os *backups* periódicos são realizados automaticamente. A integridade dos dados e a segurança das informações são garantidas pelo acesso restrito por usuários autorizados com perfis predefinidos do que eles podem e não podem acessar.
- Suporte prático: Como o sistema está na Internet, nem sempre é necessário que um funcionário passe a prestar suporte técnico, e na maioria das vezes é possível identificar o problema e resolvê-lo com eficiência, as atualizações também podem ser feitas online, sem nenhuma ação cara a cara.

Segundo Intelectua Team (2017), existe um amplo leque de opções quando se trata de desenvolvimento WEB, com muitas tecnologias e ferramentas que cabe ao desenvolvedor escolher qual se encaixa ao seu perfil e necessidades. Considerando que toda a construção de páginas web na parte do cliente se dá por três linguagens o HTML, CSS e JavaScript que serão melhor exploradas nos subcapítulos adiante.

## 1.2. HTML

O *HyperText Markup Language* (HTML) segundo o Mozilla Team (2019) é uma linguagem de marcação de hipertexto em páginas WEB, sendo o bloco de construção mais básico da web definindo a estrutura do conteúdo na web.

A internet tem como base está linguagem, que foi projetada para ser de fácil entendimento tanto para os computadores quanto para os programadores que foram utilizá-la. Tecnologias além do HTML são frequentemente usadas para construir a aparência/apresentação (CSS) ou a funcionalidade/comportamento (JavaScript) de uma página da web que serão apresentadas adiante neste capítulo.

Voltando ao HTML, as palavras marcação e hipertexto trazem a real essência do funcionamento da linguagem e como se organiza o conteúdo e a informação em uma página na web. Para adentrar nos códigos, será explicado os principais conceitos do HTML.

A Figura 1 apresenta a estrutura básica de uma página HTML com as *tags* fundamentais para que os navegadores interpretam corretamente o documento.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title></title>
5  </head>
6  <body>
7      <!-- Conteúdo -->
8  </body>
9  </html>
```

**Figura 1.1. Estrutura básica do HTML**

- **Linha 1:** A instrução “DOCTYPE” indica ao navegador qual a versão do HTML que estamos trabalhando, e se trata da mais recente versão, HTM5 e deve ser a primeira instrução a ser inserida em um documento;
- **Linhas 2 e 9:** Delimita o documento, onde todas as *tags* devem estar contidas nesse espaço;
- **Linhas 3 e 5:** o conteúdo neste espaço não é visível no navegador, esta é a abertura e fechamento da *tag* <*head*>, que define o cabeçalho do documento.
- **Linha 4:** a *tag* <*title*> define o título da página.
- **Linhas 6 e 8:** abrindo e fechando a *tag* <*body*>, marcando o espaço em que o conteúdo visual da página deve estar. Outras etiquetas que representam texto, botões etc. devem ser adicionadas. nesta faixa;
- **Linha 7:** comentário em HTML.

A princípio nota-se a semelhança com uma linguagem de programação comum, mas como já foi dito o HTML é uma linguagem de marcação, nela não existem estruturas de decisão e laços de repetição, seus comandos são dados por meios de *tags*, que serão explicadas posteriormente.

### 1.2.1 Tags

As *tags* nada mais são do que um conjunto de caracteres que formam um elemento.

Existem basicamente dois tipos de *tags*, as que necessitam de fechamento e as que não necessitam do fechamento. Sua declaração se dá pelo uso do caractere de menor “<”, o “nome_da_tag” e por fim o sinal de maior “>” para a abertura. E as que necessitam de fechamento utilizamos o sinal de menor “<”, seguido de barra “/”, “nome_da_tag” e o sinal de maior “>”, como mostra a Figura 1.

```
1  <!-- Exemplo de elemento que necessita de fechamento -->
2  <p>A tag do elemento de parágrafo necessita de fechamento.</p>
3
```

**Figure 7.1. Tag HTML com fechamento.**

Os elementos que não fecham, também conhecidos como elementos vazios, usam apenas o sinal de menos “<”, seguido pelo nome do elemento e o sinal de mais “>”.

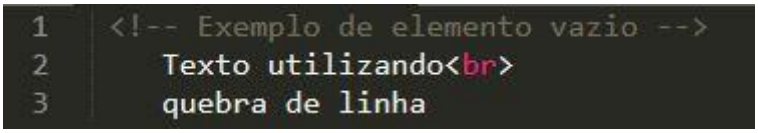



**Figura 7.2. Tag HTML sem fechamento.**

Serão apresentadas uma lista das principais *tags* HTML, são elas:

- **<!--...-->:** Define um comentário;
- **<a>:** Define um hyperlink;
- **<address>:** Define um endereço. (Passa a ser tratado como uma seção);
- **<b>:** Define um texto em negrito; (Possui o mesmo nível semântico que um SPAM, e também o estilo de negrito no texto. Contudo, ele não dá nenhuma importância para o texto marcado com ele.);
- **<br>:** Insere uma quebra de linha simples;
- **<button>:** Define um botão de comando;
- **<caption>:** Define o "*caption*" de uma tabela;
- **<code>:** Define o código texto do computador;
- **<col>:** Define os atributos da coluna da tabela;
- ** :** Define uma seção no documento;
- **:** Define um texto em ênfase;
- **<form>:** Define um formulário;
- **<h1>** até **<h6>:** Define do cabeçalho 1 até o cabeçalho 6;
- **<hr>:** Define uma regra horizontal. (Tem o mesmo nível que um parágrafo, mas também é utilizado para fazer separações e quebras de linha);
- **<iframe>:** Define uma linha sobre a janela (frame);
- **<img>:** Define uma imagem;
- **<input>:** Define um campo de inserção;
- **<label>:** Define uma "*label*" para o formulário;
- **<li>:** Define os itens da lista;
- **<link>:** Define uma referência;
- **<map>:** Define uma imagem de mapa;
- **<menu>:** Define uma lista de "menus";
- **<meta>:** Define informações meta;
- **<object>:** Define um objeto incorporado;
- **<ol>:** Define uma lista ordenada;
- **<p>:** Define um parágrafo;
- **<script>:** Define um script;
- **:** Define um texto forte (similar ao negrito);
- **<style>:** Define um estilo;
- **<sub>:** Define um texto subscrito;
- **<table>:** Define uma tabela;
- **<td>:** Define uma célula da tabela;
- **<textarea>:** Define uma área de texto;
- **<title>:** Define o título do documento;
- **<tr>:** Define uma linha da tabela;
- **<ul>:** Define uma lista desordenada;
- **<var>:** Define uma variável;

### 1.3. CSS

CSS, ou *Cascading Style Sheets*, é uma linguagem para estilizar os elementos representados no documento HTML. Esta linguagem permite que os elementos do documento possam ser personalizados, adicionando cores, tamanhos, sombras e redefinindo a disposição do elemento na tela, G. Ariane (2019).

#### 1.3.1 Seletor

O seletor, em CSS, é o objeto ao qual deseja-se estilizar no documento. São as mesmas *Tags* do HTML. Porém, através do CSS, é possível selecionar um grupo de elementos de uma só vez e também adicionar uma estilização para um elemento em específico. Porém, isso pode ser feito de diversas formas: selecionando mais de uma *Tag* ao mesmo tempo, selecionando uma classe ou selecionando um identificador (ID), para objetos únicos. Ambas as formas podem ser combinadas.

Para selecionar mais de uma *Tag* HTML ao mesmo tempo, os seletores deverão ser separados por "," vírgula.

- Ex.: p, h1, h2{

  *instruções...*

  }

Para selecionar um grupo de objetos que possuem a mesma classe HTML, podemos usar a sintaxe "." ponto, seguido do nome da classe.

- Ex.: Para selecionar a classe com nome "cor-cinza".

  .cor-cinza{

  *instruções...*

  }

Para selecionar um elemento único que possua um ID HTML de nome "cpf":

- #cpf{

  *instruções...*

  }

#### 1.3.1 Propriedades e Valor

As Propriedades no CSS são a resposta para "o que eu desejo alterar?" no elemento HTML.

Através das propriedades dos elementos selecionados, pode-se atribuir um novo valor para aquela propriedade.

As propriedades e valores são separados por ":" dois pontos e finalizados por um ";" ponto e vírgula.

Ex.: Para estilizar a propriedade de cor de um elemento parágrafo do HTML para a cor vermelha:

- p{

color: red;

}

Para adicionar mais propriedades para estilização, basta adicionar mais *propriedade:valor;* antes do fechamento do bloco estilizado, delimitado por "}" chave fechando.

A Figura 1.3 mostra como é a estrutura básica de um código CSS.

```
1  h1{
2      color: red;
3  }
4  p{
5      font-size: 24px;
6  }
7  div{
8      background-color: black;
9      width: 400px;
10 }
```

**Figura 7.3. Estrutura básica de um código CSS.**

A Figura 1.3 mostra como é a estrutura básica de um código CSS.

- **Linha 1:** Definindo o seletor, elemento que será selecionado para estilização.
- **Linha 2, 5, 8 e 9:** Definindo qual propriedade do elemento vai ser estilizada e o valor que ela vai receber. Para o exemplo da linha 2: a instrução fará com que o seletor escolhido tenha uma cor vermelha.
- **Linha 3, 6 e 10:** Fechamento dos blocos de estilização do seletor informado.

Abaixo está uma lista com as propriedades de estilização mais comuns.

- **color:** estiliza a cor do elemento. Os valores podem ser em linguagem natural, hexadecimal ou padrão rgb.
- **font-size:** estiliza o tamanho da fonte do elemento. Os valores podem ser dados em px (pixel), pt (pontos), cm (centímetros), em (valor padrão da fonte), em (valor da fonte relativo ao navegador).
- **background-color:** estiliza a cor de fundo dos objetos. Os valores podem ser dados em linguagem natural, hexadecimal e rgb.
- **font-family:** estiliza o tipo da fonte do elemento que vai ser exibido. Suporta fontes como "Arial", "Times new Roman", etc.
- **width e height:** estilizam a largura e altura dos elementos, respectivamente. Os valores podem ser dados em px, cm, em, rem, pt, etc.
- **margin:** estiliza a margem da disposição de um elemento em relação aos seus vizinhos.
- **padding:** estiliza o espaçamento interno do elemento.
- **border:** estiliza a borda do elemento HTML.

- **text-decoration:** estiliza a decoração do texto. Os valores podem ser *underline* (sublinhado), *none* (sem estilo), *overline* (linha em cima do texto) e *line-through* (linha cortando o texto ao meio).

## 1.4. JavaScript

Usando tecnologias de desenvolvimento web HTML e CSS, é capaz de criar nada além de um site estático sem a interação do usuário, como validação de entrada de dados, formulários etc. Para que essa interação seja possível, é necessário usar uma linguagem de programação. Quando se trata de criar sites, a linguagem de programação mais básica é o *JavaScript*, tornando-se um dos pilares da criação de sites.

O *JavaScript* RODRIGUES(2016) pode ser definido como uma linguagem de programação que fornece a capacidade de manipular software de outros programadores. Ou seja, o desenvolvedor da Web tem a capacidade de controlar o comportamento dos navegadores por meio de linhas de código. Possuindo uma grande tolerância a erros, dinâmica e sem compilação para executar, com o código interpretado como lido pelo navegador, uma linha de cada vez, assim como HTML.

Algumas das principais vantagens do *JavaScript* são:

- Permite inserir vários efeitos, tornando o site mais dinâmico;
- Permite executar instruções em resposta a ações do usuário;
- Ele permite que os programadores da Web melhorem suas páginas e o poder de seus projetos;
- Permite a programação de pequenos scripts e programas maiores;
- Disponibiliza ao desenvolvedor tudo o que forma o site.

Abaixo serão dados exemplos da sintaxe do *JavaScript*.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <script type="text/javascript" src="meuArquivo.js"></script>
5    </head>
6    <body>
7    </body>
8  </html>
9
```

**Figura 7.4. Referenciando um código JavaScript.**

Será chamado o arquivo salvado com a extensão *.JS* referenciando do "<*head*>" do HTML.

```
1  var nome;
2  nome = “Fulano”;
3  var idade = 20;
4  idade = 20 + 30 - 12*4;
5
```

**Figura 7.5. Declaração de variáveis JavaScript.**

Possui uma tipagem dinâmica, em outras palavras, não se tem a necessidade de definir o tipo das variáveis ao fazer a sua declaração.

## 1.5. Bootstrap

O *Bootstrap* é um *Framework* CSS criado por um dos desenvolvedores do Twitter em meados de 2010. Segundo o *Bootstrap* Team (2019), o *Bootstrap* é o Framework CSS mais popular do mundo e atualmente está na sua versão quatro.

Através deste Framework, pode-se criar qualquer tipo de *template* para os sites e sistemas web e em todos os dispositivos, já que usa os conceitos da web responsiva e *mobile-first*. A figura 1.4 mostra como incluir o *Bootstrap* na página web via *Content Delivery Network* (CDN).

```
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
       bootstrap/4.3.1/css/bootstrap.min.css" integrity="
       sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
       crossorigin="anonymous">
6  </head>
7  <body>
8      <!-- Optional JavaScript -->
9      <!-- jQuery first, then Popper.js, then Bootstrap JS -->
10     <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="
       sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
       crossorigin="anonymous"></script>
11     <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/
       popper.min.js" integrity="
       sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
       crossorigin="anonymous"></script>
12     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/
       bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/
       nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
13 </body>
14 </html>
```

**Figura 7.6. Importação do Bootstrap CSS e arquivos JS para suas funcionalidades.**

Abaixo está alguma das classes mais utilizadas do *Framework Bootstrap*:

- **Layout:**
  - **container:** Faz o bloco ficar com um recuo nas margens da esquerda e direita.
  - **row:** Cria-se um bloco horizontal imaginário (linha).
  - **col:** Cria-se uma coluna imaginária (geralmente englobado por uma classe do tipo row).

Por padrão, o *Bootstrap* divide a *row* em até 12 colunas. Caso queira exibir uma coluna do tamanho de 4, coloca-se a coluna 'col-4'. Esse número pode variar de 1 até 12.

  - **col:** Cria-se uma coluna imaginária (geralmente englobado por uma classe do tipo *row*).
- **Conteúdo:**
  - **img-fluid:** Transforma uma imagem em responsiva.
  - **img-thumbnail:** Transforma a imagem no padrão *thumbnail* (200x200).
  - **rounded:** Arredonda os cantos de um bloco (imagens, divs, etc).
  - **table:** Estiliza a uma tabela no padrão do *Bootstrap*.
  - **table-striped:** Dá um contraste entre uma linha e outra da tabela. É usado juntamente com a classe *table*.
  - **table-hover:** Adiciona um efeito de contraste a linha da tabela em que o *mouse* está passando. É usado em conjunto com a classe *table*.
  - **table-sm:** Comprime o espaçamento entre as linhas da tabela, diminuindo o tamanho da mesma. É utilizado juntamente com a classe *table*.
- **Componentes:**
  - **alert:** Estiliza a *div* para um bloco de notificação.
  - **btn:** Estiliza um *link* ou botão para o padrão de botão do *Bootstrap*.

    Pode ser utilizado em diversas cores: *btn-primary, btn-success, btn-danger, btn-waning*, etc.

  - **card:** Estiliza um bloco para maior destaque, adicionando bordas ao bloco.
  - **form-control:** Estiliza os inputs do HTML para um padrão mais agradável de formulário.
  - **modal:** Caixa de texto que 'salta' na tela. Geralmente utilizada para caixa de confirmações ou descrição de algum objeto.
  - **navbar:** Menu inicial de um site.
  - **spinner-border:** Usado para adicionar pequenas animações no documento, como por exemplo, animação de 'carregando'.
  - **toast:** Pequeno bloco de notificação ou resposta.
- **Utilidades:**
  - **border-top:** Adiciona uma borda ao topo do bloco.
  - **border-bottom:** Adiciona uma borda embaixo do bloco.
  - **border-right:** Adiciona uma borda à direita do bloco.
  - **border-left:** Adiciona uma borda à esquerda do bloco.
  - **d-inline:** Adiciona um *display inline* ao bloco.
  - **d-block:** Adiciona um *display* do tipo bloco no elemento.
  - **d-inline-block:** Adiciona um *display* do tipo *inline-block* no elemento.
  - **d-flex:** Adiciona o *display* do tipo *flex* no bloco.
  - **mt:** Adiciona uma margem ao topo do elemento.
  - **mb:** Adiciona uma margem embaixo do elemento.
  - **ml:** Adiciona uma margem à esquerda do elemento.
  - **mr:** Adiciona uma margem à direita do elemento.

O *Bootstrap* possui inúmeras classes e elementos, nos quais a consulta pode ser realizada na documentação oficial do Framework.

## 1.7. Conclusão

Neste capítulo foram apresentados conceitos gerais sobre as principais linguagens utilizadas no desenvolvimento de aplicações e sistemas Web. Tais linguagens são os pilares que moldam a construção de páginas e sites atualmente na internet.

O HTML constitui o esqueleto do site, por assim dizer, nele se encontram tudo que pode ser visto, sendo assim a estrutura base do site, onde se diferenciam os títulos dos parágrafos e estão contidos as imagens e os *links*. O CSS é a parte de estilização do site tornando-o amigável para os usuários, pode se dizer que basicamente o embelezamento do site é dado por essas linguagens. O *JavaScript* constitui a parte lógica do sistema Web, é nele que se torna possível realizar ações e definir as funcionalidades. Por fim o *Bootstrap* que é uma ferramenta que conta com todas essas linguagens já implementadas e pronta para ser utilizada, facilitando e agilizando o desenvolvimento de aplicações Web.

Com isso, foi atingido o principal objetivo do capítulo que era expor uma visão geral sobre os fundamentos do desenvolvimento Web, suas linguagens e ferramentas para um melhor aproveitamento, como um guia aos iniciantes na construção de sites.

## 1.8. Referências

G., Ariane. (2018) “Como ser um Desenvolvedor Web: Por onde começar?”, https://www.hostinger.com.br/tutoriais/como-ser-um-desenvolvedor-web/, outubro/2019.

Bootstrap Team, (2019) “Boostrap Get Starter”, https://getbootstrap.com/, outubro/2019.

FERNANDES, Gabriel. (2018) “As 5 vantagens do Sistema Web que irão alavancar o seu negócio”, https://fluxoconsultoria.poli.ufrj.br/blog/tecnologia-informacao/vantagens-do-sistema-web/, outubro/2019.

Intelectua Team. (2017) “Você realmente sabe o que é desenvolvimento web?”, https://www.intelectua.com.br/blog/o-que-e-desenvolvimento-web, outubro/2019.

Mozilla Team. (2019) “HTML: Linguagem de Marcação de Hipertexto”, https://developer.mozilla.org/pt-BR/docs/Web/HTML, outubro/2019.

EIS, Diego. (2011) “O básico: O que é HTML?”, https://tableless.com.br/o-que-html-basico/, outubro/2019.

França, Rafael. (2013) “O que é HTML? Linguagem base de websites”, https://tableless.github.io/iniciantes/manual/html/, outubro/2019.

RODRIGUES, Joel. (2010) “HTML básico - códigos HTML”, https://www.devmedia.com.br/html-basico-codigos-html/16596, outubro/2019.

RIBEIRO, Rafael. (2012) “Comandos e tags HTML5”, https://www.devmedia.com.br/html-basico-codigos-html/16596, outubro/2019.

G., Ariane. (2019) “O que é CSS? Guia Básico para Iniciantes”, https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/, outubro/2019.

RODRIGUES, Joel. (2016) “JavaScript: Tutorial”, https://www.devmedia.com.br/javascript-tutorial/37257, outubro/2019.

OLIVEIRA, Andréa. (2014) “Linguagem de Programação JavaScript: as principais vantagens”, https://www.cpt.com.br/cursos-informatica-desenvolvimentodesoftwares/artigos/linguagem-de-programacao-javascript-as-principais-vantagens, outubro/2019.

Bootstrap Team. (2019) “Bootstrap”, https://getbootstrap.com/, outubro/2019.

# Chapter 8

# Mapeamento sistemático feito com o auxílio de ferramenta computacional

Ana Beatriz Barbosa Maia da Silva, Francisco das Chagas Imperes Filho, Thales José Sousa Bezerra

***Abstract***

*Systematic Mapping of Literature (MSL) is a scientific method capable of identifying, interpreting and summarizing works relevant to a particular line of research, area or phenomenon of interest in a non-biased and replicable way [Kitchenham 2004].This methodology uses primary studies related to a Research Question (QP) with the specific objective of integrating / synthesizing the evidence related to this question. This short course introduces TheEnd, a Web service that can help plan, execute, and summarize the results of an MSL, making search more agile and replicable.*

***Resumo***

*Mapeamento Sistemático da Literatura (MSL) é um método científico capaz identificar, interpretar e sumarizar os trabalhos relevantes para determinada linha de pesquisa, área ou fenômeno de interesse de forma não tendenciosa e replicável [Kitchenham 2004]. Essa metodologia utiliza estudos primários relativos a uma Questão de Pesquisa (QP) com o objetivo específico de integrar/sintetizar as evidências relacionadas a essa questão. O presente minicurso apresenta o TheEnd, um serviço Web capaz de auxiliar o planejamento, execução e sumarização dos resultados de um MSL, tornando a pesquisa mais ágil e replicável.*

## 1.1. Introdução

Um Mapeamento Sistemático (MS) é uma revisão ampla dos estudos primários existentes em um tópico de pesquisa específico que visa identificar a evidência disponível neste tópico. Assim, um MS é um estudo secundário[1] que tem como objetivo identificar e classificar a pesquisa relacionada a um tópico amplo de pesquisa.

Resultados de um MS ajudam a identificar lacunas nesta área, capazes de sugerir pesquisas futuras e prover um guia para posicionar adequadamente novas atividades de pesquisa (KITCHENHAM; CHARTERS, 2007; KITCHENHAM et al., 2011; PETERSEN et al., 2008). Assim, MSs visam prover uma visão geral de um tópico e identificar se há subtópicos nos quais mais estudos primários são necessários.

## 1.2. TheEnd

A ferramenta TheEnd, desenvolvida pelo Laboratório de Engenharia de Software e Informática Industrial (EaSII), vinculado à Universidade Federal do Piauí, oferece um serviço Web capaz de auxiliar o planejamento, execução e sumarização dos resultados de um MSE, tornando a pesquisa mais ágil e replicável.

### 1.2.1. Conta de Usuário

A criação de um usuário no TheEnd é muito simples, basta acessar o endereço (https://easii.ufpi.br/theend/home/create) e informar o seu email, o nome e uma senha para criar sua conta.

### 1.2.2. Plataforma

Com o seu usuário logado a plataforma irá mostrar o painel inicial onde apresenta informações explicativas de cada componente de uma MSL:

- Planejamento
- Execução
- Sumarização

#### 1.2.2.1. Planejamento

Nessa etapa os pesquisadores devem planejar a execução do estudo, definindo os objetivos, as questões de pesquisa, as bases de dados que serão utilizadas, a string de busca a ser aplicada nas bases de dados, os critérios para inclusão e exclusão dos trabalhos obtidos, formulários para extração de informações relevantes, dentre outros aspectos.

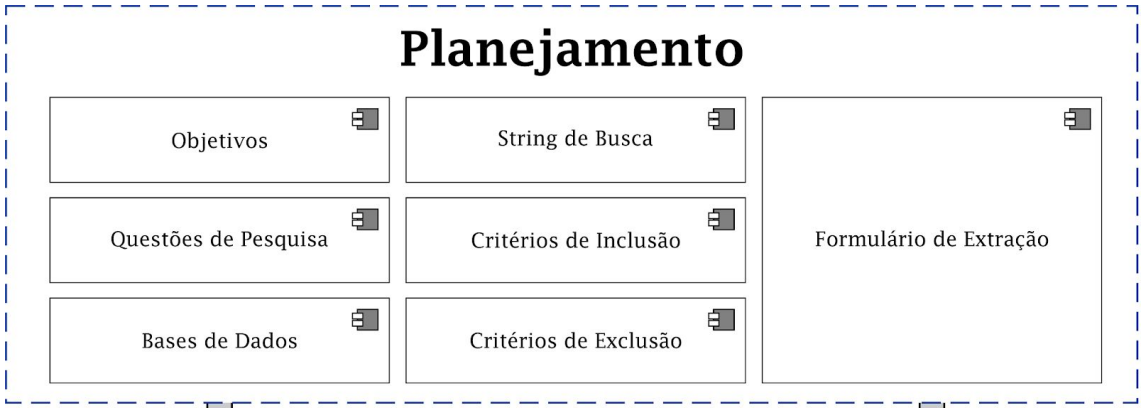



**Figura 1.1. Modelo de Planejamento**

A figura 1.1 ilustra as etapas de criação do planejamento na plataforma

### 1.2.2.2. Execução

Essa é a etapa mais trabalhosa, pois o pesquisador deve aplicar a string de busca nas bases de dados para obter o conjunto de estudos primários; depois é necessário aplicar os critérios de inclusão/exclusão considerando inicialmente a leitura do título e resumo dos trabalhos obtidos na busca; e posteriormente a leitura completa dos trabalhos incluídos para extração de dados relevantes. Além disso, durante a execução pode ser realizada uma análise das referências presentes nos trabalhos selecionados, a fim de buscar trabalhos correlatos que não foram encontrados nas bases de dados.

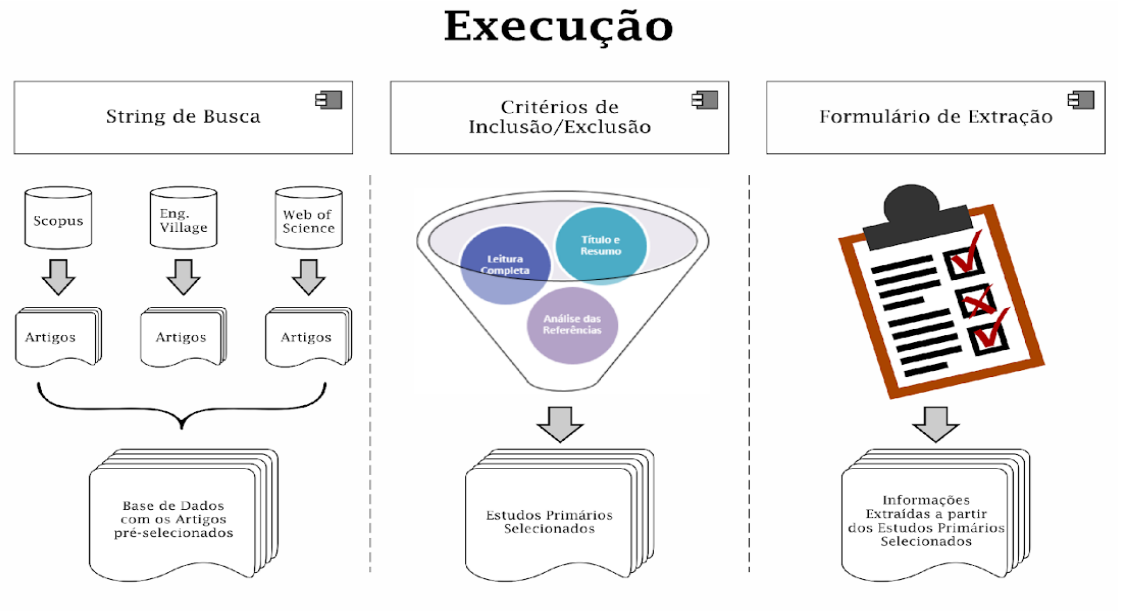



**Figura 1.2. Modelo de Execução**

A figura 1.2 ilustra as etapas de execução na plataforma.

### 1.2.2.3. Execução

Por fim, a terceira etapa é destinada à sumarização dos resultados, organizando as informações em gráficos e tabelas, a fim de facilitar o entendimento da linha de pesquisa. Nessa etapa também ocorre a discussão sobre os resultados obtidos, apresentando as principais contribuições do mapeamento sistemático.

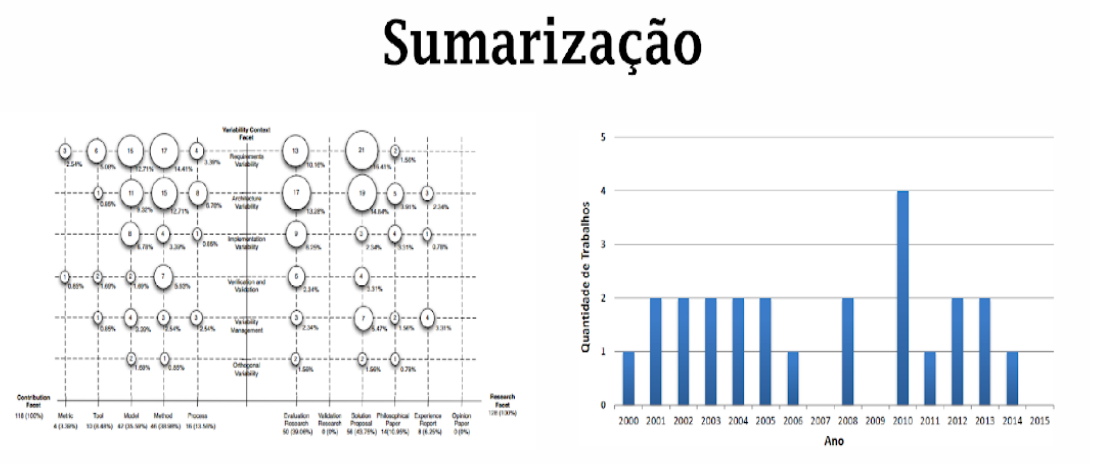



**Figura 1.3. Sumarização em gráficos**

A figura 1.3 ilustra exemplos de gráficos.

### 1.2.3. Dashboard

A dashboard apresenta as primeiras funcionalidade da plataforma, como a lista de mapeamentos e novo mapeamento onde iremos criar nosso primeiro projeto.

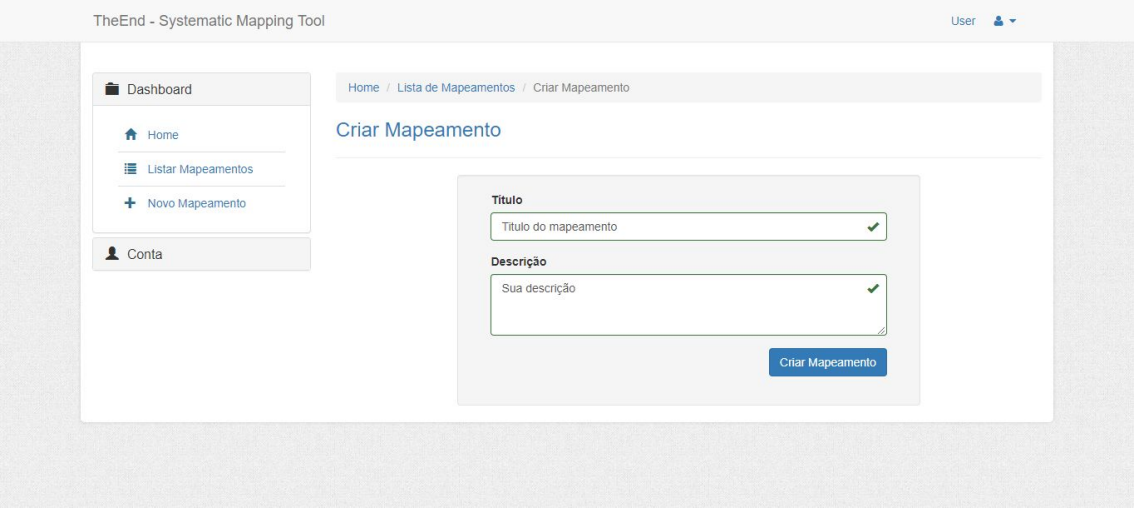



**Figura 1.4. Criação do Mapeamento**

A figura 1.4 mostra a página de criação do mapeamento com seu título e descrição.

Com a criação do mapeamento efetuado temos a lista de mapeamento cadastrado na plataforma assim como mostrado na imagem 1.5.

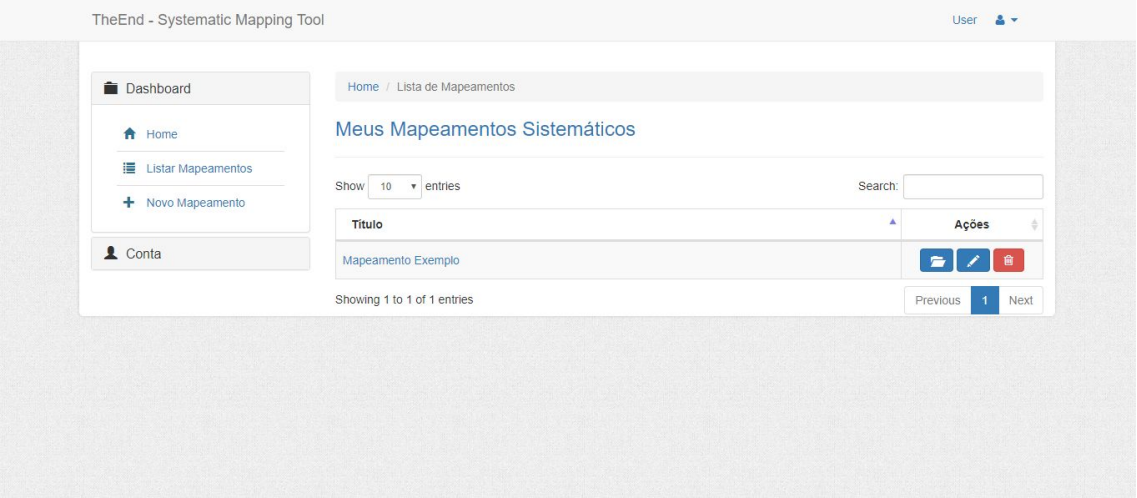



**Figura 1.5. Lista de mapeamentos**

Após a criação temos acesso as principais funcionalidades para podermos usar o mapeamento mostrado na imagem abaixo.

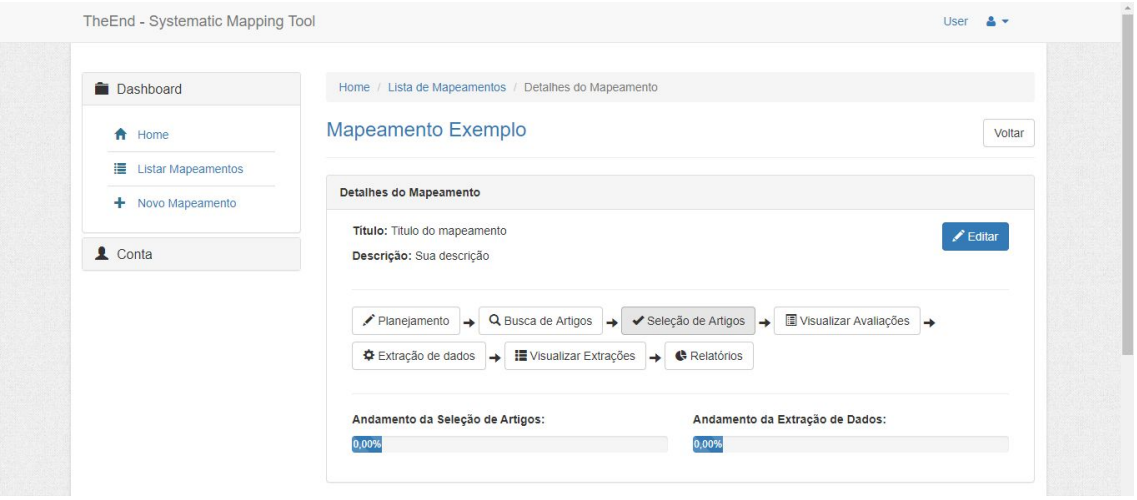



**Figura 1.6. Detalhes do mapeamento**

## References

KITCHENHAM, B. (2004). Procedures for performing systematic reviews. Keele, UK, Keele University, 33(2004):1–26.

KITCHENHAM, B.A., BRERETON, O.P., BUDGEN, D., Using Mapping Studies as the Basis for Further Research – A Participant-Observer Case Study. Information and

Software Technology, vol. 53, 2011, pp 638-651.

# Chapter 9

# Construindo um mini instagram em 4 horas com NodeJS

Marcos Paulo S. Barreto, Lucas B. M. Souza, Thaliane R. Gomes

***Abstract***

*This chapter provides a brief introduction to building an application using NodeJS and JavaScript. The application developed was based on the social network Instagram. This will explain what NodeJS is, general JavaScript features, how to install Node in the Linux and Windows environment, as well as some necessary utilities. Important concepts such as database usage and template engines will also be covered.*

***Resumo***

*Este capítulo traz uma breve introdução a respeito da construção de uma aplicação utilizando NodeJS e JavaScript. A aplicação desenvolvida foi baseada na rede social Instagram. Aqui será explicado o que é o NodeJS, características gerais do JavaScript, como instalar o Node no ambiente Linux e Windows e também alguns utilitários necessários. Conceitos importantes como utilização de banco de dados e templates engines também serão abordados.*

## 9.1. Introdução

O Node.js é uma tecnologia recente baseada no interpretador V8 *JavaScript Enginer* (código aberto da Google e que é ultilizado no Chrome), criada por Ryan Dahl em 2009, que surgiu pela necessidade de um ambiente de execução para aplicações *server-side*. É um ambiente de desenvolvimento de código aberto para a linguagem JavaScript.

Uma das principais características do Node JS é o fato de sua execução ser *single-thread*, ou seja, apenas uma *thread*(chamada de *Event Loop*) é responsável por tratar as requisições, de forma assíncrona, feitas ao servidor, poupando assim recursos computacionais e tornando-o essencial para atender um grande número de requisições.

O Node JS se destaca pela sua simultaneidade e velocidade, e isso atrai grandes empresas pois com a plataforma NodeJs a construção de aplicações web escaláveis, de alta

performance e real-time torna-se relativamente simples e fácil sendo ideal para ambientes de negócios [4]. Algumas empresas utilizam essa tecnologia para criação do seu aplicativo principal, dentre elas podemos citar Netflix, NASA, Uber, Walmart [3], entre outras.

As próximas seções deste capítulo detalham desde a instalação do Node.js, a linguagem programação usada por ele e preparação do ambiente para começar a desenvolver um projeto. Alguns elementos exclusivos do Node.js serão apresentados com devido foco ao decorrer das seções.

### 9.1.1. A linguagem JavaScript

JavaScript é uma linguagem de programação multi-paradigma, de alto nível, dinâmica, não tipada e interpretada. Originalmente desenvolvida Brendan Eich na década de 90 pelo fato dos browsers ainda serem muito estáticos. O JavaScript completou 20 anos em 2015 e tem sido uma das linguagens mais utilizadas no mundo desde então, o que impulsionou o seu crescimento e amadurecimento principalmente nos últimos anos. A sintaxe do JavaScript foi baseada na linguagem C, enquanto a sêmantica e o design vieram da Self e da Scheme[2].

## 9.2. Instalar e configurar o Node.js

Para começar a usar o Node JS, é necessário que seja feito o *download*, instalação e preparação do ambiente. As subseções seguintes apresentam o passo a passo para instalação em sistemas Linux e Windows.

### 9.2.1. Linux

Para efetuar a instalação no ambiente Linux, basta ir até o site oficial do Node JS na seção de *downloads*[1] e escolher a opção de acordo com a arquitetura do seu processador. Após fazer o *download*, navegue pelo terminal até a pasta onde está o arquivo compactado e descompacte-o com o comando `tar -xvf <nome do arquivo>`.

Para instalar via terminal basta usar `sudo apt install nodejs`. Depois de inserir a senha o processo de instalação será iniciado. Após a instalação, precisamos instalar o pacote gerenciador do Node, para isso use o comando `sudo apt install npm`. Para conferir a versão do node instalada execute `nodejs -v`. O retorno deverá ser algo como:

```
v8.10.0
```

### 9.2.2. Windows

Para o Windows, o *download* do Node JS pode ser feito através do mesmo *link*[1]. Escolha o instalador de acordo com a arquitetura do processador – 32-bits ou 64-bits –. Após baixar basta clicar duas vezes no arquivo baixado e seguir os passos dados na janela de instalação. Por padrão, você não precisa alterar nada, apenas siga clicando em *next*. Para conferir a versão do node instalada execute `node -v`.

[1] `https://nodejs.org/en/`

## 9.3. Criando o projeto

Após a instalação e configuração do nosso ambiente já podemos criar o nosso projeto no qual iremos trabalhar neste capítulo. Vale ressaltar que todos os programas que serão mostrados nas próximas seções têm foco na execução em ambiente Linux.

### 9.3.1. Como criar um novo projeto com Express

Para criar facilmente um novo projeto basta utilizar o framework *ExpressJS* que é minimalista, flexível e contém um robusto conjunto de recursos para desenvolver aplicações web, como um sistema de Views intuitivo (MVC), um robusto sistema de roteamento, um executável para geração de aplicações e muito mais[1]. A instalação do framework `npm install express-generator -g`.

Após a instalçao é necessário criar a pasta do projeto com o comando `express -v twig Minigram`. O *twig* é um Engine Template para modelagem segura, flexível e rápida, e Minigram é o nome do novo projeto.

Quando criamos o nosso projeto alguns arquivos e pastas também são criados automaticamente. Ao listar – `ls` – o conteúdo da pasta Minigram encontraremos os seguintes arquivos:

- **app.js**: Arquivo de configuração;
- **bin**: Diretório onde ficam os arquivos de configuração do *node.js*. É no arquivo `www` que é definida a porta na qual a aplicação irá funcionar;
- **package.json**: Arquivo JSON com informações e configurações do projeto criado;
- **public**: É nessa pasta que guardaremos todas as nossas imagens que vamos precisar para nossa aplicação (`\images`), também é onde vamos colocar toda a estilização das nossas páginas(`\stylesheets`) e também é onde teremos os nossos arquivos em *JavaScript*;
- **routes**: Pasta onde criaremos as rotas das nossas páginas;
- **views**: Pasta onde os nossos arquivos `.twig` ficarão armazenados, é nesses arquivos que são construidas as páginas em HTML.

### 9.3.2. Instalação das dependências do projeto

Após criar o projeto, é preciso instalar mais algumas dependências dentro da pasta. Com o comando `npm install` todas as dependências básicas serão instaladas..

Note que mais uma pasta foi criada (`\node_modules`) após instalar as dependências. Nela estão contidas algumas bibliotecas padrões das dependências do node.

### 9.3.3. Iniciando o Servidor

Para iniciar o servidor basta digitar `npm start` no terminal e a aplicação já estará rodando localmente. Porém existe um problema, com esse comando, sempre que for feita

alguma alteração nos arquivos é preciso reiniciar a aplicação. Dessa forma, será instalado o *Nodemon*, que é um utilitário para monitorar todas as alterações nos arquivos da aplicação e reiniciar automaticamente o servidor quando for necessário [5].

Para instalar o Nodemon basta rodar o comando `npm install -g nodemon`. Pronto, utilitário instalado e pronto para ser usado. Agora é possível rodar o servidor com o comando `nodemon -e twig,js`, feito isso, não é preciso nos preocupar em sempre estar reiniciando a aplicação, pois o nodemon cuidará dessa parte.

## 9.4. Introdução a rotas e templates

Depois que criar o projeto e rodar o servidor, para visualizar a aplicação pode-se acessar o *localhost* na porta 3000: `localhost:3000`. Inicialmente apenas mostra uma página de bem-vindo, isso porque ainda não foi criado outros *templates* para serem mostrado. Os *templates* são códigos em HTML, CSS, JavaScript e arquivos. Essa composição do template permite mostrar algumas informações e estilização na página WEB. Já as rotas são as *urls* que queremos associar a um *template*.

### 9.4.1. Análise e alterações do código gerado automaticamente

O aquivo que será utilizado para criar as rotas é o *index.js* que se localiza na pasta `routes`. O conteúdo que existe no arquivo é o seguinte código:

```
1 var express = require('express');
2 var router = express.Router();
3
4 /* GET home page. */
5 router.get('/', function(req, res, next) {
6   res.render('index', { title: 'Express' });
7 });
8
9 module.exports = router;
```

Esse código define que quando a url "/", que é a rota raiz, for acessada, irá renderizar(mostrar uma página web completa) o template index, em que mostra o texto de bem-vindo.

Na listagem acima é utilizado a declaração de variáveis com *var* e a função é declarada com *function*, nos códigos que serão mostrados adiante iremos adotar a declaração com *let*, por conta que mesmo no escopo de um função toda declaração feita com *var* é global e as funções que fazem parte da definição das rotas iremos usar *arrow functions* pois propociona uma melhor visualização do código[6]. Alterações:

```
1 let express = require('express');
2 let router = express.Router();
3
4 /* GET home page. */
```

```
5 router.get('/', (req, res, next)=> {
6   res.render('index', { title: 'Express' });
7 });
8
9 module.exports = router;
```

### 9.4.2. Construindo as páginas index, cadastro, login e publicação de foto

Na construção dos templates envolvendo as ferramentas para o *frontand* foi utilizado apenas o *HTML*, deixando de lado o *CSS* e códigos em *JavaScript*, pois o objetivo é mostrar como a fremework *NodeJs* funciona. Foi usado o *CSS* apenas para trabalhar com filtros nas imagens.

Então o primeiro passo é personalizar o conteúdo da página principal, editar o conteúdo do arquivo *index.twig* que está na pastas *views* e colocar o seguinte trecho de código em *HTML*:

```
1 {% extends 'layout.twig' %}
2
3 {% block body %}
4     <h1>Minigram:</h1>
5     <p> NodeJs    vida, ser  construdo com ele uma aplica  o para
6     <a href="/login"> Login</a>
7     <a href="/cadastro"> Cadastre-se</a>
8 {% endblock %}
```

É utilizado um título fixo para todas as páginas, isso vai da preferência de cada desenvolvedor. A definição do título está no arquivo textitlayout.twig, pode ser colocado somente Minigram:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Minigram</title>
5      <link rel='stylesheet' href='/stylesheets/style.css' />
6    </head>
7    <body>
8      {% block body %}{% endblock %}
9    </body>
10 </html>
```

No arquivo *index.twig* foi colocado dois links, porém para funcionar da forma correta, ainda deve-se criar as rotas *cadastro* e *login*. Nesse código é aproveitado para criar a rota de realizar publicação:

```
router.get('/login', (req,res)=>{
    res.render('login')
})

router.get('/cadastrar', (req,res)=>{
    res.render('cadastro')
})

router.get('/publicar', (req,res)=>{
    res.render('publicar')
})
```

Como foi definido que quando aquelas rotas fossem acessadas seria renderizado um template, então é só criá-los na pastas `views`.

Crie o arquivo *login.twig* e adicione o código *HTML*:

```
{% extends 'layout.twig' %}

{% block body %}
    <h1> Login</h1>
    <form method="post">
        <label> Username: </label>
        <input type="text" name="username">
        <br>

        <label> Senha</label>
        <input type="password" name="senha">
        <br>

        <button type="submit"> Enviar</button>
        <a href="/cadastro"> cadastre-se</a>
    </form>
{% endblock %}
```

E no arquivo *cadastro.twig*:

```
{% extends 'layout.twig' %}

{% block body %}
    <h1> F a  a  o seu cadastro </h1>
    <form method="post">
        <label> Nome Completo </label>
        <input type="text" name="nome_completo">
        <br>
        <label> Nome de Usu rio </label>
```

```
10          <input type="text" name="nome_de_usuario">
11          <br>
12          <label> Email </label>
13          <input type="text" name="email">
14          <br>
15          <label> Senha </label>
16          <input type="password" name="senha">
17          <br>
18          <button type="submit"> cadastro</button>
19      </form>
20  {% endblock %}
```

O publicar terá uma caixa de seleção para escolher entre alguns filtros. Ficará da seguinte forma:

```
1   {% extends 'layout.twig' %}
2
3   {% block body %}
4       <h1> Publicar Foto </h1>
5       <form method="post" enctype="multipart/form-data">
6           <label> Legenda </label>
7           <input type="text" name="legenda">
8           <br>
9           <label> Foto </label>
10          <input type="file" name="foto">
11          <br>
12          <label> Localiza  o </label>
13          <input type="text" name="localizacao">
14          <br>
15          <label> Filtro </label>
16          <select name="filtro">
17              <option value="sem_filtro"> sem filtro </option>
18              <option value="1977"> 1977 </option>
19              <option value="aden"> aden </option>
20              <option value="brannan"> brannan </option>
21              <option value="brooklyn"> brooklyn </option>
22              <option value="clarendon"> clarendon </option>
23          </select>
24          <br>
25          <button type="submit"> Enviar</button>
26      </form>
27  {% endblock %}
```

Observe que que está sendo utilizado na tag <*form*> com atributo "method" igual a "post", isso porque a página fará uma submissão de dados, então é necessário para identificação. Na próxima seção será mostrado como pegar esses dados. Em publicar

temos também o atributo *enctype* igual "multipart/form-data", pois o nessa página será feito *upload* de imagem.

### 9.4.3. Pegando os dados da requisição

Nos templates ficou tudo pronto para enviar dados, agora é necessário criar uma função para recuperar esses dados no arquivo *index.js* em *routes*:

```
router.post('/login', (req,res)=>{
    let usuario = req.body.nome_de_usuario
    let senha = req.body.senha
    res.redirect('/publicar')
})

router.post('/cadastro', (req,res)=>{
    let nome_completo = req.body.nome_completo
    let nome_de_usuario = req.body.nome_de_usuario
    let email = req.body.email
    let senha = req.body.senha
    res.redirect('/login')
})
router.post('/publicar', (req,res)=>{
    let legenda = req.body.legenda
    let localizacao = req.body.localizacao
    let filtro = req.body.filtro
    res.redirect('/publicar')
})
```

Note que para pegar os dados tem que usar o método *post* do objeto *router*[7]. Os dados estão sendo guardados nas variáveis, menos a foto, pois ainda será feito o processo de *upload* de aquivo, nas seções subsequentes será mostrado como guardar esses dados no banco de dados e como terminar o processo de upload. Depois que o processo é realizado o usuário será redirecionado para outra página utilizando o método *redirect*.

## 9.5. Salvando dados no bando de dados Mysql

O MySQL, lançado em 1995, é considerado um dos mais populares bancos de dados. Pode-se destacar inúmeras vantagens do MySQL, como a compatibilidade com os diversos sistemas operacionais, entre eles o Linux, Windows e Mac. Devido ao seu elevado desempenho, uso gratuito, ser multiusuário, robusto e seguro, o Mysql conquistou muitas empresas e desenvolvedores iniciantes.

O mysql consiste num Sistema Gerenciador de Banco de Dados (SGBD), cujo trabalha com a linguagem SQL. A sigla sql simboliza “Structured Query Language”, ou Linguagem Estruturada para Pesquisas, utilizada como padrão nos diversos bancos de dados de modelo relacional. Neste modelo, os dados são registrados em tabelas, as quais se relacionam entre si. As vantagens deste SGBD consistem em:

- Segurança;
- Integridade do banco de dados;
- Desempenho;

Por conta das vantagens apresentadas utilizaremos neste projeto o mysql como SGBD.

### 9.5.1. Instalação e configuração do Mysql

O primeiro passo é instalar o mysql na máquina onde o projeto será executado. No linux utilizamos o apt-get para instalar os pacotes, onde é necessário instalar o mysql client e server e o apache2. Segue o comando de instalação: *sudo apt-get install mysql-client mysql-server apache2*. Com o mysql instalado e configurado na máquina basta agora instalar a biblioteca no projeto node para que assim seja possível acessar todos os recursos do SGBD. A biblioteca utilizada é a *mysql2/promise* que é instalada pelo *npm* da seguinte forma: *npm install mysql2/promise -g*.

### 9.5.2. Utilizando o mysql no projeto

O primeiro passo para utilizar o SGBD no projeto é importá-lo, então utilizamos o *require* para importar a biblioteca instalada.

```
1 const mysql = require("mysql2/promise")
```

Após importamos a biblioteca é necessário criar uma função de conexão, onde a mesma se conecta na base de dados criada. Antes de tudo é necessário criar a base de dados, que é bem simples. A base de dados consiste em apenas duas tabelas, a *usuario*(com os campos id, nome, username, email e senha) e a *publicacoes* (com os campos localizacao,legenda,filtro,foto e uma chave estrangeira de usuario). Com a base de dados criada podemos acessá-la com a função abaixo.

```
1  async function connectToDB(req,res,next){
2
3      try{
4          router.db = await mysql.createConnection({
5              host:'localhost' ,
6              user:'minigram' ,
7              password:'123' ,
8              database:'meu_minigram'
9          })
10         console.log("CONNECTED...")
11
12     }
13     catch(error){
14         console.log("NOT CONNECTED!")
15     }
16
```

```
17      next()
18  }
19  router.use(connectToDB)
```

Nesta função fazemos a conexão com o banco de dados e utilizamos uma funcionalidade muito importante do javascript que é o *asyn* e *await*.

#### 9.5.2.1. Async e Await

Async/Await é uma das novas funcionalidades do ES2017. Com ela, é possível escrever códigos assíncrono como se estivéssemos escrevendo código síncrono. Utilizamos essas funcionalidades em funções de extrema importância em nosso código para garantir que as mesmas sejam executadas por completo.

#### 9.5.2.2. Executando operações no Banco de Dados

A variável *db* do nosso objeto *router* contém tudo que é necessário para as operações no banco de dados. Iremos utilizá-la para executar operações de inserção e seleção.

```
1  router.post('/publicar',uploadFile.single('img'),async(req,res)=>{
2      const local = req.body.local
3      const legenda = req.body.legenda
4      const filtro = req.body.filtro
5      const fileNameImage = req.file.filename
6      const user = req.session.userId
7
8      try{
9
10         await router.db.execute(`INSERT INTO publicacoes(
11             localizacao, legenda,filtro,foto,usuario) VALUES(?,?,?,?,?)
12
13         res.redirect('/feed')
14     }catch(error){
15         console.log(error)
16     }
17
18 })
```

Na linha 10 do código acima utilizamos o método *execute* para inserir os valores recuperados no *post* para salvá-los na base de dados.
Outra operação que podemos realizar é a recuperação de dados do banco utilizando o método *select*.

```
1  router.get('/feed',async(req,res)=>{
2
```

```
3       const[publicacoes, col] = await router.db.execute(`SELECT*FROM publ
4
5       res.render('feed',{
6           buttom: "Publicar",
7           urlPath: '/publicar',
8           publicacoes:publicacoes,
9           user: req.session.user
10      })
11  })
```

Com a função acima recuperamos todas as publicações do usuário logado, guardamos em uma lista chamada publicacoes e a mandamos para o contexto do template para que possa ser renderizada no *browser*.

### 9.6. Uploads de arquivos com o multer

Para realizar o *upload* de imagens utilizamos no projeto do minigram a biblioteca *multer*. A estratégia adotada foi realizar o *upload* da foto selecionada pelo usuário para um diretório público no servidor, chamado de `uploads`, e salvar apenas o nome da imagem no banco de dados. Antes de tudo é necessário instalar o multer com o npm, `npm install multer -g`, e depois importar a biblioteca no projeto. Feito isso é necessário criar um novo diretório na pasta `public` do projeto, nesse diretório ficarão todos os uploads.

```
1   const multer = require('multer')
2
3   const storage = multer.diskStorage({
4
5       destination : (req,file,cb)=>{
6           cb(null,"public/uploads")
7       },
8       filename: (req,file,cb)=>{
9           cb(null , file.originalname)
10      }
11  })
12  const uploadFile = multer({storage})
```

Com a função acima conseguimos realizar o *upload* de qualquer arquivo enviado pelo usuário. Com a função criada basta apenas chama-la nas rotas onde se deseja realizar o *upload*. Passando o método single definimos que o upload será de apenas um arquivo, e automaticamente quando o método post for executado a imagem selecionado pelo usuário será enviada para o diretório uploads criada na pasta public. E assim temos toda a base da nossa aplicação.

```
1 router.post('/publicar',uploadFile.single('img'),async(req,res)=>{
```

## References

[1] "Primeiros passos com Express em Node.js", http://nodebr.com/primeiros-passos-com-express-em-node-js/, Outubro de 2019.

[2] MORAES, W. B. Construindo aplicações com NodeJS. 2.ed. São Paulo: Novatec Editora Ltda, 2018. cap. 1.

[3] “Introdução ao Node.js: Básico ao Avançado com o Node.js“, https://www.devmedia.com.br/nodejs/, Outubro de 2019.

[4] Tilkov, S. and Vinoski, S. (2010). Node. js: Using javascript to build high-performance network programs. IEEE Internet Computing, 14(6):80–83.

[5] "Nodemon", https://nodemon.io/, Novembro de 2019.

[6] "Arrow Functions - JavaScript", https://expressjs.com/pt-br/guide/routing.html, Outubro de 2019.

[7] "Roteamento no express", https://expressjs.com/pt-br/guide/routing.html, Outubro de 2019.

Chapter
# 10

# Automatizando Tarefas Web com o Framework Selenium e Python

Lucas V. S. dos Santos, Rubenilson S. Lima, Samuel P. B. D. Lélis, Cícero N.A. do Ó, Carvalho Filho A. O. e Rafael Luz Araújo

***Abstract***

*In web applications that have many functionalities, during daily work, the execution of each one of them can become a dull task for the user, so the automation of these tasks becomes a viable alternative. This requires the implementation of scripts to automatically perform actions previously performed by the user, thus ensuring the reliability that all actions will be performed equally without requiring human effort. In this scenario, the Python programming language along with the Selenium framework become a good choice for the development of web task automation scripts. The sections in this chapter introduce the language and framework for their most relevant concepts and aspects.*

***Resumo***

*Em aplicações web que possuem diversas funcionalidades, durante o dia-a-dia profissional, o ato de executar cada uma delas pode se tornar uma tarefa maçante para o usuário, de modo que a automatização dessas tarefas se torna uma alternativa viável. Para isso, torna-se necessário a implementação de scripts para executar automaticamente as ações que antes seriam realizadas pelo usuário, garantindo então a confiabilidade de que todas as ações serão igualmente realizadas sem exigir esforço humano. Neste panorama, a linguagem de programação Python junto ao framework Selenium tornam-se uma boa escolha para o desenvolvimento de scripts para automatização de tarefas em sistemas web. As seções desse capítulo introduzem a linguagem e o framework quanto aos seus conceitos e aspectos de maior relevância.*

## 10.1. Introdução

O ato de realizar uma tarefa repetitiva em um sistema web pode requerer paciência, atenção e proatividade por parte do usuário, a automação tem como objetivo identificar

essas tarefas que antes eram feitas de forma manual e criar uma solução para realiza-las sem que seja preciso intervenção humana, ou seja, de forma totalmente automática [1].

Por mais bem preparada que seja uma gestão, imprevistos sempre podem ocorrer, a automação de tarefas permite uma estrutura de padronização, garantindo que todas as atividades serão igualmente realizadas e sem erros. Outra vantagem da automação é a redução de custos, já que o objetivo é usar o mínimo de mão de obra humana possível[1].

As próximas seções deste capítulo detalham a construção de scripts para automação de atividades realizadas no browser utilizando a linguagem de programação Python e o framework Selenium, desde seus conceitos principais, passando pela instalação, preparação do ambiente e sintaxe básica.

### 10.1.1. Python

A linguagem de programação Python tem ganhado enorme destaque nos últimos anos. Grande parte do que caracteriza sua ascensão tecnológica deve-se a sua maior vantagem: simplicidade [2].

O Python surgiu com o objetivo de ser simples e acessível para seus usuários como forma de agilizar o processo de desenvolvimento. Sua implementação iniciou em 1989 por Guido van Rossum. Porém, apenas em 1991 teve sua primeira versão (0.9.0). No ano de 2000 a versão 2.0 foi anunciada e, em 2008, a versão 3.0 foi lançada com diversas mudanças em sua sintaxe [2].

Outras vantagens da linguagem Python incluem a sua portabilidade, sendo multi-plataforma e podendo ser utilizada em diversos sistemas. Além de também ter Interoperabilidade, ou seja, facilidade para se comunicar com outros sistemas ou linguagens. O Python também é uma linguagem multi uso, podendo-se desenvolver aplicações desktop, mobile ou web [3].

A linguagem Python a cada dia ganha mais força. A prova disso é o crescente número de empresas que a utilizam nos dias atuais, tanto empresas internacionais como nacionais. Entre as principais companhias que usam Python em suas infraestruturas, estão: Google, Instagram, Dropbox, Spotify, Pinterest, BitTorrent, Blender 3D entre outras [4].

### 10.1.2. Selenium

O framework Selenium trata-se de uma ferramenta criada para automatização de aplicações web para fins de testes, porém, não se limitando somente a isso. Tarefas web que demandem tempo e precisão do usuário também podem ser automatizadas utilizando essa ferramenta.

O Selenium WebDriver faz chamadas diretamente ao navegador utilizando o suporte à automação nativo. Assim os testes escritos com o *framework* são bastante realistas, pois chama diretamente o navegador.

O Selenium também tem suporte diversas linguagens de programação além de estar disponivel para quase todos os navegadores disponíveis no mercado, tais como: Google Chrome, Mozilla Firefox, Internet Explorer, Opera e dentre outros[5].

## 10.2. Instalação o Python

Para começar a programar na linguagem Python, é necessário ter a linguagem instalada. As subseções abaixo apresentam o passo a passo para instalação em sistemas Linux e Windows.

### 10.2.1. Linux

Por padrão o Python geralmente já vem instalado no ambiente Linux, porém na versão 2.7, para prosseguir é necessário atualiza-lo. Para instalar a ultima versão do Python no Linux e receber as futuras atualizações: Para instalar deve-se abrir o terminal e adicionar o repositório do programa com o comando:

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

Para atualizar o gerenciador de pacotes utiliza-se o comando:

```
sudo apt-get update
```

comando abaixo para instalar a ultima versão do Python3:

```
sudo apt-get install python3.7
```

Por último, deve-se verificar a instalação executando o comando `python3 -version`. O retorno deve ser algo como:

```
Python 3.6.8
```

### 10.2.2. Windows

Para o Windows, o *download* do Python pode ser feito através do *link*: python.org/downloads/. Logo na parte superior da página, o site irá lhe sugerir a versão mais recente para o sistema operacional em uso, clique no botão para fazer o download e execute o arquivo baixado. no instalador marque as opções na parte inferior *Install launcher for all users(recommended)* e *Add Python to PATH* e clique na opção *Install now*, que irá incluir o gerenciador de pacotes pip, necessário para a instalação do selenium. Ao final do processo, o python estará instalado e poderá ser acessado através do Windows PowerShell ou CMD através do comando `python`.

## 10.3. Instalação o Selenium no Linux

Para instalar o Selenium no ambiente Linux basta abrir o terminal e executar a seguinte linha de comando: `pip3 install selenium`.

### 10.3.1. Instalação dos drivers

São basicamente módulos instalados que irão nos permitir acessar e controlar os navegadores a nossa vontade. Cada navegador possui o seu próprio driver. Nesse caso irá ser utilizado o navegador Firefox.

### 10.3.2. Instalação do Geckodriver

O Geckodriver é um módulo para controlar o navegador Firefox. Para efetuar a instalação basta abrir o terminal e executar o seguinte comando para realizar o download:

```
Wget https://github.com/mozilla/geckodriver/releases/download/
v0.18.0/geckodriver-v0.18.0-linux64.tar.gz
```

Após fazer o download, descompacte o arquivo com o comando:

```
sudo tar -xvzf geckodriver*
```

Com a extração concluída, de permissão de execução com o comando:

```
chmod +x geckodriver
```

Com a instalação concluida, basta adiconar o programa ao PATH:

```
export PATH=$PATH:/path-to-extracted-file/geckodriver
```

Por último, verifique a instalação executando o comando:

```
geckodriver -version
```

## 10.4. Instalando o Selenium no Windows

Com a instalação padrão do python, se torna, também, disponível o gerenciador de pacotes pip, que usaremos para fazer a instalação do selenium. Para instalar o selenium basta executar a seguinte linha de comando no PowerShell:

```
pip install selenium
```

### 10.4.1. Instalação dos drivers

Assim como no Linux, é necessária a utilização de drivers para permitir o acesso e controle dos navegadores. No Windows, os drivers necessários são programas executáveis (.exe)

### 10.4.2. Instalação do Geckodriver

A instalação do geckodriver é bem simples no windows, basta acessar o endereço:

*https://github.com/mozilla/geckodriver/releases*

Ao fim da página estão os links para download e suas versões (32 ou 64 bits), basta fazer o *download* e extrair arquivo .zip baixado, resultando no executável *geckodriver.exe*.

## 10.5. Estrutura básica e sintaxe

Após a instalação e configuração já é possivel criar os primeiros programas. Nesta seção serão apresentados elementos que constituem a sintaxe básica do Python. Programas em Python podem ser desenvolvidos em qualquer editor de texto que dê suporte a codificação `UTF-8`. Todos os exemplos mostrados nas próximas seções tem foco na execução em ambiente Linux.

### 10.5.1. Hello World

Para o primeiro contato com a linguagem, pode-se iniciar com um tradicional *Hello, World*. Portanto, é necessário salvar um arquivo com título qualquer, sugere-se `hello.py`. O conteúdo do arquivo deve ser o seguinte trecho de código:

```
1 print("Hello World")
```

Para compilar o código acima, basta executar `python3 hello.py`. A saída para este programa será:

```
Hello World
```

Neste primeiro exemplo pode-se notar algumas caraterísticas da linguagem. A primeira delas é que Python não possui ponto-e-vírgula ou quaisquer ponto e acentuações ao final das instruções, diferentemente de linguagens como C ou Java. Outra característica é que não existe a necessidade de declarar uma função `Main` em Python.

### 10.5.2. Variáveis

Segundo Downey (2016) Python tem tipagem dinâmica, isso quer dizer que o tipo das variáveis declaradas durante a execução de um programa podem ser alterados. Python também traz em sua sintaxe uma forma de declaração limpa, o que facilita e torna rápido o desenvolvimento das aplicações. A declaração de variáveis em Python pode ser feita da seguinte maneira, primeiro é digitado o nome da variável, depois um sinal de igualdade e em seguida o valor que a variável irá assumir. No código abaixo serão representados os 5 principais tipos de variáveis existentes em Python.

```
1  # Tipo 1 Inteiro
2  variavel_a = 7
3
4  # Tipo 2 Flutuante
5  variavel_b = 5.5
6
7  # Tipo 3 String
8  variavel_c = 'Seu nome'
9
10 # Tipo 4 Boolean
11 variavel_d = True
12
13 # Tipo 5 Nulo
14 variavel_e = None
```

O *tipo 1* é usado para armazenar valores inteiros positivos e negativos, enquanto o *tipo 2* é usado para o armazenamento de valores reais (Que possuem casas decimais). Os valores que podem ser armazenado nos tipos *tipo 3*, *tipo 4*, *tipo 5* são caracteres( letras e nomes ), verdadeiro ou falso ( True, False ) e nulo ( Declaração de uma variável sem um valor pré-determinado ), respectivamente.

#### 10.5.2.1. Entrada e saída de dados

Com as funções `input() e print()` é possível interagir com o usuário em modo texto no estilo *input/output*. As funções principais de `input() e print()` tem por finalidade a entrada e saída de valores. O código abaixo mostra as funções de saídas mais comuns:

```
string = "Python"

print("ola", string)
print("ola {}".format(string))
print("ola %s"%string)
```

As variações da função de impressão acima exibem a mesma saída:

```
Ola Python
Ola Python
Ola Python
```

Para receber uma entrada de valor, deve-se usar a função `input()`. A leitura de um valor para uma variável é estabelecida colocando a variável e em seguida o operador de atribuição = precedido da função `input()`. É possível também colocar uma mensagem para o usuário:

```
var = input("Digite alguma coisa: ")
print(var)
```

Por padrão o Python considera todos os valores lidos como strings, caso for desejado ler um valor de um tipo diferente, é necessário forçar a sua conversão:

```
var = int(input('Digite um valor inteiro')
var = float(input('Digite um valor real')
var = str(input('Digite um valor string')
```

### 10.5.3. Estrutura de seleção if / elif / else

A estrutura de seleção *if / elif / else* tem a função básica de tomar decisões perante a execução de um programa. Em Python este tipo de estrutura recebe valores lógicos verdadeiro ou falso, sendo que qualquer valor numérico equivale a `True` e enquanto que valores 0 ou None equivalem a `False`. O código abaixo representa um modelo de uso do *if / elif / else*:

```
idade = 22

if idade <= 18:
    print("Menor ou igual a 18 anos")
elif idade > 25:
    print("Maior que 20 anos")
else:
    print("Entre 19 e 25 anos")
```

### 10.5.4. Estrutura de repetição for

Umas das estruturas de repetição em Python é o comando *for*. Ele é comumente usado para fazer iterações com listas e em outros objetos iteráveis. Em sua forma mais comum é especificado uma condição lógica e o bloco código se repetirá enquanto a condição seja satisfeita com verdadeiro:

```
1 for i in range(0,10):
2     print(i)
```

O código acima trata-se de um contador que incrementa e exibe na tela o valor da variável i de 0 ate 9.

#### 10.5.5. Estrutura de repetição while

Outra estrutura de repetição utilizada em Python é o comando `while`. Ele é utilizado quando há a necessidade de realizar um laço de repetição enquanto uma determinada condição for entendida.

```
1 i = 0
2 while i < 10:
3     print(i)
4     i += 1
```

O código acima trata-se de um contador que incrementa e exibe na tela o valor da variável i de 0 até 9. Diferente do `for`, é necessário incrementar o valor da variável contadora.

### 10.6. Estrutura de dados

Conforme Downey(2016) as estruturas de dados são tipos de variáveis especiais que podem ser entendidas como um conjunto ou listas de outras variáveis ou valores. Assim como em outras linguagens – como a linguagem C com seus vetores – as estruturas de dados são geralmente uma sequência de valores encadeados em ordem predefinida ou até mesmo apenas espaços separados não alocados para que depois venham ser preenchidos.

Em Python os padrões para tipos de sequência de dados são três: *Listas*, *Tuplas* e *Dicionários*. A grande diferença entre eles esta relacionada a maneira como podem ser utilizados. As *listas* são um conjunto de variáveis mutáveis, já as *tuplas*, tem a mesma função das *listas*, porém são imutáveis. Os dicionários tem certa semelhança com os anteriores, mas a característica que o diferencia dos demais é a chave, que está relacionada com um valor.

#### 10.6.1. Listas

As *Listas* são um conjunto de valores que podem ser de diferentes tipos, cada valor contém um índice para fim de futuras utilizações. A contagem dos índices são delimitados pelo tamanho da *lista*, que pode ser mutável com algumas funções próprias da linguagem. O primeiro elemento da *lista* possui índice 0, e o último elemento é sempre `len(lista) - 1`, considerando que o tamanho da lista se dá por `len(lista)`, retornando assim, a quantidade de elementos presente nesse conjunto. Podemos declarar uma *lista* em Python utilizando as seguintes formas:

```
1 lista = []
2 num = [1,2,3]
3 valores = ['jose', 25, 5.8]
4 listas = [1, 2, valores]
```

```
5
6 print(lista, num, valores, listas)
```

A saída para o código acima é: `[] [1, 2, 3] ['jose', 25, 5.8] [1, 2, ['jose', 25, 5.8]] .`

Percebe-se que não importa o tipo, Python permite que qualquer valor possa estar presente dentro de uma *lista*, assim, facilitando ao programador uma série de possibilidades que não é permitida em inúmeras outras linguagens.

Existe também a possibilidade de criar *listas* multidimensionais em que os valores são *listas* dentro de *listas*. Declara-se da seguinte forma:

```
1 matrizA = [[1,2],[3,4],[5,6]]
2 matrizB = [['jose',1],['maria',2]]
3
4 Print("Matriz A:", matrizA)
5 Print("Matriz B:", matrizB)
```

A saída para o código acima é:

```
matriz A: [[1, 2], [3, 4], [5, 6]]
matriz B: [['jose', 1], ['maria', 2]]
```

Como já foi mencionado, as *listas* permitem a inclusão de qualquer tipo no seu escopo, independente de ser uma *lista* ou uma *lista de listas* (matriz).

### 10.6.2. Tuplas

As *tuplas* são muito semelhantes as *listas*, a sua especificação que a difere das *listas* é o fato de seus valores serem fixos, ou seja, imutáveis. Para diferenciação de ambas, nas *tuplas* é utilizado parenteses ao invés de colchetes. Veja alguns exemplos abaixo:

```
1 tuplaA = (1,2,3)
2 tuplaB = ('arroz','feijao','batata')
3 tuplaC = ('a',2.5,'joao',9)
4
5 print(tuplaA, tuplaB, tuplaC)
```

A saída para o código acima é: `(1, 2, 3) ('arroz', 'feijao', 'batata') ('a', 2.5, 'joao', 9)`

Vale ressaltar que a linguagem permite a criação de uma *tupla* com *listas* dentro, e da mesma maneira inversamente, uma *lista* com *tuplas* dentro.

### 10.6.3. Dicionários

Todos as estruturas de dados que vimos até agora — listas e tuplas — são coleções sequenciais. Isto significa que os itens na coleção estão ordenados da esquerda para a direita e eles usam números inteiros como índices para acessar os valores que eles contêm.

Dicionário é um tipo diferente de coleção. Ele é um tipo de mapeamento nativo do Python. Um mapa é uma coleção associativa desordenada. A associação, ou mapeamento,

é feita a partir de uma chave, que pode ser qualquer tipo imutável, para um valor, que pode ser qualquer objeto de dados do Python. Veja abaixo alguns exemplos de dicionários:

```
1 contatos = {"Maria": "1234-5678", "Yudi": "4002-8922"}
```

A saída para o código acima é: `'Maria': '1234-5678', 'Yudi': '4002-8922'`

Caso o usuário digite `contato["Yudi"]` a saida será `'4002-8922'`, Pois se trata do nome dado a chave, e o resultado a ser exibido é o seu valor.

## 10.7. Funções

A linguagem Python também permite o desenvolvimento de funções, trechos de código que podem ser reutilizados a qualquer momento quando são chamados. Em Python a funções podem ser de dois tipos: Funções com retorno ou funções sem retorno.

### 10.7.1. Funções básicas

Em Python o padrão básico de declaração das funções se dá com a palavra `def` seguido do nome da função e dos possíveis valores de parâmetros. Um exemplo seria uma função que imprima uma simples frase:

```
1 def imprimirString():
2     print("Imprimindo uma frase")
```

### 10.7.2. Funções com parâmetros

Agora serão passados alguns argumentos para esta função, fazendo com que ela imprima um valor do tipo *string* e outro do tipo *int*:

```
1 def imprimirString(nome, idade):
2     print("Ola, meu nome eh {} e eu tenho {} anos".format
3     (nome,idade))
```

### 10.7.3. Funções com retorno

Para retornar valores em uma função não é necessário definir seu tipo. Basta apenas fazer o uso do `return`:

```
1 def simplesSoma(x, y):
2     return x + y
```

O retorno de funções em Python também podem retornar vários valores. Caso não seja especificado o tipo do retorno, será automaticamente retornado uma `Tupla` contendo todos os valores:

```
1 def retorno():
2     return 1, 2
```

O retorno dessa função será: `(1,2)`.

## 10.8. Introdução ao Selenium

Para que sejam iniciados os projetos com o Selenium, deve-se começar pelo básico. Atráves de certos comandos da framework, será iniciada a partir daqui a introdução ao Selenium para a manipulação de diversas tarefas no browser.

### 10.8.1. Abrindo uma página

Para abrir uma pagína web com é necessário utilizar seguinte código:

```
1 from selenium import webdriver
2
3 firefox = webdriver.Firefox()
4 firefox.get('http://google.com.br')
```

Para melhor entendimento, segue a análise do código acima. Na primeira linha o webdriver está sendo importado, ele é o módulo responsável por provê implementações para diferentes browsers. Nos exemplos mostrados aqui, será utilizado o "Mozilla Firefox", pois não precisa de nenhuma configuração adicional, basta que ele esteja instalado.

Logo a seguir, será criada uma instância chamada `firefox` e depois será invocado o metódo get passando como parâmetro a URL da página que se deseja abrir.

```
1 firefox = webdriver.Firefox()
2 firefox.get('http://google.com.br')
```

### 10.8.2. Manipulando elementos

Uma página web é composta por vários elementos, sendo assim, é necessário aprender a manipulá-los. Primeiramente, será visto como encontrá-los. Para isso, um conhecimento em HTML é necessário a fim de facilitar a manipulação da página.

Para encontrar um elemento pelo seu id, é utilizado o método `find_element_by_id`:

```
1 find_element_by_id('<id>')
```

Caso seja necessário encontrar todos os elementos que possuem uma classe específica, é utilizado o metódo find_elements_by_class_name.

```
1 find_elements_by_class_name('<class_name>')
```

Existem vários outros métodos que podem ser utilizados para uma função especifica , todos eles estão presentes na documentação. A seguir estão alguns dos mais conhecidos:

```
1 # Encontrar elemento pelo ID
2 find_element_by_id('<id>')
3
4 # Encontrar elemento pelo atributo name
5 find_element_by_name('<name>')
6
7 # Encontrar elemento pelo texto vis vel
8 find_element_by_link_text('<text>')
```

```
9
10 # Encontrar elemento pelo seu seletor css
11 find_element_by_css_selector('<css_selector>')
12
13 # Encontrar elementos pelo nome da tag
14 find_elements_by_tag_name('<tag_name>')
15
16 # Encontrar elementos pela classe
17 find_elements_by_class_name('<class_name>')
```

### 10.8.3. Trabalhando com formulários

Também é possível trabalhar e manipular formulários na internet com o uso de Selenium, como por exemplo usar um buscador como o Google:

```
1  from selenium import webdriver
2  from selenium.webdriver.common.keys import Keys
3
4  # inicia o navegador e abre direto na pagina do Google
5  firefox =  webdriver.Firefox()
6  firefox.get('http://google.com.br/')
7
8  # encontra o campo de busca na pagina
9  busca = firefox.find_element_by_name('q')
10
11 # Digitar no campo de busca: Selenium
12 busca.send_keys('Selenium')
13
14 # faz com que a tecla enter seja pressionada
15 busca.send_keys(Keys.ENTER)
```

## 10.9. Considerações Finais

Neste capítulo foram apresentados conceitos gerais sobre a automação de tarefas web e a linguagem de programação Python, conceitos básicos necessários a uma boa utilização da ferramenta Selenium, Que possibilita o desenvolvimento de *scripts* de maneira simples, visto que a interação com o navegador é feita por meio de interface gráfica, simplificando as tarefas de automação.

## References

[1] LEUCOTRON.“Automação de tarefas: por que ela se tornou uma necessidade?”, https://blog.leucotron.com.br/automacao-de-tarefas-por-que-ela-se-tornou-uma-necessidade/ Outubro de 2019

[2] Leone ,L.(2016) “Por que aprender Python pode te levar mais longe na carreira¡‘, https://becode.com.br/porque-aprender-python/, Outubro de 2019

[3] Betopyt,(2009) “Vantagens e desvantagens do Python“, https://dicasdepython.wordpress.com/tag/vantagens-e-desvantagens-do-python/, Outubro de 2019

[4] Python Brasil. Empresas que usam Python. Página inicial. Disponível em <https://python.org.br/empresas/>. Acesso em: 27 de out. de 2019.

[5] SeleniumHQ. What is Selenium?. Página inicial. Disponível em <https://www.seleniumhq.org/>. Acesso em: 27 de out. de 2019.

[6] Panda. Dicionários. Disponível em <https://panda.ime.usp.br/pensepy/static/pensepy/11-Dicionarios/dicionarios.html>. Acesso em: 29 de Out. de 2019.

[7] Python Club. Selenium - O que você deveria saber. Disponível em <http://pythonclub.com.br/selenium-parte-1.html>. Acesso em: 31 de Out. de 2019.

[8] Downey, Allen B. "Pense em Python." Sao Paulo: Novatec 2016).