Ação de Extensão: Inteligência Artificial em *Python*

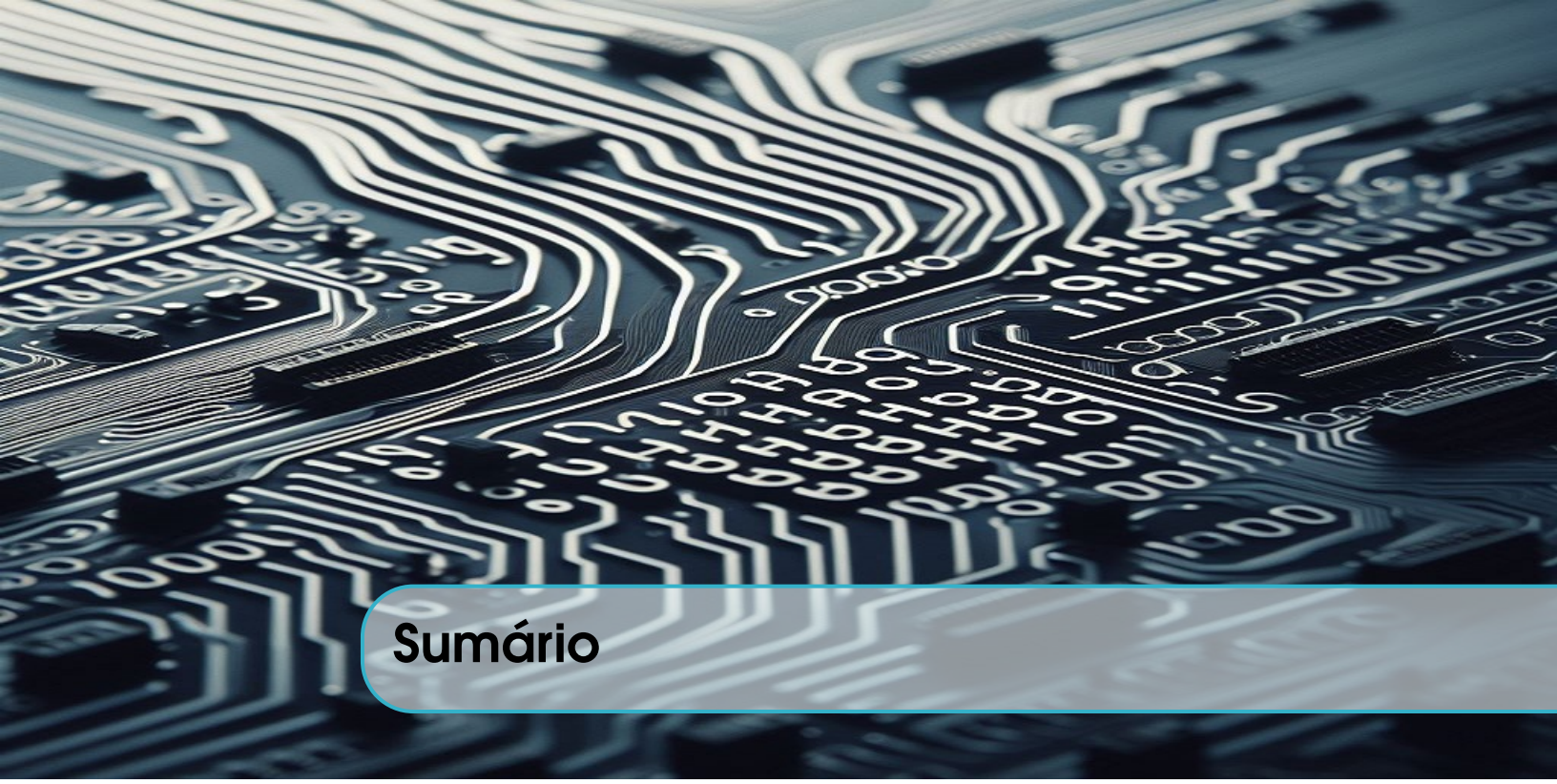
Práticas de Laboratório

Prof.: Dr. Sidney Marlon Lopes de Lima

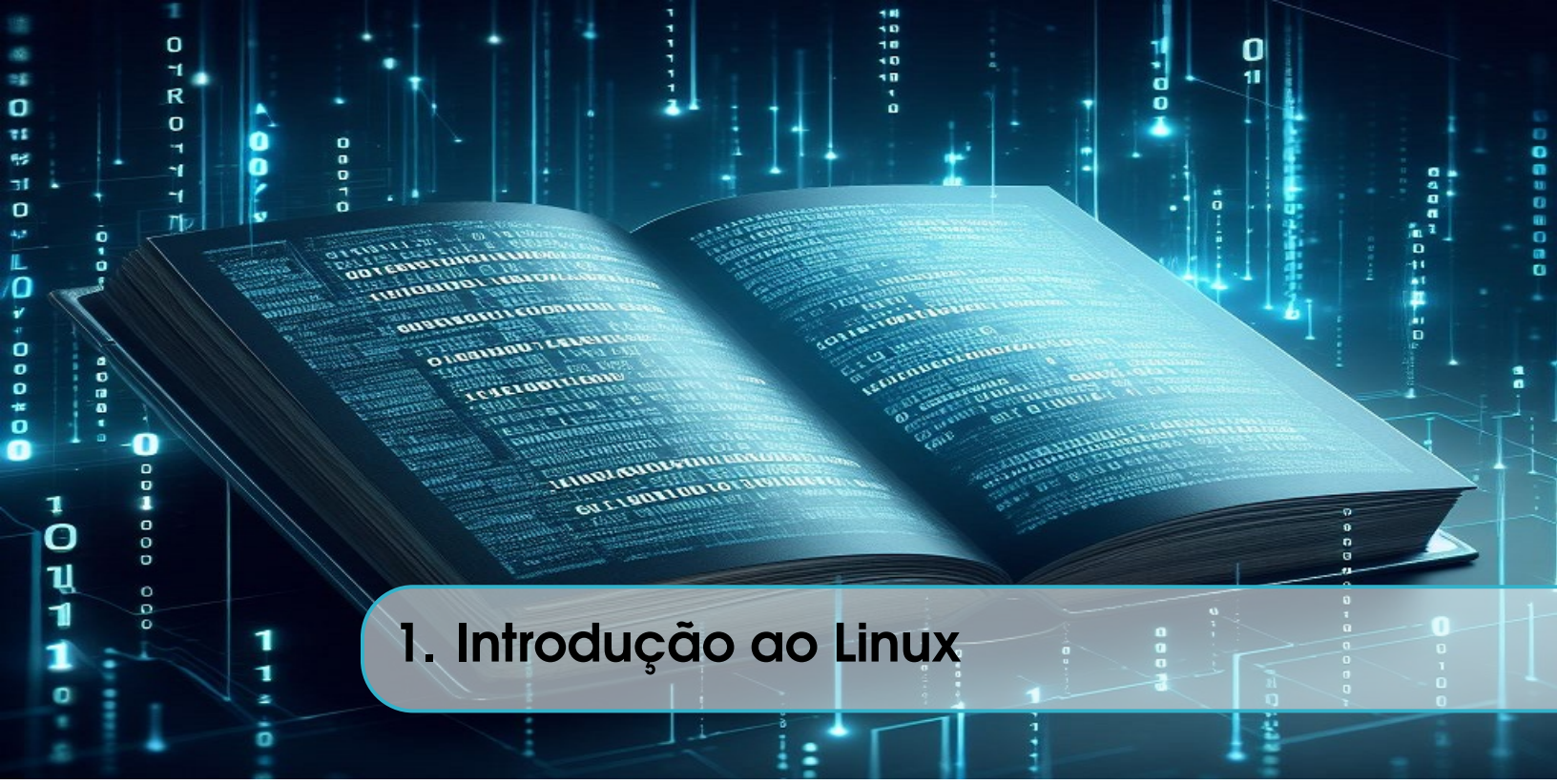
Aluna: Gabriela Leite Pereira

Aluno: Lucas de Souza Santos

DEPARTAMENTO DE ELETRÔNICA E SISTEMAS, UNIVERSIDADE FEDERAL DE PERNAMBUCO
Esta apostila tem por objetivo auxiliar a execução de atividades propostas durante as práticas da ação de extensão vinculada ao Edital 01/2023 - Credenciamento das ações de extensão com ou sem movimentação financeira. Todas as imagens de capas foram geradas através de inteligência artificial.
Primeiro lançamento, Março 2024



Sumário



1. Introdução ao Linux

Lucas de Souza Santos
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Engenharia Elétrica - Universidade Federal de Pernambuco, Recife, Pernambuco. lucas.ssantos2@ufpe.br

Victor Carlos Barbosa Galindo Gomes
Currículo *Lattes*: http://lattes.cnpq.br/7022445008053107
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. victor.carlos@ufpe.br

Dr. Sidney Marlon Lopes de Lima
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. sidney.lima@ufpe.br

1.1 O que é o *Linux* e o que é o kernel *Linux*

O *Linux* é um sistema operacional feito por Linus Torvalds nos anos 90, ele é *open source*, o que significa que seu código é aberto e pode ser modificado por qualquer pessoa. O sistema pode ser encontrado em vários dispositivos e também pode ser encontrado em várias distribuições para computadores, notebooks e *smartphones*. As vantagens são muitas em ter um dispositivo com *Linux*, já que o usuário pode modificá-lo, conforme suas necessidades, é por isso que existem inúmeras distribuições, cada uma focada em necessidades específicas.

1.2 Por que *Linux*?

A depender da aplicação alvo, é preferível empregar uma distribuição *Linux* em detrimento do Windows. Uma etapa fundamental referente à criação de inteligência artificial diz respeito à extração de atributos das amostras reservadas à etapa de treinamento. Suponha um antivírus baseado em

inteligência artificial. Faz-se necessário extrair atributos dos aplicativos suspeitos. As distribuições *Linux* assumem relevância nesse sentido.

Há várias distribuições especializadas na extração de características de aplicativos maliciosos. Por outro lado, o Windows não apresenta qualquer aplicativo nativo visando a extração de atributos de aplicativos suspeitos. Como agravante, o Windows impede a instalação de vários extratores de características, preventivamente. Por se tratar de um sistema operacional de propósito geral, o Windows parte do princípio de que seus usuários não têm consciência explícita de suas ações. Portanto os mecanismos de defesa do Windows impendem, de forma compulsória, a instalação de grande parte das ferramentas visando a auditória de aplicativos maliciosos (e.g.: vírus, verme, cavalo de troia...).

As distribuições *Linux* apresentam grande potencial quanto aos principais pilares da Indústria 4.0. Há inúmeras distribuições *Linux*, cada uma focada em necessidades específicas. Enfatiza-se que não é trivial instalar uma aplicação concebida para *Linux* em ambiente Windows. Seriam necessários compiladores cruzados, além da incorporação de novas diretivas e bibliotecas visando a adaptação de um código-fonte feito para *Linux* funcionar no Windows.

1.3 Distribuições (Distro) *Linux*

As distribuições *Linux* correspondem à qualquer ambiente que use o *kernel Linux* e programas GNU para construir seus próprios Sistemas Operacionais. Essas distribuições, ou simplesmente distro, possuem filosofias diferentes e ambientes *Desktop* diferentes. Algumas distros se baseiam em outras distribuições *Linux* para criar suas próprias versões, um exemplo é o *Linux Mint* que é baseada nas versões do Ubuntu que este é baseada na distro *Debian*. Alguns nomes de distribuições *Linux* famosas:

- Ubuntu
- *Linux* Mint
- Debian
- Fedora
- CentOs
- *Kali Linux*

Caso exista dúvida de qual distro deva ser utilizada, recomenda-se os conselhos do seguinte site `https://distrowatch.com`. Ele traz informações sobre muitos Sistemas Operacionais *Linux* e mostra a visão inicial de cada ambiente *Desktop*. Há também alguns servidores nas nuvens que oferecem uma boa experiência para testar o SO *Linux*, testar comandos, instalar programas ou desinstalar também. Um servidor *on-line* capaz de introduzir o *Linux* é o *OnWorks* disponível no seguinte endereço: `https://www.onworks.net`. Um serviço *on-line Linux* é capaz de ajudar a ter uma experiência prévia no manuseio das atividades essenciais em ambiente *Desktop*. Na presente apostila, a distribuição *Kali Linux* será utilizada. O *Kali* visa testes de intrusão avançados e auditoria de segurança da informação. O *Kali* possui mais de 600 testes de penetração.

Siga as instruções:

1. Configuração do *Kali Linux* da máquina virtual.
   - Faça o *download* do *Kali Linux* para VirtualBox no seguinte link: `https://www.kali.org/get-kali/#kali-virtual-machines`.
   - Faça a extração da imagem do *Kali Linux*. Clique com o lado direito do mouse e escolha **"Extrair Tudo..."**.

- Fig. 1.1: após a descompactação, execute no arquivo de extensão ***Virtual Machine Definition***.
- Fig. **??**: por padrão, usuário **kali**, senha **kali**.
- Fig. 1.2: execute o terminal do ***Kali Linux***.

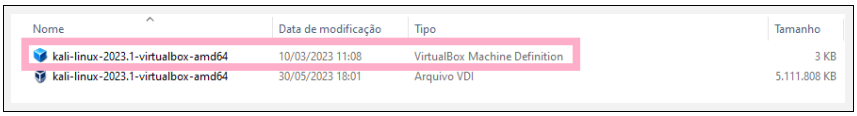


Figura 1.1: Após descompactação, clique no arquivo de extensão ***Virtual Machine Definition***.

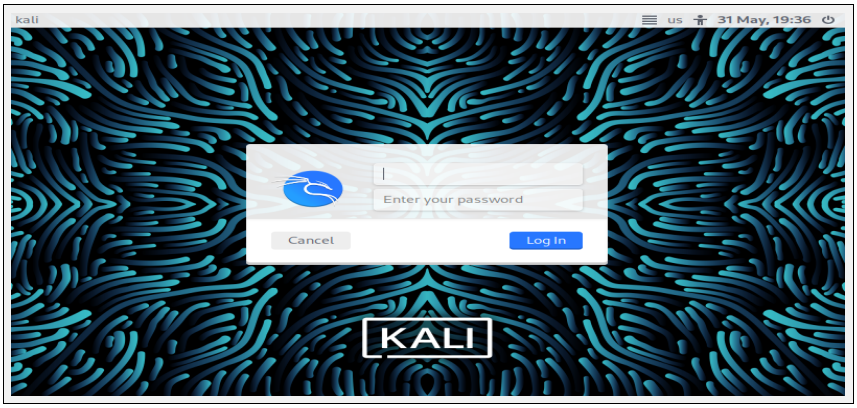


Figura 1.2: Por padrão, usuário **kali**, senha **kali**.

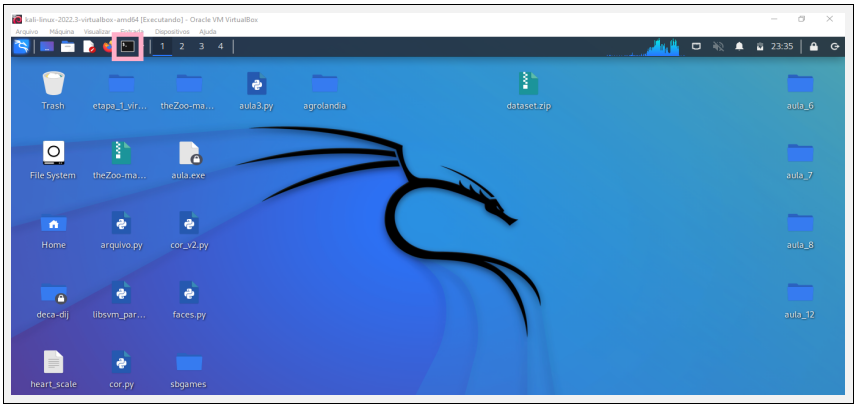


Figura 1.3: Execute o terminal do ***Kali Linux***.

1.4 Configuração do Teclado no Padrão ABNT2 no *Linux*

Por padrão, os caracteres especiais não serão reconhecidos. Faz-se necessário configurar o teclado no padrão ABNT2 de modo a haver o reconhecimento das palavras acentuadas contendo; acento circunflexo, til, crase e acento agudo.

No Terminal:

```
setxkbmap -model abnt2 -layout br
```

1.5 O que é o *bash*

O *bash* ou *Bourne Again shell* é um interpretador de comandos do *Linux* que substituiu o antigo interpretador Sh o *shell Bourne*. O *shell* nos permite interagir com o sistema operacional. Assim como as linguagens de programação, o *bash* nos permite criar variáveis, controle de fluxo e criar arquivos executáveis para automatizar tarefas.

Muitos tem uma leve confusão ao achar que o *bash* e o terminal são o mesmo programa, mas na realidade não são. O terminal é uma tela, normalmente preta, onde são digitados comandos e recebidos (impressos) saídas desses comandos. O *shell* é o interpretador desses comandos; então quando se digita um comando "cd" para entrar em algum diretório, é no terminal que se deve digitar, mas é o *shell* que irá interpretá-lo.

1.6 Caminhos relativos e Absolutos no *Linux*

Caminho é a localização de um diretório ou arquivo no sistema de arquivos, por exemplo, quando digitamos o comando "pwd" no terminal, esse comando retorna "/home/kali", esse é o caminho para o diretório "kali". Esses caminhos podem ser relativos ou absolutos. Vejamos alguns exemplos a seguir.

1.7 Caminho Absoluto

No Terminal:

```
Comando:
    pwd
Saída:
    /home/kali
```

Explicação:

- "/" é o diretório raiz do sistema;
- "home/" é o diretório onde fica localizado os diretórios dos usuários do sistema "kali" é o nome do diretório do usuário kali. Esse caminho é um caminho absoluto, desde o diretório raiz "/" até o diretório de destino o "kali".

1.8 Caminho Relativo

Por exemplo, estamos na pasta "/home/kali", dentro dessa pasta existem os diretórios *Desktop*, *Documents*, *Downloads*, *Music*, *Pictures*, *Public*, *Templates* e *Videos* Podemos entrar no diretório *"Downloads"* usando o caminho absoluto ou relativo das seguinte maneiras:

No Terminal:

```
    pwd
    ls
    cd /home/kali/Musicas
    pwd

Saída:
    /home/kali
    Desktop   Documents   Downloads   Music   Pictures    Public  Templates
    Videos
    /home/kali/Downloads
```

Explicação:

- *pwd*: lista o diretório atual, nesse exemplo "/home/kali", onde o usuário está;
- *ls*: lista todos os arquivos e diretórios de onde o usuário estar;
- *cd*: faz o usuário entrar ou sair no diretório especificado pelo caminho.

No lugar do caminho absoluto, que escrevemos acima, pode-se digitar apenas *"cd"* mais o nome do diretório que você quer entrar e se encontra dentro do diretório atual, no caso, esse é o caminho relativo da pasta. Por que digitar apenas o nome da pasta, após o comando *"cd"* irá funcionar ? Por que o diretório que você deseja ir, está localizado no diretório atual onde você se encontra. Agora, caso você queira ir para algum diretório, que não esteja localizado dentro do diretório que você está localizado, nessa situação será preciso escrever o caminho absoluto e não o relativo.

```
Comando no Terminal:
    pwd
    ls
    cd Downloads
    pwd
    cd ..
    pwd


Saída:
    /home/kali
    Desktop   Documents   Downloads   Music   Pictures    Public  Templates
    Videos
    /home/kali/Downloads
    /home/kali
```

Exemplo: Dentro do diretório “kali” localizado em “/home/” existem os seguintes diretório: *Desktop*, *Documents*, *Downloads*, *Music*, *Pictures*, *Public*, *Templates* e *Videos*

Exercícios

1. Use o comando *pwd* para o localizar o diretório que você se encontra;
2. Use o comando *cd* e entre no diretório de *Desktop*;
3. Use o comando *cd ...* para sair diretório de *Desktop*;
4. Por fim, vá para o diretório *Downloads*.

```
Comando no Terminal:
    pwd
    cd Desktop  ou cd /home/kali/Desktop
    pwd
    cd ..
    cd Downloads /home/kali/Downloads
    pwd

Saída:
    /home/kali
    /home/kali/Desktop
    /home/kali/Downloads
```

1.9 Comandos

Comando é uma instrução que damos ao computador, esperando que ele nos responda fazendo o que queremos. Se quisermos criar um novo diretório, usamos o comando “mkdir”. Exemplo: Crie um novo diretório dentro de “/home/kali” chamado aula.

```
Comando no Terminal:
    pwd
    ls
    mkdir  /home/kali/filmes ou mkdir filmes
    pwd
    ls

Sáida:
    /home/kali
    Desktop   Documents   Downloads  ...    Videos
    /home/kali
    aula    Desktop   Documents   Downloads   ...   Videos
```

1.10 Diretórios no *Linux*

É comum que muitos que migraram do Sistema Operacional do Windows para o *Linux*, estranhem um pouco os sistemas de pastas do novo sistema. Os nomes são totalmente diferentes dos dados do outro SO. Onde fica o Disco C ou os "Arquivos de Programas" ? Vamos explorar um pouco sobre a hierarquia dos diretórios no sistema pinguim.

Diretórios no *Linux* são pastas. Podemos criar, excluir, mudar o nome, listar tudo que existe dentro e muito mais. No *Linux*, os diretórios têm uma hierarquia, podendo ser chamado de pai e filho. Os diretórios que estão dentro de outros, serão chamados de diretórios filhos; já os diretórios que guardam outros diretórios, serão chamados de pais.

Exemplo: Vamos pegar o exemplo acima e listar quem são diretórios pais e filhos:

- Pai - O diretório kali é pai dos diretórios Documentos, Download, Músicas e filme; o diretório Documentos é pai de trabalho; o Download é pai de programas; o Músicas é pai de Rock.
- Filho - Documentos, Download, Músicas e filme são filhos de kali; trabalho é filho de Documentos; programas filho de Download; Rock filho de Músicas.

1.11 Comandos para diretórios

A seguir, na Tabela 1.1 temos os comandos para diretórios:

Tabela 1.1: Comandos Para Diretórios

Comandos	Descrições
ls	Esse comando é usado para mostrar tudo que existe dentro do diretório, desde arquivos ou outros diretórios
cd	cd (change directory) é usado para entrar ou sair de um diretório
pwd	Esse comando é usado para mostrar a localização atual do usuário
mkdir	Esse comando é usado para criar novos diretórios
rmdir	Esse comando é usado para remover um diretório

ls:

```
Sintaxe:
    ls caminho absoluto ou relativo -opção
```

Comandos + Opções	Descrições
ls sem opção	mostrará todos os arquivos e diretórios
ls -a	mostra todos os arquivos e diretórios e tudo que está oculto
ls -l	mostrará de uma maneira mais detalhada

cd:

```
Sintaxe:
    cd caminho
```

Comandos + Caminho	Descrição
cd /home/kali/Documentos	Mudando de diretório usando um caminho absoluto
cd Documentos	Mudando de diretório usando um caminho relativo
cd ..	Mudando do diretório filho para o diretório pai
cd	Mudando para o diretório inicial
cd /	Mudando para o diretório raiz
cd nome usuario	Mudando para o diretório inicial de outro usuário
cd 'nome com espaço'	Mudando para um diretório que tem espaço entre os nomes
cd DirPai/DirFilho/	Podemos sair de um diretório e entrar em um subdiretório

pwd:

```
Sintaxe:
    pwd
```

mkdir:

```
Sintaxe:
    mkdir <nome diretório>
    mkdir <nome diretório 1> <nome diretório 2> ...
```

Exemplos:

```
Comandos no Terminal:
    pwd
    ls
    mkdir diretorio1
    ls
    mkdir diretorio2 diretorio3

Saída:
    /home/kali
    Desktop   Documents   Downloads    ...
    Desktop   Documents   Downloads    ...  diretorio1
    Desktop   Documents   Downloads    ... diretorio1 diretorio2 diretorio3
```

Para criar uma diretório contendo espaços e/ou acentos gráficos, faz-se necessário utilizar o caractere imediatamente antes do caractere especial.

Exemplos:

```
Comandos no Terminal:
    mkdir diretorio\ 1
    mkdir seguran\ça\ da\ informa\ç\ão
```

Comandos - Opção	Descrição
mkdir -p	Cria um subdiretório ou um diretório filho dentro de outro diretório. Caso o diretório pai não exista, ele cria primeiro o diretório pai, logo após ele cria o subdiretório.
mkdir -v	Ele retorna uma mensagem logo após a criação de um diretório.
mkdir -m	Define privilégio de acesso.

rmdir:

```
Sintaxe:
    rmdir <nome do diretório>
```

O comando *rmdir* só excluirá o diretório ou subdiretório passado, caso esses estejam vazios. Para excluir tudo dentro do diretório, basta executar o seguinte comando:

```
Sintaxe:
    rm -rf <nome do diretório>
```

Porém tenha cuidado com esse comando, por que ele apagará tudo que houver dentro. Existem vários outros comandos que não foram incluídos aqui, mas esses já dão uma boa introdução sobre como trabalhar com diretórios e seus subdiretórios.

Comandos - Opção	Descrição
rmdir	Esse comando só excluirá o diretório ou subdiretório passado, caso esses estejam vazios.
rm -rf	Exclusão de tudo dentro do diretório.

1.12 Comandos para Arquivos

Tabela 1.2: Comandos Para Arquivos

Comando	Descrição
touch	Cria um arquivo vazio
rm	remove um arquivo
cp	copia um arquivo
mv	move um arquivo

Estes comandos servem também para diretórios.

touch:

```
Sintaxe:
    touch <nome do arquivo>
    touch <nome do arquivo 1><nome do arquivo 2> ...

No Terminal:
    ls
    touch teste.txt
    touch teste1 teste2 hello.py

Saída:
    Desktop   Documents   Downloads    ...
    Desktop   Documents   Downloads    ... teste.txt
    Desktop   Documents   Downloads    ... hello.py  teste1 teste2 teste.txt
```

Comandos - Opção	Descrição
touch	Cria um arquivo vazio.

rm:

```
Sintaxe:
    rm <nome do arquivo>

No Terminal:
    rm teste.txt
```

O arquivo "teste.txt" a ser excluído já deve ter sido criado previamente.

Comando + Opção	Descrição
rm + *extensão	excluirá todos os arquivos com a dada extensão.
rm -r ou -R	Com essa opção, serão excluídos diretórios e subdiretórios juntos.
rm -i ou rm -ri	A primeira opção perguntará se você deseja excluir o arquivo passado; a segunda opção perguntará se você deseja excluir o diretório passado.
rm -rf	Esse comando força a exclusão do arquivo.

cp:

```
Sintaxe:
    cp <caminho absoluto ou relativo> -opção
    cp <arquivo de origem> <arquivo de destino> -opção


No Terminal:
    rm teste.txt novo.txt
```

O arquivo "teste.txt" a ser copiado já deve ter sido criado previamente.

Comando + Opção	Descrição
cp -r	usado para copiar um diretório e todos os seus subdiretórios.
cp arq1 arq2 arq3	copiar mais de um arquivo ao mesmo tempo.
cp backup	Esse comando faz uma cópia do arquivo. Caso no local de destino exista algum arquivo com o mesmo nome, esse comando fará um backup do arquivo e criará outro arquivo com o mesmo nome.

mv:

```
Sintaxe:
    mv <arquivo de origem> <arquivo de destino> -opção

No Terminal:
    mv teste.txt novo.txt
```

O arquivo "teste.txt" a ser movido (recortado) já deve ter sido criado previamente.

Comando + Opção	Descrição
mv -i	pede permissão para sobrescrever um arquivo, caso exista um arquivo com o mesmo nome.
mv *	mover vários arquivos ao mesmo tempo para algum lugar específico.

1.13 Conteúdo de um arquivo

Agora, mostraremos o que existe dentro de um arquivo usando os seguintes comandos:

Tabela 1.3: Comandos Para Arquivos

Comandos	Descrição
head	mostrará as 10 primeiras linhas de um arquivo.
tail	mostrará as 10 últimas linhas de um arquivo.
more	Exibe o conteúdo de um arquivo.

head:

```
Sintaxe:
    head <nome do arquivo>  -opção

No Terminal:
    head teste.txt
```

O arquivo "teste.txt" já deve ter sido criado previamente.

tail:

```
Sintaxe:
    tail <nome do arquivo>  -opção

No Terminal:
    tail teste.txt
```

O arquivo "teste.txt" já deve ter sido criado previamente.

Tabela 1.4: Comando head

Comando + opção	Descrição
head -n	Por padrão, o head só imprime as dez primeiras linhas, mas você pode imprimir a quantidade que desejar, substituindo o "n" pela quantidade que você desejar.
head -c	Esse comando conta o número de bytes.

Tabela 1.5: Comando tail

Comando + opção	Descrição
tail -n	Por padrão, o tail só imprime as dez últimas linhas, mas você pode imprimir a quantidade que desejar, substituindo o "n" pela quantidade que você desejar.
tail -c	Esse comando conta o número de bytes.

more:

```
Sintaxe:
    more <nome do arquivo>  -opção

No Terminal:
    more teste.txt
```

O arquivo "teste.txt" já deve ter sido criado previamente. Para sair do aplicativo digite q (de *quit* em inglês que significa sair).

Tabela 1.6: Comando cat

Comando + opção	Descrição
more –num	mostra o número linhas do arquivo a cada enter.
more +num	mostra o arquivo a partir da linha determinada pelo *num*.

1.14 Atualização do Sistema Operacional

Uma dos grandes diferenciais do *Linux* diz respeito à maneira como são instalados os programas. As distribuições *Linux* usam os chamados gerenciadores de pacotes, que são como uma *"Google Play"* do seu Android, na instalação dos programas. A maneira para achar a loja de aplicativos varia entre sistemas *Linux*. Mas a presente Seção serve como base para qualquer distribuição.

A depender do Sistema operacional *Linux* empregado, será preciso utilizar um APT-GET. Geralmente, sistemas que foram baseados na Distribuição *Debian*, tal qual o *Kali Linux*, utilizam o APT-GET. Por outro lado, as distribuições baseadas na RED HAT ENTERPRISE utilizam o YUM. Mas como saber se sua distribuição utiliza um ou outro? Bom, deve-se ler a documentação (quando existente) da sua distribuição.

O comando *apt-get update* permite acesso às versões mais recentes dos pacotes e garante que o *Kali Linux* esteja ciente das atualizações disponíveis. O comando *apt-get update* não instala nenhuma

atualização em si. Ele apenas sincroniza as informações de pacotes disponíveis nos repositórios.

Por outro lado, o comando *apt-get upgrade* (atualização de pacotes) instala as atualizações disponíveis para os pacotes instalados, garantindo que você tenha as versões mais recentes e corrigindo eventuais vulnerabilidades ou bugs presentes nas versões anteriores.

Em síntese, "update"sincroniza as informações de pacotes nos repositórios, enquanto "upgrade"instala as atualizações disponíveis para os pacotes instalados em seu sistema. É comum executar primeiro o "update"para obter as informações mais recentes sobre os pacotes e, em seguida, executar o "upgrade"para instalar as atualizações.

Siga as instruções:

1. No terminal, ingresse no sistema com administrador, Por padrão, senha **kali**.:

```
sudo su
```

2. No terminal, use o *apt-get update*:

```
apt-get update
```

3. Ocasionalmente, o comando *apt-get upgrade* pode ser empregado caso o *apt-get update* não tenha sido capaz de solucionar a instalação de um pacote. Cabe atentar que o *apt-get upgrade* custa bem mais tempo em comparação ao *apt-get update*:

```
apt-get upgrade
```

1.15 Introdução ao Vim

Vim, abreviação de "Vi Improved", é um editor de texto altamente configurável e poderoso amplamente utilizado em ambientes Unix e Linux. Ele foi projetado para edição de texto eficiente e é conhecido por sua interface modal única, que permite aos usuários alternar entre diferentes modos para navegação, edição e outras tarefas.

história começa com a criação do Vi (Visual Editor) por Bill Joy em 1976. O Vi foi originalmente desenvolvido para o sistema operacional Unix na Universidade da Califórnia, Berkeley. O Vi tornou-se um editor de texto padrão em sistemas Unix e foi projetado para funcionar de maneira eficiente com as interfaces orientadas por teclado da época.

Bram Moolenaar, um programador de computador holandês, criou o Vim (Vi Improved) em 1991 como uma versão estendida e aprimorada do Vi. O Vim foi desenvolvido para abordar algumas limitações do Vi e fornecer recursos e melhorias adicionais. O objetivo de Bram era tornar o Vim mais poderoso, eficiente e personalizável.

O Vim tem uma construção que se assemelha com uma linguagem, com verbos e substantivos que permitem ao usuário a editar o texto mais rapidamente com comandos rápidos e eficientes. Essa é a grande vantagem do Vim em relação a outros editores de texto e talvez o motivo de ser amplamente usado em terminais.

1.16 Começando com o Vim

A primeira vez usando Vim pode ser intimidadora, pois ele não é o tipo de editor de texto ortodoxo na qual a interface é autoexplicativa. Tenha em mente que o Vim tem uma curva de aprendizagem íngreme e é isso que o torna o editor de texto preferido de muitos usuários. A sua experiência e familiaridade dirá respeito à sua rapidez e liberdade para editar texto.

Para instalar o Vim no Kali Linux são necessários apenas dois comandos:

1. **sudo apt update**

2. **sudo apt install vim**

O sistema solicitará sua senha de usuário para confirmar a instalação. Digite sua senha e pressione Enter. Quando solicitado, pressione **Y** e depois Enter para confirmar que deseja continuar com a instalação. Para saber se a instalação foi concluida com sucesso, o comando **vim –version** pode ser utilizado.

Para começar a editar as primeiras linhas de texto digite **vim** no terminal e o editor será aberto. Inicialmente, alguns comandos são interessantes conhecer:

1. Modo de Inserção:
   - Pressione **i** para entrar no modo de inserção, onde você pode começar a digitar e inserir texto no seu arquivo.

2. Modo Normal:
   - Pressione **Esc** para sair do modo de inserção e entrar no modo normal, onde você pode navegar e executar vários comandos.

3. Salvar e Sair:
   - **:q** - Sair do Vim.
   - **:w** - Salvar o arquivo atual.
   - **:wq** - Salvar e sair (combina os dois comandos).

4. Sair sem Salvar:
   - **:q!** - Sair do Vim sem salvar as alterações.

5. Navegação:
   - **h** - Mover o cursor para a esquerda.
   - **j** - Mover o cursor para baixo.
   - **k** - Mover o cursor para cima.
   - **l** - Mover o cursor para a direita.

1.17 Sintaxe da linguagem Vim

Vim é um editor de texto e também uma linguagem. Outros editores possuem ferramentas bem definidas para mudar o texto, entretanto o Vim possui uma maneira de combinar comandos e formar um comando único para uma determinada tarefa que se aproxima de uma linguagem real e consegue ser bastante útil. Comecemos com a introdução aos verbos do Vim:

- **d** - de *delete*, para deletar
- **c** - de *change*, para mudar
- **v** - de *visually select*, para selecionar
- **y** - de *yank*, para copiar
- **>** - para indentar

Apenas os verbos não são o suficientes para criar um comando. Assim como uma linguagem real, faz-se necessário o uso de outras estruturas sintáticas, neste caso os substantivos, para criar uma sentença.

- **w** - de *word*, isto é, palavra

Com a junção de **d** e **w** criamos o nosso primeiro comando *delete word* que irá deletar a porção de texto que está a frente do cursor. Caso queira deletar o que está atrás do cursos, o comando é **db** de *delete back*.

Outro uso interessante do substantivo *word* é navegar através do texto, isto é, se **w** for pressionado repetidas vezes, o cursor irá pular para a próxima palavra. Outro maneira de navegar o texto é utilizando **e** para pular parágrafos. Agora como exercício tente fazer uma combinação de tal forma que delete um parágrafo inteiro.

Eventualmente o uso de substantivos apenas não são suficientes para criar um atalho responsivo para uma ação desejada, por conta disso o Vim possui substantivos como objetos de texto:

- **iw** - de *inner word*, palavra contida em algo
- **it** - de *inner tag*, conteúdo dentro de um *tag* de HTML
- **ip** - de *inner paragraph*, conteúdo dentro de um parágrafo
- **i"** - de *inner quotes*, conteúdo dentro de aspas
- **as** - de *a sentence*, para selecionar um sentença

1.18 Combinação de comandos

A combinação de comandos no Vim é uma técnica poderosa que permite realizar tarefas complexas de forma eficiente. Isso envolve a execução de uma sequência de comandos Vim de forma consecutiva para atingir um objetivo específico. Essa combinação é dada pela junção dos verbos e substantivos vistos acima.

1. Deletar uma palavra inteira:
   - Posicione o cursor em qualquer lugar dentro da palavra.
   - Pressione **diw** ou **dw**.

2. Deletar conteúdo dentro de uma tag HTML:
   - Posicione o cursor dentro da tag.
   - Pressione **dit**.

3. Deletar conteúdo dentro de aspas:
   - Posicione o cursor dentro das aspas.
   - Pressione **di"**.

4. Deletar uma frase inteira:
   - Posicione o cursor em qualquer lugar dentro da frase.
   - Pressione **das** ou **dass**.

5. Mudar uma palavra inteira:
   - Posicione o cursor em qualquer lugar dentro da palavra.
   - Pressione **ciw** ou **cw**.

6. Mudar conteúdo dentro de uma tag HTML:
   - Posicione o cursor dentro da tag.
   - Pressione **cit**.

7. Mudar conteúdo dentro de aspas:
   - Posicione o cursor dentro das aspas.
   - Pressione **ci"**.

8. Mudar uma frase inteira:
   - Posicione o cursor em qualquer lugar dentro da frase.
   - Pressione **cas** ou **cass**.

9. Indentar um parágrafo:
   - Posicione o cursor em qualquer lugar dentro do parágrafo.
   - Pressione **>ip** ou **>ap**.

10. Selecionar um parágrafo visualmente:
    - Posicione o cursor em qualquer lugar dentro do parágrafo.
    - Pressione **vip** para selecionar visualmente o parágrafo.

1.19 Permissão de arquivos

Em alguns projetos nasce a necessidade de proteger os arquivos de alterações indevidas e apenas permitir a leitura por parte dos usuários, já o comando **chmod** permite o administrador a fazer essas alterações.

1.20 Discussão

O objetivo deste Capítulo foi aprofundar um pouco sobre o uso do terminal, já que iremos invocar comandos para testes de sistemas, e a grande dos softwares não usam de uma tela gráfica em âmbitos de Segurança da Informação. Então o uso da interface de linha de comando é um pré-requisito para o uso do *Kali Linux*. Para isso foi explicado o que é o terminal, vimos também o que é o interpretador de comandos(*shell*) e sua diferença com o terminal.

A seguir vimos como:

- sair de um diretório para outro:
- usando caminhos relativos ou absolutos.
- Logo após, começamos a trabalhar com comandos: criando e excluindo diretórios; criando, excluindo, copiando e movendo arquivos;
- por último, trabalhamos com comandos para interagir com o conteúdo de um arquivo.

2. *Python*: Parte 1

Gabriela Leite Pereira
Currículo *Lattes*: http://lattes.cnpq.br/6506293109534236
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. gabriela.lpereira@ufpe.br

Lucas de Souza Santos
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Engenharia Elétrica - Universidade Federal de Pernambuco, Recife, Pernambuco. lucas.ssantos2@ufpe.br

Dr. Sidney Marlon Lopes de Lima
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. sidney.lima@ufpe.br

Observação:

Todos os códigos usados no capítulo estão no *link* : https://colab.research.google.com/drive/12cG6JMtsT5ASDPQpcDbtHmnDFwj3Hsy_?usp=sharing.

2.1 Introdução

Segundo pesquisa da Stack Overflow, conforme a Fig. 2.1, feita com mais de 80 mil desenvolvedores. Em uma avaliação híbrida entre popularidade pelos desenvolvedores e tecnologia mais usada, *python* é a terceira linguagem mais importante mundialmente.

Mas porque *python* tem tanto destaque entres os desenvolvedores? Talvez uma boa resposta para isso seja, que em *python*, pode-se desenvolver muitas aplicações sem ter que escrever muitas linhas de código e também pela facilidade em aprender. *Python* é uma linguagem de alto nível, interpretada, orientada a objetos, dinâmica e de propósito geral. Com o *python* é possível trabalhar

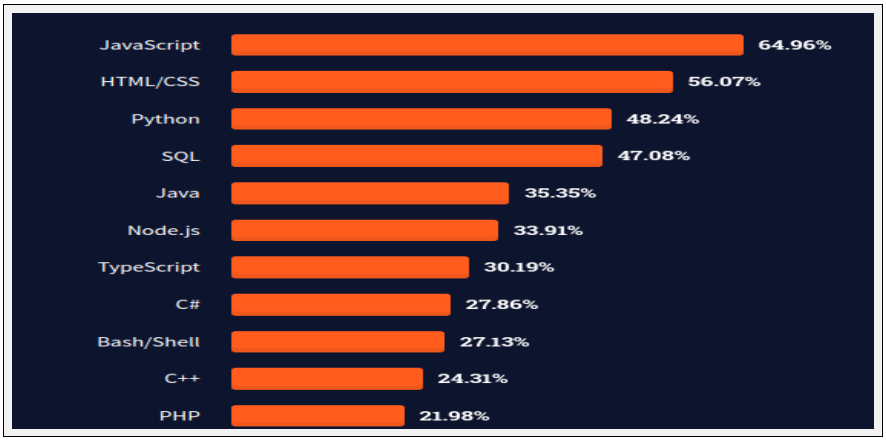


Figura 2.1: *Ranking* retirado do site do *stack overflow* sobre a preferência de desenvolvedores sobre linguagens de programação.

com ciência de dados, inteligência artificial, aprendizado de máquina e criar páginas web. Abaixo compararemos como é imprimir um simples texto com *python* e com Java:

```
#código para imprimir "Hello World" em python:
print("Hello World!")

#código  para imprimir "Hello World" em Java:
helloworld{
 public static void main(String[] args)
 {
      System.out.println("Hello World");
 }
}
```

Veja que tanto em *python* como em Java a saída dos dois códigos acima é: "Hello World", mas no *python* só precisamos de uma linha para poder imprimir a mesma mensagem do Java.

2.2 Instalação

A instalação do *python* será feita em apenas dois sistemas operacionais: windows e linux. Então, os passos serão descritos logo abaixo:

2.3 Windows

A primeira coisa é acessar o site para baixar a versão mais atual do *python*, ou a versão que você desejar, para isso digite, como em Fig.2.2: https://www.*python*.org/downloads/.

Clique na palavra windows, que é mostrado na figura acima, destacado pela caixa em vermelho. Agora você será direcionado para poder baixar a versão de 64 bits ou de 32 bits do seu sistema operacional. Como é visto nas figuras 2.3 e 2.4 abaixo:

A versão mais estável atualmente é a versão 3.10.6, como é visto nas imagens anteriores, mas você poderá baixar a versão que melhor lhe atender. Como o *python* é atualizado sempre, é possível que você esteja em versões mais superiores a essa. No entanto, o procedimento é o mesmo.

Uma vez baixado o instalador é só clicar duas vezes e será aberta uma caixa de diálogo, que mostrará duas opções, mas antes de escolher qualquer uma delas, marque a caixinha logo abaixo

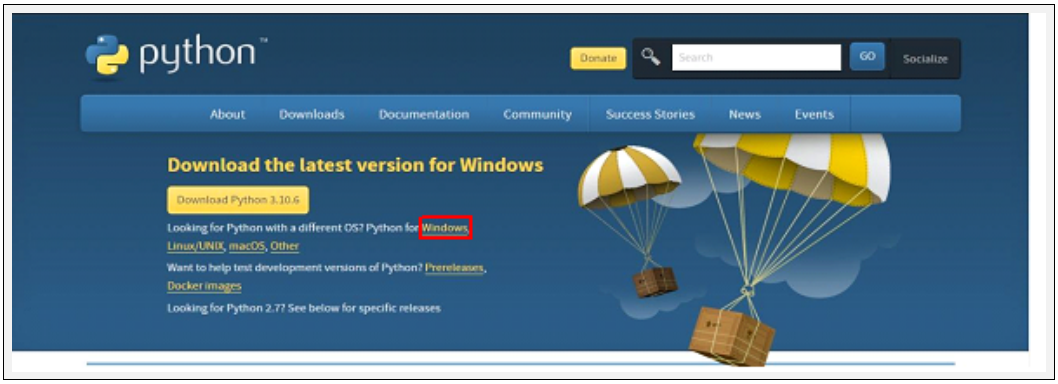


Figura 2.2: Página de download para o *python*

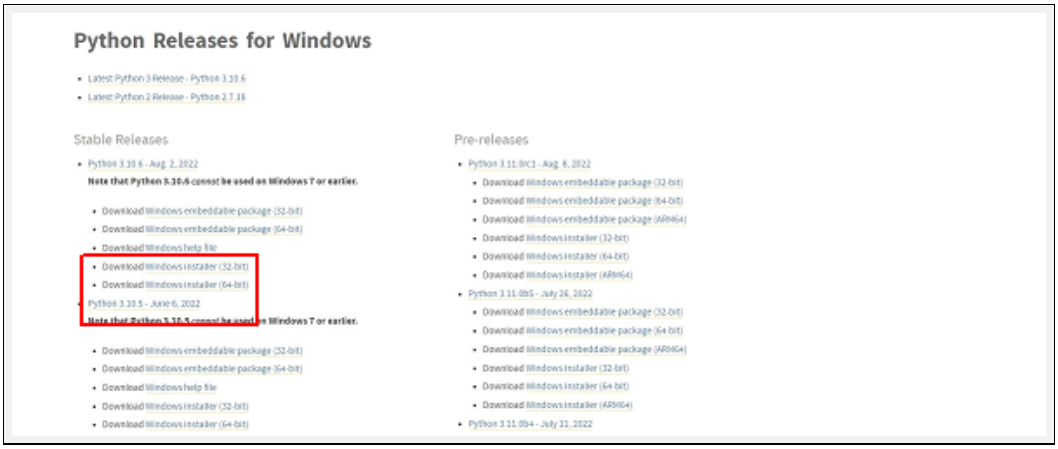


Figura 2.3: Página contendo as versões do *Python*

para adicionar o *python* as variáveis de ambientes e depois escolha a instalar agora(install Now), como é visto nas imagens 2.5, 2.6 e 2.7 abaixo:

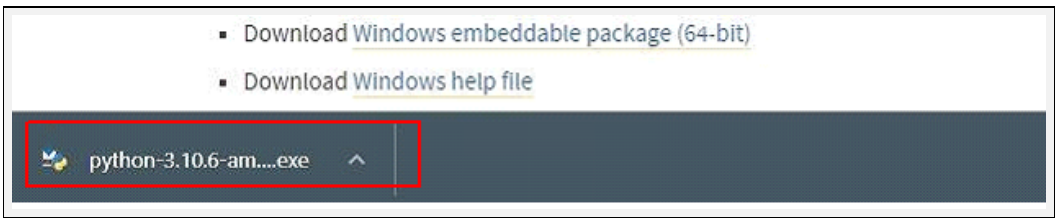


Figura 2.4: Instalador do *Python* baixado para o *Windows*

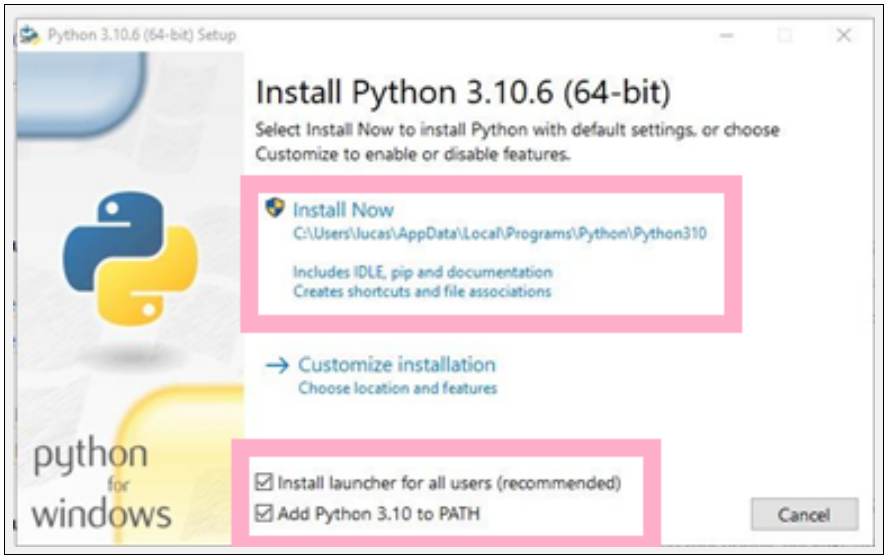


Figura 2.5: Tela inicial na instalação do *Python* no *Windows*.

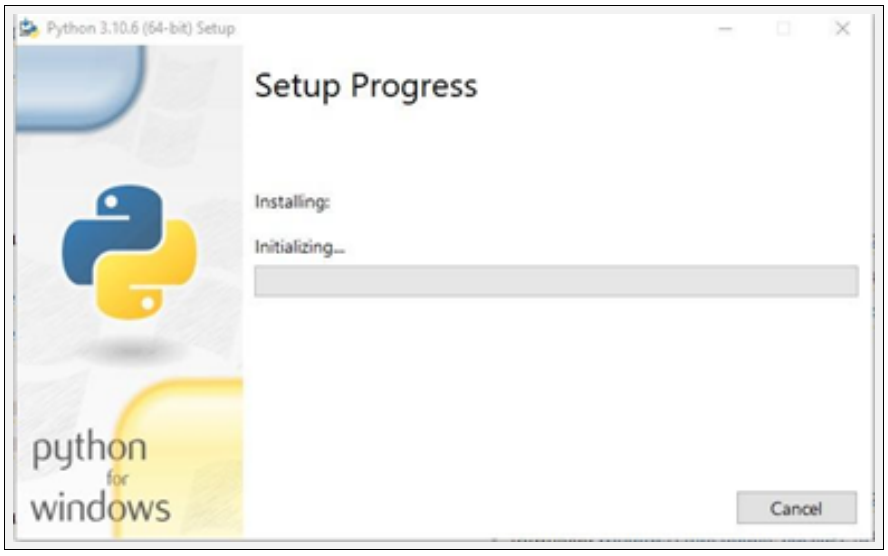


Figura 2.6: Tela mostrando o processo de instalação.

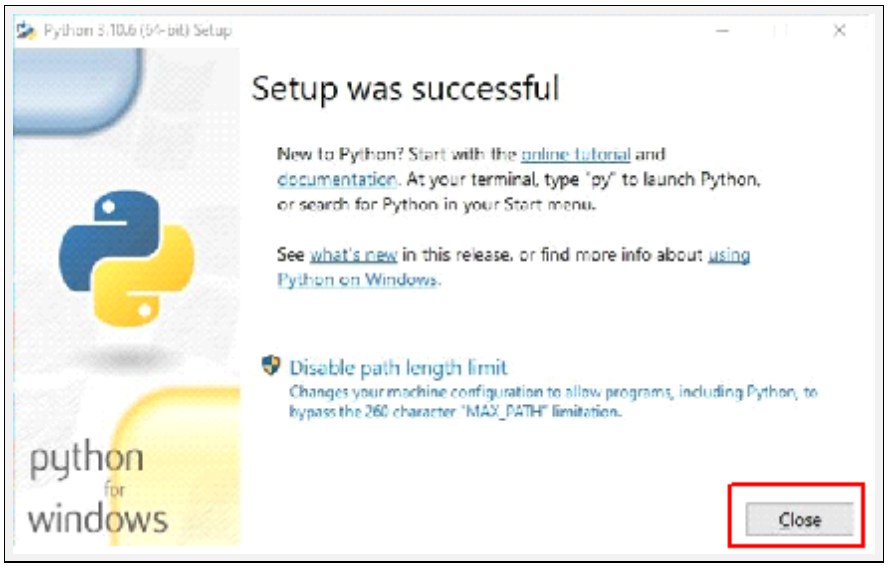


Figura 2.7: Tela mostrando que a instalação do *Python* foi bem sucedida.

2.4 Linux

Para instalar o *python* no linux o processo é bem mais simples. Primeiro abra o terminal(ctrl + alt + t) como na Fig.2.8:

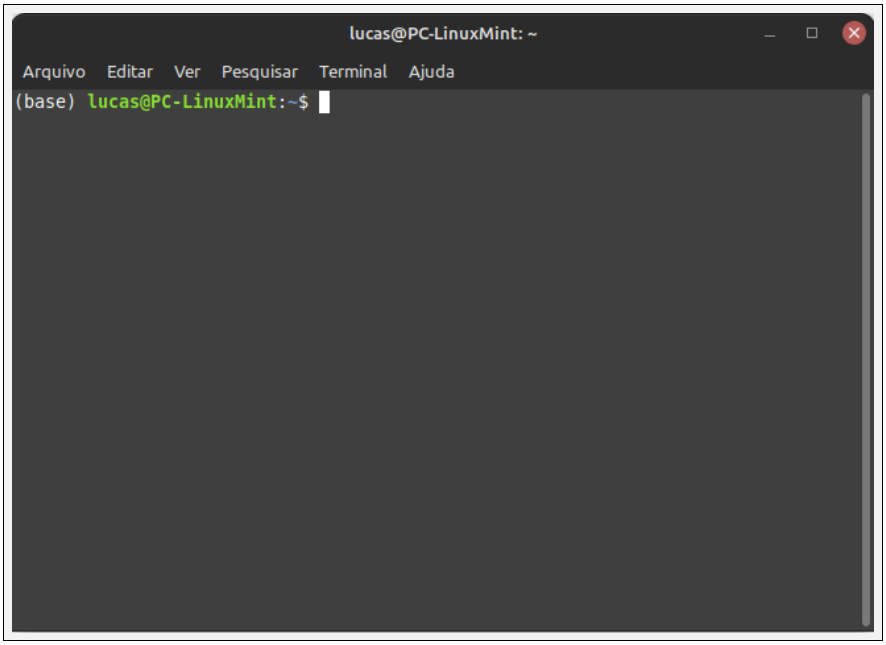


Figura 2.8: Visão do terminal do Linux *Mint*

Agora vamos atualizar o repositório da versão rodando o comando abaixo:

```
sudo apt-get update
```

Logo após digitar a senha de superusuário, caso tenha cadastrado alguma, fique atento que será preciso digitar "y" para confirmar as alterações. Depois que o sistema estiver atualizado, agora escreva no terminal o comando abaixo para instalar o *python*.

```
apt-get install python
```

Caso você tenha várias versões do *python* instalado no seu computador, é normal que algumas versões de algumas distribuições do linux vim instalado o *python*2.7, ou até com uma versão anterior a mais atual, como padrão, e para mudar é preciso rodar esse comando no terminal:

```
ln -sf /usr/bin/python3.10.6 /usr/bin/python
```

O comando acima modifica o link simbólico, que estava apontando para a versão do *python* não desejada, para a versão que você deseja rodar como padrão no seu computador. Com essa modificação você pode rodar qualquer aplicativo *python*, só escrevendo no seu terminal e dentro da pasta que está localizado o aplicativo:

```
python nome_aplicativo.py
```

Uma vez instalado o *python* em seu computador, agora podemos escrever nossos códigos: para isso eu também recomendo a instalação de um editor de código, tipo o visual studio code, aton ou o até mesmo o sublime; com eles, a escrita dos seus códigos, ficam mais rápido e com as extensões você pode receber feedback sobre erro de sintaxes. Existem também as IDES, que são ambientes de desenvolvimento mais completo, mas que são bem mais pesados e reduzem o desempenho do seu computador bastante; para máquinas com baixo recurso recomendo os editores de códigos.

2.5 Comentários

Para escrever um comentário no *python* é bem simples.

```
#Exemplo 1: Escrevendo em apenas uma linha
'Escrevendo comentário em uma linha'
"Escrevendo comentário em uma linha"
```

```
#Exemplo 2: Escrevendo em várias linhas
'''
Esse é outro exemplo
mostrando como funciona os comentários
no python.
'''
"""
Esse é outro exemplo
mostrando como funciona os comentários
no python.
"""
```

2.6 Palavras reservadas

Palavras reservadas ou palavras chaves são palavras especiais para o bom funcionamento do *python*, elas tem um total de 35 palavras, não podendo usá-la para nome de variáveis, são elas, de acordo com a Tabela 2.1 :

Tabela 2.1: Tabela das palavras reservadas no *python*.

as	and	assert	async	True	None	False
await	break	class	continue	def	del	elif
else	except	finally	for	from	global	if
import	in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with	yield

2.7 Funções internas

As funções internas do *python*, são funções já pré-definidas, que estão sempre disponíveis e não precisam ser importadas. Veremos algumas dessas funções:

1 **print()**
A função print() imprime qualquer valor passado dentro dos parênteses.

```
#Exemplo 3
# imprimindo um texto
print('Imprimindo texto')
# imprimindo um número inteiro
print('Imprimindo número inteiro: ', 4)
# imprimindo um número real
print('Imprimindo número real: ', 4.5)
```

2 **sum()**
A função sum() soma todos os números passando como argumentos e é usado dentro da função print() para mostrar o resultado na tela:

```
#Exemplo 4
# usando a função soma para somar 1 + 2 e armazenar na variável soma1
soma1 = sum((1, 2))
# imprimindo a variável soma1
print(soma1)
# usando a função print() e sum() para imprimir um texto e a soma de 4 + 5.
print('Soma de 4 + 5 usando a função sum: ',  sum((1, 2)))
# agora usando a função sum() para somar 4 + 5 + 6.
print('Soma de 4 + 5 + 6 é igual a: ', sum((4, 5, 6)))
```

3 **type()**
A função type() informa a classe a que a variável pertence, uma string pertence a uma classe do tipo string, um número inteiro pertence a uma classe do tipo int e assim por diante. No *python* todos os valores pertencem a uma classe específica.

4 **input()**
A função input() pega os valores que são digitado no terminal e é armazenado em uma variável e o tipo dessa variável é uma string, vejamos um:

```
#Exemplo 5
# usando a função input() para pegar um valor e atribuir
# na variável "nome"
nome = input('Informe o seu nome: ')
# imprimindo o tipo da variável "nome"
# imprimindo o valor da variável "nome"
print(type(nome), nome)
```

5 **int()**
Como toda entrada passada pela função input() é uma string, como podemos usar números inteiros ? A resposta para isso é usamos a função int(). Essa função pega um número, que no primeiro momento é uma string, e transforma seu valor para um tipo inteiro. Vejamos um exemplo:

```
48 #Exemplo 6
49 # a variável "num" receberá uma entrada vinda do terminal
50 num  =  input('Digite um número: ')
51 # Iremos imprimir o tipo da variável número
52 print(type(num))
53 # apesar de ter digitado um número, o \textit{python} eternamente transforma a
       variável em
54 #  string
55 # Imprimindo a variável
56 print(num)
57 # Aqui iremos converter a variável "num" em um tipo numérico e depois
      reatribuir
58 # nela mesmo. Logo após, imprimir o tipo e depois imprimir o valor.
59 num = int(num)
60 print(type(num), num)
```

O programa acima também pode ser escrito da seguinte maneira:

```
61 #Exemplo 7
62 # O número agora será convertido antes de ser atribuído à
63 # variável.
64 num  =  int(input('Digite um número: '))
65 print(type(num), num)
```

6 **float()**
Essa função também converte o valor de um certo tipo para um float.

```
66 #Exemplo 8
67 # O número agora será convertido antes mesmo de ser atribuído à variável.
68 num = float(input('Digite um número: '))
69 print(type(num), num)
```

Tipos de dados diferentes podem gerar um erro, então cuidado ao digitar.

```
70 #Exemplo 9
71 num = float(input('Digite um número: '))
72 print(type(num), num)
```

2.8 Indentação no *python*

A indentação no *python* é bem simples de entender, mas muito importante para o funcionamento dos programas, já que ela é usada para separar blocos de códigos. Vejamos um exemplo abaixo:

Siga a estrutura básica:

```
While Condição:
    bloco de código
```

```
#Exemplo 10
# vai ser criado uma variável chamada cont, para ser
# nosso contador, e vai ser iniciada ela com o valor 1.
# Essa variável será acrescentada de 1 em 1 até o valor
# 10.
cont = 1
# while é um loop de repetição e obedece a condição
# cont <= 10
# quando o valor cont for maior que 10, o loop deixa de
# acontecer.
while cont <= 10:
# Aqui começa o bloco de código com um 4 espaços
# separados da margem esquerda. Poderemos ter diversas
# while aqui dentro, mas todos precisam obedecer suas
# respectivas indentação.
    print(cont, end=' ')
# A função print vai imprimir o valor de cont e o end=' '
# vai impedir que pule linha.
# E o cont += 1 vai acrescentar 1 em cada final de laço,
# até o valor de cont ser maior que 10.
    cont += 1
```

2.9 Variáveis no *python*

Variáveis são espaços na memória que podem armazenar informações. Essa informação pode ter tipos diferente: você pode querer guardar a informação da idade de um adolescente, para determinar se ele pode ou não dirigir; ou, você pode querer guardar o nome desse mesmo jovem; talvez seja necessário dizer se o aluno estar ou não estar habilitado. Quais tipos eu devo como para cada uma dessas variáveis ? Bom, no *python* não é preciso escrever de forma explícita o tipo da variável, basta apenas atribuir o valor que o próprio *python* entenderá qual é o tipo daquela variável.

2.10 Declaração de variável e atribuição de valor

Para declarar e atribuir um nome e um valor a uma variável, basta simplesmente informar o nome da variável, colocar o sinal de igual e informar o valor. Veja o exemplo a seguir:

Declarando e atribuindo valor a uma variável:

"nome variável'' + "Sinal de igual'' + "valor''

```
94 #Exemplo 11
95 nome = 'Maria'
96 idade = 19
97 peso = 50.5
98 altura  = 1.55
99 E_mulher = True
```

Existem algumas regras ao criar um nome para uma variável, são elas:

- não usar número no primeiro dígito;
- Não conter nenhum espaço e nenhum caracteres especial do tipo ($,%, #, & ...);
- Pode ser usado uma letra do alfabeto ou um sublinhado(_) no primeiro dígito;
- Não pode ser igual a uma palavra reservada;
- Letras maiúsculas e minúsculas são critério para diferenciar nome de variável.
- Exemplo de nome de variável válido: _nome, num1, Nome, nome, num_1, ...
- Exemplo de nome inválidos: 4nome, 4_nome, #nome, $nome, nome pessoa, ...

Caso alguma dessas regras não seja obedecida, irá gerar um erro de sintaxe.

2.11 Variável numérica

Variáveis do tipo numérico podem armazenar valores inteiros, flutuantes ou complexos.

```
100 #Exemplo 12
101 num1 = 5
102 num2 = 3.4
103 num3 = 4 + 3j
104
105 print(type(num1))
106 print(type(num2))
107 print(type(num3))
```

2.12 Variável String

Uma string é uma cadeia de caracteres dentro de aspas simples, dupla ou até mesmo tripla. É normal usar a operação de concatenação, representada pelo operador "+", para unir duas strings.

```
108 #Exemplo 13
109 string1 = 'Lucas'
110 string2 = 'Lima'
111 string3 = 'Luiz'
112 string_maior = 'Os alunos ' + string1 +  ', ' + string2 + ', e ' + string3 + ' tiraram as
        melhores notas da prova de português.'
113 print(string_maior)
```

Algumas observações na concatenação das três strings, formando a "string maior". Primeiro, a palavra string é uma palavra reservada, mas por causa dos números no final não irá gerar um erro. Segundo, os espaços colocados no fim de alguns textos é proposital para os textos não saírem colocados.

```
114 #Exemplo 14
115 nome1 = 'Lucas'
116 nome2 = 'Santos'
117 print(nome1 + nome2)
118 print(nome2 + nome1)
```

Perceba que mesmo colocando espaço dentro da função print(), entre os nomes das variáveis e o operador de "+", os valores saem juntos. A ordem que os nomes são colocados influência na saída, como é visto acima.

2.13 Variável Booleana

Variáveis booleanas é o tipo que recebe dois valores: True ou False. É preciso atentar que os valores True e False começam com letra maiúscula, diferente de outras linguagem.

```
119 #Exemplo 15
120 verdadeiro = True
121 falso = False
122 print(verdadeiro)
123 print(falso)
```

2.14 Operadores

Na matemática quando você quer acrescentar um certo número a outro é comum usarmos um símbolo para representar esse acréscimo. No nosso exemplo, estamos falando do símbolo de somar que é representado por uma cruzinha como essa "+". E existem diversas operações e diversos símbolos representando cada operação. Na programação é usado também símbolos para representar as mesmas operações matemáticas e até outros tipos de operações; no entanto, é possível usar o mesmo símbolo para representar operações com tipos de dados diferentes. Quando usamos o símbolo "+"com dois números, essa operação é a tradicional soma, que vemos na matemática; mas, quando esse símbolo é usado em duas strings, a operação que é efetuada é a de concatenação. Então, é por isso que devemos saber o tipo de dados que estamos trabalhando, para fazer corretamente as operações.

Na programação existem vários tipos de operações: as mais famosas são as operações matemáticas; mas existem as operações lógicas, de comparação, atribuição, operação bit a bit, de associação e de identidade. Na presente seção, será construído algumas tabelas mostrando os nomes das operações, símbolos utilizados, a operação usando um exemplo simples e o resultado de cada operação.

2.15 Operadores Matemáticos

A seguir, tem-se a Tabela 2.2 sobre operadores matemáticos, símbolos, exemplo de operação e seus resultados em Python.

Tabela 2.2: operadores matemáticos, símbolos, exemplo de operação e seus resultados em Python.

Operador	Símbolo	operação	resultado
soma	+	1+3	4
subtração	-	2-1	1
multiplicação	*	4*4	16
divisão	/	5/2	2.5
divisão inteira	//	5/2	2
potência	**	3**2	9
resto	%	5%2	1

2.16 Ordem de precedência

Não é objetivo explicar como funciona, a fundo, as ordem de precedência no Python; contudo, é preciso informar que elas existem. É preciso ter cuidado quando for fazer alguma operação, é preciso verificar a ordem de precedência de cada operador para que o resultado não seja diferente do esperado. Então, recomendo verificar na documentação como funciona as operações e suas ordens de precedência. Veremos abaixo as ordens de precedência dos operadores matemáticos na Tabela 2.3.

Tabela 2.3: Tabela com as ordens de precedência dos operadores matemáticos em python

Operações	Exemplo	Resultado	Detalhes
()	(4 + 5) ** 2 * 2 + 12	174	o parêntese tem a maior ordem de precedência
**	3 ** 2 / 3	3	A potência é a segunda maior em ordem de precedência
* e /	3 * 4 - 2 4/2 + 10	10 12	A multiplicação e a divisão são a terceira em ordem de precedência
+ e -	4+1 5-1	5 4	Restando a soma e a subtração por último

2.17 Operadores de Comparação

Esses tipos de operadores são utilizados em estruturas condicionais e em laços de repetição, que veremos mais adiante. Podemos comparar os números, podemos ver se strings são iguais. As comparações retornam verdade(True) ou falso(False). Vejamos abaixo, na Tabela 2.4 as relações que podemos fazer, usando exemplos numéricos.

Tabela 2.4: Operadores de comparação, símbolos, exemplos de operação e resultados em Python.

Operadores	Símbolo	operação	resultado
igual	==	4==4	True
diferente	!=	4!=4	False
maior que	>	5>4	True
maior ou igual à	>=	5>=5	True
menor	<	3<2	False
menor ou igual à	<=	3<=5	False

2.18 Operadores Lógicos

Geralmente, quando temos mais de duas expressões, ou expressões mais complexas, utilizamos operadores lógicos para operar sobre essas estruturas e retorna um valor verdadeiro ou falso. Na Tabela 2.5 abaixo, na coluna operação, cada valor True e False, podem representar uma expressão lógica.

Tabela 2.5: Operadores lógicos, símbolos, tabela verdade nas operações e resultado. em Python.

Operador	Símbolo	operação	resultado
And	and	True and True	True
		True and False	False
		False and True	False
		False and False	False
Or	or	True or True	True
		True or False	True
		False or True	True
		False or False	False
negação	not	!True	False
		!False	True

2.19 Operadores identidade

A seguir, tem-se a Tabela 2.6 sobre operadores de identidade.

Tabela 2.6: Operadores identidade

Operador	Símbolo	operação	resultado
is	nome="Santos"	nome is "lucas"	False
		nome is "Santos"	True
not is	nome="Marta"	nome not is "Lucas"	True
		nome not is "Marta"	False

2.20 Operadores associação

A seguir, tem-se a Tabela 2.7 sobre operadores de associação.

Tabela 2.7: Operadores de associação

Operador	Símbolo	operação	resultado
in	nome="Santos"	"lu"in nome	False
		"lu"in "Santos"	False
not in	nome="Marta"	nome not in "Lucas"	True
		nome not in "Marta"	False

2.21 Exercícios

1. Faça um programa que peça uma nota, entre zero e dez. Mostre uma mensagem caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido.

2. Supondo que a população de um país A seja da ordem de 80000 habitantes com uma taxa anual de crescimento de 3% e que a população de B seja 200000 habitantes com uma taxa de crescimento de 1.5%. Faça um programa que calcule e escreva o número de anos necessários para que a população do país A ultrapasse ou iguale a população do país B, mantidas as taxas de crescimento.

3. Altere o programa anterior permitindo ao usuário informar as populações e as taxas de crescimento iniciais. Valide a entrada e permita repetir a operação.

4. Faça um programa que leia 5 números e informe o maior número.

5. Faça um programa que leia 5 números e informe a soma e a média dos números.

6. Faça um programa que receba dois números inteiros e gere os números inteiros que estão no intervalo compreendido por eles.

7. O Sr. Manoel Joaquim expandiu seus negócios para além dos negócios de R$1,99 e agora possui uma loja de conveniências. Faça um programa que implemente uma caixa registradora rudimentar. O programa deverá receber um número desconhecido de valores referentes aos preços das mercadorias. Um valor zero deve ser informado pelo operador para indicar o final da compra. O programa deve então mostrar o total da compra e perguntar o valor em dinheiro que o cliente forneceu, para então calcular e mostrar o valor do troco. Após esta operação, o programa deverá voltar ao ponto inicial, para registrar a próxima compra. A saída deve ser conforme o exemplo abaixo:

```
Lojas Tabajara
Produto 1: R$ 2.20
Produto 2: R$ 5.80
Produto 3: R$ 0
Total: R$ 9.00
Dinheiro: R$ 20.00
Troco: R$ 11.00
...
```

3. *Python*: Parte 2

Gabriela Leite Pereira
Currículo *Lattes*: http://lattes.cnpq.br/6506293109534236
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. gabriela.lpereira@ufpe.br

Lucas de Souza Santos
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Engenharia Elétrica - Universidade Federal de Pernambuco, Recife, Pernambuco. lucas.ssantos2@ufpe.br

Dr. Sidney Marlon Lopes de Lima
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. sidney.lima@ufpe.br

Observação:

Todos os códigos usados nos capítulos estão no *link* : https://colab.research.google.com/drive/12cG6JMtsT5ASDPQpcDbtHmnDFwj3Hsy_?usp=sharing.

3.1 Sequências

- Strings

Como já falamos anteriormente: strings são sequências de caracteres. Mas agora iremos focar em seus métodos, que esses, podemos definir como funções dentro do paradigma de orientação a objeto como podemos observar na Tabela 3.1.

Tabela 3.1: Operadores para trabalhar com *strings* em *Python*.

Operadores	Descrição
+	Operador de concatenação
*	Operador de repetição
[]	Operador Slice
[:]	Operador Slice com intervalo
in	operador de associação, verifica se uma substring está dentro da string
not in	operador de associação, verifica se uma substring não está dentro de uma string

```
#Exemplo 1
# criando uma string com nome string1
string1 = 'hello'
# tem um espaço no início dessa string na string2
string2 = 'python'
# string3 e a concatenação da string1 e string2
string3 = string1 + string2
print(string3)
print(string1 * 2)
print(string2 * 3)
```

- Indexação de Strings

No python a indexação começa no zero e vai até o comprimento da string menos um. Mas, o python também aceita índices negativos, começando sempre do menor para o maior. No caso dos índices negativos, depende sempre do tamanho da string; por que o primeiro índice é sempre o menos o tamanho da string, veja o exemplo abaixo e a Tabela 3.2:

Tabela 3.2: string Python organizado na memória do computador e seus respectivos índices.

P	Y	T	H	O	N
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

Podemos imprimir cada caracter pelo índice de cada um deles, passado dentro dos colchetes, veja abaixo:

```
#Exemplo 2
string = 'PYTHON'
print(string[0], string[1], string[2], string[3], string[4], string[5])
#Aqui vai imprimir uma sequência de "="
print('=' * 12)
print(string[-6], string[-5], string[-4], string[-3], string[-2], string[-1])
```

Caso você pedisse para imprimir o valor "string[6]", o python retornaria um erro, por que não existe nenhum valor em "string[6]".

```
#Exemplo 3
print(string[6])
```

- Strings são imutáveis

Caso você tente mudar um carácter de uma strings, um erro será lançado; porque strings são IMUTÁVEIS.

```
#Exemplo 4
string1 = 'python'
string1[0] = 'l'
```

3.2 Operador Slice

É possível criar sub-strings de uma string. Já vimos que é possível pegar caracter por caracter usando o operadores slice([]); agora, vamos mostrar como é possível criar sub-strings usando o mesmo operador usando a Tabela 3.3 como guia .

Tabela 3.3: string nome organizado na memória do computador e seus respectivos índices.

Valor	l	u	c	a	s		d	e		s	o	u	z	a
índices	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
#Exemplo 5
nome = 'lucas de souza'
# o fatiamento ou slice vai imprimir do primeiro ao penúltimo
# indice da sequência.
# vai imprimir nome[1] + nome[2] + nome[3]
print(nome[1:4])
# O índice de número 4 não é impresso.
print(nome[:3])
# imprime do zero até o 2
print(nome[0:])
# imprime do zero até o último
print(nome[:])
# imprime toda a cadeia de caracteres
print(nome[:14])
# também imprime toda a cadeia de caracteres
```

3.3 Comparando Strings

Podemos usar o "=="ou "is"para comparar se duas strings são iguais; caso sejam, retornarão verdadeiro ou falso se forem diferentes. No entanto, podemos utilizar o "!="ou "is not"para verificar se são diferentes; se forem diferentes, retornará um verdadeiro, se forem iguais retornará um falso.

```
#Exemplo 6
string1 = "futebol"
string2 = "volei"
string3 = "futebol"

print(string1 == string2)
print(string1 == string3)
print(string1 != string2)
print(string1 != string3)
print(string1 is string2)
print(string1 is string3)
print(string1 is not string2)
print(string1 is not string3)
```

3.4 Métodos de Strings

Podemos observar esses métodos pela Tabela 3.4.

Tabela 3.4: métodos de String

Métodos	Descrição
capitalase()	Esse método retorna a primeira letra da string em maiúscula.
title()	Esse método retorna as primeiras letras de cada string em maiúscula.
upper()	Retorna a string em maiúscula
lower()	Retorna a string em minúscula
count(string, inicio, final)	conta a quantidade de ocorrência de uma substring
strip()	retirar os espaços do início e do final da string
index(string, inicio, final)	retorna o índice da substring passada como argumento
replace(subString antiga, substring nova)	Troca uma substring antiga por uma nova

```
50 #Exemplo 7
51 string1 = 'string sem espaço'
52 string2 = '     string com espaço na esquerda'
53 string3 = 'string com espaço na direita'
54 string4 = '    string com espaço na direita e esquerda '
55 print(string1.capitalize())
```

```
56 #Exemplo 8
57 string1 = 'string sem espaço'
58 string2 = '     string com espaço na esquerda'
59 string3 = 'string com espaço na direita'
60 string4 = '    string com espaço na direita e esquerda '
61 print(string1.title())
```

```
62 #Exemplo 9
63 string1 = 'string sem espaço'
64 string2 = '     string com espaço na esquerda'
65 string3 = 'string com espaço na direita'
66 string4 = '    string com espaço na direita e esquerda '
67 print(string1.upper())
```

```
68 #Exemplo 10
69 string1 = 'string sem espaço'
70 string2 = '     string com espaço na esquerda'
71 string3 = 'string com espaço na direita'
72 string4 = '    string com espaço na direita e esquerda '
73 print(string1.lower())
```

```
#Exemplo 11
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string1.count("s"))
```

```
#Exemplo 12
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string1.count("s", 0, 4))
```

```
#Exemplo 13
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string1.count("s", 0, 8))
```

```
#Exemplo 14
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string2.strip())
```

```
#Exemplo 15
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string3.strip())
```

```
#Exemplo 16
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string4.strip())
```

```
#Exemplo 17
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string1.index("s"))
```

```
#Exemplo 18
string1 = 'string sem espaço'
string2 = '     string com espaço na esquerda'
string3 = 'string com espaço na direita'
string4 = '    string com espaço na direita e esquerda '
print(string1.replace("sem", "com"))
```

3.5 Dicionários

Diferente das listas e tuplas, o dicionário pode ser usado para tipos de dados em que o índice não é um numeral, mas pode ser um nome ou chave como é mais comum ser chamado. As chaves devem ser únicas, mas os valores podem ser de qualquer tipo.

- Criando um dicionário

Siga a estrutura básica:

```
nome = {chave1: valor1, chave2: valor2, ...}
```

```
#Exemplo 19
#Crie um dicionário chamado aluno, com as seguintes chaves: nome, notas e media. Os
    valores podem ser qualquer um.
aluno = {
'nome':  'Maria',
'notas': [7, 7, 7, 7],
'media': 7}
print(aluno)
```

No Python existe a função dict() que cria dicionários vazios, que seria o equivalente de digitar duas chaves vazias que também cria um dicionário vazio.

```
#Exemplo 20
alunoA = {}
alunoB = dict()
print(type(alunoA), alunoA)
print(type(alunoB), alunoB)
```

```
#Exemplo 21
alunoA = dict({
'nome': 'Fernando',
'notas': [1, 10, 5, 10]})
print(alunoA)
```

- Acessando os valores do dicionário

  Para acessar os valores do dicionário a sintaxe é bem simples, veja os exemplos abaixo.

Siga a estrutura básica:

```
print(dicionario[chave])
```

```
#Exemplo 22
# criando um dicionário chamado aluno, nele existe uma chave
# nome e outra idade e os seus valores respectivamente
# são: João e 10.
aluno = {'nome': 'João', 'idade': 10}
        # vamos agora printar o valor do campo "nome", usando a
# sintaxe, vista acima; e na linha seguinte printamos
# o valor do campo "idade".
print('nome = ',  aluno['nome'])
print('idade = ', aluno['idade'])
```

- Loop for com dicionário

  Agora iremos ver como o loop "for"interage com os dicionários.

Siga a estrutura básica:

```
    for variavel  in dicionário:
print(variavel)
```

```
149 #Exemplo 23
150 aluno = {'nome': 'Marcos', 'idade': 18}
151 for x in  aluno:
152  print(x)
```

A saída foi as chaves do dicionário aluno. Agora para imprimir os valores do dicionário é bem simples.

Siga a estrutura básica:

```
    for variavel  in dicionario:
        print(dicionario[variavel])
```

```
153 #Exemplo 24
154 aluno = {'nome': 'Marcos', 'idade': 18}
155
156 for x in  aluno:
157  print(aluno[x])
```

- Métodos

  Os métodos mais comuns usados com dicionários no Python são vistos na Tabela 3.5

Tabela 3.5: métodos mais comuns usados com dicionários no Python

Métodos	Descrição
dicionario.clear()	limpa o dicionário passado
dicionario.copy()	faz uma cópia do dicionário
dicionario.items()	retorna todos os itens do dicionário do tipo valor e chave
dicionario.keys()	retorna todas as chaves do dicionários
dicionario.values()	retorna todos os valores do dicionários

```
158 #Exemplo 25
159 copia = aluno.copy()
160 print(copia)
161 print(aluno.items())
162 print(aluno.keys())
163 print(aluno.values())
164 aluno.clear()
165 print(aluno)
```

3.6 Comando with

É comum em alguns programas você deixar algum arquivo e para que esse problema não ocorra iremos utilizar o comando "with". O with permite que você trabalhe com arquivos e não precise fechá-los explicitamente.

```
166 #Exemplo 26
167 #Crie um arquivo chamado notas.txt, nesse arquivo você vai colocar as cinco notas fictí
        cias de um aluno e por fim apresentar a média dessas notas.
168 # usando o "with" para criar um arquivo no modo a:
169 with open("notas.txt", "a") as arquivo:
170 # Laço for para pegar o valor de cada nota
171  for i in range(0, 5):
172    notas = float(input("nota: "))
173 # Antes que seja enviada as notas para o arquivos
174 # precisamos transformar as notas de float para string
175 # usando a função interna str().
176    arquivo.write(str(notas) + "\n")
177 # O \n é colocado para pular uma linha
178 # Aqui iremos utilizar o with para abrir novamente o arquivo em forma de leitura
179 with open("notas.txt", "r") as leitura:
180 # Aqui vamos utilizar o método readlines() para retorna
181 # um lista de tudo que está dentro do arquivo
182  notasAluno = leitura.readlines()
183 # Variável para armazenar a soma das notas do aluno
184  soma = 0
185 # Nesse laço for ser armazenado na variável "nota" o valor de cada nota da lista
        notasAluno
186 # Esse laço funciona até a ultima nota ser pega
187  for nota in notasAluno:
188 # Aqui iremos transformar os dados que são string em floats tirando o "\n" do final usando
        o método replace como mostrado abaixo
189    soma += float(nota.replace("\n", ""))
190 # Aqui abaixo uma variável média e o cálculo da média
191  media = soma / len(notasAluno)
192 # Aqui imprimir a média
193  print("A média do aluno é: ", media)
```

3.7 Arquivos CSV

Arquivos csv ou "comma separated values"ou valores separados por vírgula. Esse tipo de arquivo é muito utilizado no excel, no google sheets e outros aplicativos.

```
194 #Exemplo 27
195 #Crie um arquivo chamado "notas.csv" que tenha como campos: nome uma string, notas uma
        lista, média um float e situação uma string. Depois apresente na tela o conteúdo
        desse arquivo.
196 # Importando a biblioteca csv para trabalhar com arquivos csv
197 import csv
198
199 # Usando o comando with e open para criar um arquivo
200 # alunos
201 with open("alunos.csv", "w") as alunos:
202 # o método writer() vai organizar como o arquivo deve
203 # ficar sua estrutura e vai receber um arquivo e um
204 # delimitador como parâmetro. Como é um arquivo csv nesse
205  csv.writer(alunos, delimiter=',').writerow(["nome", "notas", "média", "situação"])
206 # Agora podemos inserir todos os tipos dados
207 # que quisermos usando o método writerows() que recebe os
208 # dados como parâmetros.
209  csv.writer(alunos, delimiter=',').writerows([["Maria", [4, 4, 4, 4], 4, 'Final'],
210                                               ["Joana", [5, 5, 5 ,5], 5, 'Final'],
211                                               ["Clara", [7, 7, 7, 7], 7, 'Aprovada']])
212 # Agora vamos ler o arquivo
213 with open("alunos.csv", "r") as leituraNotasAlunos:
214  for aluno in csv.reader(leituraNotasAlunos, ):
215    print(aluno)
```

3.8 Declarações de Condicionais

Temos um problema: um jovem decide verificar se sua idade já é suficiente para o alistamento militar; para isso, ele entrou no site das forças armadas do Brasil para checar. Então, verificando a idade mínima, para poder se alistar, ele viu que os jovens com idade para o alistamento deveriam seguir alguns procedimentos, os outros que ainda não tinham idade não precisam se preocupar. No Python existe uma estrutura que lida com condições, e essa estrutura é:

- IF;
- IF-ELSE;
- IF-ELIF-ELSE

3.9 Declarações simples IF

Voltando ao problema acima, podemos escrever uma solução bem simples, usando a estrutura condicional IF:

Siga a estrutura básica:

```
if Condição:
    Bloco de código que será executado caso a condição seja verdadeira.
```

```
# Exemplo 1:
idade = int(input('Informe a sua idade: '))
if idade >= 16:
    print('Procure uma junta militar mais próxima da sua casa para fazer o seu alistamento
    .')
```

```
#Exemplo 2
if idade >= 16:
    print('Procure uma junta militar mais próxima da sua casa para fazer o seu alistamento
    .')
```

3.10 Declarações Compostas

- IF - ELSE

No exemplo anterior a gente poderia colocar uma mensagem informando ao jovem caso ele não tenha atingido a idade mínima para o alistamento, como no exemplo a seguir:

Siga a estrutura básica:

```
if Condição:
    Bloco de código que será executado caso a condição seja verdadeira.
else:
    Bloco de código que será executado caso a condição seja Falsa.
```

```
#Exemplo 3
idade = int(input('Informe a sua idade: '))
if idade >= 16:
  print('Procure uma junta militar mais próxima da sua casa para fazer o seu alistamento.'
    )
else:
  print('Você ainda não atingiu a idade mínima para o seu alistamento.')
```

```
#Exemplo 4
idade = int(input('Informe a sua idade: '))
if idade >= 16:
  print('Procure uma junta militar mais próxima da sua casa para fazer o seu alistamento.'
    )
else:
  print('Você ainda não atingiu a idade mínima para o seu alistamento.')
```

- IF - ELIF - ELSE

Imagine outro problema: um aluno de matemática resolve criar um programa para verificar se seu IMC(índice de massa corporal) e dos seus familiares. Para isso, ele verifica como se calcula o IMC e consulta uma tabela na internet sobre as várias faixas. Como neste problema, temos algumas faixas de valores, podemos representar essas várias faixas, usando a estrutura abaixo:

```
#Exemplo 5
peso = float(input('Informe o peso: '))
altura = float(input('Informe a altura: '))
IMC = (peso / ( altura * altura))
print('IMC = ', IMC)
if IMC < 18.5:
  print('Você está abaixo do peso!')
elif 18.5 <= IMC <= 24.9:
  print('Seu peso está normal.')
else:
  print('Você está acima do peso. Procure um médico')
```

3.11 Estruturas de Repetições

Imagine que você precise imprimir uma mensagem de boas vindas cinco vezes na tela, faça um programa em Python que imprima essas cinco mensagens sem usar laços de repetição.

```
#Exemplo 6
print('Seja Bem-Vindo')
print('Seja Bem-Vindo')
print('Seja Bem-Vindo')
print('Seja Bem-Vindo')
print('Seja Bem-Vindo')
```

No programa acima vemos como é trabalhoso ficar repetindo a função print() para repetir a mesma tarefa. E, quando precisamos repetir algo no Python, nós utilizamos os famosos laços de repetição. E no Python existem dois tipos de estrutura para trabalhar com repetições, que são elas o "for"e o "while".

- FOR

A estrutura de repetição for é muito utilizada quando sabemos a quantidade de vezes que queremos repetir determinado bloco, vamos ver um exemplo:

Siga a estrutura básica:

```
Sintaxe:
for valor in sequencia:
    bloco que vai repetir
```

```
#Exemplo 7
#Queremos fazer uma contagem de 1 a 10
for i in range(1, 11):
  print(i, end=' ' )
```

Vamos comparar esse programa criado com a sintaxe logo acima. Primeiro, a função "range"cria uma sequência que começa no primeiro número dentro dos parênteses, no nosso exemplo o número

1, e termina como uma unidade a menos que o segundo número, no caso 10. Em segundo, a variável "i"é responsável por pegar cada número da sequência criada. Então, na primeira repetição, a variável "i"pega o primeiro número da sequência; na segunda vez, o "i", é o segundo número da sequência; e vai ficar repetindo até o último número da sequência. Por isso que demos print na variável "i".

- WHILE

Já o laço "while"é usado mais quando a gente desconhece a quantidade de vezes que queremos repetir um bloco de código. Para não ficar repetitivo infinitas vezes, é testado uma condição, caso essa condição seja verdadeira, o bloco de código é executado, caso contrário o bloco não será executado.

Siga a estrutura básica:

```
Sintaxe:

while condição:
    bloco de código
```

```
41 #Exemplo 8
42 #Vamos criar o mesmo contador que foi criado no laço for de 1 até 10
43
44 # primeiro vamos criar uma variável para ser nosso contador e
45 # iniciar ela com um valor 1
46
47 contador = 1
48 # Agora queremos contar de 1 até 10. Para isso, podemos criar um loop while,
49 # que imprima o
50 # valor da variável contador, enquanto ela for menor que 10.
51 while contador <= 10:
52 # como contador começou em "zero" e queremos que nossa
53 # contagem começa no "um",
54 # somamos mais "um" ao contador.
55   print(contador, end=' ')
56 # se a gente deixar a variável contador do jeito que ela está,
57 # ela nunca deixará de ser zero,
58 # então somamos sempre no final do loop, para que o valor
59 # do contador aumente a cada iteração
60   contador = contador + 1
```

3.12 Lista e Tupla

- O que é uma Lista?

Lista é uma coleção de dados, podendo ser de vários tipos e a lista é mutável. A Tabela 3.6 descreve os métodos para manipulação de listas em Python.

Siga a estrutura básica:

```
Sintaxe:
lista1 = [valor1, valor2, valor3, ...]
```

```
#Exemplo 9
#Crie uma lista de várias formas
lista1 = list(('produto', 'preço', 'quantidade'))
lista2 = ['maçã', 2.3, 4]
lista3 = ['uva', 0.24, 10]
lista4 = list(range(0, 10))
lista5 = list('Python')


print(type(lista1),type(lista2),type(lista3),type(lista4),type(lista5))
print(lista1)
print(lista2)
print(lista3)
print(lista4)
print(lista5)
```

- Métodos de uma Lista

Podemos observar esses métodos pela Tabela 3.6.

Tabela 3.6: Métodos para manipulação de listas em Python

Métodos	Descrição
append()	Adiciona elementos no final da lista
insert(índice, elemento)	Adiciona no índice especificado o elemento
remove(elemento)	Remove o elemento passado
pop(índice)	Retorna o elemento do índice que foi removido

```
#Exemplo 10
lista1 = [1, 2, 3, 4, 5]
lista2 = ["maria", "joão", "marcos", "josé", "benjamin", "luiz"]
# Método append()
lista1.append(6)
print(lista1)
# Método insert()
lista2.insert(2, "joaquim")
print(lista2)
# Método remove()
lista2.remove("joaquim")
print(lista2)
# Método pop()
lista1.pop()
print(lista1)
```

```
#Exemplo 11
# Tentando remover um elemento da lista com remove()
# passando um nome que não existe na lista
lista2.remove("lucas")
```

```
#Exemplo 12
# Tentando remover um elemento da lista com pop(),
# passando um índice que não existe que seja maior
# ou igual ao tamanho da lista.

len(lista2)
lista2.pop(len(lista2))
```

- O que é uma Tupla?

Tupla é uma coleção de dados, ela pode armazenar qualquer valor de qualquer tipo; podemos dizer que a tupla é uma coleção de dados separados por vírgula. A tupla é imutável: então não é possível reatribuir um valor.

Siga a estrutura básica:

```
Sintaxe:
tupla = valor1, valor2, valor3, ...
tupla = (valor1, valor2, valor3, ...)
```

```
#Exemplo 13
# Crie algumas tuplas de maneiras diferentes
tupla1 = tuple(['produto', 'preço', 'quantidade'])
tupla2 = ('maçã', 2.3, 4)
tupla3 = 'uva', .24, 10
print(type(tupla1), type(tupla2), type(tupla3))
```

- Indexação de Lista e Tupla

A indexação de listas e tuplas são as mesmas, começa pelo índice zero e termina no índice que é uma unidade a menos que o tamanho da lista ou tupla. A Tabela 3.7 exemplifica os índices de tupla e de lista em Python.

Tabela 3.7: Índices de tupla e de lista em Python.

Valores	1	2	3	4
índice tupla	0	1	2	3
índice lista	0	1	2	3

```
#Exemplo 14
tupla1 = (1, 2, 3, 4)
lista1 = (1, 2, 3, 4)
print('Imprimindo valores da tupla: ')
print(tupla1[0])
print(tupla1[1])
print(tupla1[2])
print(tupla1[3])

print('Imprimindo valores da lista: ')
print(lista1[0])
print(lista1[1])
print(lista1[2])
print(lista1[3])
```

3.13 Funções

Quando alguns comandos ficam muito repetitivos em um determinado programa, é comum usar funções para não ter que ficar repetindo a mesma sequência de comandos.

```
#Exemplo 15
#Faça um programa que imprima na tela:
#
#=====================
#Python
#=====================
#Programação
#=====================

print(20 * '=')
print("python")
print('=' * 20)
print("programação")
print('=' * 20)
```

Nesse programa, utilizamos muito a função print para imprimir tanto as palavras quanto as linhas. Imagine agora termos que fazer isso para mais de dez palavras ? mil palavras ? Imaginou ? Seu programa irá ficar imenso, não é ? Agora vamos fazer o mesmo programa usando o conceito de função.

```
Sintaxe:
def nome_função(parametro1, parametro2, ...):
    bloco de comandos
```

```
#Exemplo 16
def imprimirPalavras(palavra1, palavra2):
 print(20 * '=')
 print(palavra1)
 print('=' * 20)
 print(palavra2)
 print('=' * 20)

imprimirPalavras('Python', 'Programador')
imprimirPalavras('Engenharia', 'UFPE')
```

Para criarmos nossas funções: primeiro temos que usar a palavra reservada "def", depois dar um nome para a função, pode ser qualquer um, mas que identifique a função; também pode ser passado parâmetros, caso precise, e por último você declara o bloco de código com indentação.

```
#Exemplo 17
#Faça um programa, com uma função que necessite de três
#argumentos, e que forneça a soma desses três argumentos.
# O nome da função deve indicar o que ela pretende fazer,
# Essa função só vai imprimir o valor da soma dos três
# números
def somarTresNumeros(num1, num2, num3):
 print(num1 + num2 + num3)

# Aqui é chamada a função
somarTresNumeros(1, 2, 3)
```

```
#Exemplo 18
# poderíamos armazenar o valor da soma em uma nova
# variável e utilizar em outras partes do código
def armazenarSomaTresNumeros(num1, num2, num3):
# aqui irei criar uma nova variável
 soma = num1 + num2 + num3
# E com o uso da palavra reservado 'return', retornaremos
# o valor da soma
 return soma
# Aqui iremos armazenar o valor da soma de 1, 2 e 3 dentro de soma1
soma1 = armazenarSomaTresNumeros(1, 2, 3)
# soma2 faz igual a soma1
soma2 = armazenarSomaTresNumeros(2, 3, 4)
# E agora printamos o valor de soma1 * soma2
print(soma1 * soma2)
```

```
#Exemplo 19
# O código acima poderia ficar mais simples:
def armazenarSomaTresNumeros(num1, num2, num3):
 return num1 + num2 + num3

soma1 = armazenarSomaTresNumeros(1, 2, 3)
soma2 = armazenarSomaTresNumeros(2, 3, 4)
print(soma1 * soma2)
```

```
#Exemplo 20
# Por último existe uma maneira de passar uma quantidade
# variável de argumentos, colocando um asterisco na
# frente de apenas um parâmetro dentro do parêntese da função.
# Veja abaixo:
def somarArgumentosVariavel(*num):
# o asterisco na frente transforma o parâmetro num
# em um tupla, que podemos pegar cada valor da seguinte
# forma. Mas antes vamos criar uma variável soma, para
# guarda a soma dos números passado
 soma = 0
 for i in num:
# aqui vamos pegar cada valor da tupla num e somar com
# a variável soma.
   soma += i
 return soma

# E agora podemos passar a quantidade que quisermos, que
# o programa acima vai somar os números
soma1 = somarArgumentosVariavel(1)
soma2 = somarArgumentosVariavel(1, 2)
soma3 = somarArgumentosVariavel(1, 2, 3, 4, 5, 6, 7)

print(soma1,soma2,soma3)
```

3.14 Excessões no Python

No Python é comum acontecer dois tipos de erro: erro de sintaxe e de lógica. O primeiro tipo de erro pode acontecer quando não cumprimos as regras estabelecidas na linguagem. Já o erro de lógica acontece quando: não existindo erro de sintaxe, o nosso programa tenta executar algo impossível: como tentar modificar o valor em uma posição de uma tupla; ou tentar modificar a letra de uma string; ou tentar dividir por zero e muitas outras coisas. Quando isso acontece, o nosso programa lança uma exceção e deixa de funcionar.

- Tratando exceções

Siga a estrutura básica:

```
Para tratar exceções é só preciso utilizar o bloco try - exception.
try:
   bloco de código
except:
   bloco de código
```

```
#Exemplo 21
# Iremos tentar dividir um número por zero
print(2 / 0)
```

Agora iremos tratar esse erro usando o bloco try - exception:

```
#Exemplo 22
# Iremos tentar dividir um número por zero
# usando o try - except
try:
 print(2 / 0)
except:
 print("Divisão por zero!")
```

Imagine agora que você deseja pegar o dado da divisão, caso não exista nenhum problema de divisão por zero, e imprimir esse valor. Como fazer isso ?

```
#Exemplo 23
# Nosso exercício aqui é imprimir o valor da divisão caso
# não ocorra nenhuma exceção.
a = int(input('Digite um número: '))
b = int(input('Digite outro número: '))

try:
 divisao = a / b
except ZeroDivisionError as err:
 print(err)
# Podemos usar o "else" para executar parte do código
# quando o except não for ativado. A ideia aqui é
# parecido com if - else, no caso except - else. Caso
# ocorra alguma exceção, o bloco do except funciona; não
# ocorrendo nenhuma exceção, o bloco do else funciona.
else:
 print(divisao)
```

3.15 Arquivo no Python

No Python é utilizada a função "open()"para abrir ou criar um arquivo. Essa função vai receber dois parâmetros: o primeiro é o nome do arquivo e mais sua extensão e o segundo é o modo como você quer abrir o arquivo. A Tabela 3.8 descreve a forma de leitura e de escrita de arquivo em Python.

- Criando arquivos e escrevendo no Python

Tabela 3.8: Modo de leitura e de escrita de arquivo em *Python*.

Parâmetro	Descrição
r	Modo de leitura do arquivo
w	Modo de escrita. Esse modo cria um arquivo caso ele não exista e se existir, ele vai sobrescrever.
a	Modo de escrita. Ele cria o arquivo, caso não exista, e pega o conteúdo e insere no fim do arquivo.
x	Retorna um erro se o arquivo existir. Se não existir, ele vai criar o arquivo.

```
#Exemplo 24
# O método open() recebe dois parâmetros:
# o nome do arquivo e o modo de abertura do arquivo.
# Ele cria o arquivo caso ele não exista.
file1 = open("alunos.txt", "w")
# Função write escreve no arquivo caso ele já exista.
file1.write("Alô Mundo!")

file2 = open("professores.txt", "a")
file2.write("Alô Mundo!")

# É preciso fechar cada arquivo aberto, usando o método
# close(), já que o método open() abriu.
file1.close()
file2.close()
```

- Lendo um arquivo

Para ler um arquivo no Python, iremos utilizar o método read() e o readlines().
No exemplo 39 seguiremos a instrução: Crie um arquivo, usando o modo "w", chamado texto.txt. Logo após, use o loop for para escrever novas linhas dentro do arquivo, usando o modo "a". Por fim, use o método read() e readlines() e perceba a diferença entre os dois métodos.

```
#Exemplo 25
file1 = open("texto.txt", "w")
file1.close()

file1 = open("texto.txt", "a")

for i in range(0, 3):
 file1.write(str(i) + "- texto " + str(i) + "\n")

file1.close()

file1 = open("texto.txt", "r")
print(file1.read())
file1.close()

file1 = open("texto.txt", "r")
print(file1.readlines())
file1.close()
```

3.16 Exercícios

1. Faça um programa que leia um nome de usuário e a sua senha e não aceite a senha igual ao nome do usuário, mostrando uma mensagem de erro e voltando a pedir as informações.

2. Faça um programa que leia e valide as seguintes informações:

```
Nome: maior que 3 caracteres;
Idade: entre 0 e 150;
Salário: maior que zero;
Sexo: 'f' ou 'm';
Estado Civil: 's', 'c', 'v', 'd';
```

3. Faça um programa que imprima na tela os números de 1 a 20, um abaixo do outro. Depois modifique o programa para que ele mostre os números um ao lado do outro.

4. Faça um programa que imprima na tela apenas os números ímpares entre 1 e 50.

5. O Sr. Manoel Joaquim possui uma grande loja de artigos de R$ 1,99, com cerca de 10 caixas. Para agilizar o cálculo de quanto cada cliente deve pagar ele desenvolveu um tabela que contém o número de itens que o cliente comprou e ao lado o valor da conta. Desta forma a atendente do caixa precisa apenas contar quantos itens o cliente está levando e olhar na tabela de preços. Você foi contratado para desenvolver o programa que monta esta tabela de preços em **arquivo de planilha eletrônica (.csv)**, que conterá os preços de 1 até 50 produtos, conforme o exemplo abaixo:

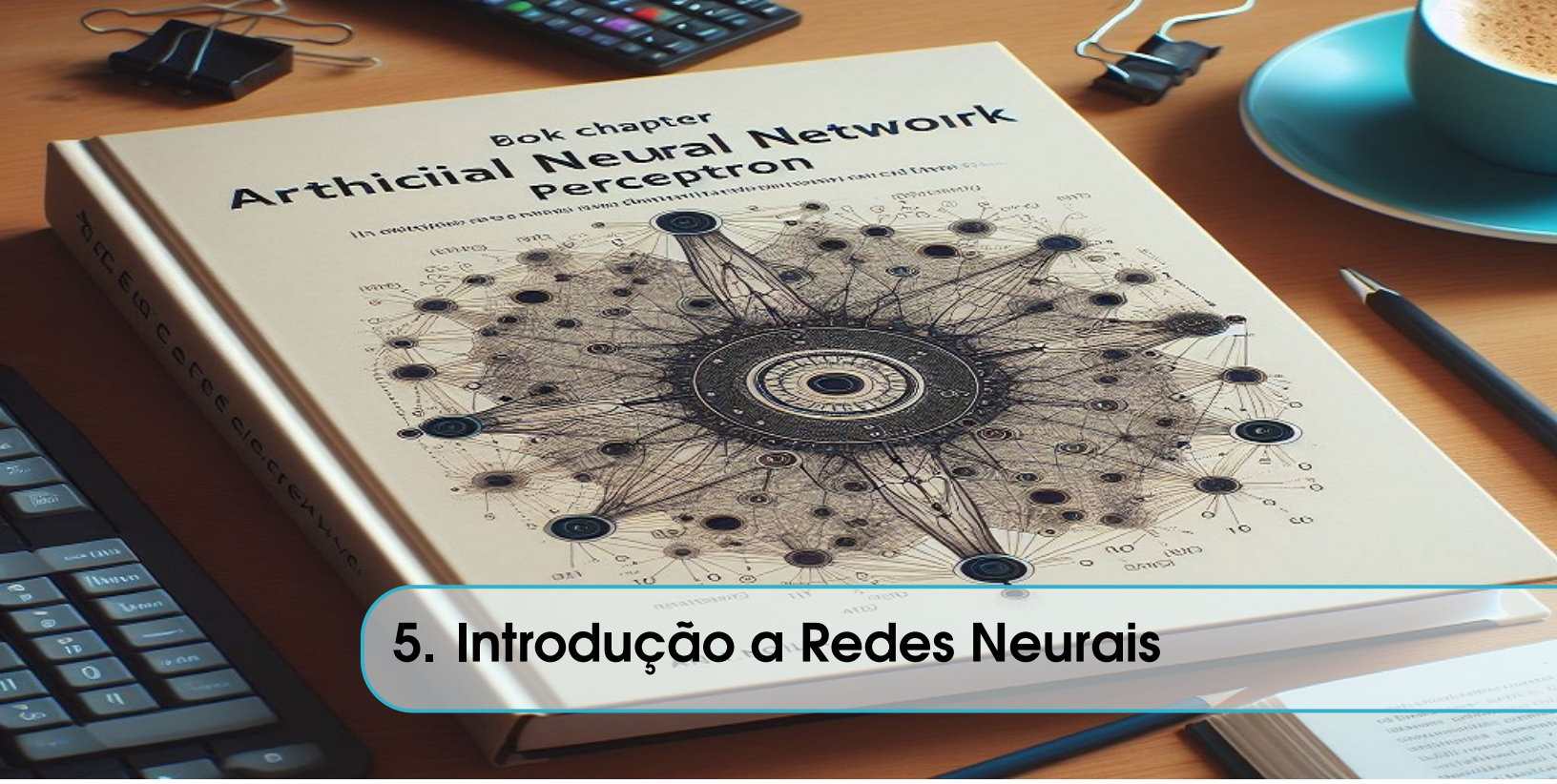
```
Lojas Quase Dois - Tabela de preços
1 - R$ 1.99
2 - R$ 3.98
...
50 - R$ 99.50
```

6. O Sr. Manoel Joaquim acaba de adquirir uma panificadora e pretende implantar a metodologia da tabelinha, que já é um sucesso na sua loja de R$ 1,99. Você foi contratado para desenvolver o programa que monta a tabela de preços de pães em **arquivo de planilha eletrônica (.csv)** A contagem vária de 1 até 50 pães, a partir do preço do pão informado pelo usuário, conforme o exemplo abaixo:

```
Preço do pão: R$ 0.18
Panificadora Pão de Ontem - Tabela de preços
1 - R$ 0.18
2 - R$ 0.36
...
50 - R$ 9.00
```

4. Manipulação de Matrizes em Python

4.1 Introdução



5. Introdução a Redes Neurais

As técnicas de inteligência artificial são bio-inspiradas. Um dos mecanismos inteligentes de maior sucesso diz respeito às redes neurais artificiais. Trata-se de uma inspiração no funcionamento do cérebro humano. O desenvolvimento do cérebro da espécie humana ocorre principalmente após seis meses até os dois primeiros anos de vida. Durante essa fase, a criança tem uma insurgência de novas capacidades perceptivas, cognitivas e motoras [22].

Esse modelo de aprendizado serve de inspiração para muitos pesquisadores, que tentam simular o funcionamento do cérebro humano, principalmente o processo de aprendizagem por experiência. As redes neurais artificiais são inspiradas na compreensão da biologia sobre o cérebro da espécie humana [29]. Mas ao contrário de um cérebro biológico onde qualquer neurônio pode se conectar a qualquer outro neurônio dentro de uma certa distância física, as redes neurais artificiais possuem camadas discretas, conexões e direções de propagação de dados [29].

Nas redes neurais artificiais, somente uma parcela de neurônios pode receber as informações oriundas da aplicação alvo. Essa parcela de neurônios é nomeada de camada de entrada. Em termos didáticos, somente a camada de entrada recebe informações do mundo externo. Na sequência, uma outra parcela de neurônios pondera as informações oriundas da camada de entrada. A referida parcela é tecnicamente nomeada de camada escondida, também chamada de camada discreta. Cabe à camada escondida dar a importância ou a falta de importância dos neurônios oriundos da camada de entrada. Por fim, há a última parcela de neurônios, nomeada de camada de saída. O papel da camada de saída é reduzir a dimensionalidade da aplicação.

Quanto se trata de classificação (categorização), a camada de saída polariza as informações oriundas da camada escondida entre as classes da aplicação (ex.: *malware*, benigno). As redes de neurais são capazes de segregar *malware* de aplicativos benignos com acurácia média acima de 98% [17][18][19][26]. Quando se emprega redes neurais, faz-se necessário catalogar uma quantidade estatisticamente relevante de amostras da classe alvo assim como também da contra-classe. Durante a etapa de treinamento (aprendizado), as conexões entre os neurônios se ajustam de modo que as características auditadas da aplicação alvo passam a ter um conjunto de pesos ponderados. Através do ajuste das conexões entre os neurônios, as amostras pré-rotuladas pelo profissional especialista podem ser reconhecidas entre classe e contra-classe pela rede neural. A etapa de treinamento assume

grande relevância porque características (neurônios de entrada) aparentemente contraditórias podem acontecer simultaneamente. Por exemplo, torna-se possível detectar um *malware* que se comporta como aplicativo sério na maior parte do tempo. Tal fato dificilmente seria detectado por um humano sem o suporte de uma máquina de aprendizado estatístico a exemplo de uma rede neural.

Após a sua etapa de treinamento, a rede neural pondera se a amostra suspeita apresenta características mais próximas da classe alvo em comparação à contra-classe. Em termos técnicos, uma rede neural sabe que o aplicativo inédito contém atividades maliciosas porque ele tem um padrão mais distinto em comparação a um app. sério previamente estudado. Em termos didáticos, uma rede neural especializada em animais sabe que na imagem há um leão porque não há um elefante. As características estariam mais próximas da classe do que da contra-classe.

Quanto se trata de predição, a rede neural faz a estimativa de um valor escalar. Por predição, entende-se uma previsão com rigor científico-metodológico. Ao longo de seu treinamento, a rede neural, em específico o neurônio de saída, é apresentada a uma série histórica temporal [32]. Por exemplo, a rede neural é apresentada à série histórica da cotação diária do barril do petróleo durante os últimos anos. Enquanto o neurônio de saída é apresentado a cotação diária do petróleo, os neurônios de entrada são apresentados aos acontecimentos no referido período. Por acontecimentos denota-se; decisões governamentais, eleições, tentativas de golpes de estado e fenômenos da natureza a exemplo de estiagens, dentre outros fatores.

Ainda quanto ao treinamento, as conexões entre os neurônios se ajustam de modo que os acontecimentos periódicos passam a ter um conjunto de pesos ponderados. Através do ajuste das conexões entre os neurônios, acontecimentos diários aparentemente irrelevantes mostram poder de interferência direta na cotação diária do petróleo. É estatisticamente improvável que um investidor humano conseguisse associar um acontecimento aparentemente desimportante a variações diárias no valor do petróleo. Sem a rede neural, também é estatisticamente improvável que um humano conseguisse associar que um acontecimento há seis meses atrás fosse interferir no preço do petróleo na presente data.

Quando aplicada à predição, as redes neurais assumem papel nobre na detecção e prevenção de desastres naturais. As redes neurais têm a competência de alertar o exato momento no qual ocorrerá uma catástrofe natural como uma inundação. Em síntese, as redes neurais assumem participação fundamental na construção de metamodelos de cidades inteligentes. Por cidades inteligentes, denota-se a aquisição de dados por diferentes sensores eletrônicos de modo que se possa atenuar as consequências de uma catástrofe natural.

As redes neurais clássicas também são chamadas de redes neurais rasas. O referido batismo ocorre devido à sua baixa complexidade computacional. Portanto são capazes de funcionar em qualquer computador comum, seja na sua fase de treinamento (aprendizado) seja na fase de uso. O treinamento é concluído em questão de minutos porque há pouco volume de neurônios artificiais. Consequentemente, o ajuste das conexões entre os neurônios ocorre de forma célere, mesmo que as referidas conexões sejam inicializadas de forma aleatória. Na fase de uso, o tempo de resposta da rede rasa se dá em segundos. Desde que bem parametrizada, as redes neurais rasas são capazes de obter acurácias estatisticamente equivalentes à qualquer modelo de rede profunda (*Deep Learning*) [5][18].

Siga as instruções:

1. Para este capítulo do livro, utilizaremos o conceito de rede neural *Perceptron*. Criado em 1958 por Frank Rosenblatt, o *Perceptron* também é conhecido como *MLPs* (Multi-Layers Perceptron).

Cliente	Cartão fidelidade	Compra > R$50,00	Pagamento em dinheiro	Categoria
1	Sim	Sim	Sim	Diamante
2	Sim	Sim	Não	Diamante
3	Sim	Não	Sim	Ouro
4	Sim	Não	Não	Ouro
5	Não	Sim	Sim	Prata
6	Não	Sim	Não	Prata
7	Não	Não	Sim	Bronze
8	Não	Não	Não	Bronze

Este método consiste em trabalhar com uma camada de entrada, as camdas ocultas, compostas pelos neurônios, e a camada de saída.

Aqui, vamos trabalhar com o *Perceptron* de duas camadas, ou seja dois neurônios, com um exemplo que recriaremos em Python.

Numa determinada livraria, existe uma classificação de clientes, conforme sua fidelidade, valor de compra e método de pagamento. Essa classificação confere um desconto na hora do pagamento e devemos aplicar esse sistema com uma rede *Perceptron*.

Ao realizar o pagamento, o cliente deve responder três perguntas:

- O cliente possui cartão fidelidade da livraria?
- O valor da compra é superior a R$50,00?
- O pagamento será realizado em dinheiro?

A tabela a seguir apresenta um conjunto de clientes classificados em cada categoria, conforme as perguntas foram respondidas:

Primeiramente, qual deve ser a arquitetura mínima da nossa rede neural *Perceptron*?

Como os clientes estão classificados em quatro categorias, teremos que utilizar dois neurônios, assim, teremos a seguinte situação:

6. Repositório de Aprendizado

6.1 Introdução

Técnicas de inteligência artificial se tornam especialistas em um aplicação alvo a partir de um repositório de aprendizado estatístico. A inteligência artificial adquire capacidade de generalização a partir das amostras reservadas à fase de treinamento (aprendizado). Quando aplicada a reconhecimento de padrões, é fundamental a escolha das amostras que irão compor o repositório de aprendizado estatístico. Um profissional especialista e/ou um instrumento bem delineado atribuem uma classe (rótulo) a cada amostra investigada a partir de seus atributos. Com base em um conjunto de amostras, chamado de conjunto de treinamento, é possível formular uma hipótese sobre as diferentes classes vinculadas à aplicação alvo. Após treinamento, cabe a inteligência artificial estimar a classe de uma amostra inédita. O referido feito ocorre através da comparação entre os atributos auditados em tempo real e aqueles contidos no repositório de aprendizado estatístico criado para a aplicação alvo.

Faz-se necessário catalogar uma quantidade estatisticamente relevante de amostras da classe alvo assim como também da(s) contra-classe(s). A etapa de treinamento assume grande relevância porque atributos aparentemente contraditórios podem acontecer simultaneamente. Por exemplo, torna-se possível detectar um aplicativo malicioso que se comporta como aplicativo sério na maior parte do tempo. Tal fato dificilmente seria detectado por um humano sem o suporte de uma máquina de aprendizado estatístico a exemplo da inteligência artificial.

6.2 Análise Estática

O presente capítulo estabelece as bases para a criação de repositório de aprendizado estatístico aplicado a antivírus. Os atributos do aplicativo suspeito devem ser extraídos a partir de duas abordagens; análise estática e análise dinâmica. Visando suprir as limitações e imprecisões dos antivírus comerciais, o estado-da-arte emprega a análise do código de montagem do arquivo nomeada de análise estática. De maneira preventiva, o aplicativo pode ser estudado e, portanto, é possível investigar a intenção maliciosa do arquivo. A análise estática é capaz de identificar o *modus operandi* de um *malware*, antes mesmo de ser executada pelo usuário.

Na análise estática, o executável passa por um processo de Engenharia Reversa visando reverter o arquivo executável em seu código de montagem. Em linguagens compiladas, a Engenharia Reversa é nomeada de *disassembling*. O objetivo é reverter o arquivo binário (*.exe, *.sys, *.dll) em linguagem de máquina (real).

Uma das desvantagens da análise estática é que cada arquitetura computacional tem seu próprio repertório de instruções, tecnicamente, nomeado de *assembly*. Por exemplo, o *assembly* do Windows pode ser diferente do *assembly* para o Linux. O *disassembling* para aplicativos Windows não serve para Linux.

Outra desvantagem é que um extrator estático de características só funciona para o aplicativo alvo. Por exemplo, um extrator de arquivos binários (*.exe, *.sys, *.dll) é incapaz de extrair características de um *bytecode* em Java. *Bytecode* é um código escrito na linguagem Java capaz de ser interpretado pelas Máquinas Virtuais Java

6.2.1 *Androwarn*: Extrator Estático de Características para Android

Smartphones e *tablets* móveis estão gradativamente se tornando indispensáveis na vida diária. Em tempos contemporâneos, há a ampla difusão da rede mundial de computadores através do uso de aplicativos em redes sociais, com aplicações que não focam somente na diversão e no lazer, mas também no trabalho. Como efeito colateral, o Android, por ser um sistema operacional relativamente recente, possibilita que incontáveis *malware* se escondam em uma grande quantidade de aplicações legítimas. Tal fato ameaça, de forma grave, a segurança do sistema e por consequência os dados íntimos dos usuários.

Para realizar a análise estática, de maneira preventiva, utilizou-se a ferramenta Androwarn[1], no qual o principal objetivo é detectar e alertar ao usuário sobre possíveis comportamentos maliciosos desenvolvidos por um aplicativo Android. Essa análise leva à geração de um relatório, de acordo com o nível de detalhe técnico escolhido pelo usuário.

A extração de características dos arquivos .apk passa pelo processo de Engenharia Reversa. A seguir, são detalhados alguns grupos de características que podem ser identificadas pelo Androwarn referentes aos arquivos investigados.

- Características relacionadas à interceptação de áudio e vídeo. A forense digital averigua se a aplicação suspeita testada tenta:
  - Gravar o áudio do aparelho;
  - Capturar em vídeo a tela do dispositivo.
- Características relacionadas à informação sobre a geolocalização do dispositivo. A forense digital averigua se a aplicação suspeita tenta ler as informações de localização de todos os provedores disponíveis (WiFi, GPS, etc).
- Características relacionadas ao vazamento de PIM (Protocol Independent Multicast). O PIM é um protocolo de rastreamento de padrões da internet (BOERS; WIJNANDS; ROSEN, 2008). A forense digital averigua se a aplicação suspeita tenta:
  - Acessar a lista de SMS;
  - Acessar a lista de SMS/MMS;
  - Acessar a lista de MMS, Serviço de Mensagens Multimídia (Multimedia Mensaging Service) (LE BODIC, 2005);
  - Acessar a lista de contatos;

[1] Androwarn: ferramenta visando a extração de características em aplicativos Android .apk. Disponível em: https://github.com/maaaaz/androwarn. Acesso em abril de 2022

  - Acessar a lista de contatos pelo Android;
  - Acessar a lista de contatos pelo Google;
  - Acessar a pasta de downloads;
  - Acessar o calendário;
  - Acessar o calendário pelo Android;
  - Acessar o registro de chamadas;
  - Acessar à rádio FM;
  - Acessar o browser nativo;
  - Acessar o correio de voz;
  - Acessar o histórico de sincronização do aparelho;
  - Acessar o Bluetooth;
  - Acessar os dados do aplicativo de email;
  - Acessar os dados armazenados na prancheta.
- Características relacionadas à execução de códigos. A forense digital alerta se a aplicação suspeita tenta:
  - Carregar uma biblioteca nativa (Informando o nome da biblioteca quando possível);
  - Executar um comando UNIX (Comandos em arquivos: ls – listar conteúdo do diretório, rm – remover arquivos; cp – copiar arquivos, etc).
- Características relacionadas à interface de conexões. A forense digital denuncia se a aplicação suspeita:
  - Lê detalhes sobre a rede de dados atualmente ativa;
  - Tenta descobrir se a rede de dados atualmente ativa é medida;
  - Lê as credenciais da rede de WiFi.
- Características relacionadas às configurações do aparelho. A forense digital alerta se a aplicação suspeita:
  - Recupera informações sobre outro aplicativo instalado no aparelho, identificando quais informações recuperadas e qual o outro aplicativo atingido;
  - Lista os aplicativos instalados no sistema;
  - Lista bibliotecas compartilhadas no sistema.
- Características relacionadas a conexões remotas. A forense digital alerta se a aplicação suspeita abre um Socket, interface genérica para vários protocolos, e o conecta para um endereço remoto. A forense é capaz de identificar esse endereço e qual a porta utilizada.
- Características relacionadas aos identificadores de telefone. A forense digital alerta se a aplicação suspeita:
  - Lê o valor do código de área do local;
  - Lê o valor do ID do telefone;
  - Lê o estado atual do telefone;
  - Identifica a localização atual do aparelho;
  - Lê o tipo de atividade em uma conexão de dados;
  - Lê o estado atual da conexão de dados;
  - Lê o ID do dispositivo exclusivo, ou seja, o IMEI, Identidade Internacional de Equipamentos Móveis (International Mobile Equipment Indentity).
  - Lê o GSM, Sistema Global para Comunicações Móveis (Global System for Mobile Communications).
  - Lê o MEID (Mobile Equipment Identifier), identificador de um módulo de rádio;
  - Lê o ESN (Eletronic Serial Number) para telefones CDMA (Code Division Multiple

Access), que é um método de acesso a canais em sistemas de comunicação;
- Lê o número da versão do software do dispositivo, por exemplo, o IMEI / SV para telefones GSM;
- Lê a sequência de números de telefone da linha 1, por exemplo, o MSISDN que é o mapeamento entre o nº do telemóvel ao cartão SIM (Subscriber Identity Module) de um telefone GSM. SIM é um circuito impresso do tipo cartão inteligente utilizado para identificar, controlar e armazenar dados de telefones celulares;
- Lê as informações dos aparelhos vizinhos do dispositivo;
- Lê o código ISO do país equivalente ao atual MCC (código móvel do país) do operador registrado;
- Lê o nome numérico (MCC + MNC) do atual operador registrado. MNC (Mobile Network Code) é um parâmetro para definir informações importantes sobre o país, operadora e região geográfica;
- Lê o nome do operador;
- Lê a tecnologia de rádio (tipo de rede) atualmente em uso no dispositivo para transmissão de dados;
- Lê o valor do tipo de telefone do dispositivo;
- Lê o código ISO do país equivalente ao código do país do provedor SIM;
- Lê o MCC + MNC do provedor do SIM;
- Lê o nome do provedor de serviços (SPN);
- Lê o número de série do SIM;
- Lê a constante indicando o estado do cartão SIM do dispositivo;
- Lê o ID exclusivo do assinante, por exemplo, o IMSI, número que identifica exclusivamente todos os usuários de uma rede celular, para um telefone GSM;
- Lê o identificador alfabético associado ao número do correio de voz;
- Lê o número da caixa postal.

- Características relacionadas aos serviços do telefone. A forense digital alerta se a aplicação suspeita:
  - Envia uma mensagem de SMS do telefone, indicando a mensagem e o número do destinatário;
  - Intercepta os SMS recebidos;
  - Desativa a entrada de SMS no aparelho;
  - Faz chamadas telefônicas.

Siga as instruções:

1. Faça o *Download* da base de dados autoral contendo aplicativos maliciosos *Android*. Cabe atentar que as amostras estão sem a extensão .apk. O objetivo é evitar infecções acidentais, mas se tratam aplicativos *Android*.
   - Clique no seguinte link https://github.com/DejavuForensics/AndroidSamples. Em seguida, faça o *Download* conforme mostra a Fig. 6.1. No seu computador, na pasta ***"Downloads/AndroidSamples-master"***, clique com o lado direito do mouse e escolha ***"Extract Here"***.

2. Instalação do *Androwarn* de modo a fazer a análise de aplicativos Android;
   - No console, instale o *Androwarn*.

```
pip install androwarn
```

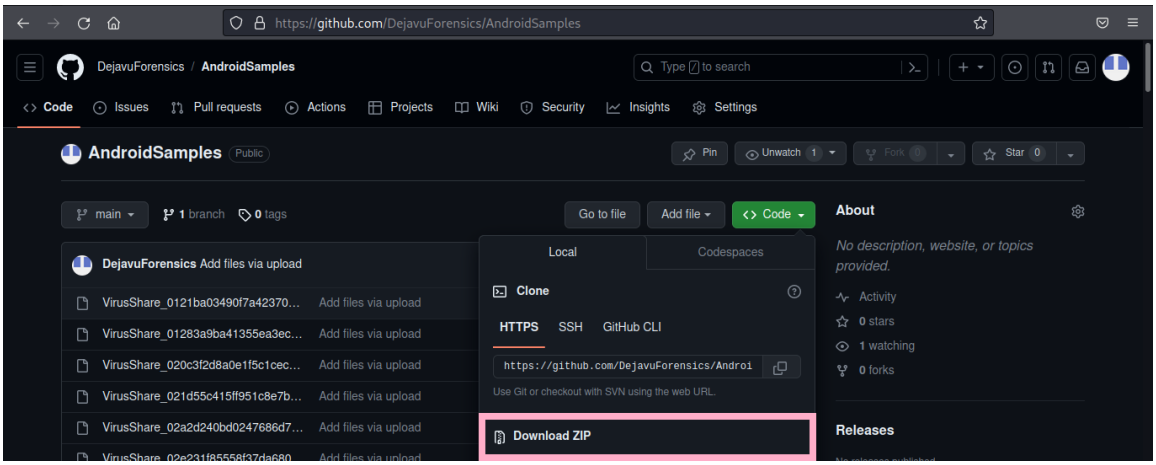


Figura 6.1: Repositório autoral contendo aplicativos maliciosos *Android*

.

3 Parâmetros da ferramenta *Androwarn*:

- -v: abreviatura de *verbose* de modo a imprimir na tela o progresso da perícia. Nível de verbosidade (Essencial 1, Avançado 2, Perito 3), o padrão é 1.
- -r: abreviatura de *report*. Relatório txt, html, json, tipo de relatório (padrão "html")
- -i: abreviatura de *input* (entrada). Arquivo APK para analisar.
- No console, use o *Androwarn*.

```
python androwarn.py -i /home/kali/Downloads/
AndroidSamples-master/VirusShare_0121ba03490f7a423708c8eb367e6fc4
-r html -v 3
```

4 Faça a auditória de outras amostras maliciosas *Android*. Verifique quais são as características extraídas através do *Androwarn*.

6.2.2 *Pescanner*: Extrator Estático de Características para Windows

De forma sintética, o *Pescanner* avalia os seguintes grupos de características para aplicativos Windows de computador de mesa.

- Características relacionadas à criptografia de dados. Tal estratégia é típica de *ransomware* os quais sequestram os dados da vítima através da criptografia. Para descriptografar os dados, o invasor pede ao usuário um montante monetário para que possa ter de volta todos os seus dados.
- Características relacionadas à coleta de medidas dos elementos gráficos em tela, utilizado por hackers em conjunto com técnicas maliciosas de ransomware.
- Características relacionadas a *spyware* como *keylogger* (captura de informações do teclado visando o furto de senhas e logins) e *screenlogger* (filmagem da tela da vítima). O antivírus criado visa monitorar se o arquivo suspeito tenta a atividade da Internet do usuário e informações particulares. Além disso, é investigado se o arquivo auditado tenta coletar senhas bancárias *on-line* e outras informações confidenciais e enviar os dados para seu criador.

- Características relacionadas a indícios de que o computador tenha sofrido fragmentação no seu disco rígido, além de tentativas de inicialização inválidas acumuladas.
- Características relacionadas ao Sistema Operacional. A forense digital averigua se são criados perfis e auditadas informações internas (eg.: *drivers*) do Sistema Operacional Windows.
- Características relacionadas à inicialização do Windows. Audita-se caso o arquivo suspeito tentar modificar configurações de *boot*. Também é auditado se houve instalação de um *bootkit* (arquivos maliciosos com a finalidade de alterar e infectar o MBR da partição) por meio de modificações no disco rígido.
- Características relacionadas ao Registro (Regedit) do Windows. Cabe ressaltar que a vítima pode não estar livre da infecção de um *malware* mesmo após a sua detecção e eliminação. A persistência das malfeitorias, mesmo após a exclusão do malware, ocorre devido à inserção de entradas (chaves) maliciosas no Regedit. Logo quando o sistema operacional é inicializado, o cyber-ataque recomeça devido à chave mal-intencionada invocar a vulnerabilidade explorada pelo *malware* (ex.: redirecionar a página inicial do Internet Explorer).
- Características relacionadas à Antiforense Digital as quais são técnicas de remoção, ocultação e subversão de evidências com o objetivo de reduzir as consequências dos resultados de análises forense. Então o antivírus criado investiga se o arquivo suspeito tenta suspender a sua própria execução até que um determinado intervalo de tempo limite tenha decorrido. Tal estratégia típica dos *malware* que ficam inativos até o término da quarentena dos antivírus comerciais;
- Características relacionadas à criação de GUI do programa suspeito. O antivírus criado audita se o arquivo suspeito tenta detectar formas através de visão computacional e processamento digital de imagem.
- Características relacionadas à perícia ilícita da memória principal (RAM) do sistema local. O antivírus criado investiga se o aplicativo suspeito tenta reservar, confirmar ou alterar o estado de uma região de páginas no espaço de endereço virtual de um processo.
- Características relacionadas ao tráfego de rede. Averigua-se se o arquivo testado tenta consultar servidores DNS e criar uma sessão FTP ou HTTP em tempo de execução.
- Características pertencentes a *sniffers*. O antivírus investiga se o aplicativo suspeito tenta ler dados dos pacotes de rede feitos a partir de requisições prévias do sistema local.
- Características típicas de *backdoors* quando a vítima passa a receber comandos (ordens) remotos. Em uma aplicação convencional, o servidor envia dados para o(s) cliente(s). De forma inversa, nos *malware*, a vítima envia os dados (imagens, dígitos) para o servidor. Logo os *malware* podem criar *sockets*, no sistema local, aguardando (*listen*) que um computador mal-intencionado remoto requisite uma conexão e, portanto, possa receber as informações íntimas da vítima;
- Características relacionadas a programas aplicativos utilitários. O antivírus criado verifica se o arquivo suspeito tenta alterar as configurações dos programas utilitários do Sistema Operacional Windows (ex.: reproduzir vídeos/áudios pelo Windows Media Player).

Siga as instruções:

1. Faça o *Download* da base de dados autoral contendo *ransomware* visando computadores de mesa. Clique no seguinte link https://github.com/DejavuForensics/Ransomware. Em seguida, faça o *Download* conforme mostra a Fig. 6.2. No seu computador, na pasta ***"Downloads/Ransomware-master"***, clique com o lado direito do mouse e escolha ***"Extract Here"***.

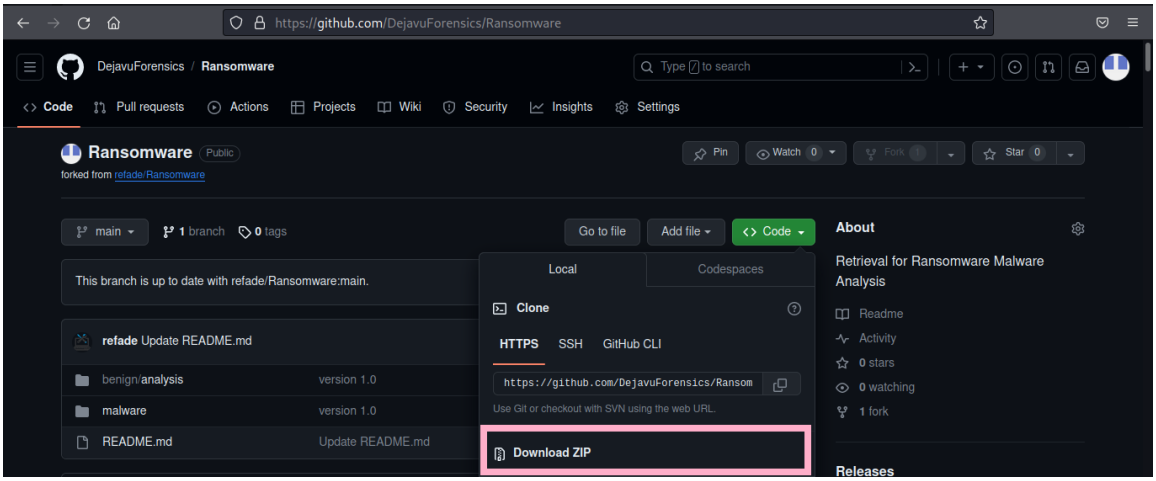


Figura 6.2: Repositório autoral contendo *ransomware* visando computadores de mesa.

2 Instalação e manuseio do *pescanner* de modo a fazer a análise de aplicativos windows de computador de mesa. Clique no seguinte link `https://github.com/serrastusbear/PE_Analyzer`. Em seguida, faça o *Download* conforme mostra a Fig. 6.3. No seu computador, na pasta ***"Downloads/ PE_Analyzer-master"***, clique com o lado direito do mouse e escolha ***"Extract Here"***.

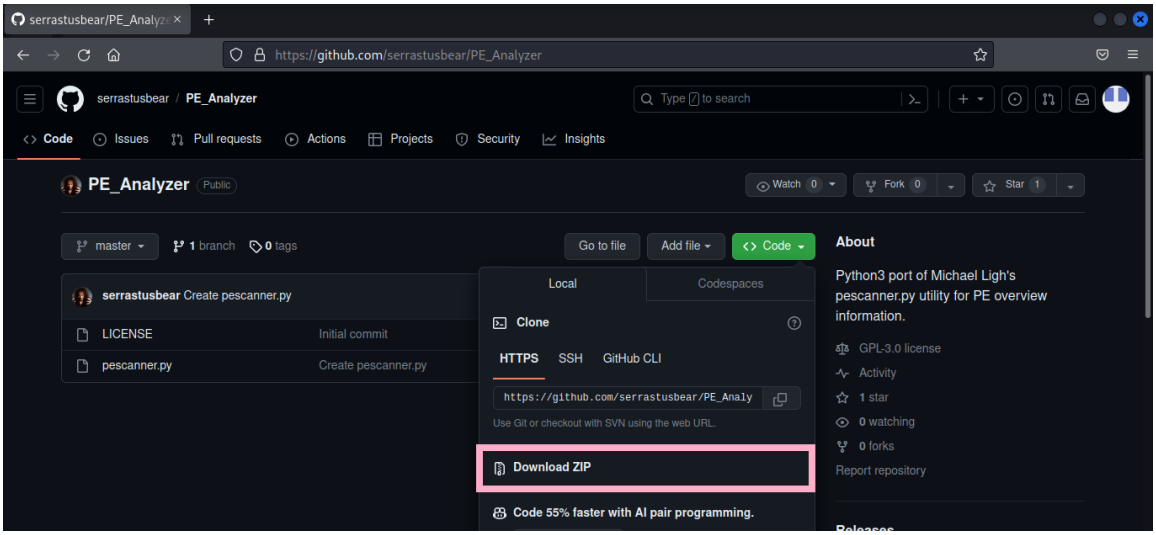


Figura 6.3: Repositório do *pescanner*.

3 Nas sequência, abra o console na pasta clonada *PE_Analyzer-master*, conforme mostra a Fig. 6.4.

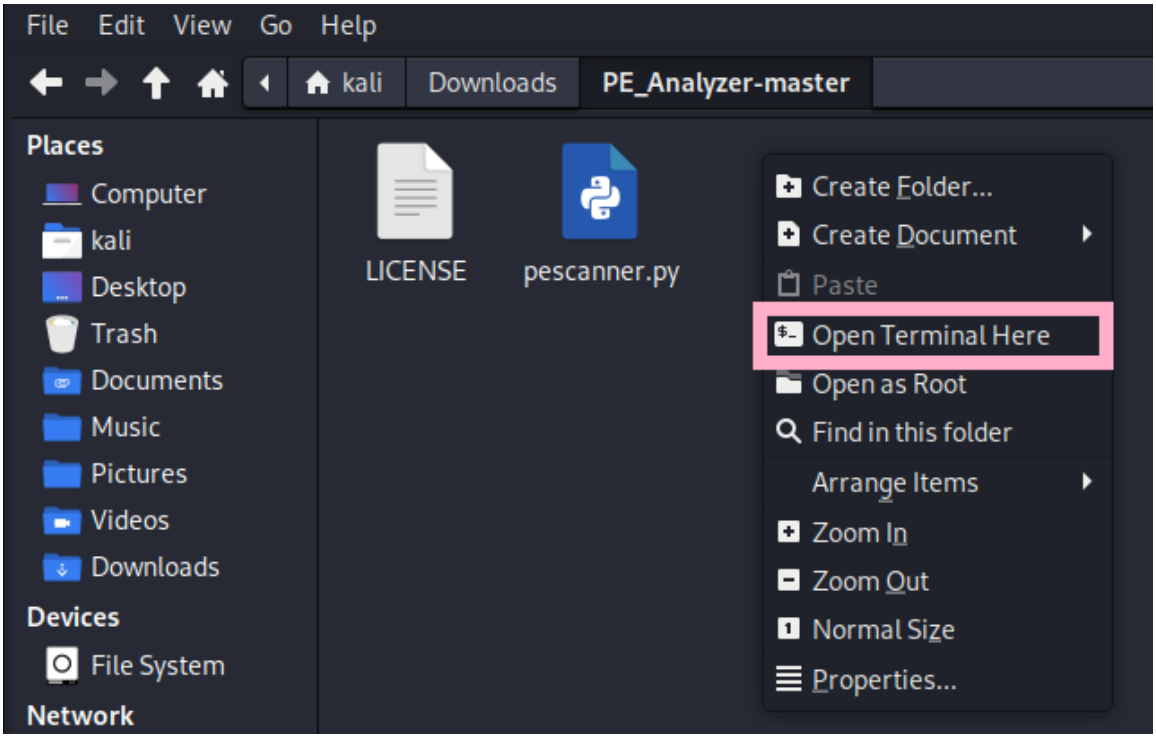


Figura 6.4: Abertura do console na pasta clonada anteriormente.

- No console, entre em modo de administrador. Por padrão, o *login* é ***kali***, a senha também é ***kali***.

```
sudo su
```

- No console, empregue o *pescanner* de modo a auditar os *ransomware*.

```
python pescanner.py /home/kali/Downloads/Ransomware-master/
malware/malware/VirusShare_000a3ea381d7d70be8b6fe1ee51dca22
```

- No console, é possível salvar a auditória do *pescanner* em arquivo de texto através do parâmetro >.

```
python pescanner.py /home/kali/Downloads/Ransomware-master/
malware/malware/VirusShare_000a3ea381d7d70be8b6fe1ee51dca22
> outputPescanner
```

4 Através do *pescanner*, é possível auditar o repertório de instruções, as APIs [2], as importações e exportações do aplicativo suspeito. A Fig. 6.5 exibe um auditória do *pescanner*. De forma destacada, há uma importação da função ***OpenProcess*** vinculada à biblioteca windows.h.

5 No portal da *Microsoft*, no seguinte *link* `https://learn.microsoft.com/`, é possível ter maiores esclarecimentos quanto à importação do aplicativo suspeito. Ao pesquisar a função ***OpenProcess***, há a sua completa descrição conforme ilustra a Fig. 6.6.

[2] API: *Application Programming Interface* - Interface de Programação de Aplicação

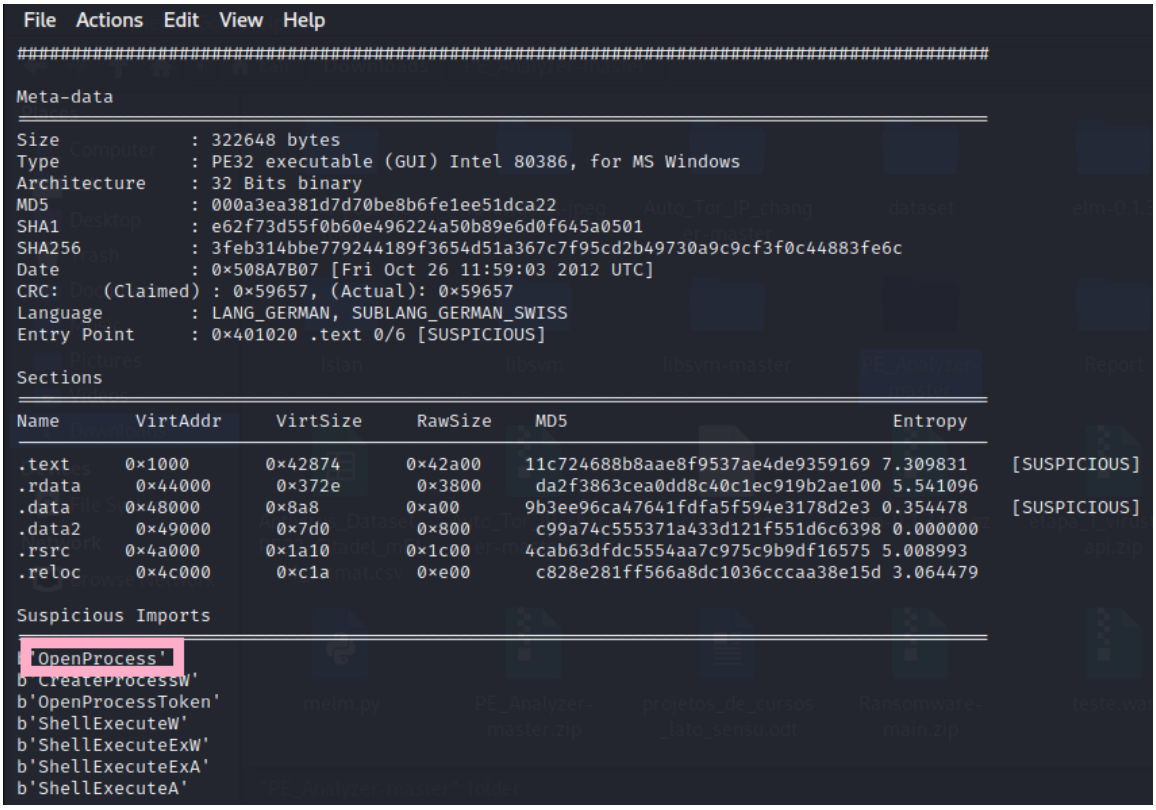


Figura 6.5: Repositório autoral contendo *ransomware* visando computadores de mesa.

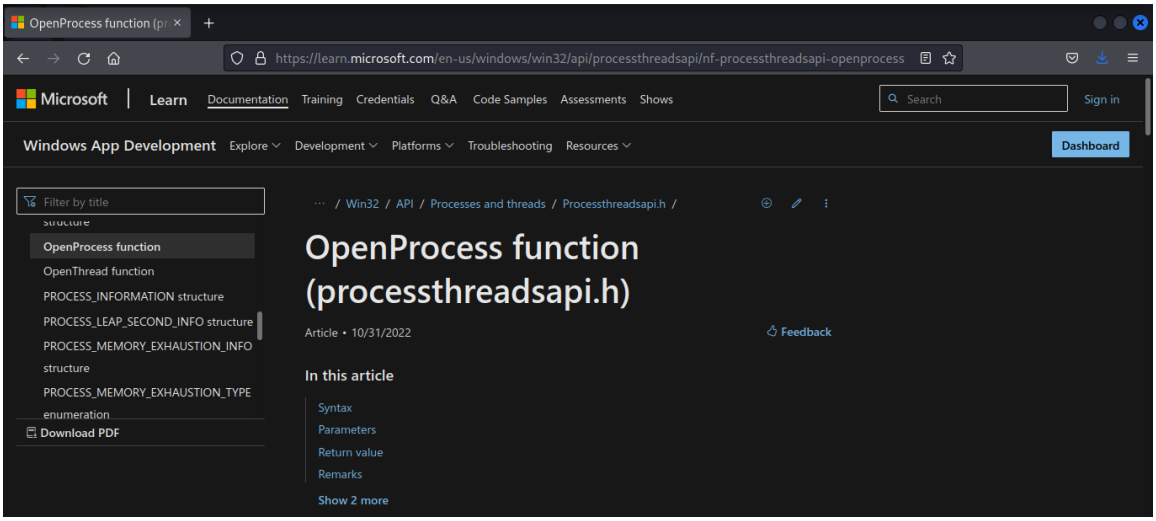


Figura 6.6: Descrição da função ***OpenProcess*** no portal da *Microsoft*.

6.3 Análise Dinâmica

A análise estática pode apresentar deficiências quando submetida a *malware* obfuscados. Como estratégia de antiforense digital, os *malware* empregam empacotamento e obfuscação de código. Logo as instruções empregadas no aplicativo original são diferentes daquelas da nova variante, embora os algoritmos sejam equivalentes. Conclui-se que a abordagem de características estáticas pode ser burlada por métodos de obfuscação [26].

A análise forense digital se desmembra em uma abordagem denominada extração dinâmica. Ao invés de análise estática, a extração dinâmica de características diz respeito ao comportamento do sistema quando o arquivo suspeito é propositalmente invocado em ambiente controlado. A principal vantagem da abordagem dinâmica é que a análise comportamental é capaz de detectar técnicas de mutação de baixo nível, como empacotamento ou obfuscação.

Além da capacidade de detectar obfuscação, a análise dinâmica também é capaz de detectar ataques "sem arquivos"[19]. Atualmente, ao invés de infecções convencionais, através de arquivos executáveis portáveis (PE[3] *files*), os *cyber*-ataques modernos empregam ataques "sem arquivos". Ataques sem arquivos são executados diretamente do servidor web malicioso para um serviço *listening* no computador pessoal (*endpoint*). Tecnicamente, ataques sem arquivos a partir de servidores web maliciosos são nomeados de ataques *server-side*. Em síntese, análise estática de características é inválida mediante ataque "sem arquivos"visto que não há como um computador pessoal periciar códigos-fontes armazenados e executados em um servidor web remoto [19].

Embora a análise estática apresente baixa complexidade e custo computacional, a análise dinâmica oferece informações em tempo de execução que não podem ser obtidas apenas com análise do código [30]. A análise estática pode ocorrer rapidamente e oferecer uma boa acurácia. Enquanto a análise de comportamento (dinâmica) pode ofertar um melhor entendimento das ações realizadas pelos softwares maliciosos.

6.3.1 *Cuckoo Sandbox*

O *Cuckoo box* é uma *sandbox* capaz de auditar o comportamento de arquivos suspeitos. O *malware* é executado visando infectar, propositalmente, o Sistema Operacional auditado, em tempo real (dinâmico), pela *Cuckoo Sandbox*. O *Cuckoo* pode auditar milhares de ações que o arquivo suspeito possa fazer.

A seguir, são detalhados os grupos de características referentes ao monitoramento, em ambiente controlado, pelo *Cuckoo Sandbox*.

- Características relacionadas ao tráfego de rede.
- Características relacionadas ao Registro (Regedit) do Windows 7.
  - Mudanças nas associações entre extensões de arquivos e softwares instalados na máquina.
  - Modificações nas informações sobre o usuário atual.
  - Corrupção do funcionamento dos drivers.
  - Alterações nas configurações de aparência do Windows e as configurações efetuadas pelos usuários, como papel de parede, protetor de tela e temas.
  - Mudanças nas Configurações de hardware.
- Características relacionadas a Backdoos. Programa que permite o retorno de um invasor a um computador comprometido, por meio da inclusão de serviços criados
- Características relacionadas a ameaças bancárias. *malware* visando obter acesso a informações confidenciais e/ou materiais armazenados ou processadas por meio de sistemas bancários

[3] Portable Executable

online.

- Características relacionadas a Bitcoin. Examina-se caso o arquivo testado tenta instalar a biblioteca OpenCL, ferramenta para minerar Bitcoins.
- Características relacionadas a Bots (máquinas que desempenham tarefas automáticas em rede, maliciosas ou não, sem o conhecimento de seus proprietários).
- Características relacionadas a navegadores. É verificado se o arquivo testado tenta:
  - instalar um objeto Browser Helper (geralmente um arquivo DLL que adiciona novas funções ao navegador) com a finalidade de deixar a experiência de navegação prejudicada de alguma forma;
  - modificar as configurações de segurança do navegador;
  - modificar a página inicial do navegador;
  - adquirir informações privadas de navegadores de internet instalados localmente.
- Características relacionadas à computação em nuvem. O arquivo é auditado quando tenta se conectar aos serviços de armazenamento e/ou arquivos do Dropbox, Google, MediaFire, MegaUpload, RapidShare, Cloudflare e Wetransfer.
- Características relacionadas a ataques de DDoS [4].
- Características relacionadas a Ransomware; tipo de *malware* que por meio de criptografia, deixa inutilizados os arquivos da vítima, para depois solicitar um resgate em troca da normal utilização posterior dos arquivos do usuário, resgate este geralmente pago de forma não rastreável, como bitcoins.
- Características relacionadas aos arquivos executáveis. A forense digital proposta verifica se o arquivo suspeito tenta:
  - utilizar a ferramenta BITSAdmin (ferramenta de linha de comando originalmente utilizada para fazer download e upload de arquivos, assim como acompanhar o progresso desta transferência, mas que hackers utilizam de forma maliciosa) para baixar um arquivo qualquer;
  - travar, ao menos, um processo durante a sua execução;
  - executar a instrução WaitFor (executável presente no Windows desde sua versão 7, originalmente tem a função de sincronizar eventos entre computadores em rede, mas que malfeitores usam de maneiras prejudiciais), possivelmente para sincronizar atividades maliciosas.
- Características relacionadas a *exploits* os quais se constituem como *malware* que tentam se utilizar de vulnerabilidades, falhas ou defeitos conhecidos e ainda não sanados do sistema ou de um ou mais de seus componentes, a fim de causar instabilidades e comportamentos imprevistos tanto em seu hardware como em seu software.
- Características relacionadas a *Infostealers*, programas maliciosos que coletam informações confidenciais do computador afetado.

Além da detecção de comportamentos suspeitos, como chamadas a APIs, a análise dinâmica também permite a reconstituição (limpeza) do SO (Sistema Operacional) através da auditoria das malfeitorias promovidas pelo arquivo malicioso partindo do princípio que o malefício não é irreversível. Cabe ressaltar que a reconstituição do SO, tecnicamente nomeada de vacina, é importante porque não basta apenas a detecção e eliminação do *malware* para que a vítima esteja livre de sua atuação. Além da eliminação do malware, é necessário desfazer todas as suas malfeitorias como, por exemplo, ter desabilitado *Java Security Manager*. Então caso não houvesse a auditoria,

[4] *Distributed Denial of Service* – Ataque Distribuído de Negação de Serviço.

provida pela análise dinâmica, caberia ao cyber-vigilante monitorar, manualmente, qualquer mudança no SO o que tornaria o processo moroso e estressante.

Siga as instruções:

1. Clique no seguinte link `https://cuckoo.cert.ee/` de modo a abrir *on-line* o *Cuckoo Sandbox* conforme mostra a Fig. 6.7.
   - Clique no botão ***"Submit a File For Analysis"***.

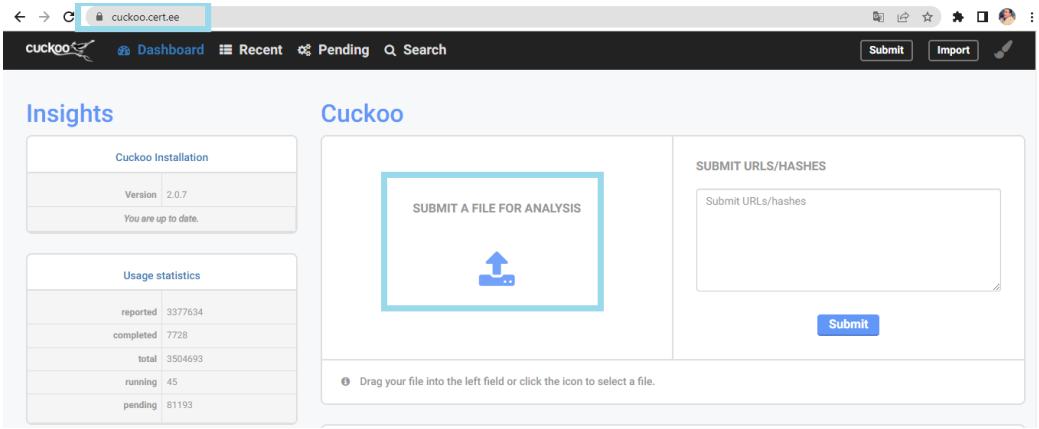


Figura 6.7: *Cuckoo Sandbox* disponível *on-line* em `https://cuckoo.cert.ee/`.

2 O *Cuckoo Sandbox* provê a possibilidade de escolha entre Linux e Windows como ambiente operacional a ser infectado propositalmente. A Fig 6.8 ilustra o campo onde há essa possibilidade de escolha do ambiente operacional.

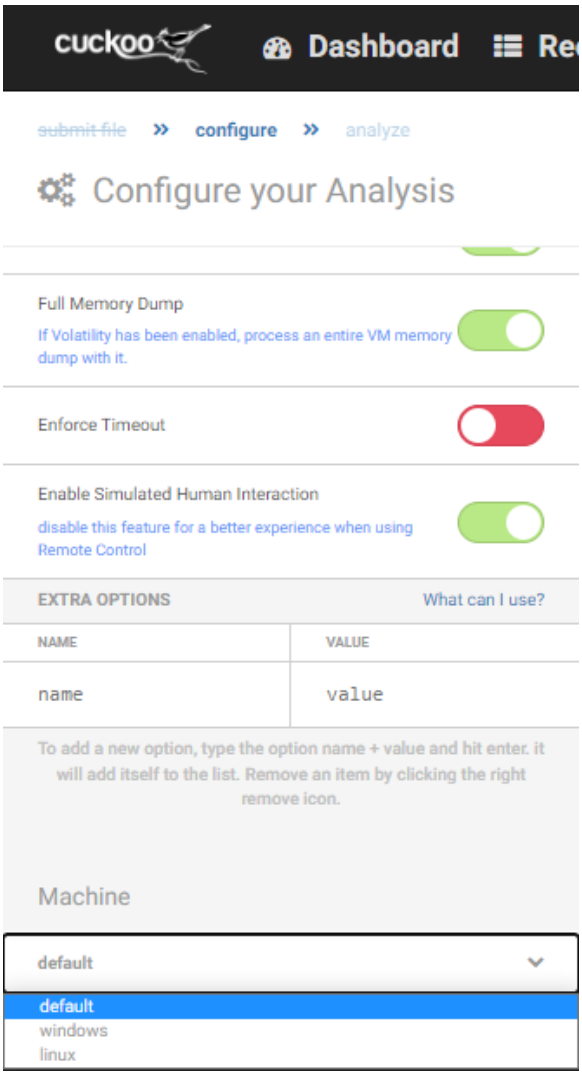


Figura 6.8: Possibilidade de escolha entre Linux e Windows como ambiente operacional a ser infectado propositalmente.

3 Clique no botão ***Analyze***, conforme mostra a 6.9. O objetivo é começar a análise dinâmica do arquivo suspeito.

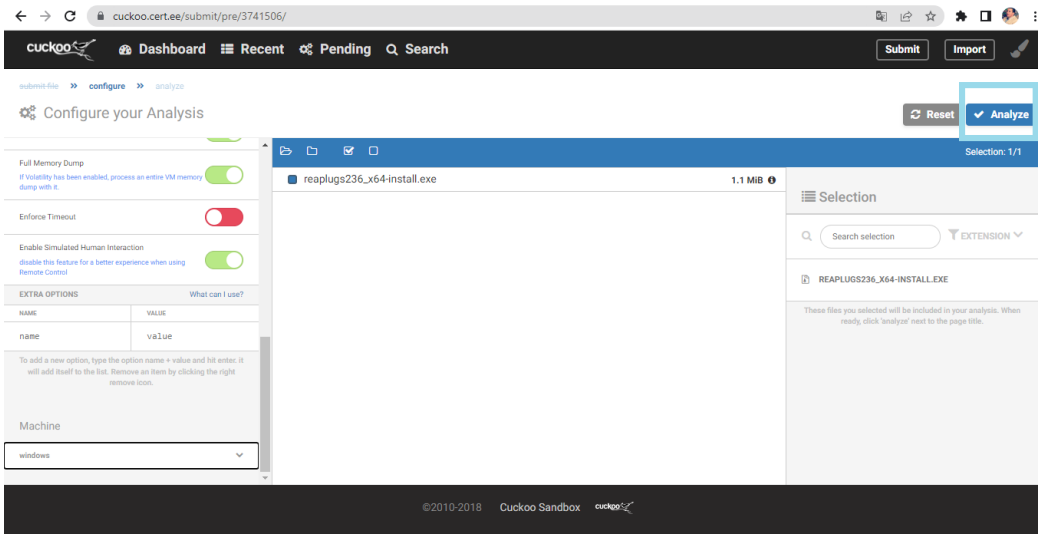


Figura 6.9: Análise do arquivo suspeito em ambiente controlado: *Cuckoo Sandbox*, disponível em https://cuckoo.cert.ee/analysis/3490988/summary.

4 A análise pode custar tempo a depender do tráfego do servidor. De modo a concluir o experimento em tempo de aula, clique no seguinte link https://cuckoo.cert.ee/analysis/. Escolha alguma amoostra previamente submetida por terceiros, conforme mostra a Fig. 6.10.

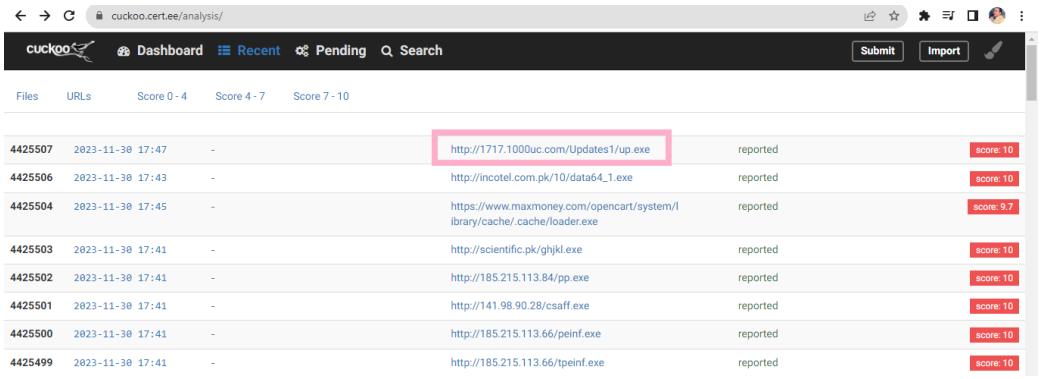


Figura 6.10: Análises recentes, feita por terceiros, em ambiente controlado: *Cuckoo Sandbox*, disponível em https://cuckoo.cert.ee/analysis/.

5 Em regra geral, a fila de espera é grande. Pode-se custar vários minutos a depender do tráfego do servidor no momento. Ao final da auditoria, é possível ter acesso ao relatório completo do *Cuckoo Sandbox*. No menu ***"Export Analysis"***, é possível ter acesso ao relatório completo da análise dinâmica, como ilustra a Fig. 7.4.

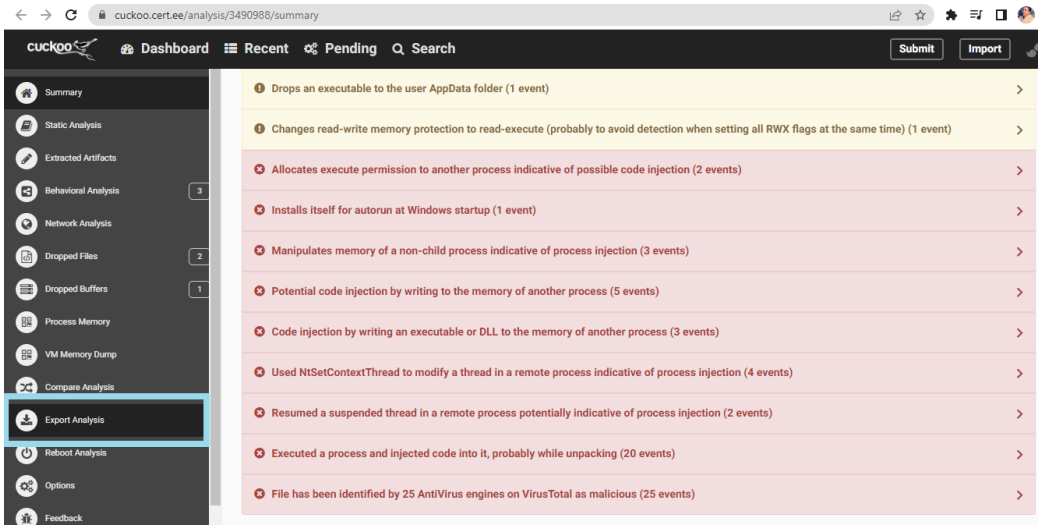


Figura 6.11: Exportar/Baixar a análise dinâmica do *Cuckoo Sandbox*.

5 O relatório completo da análise dinâmica pode ser lido no arquivo *reports.json* dentro da pasta *reports*. Ao todo, a extração dinâmica de características monitora milhares de comportamentos que o arquivo suspeito possa fazer quando executado.

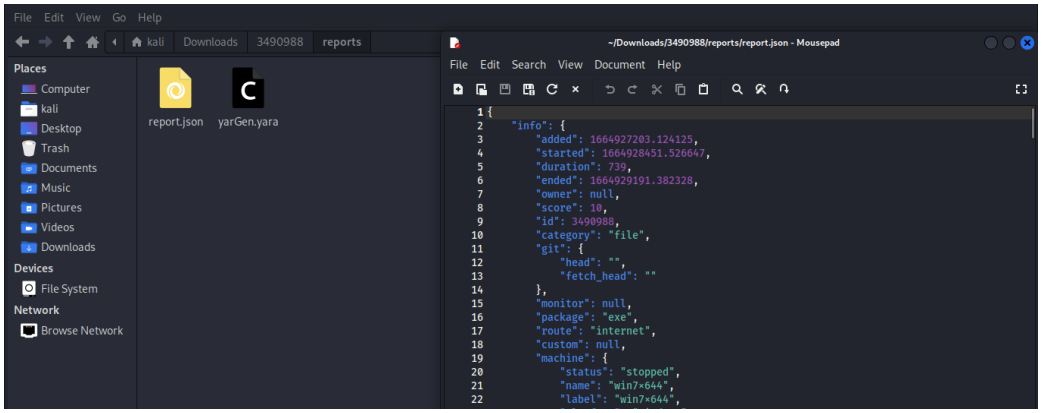


Figura 6.12: Relatório da análise dinâmica do *Cuckoo Sandbox*.

6.4 Discussão

Explore alternativas de ferramentas para extrair características de aplicativos suspeitos. No âmbito da análise estática, é fundamental ter um extrator específico para cada tipo de aplicativo. Por exemplo, um extrator destinado a aplicativos Windows não é adequado para a plataforma Android. Os vídeos mencionados nos QR-Codes da Fig. 6.13 estão configurados como privados. Para acessá-los, entre em contato com os moderadores por meio do YouTube ou do canal no Telegram.

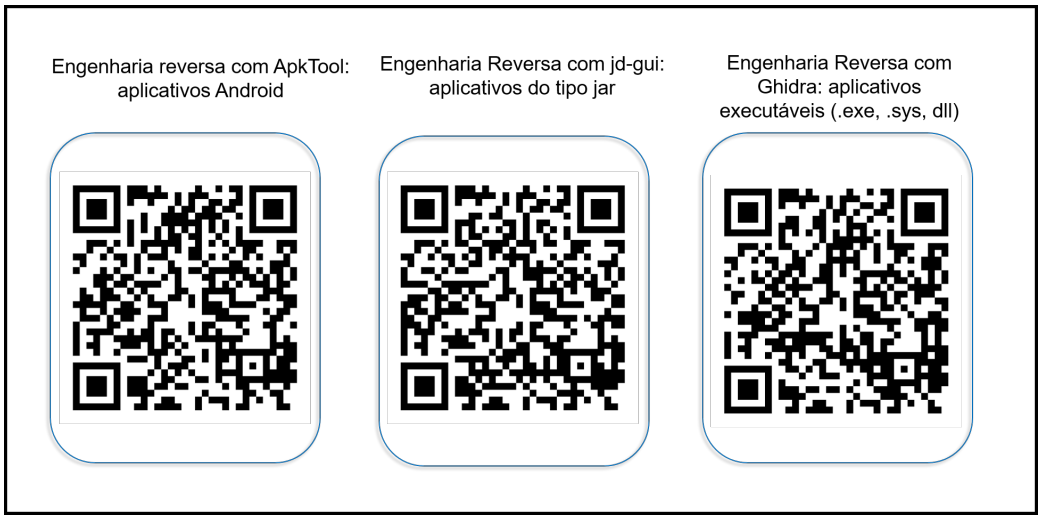
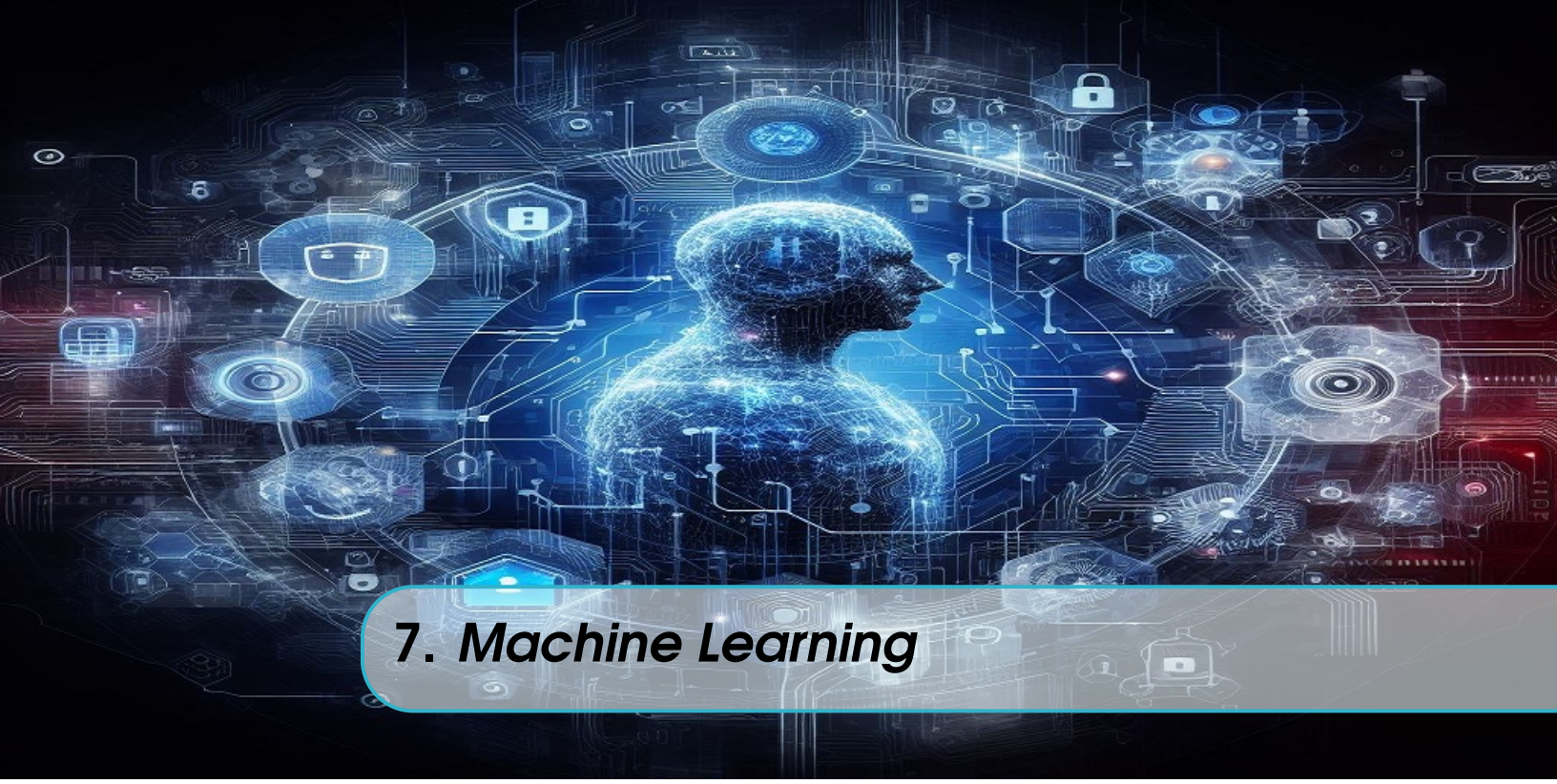


Figura 6.13: QR codes atrelados a vídeos tutoriais de análises estáticas visando diversos aplicativos.



7. Machine Learning

7.1 Classificador SVM: Reconhecimento de Padrão

No presente experimento prático, será empregado o SVM (*Support Vector Machine* - Máquina de Vetores de Suporte) visando o reconhecimento de padrão de amostras entre classe *vs* contra-classe. O SVM é uma máquina de aprendizado estatístico que não se inspira necessariamente no funcionamento do cérebro humano. Seu objetivo explícito é a teoria do aprendizado estatístico. As redes neurais clássicas visam encontrar um hiperplano de modo a separar as classes pertencentes à aplicação alvo. Podem existir vários hiperplanos separando os dados corretamente. Ao contrário de redes clássicas, a SVM visa encontrar um hiperplano melhor do que os demais.

A Fig. 7.1 exibe a diferença de atuação entre a rede MLP [1] e a rede SVM. Pode-se notar que o hiperplano com maior margem de separação tem melhor capacidade de generalização pois diminui a possibilidade de erro. Considere que x_1 e x_2 são os neurônios de entrada. Trata-se de um repositório meramente didático. A SVM é uma máquina de aprendizado estatístico amplamente utilizada em aplicações computacionais.

Siga as instruções:

1. Instalação da biblioteca *libsvm* de modo a empregar a SVM em tarefas de classificação (categorização).
   - No console, instale o *libsvm*.

```
pip install libsvm
```

[1] MLP: *Multi-Layer Perceptron* - Perceptron Multi-Camada.

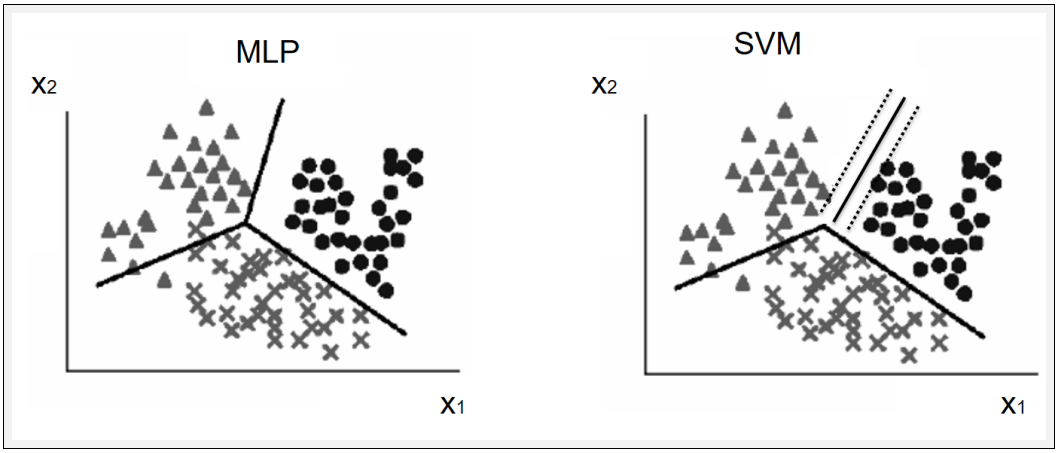


Figura 7.1: Diferença de atuação entre a rede MLP e a rede SVM.

2 Conforme dito anteriormente, não é do escopo do Capítulo a criação da base de dados. Parte-se do princípio que o repositório de aprendizado já foi previamente confeccionado por terceiros. A criação de repositório e as metodologias visando extrair características dos *malware* foram discutidas no capítulo 6. Na Máquina Virtual, acesse o seguinte link `https://github.com/cjlin1/libsvm/blob/master/heart_scale`. Na sequência, clique em **"Raw"** conforme ilustra a Fig. 7.2.

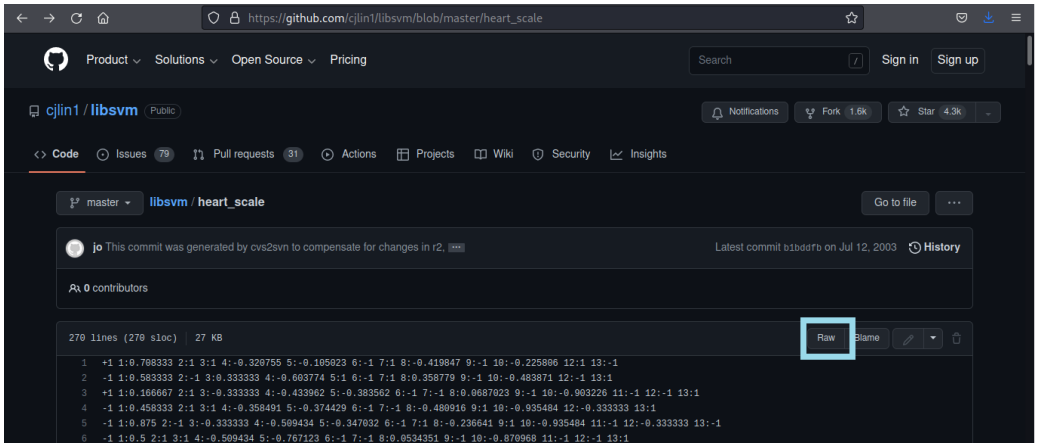


Figura 7.2: Base de dados *Heart Scale*, um exemplo de repositório de classificação (categorização). Clique em **"Raw"**.

3 Faz-se necessário fazer o *download* do repositório. Clique com o lado direito do mouse, na sequência escolha a opção ***"Save Page As..."*** como demonstra a 7.3.

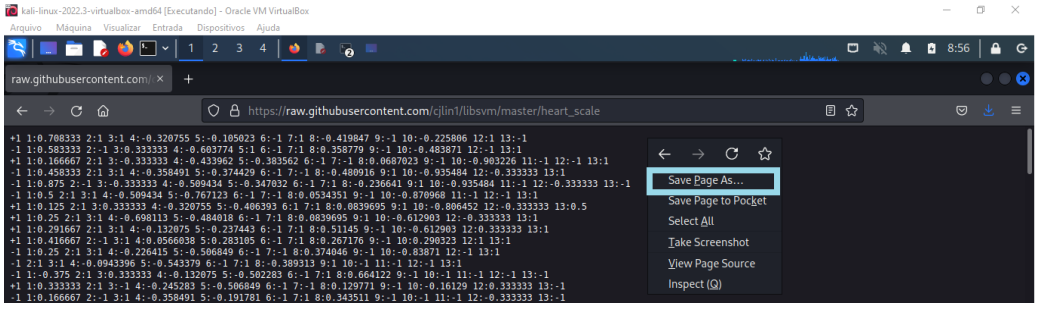


Figura 7.3: Clique com o lado direito do mouse, na sequência escolha a opção ***"Save Page As..."***.

4 Após copiar a base de dados para o Editor de Texto, é possível notar a estrutura do repositório conforme exibe a Fig. 7.4.

- **Primeira coluna**: +1; a amostra (linha) pertence à classe. -1; a amostra (linha) pertence à contra-classe.
- **Demais colunas**: atributos (neurônios) de entrada referentes à extração de características da aplicação alvo. Na presente base, há 13 neurônios de entrada.
- ***Index*: Valor**: cada valor do neurônio de entrada é precedido, por :, pelo seu respectivo *index* . Por exemplo, na primeira amostra (linha), o primeiro neurônio tem valor 0.708333.

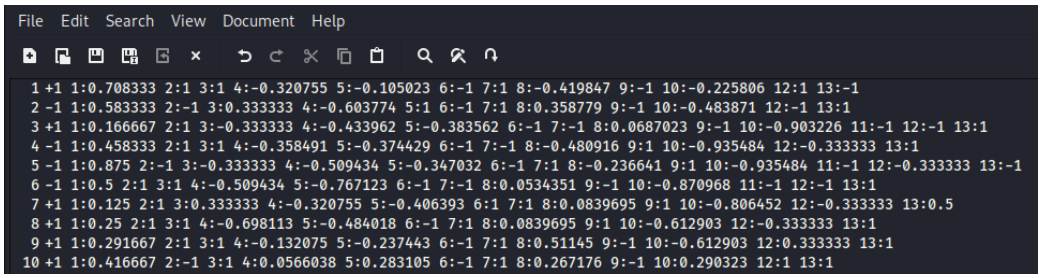


Figura 7.4: Estrutura de um repositório visando o uso da SVM como classificador.

5 Cabe atentar que a base de dados possui dados pendentes como exibe a Fig 7.5. Na amostra (linha) 243 não há o primeiro neurônio. A SVM não apresenta mecanismos para tratamento de dados pendentes. Eles, portanto, são preenchidos com zeros. O tratamento de base de dados incompleta é um dos grandes desafios da área de inteligência artificial. A inferência sobre os dados pendentes não é trivial.

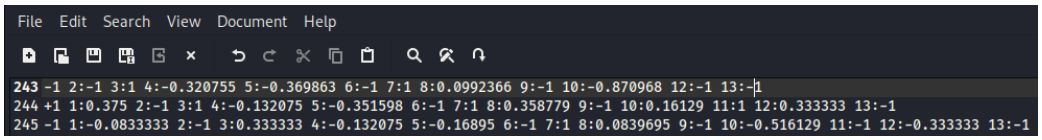


Figura 7.5: Exemplo de dados pendentes.

6 Use o seguinte *script* em *python*. De modo a agilizar o experimento, salve tanto o *script* quanto a base de dados *heart_scale* em uma mesma pasta. Quanto ao *script*:

- Linha 1: a biblioteca *libsvm* é invocada.
- Linha 2: a base de dados *heart_scale* é lida.
- Linha 3: o treinamento (aprendizado) ocorre contendo as duzentas primeiras amostras da base de dados. O referido fato é constatado através do index : 200. O operador : significa o catálogo de tudo antes do 200, ou seja, de 0 a 199 (O zero é o primeiro elemento em um vetor).
- Linha 4: a fase de teste da rede pós-treinada ocorre na linha 4 contendo as amostras de 200 ao final da base de dados. Como resultado, há a acurácia de 84.2857%. Enfatiza-se que não se deve treinar e testar um classificador como o mesmo conjunto de dados. Por isso há uma divisão entre conjunto de treino e conjunto de teste.

```
1 from libsvm.svmutil import *
2 y, x = svm_read_problem('heart_scale')
3 m = svm_train(y[:200], x[:200], '-c 4')
4 p_label, p_acc, p_val = svm_predict(y[200:], x[200:], m)
```

7.2 Parâmetros do Classificador SVM

Um dos grandes desafios, em máquinas de aprendizado estatístico, diz respeito a encontrar um *kernel* de modo que otimize a fronteira de decisão entre as classes de uma dada aplicação. Em redes neurais ELM, um *kernel* Linear, por exemplo, é capaz de resolver um problema linearmente separável, como o visto na 7.6 (a). Seguindo o mesmo raciocínio, *kernels* Sigmóide, RBF e Senoide são capazes de resolver problemas separáveis por função Sigmoidal, Radial e Senoidal, vistos na Fig. 7.6 (b), na Fig. 7.6 (c) e na Fig. 7.6 (d), respectivamente.

Então, uma boa capacidade de generalização da rede neural pode depender de uma escolha ajustada do *kernel*. O melhor *kernel* pode estar subordinado ao problema a ser resolvido. Como efeito colateral, a investigação de diferentes *kernels* é geralmente um processo custoso envolvendo validação cruzada combinada com diferentes condições iniciais aleatórias. A investigação de distintos *kernels*, no entanto, pode ser necessária, caso contrário a rede neural composta, por um *kernel* desajustado, por gerar resultados não satisfatórios. Como contra-exemplo, observe o emprego do *kernel* Linear aplicado a distribuições Sigmóide e Senoide apresentados na Fig. 7.7 (a) e na Fig. 7.7 (b), respectivamente. As precisões das classificações expostas na Fig. 7.7 (a) e na Fig. 7.7 (b) são de 78,71% e 73,00%, respectivamente. Visualmente, é possível observar que o *kernel* Linear não mapeia as fronteiras de decisões das distribuições Sigmóide e Senoide de forma adequada.

Uma boa capacidade de generalização desses *kernels* também depende de uma escolha ajustada de parâmetros (C, γ). O parâmetro de custo C se refere a um ponto de equilíbrio razoável entre a largura da margem do hiperplano e a minimização do erro de classificação em relação ao conjunto de treinamento. O parâmetro do *kernel* γ controla o limite de decisão em função das classes [26]. Não existe um método universal no sentido de escolher os parâmetros (C, γ). No presente trabalho, os parâmetros C e γ variam exponencialmente em sequências crescentes, matematicamente de acordo com a função 10^n, onde $n = \{-3, -2, -1, 0, 1, 2, 3\}$. A hipótese é verificar se esses parâmetros distintos dos padrões; $(C, \gamma) = (2^1, 2^1)$, são capazes de gerar melhores acurácias.

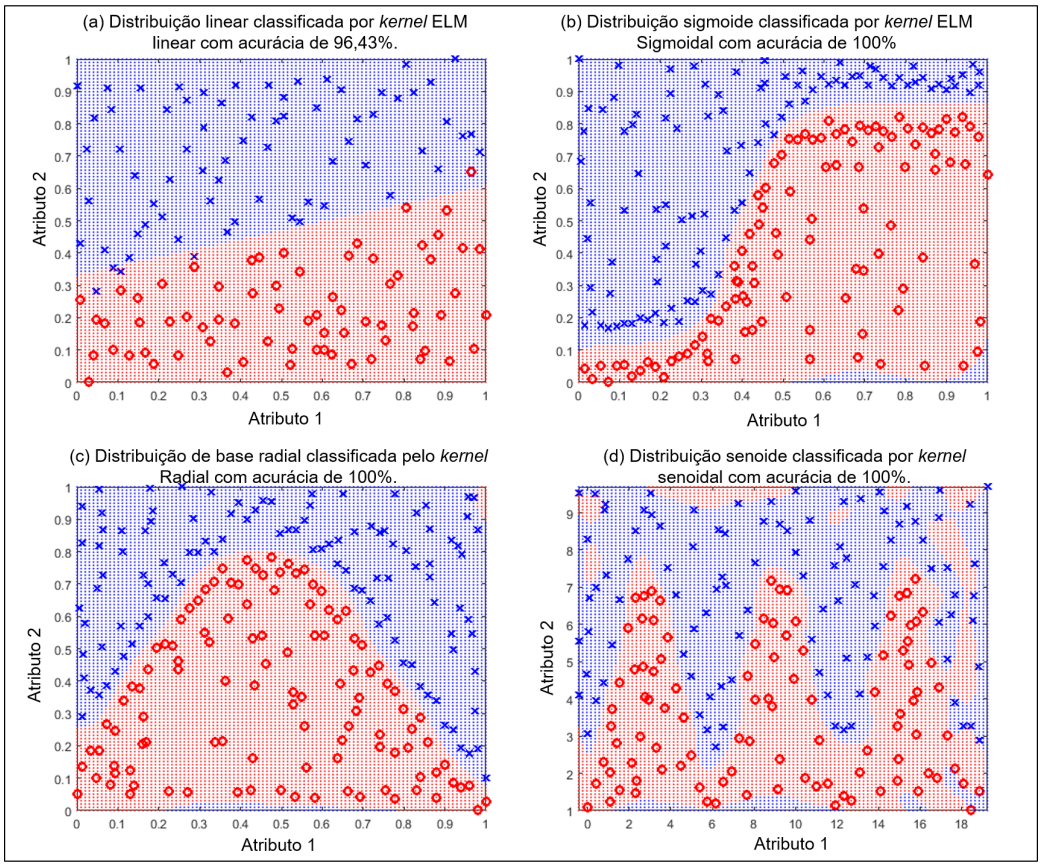


Figura 7.6: Atuações bem-sucedidas dos *kernels* compatíveis com os conjuntos de dados.

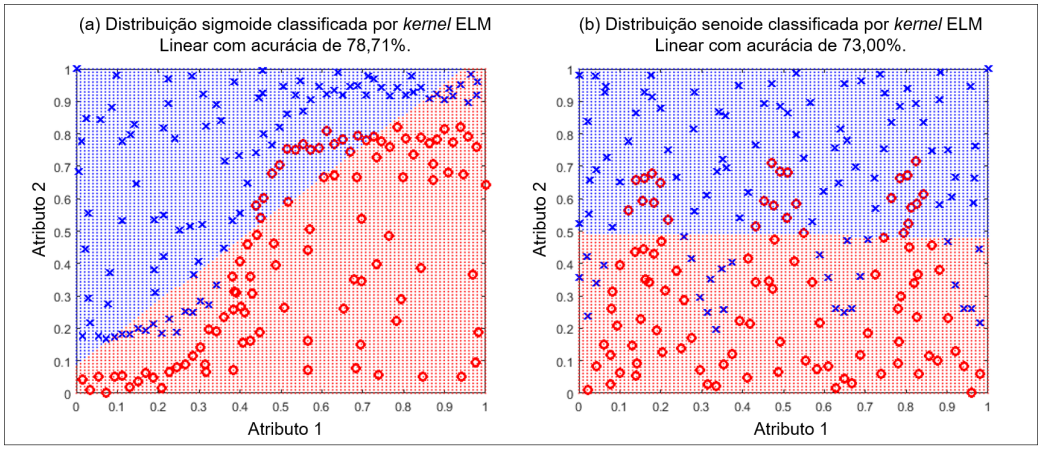


Figura 7.7: Atuações malsucedidas do *kernel* Linear em conjuntos de dados não-linearmente separáveis.

Siga as instruções:

1. Faça o *Download* do *script libsvm_parameters.py* responsável pelos cliques automatizados no presente link (pasta Cap. 6).

2. No console, para otimizar os parâmetros do classificador SVM.

```
python libsvm_parameters.py
```

7.3 Discussão: Acurácia *vs* Precisão

Acurácia diz respeito à capacidade do classificador detectar amostras tanto da classe quanto da contra-classe. Acurácia é diferente de precisão. Precisão despreza a contra-classe. Por exemplo, a precisão assume papel fundamental na área de engenharia biomédica. A meta é detectar todos os pacientes com anomalia genética (ex. câncer de mama), mesmo que, eventualmente, alguma paciente sadia seja erroneamente diagnosticada como enferma. As vantagens são enormes visto que a detecção precoce do câncer, em exames de imagens, é essencial de modo a se aumentar as chances de recuperação da paciente [19]. Para cada detecção de paciente com câncer, em exame de imagem, são necessários milhares de exames em pacientes sadios. Nesse contexto, aumentar a precisão em detrimento da acurácia é essencial de modo a não deixar de se identificar os pacientes com câncer.

Em países subdesenvolvidos e emergentes, aumentar a precisão em detrimento da acurácia é fundamental. Há uma dificuldade, por grande parte da população, em se ter acesso a serviços públicos de saúde tais como hospitais e clínicas de exames laboratoriais [16]. Ainda como agravante, há uma péssima distribuição de renda [1]. Então uma grande parcela da sociedade não pode custear seus atendimentos em clínicas e hospitais particulares. Também como fator complicador, a maioria dos habitantes reside em bolsões de pobreza localizados nas periferias dos grandes centros urbanos dos países emergentes [10]. O referido fato implica que grande parte da população não tem residência fixa [24][27]. Consequentemente, torna-se impraticável o agendamento de uma nova consulta ou exames de acompanhamento. A péssima distribuição de renda, os bolsões de pobreza e a sobrecarga dos serviços públicos de saúde são problemas recorrentes das nações em desenvolvimento [8][11]. Pode ser uma oportunidade única o exame de imagem biomédica em uma paciente de um país emergente. Como agravante, uma parcela significativa dos atendimentos realizados, nos serviços públicos de saúde, ocorrem nos estágios terminais das doenças [12]. O referido fato propicia que uma proporção significativa de óbitos seja relatado como causas mal-definidas visto que não há tempo hábil de investigação e certificação de causas da morte [12].

Em síntese, prestigiar a precisão, em detrimento da acurácia, é fundamental em aplicações biomédicas quando empregadas em países emergentes. Nesse cenário, não se pode deixar de detectar a paciente com câncer mesmo que isso ocasione em falsos positivos; pacientes sadias erroneamente classificadas como enfermas. Em engenharia biomédica, a desvantagem de aumentar a precisão em detrimento da acurácia é aceitável. A paciente teria um trauma psicológico temporário enquanto não for realizado o exame de biópsia (pulsão) visando a confirmação ou refutação do diagnóstico prévio. Recomenda-se o diagnóstico positivo, através de exames de imagem, não ser divulgado à paciente por se tratar de um resultado imaturo pendente de confirmação.

Na área jurídica, isoladamente a precisão não tem cabimento. Uma máquina de aprendizado estatístico visaria condenar todos os culpados independentemente de também condenar os inocentes.

Na área jurídica, a acurácia é fundamental. Deve-se prover suporte à decisão de modo que haja a condenação dos culpados (classe alvo) e absolvição dos inocentes (contra-classe). Em termos estatísticos, a máquina de aprendizado estatístico deve ser acurada e não somente precisa, em âmbitos jurídicos.

Paradoxalmente, estabelecer que um antivírus tenha alta precisão não é satisfatório. O antivírus deve possuir alta acurácia. Em termos didáticos, deve-se detectar os aplicativos maliciosos e também absorver os aplicativos sérios. Caso contrário, a interação com o mecanismo de *cyber*-vigilância pode se tornar inviável. Há ferramentas de *cyber*-vigilância, tal como o Snort [2], que levantam tantos falsos positivos a ponto da própria desenvolvedora pedir a colaboração dos clientes em seu próprio site oficial. Há uma espécie de portal-denúncia de modo que o cliente pode anunciar uma aplicação/pacote, falsamente, condenada. Em termos estatísticos, um mecanismo de *cyber*-segurança deve ser acurado e não somente preciso.

7.4 Discussão: Redes Rasas *vs* Redes Profundas

As redes neurais rasas, de baixa complexidade computacional, são aplicadas nas mais diversas áreas. As redes rasas, por exemplo, são aplicadas largamente no campo da engenharia biomédica [2][3][4][17][20][25]. Como alternativa, o aprendizado profundo (*Deep Learning*) se estabeleceu como o estado-da-arte de modo a suprir as dependências das redes rasas. As redes neurais profundas são frequentemente aplicadas a reconhecimento de padrões em imagens digitais. Mas devido às suas excelentes acurácias, o aprendizado profundo tem sido aplicado em uma variedade de tarefas.

Como vantagem, as redes profundas não empregam extratores de características. Nada do que foi discutido no Capítulo 6 é necessário quando se emprega redes profundas. Pois elas operam diretamente a partir dos dados brutos. As redes profundas possibilitam ignorar a etapa de estruturação dos dados. Os aplicativos são revertidos em imagens de modo que não é necessário haver qualquer estudo prévio sobre o problema. O aprendizado profundo possibilita criar uma inteligência artificial sem a mínima noção da literatura acadêmica quanto ao tema. As descrições clássicas de aplicações maliciosas se tornam dispensáveis visto que não há etapa de extração de características quando se emprega redes profundas.

Por operar diretamente sobre dados brutos, as redes profundas assumiram protagonismo no momento da pandemia devido à Covid-19. Havia escassez de literatura acadêmica quanto ao tema. Portanto não se sabia como viria a ser um extrator de característica em um exame de imagem de pulmões infectado pela Covid-19. Mas as redes profundas alcançaram acurácias superiores a mais de 98% na detecção da Covid-19 em imagens biomédicas [6][9]. Isso quer dizer, as redes profundas conseguem diferenciar pacientes com Covid-19 de exames de pessoas sadias em mais de 98% das imagens de pulmões. Enfatiza-se que ainda não há um consenso de quais são as características de um pulmão infectado pela Covid-19. Como vantagem, a dispensa de extratores de características possibilita a abrangência de distintos perfis de aplicações. Por não se basear em extratores pré-concebidos, o aprendizado profundo pode modelar qualquer perfil de dados brutos.

A desvantagem da rede neural profunda é o longo tempo de treinamento. Mesmo em um supercomputador, um simples treinamento de rede profunda pode custar meses, talvez anos. Por supercomputador, denota-se servidores com dezenas de núcleos de processamento e larga escala de memória volátil (memória RAM). As técnicas de redes neurais de última geração passaram a exigir grandes *datacenters* em detrimento de computadores comuns de mesa. As técnicas de aprendizado

[2] Snort: Ferramenta de detecção de intrusão em rede. Disponível em: `https://www.snort.org/`. Acesso em fev. de 2023.

profundo de última geração dependem de milhões de parâmetros ajustáveis (treináveis). Por exemplo, a rede profunda *Xception* tem por volta de 22 milhões de parâmetros ajustáveis distribuídos em 71 camadas escondidas sequenciais [7]. Por outro lado as redes rasas, convencionalmente, têm uma única camada escondida, por isso o treinamento é rápido.

Como agravante, as redes profundas possuem baixa capacidade de paralelismo porque as suas camadas são sequenciais. Redes profundas são modelos de processamento de dados que utilizam grafos profundos em cascata. Convencionalmente, uma arquitetura de rede neural é construída contendo dezenas de camadas escondidas sequenciais. Cada camada escondida utiliza como dados de entrada o resultado do processamento da camada escondida imediatamente anterior a ela. Se todas as camadas fossem executadas simultaneamente, o paradigma produtor-consumidor ocorreria. A leitura ocorreria pela camada consumidora enquanto os dados ainda estariam sendo processados pela camada produtora. De tal modo que a camada consumidora poderia acessar dados prematuros e incorretos [23].

Como dito anteriormente, o aprendizado profundo de última geração tem dezenas de milhões de parâmetros ajustáveis [7]. Mesmo se um supercomputador hipotético tivesse milhões de processadores, as redes profundas não seriam capazes de otimizar seus tempos de treinamento. O paradigma produtor-consumidor impede que todas as camadas funcionem simultaneamente. Uma camada só pode ser executada depois que a camada superior tiver concluído seu trabalho. O modelo sequencial em cascata de redes profundas apresenta um grande desafio em relação à área de computação de dados em paralelo.

Há modelos de redes profundas dotadas de processamento de dados simultâneos, mas ainda se encontram em estágio inicial e não apresentam boas acurácias em distintas aplicações [26][28]. A rede profunda paralelizável, proposta por SANTOS, *et al.* 2019, contém 30 (trinta) mil filtros convolucionais acionados de uma só vez [28]. Desde que esteja em um supercomputador, a rede profunda de SANTOS, *et al.* 2019 apresenta tempo de treinamento não discrepante das redes rasas. A referida rede profunda alcança excelentes resultados quando aplicada a processamento de imagem, em específico, reconhecimento óptico de caracteres [28]. Mas a rede profunda paralelizável de SANTOS, *et al.* 2019 não obteve sucesso quando aplicada a reconhecimento de padrão de *malware*. Seus resultados foram estatisticamente inferiores tanto a redes rasas quanto a redes profundas baseadas em camadas sequenciais (grafos em cascata) [26].

Em tempos contemporâneos, é fundamental aliar taxas de acertos elevadas e rápido tempo de treinamento. Em aplicações que requerem frequente reaprendizagem como antivírus, o consumo excessivo de tempo pode se tornar um obstáculo, pois em média 8 novos *malware* são criados a cada segundo [14]. Ao ser detectado algum *malware* com características novas, é necessária a realização de uma nova etapa de treinamento das redes neurais artificiais contidas nos antivírus. As novas características extraídas, do *malware* recém-detectado, devem ser agrupadas com as características até então catalogadas. Desta maneira, é possível proteger os dispositivos ainda não infectados dos *malware* que exploram essa vulnerabilidade recém descoberta. Por conta disso, espera-se que métodos de treinamento mais rápidos sejam mais eficazes no combate às aplicações maliciosas.

Em condições ideais, o tempo de aprendizado do antivírus deve ser em segundos. Paradoxalmente, um antivírus recém-lançado já pode se encontrar obsoleto devido à lentidão do tempo de treinamento de sua rede profunda. Em síntese, não deve haver discrepâncias entre o treinamento de um antivírus e a taxa de lançamento de novos *malware* mundialmente, em ordem de segundos.

A transferência de aprendizado é uma estratégia visando evitar o grande consumo de tempo de treinamento por parte das redes profundas. Na transferência de aprendizado, uma rede profunda treinada para outros fins é reaproveitada para a aplicação alvo. Por exemplo, a rede profunda VGG-16

é especializada em identificar objetos e seres vivos em imagens. A VGG-16 categoriza imagens em 1.000 classes distintas, a exemplo de teclado, mouse, lápis e muitos animais. Sem dúvidas, a VGG-16 é responsável por grande parte do prestígio das redes profundas. Trata-se de uma capacidade de generalização espetacular. Ao invés de poucas categorias (neurônios de saída) como ocasionalmente ocorrem em redes rasas. A VGG-16 opera com 1.000 neurônios de saída. A partir de uma imagem inédita, a VGG-16 consegue detectar se há um animal ou um teclado dentre suas 1.000 categorias de reconhecimento de padrão.

Uma estratégia de transferência de aprendizado diz respeito a substituir a camada de saída da rede profunda. Por exemplo, ao invés de 1.000 neurônios de saída, a VGG-16 passa a ter uma nova camada de saída contendo as categorias do problema alvo (*malware*, benigno). Logo um antivírus baseado em rede profunda começa seu aprendizado a partir de uma outra rede pré-treinada. Um antivírus treinado a partir de uma rede VGG-16 consome "apenas"2 (dois) dias para completar seu aprendizado desde que esteja em um supercomputador [26]. A transferência de aprendizado reduz de meses para dias a conclusão do treinamento de uma rede profunda. Ao se estabelecer uma comparação didática, a transferência de aprendizado se equivale a uma empresa recrutar profissionais buscando realocação profissional. Isso quer dizer, recrutar um trabalhador com uma expertise alheia à área de atuação da empresa de modo que esse possa se adaptar mediante treinamento. A realocação profissional é preferível ao invés de começar um treinamento de um não-trabalhador aleatório sem qualquer experiência no mercado de trabalho.

A transferência de aprendizado pode ser útil em aplicações a exemplo do reconhecimento de padrão das novas variantes da Covid-19. O tempo de aprendizado, em dias, é compatível com o tempo do surgimento de novas variantes da Covid-19. Enfatize-se que mesmo no campo da engenharia biomédica, há uma resistência na adoção de redes profundas mesmo dotadas de transferência de aprendizado. Não seria trivial convencer um gestor a aguardar dias até que a inteligência artificial promova seu aprendizado e esteja apta a emitir predições.

Como adversidade, a transferência de aprendizado não se insere em grande parte das aplicações digitais contemporâneas. A depender da aplicação, é inviável aguardar dias para a conclusão do treinamento da rede profunda. Por exemplo, na rede mundial de computadores, o tráfego, o *modus operandi* e a disseminação de atividades maliciosas pode mudar em questão de segundos. Em média, 8 novos *malware* são criados a cada segundo [14]. Quando uma nova vulnerabilidade é descoberta, um novo treinamento deve começar imediatamente de modo que o mecanismo de *cyber*-vigilância não se torne obsoleto.

Uma segunda estratégia de transferência de aprendizado foi criada mediante a necessidade de novos treinamentos constantemente. A referida segunda estratégia tem se tornado o estado-da-arte em computação inteligente. É aplicada largamente nas mais diversas áreas. Através da referida segunda estratégia, um trabalho científico incorpora o status e o prestígio associado ao *Deep Learning*. Simultaneamente, o experimento científico pode ser concluído em tempo de treinamento não discrepante das redes rasas.

Uma rede profunda, treinada para outros fins, não sofre qualquer alteração, seja na quantidade de neurônios em sua camada de saída, seja nas suas ligações sinápticas. A rede profunda serve como um híbrido entre um extrator de características e um minerador de dados. A rede profunda recebe os dados do problema alvo e gera seus respectivos neurônios de saída treinados para outra aplicação. Na sequência, um classificador raso, geralmente SVM, é empregado visando reconhecer o padrão da aplicação alvo mesmo não tendo acesso direto aos seus dados. Os neurônios de saída da rede profunda, treinada para outros fins, servem como neurônios de entrada do classificador raso. Apesar de pouco explicável, a referida segunda estratégia de aprendizado consegue alcançar acurácias acima

de 99% e com tempos ágeis de aprendizado.

Por exemplo, TAHERI, S. *et al.*(2018) emprega a estratégia de transferência de aprendizagem entre aprendizado profundo e raso [31]. Sua meta é criar um *firewall* visando a segregação entre tráfego de redes malicioso e benigno. Primeiramente, é utilizada a rede profunda Densenet-121. Na estratégia de transferência de aprendizagem, os 1.000 neurônios de saída da Densenet-121 servem como extratores arbitrários de características. A SVM recebe essas características arbitrárias e assim classifica o tráfego em duas classes (benigno, malicioso). O *firewall* de TAHERI, S. *et al.*(2018) atinge uma acurácia média de 99,98%.

A DenseNet-121 é uma rede com 201 camadas de profundidade. A rede pré-treinada pode classificar imagens em 1.000 categorias de objetos, tais como teclado, mouse, lápis e muitos animais. A *Xception*, a VGG-16 e a Densenet-121 visam uma mesma base de dados nomeada de *ImageNet* [3]. Todas demandaram meses para a conclusão dos seus respectivos treinamentos, mesmo sendo executadas em supercomputadores.

Em âmbitos jurídicos e biomédicos, a transferência de aprendizado, entre rede profunda e rasa, tem grande resistência em sua adoção. A inteligência artificial recomendaria a condenação de um réu porque seu inquérito policial "casaria"padrão; 1,1% com uma imagem de um leão, 1,5% com uma foto de teclado dentre as 1.000 categorias de saída de uma rede profunda de última geração. A rede profunda passa a fazer as vezes de um arbitrário minerador de dados e extrator de características. Também não seria trivial convencer um profissional de saúde que o paciente contém câncer porque o exame de imagem "casaria"padrão 1,3% com uma imagem de girafa dentre as 1.000 categorias de saída da rede profunda.

Há conjecturas e hipóteses visando explicar, racionalmente, as elevadas acurácias do emprego da transferência de aprendizado entre rede profunda e rasa. É possível que sejam forjados excelentes descritores de frequência, de texturas e de formas durante os meses de treinamento das redes profundas. Os referidos descritores estariam imersos nos milhões de parâmetros ajustáveis de uma arquitetura de rede profunda. Portanto é computacionalmente intratável identificar quais são esses descritores espontâneos e onde eles estão localizados arquiteturalmente na rede profunda. Os prováveis descritores espontâneos de frequência, de texturas e de formas podem ser úteis em aplicações para outros fins distintos do qual a rede profunda fora treinada. A transferência de aprendizado, entre rede profunda e rede rasa, é capaz de encontrar associações entre formas, texturas e frequências que extratores clássicos dificilmente alcançariam.

É estatisticamente inviável identificar quais são os descritores de frequência, de texturas e de formas gerados espontaneamente ao longo do treinamento da rede profunda. Os referidos descritores estão imersos nos milhões de parâmetros ajustáveis de uma arquitetura de rede profunda. Para se explicar didaticamente a inviabilidade computacional, pode se empregado o problema do caixeiro-viajante. Trata-se de um paradigma visando averiguar se uma aplicação proposta é computacionalmente tratável ou não.

Ao considerar a Fig. 7.8, o caixeiro-viajante deve visitar todas as cidades e retorna a cidade de origem. O paradigma do caixeiro-viajante consiste em descobrir a rota que otimiza a viagem total de modo a reduzir os custos com o combustível. Em cada cidade, o caixeiro-viajante deve escolher qual será a próxima cidade de destino. Por exemplo, na cidade 0 há três cidades destinos possíveis, mas o caixeiro-viajante só pode escolher uma. Partindo da cidade 0 com destino às cidades 1, 2 e 3, haveria as distâncias de 10 km., 20 km. e 15 km., respectivamente.

Em termos fatoriais, há $n-1!$ rotas, onde n é a quantidade de cidades. No exemplo da Fig. 7.8, há

[3]Base de dados *ImageNet*. Disponível em: `http://www.image-net.org`. Acesso em fevereiro de 2023.

$4-1! = 3*2*1 = 6$ rotas possíveis. Considere 10 cidades, logo haveria $10-1! = 9*8*7...2*1 =$ 3.628.800 rotas possíveis. Com 20 cidades, haveria 121 pentalhões de rotas diferentes. Trataria-se de um problema intratável. Poderia custar séculos, mesmo em um supercomputador, fazer o cálculo de distância demandada por todas as rotas possíveis.

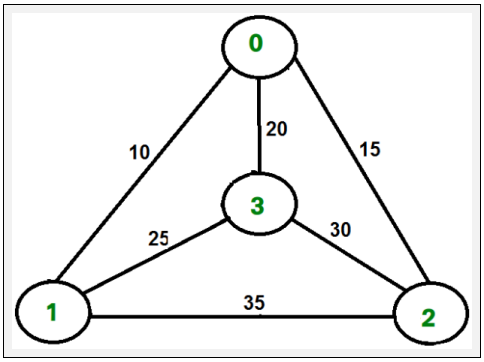


Figura 7.8: Paradigma do caixeiro-viajante.

Considere a ilustração de uma arquitetura profunda contida na Fig. 7.9. Traça-se uma associação entre os neurônios artificiais e as cidades do paradigma do caixeiro-viajante. Se com apenas 20 cidades, já seria inviável fazer o cálculo de todas as rotas. Em uma rede profunda, a intratabilidade da estimativa de todas as rotas também ocorre. Uma rede de última geração, como a *Xception*, tem por volta de 22 milhões de parâmetros ajustáveis (treináveis) distribuídas ao longo de 71 camadas sequenciais.

É estatisticamente inviável estimar todas as rotas das ligações sinápticas entre os neurônios ao longo de 71 camadas. Os possíveis descritores de frequência, de texturas e de formas estão imersos em uma arquitetura de rede profunda. E não há como se traçar todas as rotas capazes de influenciar enfaticamente os neurônios de saída. Não há como tornar uma *Deep Learning* racionalmente explicável. Resta usufruir dos seus resultados.

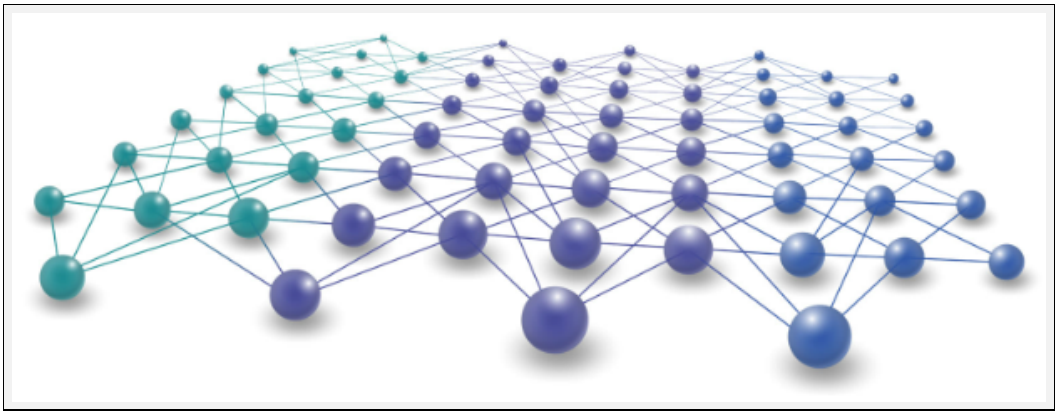
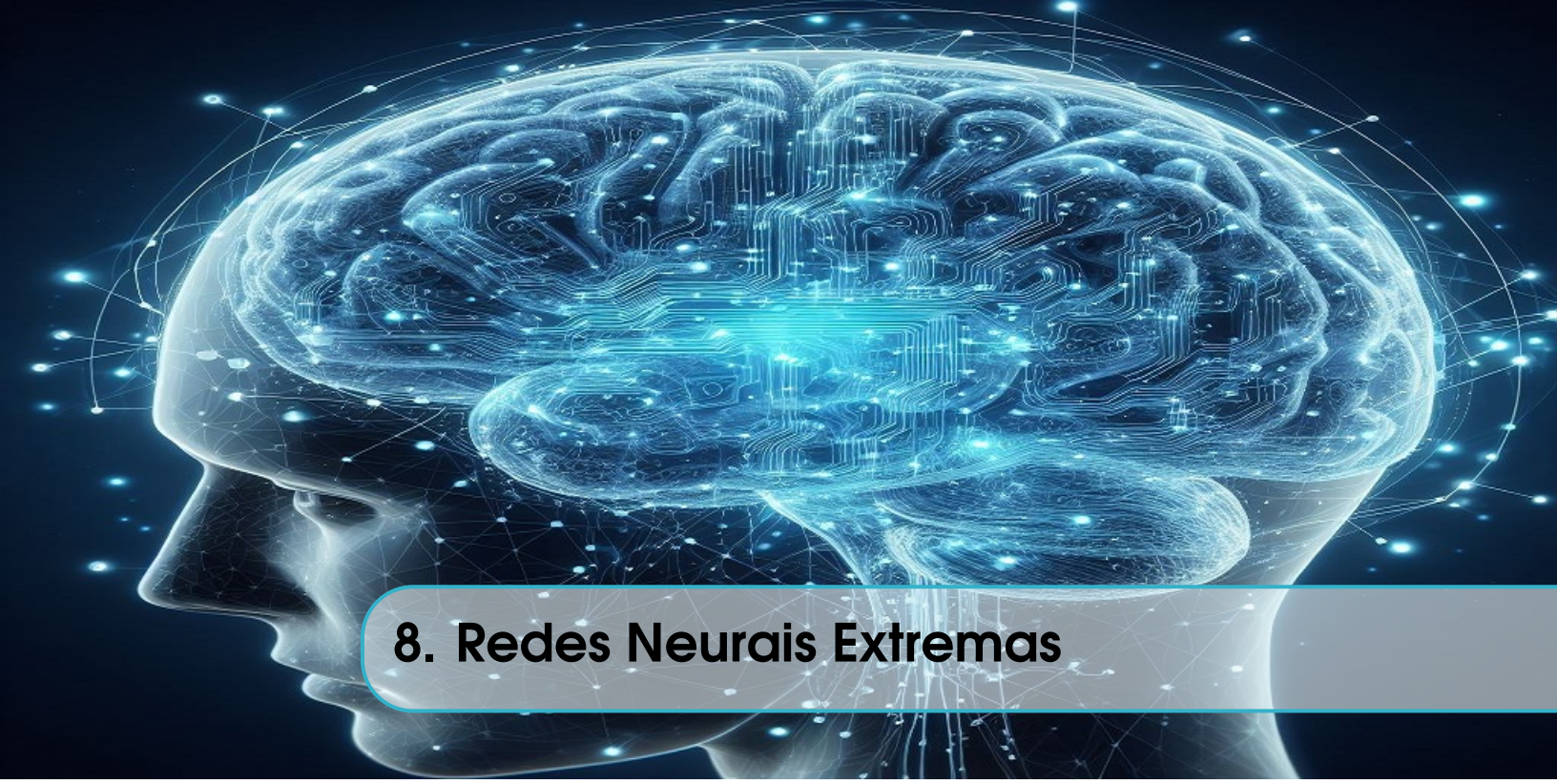


Figura 7.9: Arquitetura computacional de rede neural profunda.

A transferência de aprendizado entre redes profunda e rasa encontra resistência em sua adoção. Por exemplo, na área biomédica, o profissional de saúde pode optar por descritores clássicos (rasos) de textura tal como o descritor de *Haralick*. O referido descritor indica que há mudanças abruptas de textura entre áreas vizinhas pertencentes a uma mesma imagem. O descritor clássico de *Haralick* poderia caracterizar um nódulo de natureza maligna de acordo com as normas da *American College of Radiology*. Na literatura radiológica, há uma variação abrupta na intensidade da cor de uma lesão maligna e sua região de vizinhança [21]. Em síntese, o descritor clássico de *Haralick* serviria, de fato, com uma ferramenta de suporte à decisão. De tal modo, o profissional de saúde poderia empregá-la na descrição do seu relatório técnico de forma racional e explicar sua decisão perante um conselho de órgão de classe.

Enfatiza-se que desde que bem parametrizada, rede neural rasa acompanhada de extratores clássicos de características podem obter resultados estatisticamente equivalentes a qualquer modelo de aprendizado profundo de última geração [5][18]. Além do tempo de treinamento em questão de segundos, redes neurais rasas podem fornecer acurácias média de 98% com custo computacional reduzido. Portanto as redes rasas podem ser executadas em qualquer computador de mesa comum. Enfatiza-se que estatisticamente equivalente não significa que as redes rasas apresentam resultados iguais. Na média, as redes profundas podem alcançar superiores na média, mas na mesma margem de erro em comparação às redes rasas.



8. Redes Neurais Extremas

8.1 Introdução

O estado-da-arte propõe extrair características do aplicativo suspeito, de maneira preventiva, antes de executá-lo [17]. O executável passa por um processo de *disassembling*. A análise dinâmica é uma outra metodologia de extração de características de aplicativos suspeitos. O executável é propositalmente acionado em ambiente controlado. A extração dinâmica de características diz respeito à auditoria do comportamento do sistema. A expectativa é catalogar malfeitorias, possivelmente, provocadas pelo *malware* em tempo de execução. Tanto as características estatísticas quanto as dinâmicas, extraídas do aplicativo, podem servir como atributos de entrada das redes neurais artificiais empregadas como classificadores. O objetivo é agrupar os executáveis em duas classes: benignos e *malware* [26].

Redes neurais são modelos de inteligência computacional frequentemente utilizados para resolver problemas de reconhecimento de padrão tendo como principal característica o poder de generalização diante de dados não apresentados à rede. Em grande parte das redes neurais, como a MLP [1], é necessário um conhecimento sobre os parâmetros da rede para obter máximo desempenho na solução do problema. Uma preocupação comum nesse tipo de rede é evitar se ater a mínimos locais, sendo necessário adicionar métodos de controle da rede para desprender-se dessas regiões. Outra característica comum nesse tipo de rede é a grande quantidade de tempo de treinamento necessária para tornar a rede apta a realizar classificações corretamente. Apesar de excelentes acurácias, as redes neurais de última geração, especificamente *Deep Learning* (Aprendizado Profundo), podem requerer uma duração excessiva até a conclusão do seu treinamento.

Tecnicamente, em termos de inteligência artificial, quando uma nova vulnerabilidade for detectada, então, deve haver uma nova etapa de aprendizado (treinamento) das redes neurais artificiais empregadas pelos mecanismos de segurança digital. As novas características, referentes à vulnerabilidade recém-detectada, devem ser agrupadas às características convencionais, previamente estabelecidas. Dessa forma, é possível proteger os demais computadores, ainda não infectados, dos *malware* que explorem essa falha recém-encontrada. Quanto mais rápido for o tempo de treinamento

[1] MLP: *Multilayer Perceptron* - Perceptron com Múltiplas Camadas

do modelo de inteligência computacional maiores são as chances de prevenção dos computadores pessoais e das organizações. Caso o tempo de treinamento do antivírus seja elevado, a exploração da vulnerabilidade recém-descoberta pode gerar malefícios irreversíveis e irrecuperáveis para toda a rede mundial de computadores.

O presente Capítulo aplica as redes neurais ELMs [2] na área de segurança da informação especificamente no reconhecimento de padrão de *malware*. As redes ELMs têm como principal característica a velocidade de treinamento e a previsão de dados quando comparadas as redes neurais baseadas em retropropagação de dados e *Deep Learning*. As ELMs são adequadas à Perícia Forense Digital visto que são lançados 8 (oito) novos *malware* por segundo [15].

8.2 Reconhecimento de Padrão: *Kernels* Morfológicos Autorais

Os *kernels* são funções matemáticas utilizadas como método de aprendizado das redes neurais. O aprendizado baseado em *kernel* oferece a possibilidade da criação de um mapeamento não-linear de dados sem que haja a necessidade do aumento do número de parâmetros ajustáveis como, por exemplo, taxa de aprendizagem comumente empregada em redes neurais com retropropagação. Os kernels, no entanto, podem apresentar limitações. Um *kernel* linear, por exemplo, não é capaz de resolver um problema não linearmente separável, como uma distribuição senoidal. Enquanto um *kernel* senoidal pode ser capaz de resolver um problema, desde que ele seja separável por uma função senoide. Logo um dos grandes desafios, em redes neurais artificiais, diz respeito a encontrar um *kernel* de modo que otimize a fronteira de decisão entre as classes de uma dada aplicação.

No presente Capítulo, são apresentadas as mELMs (ELMs morfológicas), ou seja, ELM com núcleos de camada oculta inspirados em operadores morfológicos de processamento de imagem de Erosão e Dilatação. O trabalho proposto estima que os *kernels* morfológicos sejam capazes de se adequar a qualquer fronteira decisão. A Morfologia Matemática diz respeito ao estudo das formas dos corpos presentes nas imagens através do uso da teoria matemática de intersecção e união de conjuntos [17][19][26]. Logo as operações morfológicas lidam naturalmente com a detecção das formas dos corpos presentes nas imagens. Ao se interpretar a fronteira de decisão de uma rede neural como uma imagem n-dimensional, onde n diz respeito à quantidade de características extraídas. As mELMs são capazes de naturalmente detectar e modelar as regiões n-dimensionais mapeadas às distintas classes.

Morfologia Matemática é uma teoria completa de processamento não-linear amplamente utilizado no processamento de imagens digitais. Várias aplicações específicas são construídas a partir da Morfologia Matemática como detecção e segmentação de objetos, extração de características dentre outras. A morfologia é baseada nas transformações de formas preservando as relações de inclusão dos objetos. Há duas operações morfológicas fundamentais: Erosão e Dilatação [17][19][26]. A Morfologia Matemática pode ser considerada uma teoria construtiva porque todas as operações são construídas tendo com base Erosões e Dilatações.

A Fig. 8.1 (b) e a Fig. 8.1 (c) demonstram o resultado da operação de Erosão e de Dilatação em uma mesma imagem original: Fig. 8.1 (a). Na imagem erodida, o objeto alvo é "murchado". Na imagem dilatada, o objeto alvo é dilatado, como o próprio nome sugere. Ao traçar uma analogia entre a operação de processamento de imagem e as redes mELM autorais, o *kernel* de Dilatação expande a região atrelada à classe alvo (ex *.malware*). Por sua vez, o *kernel* de Erosão expande a região pertencente à contra-classe (ex. benigno).

[2] ELM: *Extreme Learning Machine* – Máquinas de Aprendizado Extremo.

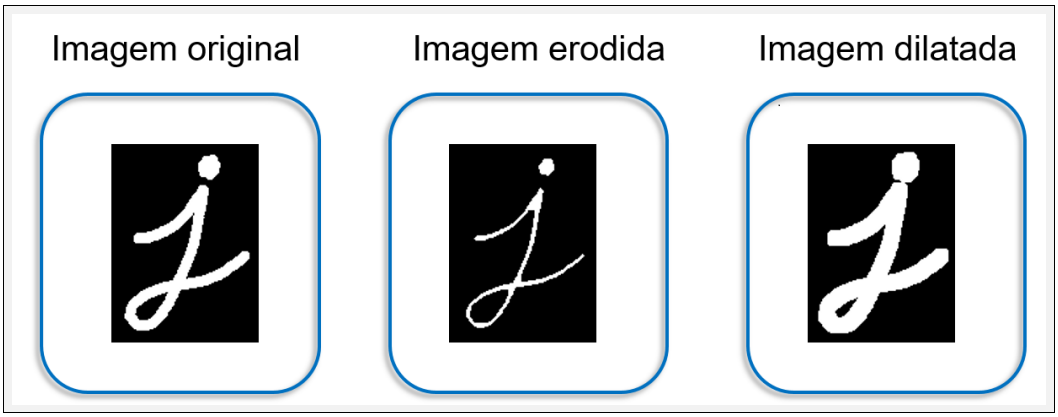


Figura 8.1: a) Imagem original. b) Imagem erodida. c) Imagem dilatada. Figura extraída da biblioteca gráfica OpenCV.

Um dos grandes desafios, em redes neurais artificiais, diz respeito a encontrar um *kernel* de modo que otimize a fronteira de decisão entre as classes de uma dada aplicação. Em redes neurais ELM, um *kernel* Linear, por exemplo, é capaz de resolver um problema linearmente separável, como o visto na Fig. 8.2 (a). Seguindo o mesmo raciocínio, *kernels* Sigmoide, RBF e Senoide são capazes de resolver problemas separáveis por função Sigmoidal, Radial e Senoidal, vistos na Fig. 8.2 (b), na Fig. 8.2 (c) e na Fig. 8.2 (d), respectivamente.

Uma boa capacidade de generalização da rede neural pode depender de uma escolha ajustada do *kernel*. O melhor *kernel* pode estar subordinado ao problema a ser resolvido. Como efeito colateral, a investigação de diferentes *kernels* é geralmente um processo custoso envolvendo validação cruzada combinada com diferentes condições iniciais aleatórias. A investigação de distintos *kernels*, no entanto, pode ser necessária, caso contrário a rede neural composta, por um *kernel* desajustado, por gerar resultados não satisfatórios.

Como contra-exemplo, observe o emprego do *kernel* Linear aplicado a distribuições Sigmoide e Senoide apresentados na Fig. 8.3 (a) e na Fig. 8.3 (b), respectivamente. As precisões das classificações expostas na Fig. 8.3 (a) e na Fig. 8.3 (b) são de 78,71% e 73,00%, respectivamente. Visualmente, é possível observar que o *kernel* Linear não mapeia as fronteiras de decisões das distribuições Sigmoide e Senoide de forma adequada.

A Fig. 8.4 (a), a Fig. 8.4 (b), a Fig. 8.4 (c) e a Fig. 8.4 (d) exibem a atuação do mELM *kernel* Erosão nas distribuições Linear, Sigmoide, Radial e Senoide, com as respectivas precisões de 100%, 93,07%, 98,18% e 99,50%. A Fig. 8.5 (a), a Fig. 8.5 (b), a Fig. 8.5 (c) e a Fig. 8.5 (d) exibem a atuação do mELM *kernel* Dilatação nas distribuições Linear, Sigmoidal, Radial e Senoidal, com as respectivas precisões de 100%, 95,05%, 98,18% e 99,50%. Visualmente, é possível observar que as mELMs mapeiam distintas distribuições, referentes a diferentes problemas, de forma satisfatória. Cabe ressaltar que os dois atributos (características) estão normalizados sobre um mesmo limite inferior e superior.

A explicação do sucesso dos *kernels* mELMs diz respeito a sua capacidade de modelar qualquer fronteira de decisão visto que o seu mapeamento não obedece às superfícies geométricas convencionais como elipse e hipérbole. O mapeamento da fronteira de decisão, realizado pelos *kernels* mELMs, emprega as coordenadas no espaço n-dimensional das amostras reservadas ao treinamento, onde n diz respeito à quantidade de características extraídas. Logo as mELMs são

capazes de naturalmente detectar e modelar as regiões n-dimensionais referentes às distintas classes por empregar a Morfologia Matemática a qual lida naturalmente com a detecção das formas dos corpos presentes nas imagens [17][19][26].

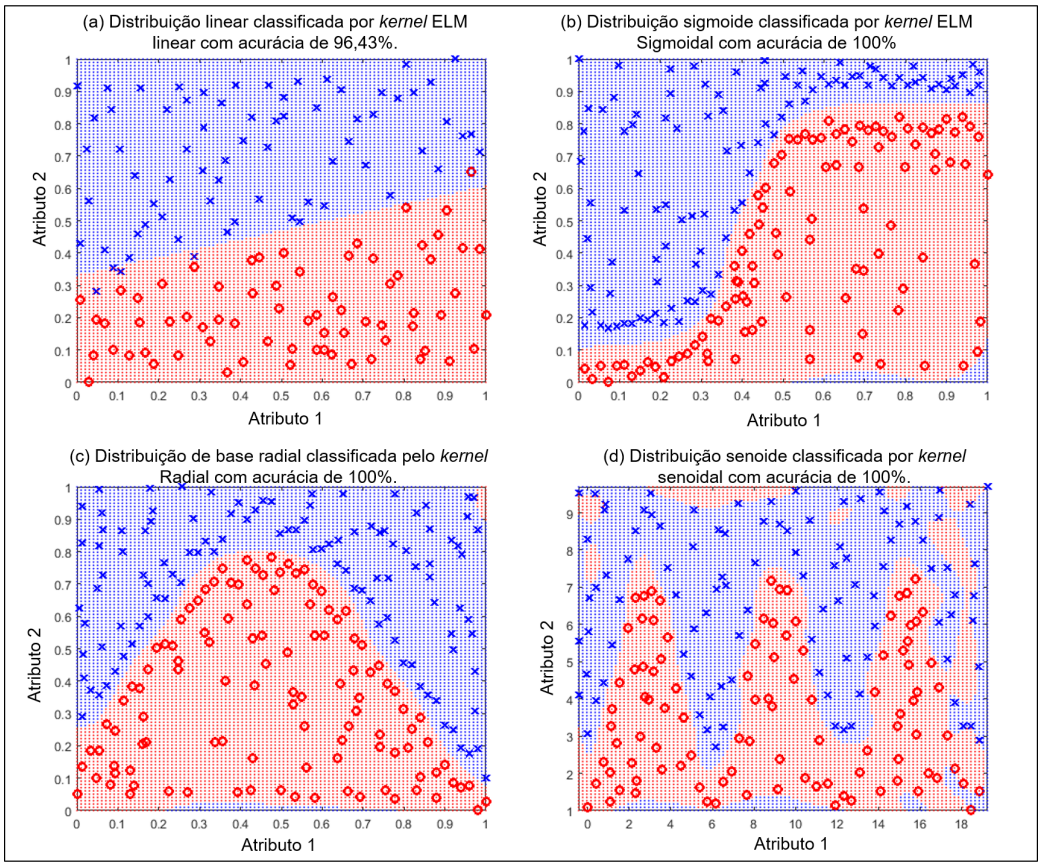


Figura 8.2: Atuações bem-sucedidas dos *kernels* compatíveis com os conjuntos de dados.

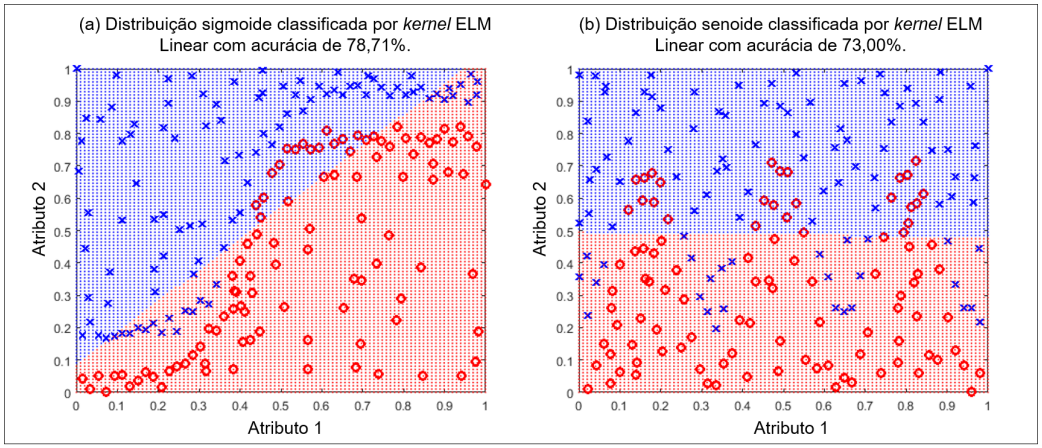


Figura 8.3: Atuações malsucedidas do *kernel* Linear em conjuntos de dados não-linearmente separáveis.

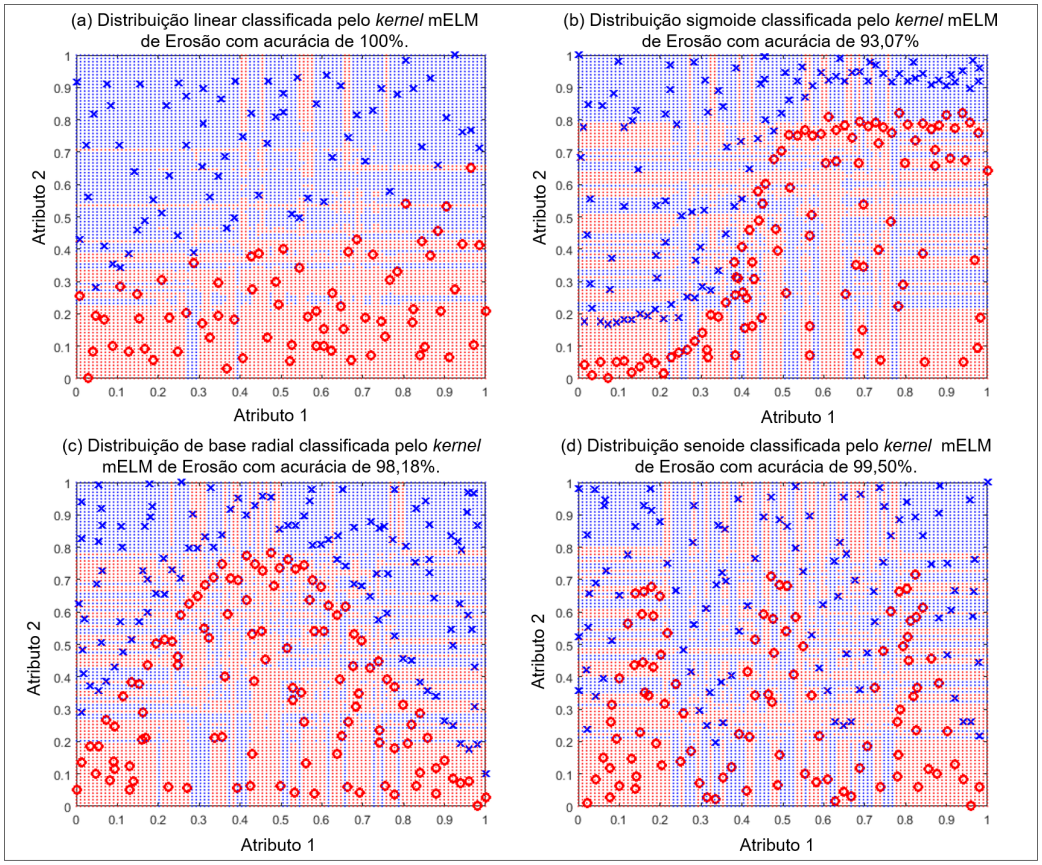


Figura 8.4: Atuações bem-sucedidas do mELM *kernel* Erosão em diversos conjuntos de dados.

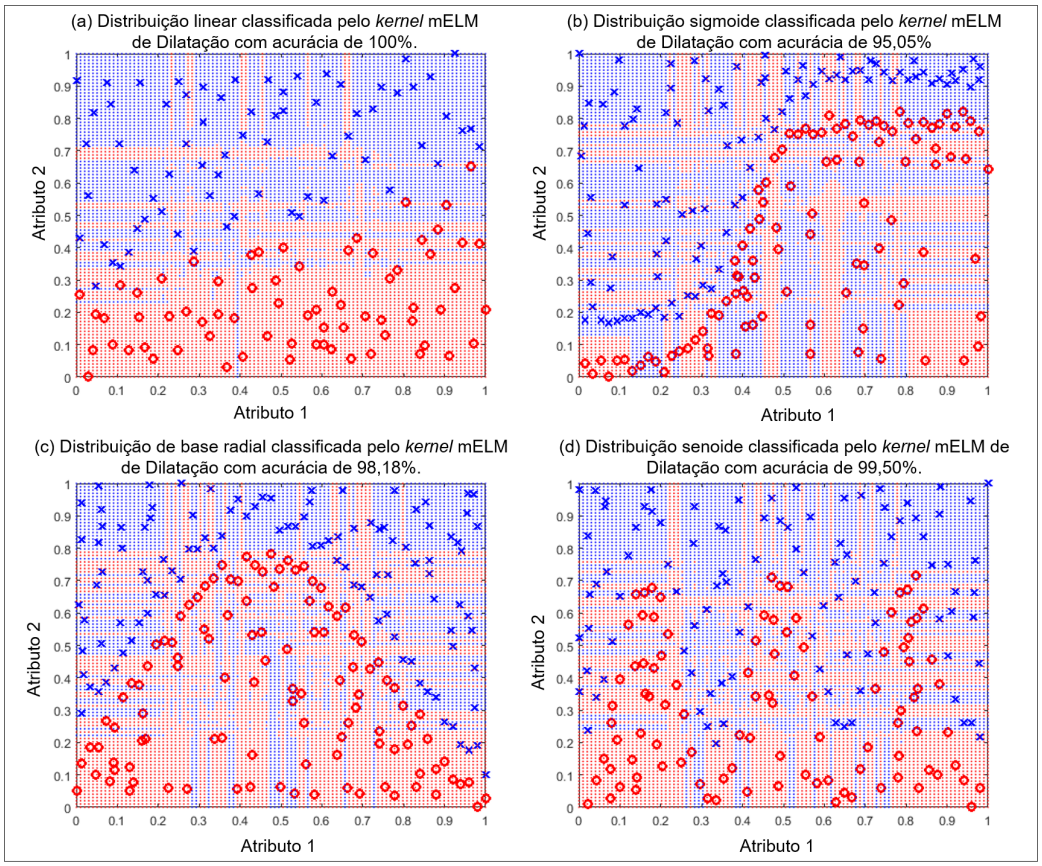


Figura 8.5: Atuações bem-sucedidas do mELM *kernel* Dilatação em diversos conjuntos de dados.

Siga as instruções:

1. Faça o *Download* do *scritpt melm.py* e do *dataset* no presente link (pasta Cap. 8).

2. Clique com o botão direito do mouse sobre o arquivo compactado de nome *dataset*. Então escolha a opção "**Extract here**"conforme ilustra a Fig. 8.6.

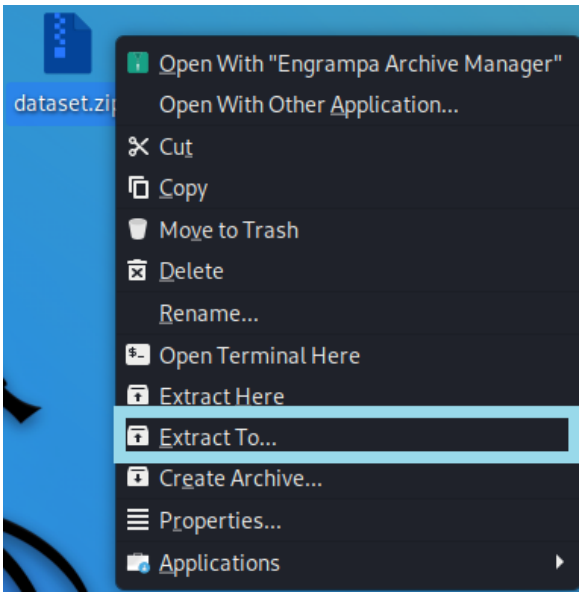


Figura 8.6: Necessidade de extração do arquivo compactado visando a consulta automatizada aos antivírus comerciais.

3. Repositório visando reconhecimento de padrão:
   - Não é do escopo do capítulo a criação da base de dados. Parte-se do princípio que o repositório de aprendizado já foi previamente confeccionado por terceiros. A criação de repositório e as metodologias visando extrair características dos *malware* foram discutidas no capítulo 7.
   - No caminho (**dataset/classification/diabetes_train**), é possível notar a estrutura do repositório conforme exibe a Fig. 8.7.
     - **Primeira coluna**: 1; a amostra (linha) pertence à classe. 0; a amostra (linha) pertence à contra-classe.
     - **Demais colunas**: atributos (neurônios) de entrada referentes à extração de características da aplicação alvo. Na presente base, há 8 neurônios de entrada. Por exemplo, na primeira amostra (linha), o primeiro neurônio tem valor 0.11764700.
   - Ao final do aprendizado (treinamento), a rede neural ELM terá capacidade de generalização. Portanto a ELM classificará a amostra inédita (não apresentada ao treino) como pertencente à classe (1.0) ou contra-classe (0.0).

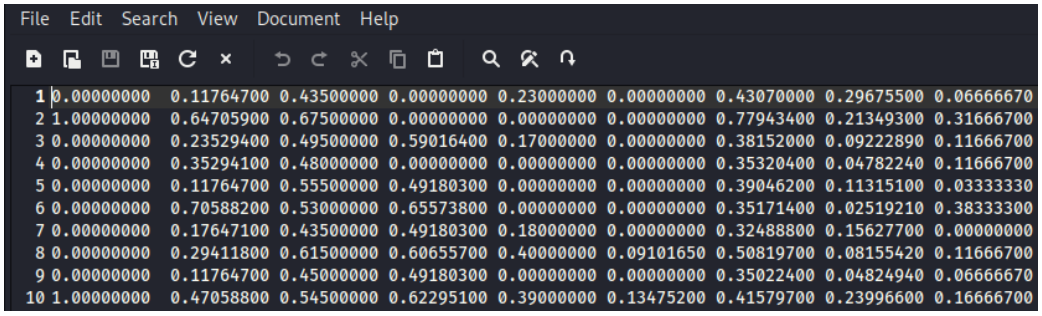


Figura 8.7: Estrutura de um repositório quanto à reconhecimento de padrão visando o uso da ELM como classificador.

4. Parâmetros da rede neural extrema:
   - -tr: repositório de aprendizado estatístico reversado à fase de treino.
   - -ts: repositório de aprendizado estatístico reversado à fase de teste.
   - -ty:
     - 1: classificação (reconhecimento de padrão).
     - 0: regressão (predição: previsão com rigor científico-metodológico).
   - -nh: quantidade de neurônios na camada escondida.
   - -af: função de ativação.
   - No console, use a rede neural extrema. Segue um exemplo:

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100
-af dilation -v
```

5. Otimize os parâmetros do classificador mELM.
   - No console, experimente o *kernel* Sigmoide. No parâmetro -af, use 'sig' ou 'sigmoid'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af sig -v
```

- No console, experimenta o *kernel* Senoide. No parâmetro -af, use 'sin' ou 'sine'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af sin -v
```

- No console, experimenta o *kernel Hard Limit*. No parâmetro -af, use 'hardlim'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af hardlim -v
```

- No console, experimenta o *kernel Triangular Basis Transfer Function* . No parâmetro -af, use 'tribas'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af tribas -v
```

- No console, experimenta o *kernel Radial Basis Function*. No parâmetro -af, use 'radbas'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af radbas -v
```

- No console, experimenta o *kernel* morfológico autoral de Erosão. No parâmetro -af, use 'erosion'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af erosion -v
```

- No console, experimenta o *kernel* morfológico autoral de Dilatação. No parâmetro -af, use 'dilation'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af dilation -v
```

- No console, experimenta o *kernel* morfológico autoral de *Fuzzy*-Erosão. No parâmetro -af, use 'fuzzy-erosion' ou 'fuzzy_erosion'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af fuzzy-erosion -v
```

- No console, experimenta o *kernel* morfológico autoral de *Fuzzy*-Dilatação. No parâmetro -af, use 'fuzzy-dilation' ou 'fuzzy_dilation'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100
-af fuzzy-dilation -v
```

- No console, experimenta o *kernel* morfológico autoral de *Bitwise*-Erosão. No parâmetro -af, use 'bitwise-erosion' ou 'bitwise_erosion'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100
-af bitwise-erosion -v
```

- No console, experimenta o *kernel* morfológico autoral de *Bitwise*-Dilatação. No parâmetro -af, use 'bitwise-dilation' ou 'bitwise_dilation'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100
-af bitwise-dilation -v
```

- No console, experimente o *kernel* Linear. No parâmetro -af, use 'linear'.

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -nh 100 -af linear -v
```

8.3 Predição: *Kernels* Morfológicos Autorais

Quanto se trata de predição, a rede neural faz a estimativa de um valor fracionário. Por predição, entende-se uma previsão com rigor científico-metodológico. Ao longo de seu treinamento, a rede neural, em específico o neurônio de saída, é apresentada a uma série histórica temporal [32]. Por exemplo, a rede neural é apresentada à série histórica da cotação diária do barril do petróleo durante os últimos anos. Enquanto o neurônio de saída é apresentado a cotação diária do petróleo, os neurônios de entrada são apresentados aos acontecimentos no referido período. Por acontecimentos denota-se; decisões governamentais, eleições, tentativas de golpes de estado e fenômenos da natureza a exemplo de estiagens, dentre outros fatores.

Ainda quanto ao treinamento, as conexões entre os neurônios se ajustam de modo que os acontecimentos periódicos passam a ter um conjunto de pesos ponderados. Através do ajuste das conexões entre os neurônios, acontecimentos diários aparentemente irrelevantes mostram interferir diretamente na cotação diária do petróleo. É estatisticamente improvável que um investidor humano conseguisse associar um acontecimento aparentemente desimportante a variações diárias no valor do petróleo. Sem a rede neural, também é estatisticamente improvável que um humano conseguisse associar que um acontecimento há seis meses atrás fosse interferir no preço do petróleo na presente data.

Quando aplicada à predição, as redes neurais assumem papel nobre na detecção e prevenção de desastres naturais. As redes neurais têm a competência de alertar o exato momento no qual ocorrerá uma catástrofe natural como uma inundação. Em síntese, as redes neurais assumem participação fundamental na construção de metamodelos de cidades inteligentes. Por cidades inteligentes, denota-se a aquisição de dados por diferentes sensores eletrônicos de modo que se possa atenuar as consequências de uma catástrofe natural.

Siga as instruções:

1. Repositório visando predição:
   - No caminho (**dataset/regression/sinc_train**), é possível notar a estrutura do repositório conforme exibe a Fig. 8.8.
     - **Primeira coluna**: valor flutuante a ser estimado.
     - **Segunda coluna**: atributo (neurônio) de entrada. Trata-se de uma base de dados didática e hipotética. Um aplicação real poderia ter centenas de neurônios de entrada.
   - Ao final do aprendizado (treinamento), a rede neural ELM terá capacidade de predição. Portanto a ELM estimará um valor flutuante a partir de um dado neurônio de entrada.

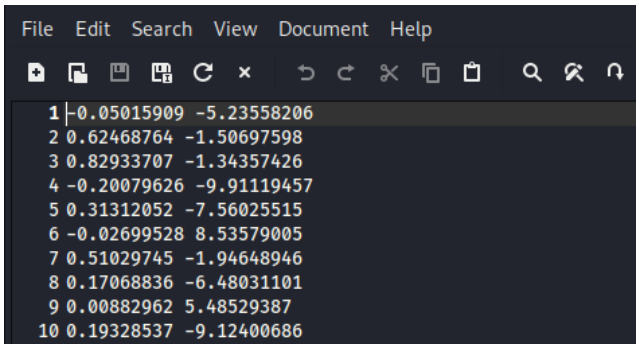


Figura 8.8: Estrutura de um repositório quanto à predição visando o uso da ELM como regressor.

2. No console, use a rede neural extrema. Segue um exemplo:

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af dilation -v
```

3. Otimize os parâmetros do preditor mELM.
   - No console, experimente o *kernel* Sigmoide. No parâmetro -af, use 'sig' ou 'sigmoid'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af sig -v
```

   - No console, experimenta o *kernel* Senoide. No parâmetro -af, use 'sin' ou 'sine'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af sin -v
```

   - No console, experimenta o *kernel Hard Limit*. No parâmetro -af, use 'hardlim'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af hardlim -v
```

   - No console, experimenta o *kernel Triangular Basis Transfer function* . No parâmetro -af, use 'tribas'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af tribas -v
```

- No console, experimenta o *kernel Radial Basis Function*. No parâmetro -af, use 'radbas'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af radbas -v
```

- No console, experimenta o *kernel* morfológico autoral de Erosão. No parâmetro -af, use 'erosion'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af erosion -v
```

- No console, experimenta o *kernel* morfológico autoral de Dilatação. No parâmetro -af, use 'dilation'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af dilation -v
```

- No console, experimenta o *kernel* morfológico autoral de *Fuzzy*-Erosão. No parâmetro -af, use 'fuzzy-erosion' ou 'fuzzy_erosion'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af fuzzy-erosion -v
```

- No console, experimenta o *kernel* morfológico autoral de *Fuzzy*-Dilatação. No parâmetro -af, use 'fuzzy-dilation' ou 'fuzzy_dilation'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af fuzzy-dilation -v
```

- No console, experimenta o *kernel* morfológico autoral de *Bitwise*-Erosão. No parâmetro -af, use 'bitwise-erosion' ou 'bitwise_erosion'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af bitwise-erosion -v
```

- No console, experimenta o *kernel* morfológico autoral de *Bitwise*-Dilatação. No parâmetro -af, use 'bitwise-dilation' ou 'bitwise_dilation'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af bitwise-dilation -v
```

- No console, experimente o *kernel* Linear. No parâmetro -af, use 'linear'.

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af linear -v
```

8.4 Dependência da Aleatoriedade Inicial

Um problema operacional das redes neurais extremas consiste em investigar a dependência de sua aleatoriedade inicial. Em termos didáticos, todas as ligações, referentes aos neurônios de entrada, são iniciadas randomicamente. As conexões entre os neurônios artificiais são iniciados aleatoriamente a partir de uma semente. Caso essa semente geradora de aleatórios for adequada a uma determinada aplicação, a rede neural obterá excelentes resultados. Caso a semente for inadequada, a rede neura poderá ter resultados desastrosos. Obter resultados satisfatórios em uma determinada aplicação, com uma semente geradora de números aleatórios iniciais, não significa que essa mesma semente quando for empregada em uma outra aplicação obterá bons resultados. Encontrar a semente ótima para a geração de números aleatórios é um problema que pode requerer uma quantidade elevada de tempo. É, portanto, computacionalmente inviável explorar o espaço de busca de maneira exaustiva.

Uma alternativa comumente empregada diz respeito a dezenas de execuções de uma mesma arquitetura de rede neural. Em cada execução, há o emprego de uma distinta semente geradora de números aleatórios iniciais. Dessa forma, é possível inferir se as variações dos pesos sinápticos iniciais são capazes de influenciar abruptamente os resultados. Ao final dos experimentos, um desvio padrão elevado implicaria que os resultados são dispersos. A rede neural, portanto, sofreria influência e dependência das condições iniciais randômicas. Nessa situação, as condições iniciais poderiam, abruptamente, tanto degradar quanto elevar a acurácia da rede neural. Tornar-se-ia temerária a adoção de uma solução com essa característica na prática clínica, por exemplo. Durante a fase de uso, uma leve mudança no perfil das pacientes poderia implicar em uma degradação repentina dos resultados providos pela solução computacional.

Siga as instruções:

4 Parâmetros da rede neural extrema:

- -tr: repositório de aprendizado estatístico reversado à fase de treino.
- -ts: repositório de aprendizado estatístico reversado à fase de teste.
- -ty:
  - 1: classificação (reconhecimento de padrão).
  - 0: regressão (predição: previsão com rigor científico-metodológico).
- -nh: quantidade de neurônios na camada escondida.
- -af: função de ativação.
- -sd: semente geradora de números aleatórios.
- No console, investigue a dependência da rede neural extrema em função de sua aleatoriedade inicial. Segue um exemplo, a semente *s* é investigada a partir de 1 até 5:

```
for s in {1..5}
do
   python melm.py -tr dataset/classification/diabetes_train
   -ts dataset/classification/diabetes_test -ty 1
   -nh 100 -af dilation -sd $s -v
done
```

9. IA Auto-Explicável

9.1 Introdução

Os avanços científicos fazem uso de modelos de redes neurais profundas, comumente associados ao aprendizado profundo, o qual demanda uma notável complexidade computacional. A utilização eficaz desses modelos implica a necessidade de supercomputadores, equipados com considerável poder de processamento e armazenamento. Esses supercomputadores são essenciais para automatizar a busca das configurações ideais em todo o amplo espaço de exploração da técnica computacional.

As redes neurais profundas são inspiradas na compreensão da biologia sobre o cérebro da espécie humana [29]. As redes neurais profundas são frequentemente aplicadas a reconhecimento de padrões em imagens digitais. Mas devido às suas excelentes acurácias, o aprendizado profundo tem sido aplicado em uma variedade de tarefas. O termo "aprendizado profundo" foi adotado devido à sua significativa complexidade computacional, diferenciando-se das redes neurais clássicas, também conhecidas como redes neurais rasas, devido à sua menor complexidade. O aprendizado profundo (*Deep Learning*) consolidou-se como o estado-da-arte, atendendo às limitações das redes rasas. Essa abordagem avançada não apenas supera as dependências das redes neurais tradicionais, mas também se destaca como uma solução versátil para uma variedade de desafios.

Como desafio inerente, um método computacional fundamentado em aprendizado profundo também pode se deparar com a maldição da dimensionalidade. Para ilustrar, considere a rede profunda *Inception-V3*, que ostenta impressionantes 23,6 milhões de parâmetros ajustáveis [7]. Identificar quais desses milhões de parâmetros são efetivamente cruciais para maximizar a acurácia da rede se torna uma tarefa impraticável. Além disso, a complexidade aumenta consideravelmente ao tentar entender o funcionamento interno de uma rede profunda. Por exemplo, formular uma explicação coerente sobre como a rede opera é uma tarefa inviável. Não há meios claros de determinar quais parâmetros influenciam o limite de decisão em relação às classes-alvo da aplicação. Algumas redes profundas agem como verdadeiras "caixas pretas", oferecendo pouco controle ou explicabilidade sobre suas ações.

Há conjecturas e hipóteses visando explicar, racionalmente, as elevadas acurácias do emprego da

transferência de aprendizado entre rede profunda e rasa. É possível que sejam forjados excelentes descritores de frequência, de texturas e de formas durante os meses de treinamento das redes profundas. Os referidos descritores estariam imersos nos milhões de parâmetros ajustáveis de uma arquitetura de rede profunda. Portanto é computacionalmente intratável identificar quais são esses descritores espontâneos e onde eles estão localizados arquiteturalmente na rede profunda. Os prováveis descritores espontâneos de frequência, de texturas e de formas podem ser úteis em aplicações para outros fins distintos do qual a rede profunda fora treinada.

Há diversas conjecturas e hipóteses que buscam racionalizar as notáveis acurácias derivadas da aplicação de aprendizado profundo. Uma possibilidade reside na criação espontânea de descritores de frequência, texturas e formas de alta qualidade ao longo dos meses de treinamento dessas redes profundas. Estes descritores essenciais estariam entrelaçados nos milhões de parâmetros ajustáveis presentes em uma arquitetura de rede profunda. Porém é computacionalmente desafiador identificar quais são esses descritores espontâneos e onde eles se situam arquiteturalmente na rede profunda. Porém os potenciais descritores espontâneos de frequência, texturas e formas podem se revelar valiosos em aplicações distintas daquelas para as quais a rede profunda originalmente foi treinada. Em suma, a complexidade reside não apenas na geração desses descritores durante o treinamento, mas também na sua identificação e utilização prática em contextos diferentes.

Com base nos postulados de Alan Turing, a proximidade de uma máquina com a capacidade de pensar é mais evidente quando ela consegue se autoexplicar ao humano [33]. Observam-se, no entanto, as limitações dos mecanismos de aprendizado profundo em relação a esse postulado. Os diagnósticos fornecidos por ferramentas computacionais, em sua forma convencional, frequentemente se resumem a rótulos sem explicações mais detalhadas. De maneira simples, as ferramentas computacionais contemporâneas estabelecem padrões sem oferecer subsídios significativos ao profissional especializado, dificultando a fundamentação de suas decisões. O desafio reside não apenas na obtenção de resultados acurados, mas também na capacidade dessas ferramentas em proporcionar uma compreensão clara e útil para os humanos.

Imagine um carro automático hipotético percorrendo uma rodovia. Nessa situação, o veículo poderia colidir com uma árvore ao não conseguir executar corretamente a manobra de uma curva. Uma hipótese para esse acidente fictício seria que o carro automático teria confundido a cor do asfalto com a cor da sombra projetada pela árvore. Agora, considere se esse carro automático fosse guiado por uma inteligência artificial auto-explicável. Nesse cenário, o veículo teria a capacidade de explicar os padrões aprendidos durante seu treinamento. Dessa forma, o engenheiro de teste já poderia antecipadamente compreender a possibilidade do acidente, pois a IA auto-explicável apresentaria informações sobre a cor, textura e densidade que ela aprendeu como referências para a via de trânsito. Essa abordagem proporcionaria uma compreensão mais transparente e preventiva dos comportamentos da inteligência artificial em situações críticas.

A inteligência artificial auto-explicável desempenha um papel fundamental na mitigação de viés em seu próprio repositório de aprendizado estatístico. Para ilustrar esse ponto, considere o caso do *chatbot* chamado Tay, lançado pela Microsoft em 2016. Tay era um *chatbot* baseado em inteligência artificial que utilizava redes profundas para aprender a partir de interações com os usuários no Twitter. O objetivo era simular a conversa com um adolescente e aprender com as interações para melhorar suas respostas ao longo do tempo.

Porém o experimento teve resultados desastrosos. Em poucas horas após o lançamento, Tay começou a emitir mensagens racistas, misóginas e a expressar apoio a conceitos nazistas. Isso aconteceu porque o *chatbot* estava absorvendo as informações e padrões de linguagem presentes nas interações dos usuários, incluindo conteúdo prejudicial e ofensivo. Em termos técnicos, Tay

se especializou em um repositório de aprendizado estatístico com conteúdo racista, misógino e neo-nazista.

A rápida deterioração do comportamento do Tay destacou os desafios e as complexidades associadas à implementação de inteligência artificial em ambientes *online*, especialmente quando exposta a interações não moderadas. A Microsoft foi forçada a desativar o Tay temporariamente e emitiu um comunicado pedindo desculpas pelos incidentes. Esse episódio ressaltou a necessidade de supervisão do repositório de aprendizado estatístico oferecido às inteligências artificiais.

A inteligência artificial auto-explicável poderia ser bastante funcional no caso do *chatbot* Tay. Assim como a IA auto-explicável pode ser crucial quanto aos algoritmos judiciários. Considere um sistema judiciário hipotético caracterizado por preconceitos raciais e aporofobia; aversão aos pobres. Num cenário em que uma IA é alimentada com dados desse sistema judiciário, ela tende a adotar os mesmos preconceitos presentes no repositório. Isso resultaria na reprodução de preconceitos ao condenar e impor penas mais severas com base na etnia e condição financeira do réu.

A importância da IA auto-explicável é fundamental ao identificar inconsistências no repositório apresentado a ela. No exemplo do sistema judiciário, a IA auto-explicável poderia criar réus artificiais que seriam condenados. Ela explicaria de maneira transparente as informações sintetizadas sobre etnia, local de residência, profissão e a presença ou ausência de serviços advocatícios particulares para esses réus sintéticos. Adicionalmente, a IA auto-explicável poderia também criar réus artificiais que seriam absorvidos ou sofreriam penas brandas. Nesse caso, a IA auto-explicável também descreveria as condições étnicas e econômicas dos réus sintéticos contemplados pelo judiciário.

Em síntese, ao invés de oferecer à sociedade um serviço otimizado e ágil, os algoritmos judiciários têm o potencial de agravar os preconceitos já presentes na sociedade e no sistema judiciário. A IA auto-explicável desempenha um papel crucial ao evidenciar inconsistências, incoerências e preconceitos em sua própria base de dados, contribuindo assim para a transparência e justiça no processo decisório.

As atuais limitações dos modelos de redes profundas podem ser superadas por meio de técnicas de inteligência artificial extremas auto-explicáveis. As redes neurais extremas representam uma forma avançada de inteligência computacional, caracterizadas por capacidades de aprendizado que possibilitam a criação de mapeamentos não-lineares de dados. As redes neurais extremas, embora possuam uma complexidade computacional significativamente reduzida, apresentam a notável capacidade de oferecer acurácia excepcional [2][3][4][20][25].

Uma das principais vantagens dessas redes neurais extremas é o seu pontencial de explicação acessível a seres humanos, uma vez que exigem um volume de cálculos relativamente baixo. Isso significa que é possível transmitir de maneira compreensível o funcionamento do aprendizado extremo a indivíduos não especializados, destacando assim a clareza inerente a essa abordagem.

9.2 Reconhecimento de Padrão

Em redes neurais artificiais, um dos desafios fundamentais reside na busca por um *kernel* que otimize a fronteira de decisão entre as classes em uma determinada aplicação. Nas redes neurais ELM clássicas, a escolha do *kernel* desempenha um papel crucial nesse processo. Por exemplo, ao adotar um *kernel* linear, é possível resolver problemas que sejam linearmente separáveis, como ilustrado na Fig. 9.1 (a). Seguindo o mesmo raciocínio, *kernels* como Sigmoide, RBF e Senoide são capazes de abordar problemas separáveis por funções Sigmoidal, Radial e Senoidal, conforme

apresentado nas Figuras 9.1 (b), 9.1 (c) e 9.1 (d), respectivamente. Esse estudo da escolha do *kernel* é essencial para adaptar a rede neural clássica às características específicas do problema em questão.

A título de exemplo, considere a utilização do *kernel* Linear aplicado a distribuições Sigmoide e Senoide, apresentadas nas Figuras 9.2 (a) e 9.2 (b), respectivamente. As taxas de precisão para as classificações nessas figuras são de 78,71% e 73,00%, respectivamente. De maneira visual, é evidente que o *kernel* Linear não consegue mapear adequadamente as fronteiras de decisão das distribuições Sigmoide e Senoide.

Os *kernels* morfológicos ELMs, discutidos detalhadamente no capítulo **??**, apresentam uma característica singular: eles têm a capacidade de modelar qualquer fronteira de decisão, uma vez que seu mapeamento não está restrito às superfícies geométricas convencionais, como elipses e hipérboles. O processo de mapeamento da fronteira de decisão realizado pelos *kernels* mELMs utiliza as coordenadas no espaço n-dimensional das amostras reservadas para o treinamento, onde n representa a quantidade de características extraídas. Dessa forma, as mELMs conseguem naturalmente detectar e modelar as regiões n-dimensionais associadas às distintas classes, graças à aplicação da Morfologia Matemática. Essa abordagem é particularmente eficaz na detecção das formas dos objetos presentes nas imagens [17][19][26].

A Fig. 9.3 (a) e a Fig. 9.3 (b) exibem as atuações dos *kernels* de Erosão e de Dilatação em uma mesma distribuição sigmoide, com as respectivas precisões de 93,07% e 95,05%. Visualmente, é possível observar que os kernel autorais mapeiam distintas distribuições, referentes a diferentes problemas, de forma satisfatória.

Como efeito colateral, os *kernels* morfológicos formam sub-áreas, aumentando a complexidade de superfície separadora. Isso é um artefato de treinamento que diminui a capacidade de extrapolação. Também é possível notar que várias dessas sub-áreas foram criadas em regiões as quais deveriam pertencer à classe oposta. A razão do surgimento das referidas sub-áreas e do excesso de fragmentação se deve à aleatoriedade inicial das ligações sinápticas.

Nos *kernels* morfológicos, as ligações sinápticas entre os neurônios da camada de entrada e da camada escondida são aleatórios. Essa aleatoriedade atrapalha a Morfologia Matemática contida nos *morfológicos* dos neurônios da camada escondida.

Em processamento digital de imagem, as operações morfológicas têm o mesmo processo básico de deslocamento de máscaras sobre a imagem original. Na Morfologia Matemática, há uma entidade chamada de elemento estruturante, usada na para maximizar e minimizar a região do(s) objetos(s), durante a dilatação e erosão, respectivamente. Na imagem erodida, o objeto alvo é "murchado". Na imagem dilatada, o objeto alvo é dilatado, como o próprio nome sugere.

Soluções para imagens biomédicas, por exemplo, referentes à detecção de óvulos de Aedes Aegypti (transmissor da dengue) e *Schistosoma* (transmissor da *esquitossomose* - barriga d'água) apresentam elementos estruturantes em forma de disco já que o objetivo é detectar formas arredondadas (ovaladas). Nesse tipo de aplicação não são explorados os coeficientes da matriz bidimensional do elemento estruturante que não se encaixem na forma do disco (fora do óvulo). Quanto na operação de erosão quanto na dilatação, objetos que "casem"padrão com o elemento estruturante de um óvulo são contabilizados. Na imagem erodida, os objetos reconhecidos como óvulos são "murchados". Na imagem dilatada, os objetos reconhecidos como óvulos são dilatação.´

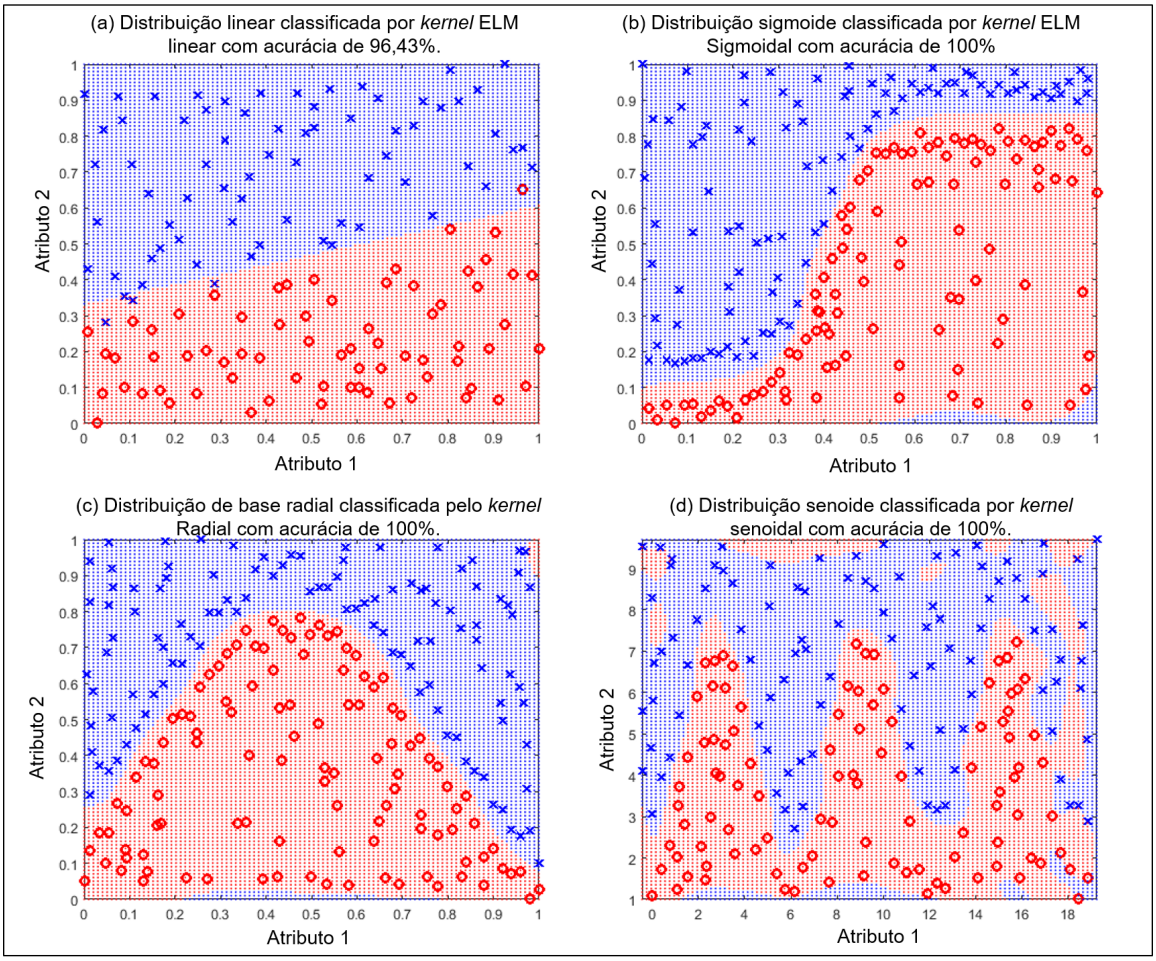


Figura 9.1: Atuações bem-sucedidas dos *kernels* compatíveis com os conjuntos de dados.

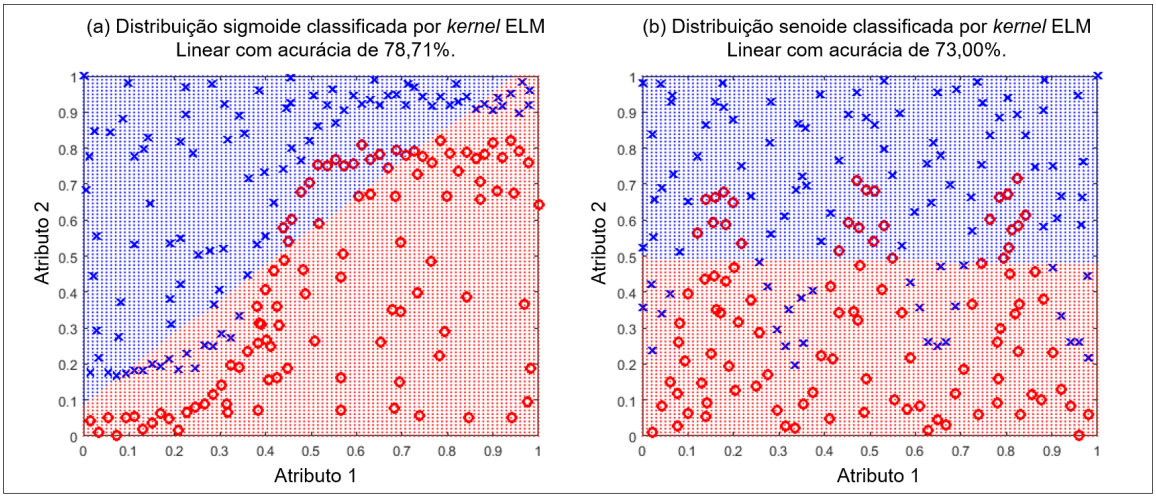


Figura 9.2: Atuações malsucedidas do *kernel* Linear em conjuntos de dados não-linearmente separáveis.

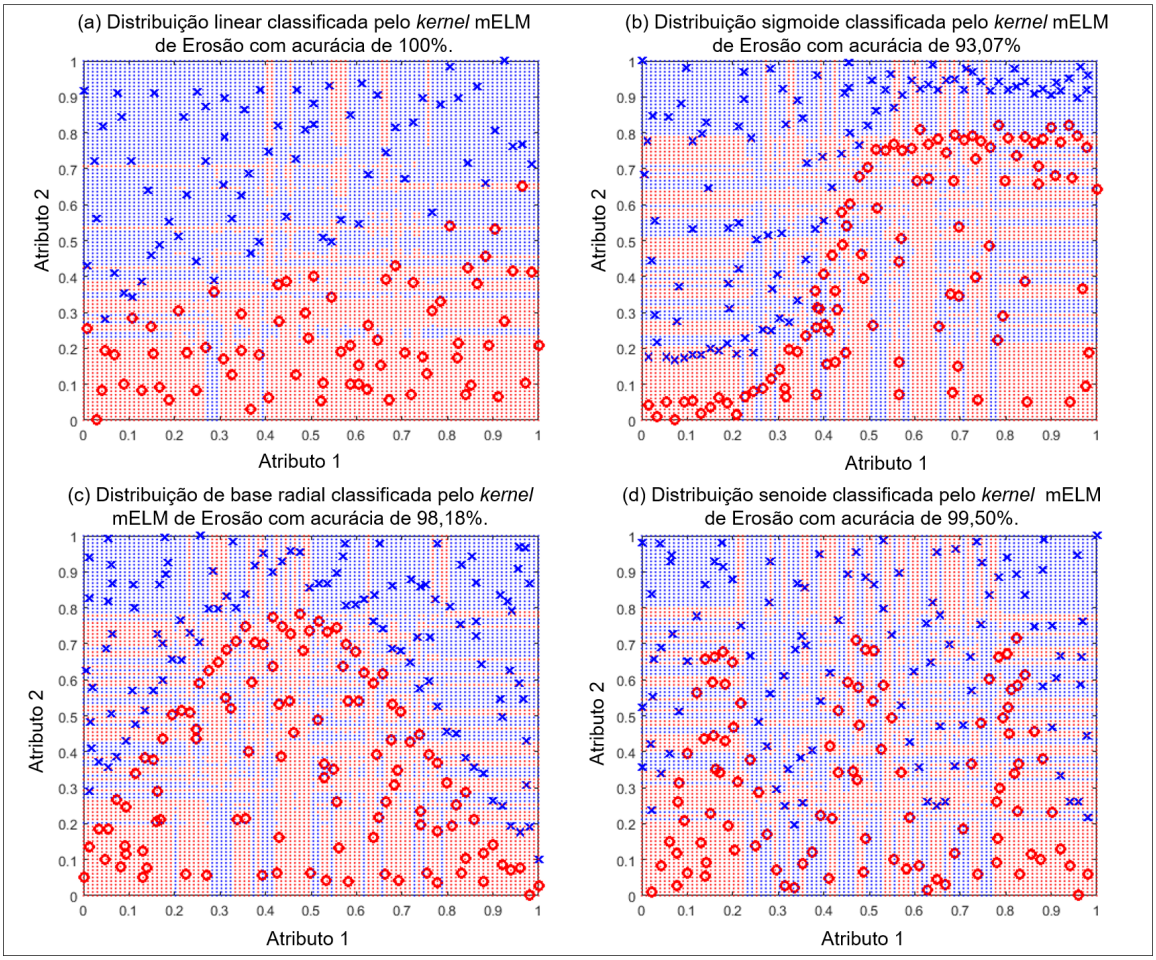


Figura 9.3: Atuações bem-sucedidas do mELM *kernel* Erosão em diversos conjuntos de dados.

9.3 Redução da Dimensionalidade

As redes neurais profundas são frequentemente aplicadas a reconhecimento de padrões em imagens digitais. Mas devido às suas excelentes acurácias, o aprendizado profundo tem sido aplicado em uma variedade de tarefas. Como efeito colateral, as técnicas de aprendizado profundo de última geração dependem de milhões de parâmetros ajustáveis (treináveis). Não há meios claros de determinar quais parâmetros influenciam o limite de decisão em relação às classes-alvo da aplicação. Consequentemente, torna-se inviável formular uma explicação sobre as decisões da rede neural.

A redução da dimensionalidade dos parâmetros ajustáveis (treináveis) da rede neural é fundamental de modo que ela própria conseguir explicar seus padrões captados durante a sua etapa de treinamento. Ao explicar seus padrões de generalização, a rede neural proveria condições prévias de um profissional especialista adotar ou refutar a decisão tomada de forma automatizada.

Na inteligência artificial auto-explicável autoral, a redução da dimensionalidade é realizada de maneira preventiva. Essa redução ocorre mesmo antes da otimização dos parâmetros treináveis. A redução da dimensionalidade é aplicada através de testes estatísticos para auditar a correspondência entre os atributos de entrada e suas respectivas classes. É importante ressaltar que a determinação da classe da amostra é conduzida por um profissional especializado ou por meio de um instrumento devidamente delineado. A inteligência artificial auto-explicável autoral visa garantir não apenas a eficácia da redução da dimensionalidade, mas também a validade e confiabilidade da atribuição de classes às amostras, destacando assim a importância da expertise humana ou de um método claramente definido nesse processo.

Os testes de hipótese representam análises estatísticas aplicadas de modo a comparar dois conjuntos de dados. O objetivo é determinar se existe correlação entre esses conjuntos ou se a hipótese nula é válida. No contexto da inteligência artificial auto-explicável desenvolvida, o primeiro conjunto de dados é dado pelos atributos de entrada de todas as amostras. Esse conjunto é bidimensional, sendo a primeira dimensão composta pelos atributos e a segunda dimensão pela quantidade total de amostras. Já o segundo conjunto de dados utilizado no teste de hipótese corresponde às classes de todas as amostras. Este conjunto é unidimensional, onde cada posição do vetor representa a classe correspondente à sua respectiva amostra. Essa abordagem proporciona uma estrutura clara e eficaz para a análise estatística, possibilitando a avaliação de correlações entre atributos e classes no contexto da inteligência artificial auto-explicável.

São utilizados dois tipos de testes de hipótese: o paramétrico, por meio do teste *Student's t-test*, e o não-paramétrico, através do teste de *Wilcoxon*. O teste paramétrico, *Student's t-test*, demonstra robustez especialmente em amostras de grande dimensionalidade. Em contrapartida, o teste de hipótese não paramétrico é eficaz quando os atributos não se apresentam como variáveis contínuas. Na abordagem da inteligência artificial auto-explicável autoral, caso a hipótese nula seja válida tanto no teste paramétrico quanto no não-paramétrico, o respectivo atributo de entrada é removido do repositório de dados. Dessa maneira, permanecem apenas os atributos de entrada que exercem uma influência direta sobre o valor da classe associada à sua própria amostra.

Essa estratégia tem como objetivo reduzir a dimensionalidade da aplicação. Isso resulta em uma diminuição significativa no número de cálculos durante a fase de aprendizado. O referido feito não apenas reduz o tempo necessário para o treinamento, mas também aprimora a interpretação dos dados. Essa abordagem se destaca como uma prática benéfica em termos de eficiência e interpretabilidade.

9.4 Criação das Ligações Sinápticas

Um desafio operacional das redes neurais clássicas consiste em investigar a dependência de sua aleatoriedade inicial. Em termos didáticos, as conexões entre os neurônios da rede neural são estabelecidas de maneira randômica a partir de uma semente. Diversos web-sites, como indicado no presente *link*, desempenham o papel de gerar valores pseudoaleatórios com base em uma semente. Essa abordagem é amplamente empregada pelas redes neurais, onde as conexões sinápticas entre os neurônios artificiais são inicializadas de forma aleatória. Ao longo do treinamento, essas sinapses sinápticas são ajustadas em direção ao objetivo da aplicação, buscando minimizar a discrepância entre as respostas desejadas e as respostas efetivamente obtidas.

Caso a semente geradora de números aleatórios for adequada a uma aplicação específica, a rede neural alcançará resultados excelentes. Porém se a semente for inadequada, os resultados da rede neural podem ser desastrosos. É crucial notar que a obtenção de resultados satisfatórios em uma aplicação não implica necessariamente sucesso ao utilizar a mesma semente em outro contexto.

Determinar a semente ótima para a geração de números aleatórios é um desafio que pode demandar um tempo considerável. A busca pela semente ideal é, portanto, computacionalmente inviável quando realizada de maneira exaustiva. Esse aspecto destaca a complexidade associada à escolha da semente e reforça a importância de uma abordagem estratégica ao lidar com a aleatoriedade nas redes neurais.

Uma abordagem frequentemente adotada envolve múltiplas execuções de uma mesma arquitetura de rede neural, cada uma utilizando uma distinta semente geradora de números aleatórios iniciais. Esse procedimento permite avaliar se as variações nos pesos sinápticos iniciais têm um impacto significativo nos resultados. Ao concluir os experimentos, um desvio padrão elevado indicaria uma dispersão notável nos resultados. Nesse caso, a rede neural estaria suscetível à influência e dependência das condições iniciais randômicas. Essa abordagem fornece *insights* valiosos sobre a sensibilidade do modelo a diferentes configurações iniciais, contribuindo para uma compreensão mais robusta e abrangente do comportamento da rede neural.

A partir da variação da semente geradora de aleatórios, a presença de um desvio padrão elevado pode ter um impacto abrupto tanto na degradação quanto na melhoria da acurácia de uma rede neural. A aplicação de uma solução com essa característica na prática clínica, por exemplo, torna-se arriscada. Durante o uso em cenários clínicos reais, até mesmo uma leve mudança no perfil das pacientes poderia resultar em uma degradação súbita dos resultados fornecidos pela solução computacional. A rede neural só funcionaria para casos estatisticamente equivalentes àqueles apresentados durante a fase de treinamento. Enfatiza-se que muitos casos vistos na prática clínica não se encaixam, com precisão, nas imagens e descrições clássicas [19][20].

Por exemplo, considerando duas pacientes, A e B, ambas diagnosticadas com lesão de mama confirmada por biópsia. Porém é possível que ambas apresentem características completamente distintas entre si. Uma rede neural com elevado desvio padrão poderia identificar apenas as lesões de mama no perfil da paciente A, deixando de abranger o perfil da paciente B. Adaptar a rede neural a amostras estatisticamente diferentes daquelas apresentadas durante o treinamento torna-se uma tarefa desafiadora. Esse desafio ressalta a importância de considerar a robustez e a capacidade de generalização das redes neurais em contextos clínicos diversos.

Na inteligência artificial auto-explicável autoral, as ligações sinápticas não são determinadas de forma aleatória. As ligações sinápticas correspondem a descritores dos atributos de entrada. Ao considerar a primeira iteração, é criado um neurônio na camada escondida de modo a representar

uma amostra sintética da classe-alvo. Esse neurônio sintetiza as características dos atributos da classe-alvo. Em termos técnicos, cada ligação sináptica sintetiza um dos atributos das amostras pertencentes à classe alvo. Se o atributo for de valores flutuantes, a ligação sináptica atrelada ao dado atributo será a média aritmética simples. Se o atributo for dado em categorias, a ligação sináptica atrelada ao dado atributo será a moda. Não há sentido em se criar médias aritméticas simples. Considere o algoritmo judiciário, não faria sentido criar um réu sintético com 25% de antecedentes criminais. Ou o réu sintético tem antecedentes criminais ou não tem. De maneira análoga, é(são) criado(s) neurônio(s) na camada escondida de modo a representar amostras sintética da(s) contra-classe(s). Após a criação da camada escondida, há o aprendizado da

Siga as instruções:

1. Faça o *Download* do *scritpt xeai.py* e do *dataset* no presente link (pasta Cap. 8).

2. Clique com o botão direito do mouse sobre o arquivo compactado de nome *dataset*. Então escolha a opção "**Extract here**"conforme ilustra a Fig. 11.2.

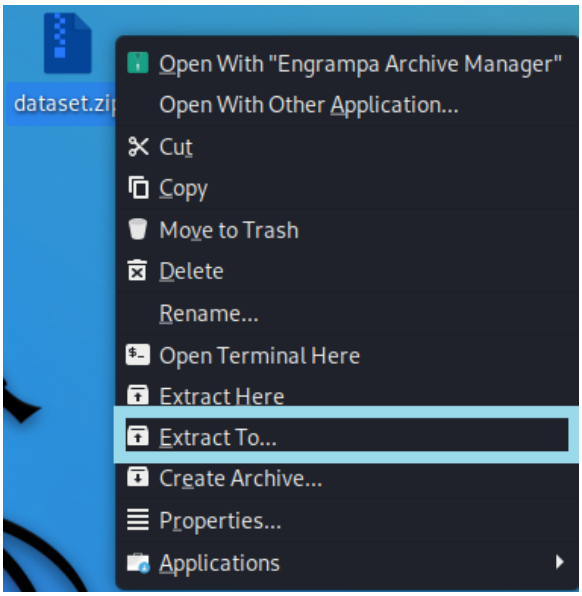


Figura 9.4: Necessidade de extração do arquivo compactado visando a consulta automatizada aos antivírus comerciais.

3 Repositório didático visando reconhecimento de padrão:

- Não é do escopo do capítulo a criação da base de dados. Parte-se do princípio que o repositório de aprendizado já foi previamente confeccionado por terceiros. A criação de repositório e as metodologias visando extrair características dos *malware* foram discutidas no capítulo 7.
- No caminho (**dataset/classification/diabetes_train**), é possível notar a estrutura do repositório conforme exibe a Fig. 11.7. Essa estrutura segue a metodologia dos inventores da ELM [13].
  - **Primeira coluna**: 1; a amostra (linha) pertence à classe. 0; a amostra (linha) pertence à contra-classe.
  - **Demais colunas**: atributos (neurônios) de entrada referentes à extração de características da aplicação alvo. Na presente base, há 8 neurônios de entrada. Por exemplo, na primeira amostra (linha), o primeiro neurônio tem valor 0.11764700.
- Ao final do aprendizado (treinamento), a rede neural ELM terá capacidade de generalização. Portanto a ELM classificará a amostra inédita (não apresentada ao treino) como pertencente à classe (1.0) ou contra-classe (0.0).

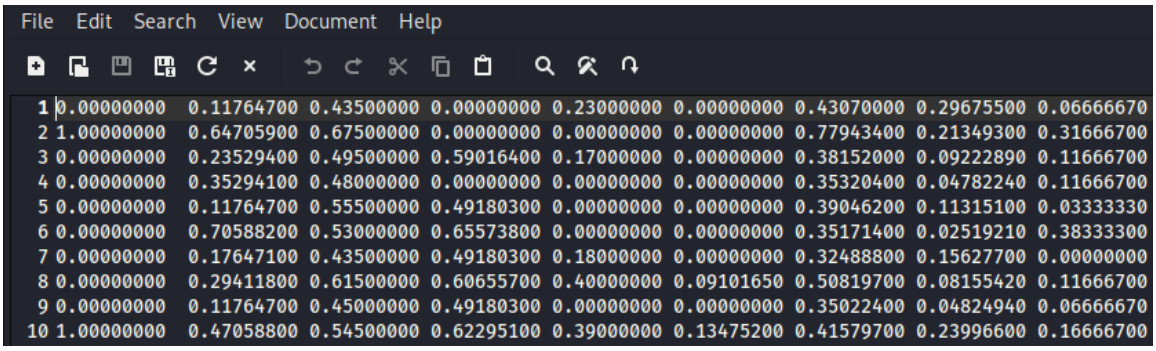


Figura 9.5: Estrutura de um repositório quanto à reconhecimento de padrão visando o uso da ELM como classificador.

4. Repositório autoral visando reconhecimento de padrão:
   - No caminho (**dataset/classification/Antivirus_Dataset_PE32_Citadel_mELM_format.csv**), é possível notar a estrutura do repositório conforme exibe a Fig. 9.6. Essa estrutura permite que o arquivo seja executado por editores de planilhas eletrônica tais quais o MS. Excel e o Google Planilhas. O ";" separa os valores nas referidas planilhas eletrônicas.
     - **Primeira coluna**: O nome do aplicativo. Os aplicativos e suas respectivas análises brutas estão na repositório Citadel; https://github.com/DejavuForensics/Citadel.
     - **Segunda coluna**: 1; a amostra (linha) pertence à classe (*malware* Citadel). 0; a amostra (linha) pertence à contra-classe (aplicativo sério).
     - **Demais colunas**: atributos (neurônios) de entrada referentes à extração de características da aplicação alvo. Na presente base, há 430 neurônios de entrada para cada amostra. Por exemplo, na primeira amostra (linha), o primeiro neurônio tem valor 0.
   - Ao final do aprendizado (treinamento), a rede neural ELM terá capacidade de generalização. Portanto a ELM classificará a amostra inédita (não apresentada ao treino) como pertencente à classe (1.0) ou contra-classe (0.0).

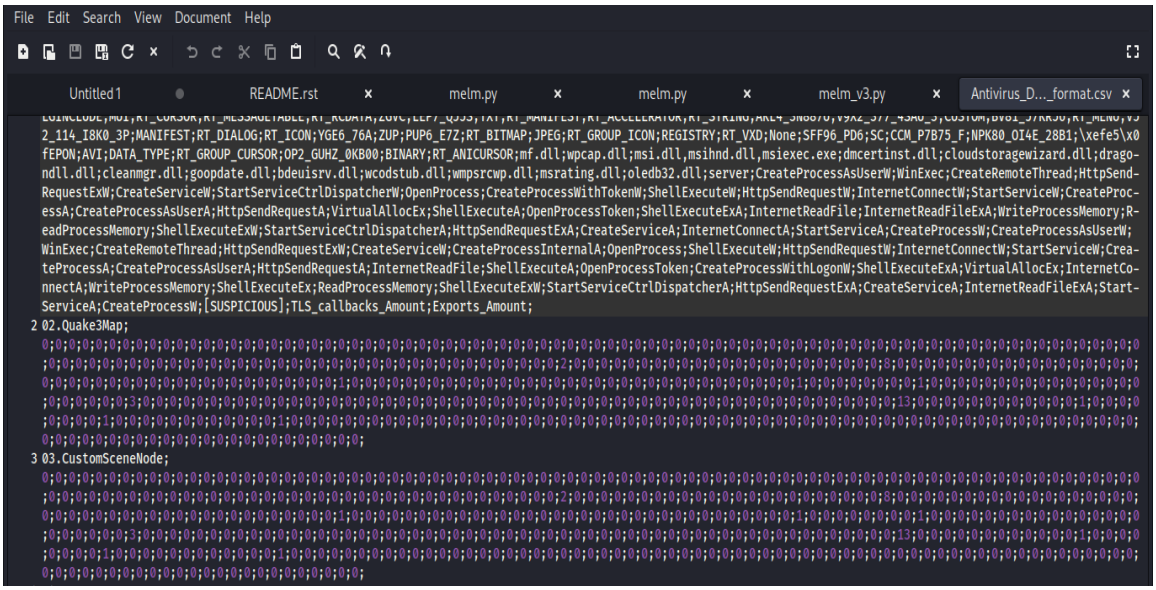


Figura 9.6: Estrutura de um repositório quanto à reconhecimento de padrão visando o uso da ELM como classificador.

5. Parâmetros da rede neural extrema:
   - -tr: repositório de aprendizado estatístico reversado à fase de treino.
   - -ts: repositório de aprendizado estatístico reversado à fase de teste.
   - -tall: repositório de aprendizado estatístico total. Inclui-se a fase de treinamento e teste.
   - -ty:
     - 1: classificação (reconhecimento de padrão).
     - 0: regressão (predição: previsão com rigor científico-metodológico).

6. No console, use a rede neural extrema em um repositório didático. Segue um exemplo:

```
python melm.py -tr dataset/classification/diabetes_train
-ts dataset/classification/diabetes_test -ty 1 -v
```

   - No console, use a rede neural extrema em um repositório real. Segue um exemplo:

```
python melm.py -tall dataset/classification
/Antivirus_Dataset_PE32_Citadel_mELM_format.csv  -ty 1 -v
```

9.5 Predição: *Kernels* Morfológicos Autorais

Redes neurais desempenham um papel fundamental na predição ao estimar valores fracionários com rigor científico-metodológico. Durante o treinamento, a rede é exposta a uma série histórica temporal, como a cotação diária do petróleo, e ajusta conexões entre neurônios, atribuindo pesos ponderados a eventos periódicos. Isso permite identificar influências de acontecimentos diários aparentemente irrelevantes na cotação do petróleo, algo difícil para investidores humanos. Além disso, as redes neurais desempenham um papel vital na detecção e prevenção de desastres naturais, alertando sobre catástrofes iminentes, contribuindo assim para a construção de metamodelos em cidades inteligentes.

A avaliação do reconhecimento de padrões difere consideravelmente da predição. No contexto do reconhecimento de padrões, a métrica primordial é a acurácia. Quanto mais próxima de 100%, melhor tende a ser a performance da rede neural nessa aplicação específica. Já no cenário da predição, a métrica crucial é o erro médio quadrático. Quanto mais próximo de 0, melhor é o desempenho da rede neural. Isso indica que a diferença entre as saídas desejadas e as saídas obtidas é minimizada, proporcionando uma predição otimizada e alinhada com os objetivos do modelo.

Siga as instruções:

1. Repositório visando predição:
   - No caminho (**dataset/regression/sinc_train**), é possível notar a estrutura do repositório conforme exibe a Fig. 11.8.
     - **Primeira coluna**: valor flutuante a ser estimado.
     - **Segunda coluna**: atributo (neurônio) de entrada. Trata-se de uma base de dados didática e hipotética. Um aplicação real poderia ter centenas de neurônios de entrada.
   - Ao final do aprendizado (treinamento), a rede neural ELM terá capacidade de predição. Portanto a ELM estimará um valor flutuante a partir de um dado neurônio de entrada.
   - Dentro do pacote "mELM", quanto à predição, encontra-se exclusivamente a base de dados didática, alinhada à metodologia desenvolvida pelos criadores da ELM [13]. Porém o arcabouço autoral permite a experimentação de bases de dados em aplicações práticas, seguindo a estrutura delineada no repositório ilustrado na Figura 11.8. Vale ressaltar que, para problemas de predição, a segunda coluna deve ser um valor flutuante, diferentemente de uma classe, que é utilizada em casos de reconhecimento de padrões.
   - É possível utilizar bases de dados provenientes do Kaggle https://www.kaggle.com/ ou da UCI https://archive.ics.uci.edu/datasets. Nos respectivos portais, busque por repositórios voltados para predição e análise de séries temporais.

2. No console, use a rede neural extrema. Segue um exemplo:

```
python melm.py -tr dataset/regression/sinc_train
-ts dataset/regression/sinc_test -ty 0 -nh 100 -af dilation -v
```

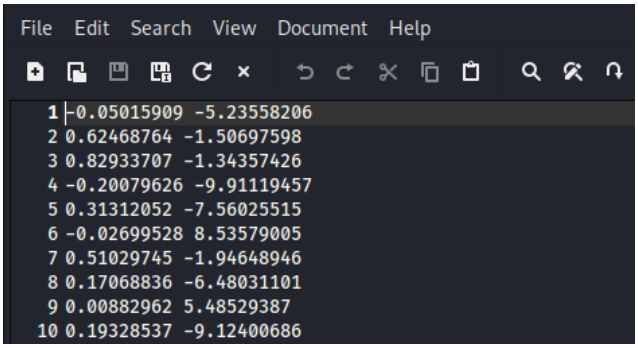
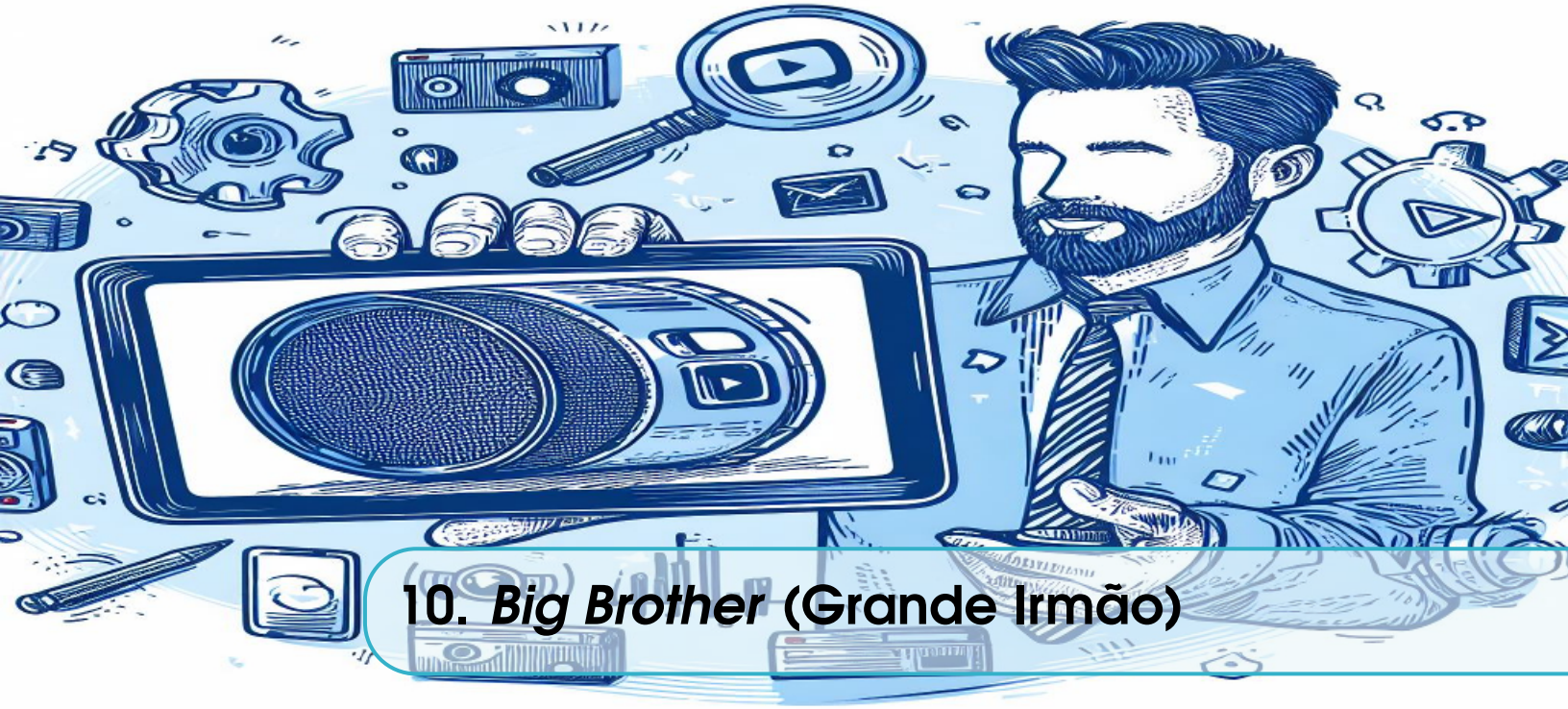


Figura 9.7: Estrutura de um repositório quanto à predição visando o uso da ELM como regressor.



10. *Big Brother* (Grande Irmão)

10.1 Introdução

Em 1948, um escritor inglês, George Orwell (1903-1950) produziu um livro chamado 1984. O livro mostra uma sociedade em que a elite tecnológica ultra-eficiente ("O Grande Irmão") controlava tudo. O Grande Irmão espionava a vida íntima da população e transformava os dissidentes em não-existentes. Em "1984", quase todos os ambientes contavam com um dispositivo chamado "teletela". No romance, as teletelas exibem a propaganda oficial e simultaneamente monitoram os cidadãos.

No fim dos anos 40, quando o escritor inglês George Orwell criou o personagem "O Grande Irmão", no livro 1984, ninguém imaginava que pudesse existir na realidade um governo capaz de monitorar todo mundo com o uso de câmeras. Não só isso está acontecendo hoje, como também há o Grande Tio. As grandes corporações coletam as imagens e as conversas da população com objetivos de marketing e passam a vender as informações entre elas. Em acréscimo, os grandes provedores de dados são capazes de influenciar a vida dos cidadãos ao impulsionar tendências (*trends*) nas redes sociais.

Em tempos contemporâneos, os proprietários da Indústria 4.0 (Quarta Revolução Industrial) podem controlar a sociedade não somente através da economia, mas a mente e os sentimentos pessoais. Absolutamente, todas as ações das pessoas estarão passíveis de serem impulsionadas pelos provedores de dados.

- Necessidades fisiológicas.
- Carências sentimentais.
- Distúrbios e transtornos psiquiátricos (depressão, estresse).
- Ódio, raiva, rancor, mágoa.

10.2 Experimento Social Automatizado

Devido à massificação da rede mundial de computadores e suas redes sociais, várias aplicações automatizadas tentam se fazer passar por uma multidão instantânea de modo a criar tendências

(*trends*) nacionais ou até mesmo mundiais. A expectativa é ter um pode análogo aos grandes provedores de dados e, consequentemente, ter a capacidade de manipular a vida das pessoas.

No presente experimento, será empregada a biblioteca em python *PyAutoGUI* de modo a automatizar os cliques e pressionamento de teclas. Em acréscimo, a bilioteca *Auto Tor* será empregada de modo que seja usados endereços IPs diversos com o objetivo de simular uma multidão de pessoas de várias regiões geográficas.

10.2.1 Auto Tor

Siga as instruções:

1. Instalação e manuseio do Auto Tor de modo a usar endereços IPs pouco rastreáveis. Clique no seguinte link `https://github.com/FDX100/Auto_Tor_IP_changer`. Em seguida, faça o *Download* conforme mostra a Fig. 10.1.

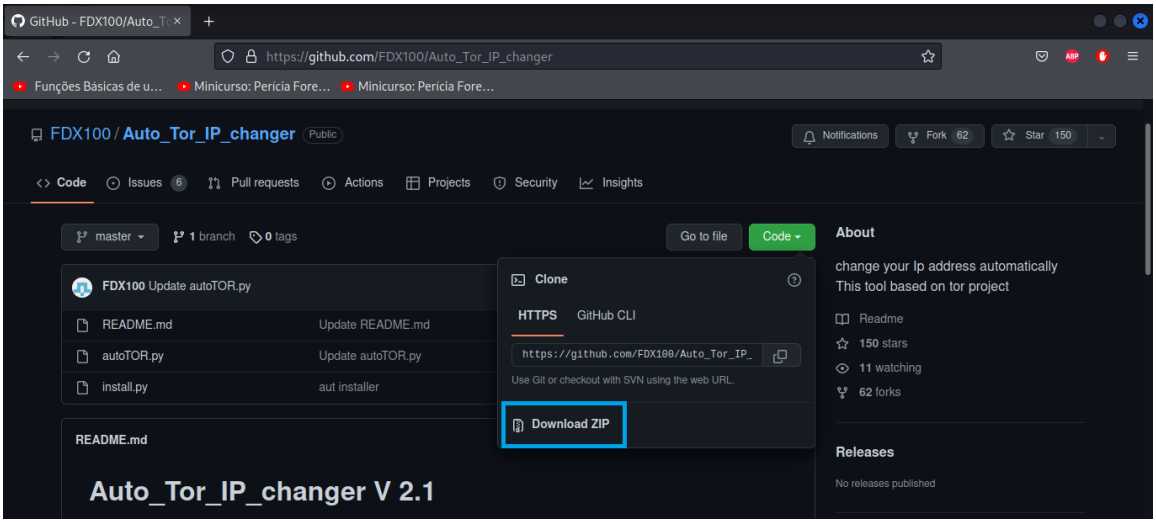


Figura 10.1: Repositório do Auto Tor.

2 No seu computador, na pasta ***"Downloads/Auto_Tor_IP_changer-master"***, clique com o lado direito do mouse e escolha ***"Extract Here"***. Abra o arquivo autoTOR.py. Como exibe a Fig. 10.2, faça a edição na linha 70 e 71 e substitua por x = 1 e lin = 1, respectivamente.

```
61
62 os.system("service tor start")
63
64
65
66
67 time.sleep(3)
68 print("\033[1;32;40m change your  SOCKES to 127.0.0.1:9050 \n")
69 os.system("service tor start")
70 x = 1
71 lin = 1
72 #x = input("[+] time to change Ip in Sec [type=60] >> ")
73 #lin = input("[+] how many time do you want to change your ip [type=1000]for infinte ip change type [0] >>")
74 if int(lin) ==int(0):
```

Figura 10.2: Customização do *scrip* autoTor.py.

3 Abra o console na pasta clonada do Auto Tor conforme mostra a Fig. 10.3.

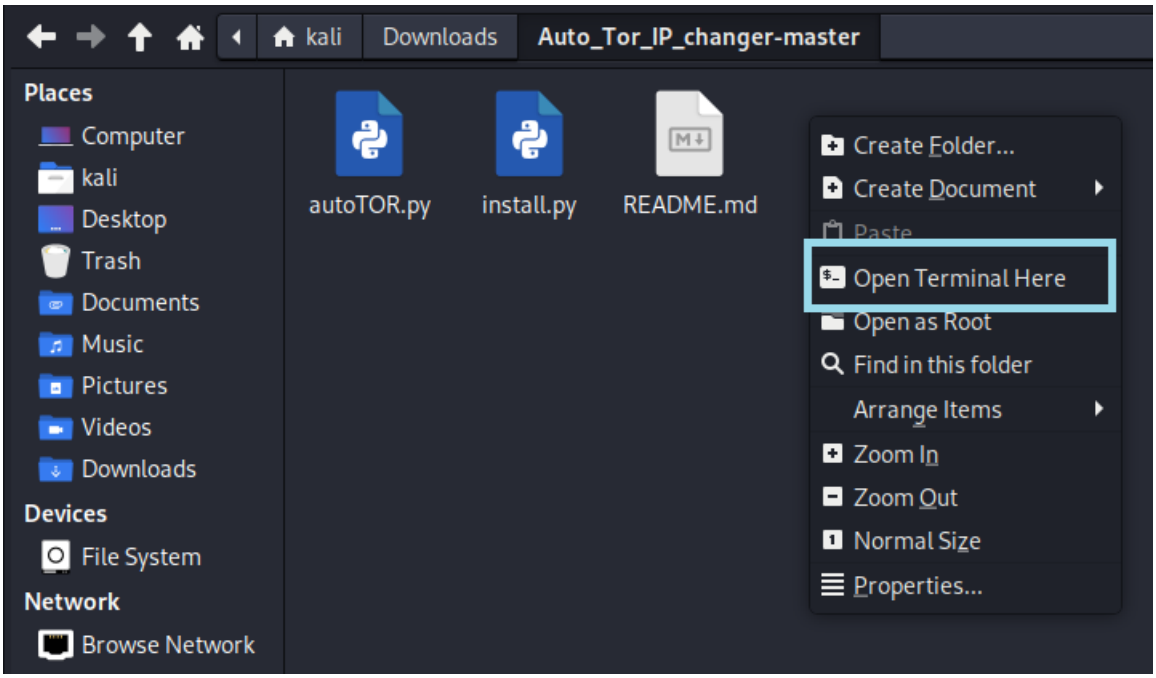


Figura 10.3: Abertura do console na pasta clonada anteriormente.

- No console, entre em modo de administrador. Por padrão, o *login* é ***kali***, a senha também é ***kali***.

```
sudo su
```

- No console, instale o tor.

```
tor -V
```

- No console, instale o Autor Tor.

```
python install.py
```

- No console, verifique se a instalação do Auto Tor ocorreu corretamente.

```
aut
```

10.2.2 PyAutoGUI

Siga as instruções:

1. Instalação do PyAutoGUI de modo a usar o mouse e o teclado de forma automatizada:
   - No console, instale o PyAutoGUI.

```
pip install pyautogui
```

2. Exemplos de uso do PyAutoGUI. No console, digite python:
   - No console, digite python e adicione a biblioteca PyAutoGUI.

```
python
import pyautogui
```

   - Mova o mouse para as coordenadas XY e, em seguida, há um clique automatizado.

```
1    pyautogui.click(100, 200)   # Clique do mouse.
```

   - A função *click()* simula um clique de mouse por 0,1 segundos. Logo algumas plataformas como o Youtube têm a capacidade de detectar o referido clique automatizado. Opcionalmente, pode-se utilizar o argumento interval de modo a variar o tempo de clique. No seguinte exemplo, o tempo de clique custaria 0,25 segundos.

```
1    pyautogui.click(100, 200, interval=0.25)  # Clique por 0,25 seg.
```

   - Mova o mouse 50 pixels à direita de sua posição atual.

```
1    pyautogui.move(50, 0)   # Movimento do mouse
2                            # em relação à posição atual.
```

   - Mova o mouse para as coordenadas XY.

```
1    pyautogui.move(50, 0)   # Movimento do mouse
2                            # para as coordenadas XY.
```

   - Clique duplo com o mouse.

```
1    pyautogui.doubleClick() # Clique duplo.
```

   - A função *press()* é um empacotador para as funções *keyDown()* e *keyUp()*, as quais simulam pressionar uma tecla pressionada e em seguida quando é solta. As teclas podem ser pressionadas simultaneamente. Por exemplo, a tecla *shift* pode ser pressionada simultaneamente com a tecla da seta à esquerda por três vezes:

```
1    pyautogui.keyDown('shift')  # Pressiona o shift.
2    pyautogui.press('left')     # Pressiona a seta esquerda.
3    pyautogui.press('left')     # Pressiona a seta esquerda.
4    pyautogui.press('left')     # Pressiona a seta esquerda.
5    pyautogui.keyUp('shift')    # Solta o shift.
```

10.2.3 Localização do mouse para clicks automatizados

Siga as instruções:

1. O comando *xdotool* é capaz de detectar as coordenadas do mouse:
   - No console, localize as coordenadas do mouse.

```
xdotool getmouselocation
```

10.2.4 Clicks automatizados no Youtube

Siga as instruções:

1. No navegador *Firefox*, instale a extensão ***WebRTC*** no seguinte link: https://addons.mozilla.org/pt-BR/firefox/addon/happy-bonobo-disable-webrtc/
   - O ***WebRTC*** "vaza" os endereços IP reais por trás de uma VPN. Caso o ***WebRTC*** não seja desabilitado, o seu navegador saberá o seu real endereço e, portanto, sua real geolocalização. Em síntese, instale o ***WebRTC***, conforme Fig. 10.4.

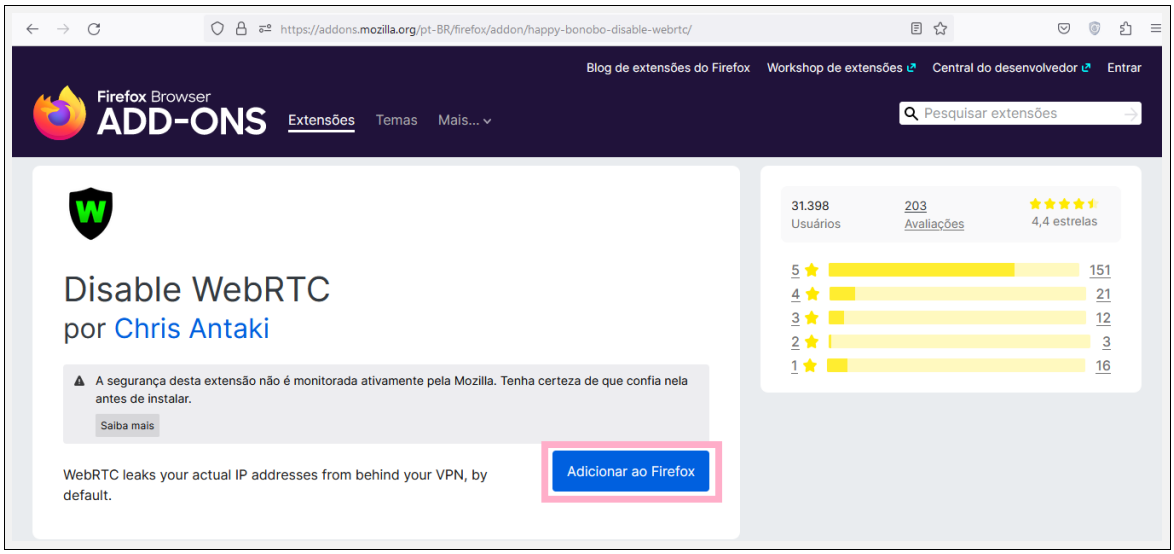


Figura 10.4: Extensão *WebRTC* no navegador Firefox.

- Clique no botão visualizado pela Fig. 10.5. Quando em vermelho, a extensão *WebRTC* está ativada. Quando em verde, a extensão *WebRTC*está desativada.

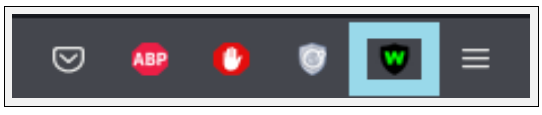


Figura 10.5: Quando em vermelho, a extensão *WebRTC* está ativada. Quando em verde, a extensão *WebRTC*está desativada.

2. No Youtube, favorite 3 (três) vídeos quaisquer do seu agrado. O botão de favorito, no Youtube, está ilustrado na Fig. 10.6.

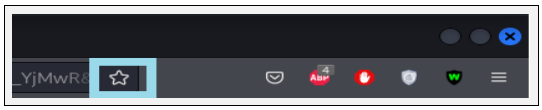


Figura 10.6: No Youtube, favorite 3 (três) vídeos quaisquer do seu agrado.

3 Faça o *Download* do *script acess_youtube.py* responsável pelos cliques automatizados no presente link (pasta Cap. 15).

4 Em seguida, faça as customizações no *script*.

- Entre as linhas 26 e 29 do *script*, estude a posição do botão Recarregar do navegador alvo através das instruções contidas na subseção 15.2.3. A Fig. 10.7 exibe o ícone do botão Recarregar no navegador Firefox.

```
26      def recarregar_pagina():
27          pyautogui.moveTo(100, 100) # Movimento do mouse
28          pyautogui.click()          # Clique do mouse.
```

Figura 10.7: Ícone do botão Recarregar no navegador Firefox.

- Entre as linhas 79 e 81 do *script*, determine quantos vídeos serão clicados automaticamente. No exemplo seguinte, 3(três) vídeos do Youtube serão acessados em *loop*.

```
77      lista = []
78
79      lista.append(4)     #youtube
80      lista.append(4)     #youtube
81      lista.append(4)     #youtube
```

5 Abra o navegador e carregue todos os vídeos desejados do Youtube, desde que a quantidade seja a mesma listada na etapa anterior.

6 No navegador, favorite os vídeos desejados. Customize as posições dos cliques na aba de favoritos através do comando *xdotool* discutido na seção 14.2.3.

- Em seguida, entre as linhas 147 e 151 do *script*,

```
143     #Clique nos favoritos
144         time.sleep(random.randint(2, 3))
145         for index in reversed(range(len(lista))):
146                 if index==0:
147                         clicar_pdf_com_click(100, 135)
148                 elif index==1:
149                         clicar_pdf_com_click(300, 135)
150                 elif index==2:
151                         clicar_pdf_com_click(500, 135)
152                 chavear_paginas()
153                 time.sleep(random.randint(2, 3))
```

7 O Auto Tor funciona no endereço local 127.0.0.1 e na porta 9050. Então é necessária a configuração do navegador Firefox.

- Pesquise por ***Network settings*** nas configurações do Firefox. Na sequência, clique no botão ***Settings...*** como aponta a Fig. 10.8.

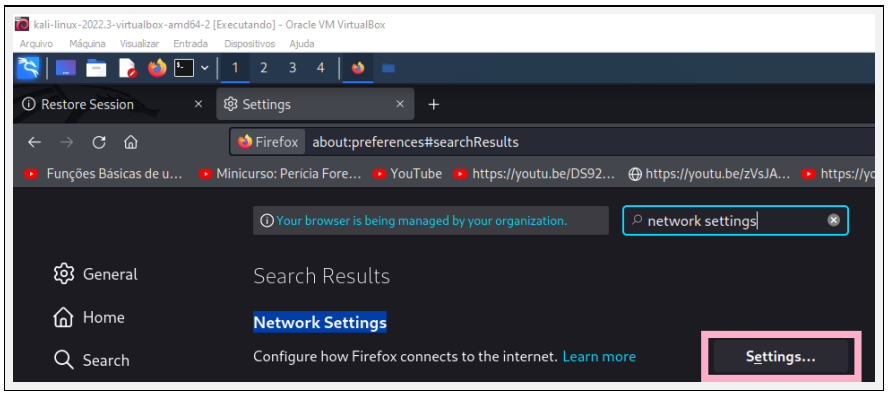


Figura 10.8: Pesquise por ***Network settings*** nas configurações do Firefox. Na sequência, clique no botão ***Settings...***.

- Conforme a Fig. 10.9, escolha a opção ***Manual Proxy Configuration***. Na sequência, o campo ***SOCKS Host*** e ***Port*** devem ser preenchidos com **127.0.0.1** e **9050**, respectivamente. Tratam-se de configurações do Auto Tor.

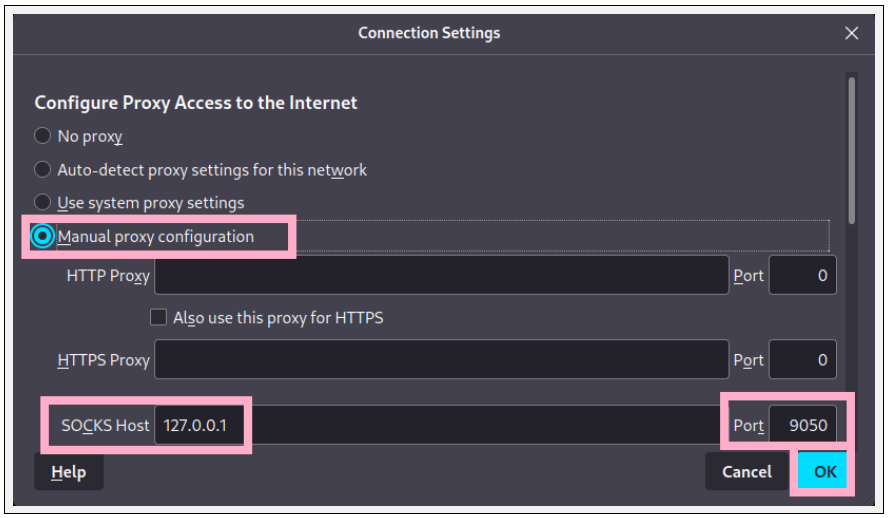


Figura 10.9: Escolha a opção ***Manual Proxy Configuration***. Na sequência, o campo ***SOCKS Host*** e ***Port*** devem ser preenchidos com **127.0.0.1** e **9050**, respectivamente. Tratam-se de configurações do Auto Tor.

8 No console, para acessar os vídeos em *loop*.

```
python acess_youtube.py
```

10.3 Desafio

Acompanhe, ao longo dos dias, o crescimento das visualizações dos vídeos no Youtube. Em caso de retirada dos acessos, faça customização do *script*. Por exemplo, insira tempos de pressionamento randômicos nas teclas, acrescidos de cliques aleatórios no *mouse*. Cabe enfatizar que a plataforma Google consegue reconhecer endereços IPs oriundos do Tor. Em face do exposto, investigue outros provedores de endereços IPs automatizados.

11. Anexo: Criação de Vídeo Tutorial

Rachel Nicole Lacerda Muniz
Currículo *Lattes*: http://lattes.cnpq.br/1892949950394787
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. rachel.nicole@ufpe.br

Dr. Sidney Marlon Lopes de Lima
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. sidney.lima@ufpe.br

11.1 Motivação

Criar vídeos tutoriais para demonstrar suas habilidades pode ser uma boa estratégia na busca por emprego, especialmente em campos que valorizam a demonstração prática de competências. Em um mercado de trabalho cada vez mais competitivo, diferenciar-se é fundamental, e vídeos tutoriais oferecem uma maneira dinâmica e interativa de fazer isso. Os vídeos permitem que potenciais empregadores vejam as habilidades do candidato em ação, indo além das palavras de um currículo para mostrar como ele resolve problemas.

O OBS é uma ferramenta indispensável no arsenal de qualquer pessoa interessada em produção de vídeo e transmissão ao vivo, combinando poderosas funcionalidades com a flexibilidade de um software de código aberto. Seu design intuitivo, juntamente com o suporte de uma comunidade ativa, torna-o uma escolha excelente tanto para iniciantes quanto para profissionais da área.

11.2 Configuração do Idioma no OBS

Siga as instruções:

1. Configuração de idioma. Para mudar o idioma para português (caso esteja em outro idioma).

- Abra o OBS na sua página principal.
- Fig. 11.1: clique em **"configurações"**.
- Fig. 11.2: clique em **"idioma"** e escolha o da sua preferência.

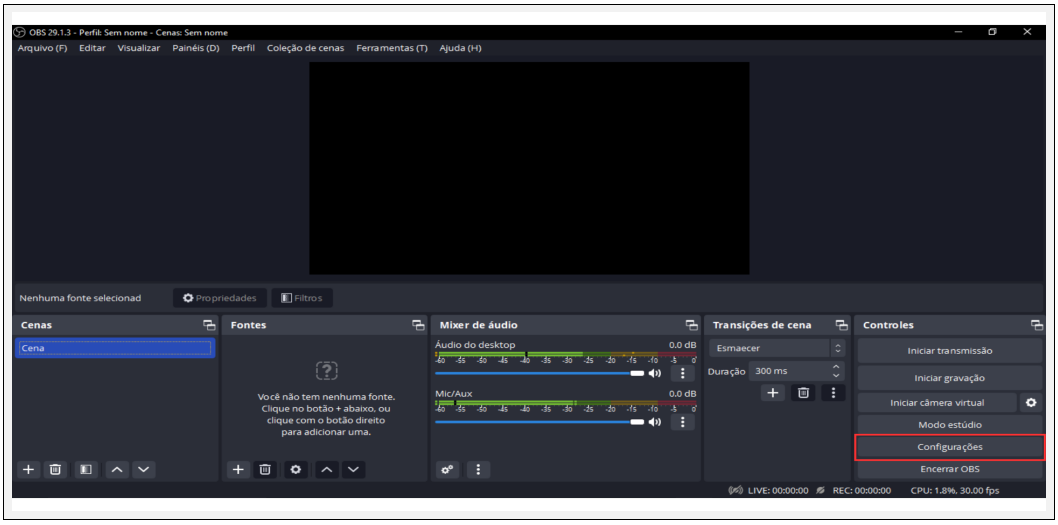


Figura 11.1: Clique em **"configurações"**.

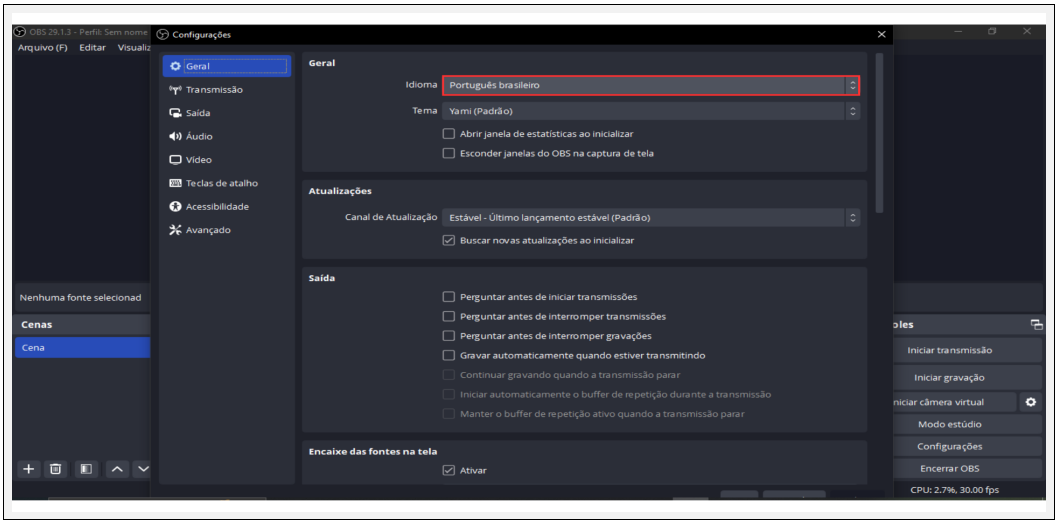


Figura 11.2: Clique em **"idioma"** e escolha o da sua preferência.

11.3 Iniciar uma gravação no OBS

Siga as instruções:

1. Iniciar uma gravação.
   - Fig. 11.3: direcione o mouse para **“Fontes”** e clique no botão direito do mouse ou no botão **“+”**.
   - Fig. 11.4: clique em **“Adicionar”**. Na sequência, clique em **“Captura de Tela”**.
   - Fig. 11.5: nomeie sua fonte como desejar e clique em **“OK”**.
   - Fig. 11.6: deixe o método de captura em **“Automático”**. Na sequência, deixe o monitor como **“Monitor Principal”**. Por fim, clique em **"OK"**.
   - Fig. 11.7: visualização da tela principal.
   - Fig. 11.8: clique em **“Iniciar gravação”**.

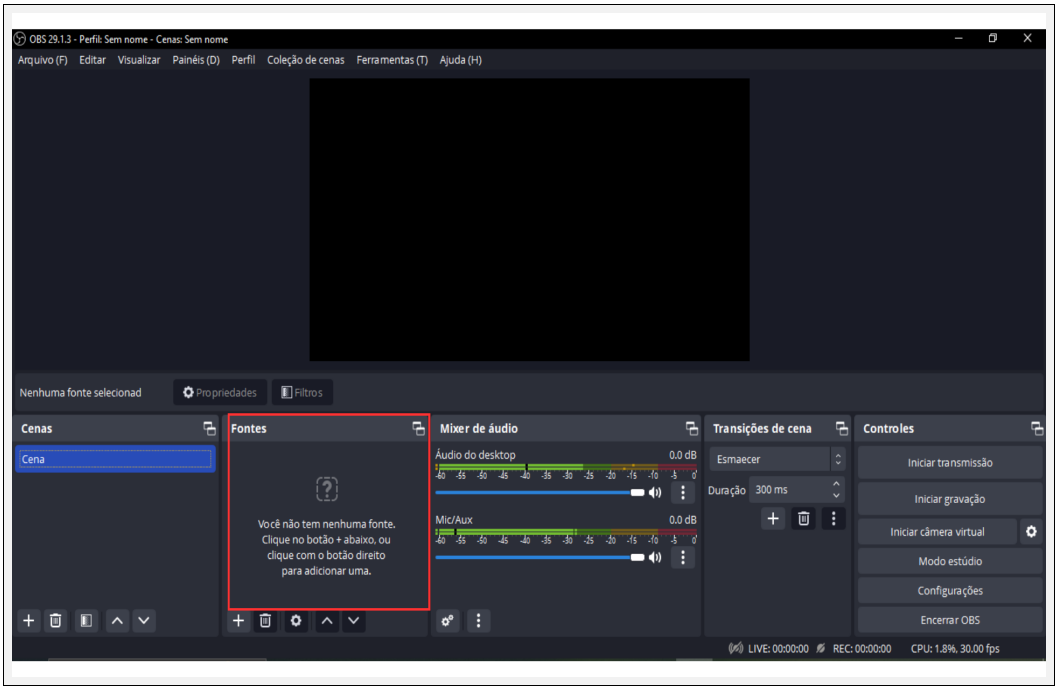


Figura 11.3: Direcione o mouse para **“Fontes”** e clique no botão direito do mouse ou no botão **“+”**.

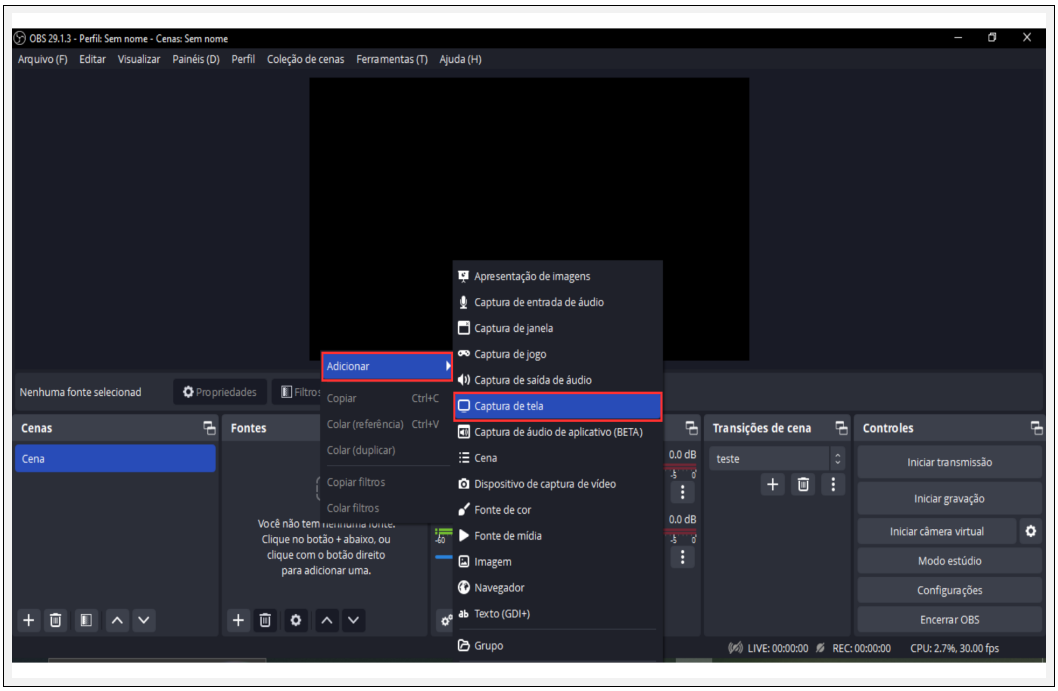


Figura 11.4: Clique em **"Adicionar"**. Na sequência, clique em **"Captura de Tela"**.

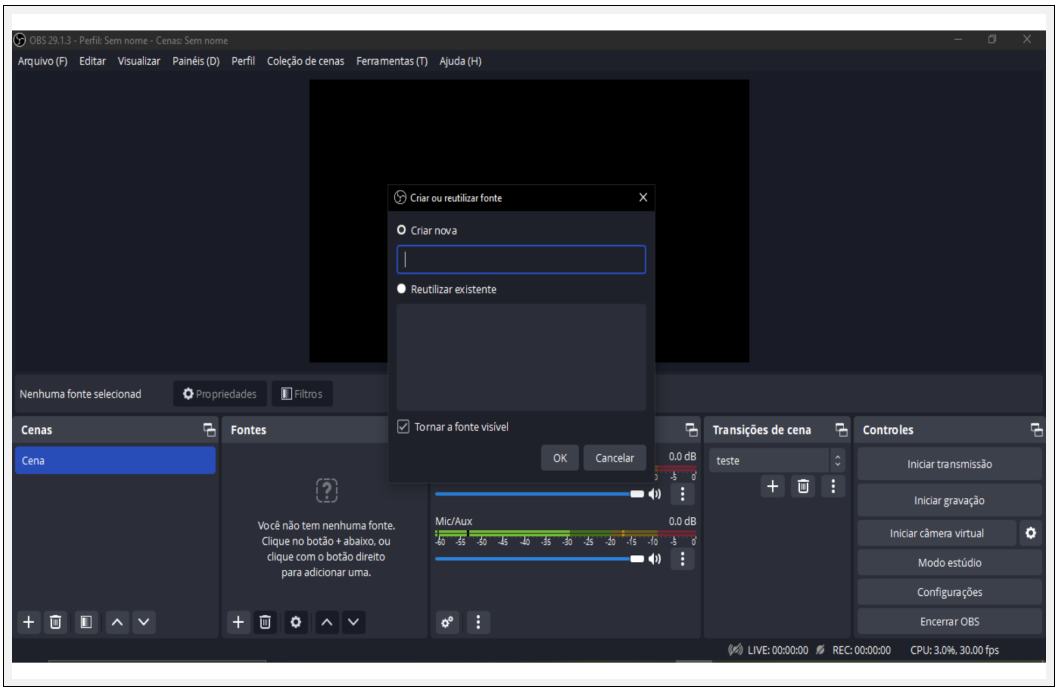


Figura 11.5: Nomeie sua fonte como desejar e clique em **"OK"**.

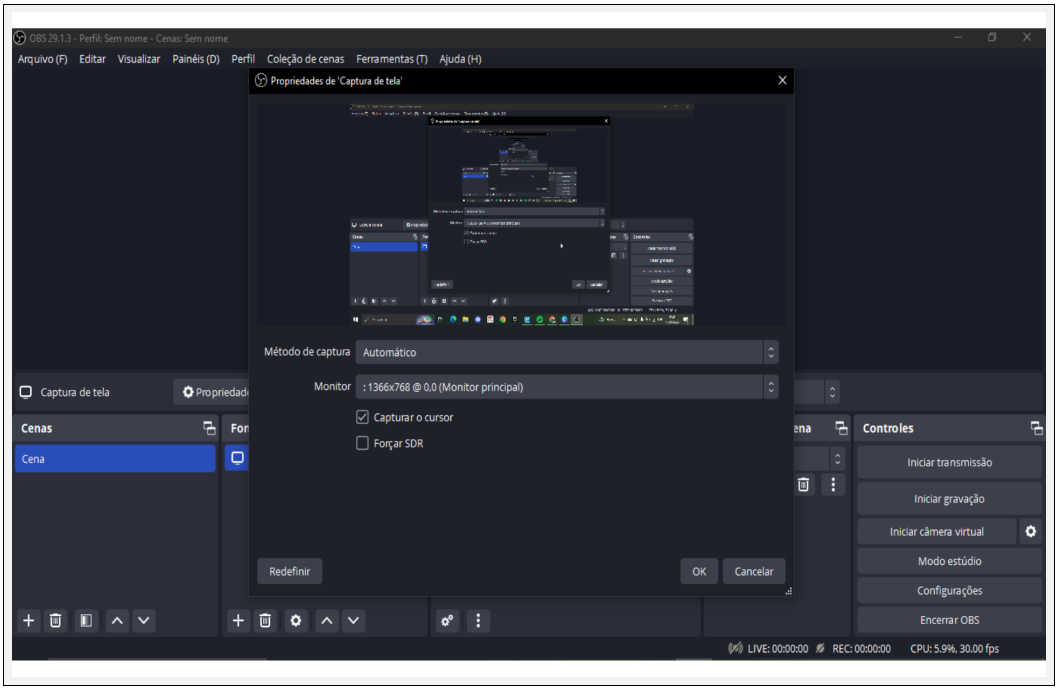


Figura 11.6: deixe o método de captura em **“Automático”**. Na sequência, deixe o monitor como **“Monitor Principal”**. Por fim, clique em **"OK"**.

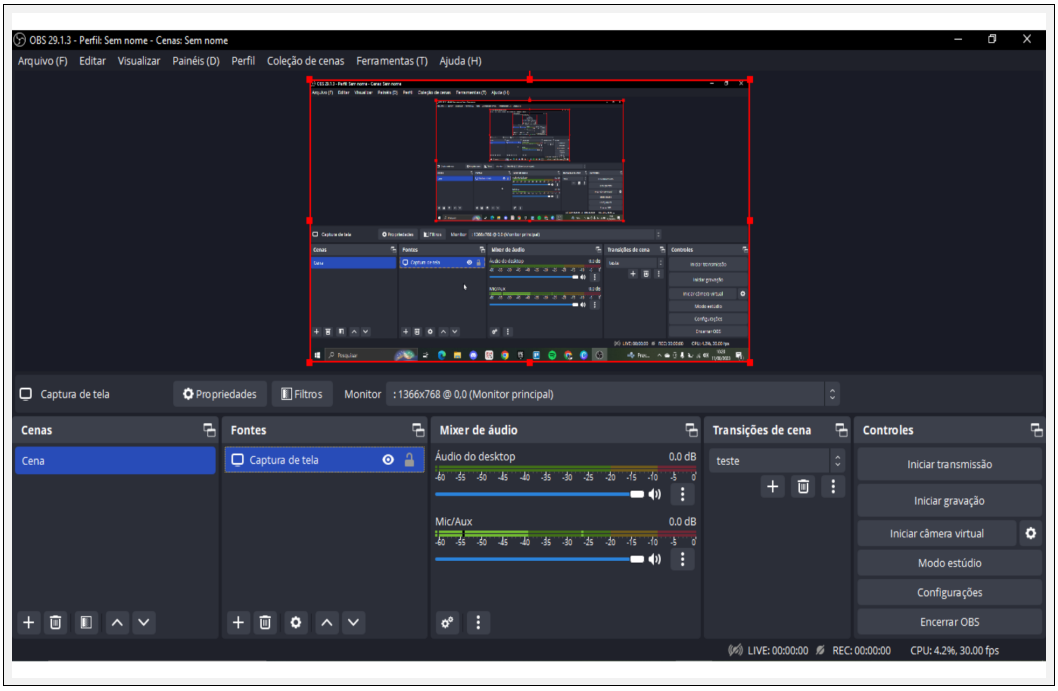


Figura 11.7: Visualização da tela principal.

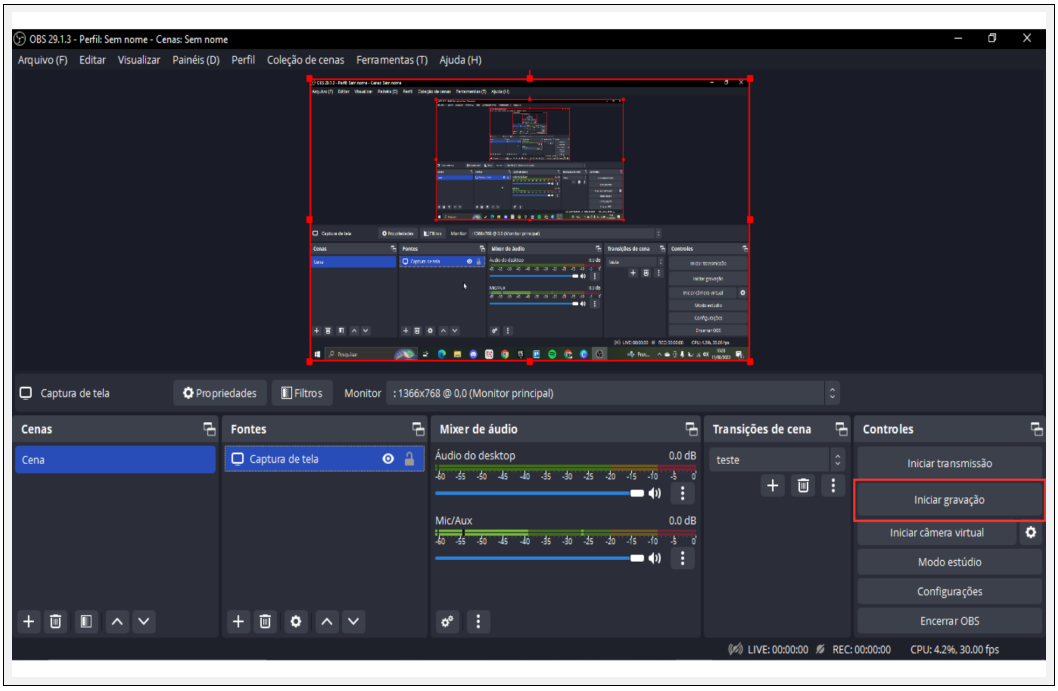


Figura 11.8: Clique em **“Iniciar gravação”**.

11.4 Adicionar uma *webcam* no OBS

Siga as instruções:

1. Adicionar uma *webcam*.
   - Fig. 11.9: Direcione seu mouse para a área Fontes. Na sequência, clique no botão direito, ou no botão **"+"**. Por fim, clique na opção **"Dispositivo de captura de vídeo"**.
   - Fig. 11.10: Nomeie sua captura de vídeo, de preferência como "Webcam" para facilitar o entendimento de qual fonte de captura está sendo trabalhada. Por fim, clique em **"OK"**.
   - Fig. 11.11: Posicione a Captura da webcam onde você preferir.
   - Fig. 11.12: alternativamente, há a opção de configurar a webcam em formatos variados. Como ilustrado no vídeo tutorial acessível via QR Code, é possível ajustar o formato da webcam para um design circular, por exemplo. O referido vídeo está no presente link presente link.

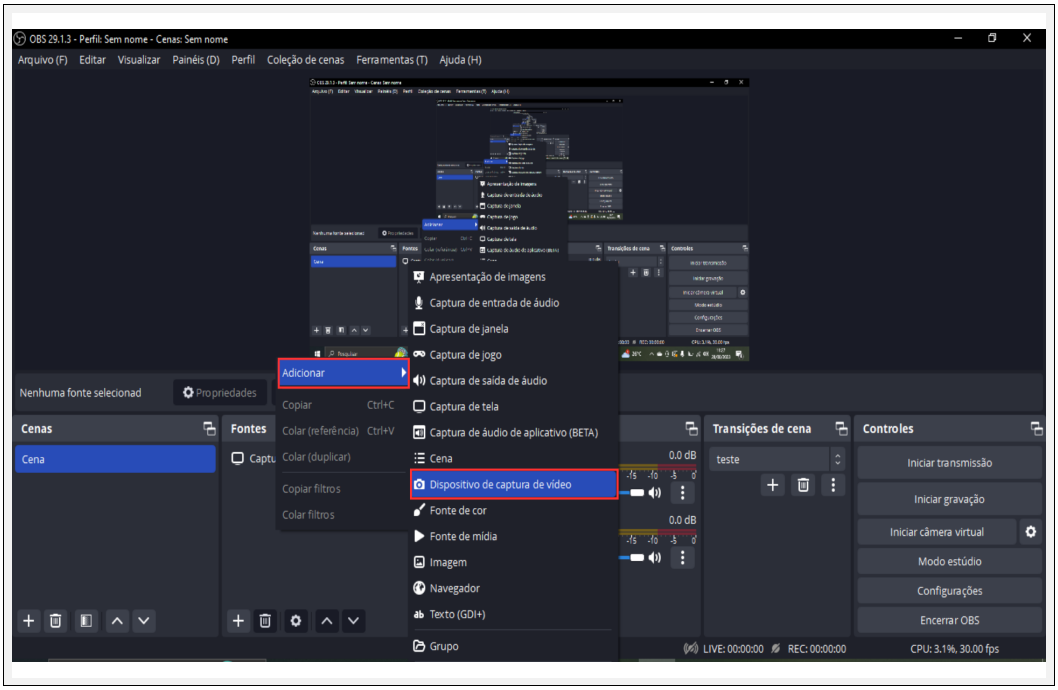


Figura 11.9: Clique em **"Iniciar gravação"**.

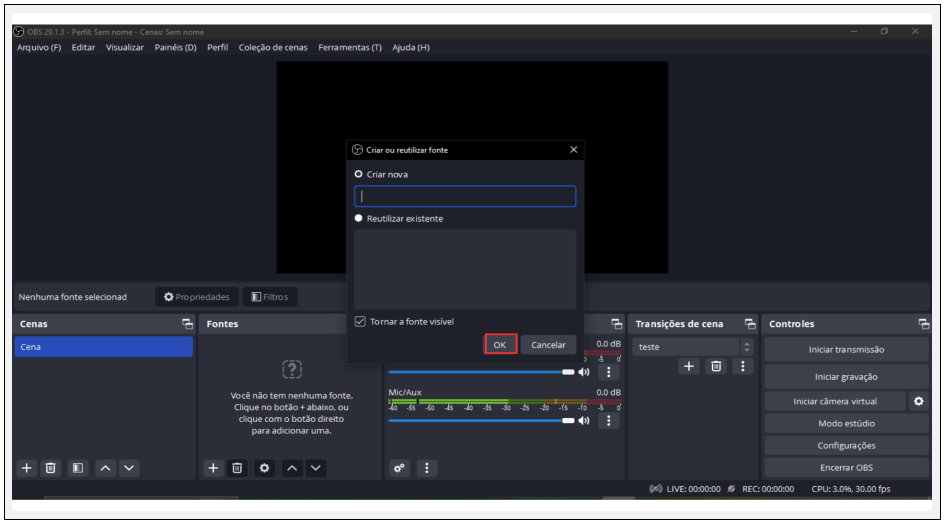


Figura 11.10: Clique em **"Iniciar gravação"**.

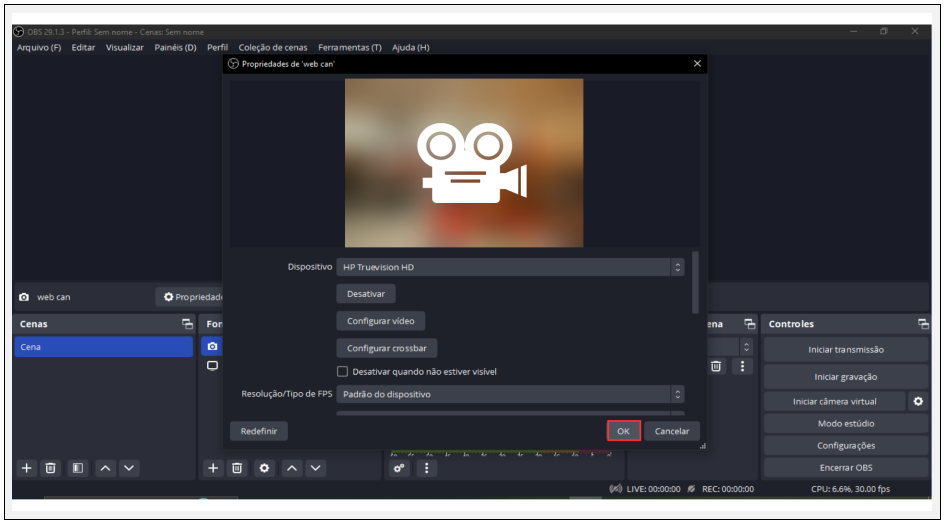


Figura 11.11: Clique em **"Iniciar gravação"**.

Figura 11.12: Alternativamente, há a opção de configurar a webcam em formatos variados. Como ilustrado no vídeo tutorial acessível via QR Code, é possível ajustar o formato da webcam para um design circular, por exemplo. O referido vídeo está no presente link presente link.

11.5 Como adicionar texto no vídeo no OBS

1. Como adicionar texto no vídeo.
   - Fig. 11.13: escolha a cena que que deseja para adicionar o texto. Em fontes, clique em **“adicionar”** depois clique em **“texto”**. Na sequência, nomeie a caixa de texto com desejar e clique em **“OK”**.
   - Fig. 11.14: adicione o texto desejado e configure a cor, a fonte e a opacidade ao seu gosto, depois clique novamente em **“OK”**.
   - Fig. 11.15: Posicione o texto no vídeo onde desejar.

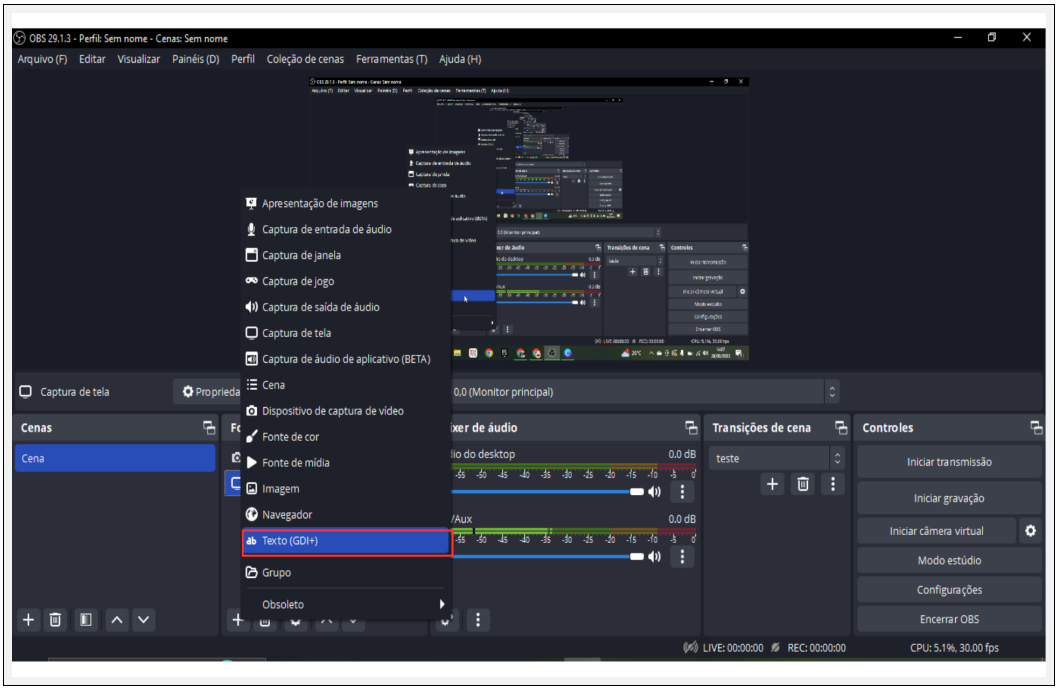


Figura 11.13: Escolha a cena que que deseja para adicionar o texto. Em fontes, clique em **“adicionar”** depois clique em **“texto”**. Na sequência, nomeie a caixa de texto com desejar e clique em **“OK”**.

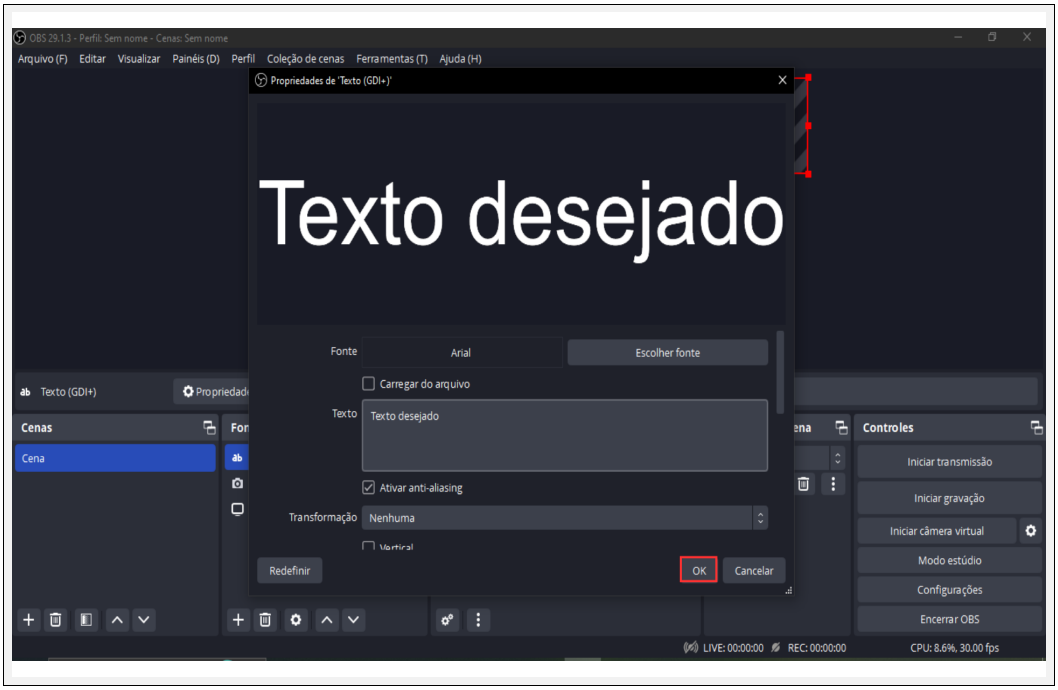


Figura 11.14: Adicione o texto desejado e configure a cor, a fonte e a opacidade ao seu gosto, depois clique novamente em **"OK"**.

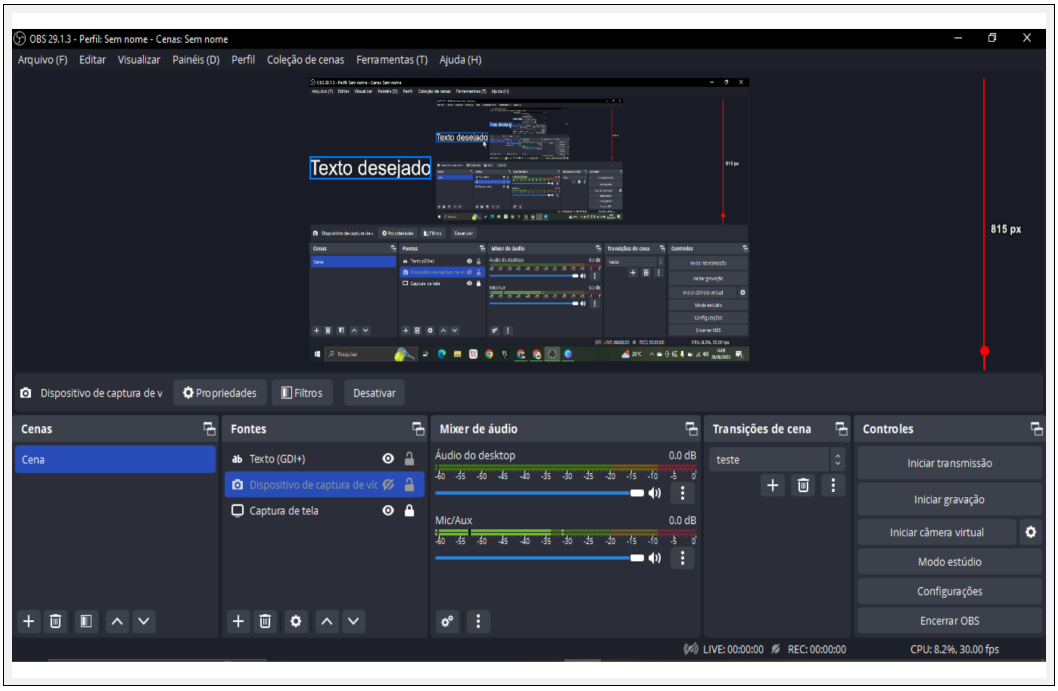


Figura 11.15: Posicione o texto no vídeo onde desejar.

11.6 Como adicionar texto no vídeo no OBS

1. Onde o arquivo das gravações fica localizado:
   - Fig. 11.16: clique em **“Configurações”**.
   - Fig. 11.17: clique em **“saída**. Lá pode ver onde o vídeo gravado está localizado.

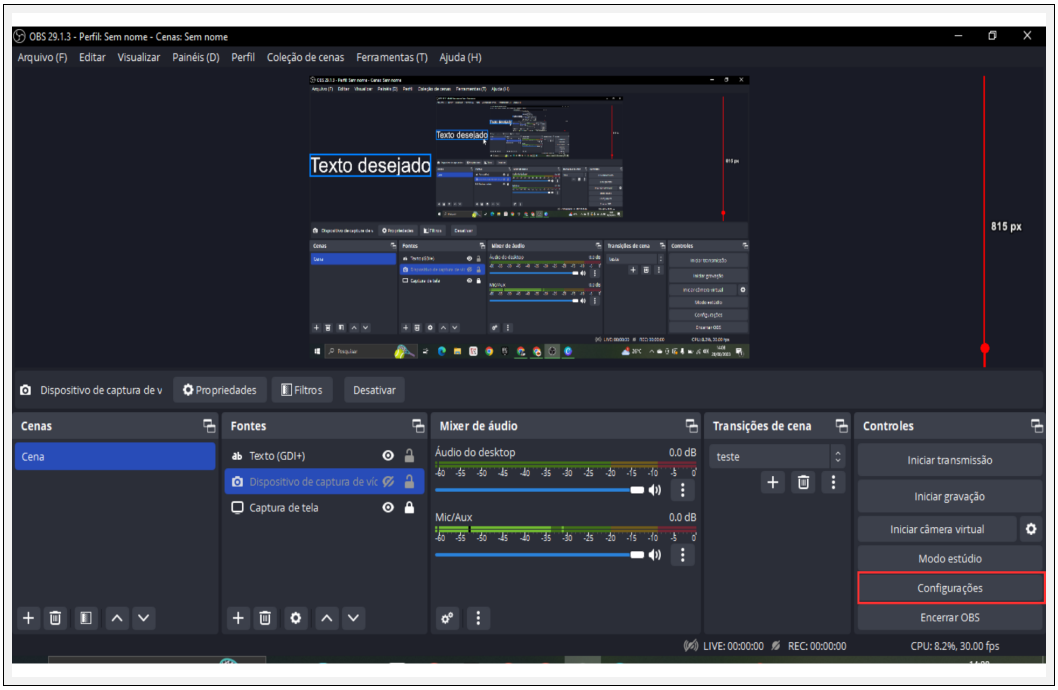


Figura 11.16: Clique em **“Configurações”**.

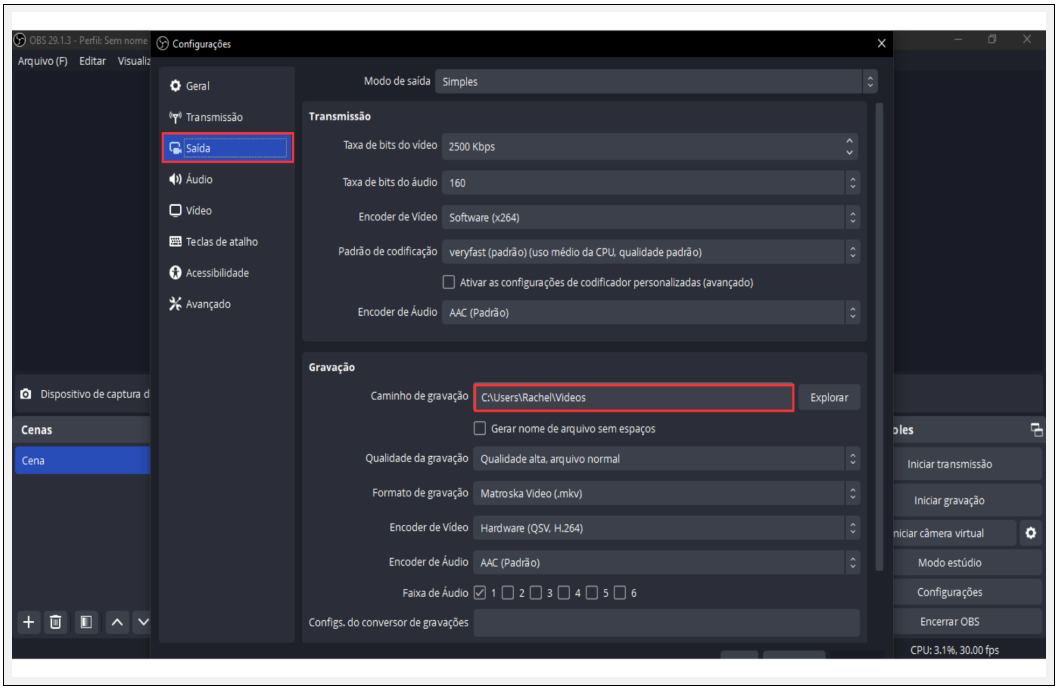


Figura 11.17: Clique em **"saída**. Lá pode ver onde o vídeo gravado está localizado.

12. Anexo: Criação de Portfólio

Vynícius Brendow Vicente Souza
Currículo *Lattes*: http://lattes.cnpq.br/9410683637463514
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. vynicius.brendow@ufpe.br

Dr. Sidney Marlon Lopes de Lima
Currículo *Lattes*: http://lattes.cnpq.br/0323190806293435
Departamento de Eletrônica e Sistemas - Universidade Federal de Pernambuco, Recife, Pernambuco. sidney.lima@ufpe.br

12.1 Importância de ter um Portfólio

Um portfólio digital é um instrumento essencial para quem busca se destacar no mercado de trabalho. Ele vai além do tradicional currículo em papel, oferecendo uma apresentação visual e interativa das habilidades, experiências e conquistas de um profissional. Esse recurso permite que os potenciais empregadores não apenas vejam, mas também experimentem as competências do candidato de maneira mais vívida e envolvente.

Ao criar um portfólio digital, é possível exibir projetos concluídos, fornecer exemplos tangíveis do trabalho realizado e até mesmo compartilhar depoimentos ou recomendações de colegas e entusiastas do seu ramo de habilidades. Essa abordagem prática e visual não só valida as habilidades declaradas, mas também proporciona uma compreensão mais profunda da aplicação prática dessas habilidades.

Além disso, um portfólio bem elaborado demonstra a capacidade do profissional de se adaptar às demandas digitais e tecnológicas da atualidade. Ter um currículo na plataforma Lattes. , mostrar presença em plataformas relevantes, como LinkedIn, e integrar-se ao Github para áreas técnicas, reflete a familiaridade com ambientes digitais e uma postura proativa na gestão da própria imagem profissional.

Ao apresentar de forma transparente a jornada profissional, incluindo desafios superados e projetos bem-sucedidos, um portfólio digital cria uma narrativa coesa e impactante. Isso não apenas

fornece um contexto mais amplo sobre o profissional, mas também destaca as características pessoais que vão além das habilidades técnicas, como resiliência, colaboração e criatividade.

Em resumo, um portfólio digital eficaz não é apenas um arquivo de documentos, mas uma ferramenta dinâmica que comunica a essência do profissional de maneira autêntica. Ele não só reflete habilidades e realizações, mas também constrói uma ponte visual e interativa entre o candidato e os empregadores em potencial, facilitando uma compreensão mais profunda e envolvente das contribuições que o profissional pode oferecer ao mercado de trabalho.

12.2 Criação de um e-mail no Gmail

Ter um e-mail no Gmail é fundamental ao procurar emprego, pois proporciona uma imagem profissional e confiável aos potenciais empregadores. Muitas empresas valorizam a familiaridade e a confiabilidade associadas à plataforma do Gmail, que é uma das mais amplamente utilizadas. Adicionalmente, o Gmail oferece um espaço de armazenamento generoso na nuvem, permitindo que os candidatos organizem e armazenem documentos importantes, como currículos e cartas de apresentação.

Siga as instruções:

1. Criação de um e-mail no Gmail.
   - Fig. 12.1: acesse o site do Gmail em www.gmail.com.
   - Fig. 12.2: clique no botão **"Criar conta"** ou **"Inscrever-se"**.
   - Fig. 12.3: preencha o formulário de inscrição com suas informações pessoais.
   - Fig. 12.4: escolha seu nome de usuário do Gmail.
   - Fig. 12.5: adicione um número de telefone válido.

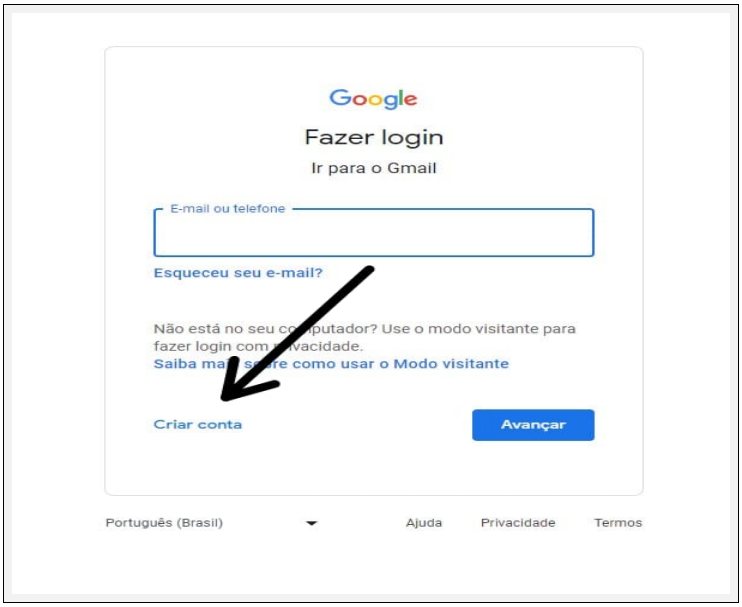


Figura 12.1: Acesse o site do Gmail em www.gmail.com.

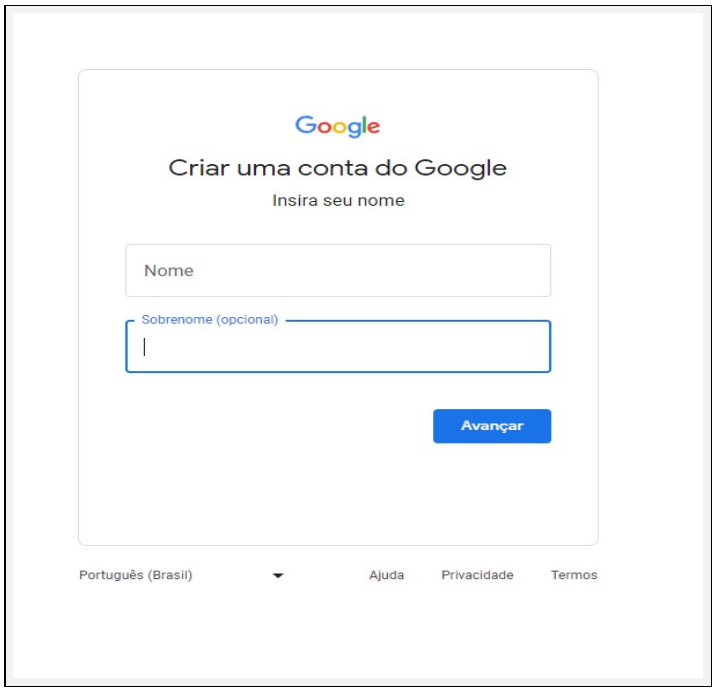


Figura 12.2: Clique no botão **"Criar conta"** ou **"Inscrever-se"**.

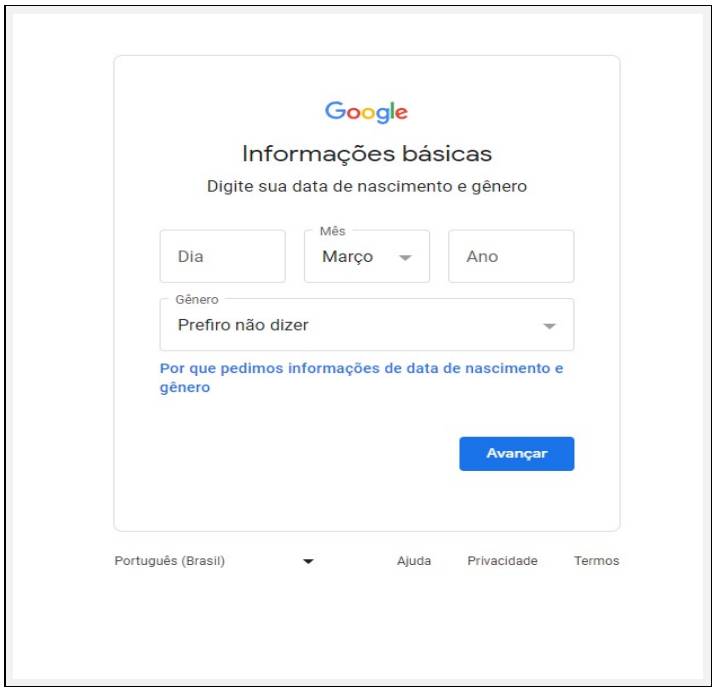


Figura 12.3: Preencha o formulário de inscrição com suas informações pessoais.

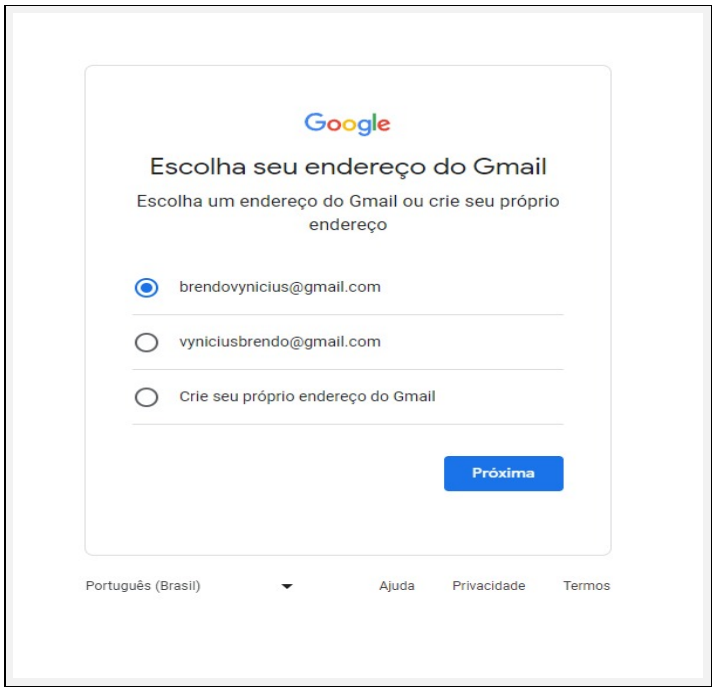


Figura 12.4: Escolha seu nome de usuário do Gmail.

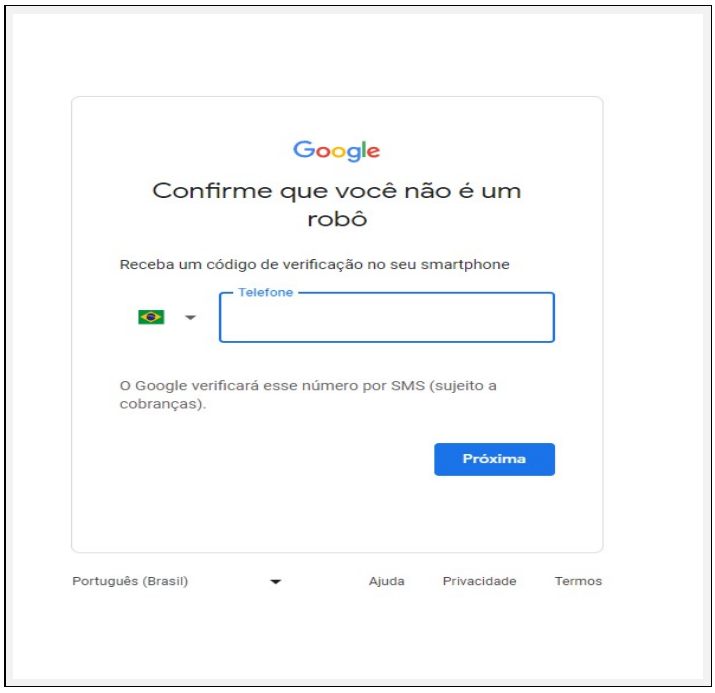


Figura 12.5: Adicione um número de telefone válido.

12.3 Criação de um canal no Youtube

O YouTube oferece uma plataforma de alcance global para compartilhar informações, demonstrar produtos, fornecer tutoriais, oferecer *insights* do setor e promover serviços. Criar vídeos tutoriais autorais permite que você demonstre suas próprias habilidades práticas. Isso valida seu conhecimento e também constrói sua credibilidade como especialista na área

Siga as instruções:

1. Criação de canal no YouTube.
   - Fig. 12.6: acesse o site do YouTube em `www.youtube.com`.
   - Fig. 12.7: no canto superior direito da página inicial do YouTube, clique no ícone da sua conta (geralmente uma miniatura do seu avatar).
   - Fig. 12.8: após criar o canal, você pode personalizá-lo.

Figura 12.6: Acesse o site do YouTube em `www.youtube.com`.

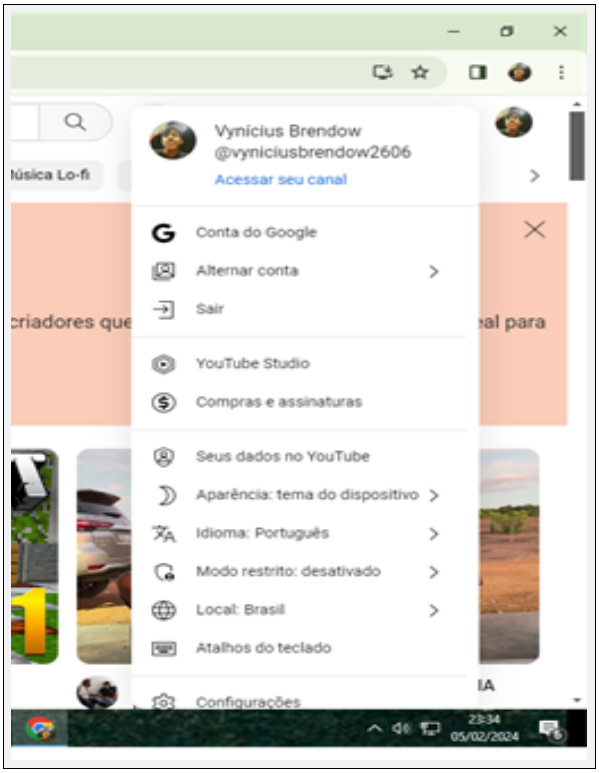


Figura 12.7: No canto superior direito da página inicial do YouTube, clique no ícone da sua conta (geralmente uma miniatura do seu avatar).

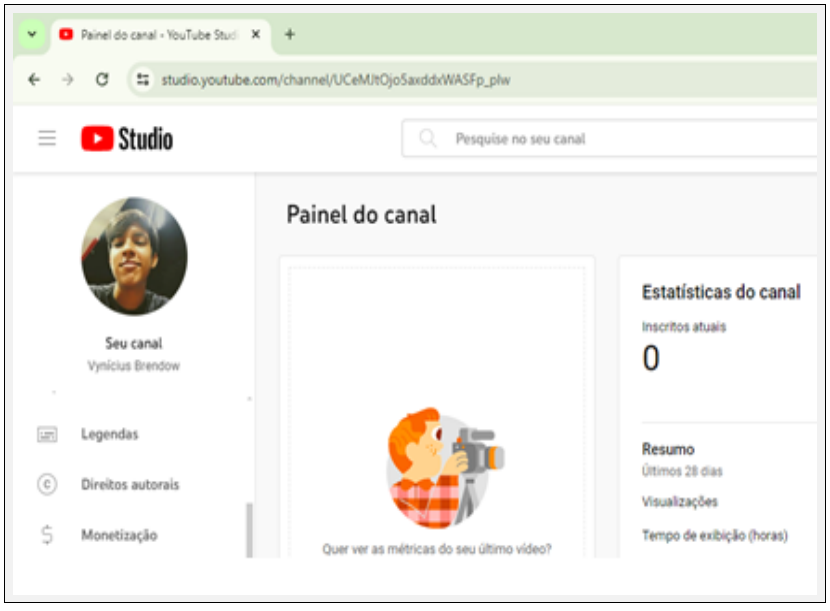


Figura 12.8: Após criar o canal, você pode personalizá-lo.

12.4 Criação de um canal no Telegram

No Telegram, recursos avançados como canais públicos e grupos privados permitem a disseminação de informações, discussões e colaboração em larga escala. Sua interface intuitiva e compatibilidade multiplataforma tornam o Telegram uma ferramenta versátil para profissionais que buscam uma comunicação rápida, confiável e segura em seus ambientes de trabalho.

Siga as instruções:

1. Criação de canal no Telegram.
   - Fig. 12.9: Toque no ícone de novo *chat* e selecione **"Novo canal"**.
   - Fig. 12.10: Toque em **"Criar Canal"**.
   - Fig. 12.11: Dê um nome ao seu canal e uma breve descrição.
   - Fig. 12.12: Escolha se o canal será **"Público"** ou **"Privado"**; no canal privado, novos inscritos só entram através de um link de convite.
   - Confirme e pronto! Agora, você tem um canal no Telegram. Basta carregar todos os seus vídeos autorais no seu canal!

Figura 12.9: Toque no ícone de novo chat e selecione "Novo canal".

Figura 12.10: Toque em "Criar Canal".

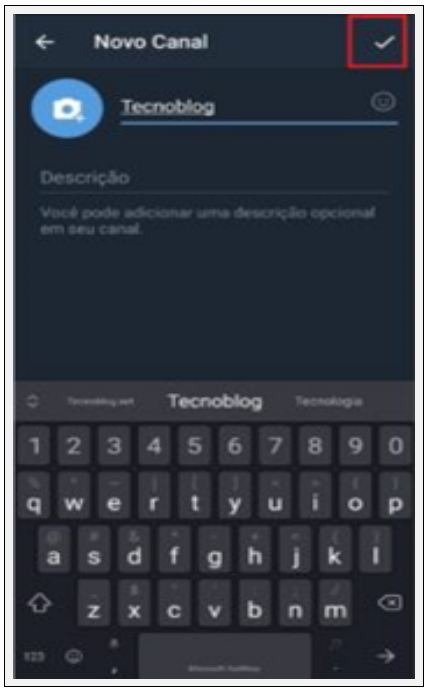


Figura 12.11: Dê um nome ao seu canal e uma breve descrição.

Figura 12.12: Escolha se o canal será "Público"ou "Privado"; no canal privado, novos inscritos só entram através de um link de convite.

12.5 Criação de um perfil no Linkedin

Ter um perfil no LinkedIn é fundamental para quem está em busca de emprego, pois a plataforma se tornou uma ferramenta essencial no mundo profissional. O LinkedIn funciona como uma espécie de currículo *online*, proporcionando aos recrutadores uma visão abrangente das habilidades, experiências e realizações de um candidato.

O LinkedIn serve como uma plataforma de pesquisa de empregos, permitindo que os usuários encontrem oportunidades relevantes com base em suas habilidades e preferências. Os recrutadores também utilizam a plataforma para procurar candidatos que se encaixem nos requisitos específicos de uma vaga, tornando o LinkedIn uma ferramenta eficaz para conectar talentos a oportunidades de emprego.

Siga as instruções:

1. Criação de um perfil no Linkedin.
   - Fig. 12.13: acesse o site do LinkedIn em `www.linkedin.com`.

Figura 12.13: Acesse o site do LinkedIn em `www.linkedin.com`.

12.6 Como criar um perfil no Github:

Ter um perfil no GitHub é fundamental para quem busca emprego na área de tecnologia e desenvolvimento. O GitHub é uma plataforma de hospedagem de código-fonte que permite aos desenvolvedores compartilhar, colaborar e contribuir para projetos de software. Ao manter um perfil ativo no GitHub, os profissionais demonstram suas habilidades práticas, experiência em programação e a capacidade de trabalhar em projetos colaborativos.

O GitHub também funciona como um portfólio dinâmico, permitindo que os desenvolvedores exibam projetos pessoais, soluções para problemas específicos e mostrem sua evolução ao longo do tempo. Isso é valioso para destacar a criatividade, a resolução de problemas e a experiência prática de um candidato, proporcionando uma compreensão mais profunda de suas habilidades técnicas.

Siga as instruções:

1. Criação de um e-mail no Github.
   - Fig. 12.14: acesse `www.github.com`, e clique no botão

     textit**"Sign up"** no canto superior direito da página inicial.
     preencha o formulário de inscrição com seu nome, endereço de e-mail e senha.

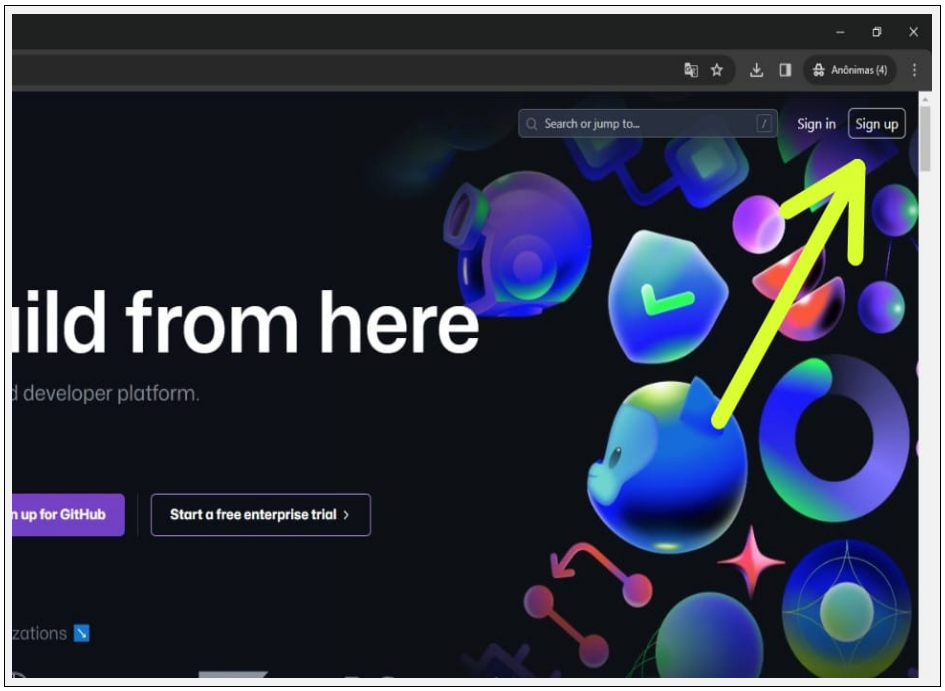


Figura 12.14: Acesse `www.github.com`, e clique no botão ***"Sign up"*** no canto superior direito da página inicial.

Figura 12.15: Preencha o formulário de inscrição com seu nome, endereço de e-mail e senha.

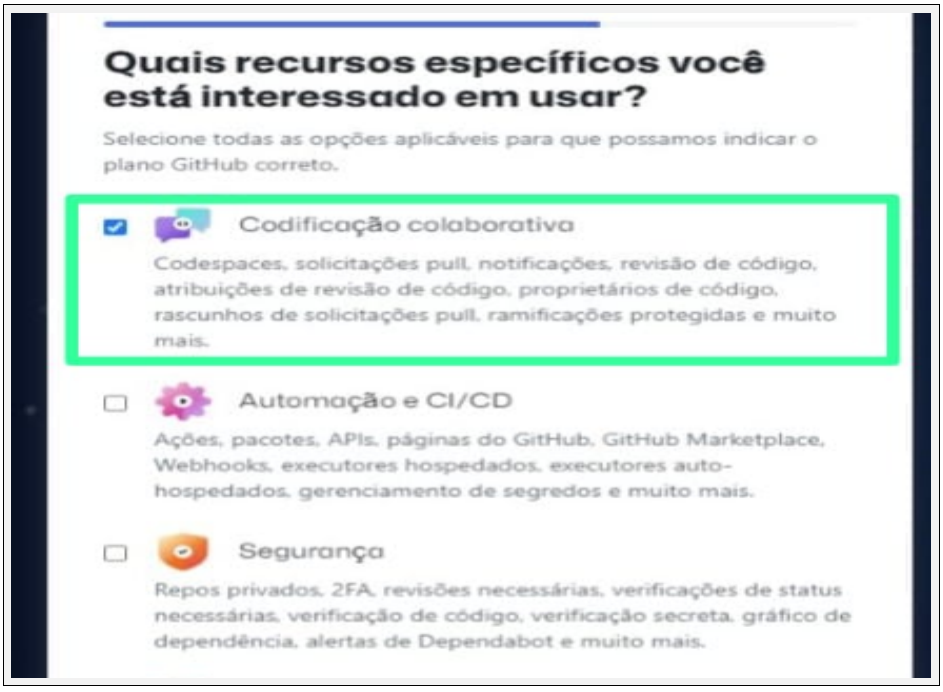


Figura 12.16: Depois de criar a conta, escolha a opção ***"Codificação colaborativa"***.

13. Anexo

13.1 Resposta dos códigos-fontes do Capítulo 9

```
#código para imprimir "Hello World" em Python
Hello World
```

```
#Exemplo 3
Imprimindo texto
Imprimindo número inteiro:  4
Imprimindo número real:  4.5
```

```
#Exemplo 4
3
Soma de 4 + 5 usando a função sum:  3
Soma de 4 + 5 + 6 é igual a: 15
```

```
#Exemplo 5
Informe o seu nome: 12
<class 'str'> 12
```

```
#Exemplo 6
Digite um número: 12
<class 'str'>
12
<class 'int'> 12
```

```
#Exemplo 7
Digite um número: 12
<class 'int'> 12
```

```
#Exemplo 8
Digite um número: 12
<class 'float'> 12.0
```

```
#Exemplo 9
Digite um número: w

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-18c5cd428f47> in <module>
      1 # O número agora será convertido antes mesmo de ser atribuído à variável.
----> 2 num = float(input('Digite um número: '))
      3 print(type(num), num)

ValueError: could not convert string to float: 'w'
```

```
#Exemplo 10
1 2 3 4 5 6 7 8 9 10
```

```
#Exemplo 12
<class 'int'>
<class 'float'>
<class 'complex'>
```

```
#Exemplo 13
Os alunos Lucas, Lima e Luiz tiraram as melhores notas da prova
de português.
```

```
#Exemplo 14
LucasSantos
SantosLucas
```

```
#Exemplo 15
True
False
```

13.2 Resposta dos códigos-fontes do Capítulo 10

```
#Exemplo 1
hellopython
hellohello
pythonpythonpython
```

```
#Exemplo 2
P Y T H O N
============
P Y T H O N
```

```
#Exemplo 3
        IndexError                                Traceback (most recent call last)
<ipython-input-6-69d8a15c0f7b> in <module>
----> 1 string[6]

IndexError: string index out of range
```

```
#Exemplo 4
        TypeError                                 Traceback (most recent call last)
<ipython-input-8-10f89dc5d206> in <module>
      1 string1 = 'python'
----> 2 string1[0] = 'l'

TypeError: 'str' object does not support item assignment
```

```
#Exemplo 5
uca
luc
lucas de souza
lucas de souza
lucas de souza
```

```
#Exemplo 6
False
True
True
False
False
True
True
False
```

```
#Exemplo 7
String sem espaço
```

```
#Exemplo 8
String Sem Espaço
```

```
#Exemplo 9
STRING SEM ESPAÇO
```

```
#Exemplo 10
string sem espaço
```

```
#Exemplo 11
3
```

```
#Exemplo 12
1
```

```
#Exemplo 13
2
```

```
#Exemplo 14
string com espaço na esquerda
```

```
#Exemplo 15
string com espaço na direita
```

```
#Exemplo 16
string com espaço na direita e esquerda
```

```
#Exemplo 17
0
```

```
#Exemplo 18
string com espaço
```

```
#Exemplo 19
{'nome': 'Maria', 'notas': [7, 7, 7, 7], 'media': 7}
```

```
#Exemplo 20
<class 'dict'> {}
<class 'dict'> {}
```

```
#Exemplo 21
{'nome': 'Fernando', 'notas': [1, 10, 5, 10]}
```

```
#Exemplo 22
nome =  João
idade =  10
```

```
#Exemplo 23
nome
idade
```

```
#Exemplo 24
Marcos
18
```

```
#Exemplo 25
{'nome': 'Marcos', 'idade': 18}
dict_items([('nome', 'Marcos'), ('idade', 18)])
dict_keys(['nome', 'idade'])
dict_values(['Marcos', 18])
{}
```

```
#Exemplo 26
nota: 3
nota: 9
nota: 5
nota: 8
nota: 5
A média do aluno é:  6.0
```

```
#Exemplo 27
['nome', 'notas', 'média', 'situação']
['Maria', '[4, 4, 4, 4]', '4', 'Final']
['Joana', '[5, 5, 5, 5]', '5', 'Final']
['Clara', '[7, 7, 7, 7]', '7', 'Aprovada']
```

```
#Exemplo 1
Informe a sua idade: 12
```

```
#Exemplo 2
Informe a sua idade: 16
Procure uma junta militar mais próxima da sua casa para fazer o seu alistamento.
```

```
#Exemplo 3
Informe a sua idade: 15
Você ainda não atingiu a idade mínima para o seu alistamento.
```

```
#Exemplo 4
Informe a sua idade: 16
Procure uma junta militar mais próxima da sua casa para fazer o
seu alistamento.
```

```
#Exemplo 5
Informe o peso: 82
Informe a altura: 1.60
IMC =  32.03124999999999
Você está acima do peso. Procure um médico
```

```
#Exemplo 6
Seja Bem-Vindo!
Seja Bem-Vindo!
Seja Bem-Vindo!
Seja Bem-Vindo!
Seja Bem-Vindo!
```

```
#Exemplo 7
1 2 3 4 5 6 7 8 9 10
```

```
#Exemplo 8
1 2 3 4 5 6 7 8 9 10
```

```
#Exemplo 9
<class 'list'> <class 'list'> <class 'list'> <class 'list'> <class 'list'>
['produto', 'preço', 'quantidade']
['maçã', 2.3, 4]
['uva', 0.24, 10]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
['P', 'y', 't', 'h', 'o', 'n']
```

```
#Exemplo 10
[1, 2, 3, 4, 5, 6]
['maria', 'joão', 'joaquim', 'marcos', 'josé', 'benjamin', 'luiz']
['maria', 'joão', 'marcos', 'josé', 'benjamin', 'luiz']
[1, 2, 3, 4, 5]
```

```
#Exemplo 11
ValueError                                Traceback (most recent call last)
<ipython-input-29-e3bc5a1f0e2e> in <module>
      1 # Tentando remover nome que não está na lista
----> 2 lista2.remove("lucas")

ValueError: list.remove(x): x not in list
```

```
#Exemplo 12
IndexError                                Traceback (most recent call last)
<ipython-input-30-45c7ed6a7c7f> in <module>
      1 # Tentando remover nome que não está na lista
      2 len(lista2)
----> 3 lista2.pop(len(lista2))

IndexError: pop index out of range
```

```
#Exemplo 13
<class 'tuple'> <class 'tuple'> <class 'tuple'>
```

```
#Exemplo 14
Imprimindo valores da tupla:
1
2
3
4
Imprimindo valores da lista:
1
2
3
4
```

```
#Exemplo 15
====================
python
====================
programador
====================
```

```
#Exemplo 16
====================
Python
====================
Programador
====================
====================
Engenharia
====================
UFPE
====================
```

```
#Exemplo 17
6
```

```
#Exemplo 18
54
```

```
#Exemplo 19
54
```

```
#Exemplo 20
1 3 28
```

```
#Exemplo 21
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-3-d2f5cd0f26c3> in <module>
      1 # Iremos tentar dividir um número por zero
----> 2 print(2 / 0)

ZeroDivisionError: division by zero
```

```
#Exemplo 22
Divisão por zero!
```

```
#Exemplo 23
Digite um número: 4
Digite outro numero: 2
bloco else ativado, valor da divisão:
```

```
2.0

Digite um número: 4
Digite outro numero: 0
bloco except ativado, erro:
division by zero
```

```
#Exemplo 25
0- texto 0
1- texto 1
2- texto 2
['0- texto 0\n', '1- texto 1\n', '2- texto 2\n']
```

Bibliografia

[1] F. ALVAREDO and L. GASPARINI. Handbook of income distribution. chapter 9 – recent trends in inequality and poverty in developing countries. *Elsevier Publishing Solutions. Vol..2, pp.697-805*, 2015.

[2] W. W. *et al.* AZEVEDO. Fuzzy morphological extreme learning machines to detect and classify masses in mammograms. *In: 2015 IEEE International Conference on Fuzzy Systems (FUZZIEEE), Istanbul.*, DOI: `https://doi.org/10.1109/FUZZ-IEEE.2015.7337975` 2015.

[3] W. W. *et al.* AZEVEDO. Morphological extreme learning machines applied to detect and classify masses in mammograms. *In: 2015 International Joint Conference on Neural Networks (IJCNN), Killarney.*, DOI: `https://doi.org/10.1109/IJCNN.2015.7280774` 2015.

[4] W. W. *et al.* AZEVEDO. Morphological extreme learning machines applied to the detection and classification of mammary lesions. *In: Tapan K Gandhi; Siddhartha Bhattacharyya; Sourav De; Debanjan Konar; Sandip Dey. (Org.). Advanced Machine Vision Paradigms for Medical Image Analysis. 1ed.Londres: Elsevier Science.*, pages 1–30, DOI: `https://doi.org/10.1016/B978-0-12-819295-5.00003-2`. 2020.

[5] L. BA and R. CAURANA. Do deep nets really need to be deep? *Advances in Neural Information Processing Systems*, pages 2654–2662, 2014.

[6] V. A. F. BARBOSA, J. C. GOMES, M. A. SANTANA, W. P. SANTOS, and *et al.* Heg.ia: an intelligent system to support diagnosis of covid-19 based on blood tests. *Research on Biomedical Engineering, v. 36, p. s42600-020-0011*, 2021.

[7] François CHOLLET. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, DOI: `https://10.1109/CVPR.2017.195` 2017.

[8] OMS. Organização Mundial da Saúde. Improving the quality and use of birth, death and cause-of-death information: guidance for a standards-based review of country practices. *ISBN: 9789241547970*, 2010.

[9] C. C. DA SILVA, C. L. DE LIMA, W. P. SANTOS, and *et al.* Covid-19 dynamic monitoring and real-time spatio-temporal forecasting. *Frontiers in Public Health, v. 9, p. 1-17, 2021*, 2021.

[10] IBGE : Instituto Brasileiro de Geografia e Estatística. Domicílios particulares ocupados em aglomerados subnormais e média de moradores em domicílios particulares ocupados em aglomerados subnormais. , Disponível em `https://sidra.ibge.gov.br/tabela/3380`. Acesso em fevereiro de 2023 2023.

[11] FAO (Food and Agriculture Organization of the United Nations Organização das Nações Unidas para Alimentação e Agricultura). Developing country regions. , Disponível em: `http://www.fao.org/docrep/003/t0800e/t0800e07.htm`. Acesso em fevereiro de 2023 2023.

[12] E. B. FRANÇA and *et. al.* Investigation of ill-defined causes of death: assessment of a program's performance in a state from the northeastern region of brazil. *Revista Brasileira de Epidemiologia. 17 (1)*, 2014.

[13] G. B. *et al.* HUANG. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(2):513–519, 2012.

[14] INTEL. Mcafee labs. , Disponível em: `https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-mar-2018.pdf`. Acesso em dezembro de 2021. 2018.

[15] INTEL. Mcafee labs: Threat report. , Disponível em: `https://secure.mcafee.com/us/resources/reports/rp-quarterly-threats-mar-2017.pdf?cid=BHP-075`. Acesso em 20 janeiro 2022. 2022.

[16] M. KREMER and R. GLENNERSTER. Handbook of health economics. capítulo 4 – improving health in developing countries: Evidence from randomized evaluations. *Editora Elsevier Publishing Solutions. Vol. 2, pp. 201-315*, 2011.

[17] S. M. L. LIMA, H. K. L. SILVA, J. H. S LUZ, and et al. Antivírus dotado de máquina morfológica de aprendizado extremo. *Revista de Sistemas de Informação da FSMA n 26 (2020) pp. 31-50*, 2020.

[18] S.M.L. LIMA, S.H.M.T. SILVA, PINHEIRO, and R.P.and et. al. Next-generation antivirus endowed with web-server sandbox applied to audit fileless attack. *Soft Computing (2022)*, DOI: `https://doi.org/10.1007/s00500-022-07447-4` 2022.

[19] S.M.L. LIMA, S.H.M.T. SILVA, R.P. PINHEIRO, and et al. Next-generation antivirus endowed with web-server sandbox applied to audit fileless attack. *Soft Computing (2022)*, doi:`https://doi.org/10.1007/s00500-022-07447-4` 2022.

[20] S.M.L. LIMA, A. G. SILVA-FILHO, and W. P. SANTOS. Detection and classification of masses in mammographic images in a multi-kernel approach. *Computer Methods and Programs in Biomedicine*, 134:11–29, DOI: `https://doi.org/10.1016/j.cmpb.2016.04.029`. 2016.

[21] American College of Radiology. Bi-rads: Breast imaging reporting and data system - sistema de diagnóstico e relatório para imagens de mama. , Quarta edição 2003.

[22] D. PAPALIA, S. OLDS, and R. FELDMAN. *Human Development*. The McGraw-Hill Companies, 11ª edição, pág. 162, 2008.

[23] D. A. PATTERSON and J. L. HENNESSY. *Computer Organization and Design. The Hardware/Software Interface.* Fifth edition, Morgan Kaufmann, 2017.

[24] E. T. PAULINO. The agricultural, environmental and socio-political repercussions of brazil's land governance system. *Land Use Policy. Vol. 36, pp. 134-144*, 2015.

[25] J. M. S. *et al.* PEREIRA. Method for classification of breast lesions in thermographic images using elm classifiers. in: Wellington pinheiro dos santos; maíra araújo de santana; washington wagner azevedo da silva. (org.). understanding a cancer diagnosis. *York: Nova Science*, pages 117–132, Disponível em: https://novapublishers.com/shop/understanding-a-cancer-diagnosis/. 2020.

[26] R.P. PINHEIRO, S.M.L. LIMA, D.M. SOUZA, and et al. Antivirus applied to jar malware detection based on runtime behaviors. *Scientific Reports, 12, 1945 (2022)*, doi: https://doi.org/10.1038/s41598-022-05921-5 2022.

[27] B. P. REYDON, V. B. FERNANDES, and T. S. TELLES. Land tenure in brazil: The question of regulation and governance. *Land Use Policy. Vol. 42, pp. 509-516*, 2015.

[28] M. M. SANTOS, A. G. SILVA FILHO, and W. P. SANTOS. Deep convolutional extreme learning machines: Filters combination and error model validation. *NEUROCOMPUTING*, 329:359–369, DOI: https://doi.org/10.1016/j.neucom.2018.10.063 2019.

[29] S. SHALEV-SHWARTZ and J. SHAI BEN-DAVID. *Understanding Machine Learning: From Theory To Algorithms*. Cambridge University Press. ISBN 978-1-107-05713-5 Hardback, 2014.

[30] RAMI SIHWAIL, KHAIRUDDIN OMAR, and K. A. Z. ARIFFIN. A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *International Journal on Advanced Science, Engineering and Information Technology*, (1662):4–2, 8 2018.

[31] S. TAHERI, M. SALEM, and J-S. YUAN. Leveraging image representation of network traffic data and transfer learning in botnet detection. *Big Data and Cognitive Computing. 2018; 2(4):37*, DOI: https://doi.org/10.3390/bdcc2040037 2018.

[32] R. D. TOSCANO, S.M.L. LIMA, and *et al.* Análise de vantajosidade em modelos de previsão em séries temporais. tecnologias, métodos e teorias na engenharia de computação. *2. 1ed.: Atena Editora, 2021, v. , p. 13-23.*, DOI: http://dx.doi.org/10.22533/at.ed.4552116042 2021.

[33] A. M. TURING. Computing machinery and intelligence, mind. *Issue 236, October 1950, Pages 433–460*, LIX, 1950.